Coventry University

DOCTOR OF PHILOSOPHY

Acoustic localisation for real-life applications of wireless sensor networks

Allen, Michael

Award date:
2009

Awarding institution:
Coventry University

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of this thesis for personal non-commercial research or study
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission from the copyright holder(s)
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 04. Jul. 2025

Acoustic Localisation for Real-Life Applications of Wireless Sensor Networks

M.P. Allen

A thesis submitted in partial fulfilment
of the University's requirements for the Degree of
Doctor of Philosophy

2009

Coventry University
Faculty of Engineering and Computing

Dedicated to Stephanie
for travelling the world to be with me

Abstract

The work described in this thesis is concerned with *self-localisation* (automated estimation of sensor locations) and *source-localisation* (location of a target) using Wireless Sensor Networks (WSNs). The motivation for the research in this thesis is the on-line localisation of marmots from their alarm calls. The application requires accurate 3D self-localisation (within a small percentage of sensor spacing) as well as timely operation. Further challenges are added by the high data-rate involved: sensor nodes acquire data at a rate that is greater than the available network bandwidth. This data cannot be streamed over a multi-hop network, implying a need for data reduction through in-network event detection and local data compression or filtering techniques.

The research approach adopted in this thesis combined simulation, emulation and real-life experimentation. Real-life deployment and experimentation highlighted problems that could not be predicted in controlled experiments or simulation. Emulation used data gathered from controlled, real-life experimentation to simulate proposed system refinements; this was sufficient to provide a proof-of-concept validation for some of the concepts developed. Simulation allowed the understanding of underlying theoretical behaviour without involving the complex environmental effects caused by real-life experimentation.

This thesis details contributions in two distinct aspects of localisation: acoustic ranging and end-to-end deployable acoustic source localisation systems. With regard to acoustic ranging and 3D localisation, two WSN platforms were evaluated: one commercially available, but heavily constrained (Mica2) and one custom-built for accurate localisation (Embedded Networked Sensing Box (ENSBox)). A new proof of concept platform for acoustic sensing (based on the Gumstix single-board computer) was developed by the author (including the implementation of a ranging mechanism), based on experiences with the platforms above. Furthermore, the literature was found to lack a specific procedure for evaluation and comparison of self-localisation algorithms from theoretical conception to real-life testing. Therefore, an evaluation cycle for self-localisation algorithms that encompassed simulation, emulation and real-life deployment was developed.

With respect to source localisation, a hardware and software platform named *VoxNet* was designed and implemented. This was the result of a collaboration between the author and researchers at UCLA and MIT, as part of the first iteration of the National Science Foundation-funded VoxNet project. The author contributed to the software development and in-situ deployment of VoxNet, as well as leading the experimentation and data analysis resulting from the first deployment.

Following in-situ deployment experiences, two strategies to improve end-to-end system timeliness were developed by the author: The first, **Lazy Grouping**, was a centralised algorithm that performed on-line grouping of event data and facilitated its collection from the network. The second strategy led to the development of **Adaptation policies**, allowing nodes in the network to evaluate whether to process data locally based on previous data transfers. Through simulation based on realistic data traces, Lazy Grouping was observed to reduce end-to-end system latency significantly by reducing the amount of data that nodes send (whilst maintaining all other functional characteristics). The Adaptation policies

implemented delivered correct choices on whether to process locally 70–97% of the time in scenarios derived from in-situ data traces, hence improving the system timeliness by choosing the fastest processing path.

This research furthers the state of the art with respect to high data-rate sensing systems, and the evaluation of real-life Wireless sensor network (WSN) systems, both important aspects to the development of this field of research.

Acknowledgements

The person to whom I am most indebted as a researcher is my Director of Studies—Dr. Elena Gaura, director of the Cogent Computing Applied Research Centre. Her unwavering confidence in my ability as a researcher gave me great motivation to succeed, and I have been inspired by her drive, determination and tireless work ethic. It has been reassuring for me to know that no matter how confused I got with the minutiae of my current research, Elena has always had the bigger picture in mind. At points when I felt disillusioned by my achievements, Elena could quickly put everything into sharp focus and instill new confidence.

Throughout my time with Cogent, Elena has provided many fantastic research opportunities. Thanks to her, I was fortunate to spend a year of my PhD studies at the Centre for Embedded Network Sensing (CENS) at the University of California, Los Angeles. During this time I was advised and mentored by Professor Deborah Estrin and Dr Lewis Girod, whose support, advice and insightful comments shaped a large amount of the work I did whilst at CENS. Much of the practical work in this thesis is taken from experimentation I performed at UCLA whilst collaborating on the ongoing acoustic projects there, and VoxNet in particular.

I made many friends and acquaintances at UCLA, whose expert knowledge and willingness to share it helped me learn more in a few short months than I thought possible. In particular, Tom Schoellhammer, Martin Lukac and Travis Collier were tremendously supportive throughout my time at CENS, always happy to put up with my interferences and advise me on the intricacies of statistics, marmots and EmStar. Tom, Martin and Travis all cheerfully helped me run experiments and deployments at ridiculous hours in the morning, all in the name of science, and for that I am eternally grateful.

I would also like to thank Ryan Newton at CSAIL, MIT for his support whilst we were working on VoxNet, and Andreas Ali at EE, UCLA for his advice and support with the Approximated Maximum Likelihood algorithm and the first iteration of the acoustic source localisation system.

Back in Coventry, I would like to thank Dr James Brusey, who joined my advisory team late on but still managed to provide a great deal of useful advice and support, particularly in preparing the last stages of this thesis. His rigorous comments have improved this work greatly.

There are many other colleagues and acquaintances too numerous to list who have enriched my time as a graduate student, but particularly the staff and PhD students of the Cogent Computing Applied Research Centre. I have taken great pleasure in being a part of Cogent, and am immensely proud to be the first PhD student to submit!

I would also like to thank my parents, Paul and Diane, for not only showing unwavering support throughout my PhD research, but encouraging me in every venture I have undertaken. Your faith in me provided more motivation than you will ever know. Finally, I would like to extend my gratitude to Stephanie, who will finally get to spend some time with the man she married all those months ago! Stephanie has provided boundless love, emotional support and consideration throughout my PhD research, and now I can repay her patience and self-sacrifice as we begin our journey together.

Nomenclature

δ	the uncertainty factor for grouping detections
$\ell(d)$	the time taken to transfer a raw detection from node to sink
$\hat{\ell}_1(d)$	goodput based estimator to predict $\ell(d)$
$\hat{\ell}_2(d)$	pseudo-goodput based estimator to predict $\ell(d)$
$\hat{t}$	ToF estimate
$\hat{x}$	estimated distance between nodes
ν	number of dimensions
ϕ	zenith (degrees)
τ	empirically observed ETX threshold for adaptation
θ	azimuth (degrees)
a	Number of anchors in a network
B	local buffer that a node maintains of raw detections d_{raw}
C	largest difference in a given axis between all nodes in a network
c	list of all X, Y, Z coordinates of nodes in a network
$c(E)$	the current ETX value for a node
C_r	the scaled resolution of the localisation search space
d	event detection triggered by a node
d_{det}	global network time detection was triggered (part of d)
D_i	the amount of data sent (number of hops $\cdot$ detection size)
d_{ns} or d_i	unique global identifier of a detection
d_{raw}	raw data of detection (part of d)
f	frequency (Hz)
F_{max}	highest frequency that can be detected without spatial aliasing (Hz)
f_s	sampling frequency

G	list of all groups of detection events, maintained by the sink
g	a group within G
g_{curr}	the most recent group created by the sink
h	number of hops a node is from the sink
$I(d)$	the node id related to a detection d_i
$I(g)$	the id related to a given group created at the sink
J	AML result vector
K	distance between speaker and microphone
l	the watchdog timeout value
$l(g_i)$	the latency to gather data from all nodes in a group
$m(g)$	the mean timestamp of all detections currently in the group
$M_{i,err}$	accuracy of latency estimates made by $\hat{\ell}(d)$
n	number of nodes in the network
$n(G)$	the number of groups in G
$n(g)$	the number of detections currently in group g
$n(q_{aml})$	the number of raw detections in the local AML processing queue
$n(q_{send})$	the number of raw detections in the sending queue
q_i	time a detection spends on a node's message queue
R	reply message sent to the sink in response to a data request
r_{ij}	range estimate made by node i to node j
S	Samples
s	sequence number for detection
$s(d_{raw})$	size of raw detection data, in KB
t	time
$T(d)$	the detection timestamp related to a detection d_i
t_A	time between sending and receiving ranging signals for node A

t_B	time between sending and receiving ranging signals for node B
t_{aml}	empircally observed time to process the AML algorithm locally
t_i	time taken to transfer a raw detection message from node to sink
$t_{q,a}$	arrival time of timesync query packet
$t_{q,s}$	sending time of timesync query packet
$t_{r,a}$	arrival time of timesync reply packet
$t_{r,s}$	sending time of timesync arrival packet
u	uncertainty value
v	speed
v_s	speed of sound
W	Wire transmission time between gateway and control console
w_g	the watchdog timer associated with a given group g
x	distance

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless Sensor Networks (WSN)s are composed of embedded devices called *sensor nodes* that are equipped with wireless radios and one or more sensors.

Sensor nodes can be deployed—that is, placed over some physical area where they can automatically acquire data from their sensors about a given physical phenomena (for example, temperature or humidity). A sensor node can process this data and wirelessly transmit it to a central processing point for further analysis. Typically, sensor nodes are physically small and are *resource-constrained* by being battery powered, having only a small amount of memory and limited local processing capability.

The promise of using WSNs for sensing lies in the deployment of sensor nodes in large numbers over an environment, particularly in places which are difficult to reach with larger sensing devices that require connection to a mains power supply. Furthermore, providing a larger number of measurement points over a space enables the analysis of physical phenomena at high spatial and temporal density.

There are many applications which can make use of the spatially distributed nature of WSNs: in structural health monitoring, vibration data can be gathered from many points in a building to determine potential structural faults. In agricultural monitoring, temperature and soil moisture data can be continuously gathered by nodes over a field of crops to determine areas which are too dry or moist, too hot or cold.

An important notion for both of these applications and many others is *location*. When physical measurements are made by many nodes over an area, it is of interest to understand how these values change not only over time (at each node), but also space (across the whole network). This might take the form of a *time-lapse* 2D or 3D visualisation of the area that the nodes are deployed in, where the different sensed values can be displayed to help understand the spatial and variation over time.

To enable the spatial analysis of the data gathered by a WSN, it is necessary to determine the physical locations of each node in the network. The procedure of determining physical position is called *localisation*. Position could be assigned manually by taking Global Positioning System (GPS) measurements, or relative distance/angle measurements between sensor nodes. However, this process is time consuming, and prone to error and inaccuracy (based on the accuracy of the measurement tool and the skill of the person operating it). Automating the localisation process within the WSN should reduce deployment time and eliminate human-related error.

There are two types of automated localisation that are relevant for WSNs: *self-localisation*, where nodes in the network collaborate to estimate their relative positions, and *source-localisation*, where nodes in the network detect and estimate the location of some other target, such as an animal or a vehicle.

This thesis is concerned with both self- and source-localisation. The motivating application studied

in this thesis is the distributed, on-line localisation of an animal in its natural habitat. This application motivates the need for automated self-localisation as it requires that sensor node positions are determined accurately in 3D. Any error in node position will translate to error in the position estimate of the animal.

1.1 Research justification

The dominant theme that runs through all of the work in this thesis is that of *real-life applications*. It is the author's strongly held opinion that the development of a sensor network must be motivated by an application that provides realistic requirements. This thesis is focused on the design, implementation and evaluation of WSN systems that can be used for real applications. In particular, this thesis is focused on WSN systems which support *high data-rate* applications, where data is being sampled by sensor nodes at 100 Hertz (Hz) or more, necessitating in-network processing. As noted in the previous section, the motivating application considered in this thesis is *acoustic localisation*. Localisation is a strong example of a compelling and challenging WSN application and acoustics are a good example of a high-data rate phenomena. These two aspects (source-localisation and high data-rate systems) are discussed in more detail in the rest of this section.

1.1.1 Source-localisation

In general, the goal of source localisation is to estimate the location of an unknown target, based on characteristic signals that it produces (either an acoustic or electromagnetic signal).

The specific source localisation application in this thesis is acoustic source localisation of marmots (a small rodent) in their natural habitat. This application is useful for biologists studying bio-acoustic data, where animals' calls are used to study their behaviour. Using a WSN-based source localisation system is desirable because of the increased density of measurements, and the potential to automate the localisation process so that it can be performed in-situ, in real-time. When performed in real-time (or *on-line*), WSN-based marmot position estimates can be used to help the user take photographs of the calling animals to augment observations, either in a manual (the scientist takes the photo) or automated fashion (cameras are actuated to the estimated position). If the WSN can perform some form of data processing within the network, then the time taken to determine position estimates could be reduced (compared to performing the localisation at a central point). Furthermore, marmot localisation is the first stage in enabling automated census of marmots, where individuals can be identified, classified and counted. Automation of this process is an even more complex, but potentially enabling research tool for scientists in the field.

1.1.2 Acoustics in high data-rate systems

High data rate sensing applications, particularly those associated with acoustics are compelling for WSN research because sensor data is being generated at such high rates that it cannot be streamed to a central point for further processing. Therefore, some of the processing for the application *must* be carried out by the sensor nodes. This represents a smarter use of sensing than is commonly seen in current real-life WSN research, where data is typically sensed within the network at low sample rates (less than 1 Hz) and forwarded through the network to a central point, where it is processed.

Acoustics in the higher part of the audible frequency spectrum (which ranges from 20 Hz to 20 Kilo-

hertz (kHz)) require data sampling on the order of kHz, representing a particularly high data-rate phenomena. Animal calls, and marmot calls in particular are good examples of this type of acoustic phenomena.

On-node processing and distributed collaboration are two desirable features of WSNs in theoretical research and the motivating application provides a compelling reason for these techniques to be implemented in a real WSN.

1.2 Approach to research

This thesis takes the view that because WSNs are intended to be deployed in real physical environments to monitor real phenomena, they must be deployed and tested in-situ.

Simulations of WSN systems must model the effects of physical phenomena, either the environment or the phenomena of interest. For acoustics and radio signals, it is particularly difficult to model behaviour purely in simulation. Whilst this may be adequate to verify theoretical algorithmic performance, it only represents a partial validation of a system because it is practically impossible to predict all possible failure conditions for a system without testing it in a real environment. For example, several important observations leading to essential system improvements are made in Chapter 5, as an outcome of in-situ deployment experience and close familiarity with the specific application. These observations and their effects on the system's operation would have been difficult or impossible to imagine in the context of purely simulated experiments.

Consequently, the findings in this thesis are the result of several experimental approaches: in-situ operation and experimentation, controlled indoor and outdoor experimentation, and simulations using real data traces. This approach to research provides a clear and complete understanding of the crucial issues raised by designing, building, deploying, and using WSNs.

In Chapter 3, controlled, indoor and outdoor experimentation was carried out with respect to several ranging mechanisms and localisation algorithms. This was necessary to understand the relative performance of both ranging mechanisms and localisation algorithms in realistic environments. In addition, simulation of sensor localisation was performed to understand and isolate the effects of varying ranging precision and Geometric Dilution of Precision (GDoP). Simulation was sufficient to provide a theoretical understanding of precision and GDoP effects and how they affect the design parameters involved in ranging mechanism and localisation algorithm selection.

In Chapter 4, controlled indoor experimentation was performed to benchmark key components of two implemented source localisation systems. This was sufficient to allow for comparative performance measurement between them.

In Chapter 5, in-situ deployment and operation of the implemented system was used to identify usage problems and failure modes that could not be determined in controlled experimentation, as well as areas for improvement.

In Chapters 6 and 7, network transmission data was gathered through controlled, in-situ experimentation and used in proof-of-concept simulations to improve the network latency aspect of acoustic source localisation in a WSN. Controlled, in-situ experimentation was necessary to provide data traces that had environmental effects which are complex to simulate using theoretical models. Simulation using controlled data traces was sufficient to provide proof-of-concept validation that could be readily extended to real

deployment.

1.3 Research questions

This thesis aims to answer the following questions with respect to acoustic self- and source-localisation for high data rate sensing systems:

1. Can a class of WSN applications be identified which require 3D self-localisation?
2. Does the performance of existing 2D self-localisation algorithms change for these applications?
3. Are there design trade-offs which can ensure adequate 3D self-localisation performance for a given application? (The design trade-offs of interest are equipment requirements and cost, and processing complexity with respect to the software algorithms and hardware platforms chosen).
4. How is the design and integration of an acoustic marmot localisation system affected by real-time, interactive requirements?
5. What are the user and deployment related challenges to be overcome to ensure adoption of an end-to-end marmot localisation system?
6. How can in-network processing capabilities aid the robustness and timeliness of on-line acoustic localisation?
7. Are there data processing approaches used in the developed system that are shared by a class of on-line, high data rate WSN systems?

It should be noted that this work does not aim to provide a generalised approach to wireless sensing. (For example, low power and scalability factors are not explicitly examined in this work.)

1.4 Research contributions

Increasingly in WSN research, scientific progress is coming from multidisciplinary teams rather than individuals and this work is no exception. An outcome of this trend is that it becomes somewhat difficult to establish what the contribution of the individual is both scientifically and towards the goals of the group. Nevertheless, in addition to contributing to the success of various projects, including most notably VoxNet (a project to develop hardware and software for a distributed acoustic sensing platform) and Wavescope (a streaming-data processing engine for high data rate sensing applications), the central contributions of this thesis are as follows:

- The establishment and justification of an evaluation cycle for self-localisation algorithms based on simulation, emulation and deployment (Chapter 2).
- Experimental characterisation and evaluation of three acoustic ranging techniques and two localisation algorithms on platforms with varying computational capabilities, in both indoor and outdoor environments (Chapter 3).

- Implementation and evaluation of a proof-of-concept platform for acoustic ranging, including hardware and software integration and implementation of a suitable ranging algorithm (Chapter 3).
- Two designed, implemented and deployed iterations of an end-to-end, on-line, marmot localisation system. This included in-situ deployment and micro-benchmarks of application specific and general aspects of the system: on-node processing, on-node data archiving, data transfer reliability (Chapter 4).
- The identification and analysis of deployment-related systems issues that are not easily resolved without in-situ operation (Chapter 5).
- The design, implementation and proof-of-concept evaluation of two refinements that increase robustness and timeliness of the marmot localisation system—*Adaptation* and *Lazy Grouping* (Chapters 6 and 7).

The initial source-localisation work described in Chapter 4 in this thesis was produced as part of a collaboration on the acoustic sensing project at Centre for Embedded Networked Sensing (CENS) and the VoxNet project (at UCLA and MIT). A full description of the author's contributions to the project is provided in Section 1.5.1 and is also noted in each chapter where relevant. It should be noted that the actual VoxNet deployment described in Chapter 5 was a team effort, although the network testing, implementation and experimentation was solely performed by the author. Chapters 5, 6 and 7 represent work that furthers the initial source localisation application—this work is solely the contribution of the author and was developed beyond the collaborative effort of VoxNet.

1.5 Thesis structure

This thesis is organised as follows: Chapter 2 establishes the state of the art in localisation techniques, with particular emphasis on WSNs. Chapter 3 examines acoustic, range-based self-localisation with platforms of varying computational capability through in-situ experimentation and simulation. Chapter 4 provides the design and implementation of two iterations of an on-line acoustic localisation system that is used to localise marmots. Chapter 5 describes the deployment of the acoustic localisation system in-situ and discusses the challenges towards non-expert adoption of the system. Chapter 6 identifies latency of data transmission as a key problem in the on-line localisation system and presents two approaches to reducing this latency: Lazy Grouping and Adaptation. Lazy Grouping is evaluated during this chapter. Chapter 7 evaluates Adaptation through simulation using experimentally gathered network data and presents a framework for an adaptive localisation system. Chapter 8 presents future work and concludes on the work in this thesis.

Figure 1.1 shows the structure of the thesis. This thesis has two distinct, but inter-related paths, *self-localisation* and *source-localisation*. They are indicated on the graph, and the relation between the chapters is clearly marked. The casual reader may be more interested in *self-localisation* than *source-localisation*. In this case, the relevant parts that should be read are chapters 2 and 3, as they provide an overview of the self-localisation problem and state of the art with respect to *realistic* solutions; the experimental work reflects this position.

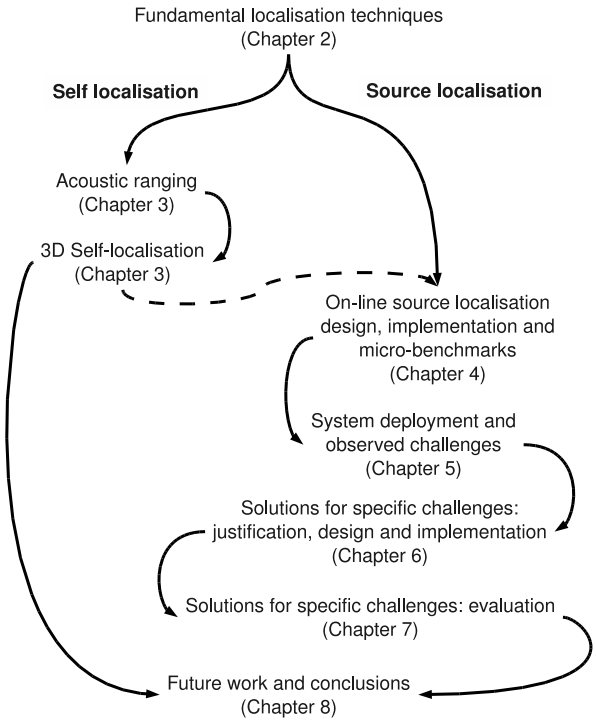


Figure 1.1: The structure of this thesis.

For readers more interested in *source-localisation*, Chapters 2, 4 and 5 provide a foundation in the signal processing principles for source localisation, its challenges, and the design and deployment of end-to-end systems to support it. Chapters 6 and 7 provide strategies to improve the timeliness and robustness of a self-localisation system based on the work in Chapters 4 and 5.

1.5.1 Acknowledgement of collaborative work

This section details the author's specific contributions to the *collaborative efforts* that formed part of this thesis.

Chapters 4 and 5 include work that was completed as part of collaborative work on acoustic source localisation at the CENS at the University of California, Los Angeles, from January 2007 to January 2008. The work presented in Chapters 4 and 5 are a result of the author's contributions to VoxNet and the CENS acoustic project. The author has contributed to both developmentally and experimentally.

Before the author began collaboration with the CENS acoustic project and subsequently VoxNet, the acoustic sensing platforms (Acoustic ENSBox) and the EmStar software framework already existed. The CENS acoustic project has been ongoing since 2002 (or thereabouts), and has seen the development of two iterations of the Acoustic ENSBox platform. This project has been a collaboration between several departments at UCLA (through CENS), namely Computer Science, Electrical Engineering and Evolutionary Biology. CENS' EmStar project has been running in parallel with this effort (although

seperately funded).

The work presented in the first half of Chapter 4 is related specifically to the CENS acoustic project. In the demonstration source localisation system presented in Chapter 4, the author integrated several pre-written components so they would function as a coherent end-to-end system. This included the development of an on-node application (integrating an on-line event detector and a TCP-based networking component), and a server application to receive data from nodes and perform data fusion for position estimation (according to a reference Matlab implementation). The author also wrote a visualiser to display the results of position estimation.

The VoxNet project, a collaboration between UCLA Computer Scence, Electrical Engineering and Evolutionary Biology and MIT Computer Science was awarded funding (and was named VoxNet) in early 2008. This was towards the end of the author's collaboration with CENS. Therefore, with respect to VoxNet, the work presented in the second half of Chapter 4 and 5 of this thesis represents a proof of concept, first version of the VoxNet software platform.

The main co-workers on VoxNet were Lewis Girod and Ryan Newton. The design of VoxNet and intended interaction model presented in Chapter 4 was based on discussions between Lewis Girod and the author. The original design for VoxNet came from Lewis Girod. Lewis Girod also wrote the original publish/subscribe networking library used in Chapters 4 and 5, which was tested and further developed by the author (also documented in Chapters 4 and 5). VoxNet makes use of the Wavescope project at MIT, namely the Wavescope stream processing engine. Both Wavescope and Wavescript, the compiler for the Wavescope engine is developed and maintained by Ryan Newton and Lewis Girod, and the author of this thesis makes no claims to development of either the language or the compiler.

In the VoxNet system presented in Chapter 4, the author was responsible for the development and evaluation of key system components, specifically the real-time data recording functionality (spill to disk), an interactive system shell to send commands to nodes and receive status updates (the WaveScope Shell), a file dissemination protocol and a proof-of-concept data stream visualiser. The author also developed a framework for experimental data gathering, built into the WaveScope shell. This framework was used by the author to carry out several data gathering experiments, both in-situ, and in controlled experiments. Some of the experimentally generated results were further used in Chapters 6 and 7 as data traces to evaluate the performance of the self-organising algorithms. The Dynamic Source Routing (DSR) multi-hop routing component used in VoxNet was implemented by Thanos Stathopoulos as part of his 2006 thesis entitled *Exploiting Heterogeneity for Routing in Wireless Sensor Networks* (Stathopoulos 2006). The author was responsible for integrating the software and providing code changes to support VoxNet directly gaining access to DSR internals (such as current routes and link quality estimates).

Chapter 2

Literature review

This thesis is motivated by WSNs that are built to address real-life problems, specifically localisation. To appreciate the state of the art with respect to localisation and also real-life deployment of localisation related systems, it is important to understand the recent history of WSNs. Section 2.1 discusses WSNs from a historical perspective and pinpoints the reasons why real-life research has not mirrored theoretical work. In Section 2.2, this difference is described by three different views on the WSN design space: the application-centric, the device-centric and the network-centric. The implication of high data-rate systems with respect to these views on the WSN design space is considered in Section 2.3.

Subsequently, the localisation problem is considered in detail. Section 2.4 provides an overview of the localisation process, followed by four sections concentrating on its specifics: range and angle estimation (Section 2.5), signal detection techniques and challenges (Section 2.6), self-localisation algorithms (Section 2.7) and source-localisation algorithms (Section 2.8). Section 2.9 presents the author's contribution to localisation algorithm evaluation by means of a comprehensive discussion of localisation algorithm performance metrics and the proposal of an approach to the evaluation of localisation algorithms.

2.1 Wireless Sensor Networks

Early visions of WSNs in the late 1990s followed the *Smart Dust* (Warneke et al. 2001) concept, where thousands or millions of microscopic sensor nodes (or motes) could be scattered like dust over an area, to sense particular phenomena. Although the nodes were evisaged to be individually resource constrained, the network would be computationally powerful through distributed, collaborative processing. These devices would be so cheap that they could be discarded if broken, and failures of individual nodes would be tolerated by the network, which would *heal* itself, by re-routing communication links between devices in the event of failures. The Smart Dust project at the University of California at Berkeley (UCB) ended in 2001, having provided proof of concept hardware and demonstrations of sub-centimetre sensing devices. It did not, however, provide any complete, or commercially viable solutions to any specific sensing applications. The proof-of-concept nodes created were not fully functional, although they did motivate the creation of the RENE mote (Hill 2003), which was part of a successful controlled vehicle tracking demonstration using a WSN.

Smart Dust left a legacy of parameters deemed important for WSN design and development: a need to focus on distribution of processing, network scalability, miniaturisation and power conservation in homogeneous WSNs. In the years following Smart Dust, many theoretical algorithms and protocols to support hundreds or thousands of networked sensor nodes have been (and continue to be) developed, to prepare for future large-scale deployments. Areas of particular theoretical interest have been ad-hoc

multi-hop network formation and routing, Media Access Control (MAC) schemes, energy management for node and network lifetime, fault tolerance, data collection and middleware to support distributed processing.

Areas of engineering interest have been primarily focused on the development of severely constrained, general-purpose hardware platforms, usually consisting of 8 or 16-bit low power micro-controllers, low data rate radios up to 250 Kilobits per second (kbps) and Kilobyte (kB) of RAM and program memory. The earliest generations of these platforms were developed at UCB (Hill 2003, Polastre et al. 2005).

A problem throughout the history of WSNs however, has been realistic deployment. There is far more research on the theoretical aspects of large-scale wireless sensing than realistic in-situ deployments of sensor networks. Three reasons for this are:

1. **Motivating applications**: WSNs need applications to drive development and real deployment of systems, and to provide requirements and constraints. It is not only difficult to invent such constraints and requirements for systems development, but also problematic, as this approach could lead to systems that can only be used in controlled environments and provide limited suitability.

2. **Cost**: sensing hardware and platforms are expensive, so even deploying tens of devices can be prohibitive. Furthermore, costs surrounding deployment and maintenance often far outweigh the price of sensors and sensor nodes.

3. **Expertise**: WSN systems development requires embedded development and engineering skills in addition to software systems design skills. This is particularly troublesome when Commercial Off The Shelf (COTS) solutions do not exist for the motivating application and the research team consists mainly of computer scientists, rather than engineers.

More often than not, real-life deployments tend to invalidate assumptions made by theoretical studies. This leads to a situation where little theoretical work can cross-pollinate to real-life, as Raman & Chebrolu (2008) observed in a critique of the current state of the art in sensor networks. As an example, the process of WSN deployment is often an unanticipated aspect of theoretical work, yet deploying systems and making them work in real-life is a key point of evaluation of a WSN system.

2.1.1 Application domains

Generically, the commonly accepted benefits of WSNs to any given application domain are: (1) deployment of many devices gives a greater spatial resolution of measurements, (2) wireless communication removes the need for heavy, cumbersome cabling and (3) battery power removes the need for wired power supplies. The potential to harness wireless embedded sensing is relevant to a huge variety of application domains, for example:

- **Medical**: in-hospital monitoring of patient vital signs and location (Malan et al. 2004), posture monitoring (Wu et al. 2008), predictive health care, in-home monitoring for the elderly (Virone et al. 2007)

- **Scientific**: answering scientific research questions and helping improve existing models of phenomena in areas such as seismology (Lukac et al. 2009), biology (Pon et al. 2005), and ecology (Allen

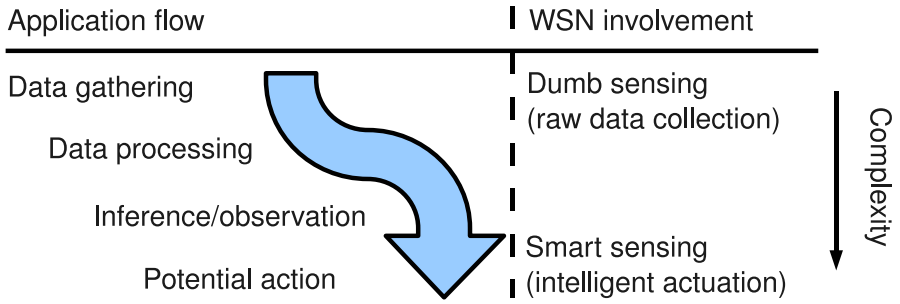


Figure 2.1: WSNs help support application goals to a greater or lesser extent. This figure shows the link between application flow and the increase in complexity for WSNs as they encapsulate more of the application goals.

et al. 2008)

- **Military**: battlefield detection and tracking (Arora et al. 2005), sniper localisation (Simon et al. 2004), border protection monitoring (Wittenburg et al. 2007)
- **Agricultural**: crop monitoring (Langendoen et al. 2006), smart cattle fences (Wark et al. 2007)
- **Industrial**: structural health monitoring (Xu et al. 2004), energy usage analysis, process control (Khakpour & Shenassa 2008)
- **Home**: Automated ambient temperature and lighting control (Haenselmann et al. 2007), personal energy usage monitoring (Kim et al. 2008)
- **Emergency services**: fire detection (Lim et al. 2007), fire fighter safety (Wilson et al. 2007), disaster response (May et al. 2007)
- **Urban sensing**: collaborative sensing and data sharing with respect to the urban environment (Burke et al. 2006)

In each of these applications, WSNs can either enhance the performance of existing solutions, or enable new approaches and processing which were previously impossible (or not apparent). The above represent real motivations to develop and deploy sensor networks.

2.1.2 From dumb to smart sensing

Any application that might make use of a WSN can be broken down into a continuous chain of events: (1) data gathering, (2) data processing, (3) inference/observation, (4) potential action; this chain is shown in Figure 2.1.

The most basic use of a WSN within an application is to gather data and report it back to some data sink for further analysis. This type of network is commonly known as a *sense and send* network. Sensor nodes in this type of network are *dumb* in that the sensor nodes are not required to process any of the

values they sense, but merely to forward them to the data sink. The rest of the application goals are met outside of the WSN.

More complex uses of WSNs in applications may push some, or all, of the processing, information inference and actuation to be made in the network. This is *smart sensing*, where the network is trusted to make decisions and inferences based on data it acquires. The complexity of WSNs increases with the transition from dumb to smart sensing, as shown in Figure 2.1.

The logical aim of smart sensing is to fully automate an application, such that it does not require human interaction (except to view the information produced)—this is colloquially termed *closing the loop*. The added complexity of closing the loop depends on how rigid the application's goals are. For clearly defined applications, closing the loop becomes a case of implementation of the specific requirements. Of course, this is not trivial in itself, and provides potential for many novel approaches.

In contrast, many scientific applications are of exploratory nature—the phenomena being sensed is not well understood, making it difficult to provide clear goals for automation. In these cases, an iterative or staged approach may be applied: initially, all raw data is gathered, such that the domain expert (such as biologist, ecologist, seismologist) can process it and make observations. Subsequently, the domain expert's inference can be transferred to the sensing nodes, allowing them to provide higher level observations about the phenomena. Exploratory research represents the most complex edge of WSN research: goals are not always clearly defined or understood, and there is a need for the complexity of WSNs to grow as the scientific understanding grows. As scientific understanding grows, the abstraction level of inferences made by WSNs can increase, in turn enabling higher level analysis by the user. In this sense, the WSN becomes an *interactive* tool for the scientist to use to further understand phenomena under observation (Allen et al. 2008).

2.2 The WSN design space

Twelve qualitative metrics were presented by Romer (2005) to describe the design space of existing, deployed WSN applications. The metrics also help to identify broad differences or commonalities between existing and future WSN applications. These metrics can be categorised into sensor node- and sensor network-specific properties:

- **Sensor Node properties**: mobility, size, cost, power source, communication modality
- **Sensor Network properties**: heterogeneity, deployment process, communication topology, coverage, connectivity, expected lifetime.

The author proposes that there are three views on this design space described by these metrics which can drastically affect (in some cases over-complicate) the development process for WSN systems. These views are: *application-centric*, *network-centric* and *device-centric*.

The *application-centric* view maintains it is the application's requirements that indicate the software and hardware functionality which is required to be developed. Therefore, network and middleware protocols are designed and implemented as they are needed; the services and protocols which are generically useful will therefore emerge as the number of realistic deployments of WSNs increases. An example drawn from the literature is the Flooding Time Synchronisation Protocol (FTSP) which was originally

developed as part of a sniper localisation system (Ledeczi et al. 2005). This approach has gone on to be the de-facto time synchronisation mechanism in mote-based WSN research, and is integrated into TinyOS 2.x (the most recent version as of writing) as a default service.

The *network-centric* view focuses on designing generic components for building sensor networks as a first principle, so that arbitrary applications at arbitrary scales can be accommodated. Examples include MAC protocols, multi-hop routing protocols, data collection and dissemination protocols. Network-centric research forms the basis of the early period of WSN research (from the late 90s to early 00s), as researchers tried to prepare for the reality of cheap devices that could be deployed in their thousands.

The *device-centric* view builds WSN design choices around an existing hardware platform. This means the platform dictates the extent to which the applications goals can be met, as well as the types of protocols which can be implemented on the device. Many WSN platforms created since Smart Dust, such as the Mica2 and Telos, were optimised for size (miniaturisation) and low energy usage. This comes at the cost of low bandwidth radios, minimal RAM and Flash memory, and low processing power (microcontrollers instead of microprocessors). The device-centric view has heavily biased the approach that is taken in deploying WSNs. By virtue of commercial availability presently, for example highly constrained sensing devices like the Telos have been adopted as the de-facto *general-purpose* WSN platforms. Consequently, real-life application functionality is to be fitted to the capabilities of an unreasonably constrained platform, forcing early optimisation of software and providing limited capability for prototyping software before or during deployment.

As previously mentioned, these are all views on the same WSN design space, however the network- and device-centric views impose restrictions which limit their usefulness. The device-centric forces optimisation before applications are fully realised, due to device constraints. The network-centric view tries to produce generic answers without fully understanding the differences between applications that prevent generality. Both approaches carry an implication that the applications must be fitted to the device, or that a generic enough set of protocols or approach will be sufficient to describe any possible application.

It is the author's expert opinion that in order to understand the realistic benefits of WSNs for a particular domain, it is important to adopt an application-centric view. In application-centric development, the requirements of the application are the over-riding priority. These requirements do not necessarily have to be met using a specific approach: it is far more important to provide a working system that meets the application goals than treating in-network or distributed processing as first principles. Part of enabling wider adoption of WSNs is about being able to provide meaningful results. After application needs have been met, the system and platforms can be optimised to meet other criteria in the design space, such as size and energy consumption.

In spite of the disparity between theoretical and practical sensor network research, many real-life application-motivated deployments have been successful. Additionally, commercial, industrially-motivated products have been created. However, both research deployments and commercial platforms tend to concentrate on sense and send, leaving most of the application logic at the sink. This is largely due to the limitations of the devices being used. Real-life WSN work relating to *smart sensing* is discussed below, as well as sample commercial *dumb sensing* products.

Commercial sensing

Commercial, industrial oriented WSN products tend to follow a sense and send model in gathering data about low-frequency phenomena (such as temperature or humidity). This data is usually made available in an on-line fashion for the user, via a web interface, with appropriate visualisations (graphs, etc). The data is most likely archived into a database for later analysis. Sensor networks in this area tend to optimise against reliability for routing and data transport, and long network lifetime through low power usage. Commercial products tend to vary in which communication standards they adhere to and sensor node extensibility (such as adding new sensors). Some emerging standards exist for lower level communication, such as the IEEE 802.15.4 standard which defines the MAC and PHY layers. Other standards, such as WirelessHART, ZigBee and 6Lowpan sit on top of the 802.15.4 layers and provide ad-hoc network formation, routing and reliable data transfer. Example companies that produce industry-oriented WSNs are Archrock (Anonymous 2008*a*), Dust Networks (Anonymous 2009*c*), Sentilla (Anonymous 2009*g*) and Sensinode (Anonymous 2008*d*). CrossBow (Anonymous 2009*b*) are the main provider for the most widely used research-developed sensing platforms, such as IRIS, Imote2, MicaZ, Mica2, Cricket and TelosB.

Smart sensing

Whilst the *dumb sensing* approach of sense and send logic is most prevalent in real-life deployed WSNs, there are several examples where sensor nodes perform local data processing or filtering. This approach to sensing is *smart* in that the sensor nodes have some degree of intelligence toward the data they are sampling. However, they tend to be limited by resource constraints imposed by the sensing devices. For example, TinyDB (Madden et al. 2005) allows nodes to respond to continuous SQL-style queries to enable more complex data-extraction from a WSN than sense and send. The Tenet (Gnawali et al. 2006) architecture for sensor networks, and before it the Sensor Network Application Construction Kit (SNACK) (Greenstein et al. 2004) take a tiered approach to WSN building, making sensor nodes the lowest class of device, and adding multiple *microserver* class platforms that have a Linux based Operating System (OS), a 32-bit Central Processing Unit (CPU) and Megabyte (MB)s of RAM to form a backbone of communication over the network (and to a sink). This approach provides a reconfigurable sensing framework by defining a clear set of tasks that a sensor node could be expected to perform and pushing more complex application logic to the microserver.

2.2.1 Implications of real-life sensing

The process of moving WSNs out of simulation and lab experimentation into realistic deployment is complex, and brings with it many challenges, hence the relatively slow progression of deployments. Some specific challenges are: deployment cost and effort, deployment environment, the influence of high data-rate applications, and user interaction. The remainder of Section 2.2 discusses these aspects in more detail.

Deployment cost and effort

The cost of a deployed WSN is not just the price of the sensor nodes and sensors. The cost of physically deploying sensors should also be taken into account: tools, mounting posts, equipment testing and calibration, on-site security measures and manual labour are among the expenses. Sensors and sometimes

sensor nodes may have to be buried into the ground, or have special platforms erected on which to be placed. In some cases, the cost of preparing a site for deployment may be higher than the cost of the hardware being deployed. Additionally, long-term WSNs deployed in outdoor environments will require maintenance. Examples include battery replacement, repair of sensors/sensor nodes and obstruction mitigation (for example, cutting back of tree or plant growth that might cover solar panels). Moreover, packaging and security of sensors is important as sensor nodes need shielding from extreme conditions, and protection against being stolen or tampered with. Connectors between sensor and sensor node may easily break, and so must be made robust.

All of the above are costs and challenges which are not immediately obvious when applications are developed for evaluation in the laboratory environment or through simulation. For this reason, they tend to be missed or trivialised in many works.

Deployment environment

Deployment of nodes is a time-consuming and sometimes delicate task. The environment in which a WSN is deployed (dictated by the application) may have many obstacles which absorb, obstruct and reflect signals of interest. This may affect the quality of wireless links, or the quality of data being sensed by sensor nodes. As mentioned previously, the environment may change over time, for example in terms of weather conditions and foliage/obstruction growth. Wireless communication is unreliable in general and environmental effects can cause unpredictable communication problems. This may lead to missed deadlines or a perceived lack of responsiveness in WSN systems that require real-time feedback.

High data rate applications

Some applications require data to be gathered at hundreds to tens of thousands of samples a second. These types of applications cannot be developed as sense and send applications, because the volume of data is too high. This implies that nodes must perform local data filtering and computation.

This class of applications cannot be easily-described by frameworks that assume low data rates and low duty cycles (where nodes sleep to conserve energy). The requirements of high data-rate applications are difficult to meet with resource constrained platforms as they tend to be optimised for reduced energy consumption (thus low rate sampling), and require optimised data processing implementation (such as fixed point or integer arithmetic). Limited memory and processing resources make prototyping next to impossible without including customised hardware. This problem is discussed in more detail in Section 2.3.

Interaction and the non specialist user

Ideally, WSN systems will be deployed by non-experts (that is, non-engineers or non-computer scientist's). These users should only be concerned with the application level details of the system: what the data and information that is coming out of the system looks like. Users should not have to care about issues related to the network level: it should be assumed that the system is robust enough at a network level to be able to deal with network formation, reconfiguration and data transfer in a way that is abstracted from the user.

For the non-expert WSNs need to be easily deployed and initialised and provide suitable means to allow interaction (to understand the state of the system) whilst the system is running. Many scientific

deployments will likely be attended in the first instance. In these cases, the scientist may want to use the network as an exploratory tool. The WSN should be able to support this: particularly tuning and reconfiguration of software that is running.

2.2.2 Summary

Section 2.2 discussed the gap between theoretical and real-life WSN research, caused by both the difficulty in building real WSN systems and a device or network-centric view of the WSN design space, rather than an application-centric view. Device and network-centric views come from the early visions of WSNs as Smart Dust, and the resource-constrained sensing platforms created alongside this. Many solutions to WSN application currently begin and end with dumb sensing. The use of a WSN as a scientific tool, in research applications is a strong motivator for smarter, more intelligent sensing. When systems are sensing high data rate phenomena, and scientists are in attendance of deployments, there are greater requirements on in-network processing and timely presentation of results to enable interaction. This is dicussed further in Section 2.3.

2.3 High data-rate systems and frameworks

High data rate sensing systems are those which sample a phenomena and thus generate data at 100 Hz or more (Werner-Allen et al. 2008). Phenomena that require high sampling rates tend to be high frequency phenomena. This is due to the Nyquist sampling rule: to correctly represent a signal of F Hertz, it must be sampled at least $2F$. Therefore, a phenomena of 100 Hz signal must be sampled at 200 Hz, and so on. Examples of high data-rate phenomena that have been proposed to be sensed using WSNs are:

- Audible acoustics applications: gunshot (Ledeczi et al. 2005) and animal classification and localisation (Ali et al. 2007). Sampling rates are typically in the range of 1 kHz—48kHz.
- Water pressure applications: transient monitoring in water distribution pipes (Stoianov et al. 2007). Sampling rates are typically from 1 Hz-2 kHz, depending on transient frequencies.
- Seismic/vibration applications: earthquake monitoring (Lukac et al. 2009) and volcano monitoring and event detection (Werner-Allen et al. 2006) and structural health monitoring (Chintalapudi et al. 2006). Sampling rates are on the order of 100 Hz-200 Hz.

Several systems which expressly deal with the problem of sensing high data-rate phenomena with wireless sensing networks have been proposed. In Section 2.3, some of these high data-rate systems and frameworks are described. Characteristic of all high data-rate sensing systems is that the sensing nodes generate more data than they can send in real-time to a higher network tier or central controller. The systems and projects surveyed are: VanGo (Greenstein et al. 2006), Lance (Werner-Allen et al. 2008), Sniper localisation (Ledeczi et al. 2005), Toad monitoring (Hu et al. 2005), Wisden (Xu et al. 2004) and NetSHM (Chintalapudi et al. 2006).

In all the applications above, there is either an implicit or explicit notion of timeliness. Most of the above applications have a conceptual model whereby the user attends the deployment and requires feedback from the sensor network: in VanGo, the sensor network provides data for the user so that

they may interactively tune the filtering parameters currently running on sensor nodes, in NetSHM and Wisden, the user wants quick feedback on any interesting structural vibrations as they occur. The operation of WSN systems that provide timely results to the user are of interest to the work in this thesis, especially the processing techniques used to gather and filter data for sending.

There are however, other high data rate WSN systems that do not aim to provide on-line feedback to the user. An example of this is EnviroMic (Luo et al. 2007), which aims to record a single stream of audio over severally spatially distributed nodes. These types of system are not considered further.

2.3.1 Gunshot localisation and classification

Ledeczi et al. and Volgyesi et al.'s work on real-time gunshot localisation and classification applications using WSNs (2005, 2007) dealt with the problem of high data rate transfer by implementing on-node processing using custom-made hardware. Given that the application requirements are so clearly-defined, the approach taken by the solution is highly optimised: classification and localisation are specifically tuned for gunshot characteristics. Raw acoustic data analysis is not necessary for the application, whose main constraint is providing the estimated location of the target and weapon type as fast as possible. On-node processing is enabled by custom Field Programmable Gate Array (FPGA) based hardware.

2.3.2 Toad monitoring

The toad monitoring application is motivated by a need for automatic recognition of vocalisations for frog and toad census (Hu et al. 2005). The application specific goal of the work is similar to that of the gunshot localisation application: not all data needs to be extracted from the network, only the high-level inference (existence of cane-toads) need be reported to the user. There are two prototypes presented: a stargate only system, and a hybrid stargate and mote system. The stargate is a microserver class platform, equipped with a 32-bit ARM-based microprocessor.

In the stargate-only system, a raw audio stream is sampled at 22 kHz and event detection and classification is performed on windows of data to detect and classify toads or frogs in real-time. In the hybrid system, motes cannot perform the necessary classification operations or stream back the raw data in real-time. Their role is therefore limited to sampling and compressing windows of raw audio, sending them back to the stargate.

The compression scheme that runs on the nodes destructively gates the raw audio data being sampled, such that any periods of audio below a certain amplitude threshold are replaced by a single code and number of samples. Motes are required to sample at 10 kHz in a non-continuous cycle: sample for 15 seconds, compress data and send to sink (30 seconds). To address the problem of only being able to sample 15 out of every 45 seconds of data, a scheduling algorithm is suggested, where physically close motes are alternately tasked between sampling and sending data.

2.3.3 VanGo

VanGo (Greenstein et al. 2006) is a system which is designed to capture high frequency phenomena (sample rates in kHz) using mote devices which are constrained not only by processing capability, but also network communication bandwidth. The target platform for VanGo is the TMote sky (or TelosB) platform. VanGo is designed around an interactive model where data is sent from sensor nodes to a

microserver or sink for further analysis. The user at the sink can experiment with and tune the application on the nodes at run-time. VanGo is evaluated with two different applications: one for acoustic data monitoring and tuning, and one for the analysis of electromagnetic activity in the brains of rats.

A VanGo program is composed as a chain of linear filters, which consume and produce windows of audio data. VanGo has a library of classification, measurement, compression and transformation filters which were developed for the two motivating applications. Generic filters include: amplitude and frequency gates, a Finite Impulse Response (FIR) filter and acoustic compression filter (ADPCM). There also exists an application-specific spike-detector filter which can detect and compress neuron spikes for the electrophysiological application also exists.

VanGo limits node operations to time domain filters in the following categories: classifiers, transformation and compression algorithms and measurements. This is because more complex signal processing operations cannot be performed in real-time (such as the Fast Fourier Transform, which is observed to run at 1/8th the speed of real-time in one example).

2.3.4 Wisden

Wisden (Xu et al. 2004) is an application-specific system for collection of structural health monitoring (SHM) data. It is the forerunner of NetSHM (Chintalapudi et al. 2006). The motivating application is the monitoring of structural vibrations using motes equipped with accelerometers in order to identify interesting events. The goal of motes in the Wisden network is to detect vibration events of interest, then sample and compress data before sending back to a central point.

The event detection approach is based on an energy thresholding algorithm. When an event occurs, the data is compressed and sent to a sink for further analysis. Wisden uses two lossy data compression techniques: wavelet analysis and Run-Loss Encoding (RLE), where continuous values within a range are replaced with a single code and number of samples. In experimental evaluation, the Wisden system was shown to have large latency: in a ten node network, it took 5 minutes to gather 20 seconds worth of seismic data corresponding to an event. This time lag was noted to be adequate for SHM application purposes.

2.3.5 Lance

Lance (Werner-Allen et al. 2008) is a framework for high frequency data collection. The motivating application for Lance's design was data collection from a sensor network monitoring seismic events around volcanoes. Specifically, the system aims to maximise the network lifetime by extracting only important data from nodes in the network, rather than gathering all of the data. Of all of the systems presented in Section 2.3, Lance is purposely the most generic: it defines a formal framework for high data-rate collection applications.

A Lance data collection system is built around a network of motes and a single data sink. The basic application flow is as follows: sensor nodes sample raw data into evenly-sized windows, referred to as Application Data Units (ADUs). These ADUs are stored locally in flash memory, where they can be indexed and retrieved upon request from the sink. For each ADU that it samples, a node computes a summary using a pre-defined *summarisation function* which it sends to the sink. The summarisation function is user-defined, and related to the application: an example is computing the mean of the samples

in the ADU.

At the sink, the summaries received by each node in the network are passed through a linear chain of *policy modules*. The policy modules assign *value* to ADUs, based on the content of the summaries provided by nodes. An example policy might be: all ADU summaries over a given threshold are valuable. Each policy module modifies the *value* associated with a summary before passing it to the next policy module.

The output of the linear chain of policy modules goes to an *optimiser* module, which based on their value and the current energy profile of the network, decides the ADUs that should be scheduled for collection from given nodes in the network. The goal of the optimiser (and thus Lance) is to: download the set of ADUs which maximise the total *value*, subject to the lifetime target (Werner-Allen et al. 2008).

2.3.6 Summary

In all of the high-data rate systems discussed in Section 2.3, there are device-centric constraints imposed by the capability of the WSN platforms used (typically Mica2 or Telos). This causes a tension between the application requirements which the WSN researchers are trying to meet and the devices which are available to perform the processing. Based on the systems discussed above, the main issue for high data-rate sensing is that devices cannot send the all the raw data they sample (in a timely manner), but in many cases they cannot perform the processing required to filter it either. One means to relieve this tension is by adding hardware specific support to the devices to make the platforms more powerful. This was done by Ledeczi et al. (2005) in the gunshot localisation application, where an FPGA board is attached as a custom add-on board to the mote device, effectively treating the mote as a smart, time synchronised radio. The other two approaches to resolve the tension are to either compress the data using some lossy or lossless mechanism, or decide which data is useful and send only that. VanGo and Wisden supports both of these approaches: VanGo allows local filtering and data compression, whilst Wisden detects events and then compresses them before sending. Other systems choose one approach or the other: the gunshot localisation application, Wisden and Lance only send useful data, whereas the toad monitoring application chooses lossy compression of signals. However, the extent to which data can be filtered and transformed is severely limited by the devices being used. For example, VanGo asserts that only optimised time domain filters may be used on the node in order to maintain real-time functionality. Similarly, because the toad monitoring application cannot perform classification on the nodes, it resorts to data compression, meaning the node can only non-continuously process 15 seconds of data out of every 45 seconds.

Some applications may not be able to deal with compressed signals, especially if signal processing operations are going to be performed after the data has been received.

Given the above problems, it is not clear that the device-centric approach to the high data-rate WSN design space is justified in the context of real-life applications: the toad monitoring application runs perfectly on stargate class nodes, and the addition of Mica2 motes seem to drastically reduce the quality of the sensing that can be performed. Wisden justifies the use of mote class devices by virtue of their form factor, not only in terms of the device, but also their packaging (and potential associated battery size).

It is apparent that within the wide class of high data-rate applications, there is a subtle difference between the needs of different phenomena: acoustic applications tend to have higher data rates (order of kHz), and as such have memory, processing and timeliness needs which are more stringent.

The device and network-centric views have far-reaching consequences not just for the general WSN design space, but specifically for the design space associated with localisation in WSNs. Examples of this with reference to theoretical and practical localisation will be seen throughout the remainder of this chapter.

2.4 Localisation in WSNs

In a physical context, *localisation* is the process of determining the position of a *target* based on information that can be gathered about it by *observers*. Specifically in WSN systems, localisation holds a dual role: firstly, determining the physical positions of nodes in the network (*node-localisation*) and secondly, determining positions of other phenomena of interest relative to the network (*source localisation*). Self-localisation, where node-localisation is carried out by nodes in the network is especially important for ad-hoc networks where the positions of each node in the network cannot be manually acquired. This may occur because of limited time during deployment, or because de-facto position estimation hardware does not function correctly in the environment (for example, GPS receivers). Source localisation has application in a variety of areas, many of which are complementary to the advantages wireless networked sensing offers. For example: animal localisation and tracking (for ecological research), person tracking and service proximity (in office or medical environments), asset tracking (in warehouses), gunshot localisation or vehicular tracking (in battlefield environments), speaker localisation (as part of smart offices or for lectures and presentations), smart fences (for border monitoring/enforcement) and seismic event localisation (for earthquakes). In any of these examples, a network of devices, either scattered or installed into an infrastructure provide a compelling way to monitor phenomena and detect events.

The localisation process can be thought of as *cooperative* or *passive* (Savvides et al. 2004). When targets assist observers in the localisation process (for example by exchanging data or responding to messages), it is cooperative. Self-localisation is the most distinct example of cooperative localisation. When targets do not assist in the localisation process, they are passive—this style of localisation is more often associated with localisation of external sources, such as animals or vehicles. In general, a localisation algorithm transforms the input data (ranges, angles, known positions) into relative or absolute position estimates of one or more unknown signal sources. The input data provided to a localisation algorithm component will typically be the ranges and bearing estimates made by each node in the network, as well as any pre-determined node positions.

The high-level localisation process is shown in Figure 2.2. Observers sense data about a target, which along with the observers' positions are used as input to a localisation algorithm. The localisation algorithm produces an estimate of the position of the target. The type of data that is sensed about a target will vary depending on the context of the localisation. It could be electromagnetic signals which are being bounced off a target by an observer (as in RADAR), or it could be acoustic signals emitted by the target and detected by the observers (for example, a calling bird). In both self and source localisation in WSNs, the observers are sensor nodes, equipped with sensors enabling them to sense the signals emitted

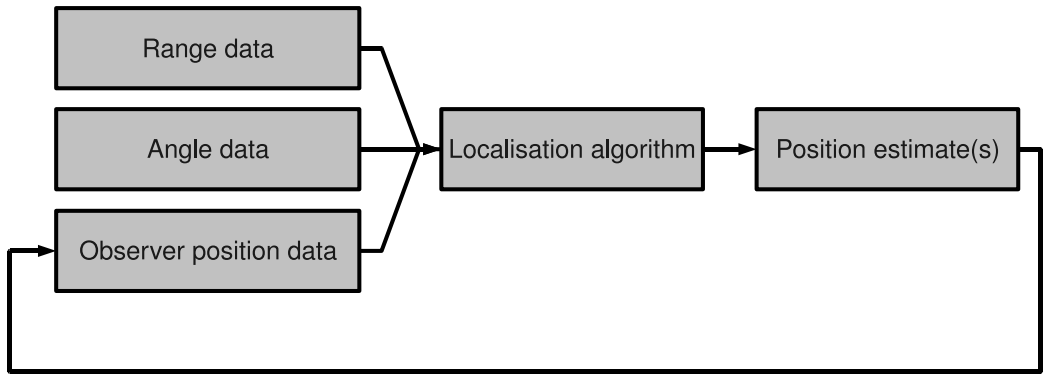


Figure 2.2: The localisation process. Input data of ranges, angles and prior positional knowledge of observers can be used as input to a localisation algorithm. The positional output may be used for input to a further iteration of a localisation algorithm.

by targets (for example microphones or hydrophones). In self localisation, the targets are also sensor nodes, but in source localisation, the target is external to the network.

2.5 Range and angle estimation

The first part of the localisation process involves gathering observation data about targets, which can be used as input to the localisation algorithm. These observations are usually made by estimating spatial or angular separation between observer and target. Therefore, most approaches can be categorised as either *distance* or *angle based*. Typically, the target will emit or reflect some characteristic signal which can be detected by the observer and used to estimate the distance or angle between them. These signals will most likely be electromagnetic or acoustic. Signals in the visible spectrum which can be sensed in two dimensions (images) by cameras are not considered in this chapter. The interested reader is directed to Barton-Sweeney et al. (2006), which describes the technique of estimating distance between devices using epipoles.

The purpose of Section 2.5 is to highlight the various methods of range and angle estimation which have been presented in the literature for both self and source localisation. From this, the relative complexities and benefits can be understood. A taxonomy of different approaches to range and angle estimation is shown in Figure 2.3. For range estimation, techniques are based on either the speed of a signal's propagation, its strength at the observer, the interference between signal or the average communication radius of the signal. For angle estimation, approaches are based either on a closed-form combination of Time difference of arrival (TDoA) and Least Squares (LS) Maximum Likelihood (ML). Connectivity and interferometry approaches can only be used in a self-localisation context, whereas all the other range and angle approaches can be used in both a self-localisation and localisation context. Connectivity-based approaches are not considered in this thesis: they provide only a coarse estimate of range that is not applicable to many real-world applications, and certainly not any which require accurate self-localisation.

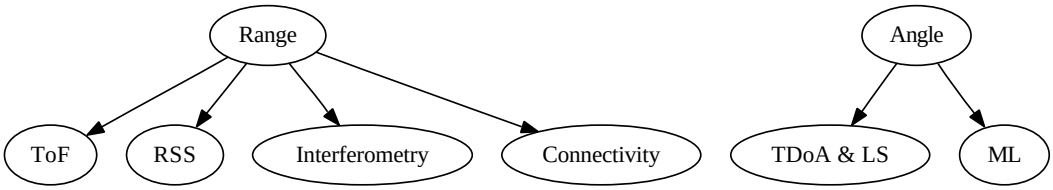


Figure 2.3: A taxonomy of the different range and angle estimation techniques.

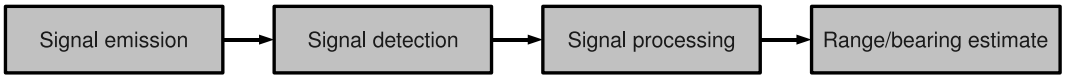


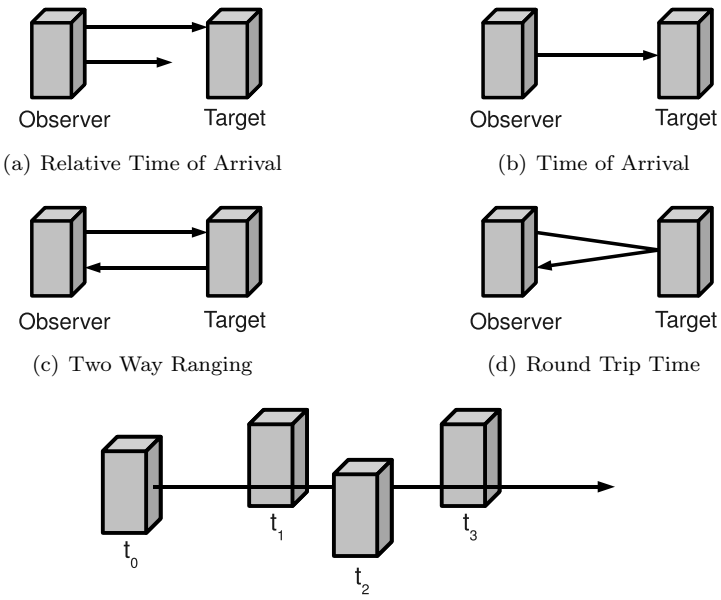
Figure 2.4: The ranging and angle estimation processing pipeline.

For mechanisms which involve measuring either the propagation strength, speed or interference pattern of a signal to estimate distance or angle, there is a basic processing pipeline: (1) a signal is emitted, (2) this signal is detected and (3) the signal is processed to produce a distance or angle measurement. This pipeline is shown in Figure 2.4.

The performance of these mechanisms are quantified by three metrics: *accuracy*, *precision* and *operational range*. Accuracy describes how the estimate matches the true quantity (such as range, angle or position), and precision describes the width of the residual error distribution observed through repeated estimates. A high accuracy ranging mechanism will give a close match between the estimated and true distance between two devices (or the true distance with a systematic bias). The quantification or classification of accuracy (i.e. *how* accurate) is relative to the requirements of the measurement. For example, centimetre-accurate distance estimation between two devices that are several hundred metres apart could represent a high accuracy, but a low accuracy if the devices are only several centimetres apart.

The measurement of precision assumes that the estimation mechanism exhibits a normal distribution of error given repeated measurements; therefore taking the mean of multiple samples will give a more accurate estimate. A high precision estimation mechanism will require fewer samples than a low precision estimation mechanism to be taken between target and observer to ensure that the sample mean is close to the actual mean of the estimate distribution. Precision can be quantified using the standard deviation of the estimation mechanism: if the standard deviation is low, then the width of the distribution of residual error is narrow, implying a higher precision. As previously with accuracy, the measurement of high precision is relative to the actual quantities being measured.

Operational range describes the distance over which the ranging implementation can be used. Obviously, this depends on the environment in which ranging takes place, but has an effect on the density at which sensor nodes can be deployed, for example. Of course, it is possible that both accuracy and precision can change over distance. It is convenient to classify both the degree of accuracy and precision of an estimation mechanism relative to the average spacing of devices or the mechanism's maximum operational range (angle or distance).



(a) Relative Time of Arrival

(b) Time of Arrival

(c) Two Way Ranging

(d) Round Trip Time

(e) Time Difference of Arrival

Figure 2.5: The five different approaches to ToF estimation.

In the rest of Section 2.5, Time of flight (ToF), Received Signal Strength (RSS), interferometry and Angle of Arrival (AoA) approaches are considered.

2.5.1 Time of Flight estimation

Time of flight (ToF) approaches estimate the distance d between target and observer by measuring how long a signal with known propagation speed v takes to travel between them. This is expressed as

$$x = vt \tag{2.1}$$

where t is the time taken. Both electromagnetic and acoustic signals can be used to infer separation; commonly used signals are Radio Frequency (RF), ultrasonic and audible (wideband) acoustic. These are discussed further in Section 2.6.5.

There are five different methods to estimate ToF: Relative time of arrival (RToA), Time of arrival (ToA), Two-way ranging (TWR), Round trip time (RTT) and TDoA. These methods are differentiated by (1) whether they require cooperation between observer and target, and (2) how cooperating stations are synchronised. Each of these approaches are now discussed in more detail.

Relative Time of Arrival

The Relative time of arrival (RToA) approach uses the relative difference in propagation of two signals in order to measure the distance. Typically, this means that a radio and acoustic signal are sent from the target at the same time. Because the radio and acoustic signals propagate at different speeds (speed of

light and sound respectively), they will arrive at the listener at different times (the radio signal is treated as arriving instantaneously). The listener must detect the arrival times of both the radio and acoustic signals and use the difference between them to calculate the distance between itself and the sender. RToA is implicitly synchronised at the sender, by assuming that both signals are sent at $t_s = 0$.

The accuracy of the ToF estimate is affected by (1) how accurately the observer can detect the RToA of the signals and (2) how accurately the target can send both signals at the same time (or at some known offset from one another). The RToA approach is common in constrained embedded systems. For example, the PushPin lateration system (Broxton 2005) uses a *pinger* which generates both ultrasonic and light events at the same time; Calamari (Whitehouse 2002), Resilient Localisation (Kwon et al. 2005) and (Simon et al. 2004) all use acoustic and RF signals on the Mica2 platform.

Time of Arrival

In the Time of arrival (ToA) approach, both target and observer are synchronised to a common time-frame. The target sends a signal, identifying the time it was sent, and the receiver records exactly when the signal arrives. The ToF is then calculated by subtracting the send time from the receive time. The accuracy of the ToF estimate is affected by (1) how accurately the observer can detect the ToA of the signal and (2) how accurately the two devices are synchronised. The higher the frequency of the signal to be detected, the more accurately the devices must be synchronised, and the more the observer must sample the channel that the signal is being sent on (to detect its ToA). For example, given that an RF signal will propagate at the speed of light, a ToA detection error of 1ns equates to a range estimate error of 30cm. If the ranging accuracy required is around a metre, synchronisation between the two nodes must be at least within 3ns (Savvides et al. 2004).

The most well-known implementation of the ToA approach is the Global Positioning System (GPS). Each satellite transmits a coded radio signal which is picked up by the GPS receiver, which simultaneously replicates the signal itself. GPS receiver clocks become highly accurate by locking-on to the satellites high accuracy signals. The physical distance between the two is worked out by calculating the phase difference between the signal the GPS receiver is creating and the signal which is being received by the satellite, and multiplying it by the speed of light.

Two Way Ranging

Two-way ranging (TWR) is a cooperative ranging mechanism, where one node emits a signal and the other node replies with a signal. The target keeps a track of how long it took from sending its own signal to receiving the reply (t_A), and the observer keeps track of how long it took from receiving the first signal to sending its reply (t_B). The ToF $\hat{t}$ is then inferred by

$$\hat{t} = \frac{1}{2}(t_B - t_A) \tag{2.2}$$

The accuracy of the ToF estimate is affected by how accurately the observer and target can detect ToA of their signals. There are two examples of this approach in the WSN literature: the BeepBeep (Peng et al. 2007) and Two-Way Time Transfer (TWTT) (Lanzisera et al. 2006, Lanzisera & Pister 2008).

Round Trip Time

In Round trip time (RTT) distance estimation, a signal is bounced off the target by the observer. The time taken for the signal to return to the observer divided by two gives the ToF. This approach does not require target cooperation, and is not based on any monitoring any signals the target emits. The main challenges associated with the approach (and its accuracy) are related to the observer's ability to detect the reflected signal, which may be weak due to environmental effects, and also subject to multi-path and reverberation effects.

There are several well-known implementations of RTT in current usage: ground RADAR (for tracking aircraft, for example), air plane altimeters (Anonymous 2009*a*), as well as proximity sensors, and domestic laser (Anonymous 2009*d*) or ultrasonic range-finders (Anonymous 2009*e*).

Time Difference of Arrival

Time difference of arrival (TDoA) estimates distance by measuring the difference in arrival time of a signal at spatially diverse sensors. Figure 2.5(e) on page 23 shows a signal emitted at t_0 passing through a sensor network, arriving at each sensor node at times t_1, t_2 and t_3 respectively. The relative ToFs between each of the sensors can be determined and then used to estimate the distance between them. The sending time of the signal does not need to be known, as it is only the relative difference in arrival time which is required to estimate relative distance. Although the example in Figure 2.5(e) on page 23 shows a signal arriving at different sensor nodes, it is also common for TDoA techniques to be used on an array of sensors *co-located* on a single device. For example, an array of microphones connected to a single data acquisition board. In this case, the spacing between the sensors is already known, and so can be used to estimate the AoA. This is discussed further in Section 2.5.4 on the following page.

A requirement on TDoA is that the channels being compared are synchronised to the same time frame. In cases where sensors are physically connected to the same channel, time synchronisation is implicit, but in other cases a suitable time synchronisation mechanism must be employed. Therefore, the overall accuracy of TDoA estimates between nodes is affected by (1) how accurately each observer can detect the arrival time of the signal and (2) how accurately time-synchronised the sensors are. Thunder (Zhang et al. 2007) uses acoustic signals emitted from a mobile speaker to create events in a time-synchronised mote network.

Summary

Each of these approaches to ToF estimation have different requirements on target and observer in the manner that devices are synchronised. The method that is most commonly used for constrained WSN devices is RToA, although when suitable time-synchronisation is available, ToA approaches using acoustic signals are feasible (Girod 2005). TWR represents a compelling way to estimate range with little or no inter-node synchronisation, at least for acoustic methods (Peng et al. 2007). TDoA is a non-cooperative approach to ToF estimation, which is suitable when nodes cannot generate their own signals and must rely on an external device to do it (Zhang et al. 2007).

2.5.2 Received Signal Strength/Signal Energy

Range estimation approaches based on received signal strength or energy take advantage of the fact that an emitted signal spreads over a greater area the further it gets from its source. This means the intensity of the energy becomes reduced with distance, as it is covering a greater area. The inverse-square law dictates that the strength (or energy) of a signal is inversely proportional to the square of the distance from the source of the signal (this law is true for both electromagnetic and acoustic signals). The strength of a signal measured by an observer can therefore be used to estimate the distance to the target (or *signal source*), given sufficient reference information by which to calibrate measurements. Reference information may be the strength that the signal was originally emitted at, or a priori reference measurements.

On many WSN platforms, the instantaneous Received Signal Strength (RSS) for radio communications is made available in a hardware register, which can be read and used in software. Because the transmission strength at the sending device is also known, the distance between observer and target (in this case sender and receiver) can be calculated. The approach of using RSS for distance estimation is highly desirable in self-localisation, because it can be gathered using hardware which is already required for communication. As a result, several theoretical works in localisation assume distance estimation by RSS.

In real-world environments however, RF signals are highly affected by problems such as multi-path, shadowing and channel fading (Patwari et al. 2005). Empirically, Received Signal Strength Indicator (RSSI) has been found to be an inaccurate indicator of distance (Whitehouse 2002, Lymberopoulos et al. 2006). The Mica2 and Telos WSN platforms were used in the experimentation of both works above. Section 2.6.4 on page 32 discusses the specific effects of interference on signal propagation (both acoustic and electromagnetic).

2.5.3 Interferometry

Interferometry can be used to estimate distance between targets and observers based on interference of signals they emit. The interferometry approach requires that two cooperating devices emit signals at slightly different frequencies. The composite waveform produced has a low beat frequency *interference pattern*, which can be used to infer the distance between them. An interferometric ranging system was implemented in (Maroti et al. 2005), using RF communication between Mica2 motes. The interference patterns of two transmitting devices were observed by two other devices. In this case, the relative phase offset between two receivers depended on the four distances between the two transmitters and two receivers. In measuring the phase offset at different carrier frequencies, the linear combination of the four distances was calculated. In an ideal scenario, range estimates were shown to have as little as 3cm error at ranges up to 160m. However, it has been observed that interferometry is highly likely to be affected by multi-path conditions present in many obstructed environments (Girod 2005).

2.5.4 Angle of Arrival

Angle of Arrival (AoA) (AoA) is an estimate of the angle at which a signal of interest arrived at an observer. Like range estimates, AoA estimates form the basis of data to be used as input into localisation algorithms. The AoA is described by one parameter in two dimensions: the azimuth (θ), which is typically measured in degrees or radians. In three dimensions, the AoA is really a Direction of arrival (DoA), and

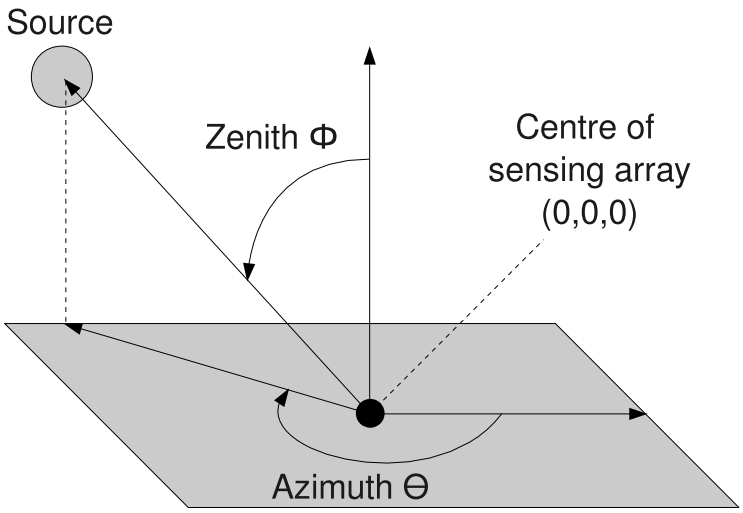


Figure 2.6: This diagram shows the relationship between azimuth θ and zenith ϕ angles with respect to a sensing array's geometry and an unknown sound source. Angles are derived from the centre of the sensing array, which is assumed to be (0,0,0). Note that the sensing array could be a set of co-located sensor nodes, or a single node with multiple sensors.

is described by the azimuth and zenith (θ and ϕ, both measured in radians or degrees). Figure 2.6 shows both azimuth and zenith with respect to the central point of a sensing array's geometry (which could be several sensor nodes, or a single node with multiple sensors).

To avoid confusion, AoA and DoA are used synonymously throughout this chapter, regardless of the dimensionality considered. It is important to note that unlike range estimates, angle estimates are made relatively to the *orientation* of the observing device. This means that in order to translate these relative angle estimates to a global frame of reference (that is, the whole sensor network), the relative orientations of nodes in the network must be known.

Localisation algorithms can make use of angle estimates when range information is not available, or potentially highly inaccurate. A good example of this is source localisation where the time of arrival of a signal is difficult to detect (due to the complexity of a given signal), and lacks coherency (this is discussed further in Section 2.6.4 on page 33). Angle and range estimates can be used together in self-localisation algorithms to help remove outliers and reduce density requirements of network deployments (Girod 2005).

This thesis makes use of, but does not innovate in, localisation based on DoA estimates. Therefore it is important to describe these techniques, but it is not necessary to go into great mathematical detail except where relevant.

In general, approaches to angle of arrival estimation are based on either Maximum Likelihood (ML) or TDoA and Linear Least Squares (LS) estimation. These approaches are detailed below.

TDoA and LS

Section 2.5.1 showed that TDoA could be used to estimate the relative distances between sensor nodes (or several co-located sensors on a node) by finding the difference in arrival times between a signal which all nodes detected. When the relative positions of all sensor nodes (or co-located sensors) is known, the TDoA values can be used to estimate the AoA relative to the centroid of the geometric shape created by the positions of the sensors. This is done by using the relative TDoAs and known positions of the sensors as input to a closed-form Least Squares (LS) system. The system is solved for the unknown AoA relative to the known geometry of the sensors.

LS is a commonly used approach for solving closed form, linearised equations because there are well-known approaches to pose the and solve the problem using matrix notation and manipulation techniques. Least Squares is discussed more thoroughly in Section 2.7.1 with respect to lateration.

The accuracy of a TDoA/LS approach is limited by how accurately the arrival time of the signal can be estimated at each sensor. Approaches to signal detection are discussed in Section 2.6. The software to support self-localisation on the Acoustic ENSBox platform (equipped with a tetrahedral array of four co-located microphones) was implemented an acoustic Angle of Arrival algorithm in 3D using a TDoA/LS approach (Girod 2005). AoA was resolved with a ± 2 degree azimuthal accuracy (with standard deviation of 0.96 degrees). Another approach is presented in Yao et al. (1998).

The TDoA and LS method of calculating AoA is a two-step method, as it firstly calculates the relative arrival times between all the sensors, and secondly solves a system of equations using a LS approach to find the AoA (for azimuth and potentially zenith). The Maximum Likelihood approach to AoA estimation however operates in a single step, directly on the raw sensor data. This distinction is important when considering multiple targets.

Maximum Likelihood

Maximum Likelihood (ML) is a general statistical approach to fit a parametrised mathematical model of data to empirical observations. ML attempts to find the most likely parameters for the mathematical model given the input data. For AoA estimation, the ML approach essentially finds an estimate of the AoA which is most likely given the input data. The model used by ML for signals sampled at each sensor node (or co-located sensor) is of the actual signal plus a Gaussian noise component. It is assumed that the noise component is uncorrelated, but that the signal component at each sensor is highly correlated.

The classical example of ML-based AoA estimation is the time-domain delay-and-sum *beamforming* filter (Birchfield 2004). In this approach, a window of data from all sensor channels being used to estimate DoA are iteratively *delayed* relative to one another, and summed at each iteration. This approach is also called *steering*. The energy of the summed signal at each iteration can be calculated, and the point at which maximum energy is found indicates the point at which the signals in each channel most match. Since the relative delays correspond to an angle, once the maximum energy is found, so is the AoA estimate.

It has been noted that the classical time-domain approach is not necessarily suitable for wideband signals (such as audible acoustic signals), prompting frequency domain-based ML approaches (Chen et al. 2002*a*). In actually implementing a frequency domain ML solution, it is necessary to apply the

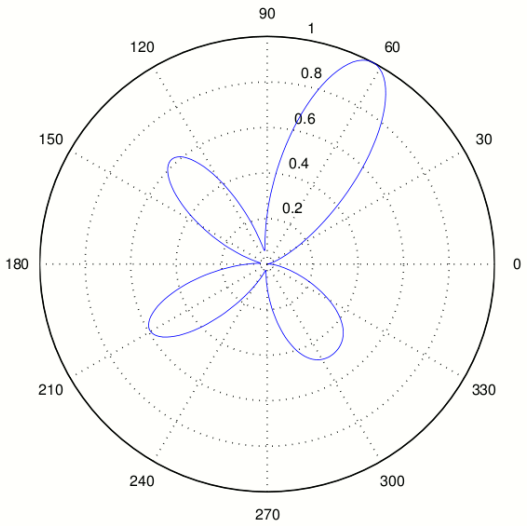


Figure 2.7: The data of an AML vector shown in a polar plot. Numbers around the outer circle represent the angle of arrival, relative to the centre of microphone array on the sensor node. Inner circles represent varying degrees of likelihood, scaled between 0 and 1. In addition to the main (most likely) beam at around 65 degrees, this particular plot shows weaker *side lobes* which are indicative of spatial aliasing (at 135, 210 and 310 degrees respectively).

Discrete Fourier Transform (DFT) to convert into the frequency domain. This causes edge effects which mean that an *exact* ML solution for finite length data does not exist. To address this, an approximation of the ML called the Approximated maximum likelihood (AML) algorithm (Chen et al. 2002*a*).

When transformed into the frequency domain, the signal is binned into frequency bands. Between each channel, the difference in phase information is a function of the DoA of the source. The AML solution finds the largest magnitude of energy across all frequency bins and channels in an equivalently to the delay and sum beamforming operation. The result of the AML algorithm is a likelihood vector J, each element of which refers to a specific AoA. The value at each element is the likelihood, thus the ML estimate is the maximum value of J. The likelihood vector is often presented as a polar plot, as in Figure 2.7. Another example of a frequency-domain ML based AoA algorithm (for wide-band sources) is the MUSIC algorithm (Tung et al. 1999).

Signal propagation and sensor geometry challenges

Assumptions about the propagation of signals and the geometric configuration of sensor nodes (or co-located sensors) affect the accuracy of AoA measurements.

AoA algorithms make assumptions about the behaviour of signals as they propagate from the target (or source). Unless explicitly specified, algorithms make a *far-field* assumption about the signal propagation. The far field model of propagation assumes that the wavefront of the signal is *planar*, meaning it will arrive at all sensors at the same angle. However, signals that are emitted close to the sensors follow the *near-field* model of propagation, in which the wavefront is curved. This means that the wavefront of the

signal will hit the microphones at different angles (and thus times) than would be expected for a planar wavefront. This will cause inaccuracies in time-delay estimates which will lead to inaccuracies in AoA estimates.

In addition to signal propagation assumptions, the accuracy to which the locations of sensor nodes (or co-located sensors) are known will affect how accurately AoA can be determined. Additionally, if sensor nodes or co-located sensors are placed too far apart, time-delay estimates may be inaccurate due to lack of coherence of the emitted signal between sensors (this is discussed in detail in Section 2.6.4). However, if the sensors are placed too closely together, a phenomena known as *spatial aliasing* occurs. This happens when the frequencies whose wavelengths are greater than the separation between sensors will be aliased to lower frequencies. According to (Ali et al. 2007), the highest frequency F_{max} that can be detected without aliasing is

$$F_{max} = v/(2x) \tag{2.3}$$

where x is the distance between sensors and v is signal propagation speed. This is analogous to the Nyquist rate, which dictates that signals of frequency f must be sampled at $2f$ to avoid aliasing to lower frequencies. The effects of spatial aliasing effectively limit the size of co-located sensor arrays, dependent on the frequencies of interest of emitted signals.

2.5.5 Summary

Section 2.5 has presented the various techniques of range and angle estimation used in self and source localisation in WSNs. Whilst there are many different ways to determine the distance and angle between target and observer, most approaches are limited by the accuracy of the time synchronisation required, and the ability to determine the start of the signal of interest.

2.6 Event detection

A significant contributor to the accuracy of a range or angle arrival estimatation technique is signal detection. For ToF estimation, determining exactly when a signal is sent and received will directly impact the range estimate. For AoA estimation, event detection indicates where the window of data which is being used starts. How accurately emitted signals can be detected by an observer is affected by how distinct the signal is against background noise and whether the observer has a prior knowledge of the signal that is being emitted. Obstacles in the environment which block the path between target and observer or cause interfering echoes can also cause difficulties, as well as atmospheric effects which affect the propagation speed of signals.

Signal detection approaches are typically based on correlation or transient energy. Correlation techniques rely on a prior knowledge of the signal to compare against: they are computationally complex, but provide accurate results. Energy based approaches require less processing (making them more suitable for real-time operation), but provide less accuracy when the signals being detected are not distinct against background noise. Section 2.6 discusses signal detection techniques used in the literature, as well as some of the factors which affect the accuracy of these detections.

2.6.1 Correlation-based event detection

The classical approach to signal detection in a channel is that of a *matched filter*, where the sampled signal is cross-correlated with a reference version of the signal. The output of the matched filter is the correlation of the two signals as a function of time, showing the points at which the sampled signal best matches the reference signal. The cross-correlation function must be analysed to determine where the where the signals best correlate: this should theoretically be the onset of the received/sampled signal. This can be used to determine the exact point in time (or samples) that the signal was received, relative to the time sampling started. If this time can be correlated with a sending time for the signal, then the ToA of the signal can be determined. A problem here is locating the point in the correlation function corresponding to the ToA of the direct path of the signal, because this may not be the strongest correlation peak (due to multi-path and reverberation). This problem is addressed by Girod (2005) by using accurate time synchronisation between devices to focus the search into the expected part of the signal based on when it was sent, thus increasing the likelihood that the local maxima of the correlation function corresponds to the global maximum. Flanagan (2007) uses the Fast Fourier Transform (FFT) of the correlation function to detect peaks. Alternatively, a peak-to-peak maximum approach is used by Peng et al. (2007).

The cross-correlation approach is computationally expensive, and requires two stages of computation (correlation, then peak detection). This means that the approach is not ideal for performing in real-time unless it is used with narrow band signals with a clearly-defined centre frequency, where the phase information can be directly manipulated (wideband and narrowband signals are further discussed later in Section 2.6). Because the cross-correlation approach requires a reference signal, it is not suitable when the events that must be detected are not controlled: this is likely the case when ranging and localisation is not co-operative.

2.6.2 Energy-based signal detection

Monitoring the energy of the signal in the channel in order to determine whether an event of interest has occurred or not is a computationally efficient approach to signal (or event) detection. Unlike cross-correlation, this approach is suitable for performing in real-time on many embedded devices, and is useful when it is not suitable to cross-correlate with a reference signal (for example, one does not exist).

A smoothed, windowed average or instantaneous measurement of the energy in the channel is constantly computed, and when this energy goes above a certain threshold, a detection is assumed. To make sure that these detections are not false positives, consistency checks can be applied such as checking the amount of time the signal spends above the threshold.

Transient acoustic signals (opposed to electromagnetic signals) with a large amount of energy at their onset are readily detected using energy based techniques. The classic example here is a gunshot, which produces a large, wide-band transient signal whose onset is easy to distinguish against background noise. This approach has also proved successful with both bird (Trifa et al. 2007) and animal calls (Ali et al. 2007), as well as gunshots (Ledeczi et al. 2005).

Some approaches use a static threshold on energy to compare against. This is not ideal for realistic situations as the energy will vary based on how far away a sensor is, and how loud the signal is. Instead, an adaptive approach to energy-based channel monitoring is more appropriate. The example drawn here

is by Trifa et al. (2007). In this approach, the noise in the channel is constantly estimated using two Exponentially Weighted Moving Average (EWMA) filters, for mean and variance, assuming the noise in the channel is Gaussian. Any significant deviations from the estimate of noise level (computed from a pre-determined threshold) are flagged as events of interest. To provide application specific filtering, only the energy from pre-determined frequency bands are used in estimating noise or events. The energy bands of interest are determined by the characteristics of the acoustic phenomena. The energy of the signal is determined in the frequency domain, by taking the magnitude of the sum of the pre-determined frequency bands of interest.

Several tuning parameters are provided to allow the event detector to be adapted to various types of signal:

- **Hysteresis period**: how long to wait until re-estimation of noise occurs
- **Initialisation period**: amount of samples to consider in order to initialise the noise estimate.
- **Adaption rate**: rate of decay of the EWMAs (for signals with quick or slow onset)
- **Frequency bands**: the frequency bands to filter and subsequently identify the channel energy of
- **Miscellaneous parameters**: FFT size, samples per window (energy calculations), sample rate

Several factors effect the accuracy of event detection techniques. These are related either to the environment in which the signals are being emitted, or how the signals are being emitted and detected. The rest of Section 2.6 discusses these issues.

2.6.3 Sampling frequency

In the time domain, the frequency at which an observer is sampling a channel for the presence of a signal provides a natural limit to the accuracy of ToA detection. The accuracy of estimates will be quantised to the limits of the sampling (Lanzisera et al. 2006). However, it is possible to make use of the *duality* of the time and frequency domains to interpolate signals, and thus process them with sub-sample accuracy. This approach is taken by Girod (2005) in the ranging algorithm for the Acoustic ENSBox. Interpolation is readily performed by padding a frequency domain signal with zeroes before converting it back to the time domain. Not only is this method useful but it is vital when the natural sampling rate is not high enough to provide the required accuracy of onset detection. This occurs when the signal propagation speed is high (such as the speed of light for electromagnetic signals).

2.6.4 Environmental effects

As discussed in Section 2.5.2 on page 26, in an environment with no obstacles (*free space*), the energy of a signal emitted by a target spreads out as it travels further from the source. However, this ideal model of propagation is affected by the physical environment in which the signal is emitted. Signals attenuate due to absorption from obstacles in the environment, or varying atmospheric conditions (such as relative humidity).

Reverberation

Obstacles in the environment reflect emitted signals, causing reverberation (echoes or multi-path) of the signal which interfere with the original signal. Reflected signals can have a constructive or destructive influence on the original signal, that is they may increase or decrease the strength of parts of the signal. Reflected signals may interfere with the original signal and cause ambiguity in detecting its ToA. This is because the strongest signal sensed may not correspond to the first (direct) path between observer and receiver. This also provides significant potential error for multiple false detections when using energy based approaches to event detection. This is a similar effect to non-line of sight (discussed below), although it is difficult to differentiate between the two. Indoor environments are more reverberant than outdoor environments, due to the amount of reflective surfaces present. The amount of energy that gets reflected when a signal hits a surface can vary from 10-90% dependent on the material (Chen et al. 2002*b*). In an outdoor open space, signals are free to propagate naturally, and grassy environments tend to absorb rather than reflect signals, reducing significantly any multi-path effects. Because the space is open, any reflections quickly lose energy over space (unlike confined, indoor spaces).

Non-Line of Sight

ToF ranging and AoA estimation rely on having line of sight between target and observer. This is because the actual distance is the direct path between the two. If the direct path is blocked, the signals the observer receives will be those which are the result of reflections, and thus took a longer path to arrive. If these are used to estimate distance, the will provide over-estimates. However, non-line of sight conditions are difficult to identify, as they will not affect the distribution of error in ranging estimates, just provide a bias (Girod & Estrin 2001). Detection of NLoS is not a trivial problem, and in localisation systems requires filtering at a level above data gathering in order to be resolved. For example, the self-localisation algorithm used by the Acoustic ENSBox (Girod 2005) attempts to detect NLoS by dropping range and angle estimates from nodes which are greater than 10% of the reverse path.

Temperature/humidity effects

The speed of sound increases as the square root of the absolute temperature. For acoustic waveforms, the combination of relative humidity (RH) and temperature affect the speed of sound, and thus a signal's wavelength. This means that if a certain level of RH and temperature is assumed in computation, signals may be out of phase. This can affect narrowband operations that operate directly on phase (Bohn 1988). The effects of RH and temperature in combination can make around an 8% difference to the speed of sound, when comparing 0% RH and temperature of 5°C with 100% RH and 40°C. RH also has an effect on the absorption of sound in air, which can cause signal attenuation (the magnitude of which is relative to the frequency of the signal). Compensation for these variables *may* reduce error in range estimation.

Signal coherence

Previously in Chapter 2, the spacing of sensors has been discussed for TDoA and AoA in both sensor nodes and co-located sensors on a single node. The limit on how closely sensors could be placed together (based on expected frequency of signals being detected) was discussed in Section 2.5.4 on page 29. When sensors are placed too far apart, the signals that they sample and detect may lack coherence, that is they

have become dissimilar due to propagation. Coherence describes how correlated two signals are with respect to their frequency components (Ash & Moses 2005). This will have has an effect on the quality of cross-correlations (for time lags) for AoA estimation of unknown sources, where a reference signal is not defined. In these cases, widely separated sensors may experience low correlation when determining TDoA. This in effect means that for signals which do not have an easily identifiable onset (in terms of energy), TDoA may be an unreliable way of estimating location. Experiments with different Pseudo-random noise (PN) chirps found that both signal coherence and signal power degrade with distance, but at different rates for different signal frequencies (Ash & Moses 2005).

2.6.5 Signal types and detection complexities

The type and bandwidth of a signal affects how easy it is to perform event detection. The bandwidth of a signal is measured by the ratio of its highest to lowest frequency components (Chen et al. 2002*b*). RF signals are considered narrowband because the bandwidths they operate in are generally narrow. For example, the ISM 2.4 Gigahertz (GHz) band goes from 2.4-2.4835 GHz, a ratio of 1.03. When signals are narrowband, their nominal wavelength is clearly defined, meaning relative time delays are easily computed using just phase information (Chen et al. 2002*b*). An exception in this case is Ultra-WideBand (UWB), a communication which uses pulses of sub-nanosecond duration, enabling generation of radio signals which are broadly spread in the frequency domain. They also have no centre or carrier frequency. This spread of frequency components can make the signal more resistant to the effects of multi-path, and can feasibly allow centimetre accuracy in ranging (Patwari et al. 2005).

In general, audio waveforms are considered wideband (bandwidth between 30 Hz—15 kHz, a ratio of 500), as well as seismic waveforms. This means that they do not have a characteristic nominal frequency, and thus relative phase cannot be manipulated as readily, requiring correlation techniques (Chen et al. 2002*b*). Therefore, the detection of wideband signals are more difficult than active ones because assumptions cannot be made about their specific phase. This is because an arbitrary characteristic signal may be complex, and difficult to detect, and the channel must always be monitored for its presence. On the other hand, actively emitted signals can be specially modulated, and their detection coordinated between emitter and receiver.

2.6.6 Resilience to noise

When the emitted signal can be chosen, techniques to improve a signal's resilience to noise can increase the operational range and accuracy of range and angle estimation. An important factor in choosing the type of signal is that it has good auto-correlation qualities. This increase resilience to interference. Reference signals can be created using Pseudo-random noise (PN) sequences (Ash & Moses 2005) from $-1, +1$, given that PN sequences have strong auto-correlation properties. The reference signal can then be modulated onto the channel (RF or acoustic for example) at the desired carrier frequency. Several authors have used PN sequences to generate reference signals with good auto-correlation properties (Ash & Moses 2005, Girod 2005). Experiments have used different centre frequencies, time durations and bandwidths for chirps and found that longer sequences provide smaller error in time delay estimation (Ash & Moses 2005).

Another approach to a reference signal with good auto-correlation properties is the choice of a linear

frequency chirp. With linear frequency chirps, the signal is transmitted at a constant amplitude, but increases in frequency over time (Peng et al. 2007, Flanagan 2007). The starting and ending frequencies can be arbitrarily band limited. With both PN and linear frequency chirps, Signal to Noise Ratio (SNR) is improved by increasing the length of the signal (Ash & Moses 2005).

When trying to detect or correlate wideband signals, it is convenient to assume a Gaussian noise component which is related to the signal. In Section 2.5.4, beamforming used summation to determine the AoA by finding the angle at which the signal energy was highest. The beamforming approach is normally used to increase the Signal to Noise Ratio (SNR). Similar approaches have been used to increase SNR of signals for ToF estimation in the presence of noise (Simon et al. 2004, Kwon et al. 2005).

2.6.7 Discussion

Section 2.5 discussed techniques for range and angle estimation, and up to this point Section 2.6 has discussed the challenges which are present in implementing any of these techniques. As has been described throughout this chapter, many ranging systems have been implemented for a variety of applications in WSNs.

In cooperative ranging and angle estimation algorithms, the choice is between the use of electromagnetic or acoustic signals. As has been previously discussed, the use of purely electromagnetic signals to determine distance is sensitive to error. In terms of implementation ease, acoustic signals are clearly more desirable for range and angle estimation.

Many ranging systems that have been implemented in the literature use ultrasonic acoustics to provide range and sometimes angle estimation (Priyantha et al. 2001). The choice of ultrasonic acoustics is often motivated by application requirements: ultrasound is not audible, and transducers can achieve high accuracy at short ranges (order of centimetres at 10-20 metres). Such examples tend to be demonstrated with indoor location/self-localisation systems, such as the ad-hoc localisation system AHLoS (Savarese & Rabaey 2001), the Active Bats localisation system (Ward et al. 1997) and the Cricket location support system (Priyantha et al. 2000). Whilst useful indoors, ultrasonic ranging has a limited operational range, and is readily blocked by obstructions (Girod 2005). As the centre frequency becomes higher, ultrasound signals become increasingly directional (unlike audible acoustic signals), although this is partly dependent on the transducer design (Priyantha et al. 2005).

Acoustic signals have been used on a variety of different WSN hardware in the literature, from the heavily constrained Mica2 platform (Girod & Estrin 2001, Whitehouse 2002, Zhang et al. 2007, Kwon et al. 2005, Simon et al. 2004) to the resource-rich Acoustic ENSBox platform (Girod et al. 2006). It has been noted that audible acoustic signals are suited to obstructed environments, as they are wideband, and thus provide a greater acoustic spectrum with which to perform coding techniques on (Girod 2005). Furthermore, the hardware requirements are minimal for acoustic ranging: all that is required is a microphone and speaker with which to range (in addition to a wireless radio for communication).

2.7 Self-localisation algorithms

Section 2.7 discusses self-localisation algorithms presented in the literature. As previously described in Figure 2.2 on page 21, localisation algorithms essentially have as inputs a combination of range and angle

data and prior positional knowledge (if any) and provide as output positions of the unknown targets. In self-localisation, the localisation problem is with regard to the nodes in the WSN estimating their own relative positions in an automated manner, meaning that the nodes are both targets and observers. Similarly with the general WSN design space discussion in Section 2.2, for localisation algorithms, there are many competing parameters in their design and evaluation space, which can be examined in an *application-centric*, *device-centric* or *network-centric context*.

In a *network-centric* view of the parameter space, desirable localisation algorithms would be easily distributed throughout the network to enable arbitrary scalability, for example. Examples of such algorithms can be found in theoretical works that have only been evaluated in simulation (Langendoen & Reijers 2003, Savarese & Rabaey 2001, Capkun et al. 2001). In these works, the effects of complex, realistic environments are often ignored, or treated naively.

A *device-centric* view of the parameter space limits the performance and constrains the parameters of the localisation algorithm to accommodate the limitations of the platform. Self-localisation solutions in this context exploits the given WSN platform and use its capabilities.

Examples of this can be seen in proof-of-concept, or emulated localisation systems (Savvides et al. 2001, Savvides et al. 2003, Whitehouse et al. 2004, Whitehouse & Culler 2006, Moore et al. 2004). In some cases, the physical sensor network is only used to gather data which is then further used within emulated systems. By and large, approaching localisation from a device-centric view limits the complex processing which can be accommodated as nodes cannot perform it. Processing is commonly pushed to the sink, which is not generally desirable (at least with respect to scalability) in the network-centric view.

An *application-centric* view of the parameter space considers the requirements of the application and deployment as the primary motivator for deriving the localisation solution. The foremost design goal is for the localisation mechanism to meet the needs of the application.

In the rest of Section 2.7, the state of the art with respect to self-localisation is discussed, with emphasis on the constraints that certain algorithms impose and the performance of real-life solutions. The approaches that show most promise for realistic self-localisation apply strong constraints to how input data is used in the localisation algorithm. The best examples are based around rigidity theory (Mautz et al. 2007, Moore et al. 2004, Priyantha et al. 2001). Real localisation scenarios must deal with potentially large, unpredictable and uncorrelated amounts of error in range and angle measurement. Ambiguities caused by Non-line of sight (NLOS) effects and reverberation can further complicate realistic self-localisation. In Section 2.7.1, one of the most-well known localisation algorithms, *lateration* is introduced. This is followed by the role of anchors (or reference positions), multi-hop localisation, anchor dependent localisation algorithms, distributing processing in localisation algorithms and range only localisation algorithms.

2.7.1 Lateration

Lateration is the canonical anchor-based self-localisation algorithm. Many self-localisation algorithms from the literature use lateration as a basis for localisation, hence will be discussed before any other localisation algorithms.

To estimate the position of a target in 2D, three anchors with corresponding distance estimates r_1, r_2

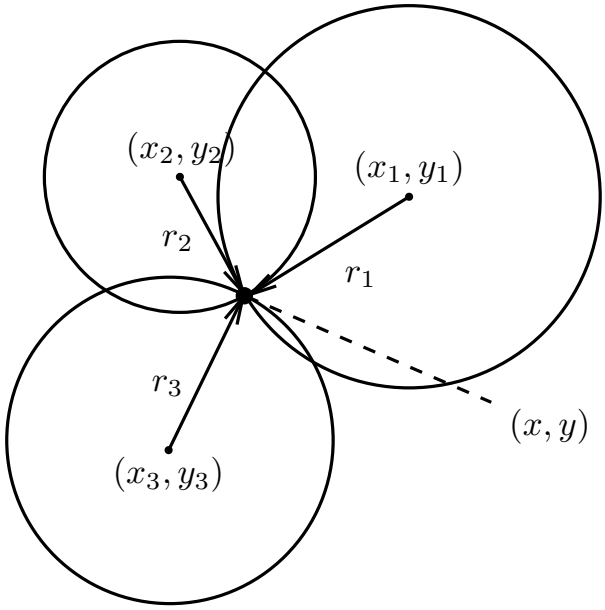


Figure 2.8: A graphical representation of lateration. The three measurements r_1,r_2 and r_3 made by nodes at (x_1, y_1), (x_2, y_2) and (x_3, y_3), intersect at the position (x, y) which denotes the position of the unknown target. Lateration is the process of estimating either the 2D or 3D position of an unknown target using several known reference points (at least three in 2D and four in 3D) and the estimated distances from these reference points to the target.

and r_3 are required. The target is located at the intersection of three circles with radii r_1, r_2 and r_3, as shown in Figure 2.8. To estimate the position of a target in 3D, four anchors with corresponding distance estimates are required. The target is now located at the intersection of three spheres.

Mathematically, the intersection problem is solved by linearising a system of distance equations and solving for two or three unknowns (the x, y and z coordinates) using the linear least squares approach. Assume a anchors, where the $i-th$ anchor's coordinates are x_i, y_i, z_i, and each anchor has a corresponding distance estimate r_i to an unknown target, with coordinates x, y, z. Also assume that $a > \nu$, where ν is the number of dimensions (or unknowns) that the position is being resolved in. From this, the following system of equations can be formed

$$\begin{aligned} r_1 &= \sqrt{(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2} \\ r_2 &= \sqrt{(x_2 - x)^2 + (y_2 - y)^2 + (z_1 - z)^2} \\ &\vdots \\ r_a &= \sqrt{(x_a - x)^2 + (y_a - y)^2 + (z_a - z)^2} \end{aligned} \tag{2.4}$$

These equations can be linearised and reordered, giving a system of linear equations of the form:

$$Ax = b \tag{2.5}$$

Where A, x and b are

$$A = 2\begin{bmatrix} (x_n - x_1) & (y_n - y_1) & (z_n - z_1) \\ (x_n - x_2) & (y_n - y_2) & (z_n - z_2) \\ \vdots & \vdots & \vdots \\ (x_n - x_{n-1}) & (y_n - y_{n-1}) & (z_n - z_{n-1}) \end{bmatrix} x = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$b = \begin{bmatrix} r_1^2 - r_n^2 - x_1^2 + x_n^2 - y_1^2 + y^2 + n - z_1^2 + z_n^2 \\ r_2^2 - r_n^2 - x_2^2 + x_n^2 - y_2^2 + y^2 + n - z_2^2 + z_n^2 \\ \vdots \\ r_{n-1}^2 - r_n^2 - x_{n-1}^2 + x_n^2 - y_{n-1}^2 + y^2 + n - z_{n-1}^2 + z_n^2 \end{bmatrix}$$

These equations can subsequently be solved using a standard least squares approach, giving $\hat{x}$ as the coordinates of the unknown:

$$\hat{x} = (A^T A)^{-1} A^T b \tag{2.6}$$

Where T is the matrix transpose, and A^{-1} is the pseudo-inverse of A. If $a > (u+1)$, then the system of equations is *overdetermined*. Using the least squares approach will find the solution which minimises the residual error. This approach will fail if there is no variation in one of the axes. For example, all anchors are co-linear in 2D, or planar in 3D. The above examples assume that all distance measurements between anchors and the unknown target contain no error—this is not an entirely realistic scenario, as distance estimates will always contain some amount of error. The magnitude of final position error will be a combination of the magnitude of the individual distance errors and the geometry of the anchor nodes (Priyantha et al. 2005).

Geometric Dilution of Precision

The contribution of the geometry of the anchor nodes to the error in a position estimate is known as the GDoP. A GDoP metric is used in GPS systems to describe the geometric *strength* of GPS satellites current positions with respect to the target. The biggest effect of error due to GDoP occurs when positions of anchor nodes are geographically close together, such that the variation of their angles to the unknown target is small.

The computation for GDoP is as follows (Dana 1994): Assuming the range identity for r as in

Equation 2.4, then matrix A is formed by

$$A = \begin{bmatrix} \frac{(x_1-x)}{r_1} & \frac{(y_1-y)}{r_1} & \frac{(z_1-z)}{r_1} & -1 \\ \frac{(x_2-x)}{r_2} & \frac{(y_2-y)}{r_2} & \frac{(z_2-z)}{r_2} & -1 \\ \frac{(x_3-x)}{r_3} & \frac{(y_3-y)}{r_3} & \frac{(z_3-z)}{r_3} & -1 \\ \frac{(x_4-x)}{r_4} & \frac{(y_4-y)}{r_4} & \frac{(z_4-z)}{r_4} & -1 \end{bmatrix} \tag{2.7}$$

and the matrix Q is given by

$$Q = \left(A^T A\right)^{-1} \tag{2.8}$$

The GDoP is given by

$$GDoP = \sqrt{Q(0,0) + Q(1,1) + Q(2,2) + Q(3,3)} \tag{2.9}$$

This GDoP calculation can be used in conjunction with lateration for example, to understand the quality of the reference positions with respect to the target in the self-localisation algorithm. GDoP is considered with respect to lateration in Chapter 3 in detail.

Map-stitching for distributed computation

Distributing a localisation algorithm throughout a network is a desirable feature for scalable wireless sensor networks. A common approach for this is *map stitching*. In map stitching, local groups of localised nodes are stitched together to make a map of the overall network. The stitching is performed by finding the set of nodes that two clusters have in common and computes a rotation, translation and reflection which best aligns them (Shang & Ruml 2004, Moore et al. 2004, Capkun et al. 2001). The problem with map stitching is that position error in one group will propagate through other groups, and throughout the network, necessitating a global refinement stage (Whitehouse et al. 2004, Kwon & Song 2008). Most algorithms have only been proposed in principle and evaluated in simulation (Shang & Ruml 2004, Capkun et al. 2001). In practical evaluations (Moore et al. 2004, Kwon et al. 2005, Whitehouse et al. 2004), the refinement stage is often performed at a central location due to local processing constraints of the sensing platforms used.

The rest of Section 2.7 examines the different approaches taken in the literature with respect to self-localisation, starting with anchor-based localisation (both single and multi-hop), followed by self-localisation algorithms which are not anchor-dependent.

2.7.2 Single-hop, anchor-based self-localisation

A large sub-class of localisation algorithms in the literature are anchor-dependent, meaning they require a presence of known positions in the network in order for any nodes to be localised. The *GPS-Less* localisation algorithm (Bulusu et al. 2000) assumes that an arrangement of anchor nodes with overlapping communication regions is present in the WSN. Perfect spherical radio communication and identical transmission range for every sensor node are also assumed. Each anchor node constantly broadcasts its position, and sensor nodes listen for these broadcasts, calculating their positions as the centroid of all of the anchor nodes it has connectivity with. A proof-of-concept implementation using the localisation algorithm was demonstrated to have an accuracy on the order of metres (1.83m average error), but re-

quired a high density of anchors, and was not suitable for indoor operation. In the *APIT* localisation algorithm (He et al. 2003), each node estimates its position by placing itself at the centroid of all anchors it can communicate with. Signal strength is used to determine whether a node is nearer or further from one anchor or another. As with GPS-Less, this approach requires a large density of anchor nodes, however this approach was only evaluated in principle.

2.7.3 Multi-hop, anchor-based self-localisation

A quirk of theoretical WSN-based self-localisation research is the multi-hop localisation algorithm, or *topological algorithm* (Mautz et al. 2007). This particular problem is entirely network and device-centric in its origins and subsequent evaluation. The generic problem scenario is as follows: where a network of sensor nodes is deployed, some percentage of which either already know their position a priori (via on-board GPS, or manual surveying, for example). Other nodes in the network do not know their position, but have some way of estimating distance between one another. Nodes aim to try and perform lateration with anchor nodes to determine their position, however, they may not be in direct ranging distance with anchors. Therefore, nodes must estimate their distance to at least three anchors (for 2D) in order to determine their position. These algorithms are intended to be distributed (each node collaborates with other nodes to determine position).

The **Ad-hoc Positioning System (APS)** (Niculescu & Nath 2001) is probably the most indicative example of the multi-hop WSN localisation algorithm. Nodes estimate their distance to three anchors, then use these measurements to determine their position through lateration. If a node does not have a direct ranging link with an anchor, it must estimate the distance using the shortest path. Other well-known examples are **N-hop multilateration** (Savvides et al. 2001, Savvides et al. 2003), **Robust positioning** (Savarese et al. 2002) and **Amorphous computing** (Nagpal et al. 2003). Each have different techniques to bound error induced by shortest path estimation, and N-hop multilateration uses a *bounding box* computation to provide position estimates.

A problem with multi-hop localisation algorithms is that they are by definition coarse grained. When nodes try to infer distance from a shortest path metric, the accuracy of these measurements are going to be compromised at best, and hopelessly inaccurate at worst. Whitehouse (Whitehouse & Culler 2006) compares the performance of APS using simulated data to its performance using empirically observed ultrasonic ranging data. It is found that the median localisation error is nearly 3m in deployments spanning a 18m grid.

A class of self-localisation algorithms attempt to localise a network by explicitly using only the range measurements between nodes, rather than using anchors. In these algorithms, anchor positions are only used to align the final results to a known coordinate system. These algorithms are either multi-dimensional scaling based (Shang et al. 2003, Shang & Ruml 2004, Ji & Zha 2004, Kwon et al. 2005), geometric consistency-based (Moore et al. 2004, Priyantha et al. 2003, Mautz et al. 2007), or refinement based (Broxton 2005, Gotsman & Koren 2004, Priyantha et al. 2003). The rest of Section 2.7 discusses range and angle based self-localisation algorithms that are not anchor-dependent.

2.7.4 Multi-Dimensional Scaling based self-localisation

Several algorithms use a statistical method known as Multi-Dimensional Scaling (MDS) as the basis for self-localisation algorithms (Shang et al. 2003, Shang & Ruml 2004, Ji & Zha 2004, Kwon et al. 2005). MDS is used to visualise the dissimilarity of data in multi-dimensional data sets. Given an input dissimilarity matrix of data specifying the estimated distances between *all* nodes in the network (these are the dissimilarities), MDS computes the relative coordinates of all nodes. Anchors can then be used to translate the relative coordinate system into a global coordinate system (Shang et al. 2003), (Shang & Ruml 2004).

The requirement that distances between all nodes is a problem, as in practise not all nodes will be able to measure distance between one another. In MDS-MAP (Shang et al. 2003), a shortest-path approach is used to approximate distances between nodes that did not make distance estimates. As with topological approaches, this means networks with irregular topologies will have shortest paths that are highly inaccurate in relation to the actual distance. To address this, an called Least Squares Scaling (LSS) which tolerates a lack of pair-wise sensor measurements was proposed (Kwon et al. 2005). A distributed version of the algorithm called MDS-MAP(P) has also been suggested (Shang & Ruml 2004). In this algorithm, each sensor node makes a local map including itself and its neighbours and applies the MDS technique to this map. These positions are then merged together in order to create a global map.

In general, MDS approaches have only been used in simulation studies. There are two notable examples of real-life evaluation of MDS based approaches (Kwon et al. 2005, Whitehouse et al. 2004). LSS (Kwon et al. 2005) was tested in a realistic, controlled experiment, where range data was gathered using an acoustic ranging mechanism on Mica2 motes. Forty-five nodes were deployed in a uniform manner on a 60m by 60m grid. The algorithm yielded an average position error of 2.469m.

2.7.5 Graph theoretic and geometric based self-localisation

Graph theoretic approaches to self-localisation pose the problem as one of embedding vertices (nodes) into a graph, based on the weights of the edges (distance estimates). The solution (in the graph theoretic sense) to the self-localisation problem is to find a unique embedding of vertices in either two or three dimensions that matches the edges (distance estimates). In order for a unique realisation of a graph to be found, it must be globally rigid. Global rigidity ensures that if any of the vertices moves, then the distances between them would not be the same (and thus is not valid for that set of measurements).

A robust quadrilateral is the smallest structure that can be globally rigid in two-dimensions. It is a structure with four vertices (nodes) that has no ambiguity in terms of the relative positions of each vertex (Moore et al. 2004). This is achieved with a minimum of six range estimates between the four different nodes. A robust quintilateral is the smallest structure that can be globally rigid in three dimensions (Mautz et al. 2007). It is made of five vertices, with at least ten different measurements between nodes to ensure global rigidity.

Global rigidity properties alone are not enough to guarantee correct localisation. This is because noisy measurements can cause ambiguities which yield an embedding of vertices which is incorrect when compared to the ground truth, but is accurate when compared to the distance estimates that are given as input data. These cause *flip* ambiguities, where the position of a node can be flipped from where it

should be in the structure.

Different algorithms provide different ways of dealing with the ambiguity caused by noise in measurement in both 2D and 3D. Both 2D and 3D versions of geometric rigidity-based algorithms perform tests to estimate the rigidity of a structure before localising a node. The approach used by Moore et al. (2004) involves dividing a quadrilateral into four triangles, and testing whether the smallest angle is above a certain threshold. If it is not, this implies there may be flip ambiguities causing the position estimate to be incorrect. If all of the triangles are robust, the quadrilateral is also robust. For 3D, Mautz et al. (2007) proposes a volume test based on barycentric (rather than geometric) coordinates.

Both 2D and 3D algorithms follow a similar approach: local neighbourhoods of nodes form clusters of robust quadrilaterals or quintilaterals based on their range measurements to one another, which can then be stitched together using local map stitching to form a global map (as discussed in Section 2.7.1 on page 39). Unlocalised nodes can be iteratively added to rigid clusters if they can meet the rigidity constraints.

Geometric-based approaches represent the most rigorous way to perform self-localisation: to avoid ambiguity, structures must be rigid, which can only be guaranteed through density of measurements and these measurements must be low noise (10% or less). The problem with these geometric approaches is that to get the best localisation accuracy, nodes are required to have a large number of neighbours, which is not always the case in sparsely populated networks. In order to obtain 100% localisation accuracy, it is noted that nodes require a degree of 10.

In removing anchors from the self-localisation problem, difficulties arise in removing ambiguity due to measurement noise. This is an important point to consider with respect to anchor-free localisation.

2.7.6 Refinement based self-localisation

Mass-spring optimisation approaches can be used to refine the results of self-localisation. In this approach, a network of sensor nodes are vertices (or masses), which are connected by springs (or edges). The springs have *tension* which exert force on the masses. *Tension* is a metric derived from the estimated distance between two vertices and the current euclidean distance between them (that is, the current estimate for the position). A metric for the overall tension on the network is also computed. The aim of the optimisation is to minimise the overall tension on the network by changing positions of nodes so their estimated distances best match the distances based on their current euclidean positions.

Like other non-linear optimisations, mass-spring optimisation is subject to false minima, where the optimisation criteria is met, but at a local minima of the function. Several self-localisation algorithms provide different techniques to determine an initial guess to a mass-spring optimisation. The Anchor Free Localisation (AFL) algorithm (Priyantha et al. 2003) uses a connectivity algorithm to produce an initial guess. The five nodes which best create a connected graph are chosen. The shortest hop-count from each of these nodes is used to compute initial coordinates for all other nodes in the network. A graph drawing technique called Spectral Graph Drawing (SGD) has also been used to provide an initial guess (Broxton 2005, Gotsman & Koren 2004). Is is noted that this approach performs better than AFL for providing an initial guess. One limitation of SGD is that it has only been demonstrated in 2D.

2.7.7 Range and angle-based self-localisation

The multilateration algorithm which provides self-localisation in the Acoustic ENSBox system uses both ranges and angles to estimate node locations (Girod 2005). The algorithm is carried out in a pseudo-centralised manner—after initial range and angle estimation, all nodes report ranges to one arbitrarily elected leader, who performs the localisation computation. The Acoustic ENSBox devices can estimate both ToF ranges and AoA using acoustic methods. However, to support this the nodes have four microphones, which means that each node's relative orientation must be estimated as well as its relative position. However, using angles in combination with ranges allows the initial positions of nodes to be guessed in 2D or 3D more readily: locations can be initially determined in a polar coordinate co-ordinate system, relative to a specific node (which is usually the node that has the most range and angle measurements with its neighbours, hence is most connected). These polar coordinates can be readily transformed to Euclidean coordinates.

After initially estimating node orientations and positions, the self-localisation algorithm optimises the positions using an iterative, interleaved, Non-linear least squares (NLLS) algorithm. The algorithm takes an interleaved approach to estimation of position and orientation. Orientations are fixed, and used as part of the constraints in building a linearised system of equations to estimate the relative positions of nodes which is solved for each iteration of the algorithm. In between each NLLS iteration, node orientations are re-estimated by averaging the error between observed DoA and angle based on the NLLS result. The localisation algorithm is considered finished when residual error for different aspects (yaw, pitch, roll, range) falls below an empirically determined threshold. This can mean that under constrained systems do not converge.

The NLLS self-localisation algorithm works best when its system of equations is over-constrained (that is, there are many range and angle measurements per node). This means that erroneous measurements can be removed at certain points during the position estimation process through outlier rejection procedures. These rejections are based on heuristics such as residual error between two nodes' range estimates and residual error between estimated position and estimated range.

The 3D self-localisation algorithm described here is evaluated in realistic conditions as part of Chapter 3, using the version 2 iteration of the Acoustic ENSBox platform.

2.7.8 Discussion

Section 2.7 has given an overview of self-localisation algorithms, including insight into the design space viewpoints from which the algorithms were designed. Whilst the specific localisation algorithm chosen by a designer has some bearing on the performance of the localisation accuracy, it is clear that for accurate self-localisation, the quality and density of input data has the greatest effect on the resulting performance of localisation. The first algorithms surveyed were single and multi-hop anchor-based algorithms. These approaches approximate distance between node and anchor (typically using a shortest path approach) if the distance cannot be directly estimated. This is an approach that can induce large amounts of error, hence cannot yield high accuracy in general.

Another related assumption with anchor-based algorithms it that anchors do not have any positional error associate with them. This is not always the case: GPS estimates can be inaccurate and human

error can induce manual measurement. However, anchors provide a benefit to anchor-free algorithms in aligning relative coordinate systems to actual geographic coordinates (if required). The addition of angles to ranges in anchor-free localisation can provide a benefit, as shown by the Acoustic ENSBox localisation algorithm, but this comes at the cost of the extra hardware required to gather DoA estimates. In general, to ensure greatest accuracy in a localisation algorithm, it is important to gather quality ranging (and angle) data, and be judicious in outlier filtering and geometric construction. Whilst geometric approaches are strict on structural rigidity for node localisation, they can provide some guarantees about the quality of the localisation that is being performed. However, the density at which measurements need to be made may make range-only geometric approaches difficult for some systems.

Section 2.8 concentrates on algorithms which are specific to the source-localisation problem.

2.8 Source localisation algorithms

The main difference between self and source localisation is in the gathering of input data to submit to the localisation algorithm. In self-localisation, it is assumed that the range and angle measurement data is gathered in a coordinated, cooperative manner, where nodes can explicitly notify their neighbours that they are about to emit ranging signals.

Conversely, in source localisation, the target does not explicitly identify itself, and events must be gathered opportunistically by continuous event detection. As a result, deciding which observations made by nodes across the network correspond to a given event from a given target can be a difficult problem, especially given the issues discussed in Section 2.6 such as coherence, reverberation and NLoS. Moreover, at each node, it is sometimes difficult to determine exactly when an event occurs, as discussed in Section 2.6. Therefore, some source localisation approaches consider the grouping of event data to be a part of the localisation process.

Some of the self-localisation algorithms described in Section 2.7 (such as lateration and the MDS algorithms) are suitable to use directly for source localisation. However, Section 2.8 discusses two techniques developed explicitly for source localisation: one based on AoA and one based on ToA/TDoA.

2.8.1 TDoA based source localisation

As previously mentioned in Section 2.6, localisation using TDoA or ToA data for complex wideband signals is complicated by coherency across a network, as well as GDoP (which also affects angle-based approaches). Balogh et al. (2005) propose two algorithms based on ToA and TDoA estimation for localisation of snipers based on gunshots. The first algorithm is based on ToA estimates made by nodes of a gunshot, and is generally applicable to ToA localisation. The second is specific to gunshots, using the trajectory of the bullet passing through the network. Because of it general appeal, only the first localisation algorithm is described here. Assuming a network of n nodes the ToA is used to estimate the time of emission $t^i(x, y, z)$ of the signal of interest, using the following relationship:

$$t^i(x, y, z) = t_i - \frac{\sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}}{v_s} \tag{2.10}$$

where v_s is the speed of sound, t_i is the time node i detected the given event and the assumed position of the source/target is (x, y, z). Measurements from all nodes that have the same value within

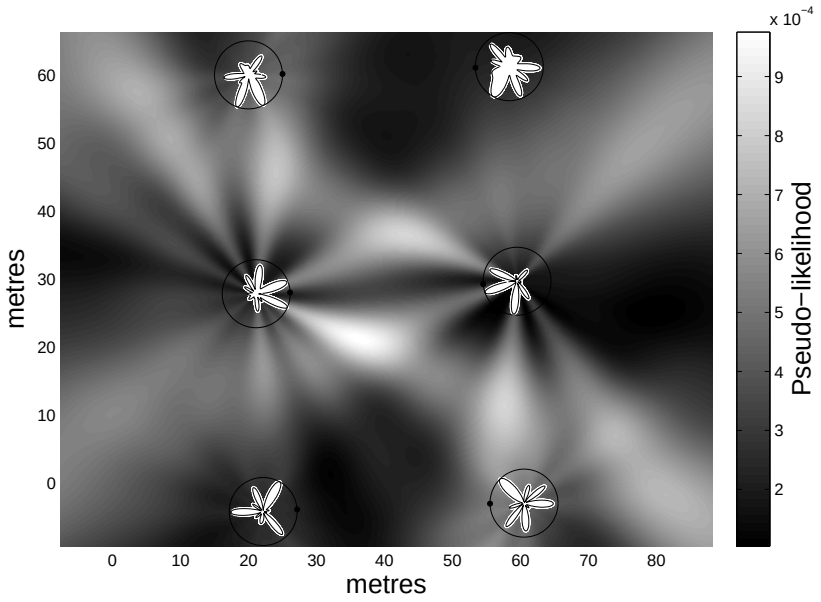


Figure 2.9: A pseudo-likelihood map, which shows the event space as a 2D grid. Each point in the grid is assigned a likelihood value based on the fusion of individual nodes' AML AoA vectors. The lighter the shade, the more likely the sound source came from the area. There were six nodes contributing to this localisation estimate, and each individual likelihood vector is plotted on this graph to aid understanding.

some *uncertainty* u for the time of emission are considered *consistent* with one another. The location of the source is found by calculating the maximum of the consistency function, defined as:

$$C_\tau(x, y, z) = \max K(x, y, z, t, \tau) \tag{2.11}$$

where $K(x, y, z, t, u)$ is the number of consistent measurements for a given position (x, y, z) with given time t and uncertainty u. This requires the function space to be searched to find the global maxima relating to the position. The authors note that exhaustively searching the event space is too time consuming, and instead use a Generalised Bisection method (Hu et al. 2002), which provides in around one second on average.

2.8.2 Pseudo Maximum Likelihood (ML) source localisation

The pseudo-maximum likelihood approach presented in (Ali et al. 2007) combines the likelihood vectors produced from individual AoA ML estimates made by sensor nodes in order to find the most likely position in a pre-defined *event space* of a given resolution. The algorithm is presented in 2D, as the AoA measurements used in its evaluation were only azimuthal. Therefore, the event space is a 2D grid, aligned to the coordinate system of the nodes in the WSN.

For each grid point in the search space, the relative angle between that point and each node in the network is determined (based on the node's position and relative orientation). This yields a relative angle

from 0-360 for each node relative to each point. This relative angle is used to determine which of the node's likelihood values project onto that point in the event space. For example, assuming the relative angle between a point in the search space and a node is 39°. Then the 39th element in the likelihood vector (i.e. the likelihood value at 39°) is used from that node to project onto the given point in the event space. For each point in the event space, the likelihood values from each node are summed to produce the pseudo-likelihood value for that point. The position estimate is the point in the search space which has the highest likelihood value. Assuming a network of n nodes, each with position and orientation (x, y, θ) and ML AoA vector $J = \{l_1..l_{360}\}$, the source localisation algorithm is as follows:

- **Define resolution and area of event space**.
  Define the event space as a padded square around the known positions of the nodes having side length C (the largest distance between all nodes in cm). Scale the resolution C_r of the search space so that each point corresponds to either 1 or 0.5m ($C_r = C/100$ or $C_r = C/200$).

- **Scale AoA vector**
  Scale each node's AoA vector J_i to between 0 and 1

$$J_i^s = \frac{J_i}{\sum_{k=1}^{n} J_k} \tag{2.12}$$

  where J_i^s is the scaled vector.

- **Compute likelihood**
  Calculate the relative angle $\theta_{a,i}$ between each point $s_r^{x,y}$ on the grid and each node's (x_i, y_i) position and orientation θ_i. The likelihood value for node i is given by indexing into J_i^s, so that $l_i = J_i^s(\theta_{a,i})$. The likelihood value for each point is given by adding together the likelihood values contributed by each node.

  The position estimate is then determined as the point with the largest likelihood in the search space.

This algorithm effectively is an exhaustive search of the event space to find the point with maximum likelihood. The visualisation of the likelihood across the event space is shown in Figure 2.9. Unlike the TDoA consistency based approach above, this approach does not explicitly consider how the events gathered are grouped together. This is because the TDoA consistency function is based on detection times, but the AoA algorithm is based on processing that is performed *after* the detection. This approach is suitable when the detection times cannot be estimated accurately, due to low SNR.

2.8.3 Summary

The performance of TDoA for source localisation is highly dependent on how accurately the start of an event can be determined not only at one node, but across the whole network. The issues of coherence, reverberation, multi-path and NLoS cause great problems for TDoA techniques in general. However, Balogh et al. (2005) were able to take advantage of the signal characteristics of gunshots to enable high accuracy. For signals which do not have a wideband, high energy onset, TDoA determination will not be accurate.

The TDoA and AoA approaches imply different levels of complexity. For a sensor node, it is easier to determine the onset of a signal based on its ToA than to estimate the AoA, because the ToA requires only one microphone (for example), and the AoA requires three co-located microphones for 2D (θ) and four for 3D (θ,ϕ).

2.9 Evaluation of Localisation performance

Evaluating the performance of a localisation algorithm is important when comparing against other algorithms, or when choosing an existing algorithm to best fit the requirements of a particular WSN application. However, there is a lack of unification in the WSN field in terms of localisation algorithm evaluation and comparison. In addition, no standard approach exists to take an algorithm through modelling, simulation and emulation stages and into real deployment. As a result there is not a specific set of criteria which quantify one algorithm as being better than another. Moreover, deciding what performance criteria localisation algorithms are to be compared or evaluated against is important for the success of the resulting implementation given that different applications will have differing needs.

Since localisation algorithms are expected to be used in real applications, it is not conclusive to verify their performance in simulation only. Localisation algorithms should be emulated (on test beds or with empirical data sets) and subsequently implemented in hardware, in a realistic WSN deployment environment, as a complete test of their performance.

Section 2.9 provides a contribution to self-localisation by proposing and an approach to evaluation for self-localisation algorithms based on simulation, emulation and in-situ deployment.

Performance evaluation metrics are discussed in the context of three important criteria: localisation accuracy, cost, and coverage. Because WSNs are typically constrained in terms of node/network lifetime and per-node computational resources, addressing these constraints leads to trade-offs in the performance of localisation algorithms. For example, if maximising localisation accuracy is the foremost priority, specific hardware may have to be added to each sensor node, increasing node size, cost and weight. Conversely, if the hardware available is already determined, then the application expectations with respect to performance criteria (such as accuracy) must be adjusted accordingly.

2.9.1 Evaluation Criteria

The intuitive measure of localisation algorithm performance is accuracy: how accurately can the algorithm estimate the positions of nodes compared to some known ground truth (and according to the granularity required by the WSN application)? However, localisation algorithms are also subject to the general constraints of wireless networked sensing. Therefore, a broader set of evaluation criteria for localisation algorithms are needed, including: accuracy, cost, coverage, robustness and scalability. These criteria reflect the constraints imposed on WSNs: computational limitation, power constraints, unit cost and network scalability requirements. The criteria form a design space similar to that presented in Section 2.2 on page 12 for general WSN applications and deployments and are described below in more detail.

Scalability

A centralised localisation algorithm will typically aggregate all input data at a central, more capable sink to carry out processing; this represents a single point of error, and a potential bottleneck for network

communication. In contrast, a distributed localisation algorithms execution is shared throughout the network with no reliance on a central sink. However, centralised algorithms are conceptually easier to implement in cases where it is known that the network will be small and will not increase. By comparison, distributed algorithms are more complex to develop and deploy, but may be advantageous for the application if the network does not have a logical topology (i.e. a tree of nodes sending data to a sink), and will need to support a large number of nodes (tens to hundreds). Theoretically, scalability is an important general consideration. However, for a specific deployment it is likely that only a small number of nodes would be deployed, so this is not necessarily an overriding realistic concern.

Accuracy

One may think that positional accuracy compared to ground truth is the over-riding goal of a good localisation algorithm. On reflection, this is largely application-dependent—different WSN applications will have different requirements on the resolution of the accuracy. Consider a tracking application—the estimated positions of nodes in the network directly affect the accuracy of the tracking. The granularity of the required accuracy may be a ratio of the inter-node spacing. For example, if the average node spacing is 100m, up to 1m error may be acceptable. However, if the average node spacing is 0.5m, the same error level is clearly unacceptable.

Resilience to error and noise

It is important to understand how well the localisation algorithm will perform without an accurate or full set of input data. Evaluation should show how measurement noise, bias or uncorrelated error in the input data affects the algorithms performance.

Coverage

Some algorithms may have problems localising the whole network if nodes do not have enough neighbours (*enough* is specific to the details of the algorithm) in terms of connectivity or distance constraints/estimates. Coverage may relate to the physical network density: one may be more likely to get 100% localisation coverage in a densely deployed network.

Cost

An algorithm which can minimise several cost constraints (such as power consumption, communication cost or specialised hardware cost) is likely to be desirable if maximising network lifetime is a primary deployment goal. For example, an algorithm may focus on minimising communication and complex processing to reduce battery usage, but at the expense of the overall accuracy.

Discussion

The perfect localisation algorithm would provide suitably accurate results (relative to the scale requirements of the application), in a decentralised manner, with low communication and processing overhead, whilst allowing incremental addition of nodes, and requiring zero anchor nodes (except for relating to a global coordinate system). Deployment practise and expertise indicate that trade-offs are best resolved when intimately related to the specifics of the class of applications that the particular WSN is deployed to address. The quantitative measurements required to understand these trade-offs are described in

Section 2.9.2.

In order to address quantitatively a localisation algorithm's performance against the criteria described in the Section 2.9.1, a set of metrics are available. In Section 2.9.2, metrics or common measures for accuracy, cost and coverage are described.

2.9.2 Accuracy metrics

The aim of a localisation accuracy metric is to show how closely matched the ground truth and estimated positions are. Accuracy is likely to be related to measurement noise, bias, accuracy and precision in the input data provided to the localisation algorithm (as discussed in Section 2.5).

Globally, the positions determined by a localisation algorithm represent a geometrical layout of the physical positions of the sensors. This layout must be compared to the ground truth, or known layout of the sensors. It is important therefore that not only the error between the estimated and real position of each node is minimised, but also that the geometric layout determined by the extent that the algorithm matches the original geometric layout.

Mean absolute error

One way to describe localisation performance is to determine the residual error between the estimated and actual node positions for every node in the network, sum them and average the result. This can be calculated using the Mean Absolute Error (MAE) metric (Broxton 2005), which, for each of n nodes in the network, calculates the residual between the node's estimated $(\hat{x}, \hat{y}, \hat{z})$ and actual (x, y, z) coordinates

$$MAE = \frac{\sum_{i=1}^{n} \sqrt{(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 + (z_i - \hat{z_i})^2}}{n} \tag{2.13}$$

The resulting metric represents the average positional error in the network, aggregating individual residual errors into one statistic. MAE is similar to Root Mean Square (RMS) error, a commonly used calculation to measure the difference (or residual) between predicted and observed values. It has also been noted that whilst the mean absolute error is important in some cases, it is also beneficial to know the Maxium Error (MaxE) exhibited in the position estimation (Slijepcevic et al. 2002)

$$MaxE = \max_{1..n} \sqrt{(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 + (z_i - \hat{z_i})^2} \tag{2.14}$$

GER

It is important for an accuracy metric to reflect not only the positional error in terms of distance, but also in terms of the geometry of the localisation result. It is entirely possible that for a given localisation result the average error metric is low, but the layout created by the algorithm does not correlate highly with the physical layout of the network. This problem was addressed by defining the Global Energy Ratio (GER) metric (Priyantha et al. 2001)

$$GER = \frac{1}{n(n-1)/2} \sqrt{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\frac{\hat{x_{ij}} - x_{ij}}{x_{ij}} \right)^2} \tag{2.15}$$

where the distance error between nodes is normalised by the known distance between the two nodes (x_{ij}), making the error a percentage of the known distance.

BAR

The Boundary Alignment Ratio (BAR) metric (Efrat et al. 2006) is a measure of how closely the estimated positions of nodes that sit on the boundary of the localised network match the actual positions. It is in essence the sum-of-squares normalised error taken from matching the estimated boundary with the actual boundary. BAR is used as the minimisation metric for the Iterative Closest Points (ICP) algorithm (Zhang 1994) which matches estimated and actual boundary points. When the change in the BAR metric is negligible, the best alignment possible has been determined. A similar technique is used to compare the shape of a localised network, irrespective of translation, scale and rotation for the Acoustic ENSBox localisation algorithm (Girod 2005). A four-step approach influenced by the Procrustes method of characterising shape is defined and used to measure estimation fit with ground truth (Kendall 1989).

This is done by establishing a scaling factor between the real and estimated maps. The maps are then translated and scaled relative to the origin, which is defined as the node closest to the centroid of the estimated topology. The estimated topology is then rotated according the angular offsets between nodes, and finally translated by the average distance between estimated and ground truth points. Average error can now be taken using any of the metrics that have been previously described (MAE, GER). Whilst both approaches take into account the shape of the network, BAR (Efrat et al. 2006) uses only a subset of the nodes on the boundary to contribute towards the computation compared to all nodes in the network (Girod 2005).

The previous accuracy metrics rely on prior knowledge of the actual node position and physical network topology in order to evaluate the localisation quality and error. In realistic WSN deployments, this information is not known, and so measurement of error should be determined relative to the available information. Toward this aim, an average distance error metric is proposed (Girod 2005), termed the Mean Range Residual (MRR) by the author

$$MRR = \frac{2}{n(n-1)} \sum_{i,j,i<j}^{n} \hat{x}_{ij} - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{2.16}$$

where the estimated distance between two nodes i and j is subtracted from the observed range between them.

2.9.3 Cost metrics

Cost is an important trade-off against accuracy, and is often motivated by realistic application requirements (this is discussed in more detail in Section 2.9.5 on page 52). As such, cost metrics are typically used to evaluate the trade-offs that are not addressed by positional error and coverage. Several common metrics are described below, along with how they may be determined.

Anchor to node ratio

Minimising the number of anchors in the network is desirable from an equipment (cost, power usage) or deployment point of view. For example, using anchors that can estimate position through the Global

Positioning System (GPS) will require extra hardware which is both expensive and power-hungry, potentially limiting the node lifetime. The anchor to node ratio is n/a, where n is the number of nodes in the network and a is the number of anchors. This metric should be used to investigate the trade-off between accuracy and increasing number of anchors used in the localisation algorithm evaluation.

Communication overhead and power consumption

Assuming that radio communication is a large consumer of power relative to the overall consumption of a wireless sensor node, minimising communication overhead is paramount in maximising the potential network lifetime. For example the average number of packets sent per node has been used as an evaluation criteria between algorithms (Langendoen & Reijers 2003). It should be noted that radio communication is not always the biggest consumer of power on a node, especially when micro-controller platforms perform intensive tasks (McIntire 2003).

The proportion of available power that a node spends on localisation can affect its lifetime (and thus the network lifetime). Power consumption will be a combination of the power used to perform local operations and the power used to send and receive messages associated with localisation.

Algorithmic complexity

Standard notions of computational complexity in time and space (big O notation) are naturally useful as comparison metrics for the relative cost of localisation algorithms. For example, as a network increases in size, a localisation algorithm with $O(n^3)$ complexity is going to take a longer time to converge than an $O(n^2)$ algorithm. The same is true for space complexity—algorithms which require less memory as they scale may be preferable. This may motivate a trade-off between centralised and decentralised algorithms.

Convergence time

Measuring the time taken for both initial measurement gathering and localisation algorithm convergence can both provide important comparison metrics. A network that takes a long time to localise may not be as useful if the application requires rapid self-localisation for immediate processing related to node positions, such as for the tracking of a moving target. If the localisation algorithm is based on non-linear optimisation, there may also be a trade-off to be made between accuracy and convergence time—the extra time taken and energy expended to get a slightly more accurate solution may not be beneficial.

2.9.4 Coverage metrics

Some self-localisation algorithms may not be able to localise all of the targets in the network because of a requirement on node density or rigidity (as discussed in Section 2.7.5 on page 41). Coverage is a measure of the percentage of nodes in the deployed network that can be successfully localised, regardless of the localisation accuracy (which is described by previous metrics). However, density of deployment, as well as placement of anchors can have effects on coverage results for different localisation algorithms. The effects and their evaluation and are discussed in the following subsections.

Density

Density is measured by the average number of neighbours a node has, as in AFL (Priyantha et al. 2003) and Robust Localisation (Moore et al. 2004). The average density can be used to determine the minimum

neighbour density required for 100% localisation coverage, or for an acceptable level of accuracy. As some localisation algorithms iteratively localise, geometrically significant nodes (i.e. nodes that might allow others to be localised) may not themselves be localised. This could result in low a coverage percentage.

A density metric based on the physical area nodes are deployed over gives an indication of the average spacing in three dimensional space. The author suggests the following metric, called Mean Density (MD) which is calculated by

$$MD = \sqrt[3]{\frac{r_{diff}(X) \cdot r_{diff}(Y) \cdot r_{diff}(Z)}{n}} \tag{2.17}$$

where n is the number of nodes in the deployment, X, Y, Z are vectors of ground truth coordinates (for 3D) of all deployed nodes and $r_{diff}(c)$ gives the largest absolute difference between coordinates in a given axis

$$r_{diff}(c) = |max(c_i) - min(c_i)| \tag{2.18}$$

Anchor placement

The position of anchors in the network may have a considerable impact on localisation error, especially if the localisation algorithm assumes that anchors are uniformly or randomly positioned in fixed locations. Assumptions about a pre-defined anchor placement scheme do not take into account environmental factors, terrain (that can affect placing of anchors), and signal propagation conditions, as well as optimal anchor placement. The geometry of anchor nodes with respect to any un-localised nodes in the network can have a varying effect on the accuracy of resulting position estimates.

2.9.5 Discussion

Accuracy, cost and coverage represent trade-offs for localisation algorithm performance. This is a consequence of the need to optimise localisation algorithms toward a set of specific constraints, such as low power operation, speed of localisation, scalability or maximum limit on positional error. A good understanding of trade-offs is important in the context of localisation, as it is in general for WSN application design. For example, deploying a network with a large number of anchors is expensive, and requires a large amount of careful placement, especially to guarantee coverage. However, in attempting to minimise or remove entirely the need for anchors, a localisation algorithm may compromise its accuracy and simplicity; anchor-free localisation algorithms are frequently centralised (even Robust Localisation (Moore et al. 2004) requires a central phase), and framed as non-linear optimisation or minimisation problems (Girod et al. 2006, Gotsman & Koren 2004). These approaches may not be tractable to run directly on resource constrained nodes.

It has been shown that accuracy metrics based on average position error may not capture the accuracy of the layout geometry. This is especially true for anchor-free localisation algorithms. It has also been shown that the cost of a localisation algorithm can take many forms, and can be highly dependent on the application requirements the WSN is designed and deployed to address. Coverage is greatly affected by placement of nodes in the network, be they anchors or regular nodes.

2.10 A localisation algorithm development cycle

The development and evaluation of a localisation algorithm should be considered in its entirety—this implies theoretical modelling and simulation as well as real-life validation of the algorithm. Each stage of the development should characterise and validate a specific aspect of the algorithm. Simulation validates how the algorithm can operate under controlled, simulated conditions—this verifies that the algorithm functions correctly. Emulation verifies that the algorithm can work correctly using empirical data that reveals conditions which are complex to simulate. Realistic validation shows that the algorithm can work in target environments and with the hardware platforms which are being targeted to support it.

It has been proposed by Whitehouse that whilst simulation is different from real-world performance, one would expect it to be indicative (within some error bound of empirical results) and decisive (an algorithm which performs best in simulation should perform best in reality) (Whitehouse et al. 2004). Therefore, when one is evaluating a localisation algorithm against others, one must make sure it performs better in both simulation and realistic deployment. The verification and validation of a localisation algorithm at each of the four stages (modelling, simulation, emulation and deployment) becomes more expensive in terms of (at least) time and cost as we approach real-life deployment. The value of simulation/emulation comes forth with respect to scalability and low cost of entry for researchers—there are no embedded hardware requirements.

2.10.1 Simulation

Researchers can use simulation to simplify some of the difficulties of real deployment (time synchronisation, for example) such that any algorithmic flaws can be isolated at an early stage. For this reason, it is not sensible to try to start with in-situ deployment without simulation verification. Environments such as Matlab (Anonymous 2009*f*), ns-2 (Breslau et al. 2000), OmNet++ (Varga 2001), Ptolemy (Eker et al. 2003) and EmStar (Girod et al. 2007) would be used to simulate the performance of localisation algorithms. Different simulation environments allow lesser or greater control over node and network parameters relevant to localisation. Simulators such as ns-2 and OmNet++ aim to provide the user with accurate models of wireless propagation and protocol performance, providing a high level language in which to implement simulations. Their wide academic use is desirable for consistency between institutions in a way custom simulators cannot guarantee. Custom simulators can be designed in a variety of languages (Java, C and its variants). Ptolemy provides a hugely powerful framework for modelling, simulation and design of embedded systems using graphical techniques to create state machines, akin to Matlabs Simulink. Development frameworks like EmStar allow researchers to develop end-to-end wireless sensing systems, allowing the same code to be used for simulation, emulation and deployment. Hardware specific simulators, such as TOSSIM (Levis et al. 2003) and AVRORA (Titzer et al. 2005) can be used when accurate profiling is required (in power consumption analysis, for example). There also exist localisation specific simulators, such as Silhouette (Whitehouse et al. 2004, Whitehouse & Culler 2006), SeNeLEx and RiST (Reichenbach et al. 2006). These environments do not need to be used in isolation, of course—measurements and observations derived from one could be used as set-up parameters in another, or to help inform custom simulation software.

2.10.2 Emulation

Using empirical data to inform simulation parameter values, rather than purely calculating them (for example, ranging or communication data) represents an addition to the realism of a simulation. Empirical data sets can capture some of the environment-specific effects that basic models cannot. The Statistical Emulation method (Whitehouse et al. 2004), is an example of gathering a data trace in-situ, and using it to power a realistic localisation simulation (thus making it an emulation). Part of the challenge of performing this type of emulation is gathering a data set which represents the environment in sufficient detail. For example, Whitehouse gathered range data using 20 ultrasound enabled nodes that have been arranged in such a way that all ranges between 0.5m and 4.5m (at 0.25cm intervals) can be measured (Whitehouse et al. 2004). This approach captures environmental specific problems, such as non-estimates (range could not be measured) and node-to-node ranging variations (induced by electronic or mechanical differences between nodes). The data is used in the Matlab based Silhouette localisation software (Whitehouse et al. 2004, Whitehouse & Culler 2006) to investigate its effects on the performance of several localisation algorithms, comparing the results with pure simulation and finding a disparity between the two.

One of the most powerful emulation frameworks to date is EmStar (Girod et al. 2007). EmStar allows the user to perform simulation, emulation and real deployment using the same framework. This means code developed and simulated can be cross-compiled and tested on real embedded hardware. This approach is advantageous as there is a reduction in the amount of porting required. EmStar allows network connectivity to be emulated in real-time using live test bed data, making it a powerful tool for transitioning to real hardware from simulation through emulation.

2.10.3 Real-life deployment

The strength of using test beds lays in actually being able to run algorithms on real hardware, and gather non-simulated data. This can be particularly useful for testing radio communication, for example. However, creating localisation test beds can often be difficult because algorithms are affected by environmental context. Ranging mechanisms will most likely work differently indoors and outdoors, for example if signal strength is being used to determine range or location. Evaluating an algorithm on a test bed in a different environment than the application targets may give an incorrect indication of the algorithms performance.

Real life deployment of a localisation algorithm on hardware in an indicative environment (i.e. similar to where the real network will be deployed) is the most important evaluation of a localisation algorithm. Unfortunately, it is also the most time consuming, costly, and error prone aspect of localisation evaluation. An in-situ evaluation of a localisation algorithm will most likely be as demanding as a real deployment of the network in terms of planning, deployment equipment and time taken to deploy.

The deployment phase of localisation algorithm evaluation is also the most error prone and unpredictable, so researchers should have a detailed plan of how and what data needs to be gathered. The aim should not be to perform a large amount of testing, but to have directed and planned experimentation. Software will most likely need to be adapted to work correctly in the field, and worst case scenarios (what to do if everything fails) should be planned for. Several days should be set aside for deployment, with the understanding that the likelihood is high that things will not work as expected first time.

2.10.4 Summary

When evaluating localisation algorithms, it is difficult to separate the issues arising from actual deployments from theoretical drawbacks and constraints of various algorithms. From a theoretical perspective, it is desirable to have an algorithm that is independent of the ranging technique used and platform capability, as well as being robust to the deployment environment and generic with respect to application requirements.

Given that a WSN is deployed for some realistic, physical monitoring and processing aim, the localisation algorithm designer should always have some set of motivating applications in mind, throughout the design process. These can be general classes of applications such as tracking and location awareness or clearly specified applications such as forest fire monitoring and animal call localisation. Different applications will place different weightings on the various criteria discussed at the start of this chapter—scalability, accuracy, coverage and cost.

In conclusion, evaluating localisation algorithms is not to be underestimated by researchers. In order to fully evaluate a localisation algorithm, its performance must be tested in simulation, emulation and realistic environments. Both the design and development process for new localisation algorithms and the process of selecting a *best fit* algorithm for a particular application requires consideration of the trade-offs between accuracy, cost, coverage and scalability the localisation system needs to achieve. Although simulation is the least costly and most used tool for evaluating algorithms within the WSN domain, with respect to localisation researchers must be aware of the limitations of purely simulated models, especially for radio communication and inter-node distance estimation.

The use of metrics to describe the quality of localisation is important for all evaluation criteria, but possibly most notably for accuracy evaluation. Using Euclidean error is a common approach, but may not be indicative of how closely a localisation solution *fits* ground truth. Also, when ground truth is not available, an equivalent metric must be found which tells the user how closely the localisation estimate matches the initial constraints (such as inter-node spatial estimates). Considering the domain's state-of-the-art, being able to instantiate a specific localisation algorithm is still not an easy thing to do. Even after choosing a localisation algorithm that is most suitable for the motivating application, it is likely that researchers will still have to implement it on specific hardware (with relevant ranging measurement mechanisms, if applicable) before being able to evaluate its performance.

2.11 Summary

This chapter has provided a detailed view of the literature that most concerns this thesis: real-life WSNs, self-localisation and source-localisation algorithms, and the techniques used to gather data to be used as input to these algorithms. Throughout this chapter, there have been observations which have influenced the work presented in this thesis, and the approaches taken to evaluation. These observations are listed below.

Real life evaluation

Any system that is built for a real-life application will require a stage of testing in-situ. This is true for evaluation of all services, including self-localisation. Consequently, in-situ evaluation of both self and source localisation was carried out and is presented in Chapters 3 (self-localisation) and 5 (source

localisation) respectively.

High data-rate systems

High data-rate systems bring greater challenges than dumb sensing networks because they require some form of in-network processing. Different authors have taken different approaches to deal with high data rates, but in general there is a need to filter out and further process or communicate only useful data and potentially apply compression. Thus, two approaches to reduce data and apply intelligent in-network processing *when it is suitable* are presented and evaluated in Chapters 6 and 7.

Signal detection and ranging

Range estimation is used in most fine-grained self-localisation algorithms. Some algorithms however use primarily angle information, or a mix of angle and range information. Signal detection is one of the most important parts in range and angle estimation as it directly relates to the accuracy and precision of a given technique. Acoustic ranging systems are readily implemented from off the shelf parts (microphone, speakers). Additionally, the audible acoustic range has a high bandwidth for signal modulation (20 Hz–20 kHz), whilst having a slow propagation speed which can be used in conjunction with faster electromagnetic signals for RToA techniques. The benefit of an accurate signal detection algorithm for acoustic ranging is shown in Chapter 3, where several ranging implementations are evaluated.

Self-localisation

Self-localisation algorithms are generally designed to meet some constraints. Whether these constraints are imposed by device, network or application, they affect how the algorithm operates. In range-only based systems, it is important to ensure that ranges are consistent with geometry. This becomes more complex in 3D. Self-localisation in 3D with both range and range and angle-based algorithms is presented in Chapter 3.

Source-localisation

Source localisation is a more complex problem than self localisation. Not only does it rely on the accuracy of node-localisation, but it must deal with complex, wideband signals which have to be detected before they can be processed through the localisation pipeline (Figure 2.2). Chapters 4 and 5 deal with the issues related to source-localisation by means of a system developed to support a specific application, and its subsequent refinement in Chapters 6 and 7.

Chapter 3

Acoustic self-localisation

Both Chapters 1 and 2 described node-localisation as the process of determining the relative physical positions of nodes in a deployed WSN. In general, node-localisation is important to give positional context to the physical data that sensor nodes are gathering. Chapter 2 presented the state of the art with respect to range and angle estimation and localisation in WSNs. This chapter concentrates on ranging and self-localisation (where the WSN performs node-localisation) in WSNs for applications with a 3D requirement.

Some WSN applications require the WSN to localise the position of a phenomena relative to the positions of the nodes in the network, and based on the data these nodes are sensing. These are *source localisation* applications, and examples are: localising the position of a structural fault in a building (Chintalapudi et al. 2006), the position of a water leak in a pipe (Stoianov et al. 2007), the position of an animal from its call (Ali et al. 2007), or the position of a vehicle from its magnetic signature (Arora et al. 2005).

Some of these WSN-based source localisation applications need only a coarse-grained sense of position of the target and the nodes, such as: *the target x to be localised is closer to node b than node a.* An example from the literature is the localisation of cane toads for migratory tracking over long periods of time (Hu et al. 2005). Another class of these applications require a more accurate, *fine-grained* estimate of the position of the target being observed, in a coordinate system (such as Euclidean or geographic). For both applications, any error in the estimated positions of the nodes will result in error in the estimated position of the target. However, for applications with fine-grained requirements, this can cause unacceptable error in position estimates. It is therefore important to minimise the positional error of the nodes in the node-localisation process.

For fine-grained applications that reference node and target position in a coordinate system, the distinction between estimating position in two dimensions (2D) and three dimensions (3D) is important. The addition of an extra degree of freedom (the z axis) requires extra measurements to be made to remove positional ambiguity (as noted in Chapter 2). In addition, measurement estimation error (range or angle) can have more of an effect on position estimates in 3D because of the extra degree of freedom in which the error can manifest.

In real-life deployments, nodes must be placed in real 3D environments. It should follow therefore, that to gain the best node-localisation and localisation estimates the positions must be estimated in 3D. However, if the deployment for the nodes shows little vertical variability, it may be suitable to estimate positions in 2D rather than 3D, thus removing a degree of complexity. This unfortunately, is not a general solution to the problem of deploying in realistic environments.

Some WSN applications with fine-grained requirements will not be able to guarantee placement over

a relatively planar area, and so node-localisation must be performed in 3D. The source-localisation application considered in this thesis is an example of this: a WSN is used to estimate the position of a marmot in its natural habitat, based on the acoustic signals (alarm calls) it creates. The result of estimating the position of the animal is to allow further observation. In an attended deployment scenario the user can take a photo based on the position estimate, and in an automated, unattended scenario, the position estimate could be used to actuate cameras to the position of the marmot, automatically taking photos or footage. The automated approach is not covered in this thesis, but the reader is directed to work on *multi-modal target tracking* in WSNs by Kushwaha et al. (2008).

Accurate node-localisation in 3D is possible to perform manually, by using surveying tools, GPS measurements, or even measured distances between nodes (using range-finders or tape measures). However, these approaches are error prone (due to human involvement), time consuming (especially as the number of nodes increases) and may inadvertently induce error into the position estimates. Additionally, if the nodes require knowledge of their own positions, there needs to be a facility to communicate these positions to the WSN. Therefore, the node-localisation method chosen in this thesis, and specifically in this chapter is self-localisation, where nodes can automatically perform the localisation process themselves. It has the potential to be quicker than manual localisation (certainly in the range and angle gathering stage) without the potential for human-induced error that manual node-localisation may create.

The contribution of this chapter is the evaluation of both acoustic ranging and 3D localisation algorithms of varying complexity on sensing platforms of varying complexity, with respect to 3D localisation.

As discussed in Chapter 2, acoustic range estimation between devices is an easily accessible and cheap option for accurate ranging in WSNs. It requires only a radio, speaker and microphone, as well as a suitable ranging algorithm. Time of flight (ToF) ranging approaches such as those introduced in Chapter 2 (Time of Arrival, Relative Time of Arrival and Two-Way Ranging) can be used to cooperatively estimate range between devices. Therefore, acoustic ranging is chosen for ranging experimentation in this chapter.

Three different acoustic-based ToF ranging algorithms described in Chapter 2 were evaluated experimentally in this chapter in order to assess their relative accuracy, precision and operational range: Relative time of arrival (RToA), Time of arrival (ToA) and Two-way ranging (TWR). The three ranging algorithms were implemented on three different platforms that have different processing and hardware resources: the Mica2 (kBs of Random Access Memory (RAM), 8-bit Atmel micro-controller), the Gumstix (MBs of RAM, 32-bit ARM microprocessor) and the Acoustic ENSBox (MBs of RAM, 32-bit ARM microprocessor). These three platforms were chosen because they represent three different points in the WSN design space: the Mica2 is a de facto low-power WSN platform, the Gumstix is a general-purpose embedded platform and the Acoustic ENSBox is a specialised platform for prototyping acoustic embedded sensing applications.

The Mica2 is a COTS platform with an implemented acoustic ranging algorithm based on RToA. The Acoustic ENSBox is a custom-made platform, with a ToA-based acoustic ranging algorithm implemented. The third platform was developed using off the shelf components to represent the middle ground in the design space. The platform is compromised of a speaker, an electret microphone, Gumstix processing board, and AC '97 compatible expansion sound board.

As will be shown in Section 3.2.1 on page 79 and Section 3.3.1 on page 90, the ENSBox and Gumstix

platforms had greater resources (RAM, CPU, custom hardware) than the Mica2 platform, and were able to achieve longer operational range, and greater accuracy and precision. The accuracy and precision of the ranging mechanism was directly related to the complexity of the algorithms that were deployed on the platform. The acoustic signal type, time synchronisation technique, signal detection algorithm and sample rate were all shown to be factors in accurate acoustic ranging.

Self-localisation in 3D was investigated on the Mica2 and the Acoustic ENSBox. First, *lateration*, a primitive of many localisation algorithms (described in Chapter 2), was used to evaluate the performance of acoustic range-based Mica2 localisation on a controlled 3D terrain. Second, a more complex and complete 3D localisation algorithm, implemented for the Acoustic ENSBox was evaluated in several outdoor environments, which varied in area, relative height and ambient noise level. The author's contributions in this chapter are as follows:

- Accuracy and precision characterisation of three acoustic ranging mechanisms on three platforms: the Mica2 (Section 3.1.2), the Acoustic ENSBox (Section 3.2.1) and the Gumstix (Section 3.3.1).

- Implementation of a proof-of-concept platform for acoustic ranging based on the Gumstix single board computer, including hardware and software integration and implementation of a suitable ranging algorithm.

- The evaluation of multiple localisation algorithms with respect to terrain complexity, geometric dilution of precision, and ranging algorithm.

Controlled, indoor and outdoor experimentation was carried out with respect to several ranging mechanisms and localisation algorithms. This was necessary to understand the relative performance of both ranging mechanisms and localisation algorithms in realistic environments. In addition, simulation of sensor localisation was performed to understand and isolate the effects of varying ranging precision and GDoP. Simulation was sufficient to provide a theoretical understanding of precision and GDoP effects and how they affect the design parameters involved in ranging mechanism and localisation algorithm selection.

The chapter is divided up into three sections, where three different acoustic ranging implementations are evaluated on three different platforms. First, the performance of an off-the-shelf ranging algorithm developed for the resource constrained Mica2 platform was evaluated, and tested with a real 3D localisation scenario, using lateration. Second, the performance of a more capable, but highly specific acoustic sensing platform (the Acoustic ENSBox) is evaluated, and tested within several outdoor 3D localisation scenarios, using accuracy and precision metrics (as discussed in Section 2.5).

A third platform was developed as an even trade-off between ranging accuracy and precision, operational range and hardware and software requirements (see Section 3.3). This platform was based on the Gumstix single board computer. This third platform was necessary because the previous two platforms represented extremes of the design trade-off. The Mica2 required a minimal amount of processing and hardware, but had short operational range, low precision and accuracy (which in turn limited 3D localisation accuracy). The Acoustic ENSBox had a long operational range, high precision and accuracy

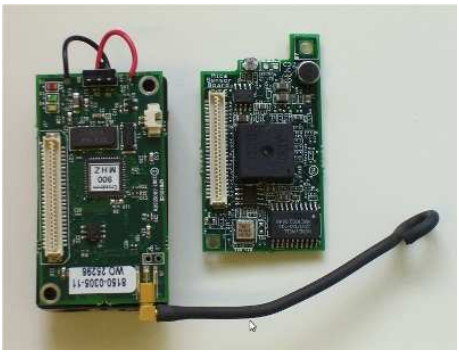


Figure 3.1: The Mica2 platform and external sensor board with microphone and sounder

(which allowed for accurate 3D localisation), but required custom hardware and had high memory and proceessing requirements.

3.1 The Mica2 platform

The experimental work described in Section 3.1 uses the Mica2 platform to perform acoustic ranging and localisation. A suitable ranging mechanism was found and used to range between several devices, and perform localisation through lateration.

The Mica2 platform (shown in Figure 3.1) is a third generation mote from the University of California Berkeley mote family. Traditionally it has been the platform of choice for various laboratory-based WSN experimental set-ups and some real-life deployments as reported in the literature (Priyantha et al. 2001, Priyantha et al. 2003, Moore et al. 2004, He et al. 2003, Sallai et al. 2004, Stoleru et al. 2004, Whitehouse 2002, Whitehouse et al. 2004, Whitehouse et al. 2005, Whitehouse & Culler 2006, Zhang et al. 2007). Although largely superseded by the Telos platform, the Mica2 still finds use in deployments where 2.4 GHz frequencies are readily absorbed by moisture and foliage, such as in rainforests. Though a variety of operating systems exist for the Mica2 (and similar platforms), the most popular is TinyOS (Hill et al. 2000). TinyOS programs are written in nesC, a dialect of C which implements the abstractions specified by TinyOS (Gay et al. 2003).

The Mica2 is an 8-bit micro-controller based platform, using the Atmel ATMega128L. It has 4kB RAM, 128kB program flash and 512kB serial flash memories. The processor runs at 8 Megahertz (MHz) and the clock runs at 32.768 kHz. The Analogue to Digital Converter (ADC) has 10-bit resolution and can be sampled at up to 17.7 kHz. Wireless communication is provided by a ChipCon CC1000 radio (Anonymous 2004), which transmits at relatively low data rates (around 19.2kbps) using either 433 MHz or 900 MHz carrier frequency.

An external sensor board (shown in Figure 3.1) provides a suite of sensors for the Mica2: a temperature sensor, light sensor, 2-axis accelerometer and an electret microphone and sounder, which generates a fixed frequency acoustic signal in the 4.5–5 kHz range.

This provides the Mica2 with the necessary hardware to enable acoustic ToF ranging. To assist the ranging process, the sensor board allows the microphone output to be passed through a National Semi-

conductor LMC567CM low power tone-decoder (or tone-detector) (Anonymous 2002). The tone detector has a Phase locked loop (PLL) circuit which responds to frequencies in the range of the Mica2 sensor board's sounder. When the response of the microphone and the PLL are high enough in combination, an interrupt is signalled, indicating that the tone-detector has identified the sounder's tone. This provides a way to process acoustic signals sent from the sounder in hardware without using up valuable mote resources. Several ranging implementations have made use of the tone detector to provide a lightweight ranging mechanism.

In Calamari (Whitehouse 2002), a RToA ranging scheme was implemented using the tone detector, with a maximum operating ranging distance of 2m. The error from the ranging mechanism was non-Gaussian, meaning averaging many samples may not converge on zero error. An improvement over this method was implemented (Kwon et al. 2005), adding a custom 9V speaker and using an encoded binary signal (rather than a single tone interrupt) to increase the operational range. Ranges of 15–30m were reported outdoors, but subsequent experimentation (Scherba & Bajcsy 2004) reported an average indoor range error of 2m at short distance (most likely due to multi-path/reverberation). In Thunder (Zhang et al. 2007), a ToF ranging mechanism based on TDoA was implemented. A non-mote acoustic source (an amplified speaker) generated an event, which was detected by nodes that were time-synchronised using the Time Flooding Synchronisation Protocol (FTSP) (Maroti et al. 2004). The experimental results of this system report an average error of 49cm and a worst case average error of around 200cm at distances ranging from 15 to 150m. The authors in all of the above works observed differences in sounder frequency and false detections from the tone decoder added to variability between different sensor boards with the same nominal specification.

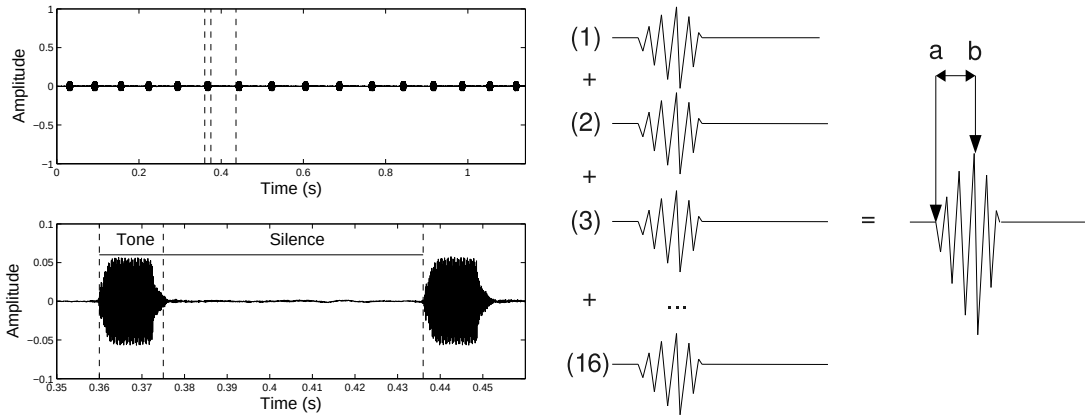
Additionally, saturation of the decoder was also a problem: when the signal is too loud, this causes the signal not to be detected. These shortcomings affect the usefulness of the tone-decoder as a basis for implementing acoustic ranging on the Mica2. Other work has addressed this by ignoring the tone decoder and performing signal detection entirely in software (Sallai et al. 2004). In outdoor experimentation with fifty motes deployed on a concrete surface at distances of 1 to 10m, the range error of the mechanism was found to be normally distributed with a mean of -8.18cm and a standard deviation of 20cm.

This implementation offers the highest precision as well as reasonable operational range on the constrained Mica2 platform, making it suitable for fine-grained ranging (and hence 3D localisation). In Section 3.1.1, the algorithm is described in detail, followed by the experimentation performed using it.

3.1.1 Vanderbilt RToA acoustic ranging algorithm

The RToA approach (Sallai et al. 2004) is similar to previously described approaches (Whitehouse 2002, Kwon et al. 2005) whereby an acoustic signal is sent at the same time as a broadcast radio message; any nodes that receive the radio message attempt to estimate the range to the target. The acoustic signal is generated by turning the Mica2 sensor board's fixed frequency (4.5kHz) sounder on and off.

A recording of the ranging signal is shown in Figure 3.2(a). In total, the signal lasts around 1.5s, and is made up of a series of sixteen 1.2ms tone periods (where the sounder is generating its fixed tone) interspersed with varying intervals of silence, from 49ms to 70ms. The lower part of Figure 3.2(a) shows an example sound/silence period.



(a) A time series of a recording of the ranging signal, including a close-up of tone and silence periods.

(b) Summing of the signal to improve SNR and inferring ToA from the point of highest energy

Figure 3.2: A recording of the ranging signal and the inference of ToA from the summed signal's energy.

Each node is pre-programmed with the length of the sounding and silence periods in a ranging signal, and upon receiving a broadcast ranging message will continuously sample its microphone (at 17.7 kHz), storing the data into a software buffer. Instead of recording the sequence as a continuous signal, the observer samples it as sixteen individual 1.2ms tones (and corresponding silence intervals), each time moving its data pointer back to the start of the recording buffer. As each of the sixteen tone and silence periods are sampled, they are summed with the rest of the data currently in the recording buffer. This illustrated in Figure 3.2(b). Summing of the sequence is performed to increase the Signal to Noise Ratio (SNR), under the assumption that noise in the channel is Gaussian (and thus will be cancelled when summed with other Gaussian noise). After sampling the ranging sequence, the recording buffer data is bandpass filtered in the 4.5–5 kHz (range using a custom 35-tap time domain filter), and the ToA determined.

Because the sounder tone does not have a high-energy onset, the ToA of the ranging signal is inferred by finding the point of highest energy in the recording buffer (point b in Figure 3.2(b). This is assumed to be the middle of the summed 1.2ms tones, from which the start of the signal is found by subtracting an empirically determined offset (line ab in Figure 3.2(b)). The point of highest energy in the buffer is assumed to be the middle of the chirp (point b in Figure 3.2(b).

This ToA is considered valid if it meets two consistency heuristics: (1) the supposed tone's envelope is within a certain width (number of samples), and (2) the energy within the tone's envelope is at least two times the average value in the recording buffer.

The envelope of the signal is determined by taking the average energy over the buffer, and determining the at which a moving average running over the buffer rises first above, then below, this average. The algorithm implementers determined the envelope size empirically.

The rest of Section 3.1 describes the author's experimental characterisation of Sallai et al.'s RToA ranging algorithm (2004) in both indoor and outdoor environments. Further, several Mica2 motes were

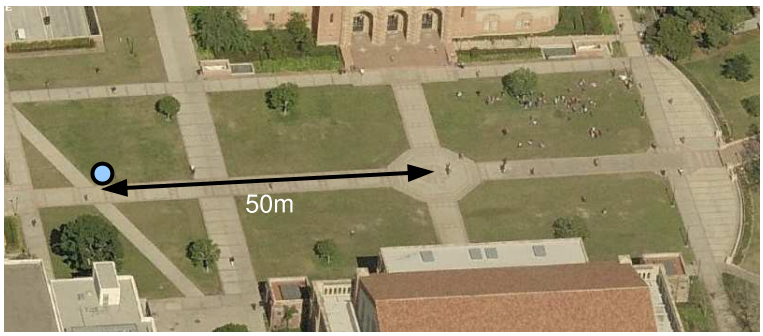


Figure 3.3: Westwood Plaza, at UCLA, where the ranging experimentation took place. The area is surrounded by buildings. To the left of the image lies the UCLA sports field, and to the right Jan steps goes upward. The circle represents the static position of the target, and the black line the trajectory of the observer (image taken from Microsoft Live Maps: www.bing.com/maps).

used to perform lateration, the most well-known range based localisation algorithm.

3.1.2 Outdoor operational range

A maximum operational range of up to 10m and a Gaussian distribution of error for outdoor environments was reported for the distance estimates made by the RToA ranging implementation (Sallai et al. 2004). To verify the reported outdoor performance, a two-node experiment was set up by the author at the University of California, Los Angeles (UCLA), at Westwood Plaza. The area was a combination of concrete pathways and grassed area, measuring around 120m by 60m and is shown in Figure 3.3.

Two Mica2 nodes were raised 1m from the ground on tripods. The target node was static (placed at the position marked with a circle in Figure 3.3) and the observing node was moved to various locations during experimentation (along the trajectory line shown in Figure 3.3). The aim of the experiment was to establish the in-situ operational range of the Vanderbilt RToA implementation, and determine the expected range error.

Nodes were initially placed 0.5m apart. The observing node was moved away from the target node in 0.5m increments until the ranging process no longer yielded range estimates. At each 0.5m increment, 50 ranging signals were emitted by the target, and the observer's estimates of range were recorded on a nearby base station laptop for offline analysis. Experiments were carried out at midnight, when there was little through traffic in the area, and ambient noise was low (59–61 decibel (dB)). Range between nodes was measured using a Hilti PD30 laser range finder (giving sub-centimetre accuracy).

Figure 3.4 shows the range error versus distance for the experiment. Each bar represents median error, circles represent mean error, and the whiskers represent 1 standard deviation.

It is clear that for the implementation of this mechanism, range estimation error does not vary linearly with distance. Additionally, standard deviation is not correlated with error: estimation at both 2m and 4m show large standard deviation but low error (less than 10cm). Mean error is largest at 5.5m (an overestimation of around 0.5m) but is bounded below 0.3m for all other ranges.

The maximum operational range was 8m, although a gap was observed at 7.5m where no measurements

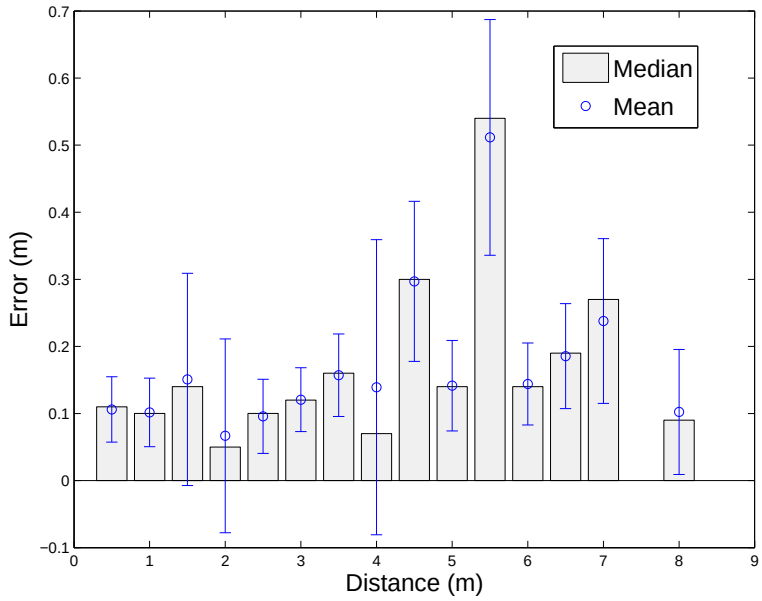


Figure 3.4: Range vs. error using the Mica2 platform outdoors. Whiskers are 1 standard deviation.

could be taken. This could be an effect of the outdoor testing environment. For example, multi-path reflections may have caused interference to the observer in the 7.5m position–the result of the ranging process at this distance indicated that the received signal did not match the expected signal envelope.

The fact that all ranges are over-estimated could be related to an under-estimation of the ambient temperature. A higher temperature would increase the speed of sound, meaning the distance would be over-estimated. The recorded temperature during experimentation was 22.4°C, giving a speed of sound of 344.767m/s. The assumed speed of sound for the ranging mechanism was 340m/s. This difference in speed of sound can account for 5.6cm of the 20cm error observed, asssuming an average distance of 4m which the sound travels over (the experimentation went from 0.5m to 8m): $(4/340 \times 344.767) - 4 = 5.6$. It is therefore highly unlikely that temperature effects alone could have been responsible the error seen in these experiments, although it may have accounted for part of it.

3.1.3 Indoor operational range

As discussed in Chapter 2, acoustic signals are affected by reflections in the environment. When the signal reflects off a surface, it can interfere with the original signal (either destructively or constructively), or cause echoes of the signal which are received by the observer. If a reflection is falsely determined to be the start of the signal, this will lead to an over-estimation of distance. Over-estimation will also occur in non-line of sight conditions, when the direct line of sight between devices is blocked, because the only version of the signal arriving at the receiver will be the reflection. Note that only over-estimates can be caused: reflection can only make a signal take longer to arrive than the direct path between devices (which is the shortest path). In the specific ranging algorithm, reflections can potentially cause over estimation

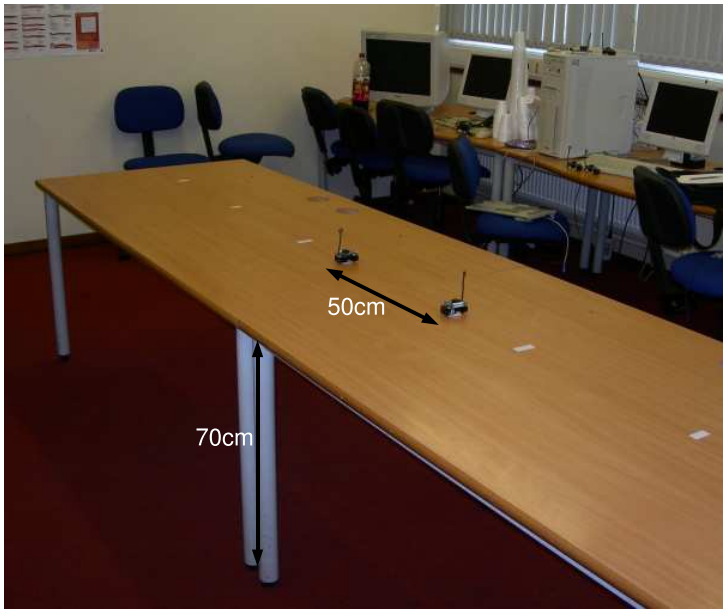


Figure 3.5: The indoor experimentation environment for ranging.

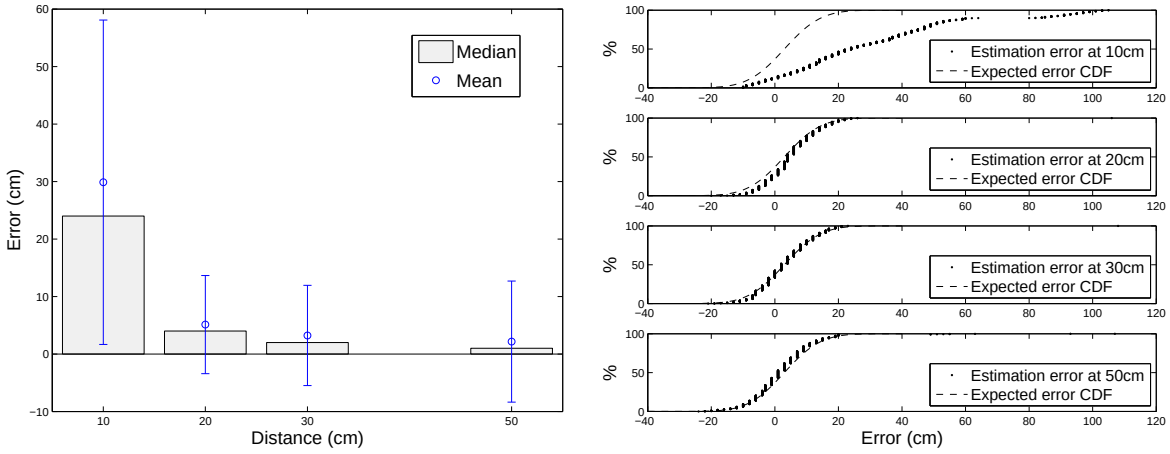
because the maximum signal energy may be slightly higher due to reflections that cause constructive interference (that is make the signal stronger).

In Section 3.1.2, outdoor experimentation yielded consistent positive overestimates of distance that were not caused solely by an incorrect measurement of the speed of sound. In addition, inconsistent error appeared that was not correlated with distance. It was hypothesised that this was related to reverberation, however the exact source could not be isolated. In Section 3.1.3, evaluation is continued from an outdoor environment to an indoor environment.

Experimentation took place in the Armstrong Siddeley Building at Coventry University, in a research laboratory measuring approximately 5m by 8m. All experiments took place approximately in the middle of the room. Motes were placed on a wooden laminate table, roughly 70cm high from the floor, as in Figure 3.5.

This was an example of a harsh indoor environment for acoustic ranging, where the closeness of the walls meant that a signal could potentially reflect multiple times, causing interference with the direct path acoustic signal between nodes. The magnitude of the interference would depend on the amount of energy that was reflected off the walls each time. For example, assuming the speed of sound to be 343m/s, it would take 23ms to reflect off the longest wall from the middle of the room and back and 15ms from the shorter wall. Given that the ranging signal was 1.15s long, it was highly likely that reflections would be able to distort the energy envelope of the signal that was sampled by the mote.

An experiment to determine operational range in the difficult indoor environment was performed. There were ten motes used in the experimentation, split into five pairs. In the outdoor experimentation time limitation meant that only 50 ranging signals were sent from one target node to an observing node



(a) (b)

Figure 3.6: Aggregated range error across all Mica2 pairs, presented as a bar graph (3.6(a)) and a series of cumulative density function (CDF) plots (3.6(b)). Whiskers are 1 standard deviation.

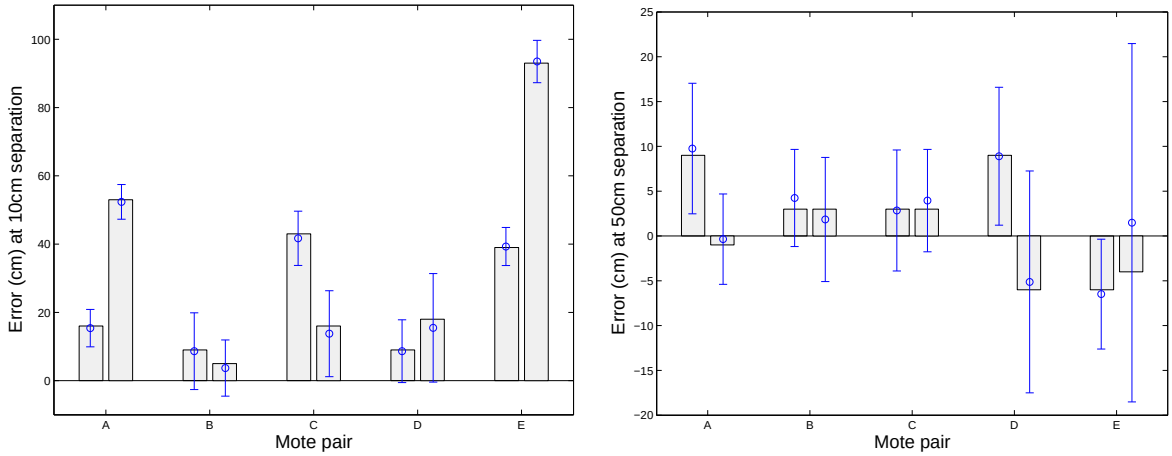
(experiments needed to be performed in the few hours between the experimental area being free of people and the automated sprinkler system coming on). The indoor experiment was not time-limited and thus it was decided that both nodes should send ranging signals, to see the difference in range estimation from two nodes the same distance apart. Additionally, the number of signals sent was increased from 50 to 100, to see if the accuracy of the range estimates could be improved by taking more samples.

Only one pair of motes was used at a time, and the position of the motes in the experimental area was kept constant. It was expected that the ranging mechanism would show similar range estimates (or range error) at different ranges and across different motes. A reduced maximum operational range was expected due to the reflections resulting from the walls in the room (compared to the 8m seen outdoors).

The goal was to determine the maximum operational range in this environment by increasing the distance between nodes in 0.5m increments, but it was found that the nodes could not make range estimates further than 0.5m (failed to give any values), indicating that the ranging algorithm was severely compromised by the indoor experimental environment. As noted in Section 3.1.1, if the ranging signal sampled was too wide or did not have enough energy at the peak, it was not valid, and so the ranging value reported would be -1 to indicate an invalid signal. Therefore, the gathered data set corresponded to four distances: 10cm, 20cm, 30cm and 50cm, with 100 range estimates attempted by each node.

Figure 3.6(a) summarises the results of the indoor ranging. The mean, median and standard deviation were calculated using all range estimates between all pairs at each distance (between 937 and 939 data points per distance out of a possible 1000 total).

The results show a reduction in error as nodes are moved further apart. Specifically, the error shown at 10cm is far greater than the error at the other distances. For each distance, Figure 3.6(b) shows a Cumulative Density Function (CDF) plot of error for all range estimates taken. CDF plotting was chosen in this case to highlight differences in the error distribution between multiple data sets. The CDF



(a) Error at 10cm

(b) Error at 50cm

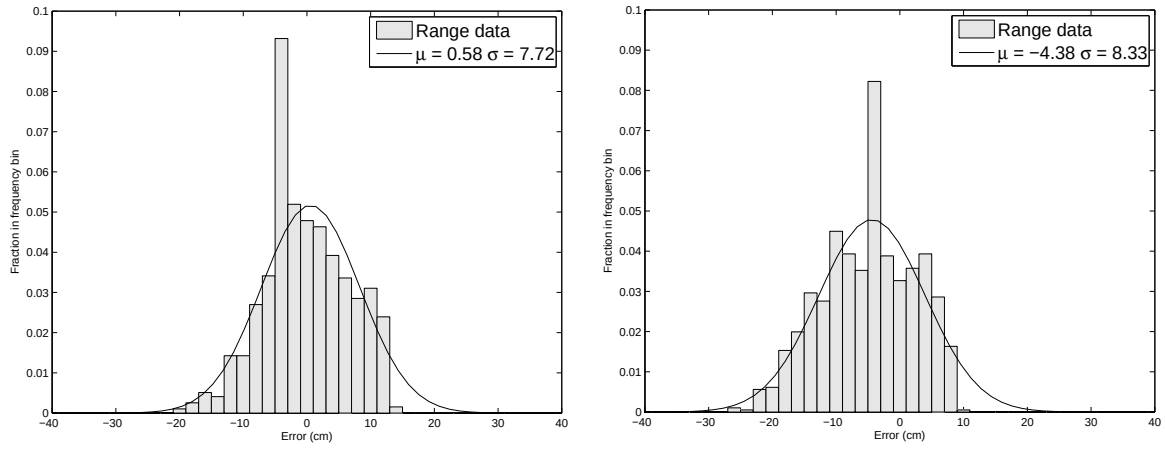
Figure 3.7: Error in range estimation at 10cm and 50cm. Whiskers are 1 standard deviation.

curves at 20cm, 30cm and 50cm are in line with the curve a normal distribution would be expected to follow, whereas the error at 10cm shows a vastly different curve which does not appear to be normally distributed. To show this graphically, a normal distibution closely fitting the data for 20, 30 and 50 was plotted under the data points (with $\mu = 3$ and $\sigma = 9$) in each graph.

Figure 3.7(a) shows the error for each pair of nodes at 10cm. Bars show median error, dots show mean error with standard deviation whiskers. Four out of the ten nodes show a median error of over 40cm, whereas the other nodes show a median error below 20cm. For comparison, error at 50cm is shown in Figure 3.7(b). One clear difference is that the error at 10cm is all due to over-estimation, whereas error at 50cm shows both under and over-estimation. Over-estimation of distance indicates that the high energy point of the signal is being detected later in the buffer than the actual start of the signal. Based on the ranging implementation, the error could potentially be due to high signal energy saturating the microphone. The differences in error between different pairs of nodes could be explained either by variations in both microphones and speakers between devices, or by variations in the number of samples taken to determine the mean distance estimate. The differences between speakers and microphones are discussed in Section 3.1.4 on page 69, and the number of samples is considered in the Section 3.1.3.

Regardless of error induced by placing nodes close together, it is necessary to revisit Figure 3.7(b), where the error at 50cm is shown across all nodes. The mean range error is spread between $\pm$10cm across pairs. This indicates that the ranging mechanism lacks accuracy, especially given that 100 estimates were made by each node. The results also show high values for standard deviation, indicating a low precision: this indicates that more than 100 estimates may have to be taken for the mean to be sufficiently close to the actual distance (assuming no bias).

Section 3.1.3 examines the effects of increasing the number of samples taken on error distribution and accuracy.



(a) (b)

Figure 3.8: Histograms of error for both motes, as a result 1000 ranging estimates each. A Gaussian distribution has been fitted to each histogram.

Indoor distribution verification

Section 3.1.3 showed that when different pairs of nodes ranged with one another at the same distances, the ranging mechanism was low precision. In order to determine whether taking more measurements could yield greater accuracy (due to the low precision of the mechanism), two motes were chosen randomly from a batch of ten to range with one another at 50cm, with the same indoor, laboratory-based experimental set-up as the previous section.

The two motes were placed 50cm apart from one another and each sent 1000 ranging signals to the other in turn. Range estimates were transmitted by each mote to a nearby laptop for offline analysis. An extremely large number of ranging estimates was taken to see if the accuracy of the ranging mechanism could be increased from what was observed when only taking 100 estimates. The rationale was that a low precision ranging mechanism would require more measurements to increase accuracy, or reveal a systematic bias.

A histogram of range estimation error for each of the two motes ((a) and (b)) is shown in Figure 3.8. A normal probability distribution has been fitted to each data set, using the mean and standard deviation of the sample set.

Whilst both error distributions correspond roughly to a normal distribution, neither were zero mean, and were not biased in the same way (that is, the means were different). The most frequent error bin is from -4 to -6cm for both nodes: the mean error in Figure 3.8(a) is in line with this (-4.42cm) but the mean error in Figure 3.8(b) is not (-0.58cm). The median error for each node is within 0.5cm of the mean in both cases (-4 and -1cm respectively). In examining the most frequent bin in the histograms, the implication is that both devices are underestimating distance by 4cm–6cm, although the mean and median error contradicts this in the case of Figure 3.8(b). It is possible that reverberation may have caused the difference in error distributions between nodes, but it is also possible that slight differences in

hardware caused slight variations.

3.1.4 Discussion

Based on the experimentation carried out in this section, the Mica2 platform and the Vanderbilt RToA acoustic ranging algorithm have shown an operational range of up to 8m outdoors and 50cm indoors. The ranging mechanism shows a broadly Gaussian distribution of error, meaning that averaging multiple samples should increase the accuracy of the distance estimate. However, in all experiments, the mean range error varied between devices (at the same distances), and had a large standard deviation. This implies the ranging mechanism has a low precision. When 1000 ranging estimates were taken between two nodes, there was still an error of several centimetres.

The error seen outdoors was different to the error seen in the indoor experimentation. This is possibly due to temperature or reverberation effects. The error observed in experimentation in this section can be thought of as the combination of two factors: internally and externally induced error, after Savvides, Garber, Adlakha, Moses & Srivastava (2003). Internally induced error is the result of platform-related factors which cause inconsistencies between devices, such as differences in speaker frequency or microphones. Externally induced error is the result of environmental effects such as reflections which cause the cause the ranging algorithm to fail or experience reduced operational range. Three factors are discussed in this section: sampling quantisation, hardware variation and environmental effects.

Sample quantisation

In Chapter 2, Section 2.6.3 described how the sample rate quantises range estimates: a low sample rate will decrease the accuracy to which a range estimate can be made. On the Mica2, the maximum sampling rate of the ADC is 17.723 kHz. This means that between each microphone sample, sound will have travelled 1.9cm. Because the *actual* arrival of the acoustic pulse may have occurred in between two samples from the ADC the estimate is quantised and could be inaccurate to $\pm$1.9cm. In addition, the ranging implementation rounds distance estimates to integers, meaning that estimates could be quantised by up to 0.5cm on top of the sampling effects. This gives a total random error of $\pm$2.4cm which could potentially occur per ranging estimate in the Vanderbilt RToA algorithm. This random error could possibly explain some of the differences seen even when taking a large amount of estimates, such as the 1000 taken to determine the accuracy of the ranging mechanism.

Hardware differences

It is suspected that variations in sounder centre frequency and microphone hardware may cause differences in range estimates. The three plots in Figure 3.9 show the energy versus frequency of three recorded ranging signals. The plots are focused on the frequency bins between 4 and 5.5 kHz (the frequency range of the sounder's output). The top and middle sub-plots show the frequency content of the same signal sampled by two different motes, and the bottom shows a ranging signal from a different mote. It is clear that both the frequency content shown in both the top and middle sub-plots are similar, with highest energy around the 4.7 kHz region (which is assumed to be the center frequency of the mote's sounder). The bottom plot shows highest energy around the 4.6 kHz region, and the distribution of energy is a different shape.

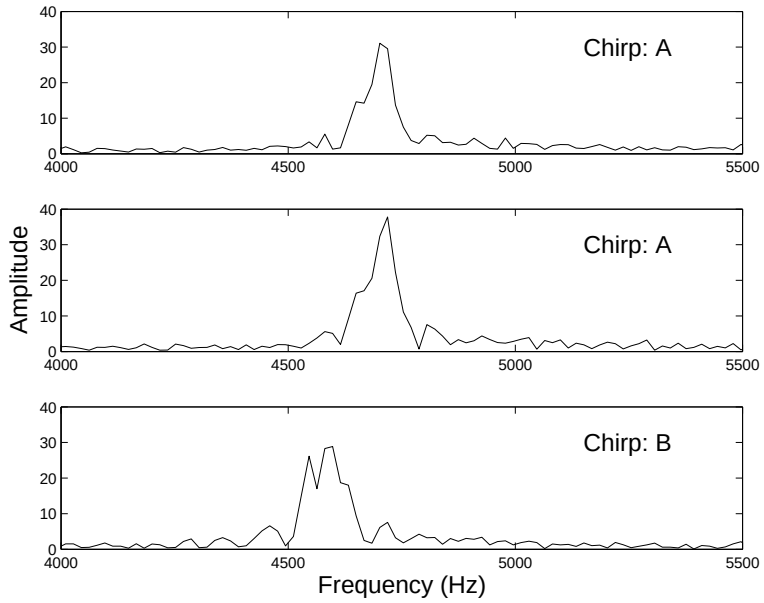


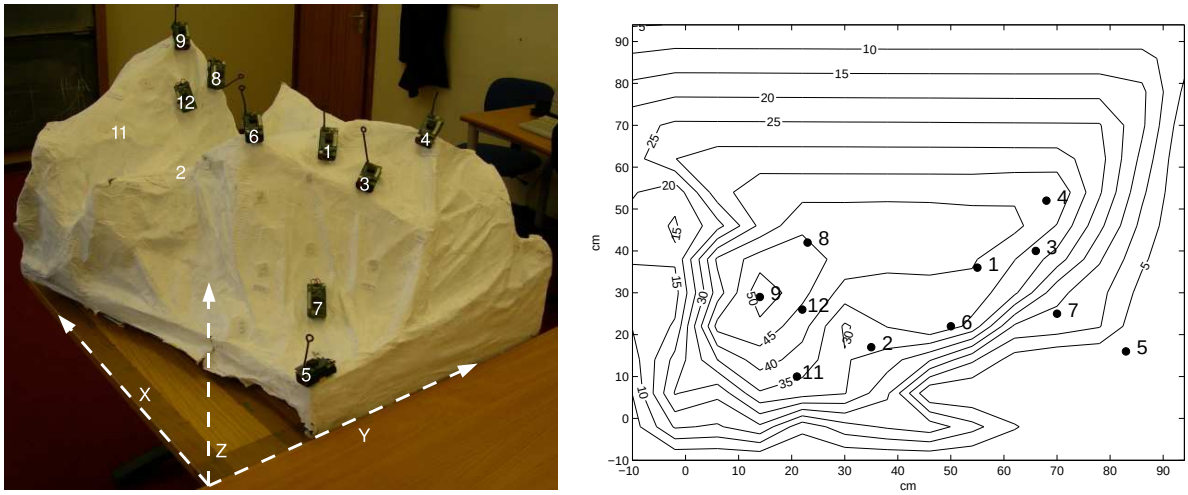
Figure 3.9: Energy versus frequency for two different ranging signals, recorded by different motes. The top and middle signals are similar (same ranging signal recorded by different motes), especially when compared to the bottom signal (from a different mote).

Varying amounts of energy in the frequency band of interest may affect the position of the highest energy point in the signal between different motes. This could lead to differences in accuracy between different sensor boards of several centimetres.

Environmental effects

The particular technique used to detect the start of a signal (using energy) is sensitive to changes in signal strength. In indoor environments, there are more reflective surfaces near the Mica2 devices and can cause constructive or destructive interference (the signal will be attenuated or amplified through destructive or constructive interference). The signal detection technique relies on several hard-coded heuristics to determine the start of the signal: (1) the maximum energy point must be two times the average energy in the buffer, (2) the number of samples between average crossing points (which determine the length of the signal) must be within a certain range, (3) the offset applied to the maximum point is 207 samples. Therefore reverberation of a signal in a given environment will alter the signal received at the node, and hence will change the average energy observed at the node in an environment and position specific manner that is difficult to predict. This is likely to affect the accuracy of the ranging mechanism: if the position of highest energy in the signal is affected by echoes, it will appear later in the buffer, creating an overestimate. In addition, the signal may be attenuated to such a degree that it is impossible to pick it out from the ambient noise after a certain distance. This will limit the maximum operational range of the mechanism.

It would appear that the best way to improve the performance of the ranging algorithm for indoor use



(a) (b)

Figure 3.10: The 3D experimental terrain, shown in a photograph (3.10(a)) and as a contour map (3.10(b)). The contour plot is intended to show the relative heights and positions of the nodes and does not necessarily reflect the heights of peaks or troughs where no nodes were placed. Throughout the experimentation, nodes 1, 3, 4 and 6 were treated as anchor nodes and used to localise other nodes.

would be to gather in-situ data and change the parameters for window size and maximum signal energy (as discussed in Section 3.1.1 on page 61). However, because reverberant environments are unpredictable and difficult to model, energy-based techniques may be too susceptible to signal interference when used indoors. Whilst sample rate and hardware differences have an effect on the precision and accuracy of the ranging implementation, approaches using correlation and coded reference signals are far more likely to produce accurate results. However, using these approaches requires more processing power and possibly frequency domain processing, for which the Mica2 nodes do not have enough memory or processing capability to support readily.

3.1.5 3D lateration

In order to understand the effect of the range errors on the performance of a ranging based localisation algorithm, a 3D localisation experiment was set up. Lateration was used as the localisation algorithm as it is based purely on range estimation, and forms the basis of most anchor-dependent localisation algorithms. Some anchor-free localisation algorithms can also use lateration if they define a local coordinate system before using lateration (Capkun et al. 2001). However, some of these algorithms will only localise nodes if there is a sufficient number of range estimates made between the node and its neighbours in order to satisfy certain rigidity criteria (Moore et al. 2004, Mautz et al. 2007), as discussed in Chapter 2.

An irregular terrain with several peaks and valleys was created using papier-mâché and wire mesh covering an area of 100 by 80cm with a maximal height of 70cm (see Figure 3.10). The terrain was placed on a table in the same room used for indoor range experimentation in Section 3.1.3, at Coventry

Table 3.1: Experiment one results: each of the six lateration runs are shown (locating node 5 using anchors 1, 3, 4 and 6) in terms of position error in cm (Euclidean distance between estimated and actual position), and mean and max range error (as a percentage of the distance between node and anchors).

Node ID	Position error (cm)	Mean range error (%)	Max range error (%)
5 (1)	12.13	3.72	5.84
5 (2)	10.89	2.51	4.22
5 (3)	34.37	3.19	7.19
5 (4)	10.87	3.49	5.48
5 (5)	18.09	3.39	4.04
5 (all)	7.39	1.84	2.86

University, Armstrong Siddeley Building. Velcro squares were attached both to the terrain surface and the underside of each mote to allow them to stick easily onto the surface. The centre of each square was assumed to be the coordinate point represented. These points were hand measured, with an estimated error of ±2cm. Three different lateration experiments were carried out, each time using nodes 1, 3, 4 and 6 as anchors. In each lateration experiment each mote in turn sent 50 ranging signals, which were received by all other nodes (this included both anchors *and* target motes). Range estimates were collected by a base-station node attached to a laptop, to enable off-line analysis of the data, including lateration. Two stages of filtering were performed on the range estimates for given experiment before being used in lateration calculations. Firstly, all measurements made by the ith node to the jth node were averaged; secondly, if range estimates had been made by both nodes, the smaller of the two averages was used. This filtering approach is based on the observation that the higher estimate is likely the result of multi-path effects (Girod 2005). The resulting estimates were used as input into a multilateration implementation developed in Matlab.

Experiment one

The aim of the first experiment was to examine repeatability of lateration performance on the terrain. Nodes 1, 3, 4, 5 and 6 were placed on the terrain, as per Figure 3.10. All nodes sent 50 acoustic signals, and each range measurement made was reported back to the base station. This process was repeated five times. For each run, the gathered ranges were filtered (as per the description in Section 3.1.5), and used in a lateration calculation to estimate node 5's position (using the known positions of 1, 3, 4 and 6). An extra run was formed by aggregating the data from the five runs under the assumption that increasing the amount of measurements would improve localisation performance. The results of each lateration run are shown in Table 3.1; the far left column shows the node ID of the target, and each run of the experiment is indicated in brackets. The positional error in the second column is shown in terms of the Euclidean distance between the actual and estimated positions. Mean and max range error were calculated across all measurements used in the lateration calculation. This error is shown as a percentage of the actual distances between target and achors.

The individual runs (from (1) to (5)) show varying levels of positional error, ranging from 10.87cm to

Table 3.2: Experiment two results: the mean and maximum range error between anchors and targets is given along with the Euclidean position error in centimetres.

Node ID	Position error (cm)	Mean range error (%)	Max range error (%)
5	97.50	7.19	13.74
7	37.06	19.84	32.03
8	23.48	3.94	5.24
9	31.02	10.76	12.54
12	n/a	22.68	22.86

34.37cm. Both the mean and max range error do not seem to be definite indicators of positional error, although the highest positional error run also has the highest max range error (run (3)).

When the range data from all experimented was aggregated (mean values were calculated across all anchor to target ranges gathered), the resulting position estimate improved to 7.39cm positional error. This was expected based on previous observations that increasing the number of measurements increased accuracy (Section 3.1.3). It is likely that in each run, measurements used from several (different) anchor nodes were not accurate, but when the measurements across all runs were aggregated, they were more accurate.

However, given that the node to anchor distances ranged from 36.73 to 50.45cm (average 44.8cm), this represents positional error that is around 16% of the average node to anchor distance. This does not represent an accurate localisation result.

Experiment two

The aim of experiment two was to examine the performance of lateration for multiple targets in different locations. In the experiment, four extra nodes were added to the terrain as targets (7, 8, 9 and 12). Lateration was carried out using 1, 3, 4 and 6 as anchors and 5, 7, 8, 9 and 12 as targets. The results of the laterations are shown in Table 3.2. Firstly, it should be noted that although it did not physically move, the positional error seen in localising node 5 is considerably larger (nearly 100cm vs. 10cm—35cm in the first experiment). It is likely that node 7 caused a partial obstruction between node 5 and the anchors, which affected the ranging estimates.

When considering the position error of nodes 7, 8 and 9, a large average and maximum range error does not seem to translate to a large Euclidean error, indicating that other factors related to the lateration calculation are causing error. For example, node 8's position is incorrectly estimated by 23.5cm, yet the maximum range measurement is only 5.2% different to the actual range, and only 3.9% different on average. This is most likely due to the relative positions of the anchors and the targets, and hence the effect of Geometric Dilution of Precision (GDoP) (discussed in Section 3.1.6 on the following page). Node 12 could not be localised, due to lack of range estimates.

Table 3.3: Experiment three results: the mean and maximum range error between anchors and targets is given along with the Euclidean position error in cm.

Node ID	Position error (cm)	Mean range error (%)	Max range error (%)
2	n/a	25.80	42.41
5	128.21	13.15	19.18
7	62.16	33.85	55.21
8	200.07	20.96	44.51
9	319.11	15.36	48.70
10	n/a	20.18	44.00
11	n/a	105.19	196.55

Experiment three

The aim of experiment three (as with experiment two) was to examine the performance of lateration for multiple targets in different locations. In experiment three, three more nodes were added, and node 12 was taken away (due to hardware failures). Lateration was carried out using 1, 3, 4 and 6 as anchors, and the rest of the nodes as targets. The results are shown in Table 3.3. Again, the lateration error is different: node 5's positional error changed from 97cm in Experiment two to 128cm in Experiment three, most likely a direct result of the average and maximum range error rising. Nodes 2, 10 and 11 could not be localised due to a lack of ranging estimates. As with experiment two, if these nodes were to be localised (assuming the newly added nodes as anchors), then their position estimates would be hugely inaccurate. For nodes 8 and 9, it can be seen that a maximum range error of nearly 50% of the actual range has led to a positional error of 200cm and 319cm respectively.

3.1.6 Discussion

It is clear from the experiments presented in this section that error in range estimation causes significant Euclidean error in the position estimates made using lateration. In the first experiment, the average ranging error was less than 4% of the distance between nodes. Even when the ranges used in this experiment were aggregated (giving an average 1.8% range error and max error of 2.9%), the positional error was still 16% of the distance between target and anchor. This implies that the lateration calculation is sensitive to noise in error measurements. However, as seen in experiments two and three, the magnitude of the lateration error does not seem to directly relate to mean or max ranging error, meaning it is hard to tell when given just ranging error how inaccurate the position error will be.

It is likely that in addition to potentially inaccurate ranging estimates, the relative geometry of the anchors with respect to the target (thus the GDoP) was an influencing factor on the accuracy of the resulting position estimate. Section 3.1.6 investigates these effects in more detail.

GDoP's influence on positional error

The arrangement of anchor nodes on the terrain in relation to the targets was not ideal in the experiments reported in Section 3.1.5. For example, calculating the GDoP for the anchors with respect to node 5

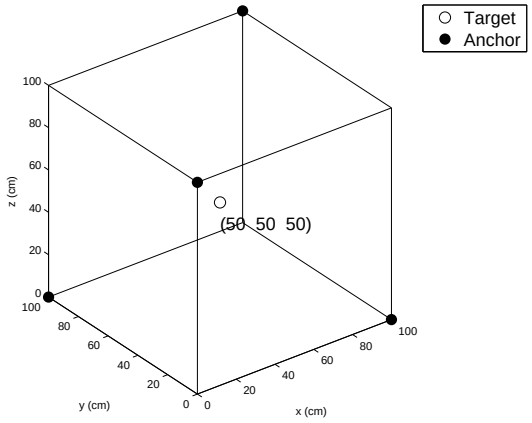
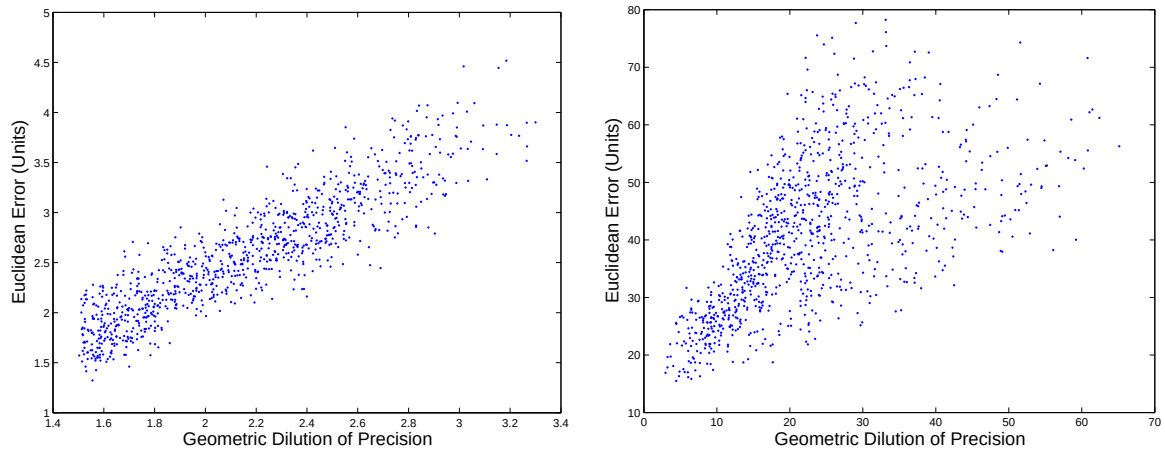


Figure 3.11: The ideal geometry for four anchor nodes locating a target in 3D.

yields a GDoP of 11.325 (using the equation given in Chapter 2, Section 2.7.1). A better geometry would involve placing anchor nodes to form a perimeter surrounding the target. For four anchors in 3D, the ideal anchor configuration is in a tetrahedral shape surrounding the target. The GDoP of this configuration with respect to node 5 is 1.836. For non-ideal configurations, the Euclidean error of a lateration will be magnified when the GDoP is high.

To demonstrate the relationship between GDoP and Euclidean error for a given position, a simulation testbed was implemented in Matlab. A 1 unit cube was created, and random target positions generated within the cube. Two sets of anchors were placed–the *ideal* configuration, and the configuration seen on the 3D terrain. For each target point generated, the GDoP was calculated and 50 multilaterations performed (for each geometry). Each multilateration consisted of a single range measurement being generated for each anchor (with respect to the target). Randomly generated noise, drawn from a normal distribution with zero mean and standard deviation of one was added to the range values. For each multilateration, the Euclidean error was calculated. For each data point, the mean of all multilaterations at a given target position was taken. In total, 1000 target positions were evaluated for each geometry. The results are shown in Figure 3.12.

The effects of anchor position in the simulation are interesting. For the ideal geometry, the GDoP is low for all target positions (a maximum around 3.3), and the Euclidean error of position estimates are similarly low (up to 4.5 units). For the terrain geometry, both the GDoP and Euclidean error is an order of magnitude higher. Consequently, for lateration, it is important not only to consider the error in ranging estimates that participate in the computation, but also the relative geometry of the sensor nodes. This experimentation has shown that anchor placement which results in a high GDoP value has a direct effect on the resulting error in position estimates. The other component of inaccurate position estimation, range error, is described in Section 3.1.6.



(a) Tetrahedral geometry

(b) Terrain geometry

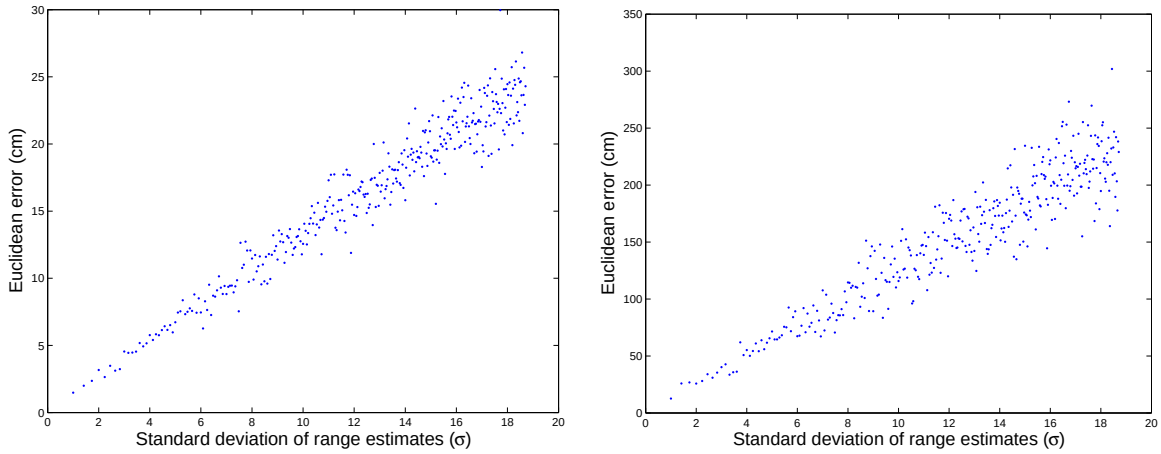
Figure 3.12: The effect of GDoP on positional error for the ideal anchor configuration and the anchor configuration used in 3D experimentation.

Effects of ranging precision

As discussed in Section 2.5, a high precision estimation mechanism will exhibit low spread of residual error, thus a low standard deviation. Thus, the more accurate the mechanism, the lower the standard deviation. To demonstrate the effects of varying standard deviation of a ranging mechanism on Eculidean error, a testbed simulation was set-up in Matlab.

In the experimentation, the anchor configurations used were the same as Section 3.1.6: an *ideal* geometry and the geometry used on the terrain. In the ideal anchor configuration, four anchors were arranged in a tetrahedron on the edges of a 1m cube. The target was placed at $50, 50, 50$, the centre of the cube. This represents the most ideal 3D anchor/target configuration for four anchors, and has a corresponding GDoP of 1.5. Second, the terrain configuration was used, with the target being node 5 (GDoP of 11.325). In the simulation, range estimates were generated by corrupting the exact distance between nodes with a noise component, drawn from a zero mean normal distribution and a standard deviation which ranged from 1 to 18.71 (a variance of 1 to 350). The results are shown in Figure 3.13. The x axis shows the standard deviation of the ranging mechanism and the y axis shows the positional (Euclidean) error of the particular lateration calculation. Each data point is the mean of fifty lateration calculations, where each lateration calculation uses only one simulated range observation from all anchors (as in Section 3.1.6).

It can be seen that Euclidean error increases as the standard deviation of the simulated ranging mechanism increases in both Figure 3.12(a) and Figure 3.12(b). However, this positional error is affected by a high GDoP value: the error in Figure 3.12(b) is an order of magnitude greater than in Figure 3.12(a), where the only difference between the two simulations is the relative arrangement of anchor nodes. For reference, the standard deviation of the Mica2 ranging mechanism experimented with in Section 3.1.2 was around 20cm. With such a low-precision ranging mechanism, one would expect position error on the



(a) Ideal anchor geometry (GDoP = 1.5)

(b) Terrain anchor geometry (GDoP = 11.325)

Figure 3.13: The effects of increasingly low precision range estimation on positional error in lateration.

order of what is shown in 3.12(b), especially if only a few ranging estimates are made from each node.

To summarise, the simulations for GDoP and standard deviation show that the lateration on the experimental 3D terrain was affected by the geometry of the anchors, but also the precision of the ranging mechanism. The positional error is a combination of the range error (discussed in previous ranging experimentation) and the position estimation algorithm. Errors in the position estimation algorithm can come from: (1) inaccuracy in the known position of anchors and (2) configuration of anchors which magnify an error which is present. The combination of these factors has made the fine grained 3D localisation presented here unpredictable and unreliable. Fine-grained ranging and localisation, where nodes are separated by less than a metre, is clearly not possible with this approach.

3.2 Acoustic ENSBox

The previous section discussed and experimented with a de facto low power WSN node: the Mica2. The Mica2 represented the *worst-case* for ranging and localisation, in that it was only capable of running an energy based ranging algorithm, and was inaccurate and limited in operational range in the presence of environmental disturbances. This platform was not designed with a specific application in mind, and so was not optimised toward a certain localisation approach. The limitations of the Mica2 make accurate self-localisation difficult.

This section describes ranging and localisation using a platform from the opposite end of the WSN design spectrum: the Acoustic Embedded Networked Sensing Box (ENSBox). The ENSBox is a platform dedicated to distributed acoustic sensing, specifically, acoustic source localisation. Two versions of the Acoustic ENSBox hardware exist: the first was a proof of concept (hereafter called the V1 node) (Girod 2005) and the second was a more compact, rapidly deployable platform, suitable for for in-situ scientific experimentation (hereafter called the V2 node). The V2 node shares the same main processor board as the original Acoustic ENSBox. The main processing board is a Sensoria Slauson: a 400 MHz PXA

(a) V1 array (b) V2 node

Figure 3.14: The microphone arrays for the Acoustic ENSBox V1 and V2 nodes. The V1 array is separate from the main node packaging (not shown), but the V2 array is part of the main node design. Note the speaker block in the V2 node sits in the middle of the array, whilst the speakers are below the microphones on the V1 array.

255 processor with 64 MB memory, 32 MB on-board flash and two Personal Computer Memory Card International Association (PCMCIA) slots containing a 4 channel sound card and an 802.11 wireless card. In addition, the V2 node has a Gumstix co-processor for data archival. V2 nodes have an 8-hour lifetime from an internal 5400mAh Li-ion battery, making them suitable for rapid, attended deployment.

The ENSBoxes have a compact array of microphones for sensing and speakers for acoustic localisation signal emission (as shown in Figure 3.14). The four microphones are arranged in a tetrahedral configuration, allowing the determination of 3D DoA (θ,ϕ). The V1 and V2 boxes have slightly different microphone and speaker configurations: the V1 node has an 8cm baseline microphone configuration (Figure 3.14(a)) and the V2 node has a 12cm baseline microphone configuration (Figure 3.14(b)). The V2 node array configuration reduces the effects of spatial aliasing, as discussed in Chapter 2. In the V1 node, a four-way speaker was mounted under the microphone array, with a separate speaker to allow the microphone channel one to record the emission of its own acoustic signal. In the V2 node, the speaker block sits in the middle of the configuration and projects directly over the channel one microphone.

The design and implementation of the ENSBox was motivated by bio-acoustic based scientific applications. Such applications enable species census, classification and behaviour studies to be performed. This motivating application class requires highly accurate self-localisation performance so as to not induce error in resulting position estimates made by the WSN.

In previous work (Girod 2005), it was reported that a network of ten V1 ENSBoxes self-localised with an average 2D error of 11.1cm and 3D error of up to 57.3cm in an 80m by 50m area. The target accuracy

was ±0.5m positional error over the given space. The difference in performance was attributed to a lack of variation in the elevation between nodes across the deployment space. In 3D localisation, less variation in the z axis (i.e. elevation) compared to the x and y axes will reduce the accuracy of the 3D localisation estimate (this is true for x, y and z axes). In the worst case, a lack of variation in any one of the axes across the network will cause a degenerate solution for the lateration calculation, meaning the position cannot be determined. This is related to the GDoP effects on lateration, seen in Section 3.1.6, where anchor nodes will ideally surround the target node(s) in the x,y and z axes.

Experimentation in this section was performed with the V2 acoustic ENSBox, which has not been previously characterised to the same extent as the V1 box by Girod (2005). Thus, the experimentation in this chapter provides the first steps of characterising the implemented ranging and localisation algorithms (Girod 2005) on the V2 box in 3D environments. It is important to note that the biggest concern when implementing the V2 node with the speaker block being placed in the middle of the array was that it potentially obstructs signals arriving from certain angles. This chapter provides experimental results to show that the speaker block does not affect the AoA estimation performance of the Acoustic ENSBox in self-localisation.

3.2.1 Operational range

The ranging mechanism used by the ENSBox is acoustic ToA. The ENSBox nodes are equipped with omni-directional speakers that are used to emit pseudo-noise ranging signals. Previous experimentation with the V1 node had revealed an operational range of at least 90m, with a mean error of 1.73cm and a standard deviation of 1.76cm (after outliers rejection). An experiment to determine operational range was carried out using the Acoustic ENSBox at Westwood Plaza, UCLA, the same location the Mica2 outdoor ranging was carried out at (Section 3.1.2). Two ENSBoxes were placed on tripods one metre from the ground. Three nodes in total were required to maintain time synchronisation for the ToA mechanism. The third node was a gateway node, with no acoustic sensing capabilities. The target node remained in a static position, and the observing node was moved increasingly further away from the target. To replicate the Mica2 experimentation, the observing node was initially moved in 0.5m increments from 0.5m to 8.5m away from the target. Subsequently, the observing node was moved at 10m intervals up to a distance of 40m. This provided a total of 21 discrete distances between observer and target from 0.5m to 40m. At each distance, five measurements were taken, based on the previously observed small standard deviation of ranging measurements with this platform. Although the V2 nodes could have been placed further apart than 50m, the laser range finder used to measure the distance became difficult to locate properly after this distance. (Approaches using tape measure were attempted, but it was difficult to maintain straight edges and uniform tension. Given the accuracy of the ranging mechanism, this type of ground truth measurement would have induced more error.) Figure 3.15 shows error at each distance measured with the ENSBox, with whiskers showing one standard deviation of the error. This demonstrates the low variance, high precision of the ranging mechanism over distance. The average standard deviation over all measurements was 0.42cm, with standard deviations at 0.5m, 1m and 30m being 1.04cm, 1.81cm and 1.87cm, respectively. All other standard deviations were below 0.5cm. Two Least Squares (LS) fits were applied to the data in Figure 3.15: the solid line (LS fit 1) does not include the 0.5m and 1m data,

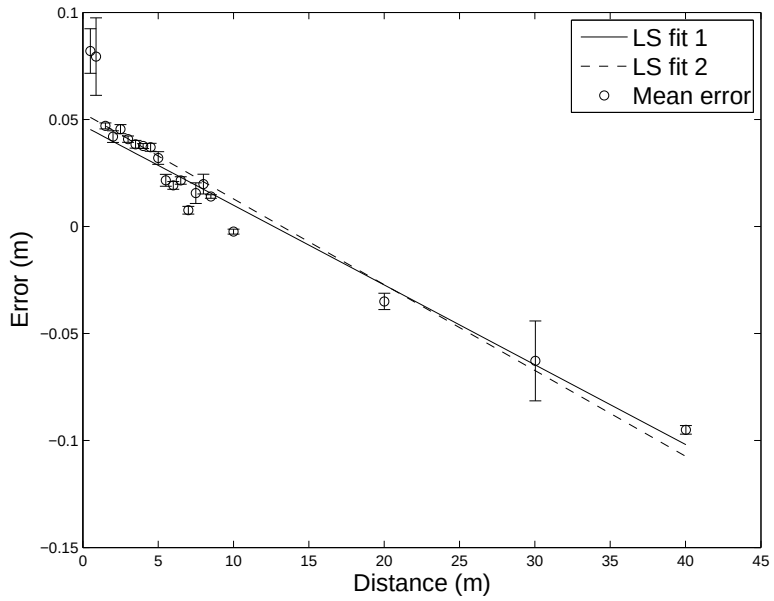


Figure 3.15: Range error vs. distance using the Acoustic ENSBox. Whiskers show one standard deviation. Solid and dashed lines represent LS fits of data: the solid line includes all data, and the dashed line includes all except 0.5m and 1m measurements.

and the dashed lined (LS fit 2) does. One explanation for the error seen at 0.5m and 1m is that it is related to the output power of the speaker, which may saturate the microphone, causing some error bias in estimation at short distances. This does not explain the larger variance at 30m however. The range becomes increasingly under-estimated as distance increases.

It is possible that underestimation is due to an assumption that the speed of sound was faster in computation than it actually was. When experimentation started, the temperature was 12.8°C (339.115m/s), and reduced during the experimentation period to a low of 10.4°C (337.695m/s). Assuming an average distance sound travelled of 20m (experiments went from 0.5m to 40m), then this is enough to account for around 8.4cm of relative error ($(20/339.115) \times 337.695 = 19.916m$) as the temperature decreased. The error seen over the distance was from +5cm at 1.5m to −9cm at 40m. The gradual drop in temperature could therefore explain the slope of the error between 1.5m to 40m, and the difference between the assumed speed of sound used in the range computations (343.371m/s) and the speed of sound based on the initial observed temperature of 12.8°C (339.115m/s) could account for a bias of +2cm ($1.5/343.371 \times 339.115 = 1.48$m). Temperature measurements were made during the experimentation with a digital temperature sensing unit, but they were too coarse to be applied as compensation to the ranging mechanism (accurate to 0.1°C) and were more likely to decrease the accuracy of the ranging estimates (Girod 2008). In this case, the best approach is to get ranging done in as quick a time as possible, and at a time where the temperature is least variable, such as early morning (Collier 2008).

Despite these explanations, it is difficult to prove conclusively that part of the range error is is not a

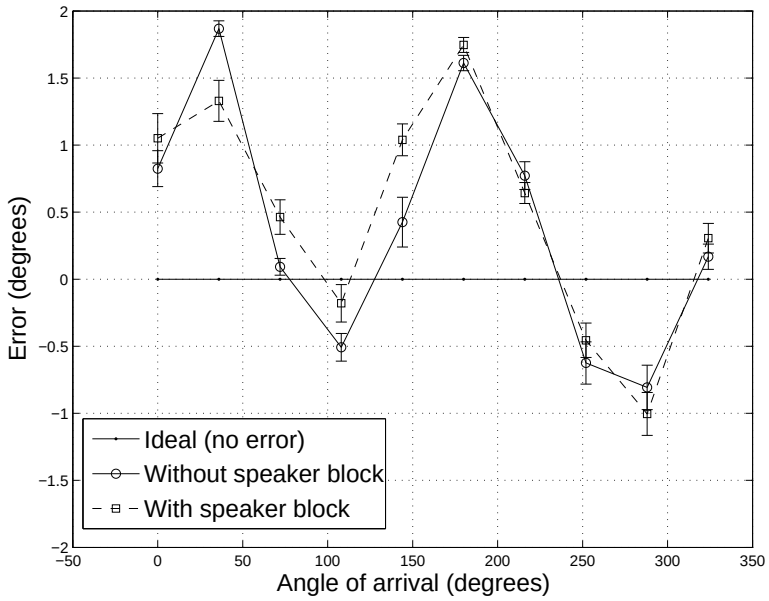


Figure 3.16: Error (in degrees) vs. Angle of Arrival, using the Acoustic ENSBox. The blue line indicates experimentation with the speaker attached, and the red line with the speaker removed. Whiskers indicate one standard deviation of error.

function of range. The experimentation in Section 3.3 also shows the same trend of increase range error at distance, but under similar experimentation conditions. However, since the mean error is below 10cm at 40m, it accounts for a relatively small amount of error (0.25%).

3.2.2 Angle of arrival estimation

A coarse grained characterisation of the ENSBox's AoA estimation was performed (azimuth only), in order to observe any potential bias induced into measurements taken with the speaker block present. Obviously, if the speaker block arrangement caused appreciable error, this would affect the ENSBox self and source localisation algorithms' accuracy. To perform the experimentation, one ENSBox was mounted on a swivelling stand, which could be manually adjusted at 2 degree intervals (at a 0.25 degree accuracy). Another ENSBox was positioned 5m away. The boxes started facing one another (zero degrees AoA), with the static ENSBox emitting five ranging signals at each new position. At each step, the swivelling ENSBox was moved 36 degrees, thus a total of 10 steps were performed. The experiment was run twice, once with the main speaker block attached to the ENSBox, and once without.

Figure 3.16 shows the error vs angle of arrival with and without the speaker block, with whiskers of one standard deviation of error around the mean. The overall standard deviation was 0.88 degrees when using the speaker block, and 0.82 degrees when not using the speaker block. Mean error was 0.37 degrees, and max error was 1.91 degrees when using the speaker block; mean error was 0.49 degrees and max error 1.84 degrees without.

Both data sets seem to follow a similar periodic pattern of error that is indicative of the ENSBox

pulling away slightly from the dead centre of the stand as it was rotated. It is also possible that slight inconsistencies between the physical array geometry and the assumed geometry (in the software) could lead to this kind of error. Both of these effects were observed by Girod (2005) in the V1 node characterisation. The error characteristic appears to be biased by +0.5 degrees from 0 degrees error. This indicates that there was a systematic bias associated with the original lining up of the nodes. Given the equipment used, a 0.5 degree error in lining up the nodes for experimentation is not unreasonable. If this bias is assumed, the error is ±1.5 degrees over the rotation of the node.

Based on the measurements gathered, there does not appear to be a large difference between having the speaker block mounted on the node or not. The standard deviation of error between the two was within 0.06 degrees, and the difference in mean error across all angles between the two configurations was 0.12 degrees. A more accurate characterisation of the Angle of Arrival performance would reveal any small differences between using the speaker block or not. This would require more accurate mounting equipment to move the node, and perhaps the use of a surveying tool or similar to measure the amount by which the node rotates. This is left for future work.

3.2.3 3D localisation performance

The Acoustic ENSBox localisation algorithm does not require anchors to estimate relative positions. Instead, it makes use of the zenith/azimuth AoA estimates as well as range estimates between devices to guess initial positions in a polar coordinate system. This guess is then used as the input into a non-linear least squares algorithm, interleaved with orientation estimation. Including AoA estimates in a localisation algorithm is difficult because localisation then requires relative node orientation to be estimated as well as position. However, this provides more information with which to construct an initial guess for the node positions: with angle and range estimates, node positions can be guessed as polar coordinates to provide a starting point for the non-linear optimisation used in the later stages of the localisation algorithm. A good initial guess of position is important for optimisation so that the global solution is found, not a local minima (this is discussed more in Chapter 2, Section 2.7.6).

The 3D localisation capabilities of the V2 ENSBox were tested in four different locations in the UCLA campus during May 2008. The four locations were chosen to provide a combination of different factors that could reduce the accuracy of localisation systems: obstruction through foliage, variation in height and variation in coverage area (thus density of deployment). To provide accurate ground truth, a mechanism which was more accurate in determining position than the ENSBox was required. The SOKKIA Set 510 Total Station for accurate surveying was used in conjunction with a prism reflector. This enabled distance measurement accuracy to within ±2mm and relative angle (azimuth and zenith) to within ±5 *arcseconds* (1 arcsecond = 1/1296000 degrees). Nodes were deployed so that they *faced* the Total Station, to allow for ease of computing ground truth orientation of nodes. Figure 3.17 shows the Total Station, and a node facing the Total Station in order to determine the relative range and angles. Measurements between node and Total Station required line of sight visibility, so in partially obscured areas (such as Boelter Hall), node deployment had to be considered in this context.

Figure 3.17: The Total Station used to determine ground truth of node positions in deployment. The Total Station is at least an order of magnitude more accurate than the ENSBox localisation system.

Boelter Hall

Boelter Hall at UCLA has an outdoor courtyard situated in the middle of the building. The area is almost fully walled on three of four sides. There is a lot of foliage from large palm trees (see Figure 3.18), and the floor is mainly concrete paving, alongside soil. Air conditioning for the building provided a low ambient noise, which was increased by works traffic arriving near nodes 108 and 109.

Seven V2 nodes were placed around the courtyard area, covering a L-shaped area of 43.4m by 20.6m, with a variation in relative node heights of 2.3m. Each node sent four ranging signals, and the raw acoustic data was recorded on each node for post-processing.

Bruin Plaza

Bruin Plaza is a fully concreted, open plaza in the UCLA campus. It offers some potential deviations in height due to several flights of stairs which overlook the plaza. In total, nine nodes were deployed in an elliptical configuration over an area 46m by 52.5m, with a variation in relative heights of 4.87m. Their approximate positions are shown in Figure 3.19. In particular, the placement of node 110, 111, 112 and 113 were on steps (see Figure 3.19). The area was largely open, although there was significant reverberation around node 112, which was placed on a small curved stage. Due to time restrictions, just two ranging estimates were taken from each node.

Jan Steps

The Jan Steps link Westwood Plaza and Royce Hall at the UCLA campus. Alongside the steps is a grassed and paved area which follows the slope of the steps as they rise up to Royce Hall. There are obstructions, such as overhanging foliage, although several trees grow on the slope. Nine nodes were deployed over the area from the bottom to the top of the slope, as shown in Figure 3.20. The area

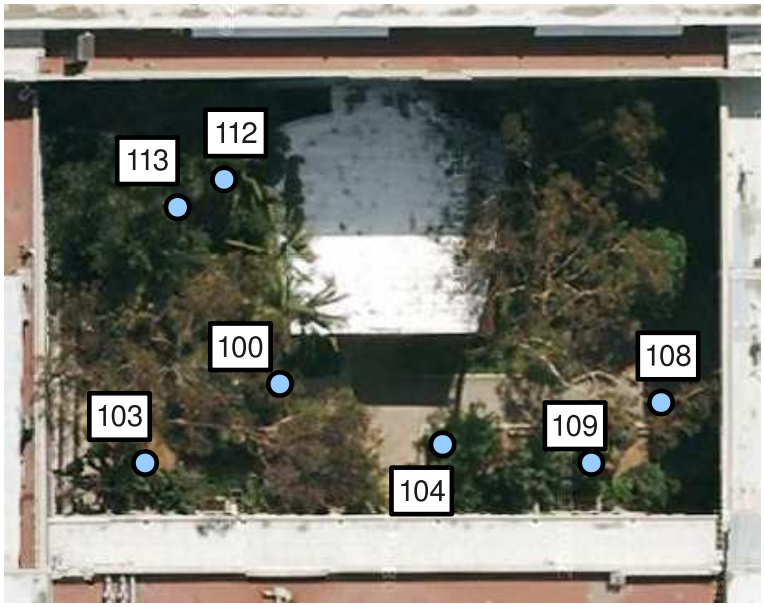


Figure 3.18: The approximate placement of nodes for the experiment carried out at Boelter Hall. The area was partially covered by large trees, and a reasonable source of noise was present from the right hand side of the figure during experimentation due to air conditioning (image taken from Microsoft Live Maps: www.bing.com/maps).

covered was roughly a square, measuring 98.4m by 97.1m, with a variation in the relative heights of the nodes of 10.8m (around a 10% grade, which is a gentle slope). Each node sent ten ranging signals to other nodes.

Royce Hall

The Royce Hall area is a flat, grassed area, enclosed on two sides by buildings, and open on the other two sides. A concrete path runs around the grassed area. The buildings reflect acoustic signals noticeably. Nine nodes were deployed over a 52.2m by 53.3m area (roughly square), with a relative height variation of 0.27m.

Results

At each deployment, the raw data corresponding to the ranging signals was recorded by each node for offline processing. The data was aggregated onto a PC, and the range processing and localisation algorithms run off-line, using existing EmStar software, written by Girod (2005). From these results, and the range table used by the localisation algorithm (the median of the lowest distance estimate between each pair of nodes), several metrics discussed in Chapter 2 were computed in order to aid analysis of the localisation results. The metrics calculated for position error were Global Energy Ratio (GER) and the Mean and Max Absolute Error (MAE, MaxE). GER measures the overall quality of the Euclidean distances between nodes compared to the actual range estimates, and MAE and MaxE give absolute position error compared to ground truth. The Mean Range Residual (MRR) metric was calculated to

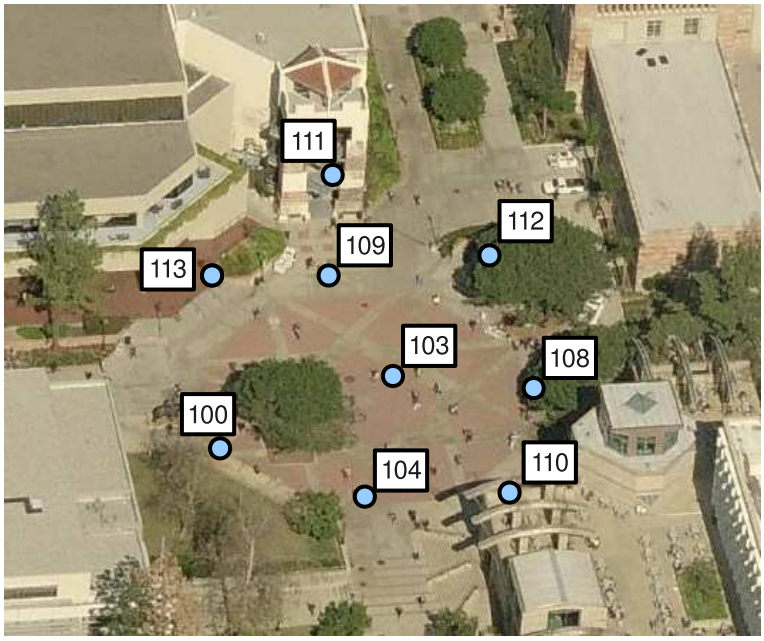


Figure 3.19: The approximate placement of nodes for the localisation experiment carried out at Bruin Plaza (image taken from Microsoft Live Maps: www.bing.com/maps).

show the mean average error between Euclidean distances between nodes and the actual estimated ranges. Finally, Mean Density (MD) metric was calculated to allow positional and range error to be considered with respect to the area over which the nodes were deployed.

Because it is anchor free, the ENSBox localisation algorithm produces a scaled representation of the map. In order to match the scaled map to the ground truth, the Procrustes technique (Kendall 1989) was used to find the scale, rotation and translation that best fits the ground truth (minimised according to the sum of squares error).

Table 3.4, shows the results of the localisation with respect to the performance metrics GER, MAE, MaxE, MRR and MD. First, across all experiments, the GER values are consistently low: this indicates a good internal geometry created by the localisation algorithm. Second, the mean absolute error is below 0.4m for all experiments and MaxE is below 0.5m for all experiments apart from Jan Steps, which had a maximum error of 0.78m. It is likely the maximum error at Jan Steps was induced by node spacing and 3D variation. The experiment covered a larger area with 9 nodes than the other experiments, in terms of length, width and height. This is reflected in the high MD value, which indicates there was on average a node every 22.55m in the 3D volume encompassed by the deployment. It appears that node spacing and 3D configuration has a greater effect on error than the provision of a limited amount of measurements has(the Bruin Plaza experiment took only two range estimates per node). This is due to the high precision of the ranging algorithm, allowing accurate localisation with minimal number of range estimates. The MRR metric compares the Euclidean ranges of the localisation (scaled according to the

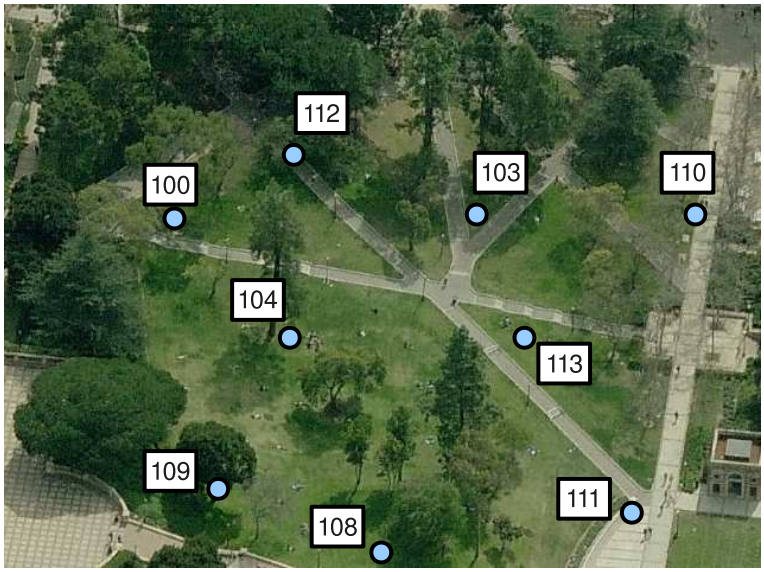


Figure 3.20: The approximate placement of nodes for the localisation experiment carried out at Jan steps. The area was open, with little reverberation. The nodes were deployed on a slope (image taken from Microsoft Live Maps: www.bing.com/maps).

Procrustes fit) with the actual ranges measured by nodes. In all cases, the ranges between nodes were underestimated, which is indicative that the set environmental temperatures used to calculate ranges were incorrect. It is important to look at the range error in an application context: recall that for the source localisation algorithm, DoA estimates are intended to be used for source localisation. Therefore, it is important that the geometry of the physical topology estimated by the self-localisation algorithm be consistent with the actual physical topology. However, actual distance error is not as important, given that for DoA triangulation, it is the angles between nodes that must be accurate. This means the topology need only be correct to a scaling factor, as was shown here. The inclusion of a thermistor to compensate for ambient temperature changes is left to future work.

3.2.4 Discussion

Clearly, there is a large difference in accuracy between the Mica2-based 3D lateration and the 3D localisation presented in this section. However, the Acoustic ENSBox platform has considerably more processing power and custom hardware and uses AoA estimates to aid with localisation.

What follows in this section is a discussion of the ENSBox localisation algorithm with respect to its motivating application. It was established at the start of this section that the dominating system requirements for the motivating application were geometrical accuracy and robustness to range error. In terms of scalability, although the algorithm is anchor free, the processing it performs is centralised, and comes at a large computational cost. The assumption in this case is that the number of nodes deployed will not be so large to take an unreasonable amount of time for the algorithm to converge on a solution (order of minutes). The algorithmic complexity in this case is $O(N^3)$, which make scalability difficult in

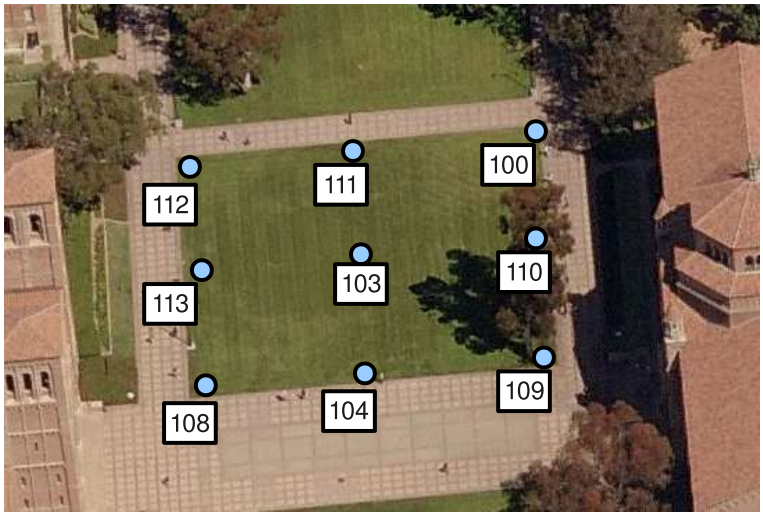


Figure 3.21: The deployment layout at Royce Hall, UCLA. this deployment was the most planar of all the areas under experimentation (image taken from Microsoft Live Maps: www.bing.com/maps).

Experiment	GER	MRR (m)	MAE (m)	MaxE (m)	MD
Royce Hall	0.0023	−0.41	0.20	0.42	4.37
Jan Steps	0.0016	−0.77	0.37	0.78	22.55
Bruin Plaza	0.0013	−0.18	0.22	0.44	10.93
Boelter Hall	0.0025	−0.14	0.24	0.46	6.65

Table 3.4: Results of 3D localisation in the four different environments. Five metrics of localisation performance as described in Chapter 2 were calculated: GER (Global Energy Ratio metric), MRR (the Mean Range Residual metric), MAE and MaxE (the Mean and Maximum Absolute Error between estimated and ground truth positions), and finally MD (the Mean node Density).

theory.

However, this localisation system is expensive. Each ENSBox has plentiful resources (64 MB RAM, 400 MHz ARM CPU), which also come at the expense of a shorter battery life. These hardware choices are understandable in the context of the application: acoustic source localisation requires data to be sampled at high rates and processed/filtered locally. In addition, performing DoA estimation requires multiple microphones. The localisation system is highly accurate, more than meeting its positional requirement in 2D (worst case 10cm error) and just going over 0.5m error in 3D. Special care is given to robust behaviour, but the cost for this is a high number of measurements for each node–the localisation algorithm requires an over-constrained linear system to remove outliers. In a topology where the number of range measurements per node is low, outliers are likely to become difficult to remove, or even identify. Because the localisation algorithm is computed centrally with all measurements, coverage is either 0% or 100%; the algorithm either converges on a result or it does not. This is clearly a problem for scalability.

In maximising the accuracy and resilience to measurement noise, the localisation system becomes

constrained in scalability and unconstrained in cost (power usage, message sending, and computational complexity). This is intuitive if one imagines the criteria in tension–one cannot be maximised without affecting the others. This would seem to limit the generality of the localisation approach, but one could argue that any self-localisation motivated by a specific application (rather than application class) will make similar optimisations to maximise performance.

The Mica2 represents the device-centric approach to WSN self-localisation, which asks the question: *given a platform and pre-determined sensor suite, how accurately can acoustic ranging and localisation be performed?* This is in contrast to the ENSBox's application-centric approach, which asks: *what level of self-localisation is required by the motivating application?* Whilst the ENSBox represents the most accurate acoustic self-localisation system available for WSNs at the time of writing, its capabilities are a reflection of the high level of accuracy and processing power required by its motivating application. Therefore, the ENSBox localisation solution is not suitable for every application. In fact it is really only suitable for high-data rate acoustic sensing applications.

There is a gap in the WSN design space which is obvious when considering the Mica2 and ENSBox: applications which require accurate localisation, devices which can provide a great deal of processing capability, but with a general appeal.

3.3 Gumstix

In general, for an application to benefit from acoustic localisation, it must be able to add this functionality at a low cost in terms of hardware and software requirements. The Mica2 energy-based ranging approach is far too constrained to be tractable for any real WSN applications, and the ENSBox approach has high overheads to perform ToA (dedicated time synchronisation, four microphones, etc).

However, in using a microprocessor based platform (like the ENSBox or other *microserver* class platforms) instead of a micro-controller based platform (motes in general), the ability to operate at low-power is severely restricted (certainly with off-the-shelf 32-bit embedded platforms). However, for some applications it is important to have processing capability available. For example, to provide on-node processing for supporting high data rate applications.

This section presents the implementation and experimentation with an embedded sensing platform developed almost entirely with off-the-shelf components which allows accurate ranging (and thus localisation) at a low cost.

Localisation experimentation is not carried out in this section: the aim is to show as a proof-of-concept that platforms can be built to meet the middle ground in the WSN self-localisation design space where accuracy and precision can both be achieved with minimal requirements on extra sensors (such as microphones and speakers).

The development of the Gumstix acoustic platform was carried out over a short time span. The basic hardware platform essentially consists of a microphone, 4-way speaker block, an audio card and main processing board with wireless radio. The main processing board was the standard 32-bit Gumstix Verdex platform, which comes equipped with a Bluetooth radio for communication, a 400 MHz processor and 64 MB RAM. The AudioStix board was used to provide a single channel audio input for the microphone and the line out to the speaker. The speaker block design and parts were borrowed from the V2 node

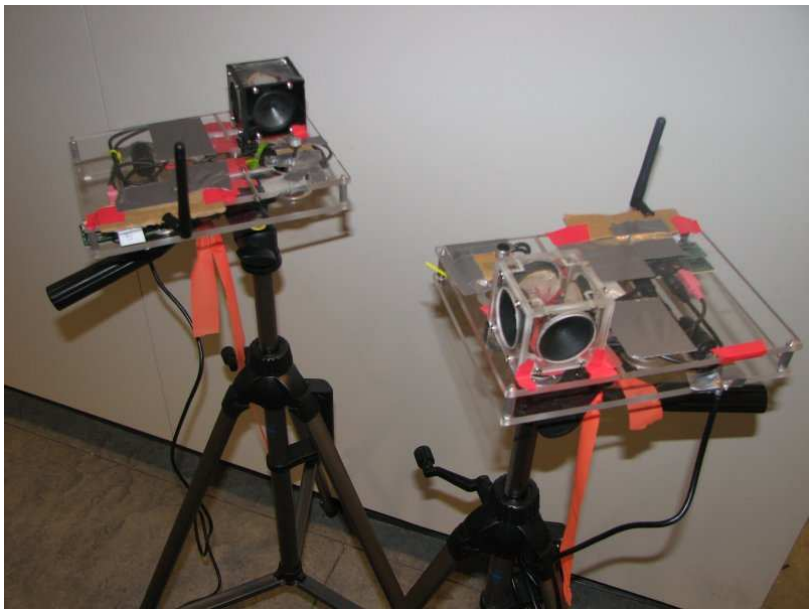


Figure 3.22: The Gumstix-based ranging platform

design, but wired to a mini jack, allowing it to be driven directly from the line-out of the AudioStix. The platform was powered through the USB port, via a USB power injector, taking two AA batteries. The nominal lifetime of the platform under experimentation (that is, frequent radio and audio operation) was around 2.5 hours. Figure 3.22 shows a picture of the platform.

The Gumstix ranging platform created strikes a balance between the extremely constrained Mica2 and the extremely application-specific ENSBox.

The Gumstix itself runs Linux, and a version of the EmStar framework was cross compiled using the provided toolchain so that components could be re-used in creating the ranging software.

The ranging algorithm is an implementation of the TWR algorithm from the Beep Beep system (Peng et al. 2007). The algorithm uses a similar approach to Girod (2005) to determine signal sending uncertainty: in order to know exactly when it emitted a signal, a node must record the signal it is sending and determine the time it physically left the speaker. This removes any sending uncertainty that may be introduced in the application software or operating system stack. The full Beep Beep system allows for multiple nodes to range in sequence. In this simplified version, only two nodes range with one another: two nodes a and b emit ranging signals one after the other, each recording both signals—the signal they emit and the one they receive from the other node. Both nodes determine the arrival times of both signals. The implemented approach assumes that both nodes are sampling at the same frequency f_s, and that their microphones are the same distance K from their speakers. The distance x between the devices is determined by

$$x = \frac{v_s}{2f_s}((S_{2,A} - S_{1,A}) - (S_{1,B} - S_{2,B})) + K \tag{3.1}$$

where $(S_{2,A} - S_{1,A})$ is the number of samples between the two ranging signals detected by node A, and $(S_{1,B} - S_{2,B})$ is the number of samples between the two ranging signals detected by node B. This description assumes signal 1 is emitted by node A, and signal 2 is emitted by node B.

The original implementation of Beep Beep used a linear frequency chirp as the acoustic signal for ToF estimation. In this implementation, the Pseudo-random noise (PN) sequence software implemented by the ENSBox was modified to work with the platform and the software (PN sequences are discussed in Chapter 2, Section 2.6.6. The Gumstix implementation of Beep Beep leverages the sub-sample accuracy of the signal detection presented by Girod (2005) with the simplicity and low time synchronisation requirements of the actual Beep Beep algorithm (Peng et al. 2007). The Beep Beep algorithm reduces complexity as it does not require explicit time synchronisation between devices, unlike Girod's highly time-synchronised ToA approach (2005).

3.3.1 Ranging experimentation

The experimentation with the Gumstix platform took place in Westwood Plaza, in the same place as the Mica2 and ENSBox experimentation. Two nodes were raised 1m from the ground on tripods (as in Figure 3.22), and one node remained static whilst the other was moved. In the ranging implementation, both nodes were required to emit ranging signals. The node that was moved started 0.5m away from the other node, and was moved away in increments of 0.5m up to 10m, then in increments of 1m up to 20m, and finally increments of 5m up to 30m (the observed maximum operational range). This made a total of 31 ranging experiments. At each distance, the ranging process was carried out 5 times, with the resulting distance estimate recorded. Since five measurements had proven to be sufficient for high precision and accuracy on the ENSBox ranging, it was decided that as the algorithm shared the same ranging signal generation and detection routines, it would also be suitable for the Gumstix.

Figure 3.23 shows error at each distance measured. Whiskers represent one standard deviation of error. In most cases (all but 16 and 30m), the precision of the ranging mechanism is high. The mean standard deviation was 0.48cm, with standard deviations for all distances except 16m and 30m coming in at 0.55cm or lower. For 16m, the standard deviation was 3.24cm and for 30m it was 4.92cm (which was the maximum standard deviation).

It is interesting that the both the Gumstix and ENSBox experimentation showed most error at 30m. This may be an indication of an environmental effect present in the experimental area, such as a consistently strong multi-path area, or area of high ambient noise. Because the 16m mark was not measured on the ENSBox, this may just be a coincidence. The range estimation does not show the same error at 0.5 and 1.0m that the ENSBox showed, but the output power of both platforms was different (see Section 3.5), so they cannot directly be compared. As with the ENSBox in Section 3.2.1, the error increases with distance. However, temperatures were not gathered for this experimentation so it is unclear whether the effects were comparable to the ENSBox results. In any case, the error is under 25cm at 30 metres (less than 1% error).

3.3.2 Discussion

With minimal development, it was possible to build a highly accurate ranging solution from material presented in the literature, and readily available software components. The ranging implementation

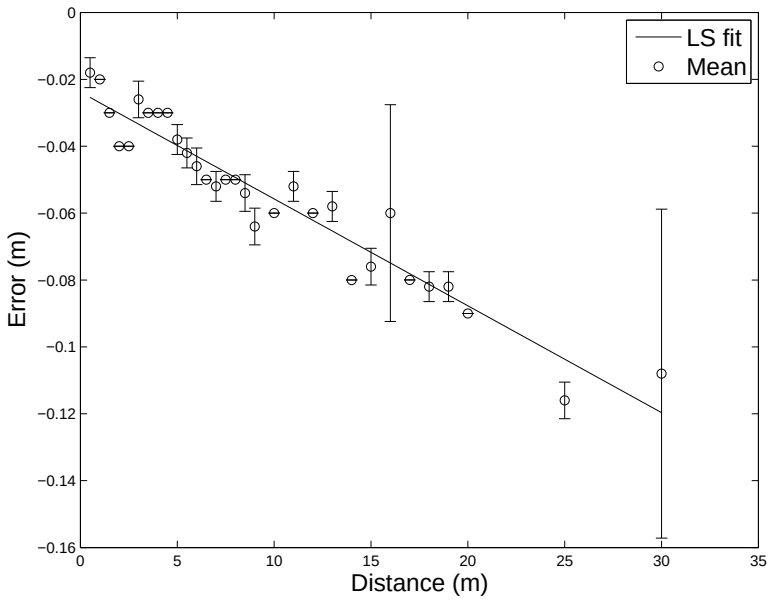


Figure 3.23: Range error vs. distance using the Gumstix-based ranging platform. Whiskers represent one standard deviation of error.

used in the Gumstix borrows heavily from the ENSBox implementation, and therefore inherits its high precision performance (mean standard deviation of 0.48cm). This shows the gains that can be made in having an accurate signal detection technique, based on correlation rather than energy. Despite the related processing requirements, this is clearly the better choice over energy-based approaches such as those seen in the Mica2. To quantify the improvement between the Mica2 and the Gumstix platforms: the standard deviation (precision) went from 20cm on the Mica2 to 0.48cm on the Gumstix, and the operational range went from 8m to 30m (a 73% improvement). It would be expected now that even using lateration for 3D localisation, the performance would be vastly improved. Recalling the simulation performed on the Mica2 experimentation in Section 3.1.6 (particularly Figure 3.13 on page 77), a standard deviation of error around 0.5cm would yield positional error of below 10cm where the Mica2 may have seen up to hundreds of cm of Euclidean positional error.

The implementation of Beep Beep on a windows-based mobile phone (Peng et al. 2007) provided operational ranges of 4m to 12m (depending on ambient noise conditions), with an average standard deviation of 0.4cm to 1.4cm. The implementation presented here provides a comparable standard deviation of error and therefore precision, but with more than double the range. This should not be taken as a direct improvement however: given the comparable precision of the signal detection algorithms, the output power of the signal is seemingly the only factor which could affect range. The relative output powers between the three devices evaluated in this thesis are discussed later in this section.

It could be argued that part of the disparity in operational range performance between the three devices is related to output power of the ranging signals of each platform. As noted in Chapter 2, (Kwon

Table 3.5: Output power by platform at 0.5m

Platform	Output Power (dB)
Mica2	60
Gumstix	92
ENSBox	100

et al. 2005) added a custom speaker to the Mica2 in order to increase range, although they found this affected the precision of the ranging mechanism (due to using the built-in Mica2 tone decoder). The main difference between the Gumstix and ENSBox platforms was in hardware: the ENSBox used a powered amplifier to increase the output power of the ranging signal, whereas the Gumstix platform used the only the output provided by the line-out of audio card.

In an attempt to quantify the difference between relative output powers of the three platforms, an experiment was set up whereby a node was raised 1m from the ground, and its speaker aligned with a sound level meter (model from radio shack). The device was positioned 0.5m away from the speaker of the node. There was up to 0.01m difference in the 0.5m measurement between different platform experiments. The ambient noise was observed to be 50 dB in the experimental environment (a 7m by 7m room). The results are shown in Table 3.5.

It is important to note that the absolute values of output power may not be reliable, as the indoor environment may have increased or decreased these due to reflection and signal interference. However, the relative difference between the output powers is clear to see—the Mica2 has a significantly quieter output than the the Gumstix and ENSBox. Regardless of the techniques and algorithms being used to modulate and process the acoustic signals (to estimate distance), a longer operational range can be expected from a speaker with higher power output (as the signal to noise ratio will be better over a longer distance). Although, the ENSBox example seemed to indicate that when a node is required to sample its own signal, this can be a limiting factor. In this case, gain may need to be adaptively altered.

The contribution to the chapter (and the thesis) of this section is the development of the proof-of-concept Gumstix ranging platform, which show that a middle ground between ranging performance and platform complexity is possible. This makes the barrier for accurate self-localisation lower in terms of hardware complexity, potentially enabling accurate 3D localisation for applications which previously could not *afford* it.

3.4 Summary

This chapter began by motivating the need for 3D self-localisation for a certain class of localisation applications with a fine-grained accuracy requirements. Acoustic ranging (and potentially angle estimation) was determined to be the most accessible way to provide accurate input data for the self-localisation process.

With this in mind, acoustic ranging and 3D localisation was evaluated with respect to several sensing platforms and implementations. Three ranging algorithms, based on RToA, ToA and TWR were evaluated on three platforms of varying capability: the Mica2 (COTS, low capability), the ENSBox (custom-made,

high capability) and the Gumstix (custom-made, high capability). The pre-existing implementations of acoustic ranging Mica2 and ENSBox platforms were characterised in terms of operational range, accuracy and precision through in-situ experimentation. Subsequently, 3D localisation was performed using lateration for the Mica2 platform, and the non-linear multilateration algorithm that is part of the Acoustic ENSBox's software. It was found that the less-capable Mica2 and its associated ranging algorithm had a short operational range (8m outdoors, 0.5m indoors) and was low precision (20cm standard deviation), and this caused significant positional error when used in conjunction with the 3D lateration calculation. The effects of node placement and low precision were investigated in simulation based on the inaccurate 3D localisation performance. This showed high GDoP values can increase positional error by an order of magnitude (tens vs hundreds of cm positional error).

In comparison, the customised Acoustic ENSBox V2 platform was found to perform significantly better: an operational range of at least 40m (up to 100m in actual localisation experimentation) with high precision (average 0.42cm standard deviation). Over four realistic outdoor 3D localisation experiments, the average positional error ranged between 0.20cm and 0.37cm, with a maximum error of 0.78m. The deployment densities ranged from one node every 4.37m^3 to one node every 22.55m^3. The excellent performance of the ENSBox comes at the cost of custom hardware that was designed for specific acoustic sensing applications, in addition to a highly accurate time synchronisation requirement. This does not make the approach general to other applications which may desire fine-grained 3D self-localisation.

The vast difference in performance motivated the creation of a new, proof of concept node that supported acoustic ranging, based on the Gumstix hardware platform. The platform was built mainly from COTS parts, except for the speaker block which was borrowed from the V2 ENSBox. The ranging implementation was based on the Beep Beep algorithm (Peng et al. 2007), but using the signal detection routines borrowed from the Acoustic ENSBox ranging implementation. In experimentation, the platform's operational range was 30m, with high precision that was comparable with the ENSBox (average standard deviation of 0.48cm). The Gumstix platform was not evaluated with respect to 3D localisation, therefore it is not possible to compare localisation performance with both the ENSBox and the Mica2 experimentation in this Chapter. Whilst the Gumstix lacks the angle of arrival measurements of the ENSBox, its accuracy and operational range is such that one would expect it to perform accurately with lateration (compared to the Mica2), and a 3D range-only, non-anchor-dependent localisation algorithm, such as robust quintilaterals (Mautz et al. 2007) as discussed in Chapter 2.

The advantage that the ENSBox and Gumstix's extra processing power gives is that more accurate signal detection can be performed, than on the Mica2. Accurate signal detection requires a correlation-based approach, rather than the energy based approach used by the Mica2. Implementing a correlation-based approach with similar accuracy on the Mica2 would be practically impossible because of the lack of memory (the Mica2 only has 4kB total RAM) and disparity in sampling rate: the Mica2 can sample at a maximum of 17.7 kHz, which is not good enough for sub-centimetre accuracy (sound travels 1.9cm in between each sample).

For the class of 3D applications motivated at the start of this chapter, the Mica2 is not sufficient to support fine-grained 3D localisation, even in a lateration experimental set-up. The ENSBox is suited to the fine-grained, 3D self-localisation requirements of the marmot localisation application. This is because

it is a custom self-localisation solution, designed specifically for acoustic source localisation and other acoustic-related applications.

The next chapter switches focus from self-localisation to source-localisation. This comprises work based on the motivating application: the localisation of animals in their natural habitat (which requires accurate 3D self-localisation).

Chapter 4

An on-line acoustic source localisation system

Chapter 3 discussed node hardware and software requirements in order to achieve accurate acoustic self-localisation in WSNs. Accurate self-localisation is extremely important for a class of applications that use node position as input to source-data fusion. Chapter 4 focuses on this class of applications, taking *acoustic source localisation* as the motivating high data-rate application. Specifically, the application considered is the localisation of marmots in their natural habitat. (Marmots are small rodents with distinctive alarm calls that can be found in the USA.) Marmot localisation is a useful tool for the automation of scientific observations that biologists need to make in order to understand marmot behaviour based on their calls. These observations are valid for other biologically-motivated source localisation systems for different types of animals and birds.

Firstly, a design for an on-line marmot localisation system was developed and implemented by the author. For this system, an existing set of components to perform acoustic localisation were integrated, producing a prototype on-line source localisation system.

Secondly, using the on-line model for acoustic source localisation as an example application, a hardware and software platform for on-line acoustic sensing called *VoxNet* was developed and deployed as part of a collaborative effort. Several individual components of the system were developed, unit tested and evaluated by the author prior to deployment. For both systems, controlled indoor experimentation was performed to benchmark key components. This was sufficient to allow for comparative performance measurement between them.

VoxNet provides support for user-interaction via a high-level interaction model which enables a variety of acoustic sensing applications. For application level programming, VoxNet uses the Wavescope stream-processing engine. Underlying components to support the Wavescope engine running on a node were integrated, such as data acquisition drivers and multi-hop network communication. In the design of VoxNet's programming and usage model, consideration was given to both the application specific and more generic aspects of the system. The VoxNet implementation required less CPU and memory than the original EmStar–only implementation, through tighter integration of existing, tested components (to reduce overhead). Minimising the resource requirements allowed for the implementation of extra functionality.

This chapter does not try to innovate in the components of the source data fusion, which includes algorithms for: event detection, bearing estimation and data fusion for position estimation. The focus here

is rather on the on-line operation of the system, and provision for human interaction and reconfigurability for the deployable source localisation system.

This chapter is organised as follows: Section 4.1 elucidates marmot localisation as an application and Section 4.2 provides an abstract overview of an on-line marmot localisation system. Subsequently, two design-implementation cycles are presented for the localisation system, in Sections 4.3 and 4.4. Section 4.5 presents micro-benchmark analysis of the two system iterations and Section 4.6 summarises the contributions of this chapter.

4.1 Application motivation

The field of bio-acoustic research is broadly concerned with the transmission and reception of acoustic signals made between animals or birds. One particular aspect of bio-acoustic research focuses on understanding the purpose behind certain acoustic signals produced by animals, and how these relate to the environment in which they are produced. Census is also an important part of bio-acoustic studies. Acoustic census counts the number of individuals present in a given audio recording, which can help the researcher estimate population size in a given area. Acoustic-based census is possible because many mammals and birds produce loud alarm calls, territorial calls, and songs that are species-specific, population-specific, and often individually identifiable (McGregor et al. 2000). As such, these vocalisations can be used to identify the species present in an area, as well as in some cases to count individuals. Acoustic monitoring for census has been shown to be useful for cane-toad monitoring (Driscoll 1998), elephants (Payne et al. 2003), birds (Hobson et al. 2002) and whales (George et al. 2004).

Recognising animal and bird calls and distinguishing between different individuals can be a difficult problem. Only a handful of ornithologists have the skill to accurately census birds in dense, tropical forests. Some animal vocalisations are impossible to observe such as infrasonic elephant vocalisations, which are out of range of human hearing (Payne et al. 2003).

Localisation of animals and birds is a key component in census since determining the position of an animal or bird based on its call can help the scientist disambiguate similar calls. Localisation can also help the scientist understand territorial behaviour of the animal or bird of interest.

The particular animal considered with respect to source localisation in this chapter is the marmot, a medium-sized rodent native to the western United States. Marmots are notable for their high pitched *alarm calls* when they sense danger. Localisation in this case is helpful for understanding where an animal was when it made a call, and why it made the call (to warn others, or protect itself). Although marmots call relatively infrequently, they are prone to *bouts* of alarm calls, where 20 or more calls are made in a single position, with an interval of around 1 second between each call. The calls are short (around 40 milliseconds) and are high energy and high frequency. This makes them ideal candidates for energy-based event detection (as discussed in Chapter 2).

4.1.1 Traditional user-interaction

Traditionally, bio-acoustic researchers gather data by attaching one or more acoustic sensors to a multi-channel recorder situated around the animal/bird habitat of interest. The researcher will attend the deployment and keep a detailed log of interesting acoustic events to aid and complement data post-

analysis in the lab.

This approach commonly results in several hours' worth of continuous acoustic recording, which must be manually examined off-line by the scientist. Analysis through listening to unfiltered audio data is enhanced with time and frequency domain data visualisation in order to find sections of audio that are of interest. A variety of domain-specific applications and frameworks for data acquisition, event detection and analysis exist. Raven (Anonymous 2008*b*) and Ishmael (Mellinger 2001) are software applications designed to help the bio-acoustic researcher visually and statistically explore the audio stream in both the time and frequency domain, and then run automated event detectors over the audio stream. When more than one channel of audio data is gathered, both software applications provide the ability to perform signal enhancement and direction estimation using beamforming techniques. Similarly, Engineering Design (Anonymous 2008*c*) provide a suite of compatible tools such as Signal, Real Time Spectrogram (RTS), Event Detector and Event Analyser, which allow automated event detection and analysis for acoustic applications. The suite of applications aims to be a general purpose event detection and analysis tool.

General purpose audio processing and recording tools such as Audacity (Mazzoni & Dannenberg 2007) and Baudline (Anonymous 2007*b*) allow the user to investigate the characteristics of audio signals, such as power and frequency content. These tools are useful, but are not integrated into any kind of processing framework. Often they can be used in combination with general purpose tools like Matlab (Anonymous 2009*f*) to perform ad-hoc processing that mimics the functionality of more specialised processing frameworks.

4.1.2 The need for in-situ automation and interaction

Any automated in-field analysis of acoustic data can potentially ease off-line analysis of the data set for the scientist. In this case, distributed acoustic sensing systems supporting on-line, real-time processing and analysis of data have the potential to be a vital research tool for the scientist in the field. Researchers can be given insight into territorial behaviour of animals and birds with a greater density of observation than would be possible using the methods described in Section 4.1.1.

4.2 Application requirements and overview

The marmot localisation system is motivated by the following scenario: the scientist wants to detect marmot alarm calls and determine the location of marmots, relative to known positions (such as their burrow locations). By obtaining the marmot position estimates as the system is running, the scientists can augment their written log-traces with pictures of animals under observation. Ideally, these pictures could be taken by automated imagers which are actuated based on the results of the position estimate provided by the localisation system.

Such a localisation system may also enable the scientist to record additional data about the current habitat conditions, such as what caused the animal alarm call, and which animal raised it. In this chapter, it is assumed that a scientist is only interested in localising one animal at any instant. (A discussion on the importance of this constraint is presented in Chapter 5.)

Such a system requires that position estimates are delivered as close to real-time as possible, in order

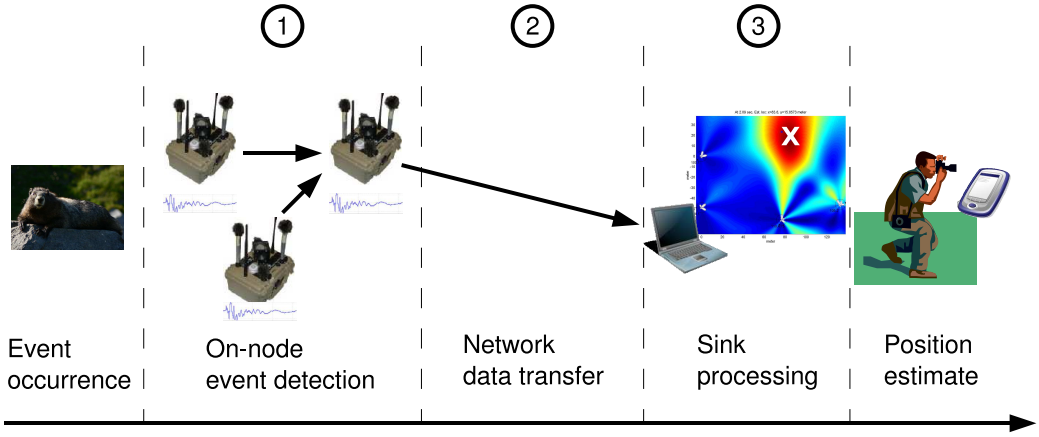


Figure 4.1: The source localisation localisation flow considered in this Chapter.

to enable actuation and image collection (either manual or automated).

Ali et al. (2007) presented a proof-of-concept distributed event detection system for this application based on the Acoustic Embedded Networked Sensing Box (ENSBox) which was discussed in Chapter 3 (Girod 2005). Their system used the on-line energy-based event detection algorithm described in Chapter 2 (Trifa 2006). The algorithm was tuned for marmot vocalisations and evaluated both in-situ with real marmots and in controlled experiments with marmot recordings.

Time synchronised, continuous, four-channel audio recordings of marmots were also gathered in-situ at each node, and combined with self-localisation results to perform source localisation off-line.

A network of Acoustic ENSBoxes were used to gather continuous, time synchronised audio over several hours. Self-localisation was also performed, giving reference positions for nodes. The gathered audio data was examined off-line by a biologist to determine the location of marmot events of interest across all nodes. This process was performed manually in each node's audio stream, rather than in an automated fashion. For each event, the Direction of arrival (DoA) estimate from each node was determined, using the Approximated maximum likelihood (AML) algorithm as discussed in Chapter 2 (Chen et al. 2002*a*). The AML results for a given event were used as input to a source localisation algorithm (along with the self-localisation results giving node reference positions). Under controlled tests (which the author contributed to) where six V2 nodes were placed in a rectangular configuration covering 35m by 60m, the data fusion yielded an RMS positional error of 0.7m when the source was inside the convex hull created by the sensors, and 2.07m when the source was outside of the convex hull (Ali et al. 2008).

The applicability of the approach above for on-line source localisation of marmots was hence demonstrated, and whilst an end-to-end system was not presented by Ali et al. (2007), the components that would be required are available. They comprise an on-line event detector, DoA estimation algorithm (AML) and DoA based localisation algorithm. As per its description in Ali et al. (2007), the conceptual flow for such an on-line, end-to-end localisation system is as follows (Figure 4.1): an acoustic event is detected within the network. Subsequently, all nodes send data corresponding to a detection to the sink.

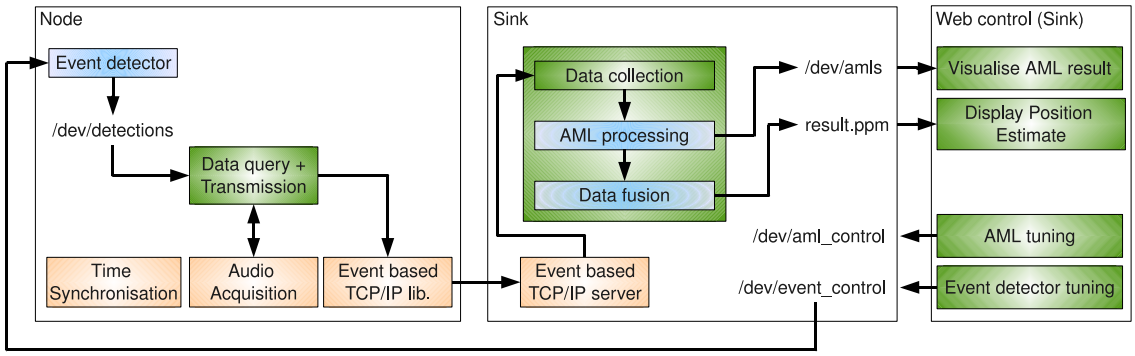


Figure 4.2: Top-down system design for the first iteration of the acoustic localisation system. Each box represents a process, with green boxes identifying components built entirely by the author. Orange boxes are EmStar libraries and blue boxes indicate existing components that were modified by the author.

The sink aggregates all the data corresponding to the acoustic event and determines the DoA of each detection. Along with the node positions, the resulting DoAs are used as input to the source data fusion, which yields a 2D position estimate.

In addition to this, the localisation system requires a way for the user to reconfigure or tune certain system parameters during operation, such as the event detector parameters (as in Chapter 2). Since the target users of this system are non-expert, it is crucial that the system be easy for them to deploy, initialise, and configure/reconfigure.

The first iteration of a system fitting the requirements above is presented in Section 4.3, based on existing components integrated to create an end-to-end source localisation system.

4.3 Top-down marmot localisation system design

The aim of the first system design exercise and its accompanying implementation and deployment was to integrate the source localisation components used by Ali et al. (2007) into an end-to-end, on-line localisation system. The physical architecture of the system was centred around a network of nodes in direct communication with a single sink node. The nodes used were Acoustic ENSBox sensing platforms (as described in Chapter 3), and the sink was an x86 laptop which was designed to allow the user to interact with the system. Both platforms ran Linux as their operating system. (The overall design and processing flow of the on-line source localisation system can be traced using Figure 4.2). Each node in the network ran the same event detection application (Event detector in Figure 4.2), and the sink ran a position estimation application (Data collection, AML processing and Data Fusion). The node-side application was responsible for event detection and transmission of detection data to the sink. The sink application was responsible for reception of detection data and its processing through the AML algorithm and data fusion processes. The sink also provided a point of interaction for the user, offering some control tools and visualisation of position estimates.

The system as a whole was built using the EmStar software development framework for WSN applications (Girod et al. 2004, Girod et al. 2007). EmStar provides libraries for standard WSN functionality,

such as neighbour discovery, link estimation, message sending, sharing of state in a distributed manner, data acquisition, and more. Each coloured rectangle in Figure 4.2 represents an individual process—green boxes are components built by the author to integrate the on-line application; orange boxes are library components. Blue boxes are components written to support the work presented in Ali et al. (2008), and subsequently modified by the author inn order to integrate them into the on-line system.

A whole EmStar application is the combination of several communicating processes, which are managed by the *Emrun* application. *Emrun* is configured by a user-defined *run file* that provides the location on disk of all of the processes that must run, and command line options to be used when each process is invoked, and how *Emrun* must react if any processes crash (to either restart the process or not). Although *Emrun* is designed to support executables written using the EmStar framework, it can be used to start and restart arbitrary processes (if they are not required to communicate). The EmStar framework enables Inter-Process Communication (IPC) by using a software framework called the Linux Framework for User-Space Devices (FUSD) (Elson 2002).

Using multiple processes to run EmStar-built software creates significant overhead compared to a threaded application (processes are more expensive in terms of system resources than threads). In addition, the FUSD implementation incurs some overhead for IPC. However, it provides great benefit, as it enables modularity in application development, meaning faults can be isolated to processes. This is vital to ensure system robustness during development and testing of WSN software (Girod et al. 2007). The node and sink applications are now discussed.

4.3.1 Node-side application

The node side-application consisted of several library components and two application level executables—an event detector executable and a detection transmission executable. Communication between components was implemented using EmStar devices. These are identified in Figure 4.2 with the prefix `/dev/`. In the remainder of Section 4.3, the application level components are firstly discussed in more detail, followed by the library components.

Event detector

The event detection process is responsible for detecting events of interest and then communicating the detection times to the detection transmission component via a *detection* device. Whenever a detection is triggered, the event detection component records the start and end times of the detection, and places them on the *detections* device, signalling that new data is available. The event detector was taken from an EmStar implementation used by (Ali et al. 2007), which showed good performance and suitability for on-line event detection of marmots in-situ.

The only modification required was to add the event detections device to the event detector process, to enable communication with other processes.

Detection query and transmission

The detection transmission process is responsible for querying the relevant data segment from the audio acquisition component whenever new detections arrive, and sending this data to the sink. This is accomplished using EmStar library functions to listen on the detection device, query data from the audio

acquisition layer, and send the data via a Transmission Control Protocol (TCP) connection to the sink.

For each triggered detection, the detection transmission process requests 4096 samples of audio per channel from the audio acquisition component, centred around the detection time. This corresponds to to 90ms of audio or 32 kB of raw data (four channels, sampled at 48 kHz with 16 bit samples).

The detection size of 4096 samples/channel was chosen as a trade-off between the capturing the acoustic signal of interest (around 40ms for a marmot call) and the smallest window size on which it is best to perform an AML (for processing complexity reasons). As noted in Chapter 2, the AML result suffers from edge effects induced by the Discrete Fourier Transform (DFT), which can be somewhat addressed by providing adequate padding either side of the signal of interest.

In addition to the application level components, several *driver* components are required to support the application running on the node. These are all provided by previously implemented and tested EmStar executables, and are described below.

Network communication

As noted in Chapter 3, the Acoustic ENSBox nodes use 802.11 radios in the ad-hoc *Independent Basic Service Set (IBSS)* mode. Communication between nodes and the sink is provided using TCP/Internet Protocol (IP).

A feature specific to the prism2 chipset on the particular 802.11 wireless cards used, called *Pseudo-Independent Basic Service Set* mode, is enabled to eliminate network partitions due to different parts of the network converging on different ad-hoc cell IDs. Whilst security is not a major consideration in the network design, the *Pseudo-Independent Basic Service Set* mode provides security by obfuscation, as it allows only nodes with a particular version of the driver and specific chipset to join the network, thus greatly reducing the threat of potential intruders. However, this work-around is not intended to be a complete security solution. If security is an issue, then a more formal approach would need to be taken.

Time synchronisation

Time synchronisation is important in order to correlate detections across a distributed network of nodes. The time synchronisation component used is a combination of Reference Broadcast Synchronisation (RBS) (Elson et al. 2002) and a protocol to propagate global time from a node, called *gsync* (which could potentially use a GPS receiver as its time source (Karp et al. 2003)). The time synchronisation service allows conversion between global and local timestamps via an API function. At a high level, it does this by performing a linear fit based on the x most recent timestamp pairs it has seen and uses this to predict one timestamp (either local or global) given the other. The sink does not require time synchronisation with the nodes, because its only function was to gather the data.

Audio acquisition

An audio acquisition server, called the *audio daemon* takes responsibility for the acquisition and interleaving of audio samples taken from the four channels on the Acoustic ENSBox's VXP440 soundcard, as well as the presentation of these samples to the application layer. Software interleaving of audio samples from the four channels is required because the hardware audio codecs are not synchronised on a per-sample basis within the sound card. The audio acquisition component maintains a ring buffer of

the most recent ten seconds of audio gathered, and provides a query interface API, which allows users to request copies of samples from the ring buffer. This functionality is used by the detection query and transmission executable, as previously described. The design and development of the audio server are discussed in detail by Girod (2005).

Self-localisation

A self-localisation component was developed for the Acoustic ENSBox by Girod et al. (2006). This provides accurate relative position estimation, based on acoustic range estimates between nodes. The self-localisation component comprises two components: acoustic ranging and multilateration (both discussed in Chapter 3). The relative node coordinates established by the self-localisation component are made available to other processes in the EmStar application through a FUSD device.

Although the sink does not directly participate in the localisation process, it runs the localisation components so that the position estimates can be propagated to it. This means the sink can use the data as input to the data fusion.

4.3.2 Sink-side application

The sink side application is divided into two executables, one (data processing) which implements DoA estimation (using the AML algorithm) and the data fusion (pseudo maximum-likelihood), and the other (interaction) which supports the interaction between the system and the user. These executables are described below.

Data processing

Support for data reception over TCP/IP is built using EmStar libraries for an event-based TCP server. Each node opens a TCP connection with the sink to send a detection, and disconnects after successful transmission. As the individual TCP segments of the 32 kB detection message from a node are received, they are copied into a buffer. When the full 32 kB detection message is received, this is pushed into a queue for AML processing.

The AML processing thread takes items from the AML queue and processes them in turn. After the processing of each AML result, a 0.5 second timer is set, and the AML result is placed into a data fusion record structure, to prepare it for the data fusion. If any other AML results arrive before the timer expires, they are added to the same data fusion record, and the 0.5s timer is re-started. If an AML result arrives from a node which is already represented in the data fusion record, the new AML replaces the previous one. The data fusion record contains all of the data required to perform the data fusion—the ID, position of each node, and its AML result. When the timeout occurs, the current state of the record is copied and placed on a data fusion queue (the current state is subsequently reset). A data fusion thread then takes records from the queue and uses them as input to the localisation algorithm. Whenever an AML result is gathered, this data is placed on a device, meaning another process can read the AML result and visualise it, for example. This means the correctness of AML results can be analysed in isolation from the data fusion/position estimate phase (in practise, a separate *AML-only* visualiser was used to view the output of a single node's detection). The settings of the AML algorithm, namely the frequency bands to be used in the computation, are tunable via a *control device* exposed by the

application (described in 4.3.2 below).

Finally, the *data fusion* algorithm takes the results from the data fusion queue and the known positions of the nodes (from the self-localisation device) and combines them. The algorithm output is a pseudo-likelihood map (as a `.ppm`) file which is then displayed to the user.

Interaction

In order to provide interaction for the user, a web interface was developed, using EmStar Hypertext Markup Language (HTML) libraries. This was possible because both the nodes and sink ran lightweight Hypertest Transmission Protocol (HTTP) servers provided as EmStar library services.

The web interface provides a web page that the user can view on the sink (laptop), giving a visualisation of the individual AML results that went into a given data fusion, as well as the resulting *pseudo-likelihood* map. The web-interface also provides a way to input parameters for both the on-node event detector and sink-side AML processing controls. In the back-end, event detection configuration is provided by an existing EmStar application for remote, synchronised recording called *netrec (network recorder)*. A sink side process maintains the status of all nodes running event detection and recording software in the network. This includes event detection settings, local hard disk space and recording settings. A node side process responds to commands to start and stop recording, as well as to change event detector settings. The sharing of configuration state between nodes and sink is provided by a library EmStar service, called *statesync* which provides efficient, log based state sharing between nodes in small networks.

4.3.3 Discussion

The system produced was intended and evaluated as a demonstrator. It was verified that the various components worked together and that the system was capable of presenting results on-line; therefore, no quantitative results are shown. The system was demonstrated on two separate occasions, firstly at the IPSN 2007 conference, where an eight-node network was set up over a 30 metre square area, and secondly at the CENS research review in 2007, where an eight-node network was set up over a 15–20 metre square area. In both demonstrations, a controlled acoustic source was used to validate the performance. The acoustic event used was a dog-whistle, chosen because of its strong dominant frequencies. The original energy band chosen for marmot frequencies was a continuous frequency band between 4.64 to 7.74 kHz. So, in order to tune the event detector and AML, the whistle was recorded and its frequency spectrum analysed. Four frequency bands were chosen to compute energy across for the dog whistle: 6.96–7.74 kHz, 13.16–15.48 kHz, 18.58–20.13 kHz and 20.9–23.22 kHz. The last three bands were harmonics of the first, dominant frequency band. The event detector was configured to monitor only these frequency using the web interface.

The integrated system produced demonstrates that an end-to-end system for source localisation was viable, and provided a reference to compare future performance to (see Section 4.5). However, in-situ non-expert user configuration was not explicitly considered, and neither was ease of deployment and initialisation. It would be difficult for a domain scientist to change the components or re-write software. The visualisation components were fixed in their implementation, and limited (AML and position estimation map). Additionally, the system did not work over multiple network hops, meaning

that single hop connectivity with the sink had to be guaranteed for all nodes in order for the system to function correctly.

The second iteration of the acoustic source localisation system formed a proof of concept for a hardware and software platform for on-line acoustic sensing, called VoxNet. The on-line source localisation application was used as an example application for the platform. Several new issues which were not raised in the development of the first proof-of-concept were addressed: multi-hop networking, interactive usage and high level application creation and dissemination.

For the VoxNet system, the author took responsibility for the development and evaluation of key system components, specifically the real-time data recording functionality (spill to disk), an interactive system shell to send commands to nodes and receive status updates (the WaveScope Shell), and a file dissemination protocol. The author also developed a framework for experimental data gathering, built into the WaveScope shell. This framework was used by the author to carry out several data gathering experiments, both in-situ, and in controlled experiments (described in Chapter 5). Finally, the author compared the resource consumption between the EmStar and VoxNet sytems (Section 4.5.1).

4.4 A bottom-up marmot localisation system design

The informal, qualitative evaluation of the first iteration of the system was sufficient to show that the components could be integrated into an on-line system, and also could be re-tuned whilst in operation. However, the system was not easily configurable, and did not support other features that would be expected by a domain scientist, such as raw data collection. It was also only a single-hop network architecture, limiting the deployment scale achievable.

Working with real users and a concrete application enabled refinement of objectives with respect to the marmot localisation system, both in terms of the capabilities of the system and the desired user experience. As a result of these experiences, the next iteration of the marmot localisation system was designed and implemented: *VoxNet*, a hardware and software platform for distributed acoustic sensing.

The general aim of VoxNet is to support high data-rate sensing applications, using a design motivated by bio-acoustic applications. The contributions that VoxNet provides beyond the original on-line marmot localisation system are:

1. A platform capable of rapid deployment in realistic environments for bio-acoustic applications
2. A high level programming interface that abstracts the user from platform and network details, whilst compiling into a high performance distributed application
3. The definition of an interactive usage model based on run-time installable programs, with the ability to run the same high level program seamlessly over live or stored data.

In VoxNet, software flexibility is a system design requirement. It is important to be able to facilitate multiple application functionality as well as reconfiguration and tuning of applications running in the field. For example, concurrently with the localisation application, a researcher might want to archive all the raw data gathered without disturbing the running system. A feature called *Spill to Disk* was developed to support this.

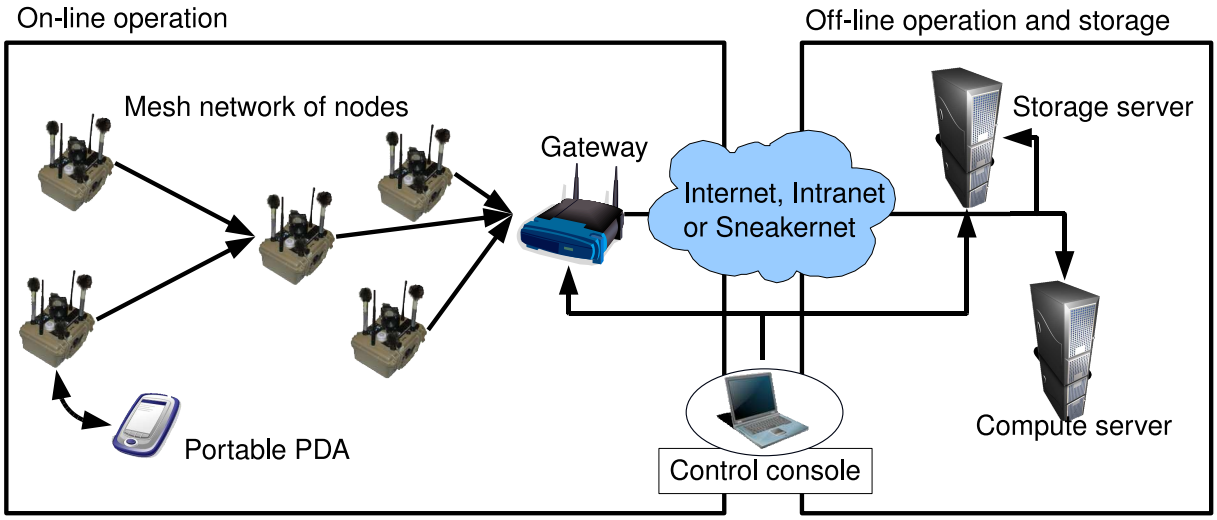


Figure 4.3: The VoxNet system architecture, showing both on-line and off-line operation contexts. The control console represents the user interface to both contexts.

Distributed VoxNet applications are written as a single logical program in the Wavescript macro-programming language (Girod et al. 2007), abstracting the programmer from the particular details of the network and particular hardware platforms. Other well-known WSN macro-programming approaches include TinyDB (Madden et al. 2005), Kairos (Gummadi et al. 2005) and Regiment, the forerunner to Wavescript (Newton et al. 2007). These programs can operate over a combination of live and static data, residing in a distributed system of sensors and back-end servers. This model enables users to tune and further develop applications during pilot deployments, and enables the system to be used as an interactive measurement tool while it is deployed. This is important for many short-term scientific deployments, because it allows a scientist to immediately explore newly observed phenomena.

4.4.1 System architecture

Figure 4.3 shows diagrammatically the VoxNet system architecture—a framework in which programs can be written, compiled, and disseminated, and the results can be archived and visualised. VoxNet is made up of several hardware components: a network of nodes, a gateway, a control console, the compute server and the storage server.

Nodes form the main part of the on-line operation of the system, sensing and potentially processing audio data. The *gateway* provides a connection to relay traffic between the control console and the network of nodes. The *control console* provides a unified interface for both the on-line and off-line portions of the platform. It hosts the interaction tools, and acts as a sink to the programs running in the network. Results and diagnostic data are returned to the control console for display and visualisation. A *PDA* may also be used to visualise data from network streams whilst mobile in the field. The *storage server* archives data acquired by nodes for later retrieval and processing. The *compute server* provides a

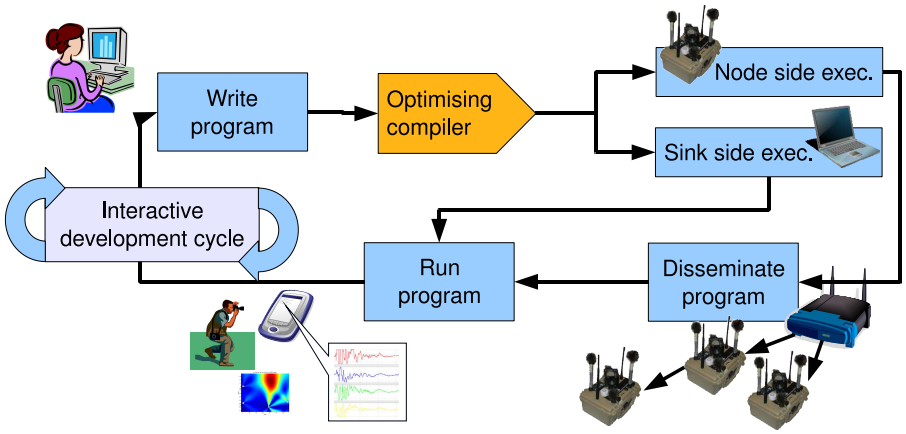


Figure 4.4: The on-line, interactive development cycle for VoxNet.

centralised unit for off-line data processing and analysis.

The full VoxNet architecture operates in two contexts: on-line and off-line.

Interaction model

When running on-line, Wavescript programs are created, compiled and disseminated to the network via the control console. Data arriving at the control console can be visualised as well as archived to a storage server. This data includes application specific streams, as well as logging data and operational statistics. When in off-line operation, streams of results and offloaded raw sensor data can be archived to a storage server and later processed off-line, using the same user interface. Applications that would previously have run on the control console and nodes, can be run on the compute server with data queried from the storage server.

Section 4.4 concentrates mainly on the design, implementation and evaluation of the on-line operation context of VoxNet. Off-line operation is considered in more detail in Chapter 5. This is because the marmot localisation application is focused on on-line localisation. It is therefore most important to address firstly the on-line aspects of the system.

4.4.2 On-line VoxNet components

The VoxNet software stack is made up of three integrated layers:

- The **high level application layer**, provided by the Wavescope stream processing engine, which handles compilation, optimisation, and operation over the network.
- The **distribution and interaction layer**, which provides a collection of sub-application level control and visualisation features important for tracking resource availability, troubleshooting and usability in the field and in off-line computations.
- The **platform drivers and services layer**, which supports the operation of the interaction and application layers on the specific hardware used in VoxNet. This layer includes core components

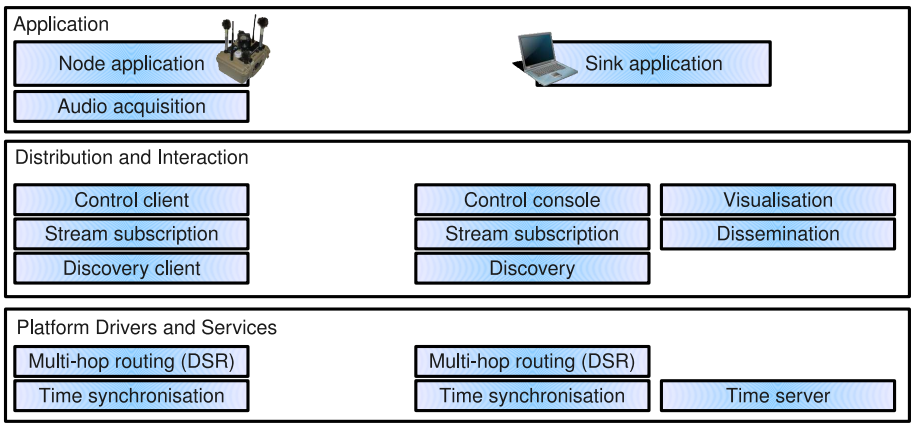


Figure 4.5: The layers in the VoxNet software stack, shown for both node and sink.

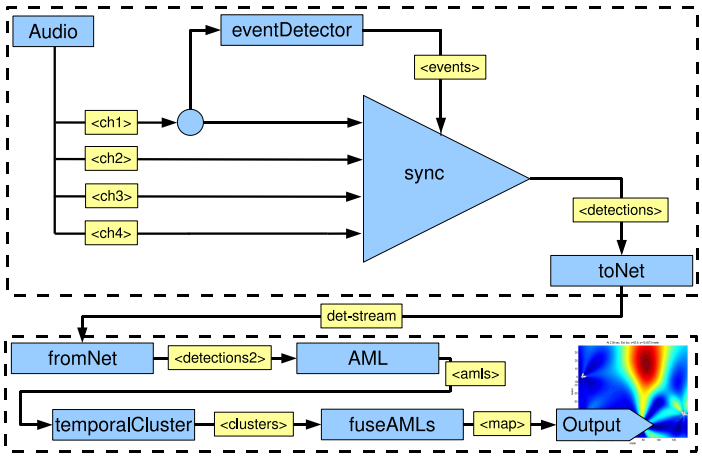
such as time synchronisation and multi-hop network communication.

The control/visualisation tools and platform drivers/services were implemented using the EmStar framework, allowing re-use of existing (and tested) components, as well as rapid development of new tools and services. These components are described in greater detail in the remainder of Section 4.4.

4.4.3 Application layer

In VoxNet, applications are run using the Wavescope stream-processing engine. The Wavescope project is an ongoing effort which started in 2006 at MIT with an aim to develop a stream processing system for high-rate sensor networks (data sampling at rates of hundreds to tens of thousands of Hertz) (Girod et al. 2007, Girod et al. 2008). Wavescope represents streams of data in units called *SigSegs*. These are windowed segments of data with associated meta-data such as sample frequency, window size (in samples) as well as a single timestamp, for the first sample in the window. Wavescope does not timestamp data on a per-sample basis for high-data rate applications because of the overhead involved in both processor time and memory requirements. Instead, Wavescope assumes that data sources provide contiguous streams of samples; this means a window of data can be indexed correctly knowing only the timestamp of the first sample.

Wavescope programs are written using the custom stream-processing programming language, *Wavescript*. Similar to other stream-processing languages, *Wavescript* structures programs as a set of communicating stream operators (also known as kernels or filters). To write a program, the user writes a *script* that concatenates stream operators to form a directed graph. Stream operators consume one or more input streams, and produce one or more output streams. During compilation, the Wavescope back-end performs a depth-first traversal of the Wavescript dataflow graph, meaning that emitting data from one operator to another corresponds to function call make from the upstream operator into the code for the downstream operator. Wavescript programs are written using a functional programming paradigm for the most part. However, unlike pure functional programming languages, Wavescript allows side-effects (stateful variables). Syntactically, Wavescript is most similar to the ML programming language. Wavescript




```
// node application
(ch1, ch2, ch3, ch4) = Audio(44100)
events = eventDetector(ch1)
detections =
  sync(events, [ch1, ch2, ch3, ch4])
BASE <= toNet(detections,"det-stream")
```

```
// sink application
detections2 = fromNet("det-stream")
amls = AML(detections2)
clusters = temporalCluster(amls)
map = fuseAMLs(clusters)
BASE <= map
```

Figure 4.6: The high-level graph of the WaveScope localisation application. Operators and named streams are shown. The corresponding node and sink-side code is shown on the right hand side of the figure.

allows users to write custom stream operators directly in Wavescript to augment the existing library of operators.

The Wavescope engine used in VoxNet requires that users write both the sink and node side of the executables, and define where network communication takes place. Two Wavescope operators are provided to allow this functionality, `toNet()` and `fromNet()`. `toNet()` maps a local stream onto a named network stream, and `fromNet()` subscribes to the named network stream, outputting it as a local stream. The Wavescript compiler converts the high-level application representation into efficient, low-level C code for either x86 or embedded architectures (such as ARM). The resulting Wavescope executables are run on the VoxNet platform using *Emrun*.

The high level data flow graph for the Wavescript implementation of the marmot localisation application is shown in Figure 4.6. Stream operators are shown in blue, and named stream variables (that flow between operators) are yellow with text in ⟨angle brackets⟩. Audio data acquired by a node is directed into Wavescope by the `Audio` operator as four channels (`ch1` to `ch4`). Stream `ch1` is passed through the `eventDetector` operator, which produces a stream of tuples called `events`, which correspond to the start and end times of marmot calls. The `events` stream is passed through the `sync` library operator, along with the four original audio channels. The `sync` operator outputs a four channel stream of raw data (`detections`) which corresponds to the data ranges specified in `events`. The `detections` stream flows through the `toNet` operator into a named network stream (*det-stream*).

At the sink, the `fromNet` operator receives data from any nodes which are publishing *det-stream*, outputting a `detections2` stream. `detections2` flows through the `AML` operator, producing `amls`—a stream of angle of arrival estimates. `amls` flows through the grouping operator `temporalCluster`, which time-correlates AML results, producing a stream of grouped data (`clusters`). This flows through a data fusion operator to produce a stream of position estimate maps, which are output to the user.

A vital part of integrating Wavescope into VoxNet is Wavescript's Foreign Function Interface (FFI). This allows Wavescript operators to be bound to library code written in C. For example, the FFI allows Wavescript to ensure that extremely intensive kernels of computation (for example, Fourier transforms) are implemented by the appropriate C or FORTRAN implementations (such as FFTW or LINPACK).

The FFI has been extremely useful for supporting Wavescope in VoxNet. For example, the network operators (`toNet` and `fromNet`) are wrapped calls to the node's network subscription server (see Section 4.4.4), implemented using the FFI. The FFI was also used to integrate the audio acquisition component as a *live* data source. This meant that acquired data could be pushed to the compiled Wavescript code, where it was viewed as a stream (via the `Audio` operator). Unlike other statically linked C libraries, the audio acquisition component was compiled directly into the node-side executable. This was because the audio driver could not afford the speed and memory overhead of using an IPC mechanism to transfer the data to the node-side executable. This was achieved by using two concurrent threads in the node executable with a buffer in between. One thread ran the audio acquisition component, and put audio data onto a buffer. The result was read by the other thread, which was the main Wavescript application.

The Wavescript language and compiler was developed in prior work (Girod et al. 2007). VoxNet builds on this, extending the functionality to support applications running with live data inputs, on an embedded platform, and over a wireless network. VoxNet is the first embedded target for the Wavescript compiler, and developing the VoxNet back-end motivated many new features and optimisations. Implementing the animal localisation application using the Wavescript programming language and optimising the compiler resulted in a 30% reduction in processor load and 12% in memory usage, compared with the first iteration of the system (see Section 4.5). Prior work on Wavescope focused on Wavescript and single-node engine performance, but the VoxNet platform motivated significant new developments. These developments formed the basis of the distribution and interaction layer, and are described below.

4.4.4 Distribution and interaction layer

The distribution and interaction layer is responsible for allowing Wavescope programs to flow in a multi-hop network, as well as providing sub-application control and diagnostics. To support this, the distribution and interaction layer offers the following functionality:

- *Network stream subscription server* to create, publish and subscribe to data streams.
- *Discovery service* to discover nodes, maintain status and connection information at the sink.
- *Wavescope Shell (WSH)*—a command line interface to the control console, allowing commands and binaries to be disseminated to the network.
- *Wavescope application dissemination mechanism* to disseminate compiled binaries to the nodes in the network via network streams.
- *Stream visualisation mechanism* to visualise data travelling over networked streams.

This distribution and interaction layer represents a large amount of the author's individual contribution to the VoxNet project in terms of design and implementation. The individual elements are now discussed, with exception of the visualiser, whose operation is secondary to the system's functionality.

Network stream subscription

To support the flow of Wavescope streams across network boundaries in VoxNet, a TCP/IP based network stream abstraction using a publish-subscribe model was developed. The network stream abstractions are used for all types of communication in the VoxNet system, including visualisation, log messages, control messages and pushing new compiled programs. Networked streams are conceptually the same as Wavescope streams, flowing uni-directionally from one point to another. Additionally, a published networked stream can flow to multiple clients.

To enable the network stream abstraction, each node (as well as the sink) in VoxNet runs a subscription server, which is responsible for creating new streams to be published, managing subscribers to each published stream, and subscribing to streams published by other nodes (or the sink). Streams are identified by a descriptive name (for example, *detections*) which must be unique to a given subscription server (but is not required to be globally unique).

Networked streams are intended to be 100% reliable to preserve the flow of data in Wavescope applications. To ensure reliable data transmission, the stream abstraction is built upon TCP/IP. The implementation of the network stream abstraction was built using the EmStar event driven TCP server and client libraries as base functionality (these are the same libraries used in the first system iteration). The network stream library offers an API to users, shown below:

- `create_subscription_server(options)`
  creates a new subscription server, with user-defined options.
- `register_stream(stream_name, new_client_cb, options)`
  registers (or publishes) a new stream with the subscription server. `new_client_cb` allows the user to provide a callback function to be triggered when new clients subscribe.
- `subscribe(stream_name, server_address, data_received_cb)`
  subscribes to a stream offered by a server. `data_received_cb` allows the user to provide a callback function to be triggered when a new message arrives.
- `unsubscribe(stream, server_address)`
  cancels subscription to a stream on a given server.
- `send_msg(stream)`
  sends a message to all clients subscribed to the given `stream`.

The `options` structure contains fields that allow the user to define certain default configurations for the server, and each stream individually. The configuration parameters are: stream semantic, connection *keepalive* timeout, *keepalive* count, buffer size and auto reconnect (discussed later in Section 4.4).

In order to subscribe to a publisher's stream, a subscriber needs to know the IP address of the publisher of the stream and the stream's name. The two subscription servers negotiate the subscription process via a two-way handshake (request subscription, acknowledge subscription). The publisher's subscription server registers the client as a subscriber of the stream, opening a TCP connection with it over which the stream data will be transmitted.

When data is requested to be sent at the application level (using the `send_msg` function), it is queued by the subscription server in the relevant stream's outgoing buffer as a *message*. Stream data gets received by clients as *messages*. The event-based TCP EmStar library takes responsibility for sending individual parts of a *message* to clients, and the TCP protocol ensures in-order, reliable delivery. Reception of a complete *message* is acknowledged by each client. Only when a message has been acknowledged by all clients subscribed to a stream can it be removed from the stream's queue.

Both the publisher and the client play a part in stream connection management. On the subscription server, each stream client has an individual TCP connection—different streams going to the same client are *not* multiplexed. In order to keep TCP connections active when no traffic is being sent over them (thus avoiding the three way handshake required to initiate a TCP connection), subscription servers send per-stream *keepalive* messages at pre-determined intervals (as noted above). If a client does not respond to a user-defined number of *keepalives*, the server assumes that the TCP connection has stalled due to repeated packet losses (stemming from signal quality or temporary routing problems); it will therefore kill and restart the TCP connection. If a client becomes disconnected from a published stream, it will attempt to reconnect indefinitely, with an increasing timeout. Upon reconnection to a given stream, the server will *replay* any messages the client had not acknowledged whilst disconnected. If client unsubscribes from a stream, the publishing subscription server will initiate a two-way unsubscribe handshake (unsubscribe request, unsubscribe acknowledge). Upon completion of the handshake, the publishing subscription server removes the client from its list of subscribers.

If a stream has no clients, any messages sent to it will be buffered, waiting for an initial subscription. If a client subscribes, it will send all of the buffered data to it. This ensures that potentially important data is not lost during initialisation of an application in VoxNet. However, each network stream has a limited buffer (the default is 512 kB). Consequently, several *lossy* semantics have been implemented to deal with buffer overflows. These can be defined by users as part of the options for initialising the subscription server or individual streams. The first causes messages to be dropped from the head of the queue upon buffer overflow, called *drop oldest*. The second, called *always request latest*, does not buffer data, and all new clients begin receiving only new data. This means data may be dropped by a client if its connection is forced to restart. The spill to disk component uses the *always request latest* stream semantic, because in the event of a disruption, the raw data will quickly overrun any in-memory buffer. In addition, the directly wired connection to the Gumstix is reliable, so loss is not a problem.

In VoxNet, nodes advertise published streams to the sink via the discovery server, which is described later in Section 4.4. There are some well-known streams that are exposed by nodes and the sink during normal operation of VoxNet:

- **Control stream (Sink)**. Allows the sink to send commands to all nodes in the network. All nodes subscribe to this stream.
- **Log stream (Node)**. Allows nodes to send information to the sink. The sink subscribes to the control streams of all nodes known to the discovery server.
- **File stream (Sink)**. Allows the sink to send data files (such as new binaries) to the nodes.

For Wavescope programs, the subscription server is already started and the FFI is used to provide the `toNet` and `fromNet` functions for the system.

While the current implementation uses end-to-end TCP sessions, in other application contexts and at larger scales this may no longer be adequate. Further work is required to experiment with other communication mechanisms including *split-TCP* and Delay Tolerant Networking (DTN) approaches. It is expected that other semantic models will arise as more VoxNet applications are developed.

Discovery service

The control console is a centralised point of contact for the network that is responsible for node discovery, application dissemination, resource usage tracking, error logging, and profiling statistics. It serves as a mediator between users who want to install a program and the VoxNet distributed system, and hosts all appropriate compiler tools and scripts. The discovery service hosted on the control console maintains a list of active VoxNet nodes and back-end server machines, and tracks their capabilities and resource availability. When VoxNet nodes or servers start up, they connect to the control console at a well-known address and register with the network.

Application dissemination

When a user submits an application for propagation to the VoxNet system, the control console compiles it for the appropriate architectures and pushes the compiled components out to nodes currently in its discovery list. The application dissemination mechanism is based around networked streams: each node subscribes to a *file* stream exposed by the control console. When a new application needs to be disseminated, it is sent to each node over the *file* stream. This approach gives the same performance as manually copying the file to each node in the network using the *secure copy* application (`scp`). The relative performance is shown in Section 4.5. To actually perform the dissemination, the user can use the Wavescope shell, defining the file to send, and the location to send it to on all the nodes.

After applications have been disseminated, they must be initialised in the system. This feature is not fully implemented in VoxNet's current incarnation and is performed by making use of symbolic linking and the *Emrun* process manager. The currently running application binary is referenced in the user-defined *Emrun run* file via a symbolic link, rather than directly. When the new binary has been successfully received, the symbolic link is changed to point to it. The Wavescope process is then killed, which causes *Emrun* to reload and restart the executable. This time, the newly disseminated executable will run, rather than the old one.

Dissemination as currently implemented does not consider several important factors: transactional binary updating (what to do if not all nodes in the network receive the new executable) and application rollback (how to swap out a faulty application and replace it with a previous (working) one). A potential solution to rollback could be that the location of the previous binary is stored, and if *Emrun* notices the new binary has crashed, then the symbolic link is reverted before the process is restarted.

WaveScope shell (WSH)

To control the VoxNet deployment, the *Wavescope Shell (WSH)* was developed: a text-based command-line interface to the control console, similar to the UNIX shell. WSH was implemented using GNU

`readline`, and is based on the *Remote Broadcast Shell (RBSH)* implemented by Girod (2005). WSH uses reliable stream connections to communicate with nodes, as opposed to RBSH's unreliable approach, where commands are flooded to nodes as many as three times to ensure delivery. Additionally, WSH distinguishes between local and network commands, whereas RBSH forwards any UNIX command given at the prompt to be remotely invoked at all nodes in the network (WSH can also perform this functionality via the `send` command).

WSH communicates with nodes over the **control stream**, which all nodes in the network subscribe to by default. As the control console also subscribes to all nodes' control streams by default, log messages sent over this stream are passed to WSH for display to the user. (Log messages sent by nodes to the control console carry a special header to identify them to be interpreted as character data. WSH knows how to parse and interpret these messages).

The WSH network command set is as follows:

- `send`
  send and execute an arbitrary UNIX command locally on the node.
- `pause`
  pauses the Wavescope application that is currently running on the nodes.
- `start`
  (re)starts the Wavescope application that is currently running on the nodes.
- `flush`
  clears the send buffers of all streams exposed by all nodes in the network.
- `list`
  provides a tabular display showing the current status of each node registered via the discovery protocol. Each stream a node currently exposes is displayed, along with the current send-buffer status for that stream (the amount of data that is queued to send).
- `test`
  request nodes to send back an arbitrarily-sized message on the control stream, including a timestamp indicating when the message was queued to be sent. Accepts arguments for the amount of data nodes should send back and the ids of nodes that should reply.
- `fsend`
  sends a file to all nodes in the network. Accepts an argument for the file to send, the nodes to send it to and the location to save the file to remotely. This mechanism allows for binary dissemination, as noted in Section 4.4.4.

Network commands (that is, commands that are node-bound) are sent to all nodes in the network by default. If the user provides a comma-delimited list of nodes to send to as an optional argument to any network-based control command, then only those nodes which are listed will respond to the message. Note that the message is always delivered to all nodes. This feature can be used to selectively address

nodes, for instance in requesting them to send data back to the control console when testing the network connection. Control client logic on each node parses WSH command messages when they arrive (on the control stream) and the node acts accordingly: executing an arbitrary command, receiving a file or replying with a timestamped data packet.

The `test` command is a useful tool for network testing at sub-application level, and was designed and implemented by the author to provide a framework to perform network transfer time analysis. The functionality of `test` is significantly extended in Chapter 6 to support more advanced network testing. WSH receives and interprets the data test messages that are sent by nodes in response to `test` commands. WSH takes a timestamp when the message arrives at the application layer as well as reading the global sending timestamp the node placed in the message payload. These timestamps, as well as the node id and message size are output to the screen by WSH so the user can used them to calculate data goodput.

In the current implementation, WSH is integrated within the control console software, although future versions would separate these two functions so they could exist on different computers if required. This would allow the shell to be located on a portable computer, for example the user may want to carry a computer nearer to the deployment, and still issue commands to the node, but still have the main control control in a static position.

Spill to disk

Experience with field scientists showed that even in instances where detection and localisation can be performed on-line, there is also value in recording a raw data set for future processing. While this desire may diminish as confidence is gained in data reduction algorithms, it will always be important in the testing phases of any new algorithm, as well as for interactive use (to replay and re-analyse recent data). To address these concerns, a *spill to disk* functionality was developed. The spill to disk component saves correctly time-stamped raw data streams to the flash card in the VoxNet node. In the main Wavescope program, an extra *toNet* operator is included so that the main processor publishes the raw data being gathered as a network stream visible to the supervisory co-processor (the Gumstix). A subscription client on the Gumstix receives the data over the wired network from the main ENSBox processing board, and marshals it to files on its flash card, properly annotating with global timestamps. Through experimentation, it was found that VoxNet can continuously archive 4 channels of audio at 44.1 kHz while concurrently running other applications, as shown in Section 4.5.2 on page 116.

In a VoxNet deployment, network limitations mean that raw data can only be saved in a node's local storage (and as such can still be accessed by a distributed Wavescript program). After the deployment, stored raw data can be dumped to a large server for archival purposes; the same programs that run in a deployment can be run against the archived raw data.

4.4.5 Platform drivers and services layer

The drivers and services layer provides important support for the application and interaction layers of VoxNet. Some of the components used in VoxNet are similar or the same as those used in the first iteration of the system, and are noted thus where relevant.

IP routing

VoxNet implements IP routing from each node back to the gateway node using an existing user-space implementation of Dynamic Source Routing (DSR) (Johnson et al. 2001), written by Stathopoulos (2006). The implementation dynamically installed routing entries into the kernel routing table, allowing nodes to automatically forward traffic on behalf of one another. Although DSR can establish routes between any two nodes on-demand, it was only used to find and maintain routes between the sink and nodes. The DSR implementation was not fully integrated into the system, meaning that to maintain routes, a special ping message has to be sent by each node to the sink periodically. Future implementations of the system may allow the traffic that is travelling over the path to act as a keepalive for the DSR algorithm. The implementation of DSR required little modification to run in the VoxNet system.

The gateway forwards between the *Pseudo-Independent Basic Service Set* network and a normal wired or wireless Local Area Network (LAN). This means that client devices are not required to use *Pseudo-Independent Basic Service Set* mode (a special feature of a particular hardware driver, discussed in Section 4.3.1).

Link estimation and neighbour discovery

Link estimation and neighbour discovery are important to support the multi-hop routing protocol used—in this case, Dynamic Source Routing. Link estimation was provided by an existing driver called *rnplite*, which periodically sends broadcast messages to determine link quality between neighbours. A moving average for each of a node's neighbours was taken, based on the amount of broadcast messages that were sent and received. Since incoming and outgoing links can experience different levels of loss (Cerpa et al. 2005), it was important that a node knew how many of its messages were received by others (outgoing) and how many messages it received from other nodes (incoming). The sent and received values are expressed as a percentage of the total sent within the moving time window.

Gateway time synchronisation

The time synchronisation service used by the nodes in the network based on RBS only works over wireless communication, hence extra logic is required to bridge the gap between the control console and the network (as they are linked through a gateway). In order to support this, an application was developed which runs on the sink and the gateway. The existing EmStar time synchronisation service (*gsync*) requires that pairs of global and local time measurements are recorded regularly. The *gsync* service uses these measurements in a linear regression to allow arbitrary local times to be converted to global time and vice versa. The aim of these pairs is to have them be as close as possible in time (ideally both are taken at exactly the same time). However, because the sink and gateway are separated by a wire, some latency is incurred between the sink requesting the current time and physically receiving it from the gateway: the time taken for the gateway to respond the query, and send a reply over the wire. The most basic way to approach this would be for the sink to request the global time from the gateway, then pair it with a local timestamp which is taken when the request reply arrives. However, this does not factor in the time taken for the request to travel back and forth along the wire. Therefore, a more complex procedure is required to increase the accuracy of timestamps.

The implemented solution is as follows: the gateway is configured as the root time source for all nodes in the network. Periodically, the sink queries the gateway for its current time, using a User Datagram Protocol (UDP) packet (a TCP packet may incur latency due to the three-way handshake required to establish a TCP connection). The sink locally records the time at which it sends a query packet ($t_{q,s}$). The gateway runs a **time server**, which services only time queries from the wired interface. When it receives a query, it timestamps its arrival time ($t_{q,a}$) and prepares a reply packet (which the arrival time is inserted into). Before the reply packet is sent, another timestamp ($t_{r,s}$) is made and inserted into the packet. The sink timestamps the arrival of the reply packet as $t_{r,a}$. The sink wants to pair $t_{r,s}$ and $t_{q,s}$, so it computes the wire transmission time W as follows:

$$W = (t_{q,a} - t_{r,s}) - (t_{q,s} - t_{r,a})/2 \tag{4.1}$$

and then the timestamp pair t_{pair} as:

$$t_{pair} = \langle t_q, t_r + W \rangle \tag{4.2}$$

meaning that t_r has the estimated wire transmission time w added to it. This means that t_q is as close as possible to the reception time.

4.5 Evaluation

In Section 4.5, the operation of the system as a whole, and its individual components are discussed and evaluated. The evaluation performed here has been performed both through micro-benchmarking and real-life deployment. Real-life deployment is evaluated in Chapter 5. Resource usage and processing times are compared for the AML and event detection components on nodes. The spill to disk and dissemination components are also benchmarked.

4.5.1 Memory consumption and CPU usage

It is important that the VoxNet implementation of the marmot localisation application be comparable in terms of resource usage to the original EmStar-only application.

In order to test this, the resource footprint (CPU and memory usage) of the on-node event detector was recorded for both applications. Figure 4.7 shows the CPU and memory usage involved in the on-node event detection for both the Wavescope and EmStar applications. For both, this comprises two components: the detector itself and the data acquisition. The figures are the mean of one minute's worth of CPU and memory usage obtained using the Linux command *top* (20 data points).

Figure 4.7 shows that the total overhead of the Wavescope version is over 30% less in terms of CPU (87.9% vs 56.5%) and over 12% less memory (20.9% vs 8.7%). The main reason for this is that the sampling layer is compiled into the main VoxNet executable, removing the overhead of inter-process communication that EmStar uses.

4.5.2 Spill to disk

Because the Wavescope version uses fewer CPU and memory resources, additional components can run concurrently with the detection application. To demonstrate this, the *spill to disk* component (which archives a copy of all raw audio data to flash storage) was tested along-side the event detection application.

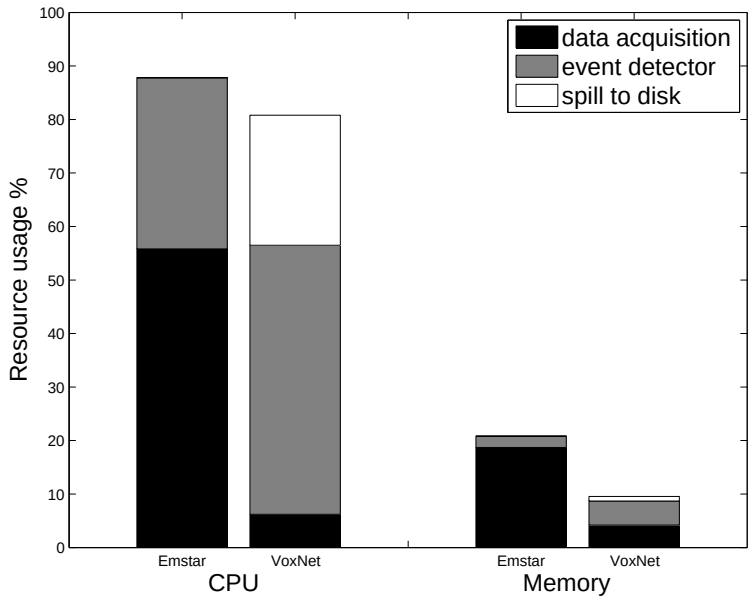


Figure 4.7: Comparison of memory and CPU usage for the on-node event detector in both EmStar and VoxNet. The left hand side set of bars show CPU usage (broken down by event detector and data acquisition components) and the right hand set show memory usage. For VoxNet, the spill-to-disk component running concurrently with the event detector and data acquisition is also included.

The event detector was run concurrently with the spill to disk component for 15 minutes. The CPU and memory usage was monitored on the Slauson (the ENSBox's main processing board), as well as the data transfer rates between the Slauson and Gumstix, and the buffered queue sizes for the incoming data stream at one second intervals. Figure 4.8 shows both the stream buffer size and data transfer rate over time. On average, the throughput from Slauson to Gumstix must keep up with the data rate—that is, four channels of 16-bit audio sampled at 44.1 kHz (344.5 kB/s). The throughput from Slauson to Gumstix over the stream was 346.0 kB/s, which is in line with the required raw data rate (including a timestamping overhead of 8 bytes per raw data segment).

The stream buffer between Slauson and Gumstix was never completely cleared—on average it had 79.5 kB (0.23 seconds) of data in it at any one instant. The sensor buffer, which stores a back log of acquired sensor data (up to 10.42 seconds) was slightly behind (2.07 seconds at worst) as the application initialised on the platform, but the buffer was emptied after around 10–12 seconds (indicating stabilisation). Including the start up period, the buffer between the data acquisition thread and the main Wavescript program had on average 0.02 seconds of data in its buffer throughout the experiment. This is a positive result, as it means that the data acquisition and the main program threads are being serviced fairly (i.e. getting the same amount of processor time). If the buffer started to fill up, this would indicate an imbalance between the two threads. Once the buffer starts to fill up, it becomes difficult for the main program thread to catch back up with the real-time audio. Under heavy load, this would be disastrous, and most likely result in the loss of audio data (getting dropped from the buffer). The third and sixth

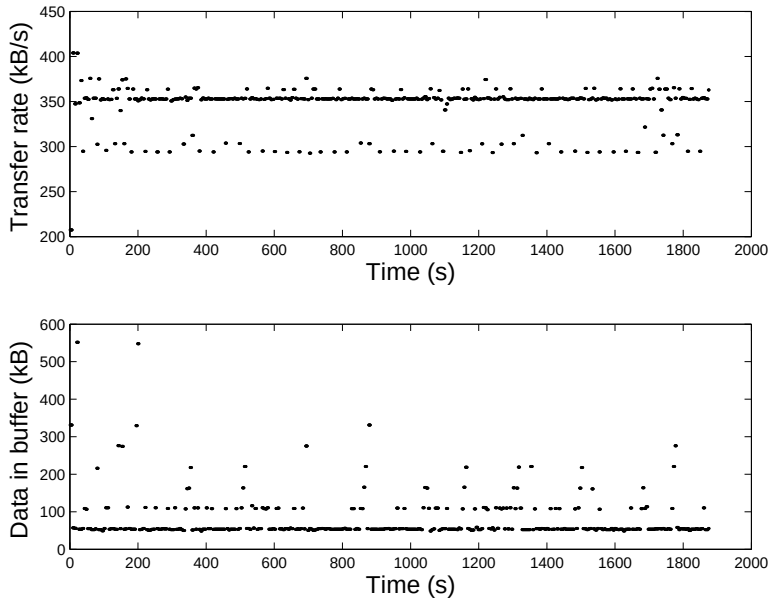


Figure 4.8: Data transmission rates between the ENSBox's main processing board (the Slauson) and the Gumstix co-processor, as well as buffered data queue size, over a 15 minute period.

bars in Figure 4.7 show that the overall resource overhead of running both spill to disk and an event detector on the node is 80.7% CPU and 9.5% memory (taken from 20 consecutive *top* measurements).

4.5.3 Binary dissemination

The network stream file dissemination approach was compared to the `scp` application, in terms of latency. `scp` is a commonly used method of securely transferring files over a network using TCP/IP so it represents a good comparison point. A 1.5 MB file representing a new binary was transferred to an increasing number of nodes (up to four). Both `scp` and the network stream file dissemination mechanism were used. For `scp`, the file was transferred to each node in sequence, and for the network stream mechanism, the file was transferred to as many clients as were subscribed to the stream. For each number of nodes, five transfers were made. It was expected that the transfer times would be similar, and that the network stream mode should not be slower than `scp`. Figure 4.9 shows the results. The performance is as expected: `scp` and the dissemination mechanism are within a second of one another for a 1.5 MB file (with dissemination mechanism quicker). However, the approach is not really scalable, as the time to transfer data will be linear in the number of nodes (which is the same as manually copying the file to all nodes, like `scp`). A different approach may be for nodes to propagate the file to its multi-hop children as follows: the sink sends the binary to its children on the routing tree, and they send to their children, until the new executable has been propagated throughout the network. This may allow nodes to take advantage of spatial re-use, where nodes that are not in the same communication range can transmit at the same time (this is discussed more in Chapter 6).

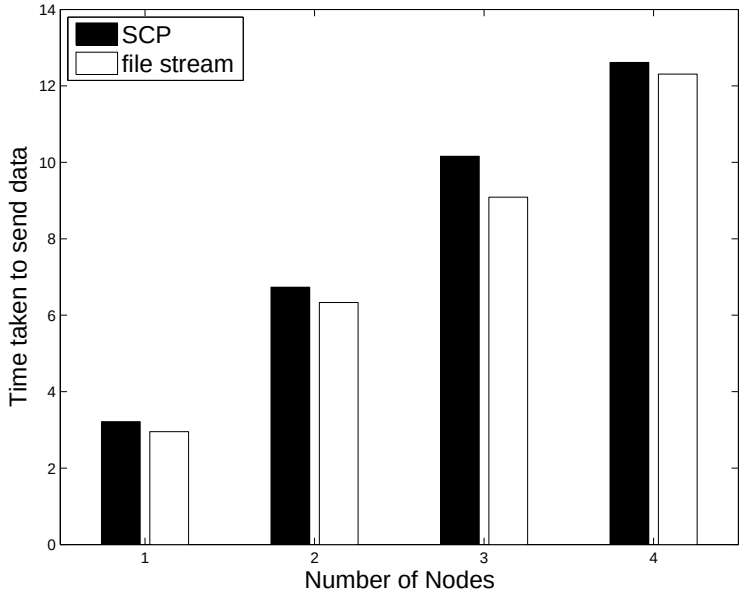


Figure 4.9: The amount of time to transfer a new binary to nodes in the network using `scp` or the network stream mechanism.

4.5.4 On node processing comparison

Providing the capability for on-node data processing is an important part of VoxNet. To further test the on-node processing capability, the time taken to compute a DoA estimate using the AML algorithm was measured. In the marmot localisation application, this is an intensive processing operation, but can potentially be performed at the node or sink. It is currently processed at the sink. The performance of the C implementation of the AML used in the EmStar-only system was compared to the corresponding Wavescope implementation. The same Wavescope program was used to test both implementations: the Wavescope version was written in natively in Wavescript, and the C version was compiled into a C library module, and accessed through a FFI call. Whenever an event detection was triggered, it would be passed through the AML operator (either the C or Wavescope version). The time taken to perform each AML computation was measured. For both implementations, 50 event detections were triggered. Table 4.1 shows the minimum, mean, median and maximum processing times for both the native Wavescope and FFI AML computations. For comparison, figures are shown for the same AML computation running on an x86 laptop-based version of the VoxNet node (Allen et al. 2007), with 256 MB RAM and a P4 2 GHz processor.

There is comparable performance between C and Wavescript generated C in both ENSBox and x86. The mean and median values are similar (within 0.1 second) between C and Wavescript, although Wavescript shows lower min and max values on the node. there is roughly a 0.4 second range in AML processing times, which is 16% of the total time taken for the AML on average. On x86, the processing times are clearly faster. As with the node latencies, the times are close together (this time within 0.02 seconds

Table 4.1: Processing times (in seconds) for C and WaveScope implementations of the AML algorithm applied to streams of event detections. Min, mean, median and max processing times are shown for x86 and ENSBox architectures.

	Min (s)	Mean (s)	Median (s)	Max (s)
C (node)	2.4430	2.5134	2.4931	2.7283
WaveScope (node)	2.1606	2.4112	2.4095	2.5946
C (x86)	0.0644	0.0906	0.0716	0.2648
WaveScope (x86)	0.0798	0.1151	0.0833	0.5349

of one another). However, the maximum latency for Wavescript is 0.53s, which is twice the amount of the maximum C latency. It is likely this value is an outlier. This means that the Wavescript generated code performs as well as a hand-coded C implementation. This is a positive result, as there is a danger that code generated automatically from a high-level language like Wavescript will produce inefficient code. Therefore, there has been no sacrifice in performance in using the high-level Wavescript language to express VoxNet's programs.

4.6 Summary

The chapter described two design/implementation cycles of an end-to-end, on-line marmot localisation system. The initial system design and implementation was taken from a top-down perspective, building on prior work to enable an end-to-end localisation system to be built using the EmStar WSN software development framework. This implementation took several existing components and provided application glue to integrate them into a proof of concept demonstrator, albeit without addressing reconfigurability or interaction concerns. The second iteration took a bottom up approach to marmot localisation, by providing a platform for distributed acoustic sensing over which the marmot localisation application could be expressed. This iteration of the end-to-end system was developed as a collaborative effort known as VoxNet. VoxNet's design stems from consideration of high data-rate applications in the context of bio-acoustic applications. It was shown that in using Wavescript instead of exclusively EmStar to define and build the marmot localisation application, less CPU and memory resources were consumed. The availability of extra resources enabled extra functionality to be implemented alongside the required program flow. It is important that the high-level programming and interaction model provided for users of the network are achieved without loss of raw performance, even on a platform as resource-rich as the V2 ENSBox/VoxNet node.

The interaction layer tools developed for stream management, stream discovery, dissemination and network subscription are only loosely coupled with Wavescope/Wavescript via the foreign function interface. These tools can therefore be re-used in non Wavescope contexts, including other applications where the model is suitable.

VoxNet has a general appeal to the general application class of source localisation (animals and birds, not just marmots) and the broader application class of distributed acoustic sensing applications. At the broadest point, high-data rate sensing applications can make use of the software platform VoxNet

provides for non-acoustic sensing applications, such as seismic sensing. Chapter 5 further discusses VoxNet's evaluation during in-situ operation, draws on the deployment lessons and highlights the issues discovered during deployment.

Chapter 5

Deployment lessons and refinements

Chapter 4 described the design and implementation of two iterations of an on-line system for the source localisation of marmots. This culminated in the implementation of VoxNet, a platform for distributed acoustic sensing on which the marmot localisation application could be expressed. The design of the system was application-led, and the marmot localisation solution was based on the integration of existing components (event detection, angle of arrival estimation and localisation) into a system which could be used to perform localisation on-line, in-situ, rather than off-line. Whilst some evaluation and micro-benchmarks were presented in Chapter 4, it was vital for the system to be deployed and tested with real-life stimuli in an uncontrolled environment. This was to identify usage problems and failure modes that could not be determined through controlled experimentation, as well as indicate areas for improvement.

This chapter reports on the in-situ, ten-day deployment of the marmot localisation application at the Rocky Mountain Biological Laboratory (RMBL) in Colorado. A detailed account of the actual deployment experience is provided, alongside the problems that were encountered during the deployment, and the lessons learnt during this time. The rest of the work presented in this thesis (Chapters 6 and 7) is informed and motivated by the deployment-related findings reported in this chapter.

The on-line operation of the marmot localisation application running on VoxNet is what differentiates it from previous work in marmot localisation, as discussed in Chapters 2 and 4. The two most important factors for the *on-line* performance of the marmot localisation application and VoxNet's operation in general are the *latency of data transfers* and the *intelligent grouping of data* at the sink for processing. These factors affect the robustness and timeliness of the localisation application, thus it is important that further platform development work will cater for ensuring timely delivery of data. Subsequent work in this thesis (Chapters 6 and 7) concentrates on designing and implementing approaches to improve the timeliness and reliability of the system.

The rest of this chapter is organised as follows: Section 5.1 describes the deployment in detail; the deployment process, pre-deployment notes and general lessons earned are described. Subsequently, the specific problems relating to the marmot localisation system and the VoxNet platform are elaborated on. In Section 5.2, three identified application-specific marmot localisation problems are discussed: the on-line grouping of detections to determine network-wide events, false detections arising from weather conditions and the efficiency of the localisation algorithm used in an on-line context.

In Section 5.3, four identified problems associated with the general operation of the VoxNet system are detailed: data logging, data loss, data consolidation and stream priority. These problems impacted the specific application and subsequent analysis of data gathered and would impact to an extent any acoustic sensing application running on the VoxNet platform.

In light of the experience of deploying and using VoxNet in the field, Section 5.4 discusses the steps that would be necessary to make the system ready to be adopted by scientists for field research. Finally, Section 5.5 concludes the chapter and explains the path for the rest of the work in Chapters 6 and 7.

5.1 In-situ deployment

From the 8th to 18th of August 2007, the VoxNet platform was tested in a deployment at the Rocky Mountain Biological Laboratory (RMBL) in Gothic, Colorado, USA. Biologists have been studying marmot behaviour around this area for a number of years. (This location was also used in the experimental work conducted by Ali et al. (2007)).

Four people in total were present at RMBL to deploy VoxNet in-situ: the *VoxNet development team* and a biologist studying marmot behaviour using the Acoustic ENSBox as a data acquisition tool. The VoxNet team consisted of the three computer scientists collaborating on the VoxNet project: Ryan Newton (MIT) was the developer of the Wavescript language and compiler for Wavescope, Lewis Girod (MIT) was the main developer of the Acoustic ENSBox (V1 and V2), EmStar and VoxNet, and the other was the author, co-developer of the VoxNet system and contributor to the Wavescript and EmStar projects. The biologist, Travis Collier (UCLA Evolutionary Biology), was an expert user of the ENSBox nodes, having deployed them for time-synchronised recording of raw audio several times previously, and was a central participant in the previous off-line experimentation for marmot localisation (Ali et al. 2007). During the ten-day period of VoxNet deployment (and several months beforehand), the biologist also used the nodes to gather raw acoustic data without VoxNet. Throughout VoxNet's deployment, the biologist was there to provide domain expert advice and guidance where required, and to generally help assist with the physical deployment.

Over the ten day period, four attended deployments of the VoxNet network were undertaken. The usage model during this time was a cycle of: physical deployment of nodes, attended operation of the localisation application, post deployment collection of devices and consolidation of data for analysis. These three phases roughly coincide with part of VoxNet's intended on-line usage model (in-situ, on-line interaction) and post-deployment/transitional stage (consolidate data to archive server). However, VoxNet also accommodates post-processing of gathered data via the archive and compute servers, which was not implemented in this deployment (although it is discussed in Section 5.3.3). Each time, a network of eight nodes was deployed from mid-morning until early afternoon (when marmots are typically active). The eight nodes were deployed over an area of 140m by 70m (2.4 acres), as shown in Figure 5.1. Each node was between 40–50 metres of its nearest neighbours. The node positions chosen by the biologist were based on expert knowledge: the deployment area encompassed three well-known marmot burrow locations. The node positions were kept consistent over different deployment days by leaving stands in position and only removing the nodes at the end of each deployment. Nodes were taken back to the accommodation for software debugging and data consolidation.

Following advice from the biologist in attendance, the gateway node and control console were placed in turn in two selected positions during the experimentation. The first was 200m west of the deployment, in a car-park. The second was 100m southwest of node 100 (see Figure 5.1). When at the second position, all nodes were within one hop of the gateway node, meaning that multi-hop communication did

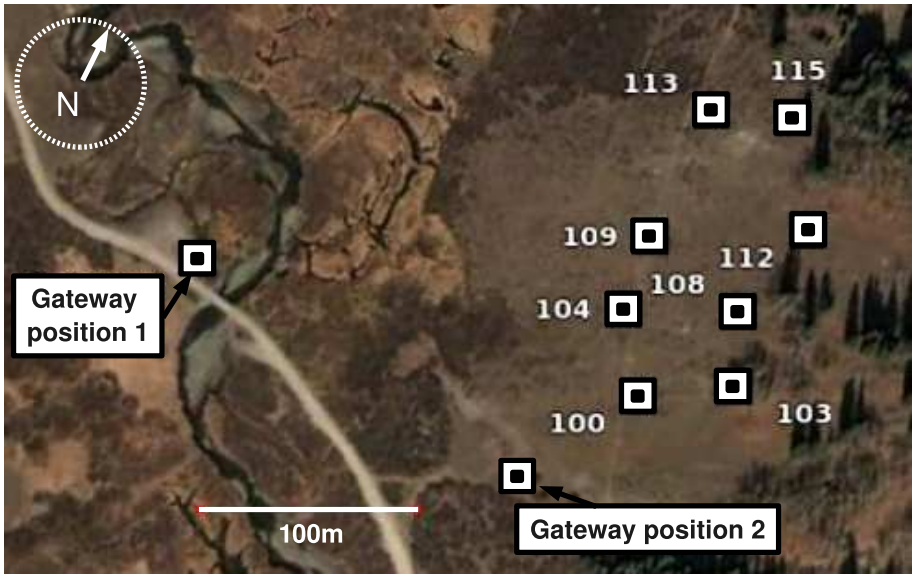


Figure 5.1: The physical deployment of the eight nodes at RMBL. The nodes were deployed over a 140m by 70m area. The gateway and control console were deployed at two locations during the deployment, approximately 200m west of node 104 (in the car park), and approximately 100m southwest of node 100 (in an open, grassed area).

not necessarily have to be used during the deployment. Both of these positions were far enough from the deployment area so as not to constantly interfere with the marmots. It was intended to use VoxNet in both a multi-hop and single-hop network configuration whilst running the localisation application, in order to isolate any problems that related directly to the multi-hop component of the system.

During deployment, the marmot localisation application as described in Chapter 4 was run in the VoxNet network. A laptop was used as the control console, which was connected via Ethernet to the gateway. The Wavescope Shell (WSH) to control the nodes was run at the control console, as well as the sink-side processing of the marmot localisation application (AML and position estimation). The control console was time synchronised with the network using the *time server* service on the gateway, as described in Chapter 4. Throughout the deployment, the WSH logged all detections and messages that were generated by the nodes in the network and transmitted to the control console on both detection and control streams. All locally generated debug and network information data was also logged by the WSH. In addition to sending control information, nodes also logged link quality and network topology information. This data was gathered from the nodes after the deployment, back at the accommodation. Controlled network data transfer experimentation was also carried out during the deployment period. This experimentation is discussed and drawn on in detail in Chapters 6 and 7.

5.1.1 System usage cycle

In total, four deployments were carried out over the ten day period. The deployments were fully attended throughout, to allow maintenance to be carried out as well as note-taking on problems or interesting

Figure 5.2: A deployed sensor node.

events. Each deployment had the same high-level usage cycle: the nodes were physically deployed and switched on, self-localisation was performed and the marmot localisation application was run. Upon ending each experiment, the nodes were shut down and collected, leaving the stands behind in order to maintain consistent node positions across all deployments. It was necessary to bring the nodes back in order to extract any log data that had been gathered, and to make any modifications to the software in order to prepare for the next day's deployment.

The VoxNet node hardware was the Acoustic ENSBox V2, which had already undergone a complete packaging re-design refinement to make it smaller and lighter, thus easier to carry and deploy. In particular, each V2 node was enclosed in single box with a handle attached, meaning that up to four nodes could be carried by one person at a time. Since there were always four people attending the deployment, each person deployed two V2 nodes, mounting nodes at each of the stands, as per Figure 5.1.

As the nodes were deployed on each stand, they were switched on. When each node completed its boot-up process, it played a short two-tone sound to indicate it had loaded up the application successfully. This feature was implemented at the request of the biologist based on previous deployment experience. The approach saved time, and confirmed that nodes were working correctly. After the nodes were initialised, self-localisation was performed before starting the Wavescript application. The results were stored in a file that was read by the sink-side application on the control console.

5.1.2 Deployments

Each of the deployments was intended to test VoxNet and the marmot localisation application in a realistic environment to allow a comprehensive picture to be built as to the necessary improvements for further

design iterations. It was important to understand the problems arising from running the localisation application on-line, with real marmots. On-line operation is the primary difference between the VoxNet marmot localisation system and the off-line version presented by Ali et al. (2007).

In the days preceding the first deployment, several trips were made to the field in order to test multi-hop networking component in a realistic scenario, and over a larger distance than had previously been attempted. A problem was observed where the control console (which was connected to the gateway) was not able to receive data from the nodes when the application was running. The gateway was forwarding the control console's data to the network, but the nodes in the network could not forward traffic to the sink. This was because when a new multi-hop route was found for a node (by the routing component), the IP address of the node's new parent was not being used as a gateway in the kernel's IP routing table (this is necessary to make sure traffic is forwarded out of the network using IP forwarding).

Throughout deployments one, two and three, there were problems with the data buffer (audio buffer) between the data acquisition component and the marmot localisation application (which used the data acquisition as a source). This problem was eventually tracked down to a thread-starvation problem between the data acquisition thread and the node-side thread in the application executable (which were complied together to reduce IPC overhead). The two threads needed to be serviced fairly, because if one started to get more processing time than the other, the audio buffer between them would get full and start dropping data. If the buffer started dropping data, then the audio stream being gathered would no longer be continuous. This would be problematic for the event detector, as non-continuous data would affect the noise estimation causing false detections. Additionally, the spill to disk component would have large gaps in the raw audio stream, although it was not utilised in this deployment. This was solved with two fixes: increasing the buffer size, and applying a more efficient data copying technique between the threads (this Whilst increasing the buffer size would not guarantee to fix starvation, the gap issue was not observed after improving the data copying technique between threads.

Because several problems occurred that were not predicted prior to deployment, further analysis of log data was required (which required

It was not always suitable to solve problems in-situ: laptops were running on batteries, and log data needed to be gathered, consolidated and analysed. Additionally, development and controlled experimentation was required, both of which were not suitable to be performed at the deployment site. Although the deployment area was near (within five miles) the accommodation, the time window in which to monitor marmots every day was limited. This meant that a non-trivial software bug would effectively hamper the day's work. In addition, the biologist present throughout the deployment had to record continuous marmot data on some of the days set aside for deployment. This meant that VoxNet could not be deployed on these days (the scientist needed to run his own, EmStar-based software).

Weather conditions were hazardous for the gateway and control console: they were not waterproofed, so had to be protected from the rain. Furthermore, sunlight made the laptop screen difficult to read (a solution to this problem is shown in Figure 5.3).

The screen glare problem was compounded by the fact that battery life at the control console was limited. Often, the screen brightness had to be reduced in order to conserve battery life. Several times, it was necessary to change laptops in order to keep experimentation running. This proved difficult for

Figure 5.3: Dealing with the problems of screen glare during in-situ experimentation.

consolidation of data post-deployment, as will be discussed in Section 5.3.3. Ideally the battery life of the control console should match the lifetime of the nodes, or at least the intended duration of the deployment.

During the deployments, it was necessary to induce marmot calls to test the marmot localisation application. This was achieved by a member of the deployment team walking slowly and quietly into the deployment area, and then making a sudden movement at the marmots. Usually, a single marmot would signal alarm and all would run back to their burrows temporarily. When many marmots were together, this behaviour could result in multiple marmot calls.

Each of the deployments identified a specific problem that had been overlooked or underestimated during the laboratory-based development of VoxNet. Therefore, with each deployment, the general operation of VoxNet was improved. The rest of Section 5.1 describes each deployment in more detail.

5.1.3 Deployment one

The first deployment was made on August 14th, 2007 and involved running the full marmot localisation application. There was light rain falling over the deployment area and as the gateway and control console were not weather-proofed, they were deployed at the car park (position one), inside the car.

VoxNet's multi-hop routing component was initially enabled for the deployment, but encountered a problem that had not been seen previously in lab or initial in-situ testing. The implementation of the DSR algorithm would find long paths from node to sink (up to eight hops) which would change often and appeared not to settle. It was not understood why this occurred, and time did not allow for a detailed analysis of the routing component (perhaps a bug in the implementation). To resolve the problem it was decided to limit the maximum length of a routing path between node and sink to be *three hops*. This change was implemented after the deployment, and worked for the remaining deployments.

The marmot localisation application was run in single hop mode whilst still at the car park (the

weather had cleared by this point). When detections were triggered at each node, a data transport problem became apparent. It was found that nodes could quickly pass short log messages back to the sink (around 30 bytes) to indicate that detections had been made, but detection data was not getting back quickly enough for detections together (recall a timer was set for 0.5s after each detection's arrival at the sink, from Chapter 4). This in turn meant there was no suitable input for the localisation algorithm, which was not invoked at all. Instead, nodes were repeatedly getting disconnected from the control console as they tried to send data. The initial assumption was that this was due to the low quality of network links, so a decision was made to move the gateway and control console to position two (as shown in Figure 5.1) and see if this would help with disconnection problems. Moving the gateway closer to the nodes did not seem to change the rate of disconnections, so it was decided to bring the nodes in to consolidate and analyse the data set that had been gathered. The total time from position one (car park) deployment to consolidation at position two was 25 minutes.

Post-deployment, problems were found with the log data. The raw detection data messages sent from nodes had detection timestamps in the node's local clock rather than the network time. Additionally, the sink-side application recorded a log message when detection data from nodes arrived at the sink, but neglected to record the id of the node in this message, or the time it arrived at the sink. This meant that the latency between detection at the node and arrival time at the sink could not be determined, which would have shown how long detections were taking to arrive and infer if moving the gateway did indeed help. The only data that could be gathered from the sink-side logs was that the detection data being sent from nodes were varying in size, either 32 kB, 65 kB or 128 kB. The detection lengths should have all been 32 kB, so the extra data generated would have been filling up nodes' sending queues. This could have meant that data was not received at the sink because it was getting dropped (although there was no log information to support this). During the 25 minute period, 86 detections were made by all nodes, but only 46 were received at the sink. This may be partly due to the fact that the application was stopped before nodes had cleared their data buffers, when feasibly they might have transmitted the remaining data.

Finally, the link information that was logged at each node showed that link quality improved between all nodes and the gateway after moving the gateway to position two. It was not clear from the gathered log information whether this had an effect on the latency of transmissions, however.

5.1.4 Deployment two

The second deployment took place on August 16th, 2007. Starting at 11am, the VoxNet network was set-up in single-hop mode with the gateway in position two, running the marmot localisation application for several hours. Subsequently, the application was stopped and VoxNet set into multi-hop mode. The intent was to gather marmot detection data in single-hop mode, and then troubleshoot multi-hop communication and data transfer latency (as seen in deployment one), before finally starting the marmot application in multi-hop mode. Controlled experiments were carried out for 30 minutes (see Chapter 6 for more details), to be followed by the evaluation of the localisation application. However, the multi-hop deployment had to be cut short due to weather conditions when a storm interrupted deployment. As the rain started, the gateway was moved from position two to position one (inside the car). The rain

Figure 5.4: Four marmots as observed at RMBL. Note that the marmots tend to gravitate around their burrows.

caused the marmots to retreat to their burrows, and the raindrops hitting the microphones and node cases caused the event detectors to constantly trigger. It was expected that the network would become heavily congested, and that nodes would drop data as their transmit queues overflowed. Over the course of just 436 seconds (7 1/4 minutes), 2890 detections were triggered (a rate of around 6 a second network-wide). At the control server side, 342 detections arrived at the sink. This episode gives an insight into the behaviour of the marmot localisation application running in VoxNet during heavy load. Each node dropped in the region of 90% of detection data queued to send. Despite these data losses, the system continued to run, demonstrating that it could deal with network overloading in a graceful manner.

After the deployment, the nodes were collected and the data consolidated at the accommodation. It was discovered that the changes made to the VoxNet application to improve logging output had neglected to record the node id associated with the raw detection, meaning that the single-hop data gathered was unusable. However, with respect to marmot localisation, rainfall is not really a situation that would warrant continued operation of the system. This is because the marmots would retreat back into their burrows, meaning they could not longer be observed. In this case, the scientist would either temporarily stop the application, or abort the experimentation for the day. However, in unattended deployments where the WSN is expected to be autonomous, it may be useful for the network to identify a change in weather conditions as a reason to temporarily stop processing data.

5.1.5 Deployment three

The third deployment took place on August 17th, 2007. The weather was bright and sunny, so it was decided to move the gateway node to position two. In the deployment, an attempt was made to manually write logs detailing marmot behaviour: recording the rough locations where marmots called from and how many calls were detected. This was done in an attempt to relate events recorded by the system to real marmot events, an aspect that had been neglected in the two previous deployments.

Only seven of the eight nodes were used as the eighth node was experiencing intermittent problems causing it fail to respond to network communication. This was put down to transient effects caused by the rainfall in the previous deployment day. The nodes were deployed for 2 hours 10 minutes. Within that, the application was run in multi-hop mode for around 1 hour 25 minutes, and in single-hop mode

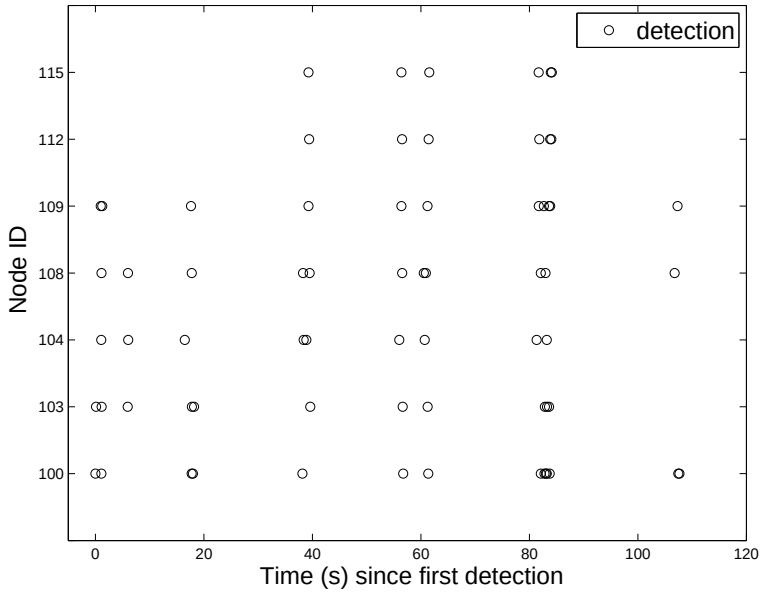


Figure 5.5: A set of events taken from deployment three, dated 17th August, 2007. The x axis shows time since the first detection (in seconds) and the y axis shows node id. Each data point represents the time a detection was triggered at a node.

for around 45 minutes.

During the deployment, three periods of marmot activity were induced by members of the deployment team. The first period of activity lead to a set of eight detections over a two-minute period (approximately) where the sixth, seventh and eighth alarm calls were from different marmots but close together. Figure 5.5 shows a graph of detection time from first detection (on the x axis) vs node id (on the y axis). The individual marmot calls are clear on the graph with detections made by several nodes appearing as tightly grouped vertical lines. Notably, the sixth, seventh and eight calls are situated around the 80 second mark, where there are indeed multiple detections from each node within a small space of time. After this point, the control console was swapped over to a different laptop and the application restarted.

The second period of activity was a bout of twenty-eight calls made by one marmot as a member of the team slowly approached it. Calls were made at intervals of roughly one second. Photos were taken, but unfortunately the raw data corresponding to the set of detections was not recovered from the control console post-deployment, and was inadvertently overwritten during the next day's deployment. Attempts made to induce another bout were unsuccessful.

The third period of activity yielded a set of six marmot calls, but the data corresponding to these were lost in the same manner as the marmot bout. To further frustrate, the version of the sink-side application running at the control console had not been updated to reflect the logging changes from Section 5.1.4, where the node id was recorded along side the AML and detection data. Thus, the log data taken by the WSH and the sink-side application were not sufficient to draw a graph similar to Figure 5.5.

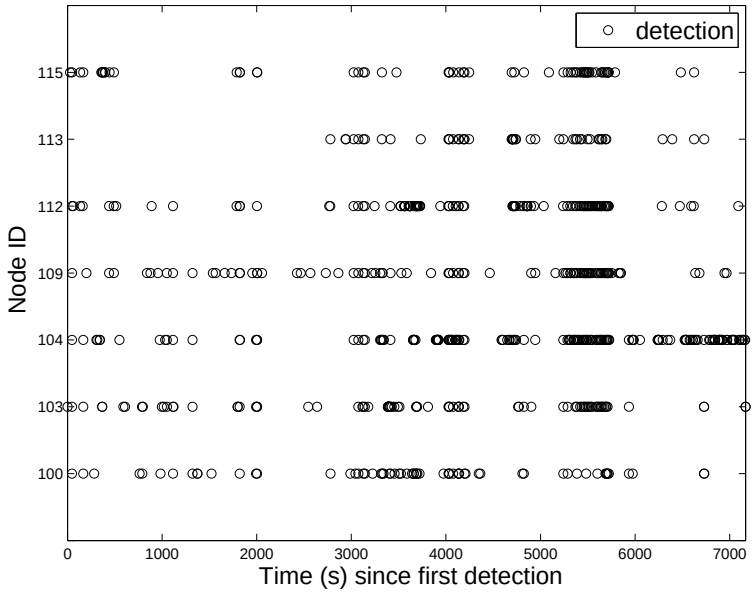


Figure 5.6: A set of events taken from deployment four, dated 18th August, 2007. The x axis shows time since the first detection (in seconds) and the y axis shows node id. Each data point represents the time a detection was triggered at a node.

5.1.6 Deployment four

The fourth deployment took place on the 18th of August, 2007. The marmot localisation application was run for just over two hours whilst marmot alarm calls were induced by a team member. The gateway and control console were placed at the car park, and the network was started in single-hop mode. A larger antenna was available to use which provided one-hop communication to all nodes from the car-park gateway location (which had not been previously possible).

The manual data logs of detected events gathered by nodes showed the frequency at which the event detectors were being triggered, however it did not give a ground truth data stream to compare against. Therefore, it was decided that one of the nodes would record a full audio stream in parallel with the marmot localisation application. This recording could subsequently be used as a ground truth for the events detected by nodes during the deployment. The resulting data set of raw data files totalled 3.8 Gigabyte (GB) (four channels of 16-bit audio sampled at 48 kHz for around 2 hours).

The marmot localisation application was run for a full two hours, and Figure 5.6 shows the entire time series as detections vs node id. The data is clearly noisy, with high incidence of potentially spurious detections (that is, detections which do not match up vertically with any other detections). It is speculated that this was caused by windy conditions, which were seen to cause unanticipated detections.

Within a 2 hour period (7194 seconds), a total of 683 detection events were sent to the sink by nodes in the network. Just 5 out of 683 detections were dropped (a 99.3% success rate). Although 100% data reliability was expected, the data drops were observed to be due to the overflow of the network buffers

(which were 512 kB per stream) during periods of network congestion and high detection rates. When the stream's sending buffer is full, new data is dropped (known as *tail dropping*). This indicates that the arbitrary buffer sizes chosen for streams were too small for the volume of detections that each node was experiencing.

5.1.7 Summary

To the casual reader or WSN theoretician lacking deployment experience, this deployment may not sound successful. However, this is not the case. Firstly, it should be noted that many, if not all WSN deployments actually follow similar success and failure patterns.

For example, the first iteration of several real-life deployments suffered with data low yield and quality, as well as debugging and operational difficulties. Examples include a volcano monitoring project (Werner-Allen et al. 2006), a redwood tree deployment (Tolle et al. 2005), a precision agricultural monitoring system (Langendoen et al. 2006) and a soil contaminant monitoring system (Ramanathan et al. 2006). At a superficial level, the source of these problems is lack of adequate preparation. However, trying to predict the behaviour of even dumb WSNs before deployment is difficult: it is practically impossible to think of every potential failure point of a system before deployment. This is partially due to the fact that some obscure failure conditions are only revealed during in-situ operation. In many cases, it is only through experiencing unpredictable, obscure problems that the deployment and operation of a system can be made more reliable and robust.

However, comprehensive preparation is difficult for first system iterations, due to the physical difficulty of deployment, as well as prediction of corner cases which will cause system to stop working, but only become apparent in unpredictable environments. In the author's opinion (aligned to that of many other deployers), the first rule of deployment is that *everything that can go wrong will go wrong*. To combat this, it is important that the WSN deployer be prepared for unpredictable eventualities by scheduling enough *time* for the deployment. Although only four days were used for deployment here, all of the ten days available were used to improve VoxNet's reliability and robustness.

In summary, this deployment was an exploratory exercise to enable understanding of the behaviour of the system in-situ. As such, the experience gained here was essential in making decisions about further research needed to increase application-specific and general VoxNet system performance and reliability. What follows is a more in depth discussion of the problems that occurred during deployment, from both the marmot localisation application and the general VoxNet context.

5.2 Application related problems

Section 5.2 discusses the localisation application related issues which became apparent during the deployment. The circumstances around the identified problems, and the reasons for their occurrence, as well as potential solutions are detailed.

5.2.1 False detections

The energy-based event detector used to detect marmot calls (as discussed in Chapter 2) is sufficiently lightweight enough to be run on high data rate audio streams: in the marmot case, the sampled audio stream was decimated from 48 kHz to 24 kHz. Because the detector looks only at energy in specific

frequency bands, it will only be triggered by transient signals that are sufficiently above the estimated noise level in the given frequency bands. However, false detections can arise from wideband acoustic sources that have energy in a wide range of frequency bands, such as a loud hand clap.

During both deployment of VoxNet, and the previous EmStar localisation demonstration, it was noted that a hand clap transients could indeed trigger the event detector. The signal would be picked up by several nodes, constituting a valid event that could be localised, even though it was not a marmot. Additionally, in the field, uncorrelated weather events such as rainfall and strong winds were observed to trigger the event detectors frequently (see Sections 5.1.4 and 5.1.6). These uncorrelated false detections may also be interpreted as observations of the same global event.

To address this problem, events detected using the energy-based detection technique could be submitted to a more rigorous examination to determine if they require further processing (for example, through the AML). A suitable candidate for this approach is the application of automated classification techniques. Work has been presented in the literature on the use of Hidden Markov Model (HMM)s to classify species of acorn woodpecker (Trifa 2006) and other types of bird (Trifa et al. 2008). In the approach presented by Trifa et al. (2008), the HMM is trained against reference bird calls to create *observation vectors*, which can be used to classify the signal as it evolves. The *observation vectors* can be created using techniques such as building Mel-Frequency Cepstral Coefficient (MFCC)s, Linear Predictive Coding (LPC) or Artificial Neural Network (ANN)s (Hosom et al. 1998). Through experimentation, Trifa et al. (2008) observes that MFCCs are more suited to recognition of bird song than LPC, due to sharp transitions that are present in bird calls.

Whilst the evaluation of an observation vector in a trained HMM is not computationally complex, the creation of the observation vector may require significant processing. For example, to find the MFCCs of a signal the following steps are required: (1) compute the DFT of the signal, (2) map the resulting spectrum to the Mel scale, (3) take the log of the power at each Mel frequency, and finally (4) take the Discrete Cosine Transform (DCT) of the powers.

Classification will require resources in addition to the requirements of energy-based event detection, and so may not be suitable for real-time analysis of signals, depending on the computational capability of the intended deployment platform. It is possible that the 32-bit microprocessor on the V2 ENSBox could support such a processing chain, although optimisation may be required.

Aside from its uses to aid event detection, classification is a generally important component of the higher-level sensing goal for smart bio-acoustic sensing: automated census for different species of animals and birds.

5.2.2 On-line event grouping

In deployment one (Section 5.1.3), it was found that detection data was not arriving from nodes fast enough to allow grouping. The latency of data transfer effectively rendered the localisation stage unusable, because it could not be guaranteed that events arriving at the sink were temporally close together in terms of detection time.

The approach to grouping detections arriving from nodes during on-line operation was developed for the original EmStar based implementation of the acoustic source localisation system, as stated in Chapter

4. In this approach, if several detections arrived at the sink, each within 0.5s of one another, they would be processed as if they were observations about the same event (DoA estimation, then input into the localisation algorithm). Any detections that arrived later than 0.5s after the previous detection were not considered part of the same group. In the first system iteration in Chapter 4, this grouping technique worked under controlled demonstration conditions (using a dog whistle). However, in the field detection data was sometimes delayed, especially when nodes were trying to send many detection messages at the same time, possibly as a result of false detections. As a result, the 0.5s timer was not sufficient to capture more than one or two detections at a time. However, increasing the time window would have increased the possibility that unrelated events would have been forced into the same localisation computation.

Regardless of the cause, a better solution for grouping data as it arrives at the sink is required, based on detection time of events rather than arrival time. Section 5.2.2 proposes a sink-side *on-line grouping* algorithm for grouping detections received from multiple nodes, based on detection time.

The *on-line grouping* algorithm uses two rules to group data arriving at the sink: *temporal* and *identity* consistency. *Temporal consistency* requires that detections are suitably close together in terms of their observed detection times, and *identity consistency* requires that only one observation from any one node must be present in a given set of data that is used to estimate a position by the sink. The aim of the algorithm is to organise incoming data into groups that bear resemblance to an actual acoustic event of interest. Subsequently, groups that have enough observations (at least three) can be further processed with the DoA and source localisation algorithms.

Assume a network of n sensor nodes, each with a unique id from 1 to n. Nodes make detections d to which they locally assign a sequence number s such that combination of node id and sequence number make a globally unique identifier i such that $d_i = d_{ns}$. The detection itself d_i consists of a *tuple* $\langle d_{det}, d_{raw} \rangle$, where d_{det} is the detection timestamp, d_{raw} is the raw data sample corresponding to the detection. These detections are sent to the sink. The sink maintains groups of detections in a list G where the most recent group is always the g_{curr}. The number of detections in G is given by $n(G)$. Each group $\{g_1, \ldots, g_{n(G)}\}$ in G consists of a set of detections $g = \{d_{i,1}, d_{i,2}, \ldots\}$. Each detection also has an associated watchdog timer w_g, used to trigger processing. The number of detections currently in a group is given by $n(g)$. When the new detection d_i arrives at the sink, the grouping algorithm checks whether it can be added to the current group, g_{curr}. To complete the definition, assume the function $I(d)$ produces a node's id n, and the function $T(d)$ produces the detection timestamp d_{det}. Assume also that function $m(g)$ produces the mean timestamp of a detection group given the group of detections. There are two tune-able global variables the sink uses—the *uncertainty factor* δ and the *watchdog timeout* value l. The *uncertainty factor* defines how close a detection's timestamp must be to the current group's mean timestamp $m(g_{curr})$ to be eligible to join the group. The *watchdog timeout* value dictates how long the sink should wait before checking whether a group's data should go through the AML and data fusion algorithms.

At least three detections are required to make a position estimate, but up to n (the number of nodes in the network) can potentially be added (if all nodes trigger a detection). Thus, it does not make sense to trigger the AML and localisation stage when only three detections have been received if there is a possibility that more may arrive from other nodes. Therefore, the watchdog timer provides a window in

which all the nodes in the network can potentially send detections to the sink. When a new group is started, the sink sets the timer associated with it to timeout after l seconds. The choice of l is a trade-off between allowing a large enough window that all nodes can transfer their detection data and providing a *timely* end-to-end position estimate to the user. Once a detection has arrived at the sink, the algorithm performs four steps:

1. Evaluate identity consistency
2. Evaluate temporal consistency
3. Place detection in correct group
4. Check and process group (timeout-based)

Step 4 is event based, in that a timer is conditionally set during the flow of steps 1–3, triggering some time after a detection has been added. Each step is now described in detail.

1. **Evaluate identity consistency**. When a detection tuple d_i arrives at the sink, it must be checked for identity consistency against the current group g_{curr}. The identity consistency rule asserts that nodes in a group are unique, or, if $I(x)$ gives the node id associated with detection x, then,

$$I(x) = I(y) \Rightarrow x = y \tag{5.1}$$

   for all $x, y \in g_{curr}$. If a detection fails the identity consistency check, it skips step 2 and goes to step 3.

2. **Evaluate temporal consistency**. After checking for identity consistency, the detection d_i must be checked for temporal consistency to see if it can be added to the current group. The detection's timestamp $T(d_i)$ is compared to the current group's mean timestamp $m(g_{curr})$. There are three possible outcomes in testing a detection for temporal consistency:

   (a) If the absolute difference between the detection timestamp and the current mean of the group is less than the uncertainty value ($|T(d_i) - m(g_{curr})| \leq \delta$), then the detection d_i should be added to the current group g_{curr}. If there are no detections in g_{curr}, the detection d_i is added, and the watchdog timer w_g set to l.

   (b) If the absolute difference between the detection and the current mean of the group is greater than the uncertainty value ($|(T(d_i) - m(g_{curr})| > \delta$) and the difference is positive ($(T(d_i) - m(g_{curr})) > 0$), then a new group g_{curr+1} should be started (because the event happened more recently than the current group and is not part of it). The previous group g_{curr} should be checked to see if it can be processed further.

   (c) If the absolute difference between the detection and the current mean of the group is greater than the uncertainty value ($|T(d_i) - m(g_{curr})| > \delta$) and the difference is negative ($(T(d_i) - m(g_{curr})) < 0$), then the event happened before the current group g_{curr}. A search is initiated to find the correct historical position of the detection (step 3: **find historical position**).

3. **Find historical position**. If the detection fails either the identity or temporal checks against the current group, the algorithm initiates a search back through previously created groups to find the correct historical position for the detection. This is done by tracking back through previously created groups, and repeating the identity and consistency checks, until a position in a group is found. There are two corner cases the algorithm must consider:

   (a) The detection cannot fit into any of the existing groups. In this case, a new group is made in between the two groups that the detection is closest to.

   (b) Although the detection passes the temporal and identity consistency checks for a group, an older existing group in G has a smaller gives a smaller residual value for $|T(d_i) - m(g)|$, the difference between the group mean and the detection timestamp. To address this, the distance from the mean of the previous group is recorded—when the current mean is larger than the previously recorded mean, this indicates the best group has been found.

4. **Check and process group (timeout-based)**. When the watchdog timer w_g times out (after l seconds), the group g associated with that timer (which *might not* be g_{curr}) is checked to see if it can be processed further. This is done by checking if the group has more than three detections in it. If $I(g) \geq 3$ then the group is ready for *further processing*

   Further processing refers to the AML and data fusion algorithms used to estimate direction of arrival and position respectively.

This algorithm can be used to group incoming detection data as it arrives at the sink in a more intelligent and effective way than trying to use detections that arrive around the same time as input to the localisation algorithm. However, as it stands, this algorithm supports only the grouping of raw data corresponding to detections as it arrives at the sink. This still means that raw data corresponding to false detections are being sent whether or not they correspond to a real network wide event (that is, a marmot call). To address this, the online grouping algorithm is further developed in Chapter 6, to support selective requesting of relevant data from nodes, based on whether it appears to be temporally correlated. This approach reduces the amount of data that needs to be sent over the network, significantly, in noisy environments. Thus, the evaluation of the modified grouping approach is presented in Section 6.3.2 on page 162, in reference to the degree of data reduction possible when combined with a suitable data collection process (a visual representation of the grouping is shown in Figure 6.11 on page 164).

5.2.3 Localisation algorithm

The localisation algorithm was not extensively used during the four deployments because of on-line data grouping issues discussed in Section 5.2.2 and in Section 5.1.3.

To briefly review the original description in Chapter 2, the localisation algorithm creates a 2D event space to search (with a given distance resolution) and computes the pseudo-likelihood of the target being at each point in the search space (based on nodes' AoA estimations). This means that the search space is exhaustively searched: every possible value in the search space is computed before determining the position. The only optimisation of processing the algorithm allows is to reduce the resolution of the

search space (that is, increase the geographical size of each point in the search). However, this approach also limits the accuracy of the position estimation (it will be quantised to the resolution of the search space).

Since each point in the two-dimensional search space is the sum of N calculations (for N nodes contributing to the localisation algorithm), the time taken to process each element is $O(n)$. However, the time taken to process all of the elements in the search space relative to the resolution is $O(n^2)$, which far outweighs the per-node time complexity. Therefore, as the size of the search area or the resolution of the search space increases, the time taken will rise by n^2. This may induce unacceptable latency for on-line localisation results.

Instead of an exhaustive search, a *multi-resolution search* would perform better. A multi-resolution search starts with a with a low resolution scan over the space, zooming in on the most likely space each time, thus increasing the resolution. For example, a 100m by 100m grid could be divided up into 10m squares, so that only 10 points would have to be searched initially. The 10m by 10m point that has the highest pseudo-likelihood could then be divided up into a 1m resolution grid, which would require a search through a further 10 points, giving a 1m resolution. This accuracy can be further improved by taking increasingly smaller grid sizes.

Assuming the search grid of 100m by 100m and a required resolution of 1m, the grid method will search 10000 points, and the multi-resolution approach will search only 200 points, a 98% reduction in computation. From the point of view of end-to-end latency with respect to position estimates, it may be important to maintain a constant time in which the localisation algorithm completes. This could be accomplished by adaptively scaling the resolution accordingly, depending on the search space size and the number of nodes contributing likelihood observations.

Regardless of the computational complexity, the assumption that the localisation algorithm makes is that that the acoustic event happens within the search space defined around the network of nodes. In the case of the marmot localisation application the scientist will want to surround the marmot burrow with nodes. For acoustic localisation of other animals it may not be sufficient to assume a set search space. For example, birds may move more dynamically over a wider area. If the area being monitored is well-known and limited, then this approach is reasonable and helps to narrow down the search space. However, if the sound sources are likely to move outside of the search area, then this approach will yield incorrect position estimates. This is because the search space will still have a maxima, but it will not be consistent with the actual node position. A similar problem was found in the ToA consistency function approach presented by Balogh et al. (2005) which was discussed in Chapter 2.

Finally, the approach of using a linear, closed-form angulation solution (similar to lateration discussed in Chapter 2) would only work if a single angle estimate was used from each node, rather than a likelihood vector. This approach would be faster, but is less likely to be accurate especially if the individual AML maximum likelihoods are inaccurate, as was observed by Ali et al. (2007).

5.2.4 Summary

Section 5.2 has discussed the application-specific problems that were observed from the deployment of VoxNet: false detections, on-line data grouping and the choice of localisation algorithm. Each of these had

an impact on the success of the RMBL deployments, and accordingly improvements have been suggested to improve each of them. Chapter 6 discusses on-line grouping in more detail, but false detections and the choice of localisation algorithm are left to future work. The problems that were more general to the VoxNet platform are now discussed.

5.3 General VoxNet problems

The challenges and features presented in Section 5.2 were specific to the marmot localisation application. Section 5.3 identifies the general problems and challenges for the on-line operation of the VoxNet platform that were observed during the deployment of the marmot localisation application. These include *data logging*, *stream priority* and *data consolidation*.

5.3.1 Data logging

A hugely underestimated factor for the successful deployment of the marmot localisation application was logging of raw detection data and system health information.

The sink-side application recorded all detections sent to it by nodes in the network, and all AML results to file for later analysis (in ASCII rather than binary format).

During each deployment, each node recorded the link quality and multi-hop data routes to neighbours, every 10 seconds. The logs were timestamped with local and global timestamps and were used to analyse the quality of the links and network routes over time. In addition, the output from both the WSH and the sink-side localisation application were saved to disk at the control console. The type of data logged by the WSH and sink-side application became increasingly more detailed with each deployment. From the second deployment onwards, the WSH recorded discovery updates from each node and outputted a summary of data to a log file every 10 seconds, providing details of all of the nodes connected to the control console, their current local and global timestamps, local up-time, connection state and the current size of audio buffer queue. In addition, information about all of the streams each node exposed and the sizes of queue for each stream were recorded.

Unfortunately, the data gathered during deployments one, two and three were practically unusable due to insufficient log data being recorded at each node. It was only during the third and fourth deployments that enough data was logged on the sink-side application and the WSH to be able to analyse the temporal patterns of detections across the network. There were some oversights which invalidated data: in the first, second and part of the third deployments, nodes sent detection data that was timestamped with a local detection time (rather than a global timestamp). This meant that detections could not be correlated between nodes. The WSH timestamped the arrival of any messages from the nodes, but that was not sufficient. Additionally, on the sink side application, the detection recording originally did not include the node id with the raw detection. This was only fixed part way through the third deployment.

A more subtle problem was that of timestamping at the control console: although the control console was time synchronised through the network, some data logs used the local system time of the control console, and some used the global network time. This made it difficult to correlate the control console and sink-side application logs post-deployment. In addition, the format of timestamps was confusing: link estimates and multi-hop routing table logs were timestamped using a coordinated universal time

(UTC) timestamp, which was offset from the local timestamps (which were in the mountain time zone in Colorado). Translating between these timestamps was open to errors, especially when trying to relate topological and link data with events that occurred in the logs.

Part of the reason for this problem is that the back-end, off-line aspects of the VoxNet system were not implemented for the deployment. Thus, consideration had not been given to making sure that all data logged was using a common time format. This would have been a necessity for automated consolidation of data from the nodes, post-deployment. This practical aspect of automated data consolidation is considered in Section 5.3.3. Additionally, during deployment, nodes did not use the spill-to-disk functionality discussed in Chapter 4. Making a full, synchronised record of the audio stream gathered (or even a single channel) on each node would have been invaluable for post-deployment analysis of false detections.

5.3.2 Stream priority

In Chapter 4, networked streams were introduced as a useful way to enable the flow of Wavescope programs over multiple network hops, as well as to communicate lower-level data, such as log messages, control data and new binaries between sink and nodes. The general operation of the subscription server is that all published streams are given their own TCP connections. Each stream is serviced equally by the subscription server (due to the underlying TCP server). However, there may be cases where it is more important for some networked streams to be treated with a higher priority than others. For example, it may be that Wavescope stream data takes higher priority than control or log data. It may be more important for the sink to be able to send commands to nodes than for nodes to send log messages.

To implement priority for networked streams, it is necessary to ensure that certain streams are serviced at a higher frequency than others. The frequency of service can be related to some arbitrary scheduling scheme.

5.3.3 Data consolidation

The off-line operation context of the VoxNet architecture was not discussed in detail in Chapter 4 because the application at hand was primarily motivated by on-line localisation. However, VoxNet is intended to be both an on-line and off-line platform, hence it is important to consider how data gathered during the on-line operation of the system will be stored, and accessed at a later date. Data consolidation is considered here whilst data access is considered in Section 5.4.1.

During the deployment of VoxNet, a significant amount of application data (raw detections, AML results, position estimates) and system health data (multi-hop routes, link quality, per-node stream status and so forth) was gathered and used to analyse the performance of the system. However, Section 5.3.1 observed that the lack of a unified system for logging and timestamping led to inconsistencies which rendered otherwise useful data unusable for analysis.

During the VoxNet deployment period, the typical way to gather data for post-analysis with the acoustic nodes was to remotely copy data files using `scp` (for system health and application logs), and manually extract Compact Flash (CF) cards for raw audio data, copying the data to a laptop. Data created at the control console was manually copied from the main application directory into the same place as the remote data. The manual approach left great potential for forgetting to transfer certain data or inadvertently over-write important data. In deployment three (Section 5.1.5), this caused the loss of

data corresponding to a rare marmot bout.

A management interface to make this process easier, more reliable or even automated would be vital to prevent data loss and organise data effectively for later access.

For the marmot localisation application, and likely other similar attended applications, the data need only be consolidated at the end of the deployment. In this case, the data consolidation could be implemented as part of the control console functionality. Nodes could be set into a special *download mode*, where all data is sent to the control console and automatically organised into streams: for example log streams and data streams. A key concept to VoxNet's interaction model is the unification of the on-line and off-line modes of VoxNet so that they are seen as part of the same process.

5.3.4 Summary

Section 5.3 discussed several problems that were observed about the general operation of VoxNet through the marmot localisation application: data logging, stream priorities and data consolidation. Out of these problems, insufficient data logging caused the most problems during post-deployment analysis, closely followed by data consolidation. These are problems which could vastly affect how easy VoxNet interact with after deployment, as well as the post deployment analysis of both application and log level data (for the scientist and WSN developer respectively). The unification of these problems in the overall VoxNet platform are left for future work, but the implications for adoption of the system are discussed below in Section 5.4.

5.4 Enabling scientific adoption

So far, marmot-localisation specific and general VoxNet issues related to deployment experiences have been described. The experience of deploying VoxNet enabled these observations to be made. However, there are still some issues that whilst not observed directly in the field, the author believes are important to enable the adoption of the system by domain scientists. The issues discussed in Section 5.4 are: data access and fault tolerance for robustness and reliability.

5.4.1 Intuitive data access

As discussed in Chapter 4, VoxNet intends to bridge the gap between on-line and off-line performance, first by allowing the same applications that are run on-line to be run on archived data, and second by providing a way to consolidate and access the on-line data gathered through deployment in an off-line context. The *archive server* is integral to this vision: Section 5.3.3 discussed how the collection of data from on-line deployments (from both the control console and the nodes) might be consolidated to the archive server.

In this approach, all data logs as well as recorded data streams are archived by the server, to be retrieved and analysed by the user. For the marmot localisation application, the time-stamped log data streams, detection streams and AML streams could be used to provide annotations for the raw data streams gathered. These streams could be visualised together, and would aid the deployer in off-line data analysis; it would also assist a user who was not present at the deployment to quickly make sense of the data. This approach would require that all log, control and other data sources were timestamped in the same format (as noted in Section 5.3.3), and that the system had a way to deal with data that was

gathered during a period with no synchronisation. Section 5.4.2 discusses more the implications of time synchronisation failures for system usage.

The aim of the archive server is to provide a search-able archive of all data gathered as part of on-line VoxNet deployments. This is to enable the user to analyse data after field experimentation. The *Seismogram Transfer Program* (STP) (Anonymous 2007*c*), for example is a program that allows users to request data from the Southern California Earthquake Data Centre and provides a web-based interface, and a shell console that users can use to query subsets of seismic data corresponding to earthquake events. This data can have filters directly applied to it, and in some way resembles the same kind of interface that Matlab might provide to data streams. Users can request lists of most recent data events, with ids, which can then be cross referenced. This allows queries along the lines of *list available data corresponding to event x*. For VoxNet, annotations provided during on-line operation could be used as information to help the user in querying data—for example *get data from all nodes which was gathered around the time of event x*.

Deciding what are correct interfaces (graphical or otherwise) to help summarise data for the browsing user is beyond the scope of this discussion, but present interesting research topics. Powerful graphical interfaces that can integrate raw acoustic data, photographs and other processing (direction of arrival, position estimates, others) and notation may help the scientist make other meaningful observations.

5.4.2 Automation for fault tolerance and robustness

In VoxNet, there are two important system-level services whose correct operation is dependent on the wireless networking links: time synchronisation and network latency. For time synchronisation, if the network, or a subset of the network loses time synchronisation at any point during system operation, this may invalidate any of the observations and processing that are being performed. Time synchronisation performance is related to the quality of the links that the data is travelling over. If links consistently fail, nodes may find themselves without neighbours to synchronise with, and will thus lose time synchronisation. For network latency, transient loss of link quality can affect the speed of data transfer in the wireless network. This can affect the perceived *timeliness* of the system when it is being used in an on-line manner. If the user is basing observations on the output of the system (as with the marmot localisation application), then it is important that the results arrive in sufficient time to allow the user to perform their observation (or take a picture, for example), while the phenomenon is occurring.

If either time synchronisation or network latency services become compromised, then the quality of the information provided by the application level processing will be affected (since data is not timely or synchronised adequately). Ideally though, the user should not have to care about these lower-level services, and should be assured that they are fit-for-purpose over time. In dealing with potentially transient problems with the network, it is important that the system attempts to mitigate these problems in an adaptive, automated manner. However, there may be some instances where the network cannot adequately resolve the situation, and must therefore inform the user of the problem so that they can adjust their confidence in the measurements being made, or can perform remedial actions. Ideally, the system would provide *hints* or advice to the user about the best way to resolve the problems.

5.5 Summary

For the marmot localisation application, the on-line usage cycle followed three main phases:

- *set-up*: physical deployment and initialisation of system
- *operation*: attended use of the system whilst running
- *post deployment*: collection of physical devices to return to laboratory environment; consolidation of data to storage

These three phases correspond to the on-line operation context of VoxNet, coupled with the subsequent transition to off-line operation (but not actually including the off-line processing of data). Deployment of VoxNet raised several challenges with respect to the usage cycle deployment and usage cycle. These challenges fell into three categories:

1. *Application specific support and improvement*
2. *General support for on-line operation*
3. *General support for off-line operation*

For the immediate operation of the system, the first two categories (application specific support and on-line operation) are the most important.

Since the application is running on-line, it is important that it is perceived to be running in a timely manner to the user: an excessive amount of time waiting for data to arrive at the control console may affect whether the scientist can make in-situ marmot observations or not. A lack of robustness relating to (potentially transient) network latency could affect the scientist's trust in the system or willingness to adopt it for in-situ use.

End-to-end latency has been at the core of several problems with the marmot localisation system during deployment, notably data transfer latency and localisation algorithm latency. Latency of data transmission is something which directly influences the performance of the on-line marmot localisation application. The latency of transferring raw detection data from the node to the sink-side application affected the data grouping algorithm's ability to decide when to group data and process it further. This was because detections could be delayed in their transfer, and arrive after the detection timer had expired (0.5s). This is a fundamental problem in the operation of the system: without a suitable grouping solution, performing localisation on-line is practically impossible.

This problem was compounded by false detections: when nodes generate a lot of detections in a short space of time, for example from false detections, this creates a lot of data which must be sent by nodes over the network. This affects the end-to-end latency, as all the data has to be sent, delaying the meaningful data that can be used by the sink-side application. Therefore, a way to filter out the noise of the unwanted detections will have an effect on the amount of data that needs to be sent by nodes, thus helping to reduce transfer latency in the network.

Based on the above observations, the rest of the work in this thesis concentrates on increasing the robustness and timeliness of the marmot localisation system by identifying the exact the causes of latency in the localisation system and providing improvements to reduce that latency through data filtering and adaptive processing of data.

Chapter 6

Reduction of end-to-end localisation latency

Chapter 5 discussed the in-situ deployment of the marmot localisation application implemented on the VoxNet platform. The deployment raised many potential issues with respect to application and platform operation. The two most important and far-reaching issues were of data transfer latency and intelligent grouping of data.

This chapter addresses these factors in more detail, in the context of end-to-end timeliness of the marmot localisation application. For WSN applications that run in an on-line manner, such as the marmot localisation application, time is an important constraint. Some applications within this class may require that data sensed in the network is processed within strict timing deadlines (these are *hard* or *soft* real-time systems depending on the implications of missing the deadline). Other applications within the class have a less strong requirement: that the system must show a level of *timeliness* with respect to results delivered to the user. The marmot localisation system is an example of a system which should maintain timeliness wherever possible.

This chapter discusses the sources of latency which affect timeliness in the marmot localisation system, as identified by the author upon design and evaluation of the system as described in Chapters 4 and 5.

The rest of this chapter is organised as follows: Section 6.1 describes three distinct areas in the end-to-end processing chain where latency is introduced: on-node processing, network data transfer and sink-side processing. Of these, network data transfer is the largest contributor to end-to-end latency. An analysis of the sources of latency in the network data transfer are considered from application level down to physical level. To improve overall system timeliness, a case is made for on-node processing in Section 6.2, and for temporal filtering of event detections at the sink before data collection in Section 6.3. This is supported by proof-of-concept simulations using network transmission data gathered through controlled, in-situ experimentation (from Chapter 5). This was sufficient to provide proof-of-concept validation that could be readily extended to real deployment.

Three algorithms developed by the author are presented in this chapter: Lazy Grouping, Static Threshold Adaptation, and Dynamic Prediction Adaptation. Lazy Grouping supports the grouping and collection of data for position estimation. Lazy Grouping is a modification of the On-line Grouping algorithm (presented in Chapter 5), which gathers only data which is useful for position estimation. Static Threshold Adaptation and Dynamic Prediction Adaptation support adaptation of processing from sink to node in the network, based on network conditions.

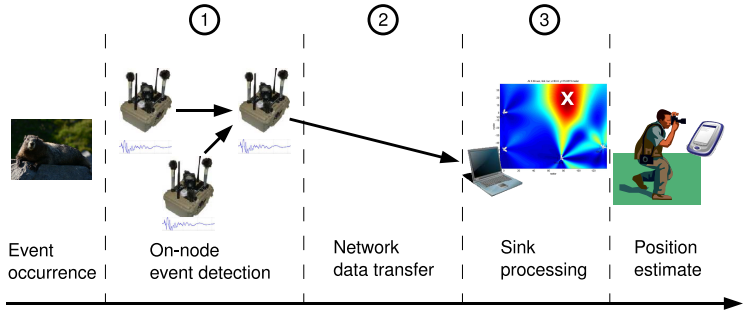


Figure 6.1: The marmot localisation use case, divided into three discrete components representing the three broad sources of latency in the end-to-end WSN system—on-node detection/processing (1), network transfer (2) and sink-side processing (3),

6.1 Causes of network latency

In the on-line marmot localisation system's motivating use case, the scientist deploys a network of nodes to monitor marmots in their habitat. The scientist waits at the sink node (most likely a laptop) to receive position estimates of marmots soon after their calls have been detected, subsequently using this information to position a camera to take photos of the marmots, or to help decide if the physical topology of the network needs changing. The end-to-end performance of the WSN system in this use case is judged by how quickly the network can identify an event of interest, estimate the location of the event and present the user with this estimate.

Figure 6.1 shows the processing chain from event detection to presentation of position estimate to the user for the use case. From a WSN perspective, the marmot localisation processing chain starts at node level, where the nodes run event detectors to detect marmot calls. When such events are detected, the corresponding data is placed on a sending queue and transferred over the network to the sink. At the sink, each detection is processed using the AML algorithm (described in Chapter 2) and related detections are then fused together to determine the position estimate, which is displayed to the user. The user's expectation is that a position estimate will arrive in a *timely* manner, such that it can be used to make specialist application related observations or take decisions. Therefore, the *timeliness* of the system is an important measure of its end-to-end performance. Whilst a specific timing deadline is not imposed on the system for converting detections to position estimates (unlike a soft or hard real-time system), the longer the user has to wait for the result, the less likely it is that the information is useful. Therefore, the focus of the system is on presenting results *as quickly as possible*.

Broadly, in the processing chain, there are three contributors to the end-to-end latency of position estimates (thus timeliness) in the system: node latency, network latency and sink latency. Node and network latency is due to on-node event detection and the transfer of that detection data to the sink. Sink latency is due to the processing detection data through the AML and data fusion algorithms, and the graphical presentation to the user.

As described in detail in Chapter 5, the first in-situ deployment of the marmot localisation system

took place at the Rocky Mountain Biological Laboratory (RMBL) in August, 2007. During this period, controlled experimentation was also carried out. Eight nodes were deployed, although not all were used in every experiment.

During the in-situ deployment, it was observed that network transfer was by far the largest contributor to the end-to-end latency in the system. The on-node event detection ran in real-time, and the sink processing load was nominal (processing each AML data took on the order of tens of milliseconds). Several factors were observed to affect the time taken to transfer data from the nodes to the sink: the event frequency, the multi-hop topology and the antenna gain. These factors were further examined by the author through in-situ operation and controlled experiments. Two other factors were also suspected to contribute to data transfer latency, but were difficult to isolate through in-situ experimentation—the data transport mechanism, and the data rate. All factors above have analogies to the abstraction layers in the TCP/IP model, as indicated by Figure 6.2 on the following page. This figure shows the individual components that contribute to the overall network latency stacked from physical to application level. The latency contribution of each of these components is discussed below, starting from the application level and working down the stack to the low-level aspects, such as those that affect the physical transmission of the data over the network.

Fundamentally, the amount of time the data will take to transmit is a function of the amount of data that needs to be sent and the rate at which it is sent (that is bits or bytes per second). This rate has different meanings at different levels in the system stack. At the physical level, this refers to the rate at which bits can be physically encoded on the communication channel (the data-rate or bit-rate). This represents a physical limit on the speed of the transfer (discussed further in Section 6.1.4), and does not consider individual packets of data. At the data transport level, the rate at which packets of data are received is called the *throughput*. Throughput includes the frames and header information included in each packet. At the application level, the rate at which useful data is received is called the *goodput*. Goodput takes into account only the data payload of a packet, disregarding the frame and header information.

6.1.1 Event frequency

From the point of view of the nodes in the network, the event frequency represents the highest level factor that can affect transfer latency in the end-to-end system. In the motivating application, individual marmots call relatively infrequently, although several marmots may call within a short period, or an individual marmot may produce what is known as a *bout*, where it makes as many as 20 calls in a row, at approximately 1 second intervals. However, in the context of the system, event frequency refers not only to the rate at which the animals call, but also to the rate at which the event detector running on the nodes is triggered. In an ideal scenario, the trigger rate and the call rate would be identical. Realistically, there are however false positives which can trigger a node's event detector. Nonetheless, as the frequency of events being detected by each node in the network increases, it follows that the amount of data that needs to be sent back to the sink will increase. Because outgoing data is sent to a message queue, it will take longer to transfer the most recent detections if there is a large amount of data waiting to be sent.

To show the effects of event frequency on data queues, and the subsequent transmission times, an example is considered from a data trace of actual detection events from the system running in-situ, at

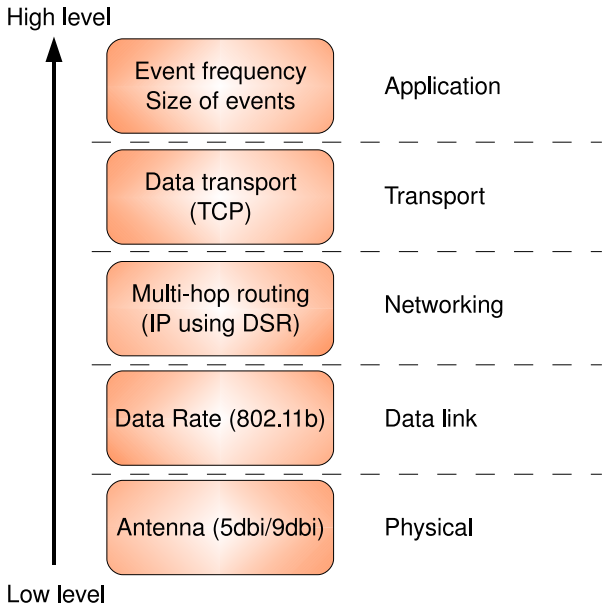


Figure 6.2: The observed and suspected factors causing latency (left), and their analogies to abstraction layers in the TCP/IP model.

RMBL. Seven of the eight nodes were used (node 108 was not used), with each node one hop from the sink.

Figure 6.3 on the next page shows the effect of a rapid number of detections across the network in a short period of time. The data shown on the graph is for a period of 3 seconds, when 19 detections were triggered across various nodes. Each data point represents a detection (where the x axis is the detection time, and the y axis is the number of events that were queued network-wide when the detection was made). The graph shows how data can quickly accumulate in the network.

Figure 6.4 on page 150 shows the relationship between the time taken for a detection to go from message queue to sink, and the aggregate number of detections in the network at the time of detection. Each data point represents a detection made by a node. The x axis is the amount of queued detections across all nodes when the detection was made, and the y axis is the time that particular detection took to arrive at the sink.

Both Figure 6.3 on the next page and Figure 6.4 on page 150 show that queued detections in the network can quickly increase, and the time taken for a detections to arrive at the sink is related to the total amount of data queued to be sent in the network. Ultimately, this shows that the event frequency, and hence local and aggregate queue size will have an effect on the time taken to transfer data to the sink.

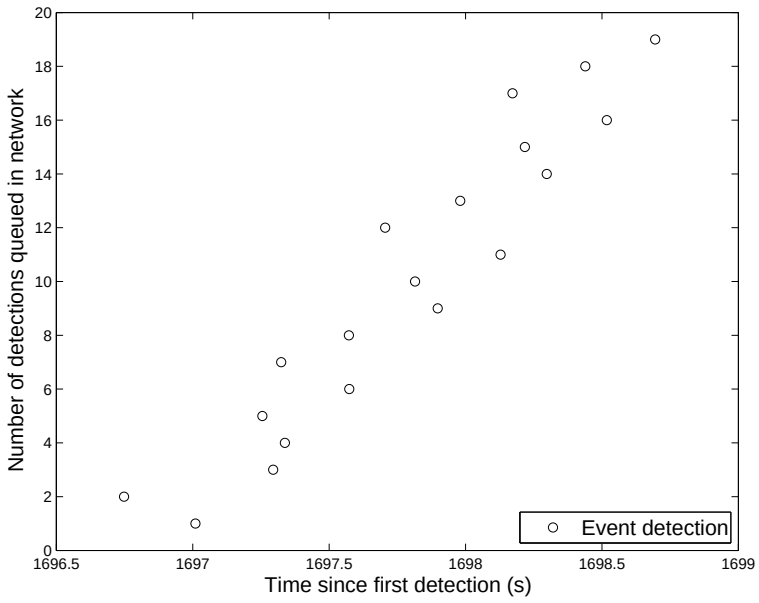


Figure 6.3: A plot of a specific time window from data gathered in-situ at RMBL. The graph shows the number of detections being sent (or queued) vs. detection time. Each point on the graph represents a detection made by a node, with the x axis representing the time since the first detection in the data set was made (not shown in this graph), and the y axis representing the number of events that were either queued or being transmitted in the network at that point. The x axis units are chosen for ease of reading.

6.1.2 Data transport protocol

Whilst the rate at which events are triggering the event detectors on nodes can explain how large amounts of data are generated, the data transport protocol is responsible for making sure that data gets from node to sink reliably and in the order it was intended. The overhead associated with ensuring this means that the data transfer protocol plays an important part in transfer latency. The system uses the Transmission Control Protocol (TCP) as its data transport protocol, a ubiquitous part of the TCP/IP suite which has been integral to data transport in the Internet. TCP takes responsibility for *reliable*, *in-order data transmission*, *congestion avoidance* and *flow control*, all of which are briefly described below. TCP also implements a *fairness* policy, where all nodes that are transmitting simultaneously are afforded an equal share of the bandwidth over time.

To ensure *reliable, in-order data transmission*, TCP implicitly acknowledges successful reception in its ACK reply to the sender by including the sequence number of the next byte it expects to receive. The sender can then re-transmit any bytes that were not received by the receiver. This means corruption of part of a packet (TCP segment) does not necessarily mean the whole packet must be retransmitted. *Flow control* allows the receiver to adjust the rate at which the sender is sending data (for example the sender may be sending too fast for the receiver). This is done by the receiver specifying the amount of data it is willing to receive before acknowledging its successful arrival. The sender cannot send more data until it

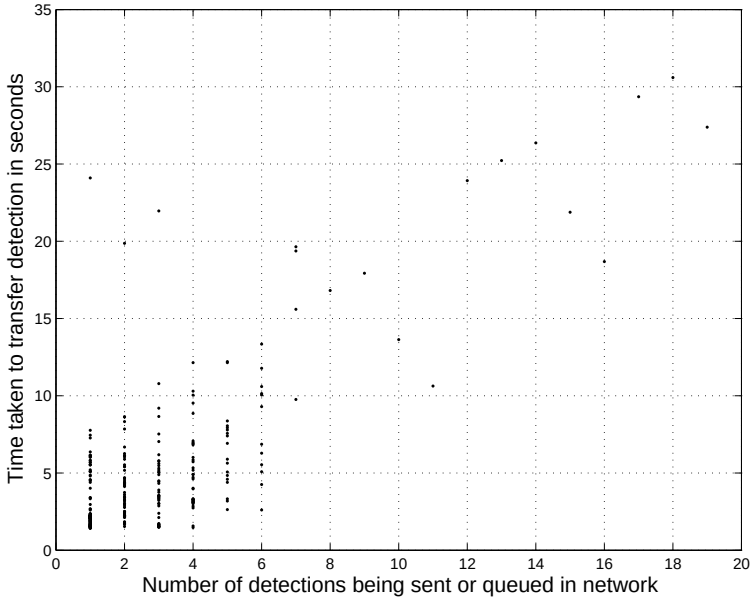


Figure 6.4: Time taken for a node to transfer 32 kB of data (a detection) to the sink, vs the number of detections currently queued or mid-transmission in the network. Each data point represents a transfer, and how long it took to arrive at the sink from when it was queued.

has received the acknowledgement (the sending of which may be delayed using a timer at the receiver), thus its rate is adaptively controlled.

Similarly to flow control, TCP implements a *congestion avoidance* scheme by using a per-connection congestion window, which limits the number of segments from a sender that may be in-transmission without acknowledgement from the receiver. On wired networks, TCP interprets packet loss as an indication of *congestion* in the network (there is more data needing to be sent that the network can actually carry at that instant). TCP infers packet loss at the sender: if the sender receives duplicate data acknowledgements for the same TCP segment from the receiver, then there must be packet loss. When congestion occurs, TCP reacts according to the policy of the congestion control algorithm being used.

Essentially, all of these algorithms provide some back-off from transmission (to ease the congestion), followed by a period of progressively increasing the congestion window size (hence the data rate). This helps the sender establish whether the channel is still congested or not (in which case it would back off again), and is known as *slow-start*. The nodes and gateway in the system discussed in this thesis run version 2.6.10 of the Linux kernel, which uses the Binary Increase Congestion control (BIC) algorithm. BIC is optimised for high bandwidth networks with long Round Trip Times (RTTs).

Whilst in wired networks the assumption that packet loss is entirely due to congestion is valid, this assumption does not hold in wireless networks. This is because wireless networks experience additional losses unrelated to congestion—that is, a packet may not reach its destination due to interference or loss of signal strength in the wireless channel. Loss caused by the physical properties of the wireless channel

(rather than congestion) can cause TCP transmissions to back off and slow transfers down in an overly conservative manner. This can result in high latency for transmissions, depending on the sequence of losses that a connection may observe.

The TCP connections in the system here operate over multiple hops, using IP forwarding (as discussed in Chapter 4). This means that the data transport protocol sits on top of the multi-hop routing layer in the system, provided by an implementation of the Dynamic Source Routing (DSR) protocol. It follows that data travelling over multiple-hops has an increased chance of experiencing latency due to the effects of TCP. VoxNet's stream management software tries to mitigate latency by disconnecting and re-establishing TCP connections that have backed off to an inappreciable degree, based on a higher level acknowledgement/timeout protocol (discussed in more detail in Chapter 4).

A reliable transport mechanism is important to the marmot localisation application, as the AML algorithm cannot operate correctly on a partial stream of data. Therefore unreliable protocols, such as the User Datagram Protocol (UDP) are not useful here. UDP is useful when streams of data can trade-off loss with real-time performance. For example, in media streaming, packet loss can be absorbed; the goal is to keep the stream timely, and as close to real-time as possible. The main limitation of TCP in ad-hoc wireless networks is the lack of interference avoidance, due mainly to the lack of communication between link layer and data transport layer.

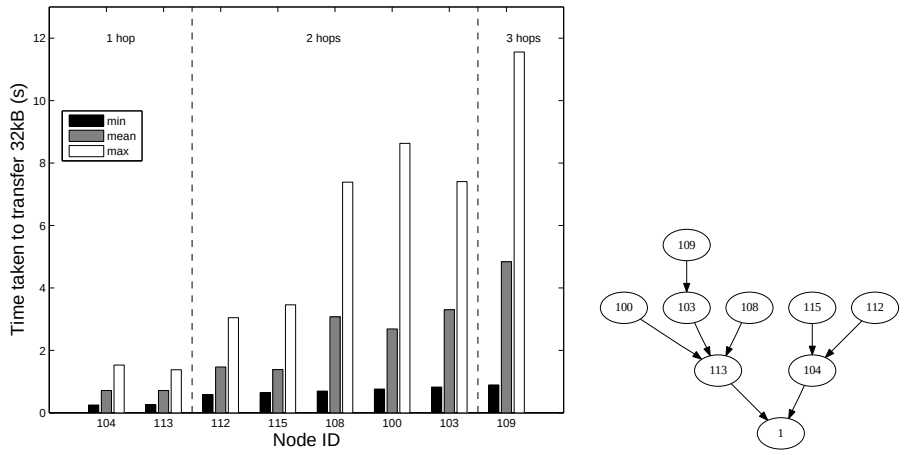
The data transport protocol plays an important part in the potential latency of data transfers. Because reliable data transfer is important for correct operation of the system, the potential latency incurred by using TCP must be traded off against lossy, but faster operation.

6.1.3 Multi-hop topology

Multi-hop message routing in ad-hoc wireless networking allows communication between nodes that are either out of physical communication range with one another or have communication links with a high loss rate.

Multi-hop networks can be used to increase the effective range of a network, but can also reduce the effective bandwidth available to nodes several hops away, as well as increasing the aggregate amount of data passing through the network. The aggregate amount of data that must be sent is a side-effect of packet-forwarding. For example, should a node that is three hops from the sink want to send a packet of data—this has to be forwarded independently by two other nodes, resulting in an aggregate amount of three packets that must be sent in total.

Because the radio channel is a shared medium, message forwarding also affects the availability of the radio channel for all nodes that are within communication range. This effectively reduces the maximum throughput available to a node, the further away it gets from the sink. Generally, the maximum throughput (and therefore goodput) available to a node in a multi-hop network degrades at a rate of $1/x$, where x is the number of hops from the node to the sink (Fu et al. 2003). For example, if the maximum throughput available to a node one hop away is nominally 500 kB/s, then a node three hops away would only be able to achieve a maximum throughput of 166.7 kB/s. With regard to the goodput, since it refers to the rate of application level data rather than overall packet size, the bandwidth values would be smaller but the per-hop ratios would be the same.



(a) (b)

Figure 6.5: Figure 6.5(a) shows the minimum, mean and maximum time taken to send 32 kB in a multi-hop network when all nodes are requested to send data at the same time. Fifteen transfers were recorded for each node, and results are sorted by minimum latency. Figure 6.5(b) shows the multi-hop routing tree for the network during the experiment.

If nodes are not within the same transmission region, some gain in bandwidth can be made through *spatial re-use*, where parts of the network which cannot physically interfere with one another can theoretically transmit at the same time. However, in this case, unexpected radio collisions are introduced by the *hidden terminal effect* where two nodes communicating with a third (or access point) experience unexpected loss (Moh et al. 1998). This occurs because the two sending nodes are not within communication range of one another, hence cannot sense that the other is transmitting.

An in-situ experiment was performed to simulate a *worst case* acoustic localisation scenario for data transmission, where all nodes made a detection within a short space of time and attempted to transmit to the sink-side application at roughly the same time (within several milliseconds of one another). The WSH was used at the sink to periodically request seven of the nodes to send back 32 kB simultaneously (using the `send` command), and the time taken for each transfer was recorded. A total of 15 data requests were made to all nodes. VoxNet's multi-hop routing layer was activated during the experimentation. The resulting multi-hop routing tree is shown in Figure 6.5(b).

Figure 6.5(a) shows the min, mean and max time taken for each node to transmit the data, sorted by the number of hops each node was from the sink. The median latencies were also calculated and found to be close to the mean values. The time taken to transmit the data in this experiment increases considerably with the number of hops (meaning that the goodput is reduced). When comparing the per-node transfer latency to the node's position on the routing tree there is a marked difference in latency between nodes who are children of 104 (115 and 112) and those who are children of node 113 (100, 103, 108). Therefore, it could be concluded that simultaneous transmission in a network such as the one created here had a marked effect on the system latency, which is further dependent on the network topology (particularly

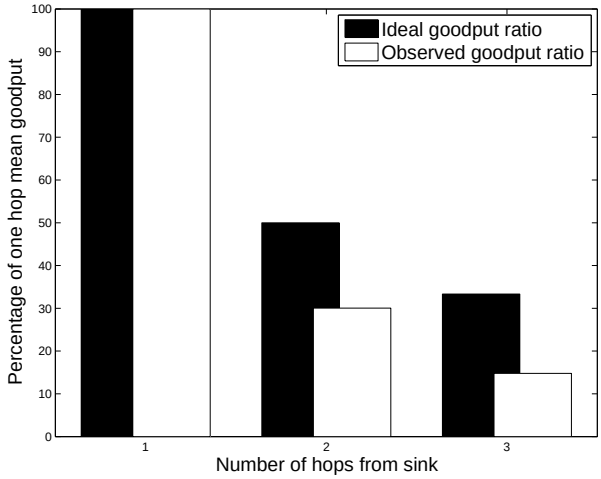


Figure 6.6: The maximum goodput ratio for nodes that were 1, 2 and 3 hops away from the sink. Each data point is the mean of all nodes that number of hops from the sink. The ideal goodput ratio is shown as a comparison.

on how many children a parent node has).

It was noted previously that the expected throughput (or goodput) should degrade at a rate of $1/x$. To verify this, the observed mean latencies were converted to goodput values, by dividing the data size of the application level payload (32 kB) by the time taken to transfer. The overall mean goodput for nodes situated one, two or three hops from the sink were taken for each node, and expressed as a ratio of the one-hop mean goodput. The results are shown in Figure 6.6, along with the ideal goodput ratio. It can be seen that the observed mean goodput ratio is less than what would be expected—30% versus 50% of the one-hop goodput at two hops, and 15% versus 33% of the one-hop goodput at three hops. This reduction of goodput is potentially a side-effect of concurrent transfers.

To make a more compelling case for the hypothesis that concurrent multi-node transmission reduces the goodput ratio below what would theoretically be expected, a controlled experiment was set up in a laboratory environment. A linear six-hop network was formed, with all nodes in the same transmission region, in order to factor out any hidden terminal effects. Nodes were set-up to ignore packets at the MAC layer from all but their one-hop neighbours in the chain. This experimental set-up recreates similar experimental environments used by Fu et al. (2003) to investigate TCP performance in wireless ad-hoc networks.

Nodes were requested to send data payloads of 32 kB or 535 kB. Fifty requests were made for each data size to all nodes individually, and then fifty requests to all nodes simultaneously for the 32 kB payload. The 535 kB payload was chosen because it represented the maximum observed throughput at one hop in the experimental set-up. It was measured using the *iperf* tool before the actual experimentation. Figure 6.7 shows a clear difference in goodput ratio between simultaneous and single requests for 32 kB. The graph shows a larger goodput ratio drop off at two hops when multiple nodes are transmitting,

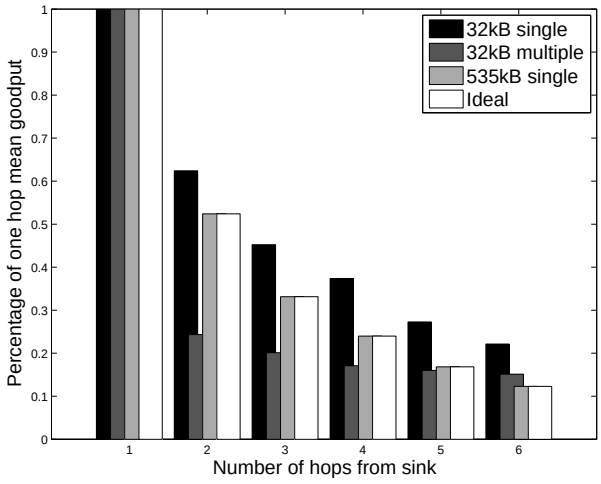


Figure 6.7: A graph of goodput ratio for transfers in a linear, 6 hop network. Each data point is the result of 50 transfers, either 32 kB or 535 kB (the observed maximum 1 hop throughput). The ideal goodput is shown for comparison. Data requests were either to the node at each hop individually (single), or to all six nodes in the network (multiple).

followed by a small decrease per hop from three hops onward. Conversely, the single node transfers degrade at a gradual rate, with the goodput ratio actually being *better* than the expected ratio. This is likely due to the fact that the data payload being sent does not fully use the bandwidth of the channel, so the relative goodputs observed are higher. The result for the single 535 kB payload confirms that when the maximum throughput possible at one hop is requested, the goodput ratio matches the expected ratio at each hop.

For the marmot application, the implications in terms of end-to-end system latency are that if all nodes are sending data simultaneously, the goodput available to nodes more than one hop from the sink will be reduced significantly from what would normally be expected. If however, only one node is sending, one can expect a better goodput ratio if the payload is small in comparison to the maximum one-hop goodput.

In conclusion, the in-situ and controlled experimentation showed that latency was affected by the multi-hop topology, potentially generated by a variety of sources: number of nodes transmitting simultaneously, hops from sink, spatial re-use and data size. However, it was not possible to specifically isolate and quantify these causes on the basis of the in-situ data set. Therefore, supplementary experimentation was carried out in controlled conditions out to show that smaller data payloads can achieve a throughput ratio that is better than the expected ratio (based on maximum throughput), and that multiple concurrent transmissions have a detrimental effect on the expected throughput.

6.1.4 Bit rate

The rate at which bits are physically encoded onto the wireless channel by a radio is called the *bit rate*, and it enforces a physical limit on the maximum speed that data can be received over a wireless link.

Bit rates are typically measured in bits per second, or bits per second (bps). The lower bit rate data is sent at, the longer it will take to be physically received, and thus the greater the latency. For example, 1024 Kilobit (kb) will take 1 second to receive if sent at 1 Megabits per second (Mbps), 0.5 seconds at 2 Mbps, 0.18 seconds at 5.5 Mbps and 0.09 seconds at 11 Mbps.

VoxNet uses wireless radios that comply with the IEEE 802.11b standard. In 802.11b, the bit rate can be dynamically adjusted between 1 Mbps, 2 Mbps, 5.5 Mbps and 11 Mbps at run-time, depending on recent packet loss rates. This is important because it has been shown that lower data rates in 802.11b have a higher probability of reception see (Aguayo et al. 2004), and that transmissions sent at lower data rates can potentially have a longer communication range see (Lundgren et al. 2002). The 802.11b standard requires that unicast transmissions (one sender, one receiver) are acknowledged at the MAC layer, and can be sent at any data rate. There can be a maximum of seven retry attempts of a packet at the MAC layer before its transmission is failed. Broadcast transmissions must be sent at a fixed data rate of 1Mbps, and that there are no MAC layer acknowledgements or retries for these transmissions.

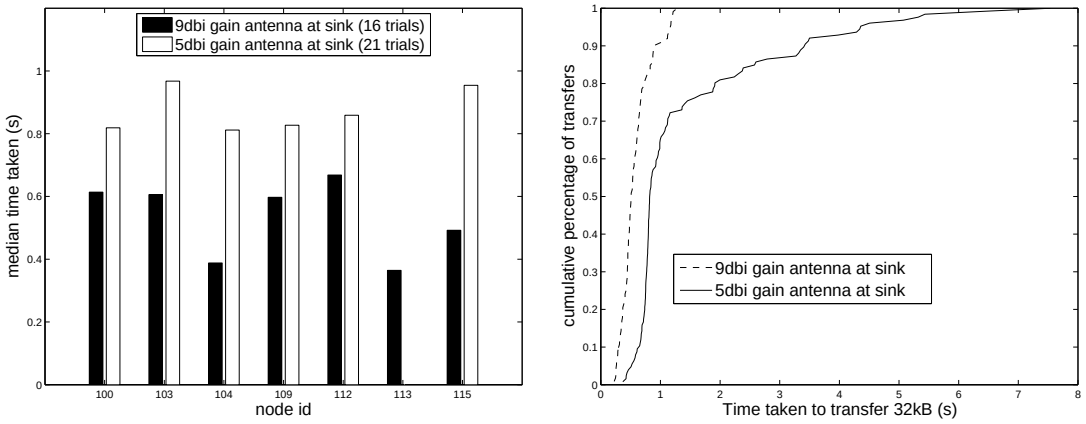
VoxNet nodes use the SMC 802.11b network cards with the Prism 2.5 firmware which dynamically chooses the data rate according to the following algorithm: a packet is originally attempted to be sent at the highest rate (11 Mbps), but if this packet is not successfully transmitted (unsuccessfully acknowledged after 8 retransmissions), the next lowest bit-rate is used (5.5 Mbps). After 10 seconds, and no more loss, the bit-rate can return to 11 Mbps. If the card experiences more loss, the rate may be lowered again. Variable data rates are not present in every wireless standard. For example in 802.15.4 (the standard for low-rate wireless personal area networks), the data rate is fixed at 250 kbps (or 0.24 Mbps).

The higher-level metrics of both goodput and throughput are affected by the bit rate of the wireless radio, and MAC layer retransmissions below that. Higher layers of abstraction are not made aware of the data rates and retransmissions being carried out at the MAC and physical levels, which can cause problems for the end-to-end latency of data transfer. This is especially true for the data transport layer (TCP) which attempts to implement its own flow control based on assumptions of congestion in the network. The combination of TCP and MAC layer retransmission and bit rate selection is therefore a contributing factor in the end-to-end latency of data transfers.

Wireless loss and interference vary depending on physical obstacles in the environment, temperature, humidity and other weather conditions, making it complicated to predict in real environments. Lower data rates may be more stable, but result in a higher transmission latency. If data rates fluctuate between different links, some transmissions may take longer than others. In multi-hop networks, one or more low-quality links on a path will lead to an increase in the overall latency that a node experiences.

6.1.5 Antenna gain

A set of data gathering experiments were carried out in-situ at RMBL to examine the difference in time taken to transfer data when the gain of the omnidirectional antenna at the sink was changed. Theoretically, using antennae with greater gain should improve the quality of the reception and transmission between nodes. The result of this would be a reduction in transmission errors which would affect data rate choice and MAC level retransmissions, as discussed in Section 6.1.4. Effectively, this should manifest itself as a reduction in the time taken to transfer data between nodes and the sink.



(a) Median time to transfer 32 kB, individual nodes (b) CDF of time taken to transfer 32 kB, all nodes

Figure 6.8: The effect of changing antenna size on nodes deployed one hop from the sink. Note that node 113 was absent from the 5 dBi antenna set of experiments. There is a clear increase in median time taken to transfer 32 kB between the two antennae. The CDF also shows a difference in the spread of latency between 5 dBi and 9 dBi experiments.

Two different omnidirectional antennae with 5 decibel isotropic (dBi) and 9 dBi gains were used at the sink (the antennae at the nodes were not changed). The gateway and control console were placed at the car park (position one from Chapter 5, Figure 5.1 on page 125), and all nodes were one-hop from the sink with both the small and large antennae. At the control console, only the WSH was running, and at the nodes, only the WSH client code. The marmot localisation application was not running.

Using the WSH, six (for the lower gain antenna) or seven (for the higher gain antenna) nodes were requested to send 32 kB back to the sink at the same time. This simulated the *worst case* data transfer scenario for the localisation application, where all nodes detect the same event and send detection data at the same time. A total of 16 requests were made for the 9 dBi antenna and 21 for the 5 dBi antenna. The number of nodes changed from seven to six due to a fault in one of the nodes (node 113) that occurred in between the 9 dBi experiment and the 5 dBi experiment. The median times for the data transfers (in seconds), per node, are shown in Figure 6.8, as well as a Cumulative Density Function (CDF) plot of all the transfers for both antennae. The median was plotted due to its robustness to outliers, although both the mean and median were in close agreement for each node in this data set.

In Figure 6.8(a), there is a marked difference in median time to transfer 32 kB between the antennae. The CDF in Figure 6.8(b) confirms this, additionally indicating that around 20% of the transfers using the 5 dBi antenna took longer than 2 seconds, whereas 100% of the transfers using the 9 dBi antenna took less than 1.5 seconds. Using a higher gain antenna is clearly a good way to increase transmission reliability and coverage, although in dense networks this may limit the amount of *spatial re-use* possible. Changing only the sink's antenna reduced both the latency and its variability, which is important for robust operation of the system here. It remains to be seen whether using high gain antenna on all nodes would increase the performance further, or create more potential for inter-node interference. This was

not investigated in the context of the work reported here.

6.1.6 Summary

At the start of Section 6.1, time taken to transfer data over the network was observed to be the largest contributor to end-to-end position estimation latency. It was shown here how this latency of data transfer could be broken down into several inter-related factors, present at different levels of the *stack* in the system. The higher levels in the stack *absorb* the effects of the previous layers, as well as adding in their own contribution to the latency. Complex interactions take place in the network transmission stack: at the application layer, the more data that is generated, the longer it will take to transmit; at the physical and data link layers, the speed at which this data can actually be transmitted is affected by the interference in the network and the number of times messages must be retransmitted. At the data transport and network layers, data must be forwarded reliably, over multiple hops. When multiple nodes are transmitting, loss and congestion can increase transfer times across the whole network, particularly for nodes that are several hops from the sink.

Given that these factors can all combine to increase transfer latency, but are difficult to predict individually, it is clear that when data payloads to be transferred are large, the network must be treated as an expensive resource, to be used only when necessary. Therefore, to reduce end-to-end latency, it is important to understand how processing can best be shared between node and sink in the system, and how the network can best be used when required.

6.2 A case for local processing

As presented in Section 6.1, there are a number of demonstrable factors which affect the latency of data transfers from node to sink. For the marmot localisation application, the interaction of these factors makes the cost of sending large amounts of data highly variable. Section 6.2 makes a case for minimising network traffic by processing locally, and analyses the implications of the ensuing trade-off.

In Chapter 4, it was shown that processing of the AML algorithm (to estimate direction of arrival of the acoustic source) *could* be performed locally, although at a slower rate relative to a typical laptop. The processing takes 2.4112 seconds on average when performed on the node, versus around 0.0833 seconds on an x86 laptop. The result of a detection processed with the AML algorithm is 800 bytes (360 2-byte integers plus some timestamp overhead), which can fit into a single TCP packet (a *segment*) assuming a Maximum Transmission Unit (MTU) of 1500 bytes (this is standard for 802.11 networks). In comparison, a similar 32 kB transfer would involve 21 packets.

Therefore, the conditions under which it is beneficial to process the data locally depend on whether the speedup in network transmission plus the time taken to process locally is less than the time it would take to transmit the raw data. To evaluate this trade-off, a simulation was drawn to compare sending 32 kB of raw data and processing at the sink with processing locally and sending 800 B of processed data. The idea is to add the empirically observed time to send 32 kB over a multi-hop network to the empirically observed time taken to process the AML at the sink, and compare that to the empirically observed time taken to process the AML locally plus the empirically observed time taken to send 800 B to the sink. The AML latencies for local and sink-side processing were gathered in Chapter 4 and 32 kB data

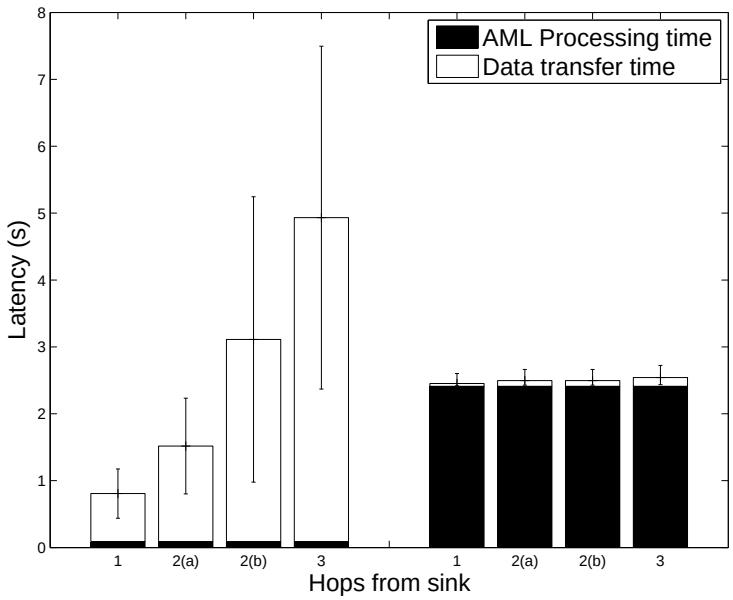


Figure 6.9: Simulation results showing the difference in end-to-end latency between processing at the sink and sending raw data (left hand side), and processing at the node and sending the AML result to sink (right hand side). Each bar contains two parts: the AML processing time (lower part) and the data transfer time (upper part). The x axis shows the number of hops from the sink, where 2(a) and 2(b) are different branches of the routing tree (as shown in Figure 6.5(b) on page 152).

transfer latencies were gathered during the RMBL deployment (Section 6.1.3). However, the latencies for 8 nodes transmitting 800 B simultaneously over multiple hops were not gathered during controlled experimentation at RMBL (when the 32 kB data was gathered).

To address the lack of multi-hop 800 B transfer latency data, it was necessary to model the behaviour of a single TCP packet travelling over multiple hops, based on data gathered over a single hop. It was only necessary to model the behaviour of a single segment in an established TCP stream, because the AML result could easily fit in a single 1500 byte packet.

The data used with the packet latency model was gathered from a controlled one-hop experiment performed at RMBL, where 8 nodes simultaneously sent 800 B to the sink. Over 22 simultaneous data transfer requests (154 transfers total), the mean transfer time was 0.0212s (min 0.008s, max 0.0607s).

The model was formed using an expected packet loss rate per hop. The packet loss probability p was set at 1/50 for each hop that a packet must travel over. The expected loss rate for a packet travelling over N hops is then

$$P(N) = 1 - (1 - p)^N \tag{6.1}$$

assuming the TCP retransmission timer has an exponential back-off starting at 1 second. Using the

formula for a geometric series, the expected latency $E(x)$ is

$$E(x) = N \cdot H + P(N)/(1 - 2 \cdot P(N)) \tag{6.2}$$

where H is the mean latency for transferring 800 bytes over one hop (H has been empirically determined to be approx. equal 0.212).

The model was used to predict the latency of transferring 800 B over 1,2 and 3 hops, and added to the observed time to process an AML locally at the node. This was compared to the latency to transmit 32 kB over 1, 2 and 3 hops, plus the latency to process the AML at the sink. The results of this comparison are shown in Figure 6.9 on the preceding page. Between 2 and 3 hops, there is a definite advantage to local processing: on one side of the routing tree (marked 2(b) on Figure 6.9), transfers take almost 3 seconds on average at 2 hops, and even longer at 3 hops (around 5 seconds on average). The simulated times for local processing and reduced data transmission show AML and sending times of around 2.5 seconds at two and three hops.

Section 6.2 has established that there is a viable case processing data locally when it would take longer to send raw data to the sink. However, identifying *when* this is the case is a complex matter. Previously, it was established that the network is a valuable resource. Only data that is required to be sent over the network should be sent.

6.3 An approach to reducing latency

So far in terms of network latency, it has been shown that data transfer over multiple hops in an ad-hoc network induces latency that increases with the number of hops that a node is away from the sink. However, this latency is compounded when a node has a large amount of data to send, and also when the network is congested or experiencing loss. Section 6.2 showed through simulation that it is best to process locally when it would take longer to send raw data over the network. A specific example was presented, using data gathered from a controlled eight node in-situ experiment. The specific simulation found that nodes should process locally when they were between 2 and 3 hops from the sink.

Different approaches to address network latency in networking have been reported in the literature, depending on its root cause. When some nodes are transmitting more than others, bandwidth equalisation techniques can be used to deliberately and selectively *throttle* certain connections, so that other nodes get a fairer share of the network bandwidth. This type of equalisation is applicable for both wired (802.3) and wireless (802.11) networks, where each node has a direct link with the access point, which arbitrates connections. An example of this in managed wireless networks is the AirEqualizer (Anonymous 2007*a*), which monitors aggregate and per-stream traffic on per second basis, artificially inducing latency on connections which are hogging bandwidth. This can happen in wireless networks when nodes are unfairly represented due to the hidden terminal problem, as noted in Section 6.1.3. Traffic can also be prioritised, depending on the content of the packet headers. The approach is generally termed *traffic shaping*, and is performed by either (1) rate-limiting the data transfer speeds of specific users to provide a given quality of service (QoS) or (2) using packet classification techniques to block or drop traffic.

In multi-hop ad-hoc networks, this issue is rather more complex, as nodes do not always have a

direct link to the sink, meaning a bottleneck or low quality link may be at one or several points along the path. To improve wireless network performance in hard real-time systems, a time division multiple access (TDMA) approach is a way to ensure nodes get an equal share of the network bandwidth. An example of this approach is the WirelessHart standard (Anonymous 2007*d*), which is aimed at process control, where it is important to provide real-time updates of sensor status to make control decisions.

The approach taken in Section 6.3 to latency reduction is application-specific. The characteristics of the marmot localisation application are used to determine what traffic is actually *necessary*, rather than concentrating on general rate limiting or particular stream limiting. From an application perspective, the author argues that there are two principles that should be met when accounting for the relative expense of transmitting data over multiple hops and attempting to reduce end-to-end latency:

1. **Only send data when it is useful for the network's overall aim**

2. **Only process locally when it is advantageous to do so**

The first principle requires that only useful data is sent over the wireless channel, ideally reducing transfer latency. This implies that it must be decided exactly what useful data is. The second principle requires that the cost of transferring raw data should be traded off against the cost of local processing (but makes no requirement on whether this data is useful or not) Section 6.4 proposes two algorithms of differing complexity to help decide *when* a node should process data locally to meet the second principle. Section 6.3 proposes and justifies through simulation an algorithm to reduce the amount of data sent over the network by determining which data is actually useful to meet the first principle.

An important aspect of meeting the goal of only sending data that supports the application's overall aim is to provide some way of deciding if data is *useful* before sending it. In the context of the motivating marmot localisation application, data sent by the nodes is only *useful* if it is being used as part of a position estimate calculation. The author proposes an algorithm called *Lazy Grouping* to decide whether data is useful and hence whether it should be sent or not. The rest of Section 6.3 describes Lazy Grouping in detail and justifies its approach through simulation.

6.3.1 Lazy Grouping

Lazy Grouping is a centralised algorithm that builds on and modifies the original on-line grouping algorithm presented in Chapter 5. Where the basic on-line grouping algorithm attempts to group all event data that it receives (regardless of whether it could be used in the localisation algorithm or not), Lazy Grouping takes a more proactive approach in data collection by only requesting event data when the corresponding detection has been confirmed to be part of a group. For the marmot localisation application, this means that if at least three different nodes report detections which are close enough in time, then the raw data should be requested by the sink-side application for AML processing and input into the localisation algorithm. The Lazy Grouping approach is inspired by Bentley's *lazy evaluation* approach (Bentley 1986):

> Lazy evaluation: the strategy of never evaluating an item until it is needed avoids evaluations of unnecessary items.

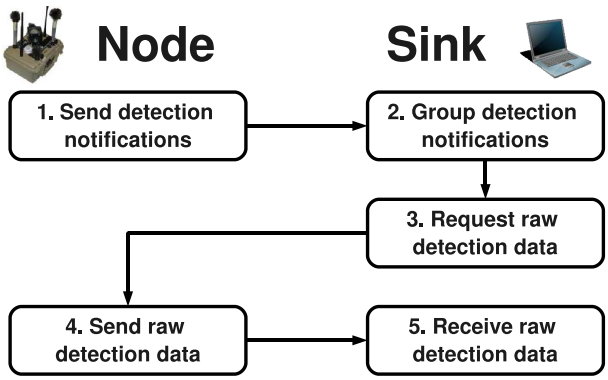


Figure 6.10: The flow of Lazy Grouping's on-ling grouping and data collection stages.

Lazy Grouping breaks into two distinct parts: on-line grouping and data collection. Lazy Grouping's on-line grouping is largely based on the on-line grouping approach in Section 5.2.2 on page 134, but it requires that nodes send *detection notifications* when they make detections, instead of raw data. Detection notifications are similar to the original detection tuples d_i (in Section 5.2.2 on page 134, but they do not contain the raw data, d_{raw}, only the detection timestamp d_{det}. Therefore, they are smaller (tens of bytes instead of tens of kilobytes) and thus faster to transfer. To support this, nodes maintain a local buffer B with all of the i current raw detections $B = \{d_{1,raw}..d_{i,raw}\}$ (see Section 6.3.3 for issues relating to buffer overflow and data loss).

Lazy Grouping's data collection is initiated when the sink has a group of at least three detection notifications: it sends out requests to all the nodes in a given group. The nodes locate the relevant raw detection data in their local buffers and send it to the sink. When the sink has received all of the data in a given group, it triggers the next stage of the processing for the group (AML and data fusion). The event flow for Lazy Grouping is shown in Figure 6.10, and described in greater detail below.

1. **Evaluate identity consistency (Sink).** This stage is *exactly* the same as the equivalent stage in the on-line grouping algorithm presented in Section 5.2.2 on page 134.

2. **Evaluate temporal consistency (Sink).** This stage is *exactly* the same as the equivalent stage in the on-line grouping algorithm presented in Section 5.2.2 on page 134.

3. **Place detection in correct group (Sink).** This stage is *exactly* the same as the equivalent stage in the on-line grouping algorithm presented in Section 5.2.2 on page 134.

4. **Check group and request data (Sink: timeout-based).** When a set amount of time l second has elapsed, the watchdog timer set when the group was created w_t triggers. At this point the group g associated with w_t is checked to see if it can be processed further. If the number of detections in the group is three or more ($n(g) > 2$) then the group is flagged for collection. At this point, the sink sends each node $I(d)$ in the group g a *data request message*, containing the sequence number s

the node assigned to the detection, as well as the group number $I(g)$ that the sink assigned to the group when it was started.

5. **Locate data and send to sink (Node).** Node $I(g)$ that receives a *data request message* finds the raw detection in its the local buffer (given by $B_s^{I(g)}$) and sends a reply *tuple* to the sink $R = \langle d_{raw}, t \rangle$.

6. **Place data in correct group and trigger next processing stage (Sink).** When the raw data d_{raw} arrives at the sink, it can then be referenced to the group g_t. The raw data can then be placed into the node's detection tuple. When all the raw data has been gathered for g_t, the sink can trigger the next stages of processing—the AML and data fusion algorithms.

The Lazy Grouping approach to filtering of detections is necessary because the event detectors that the nodes run to detect marmot calls can be susceptible to false positives (for example, caused by weather events, as discussed in Chapter 5). If a node's event detector is triggering due to many false detections, this will result in raw data of many false detections sent unnecessarily back to the sink, creating unnecessary traffic. In this case, traffic equalisation approaches would attempt to lower the node's sending rate so that other nodes can get a fair share of the available bandwidth, which does not solve the problem that the data being generated is of no use.

The application-informed approach taken to reducing latency by reducing traffic is more suitable for the localisation application. The filtering should make it more difficult for a node to *steal* bandwidth from other nodes as its detections are only useful when correlated with others.

6.3.2 Algorithm validation

In order to validate the approach used in Lazy Grouping algorithm, two simulations were carried out using data gathered in-situ at RBML. The first simulation used an off-line implementation of the Lazy Grouping algorithm on the in-situ data trace to determine to what extent it could reduce unneeded data transmissions. The second simulation used the same in-situ data trace in conjunction with a set of data transfers gathered from a controlled experiment from the same network configuration (also at RMBL) to observe the relative reduction in latency in gathering data corresponding to groups. It should be noted that all of the data gathered and used in these simulations was from a one-hop network, rather than a multi-hop network. The experiments are discussed below.

Experiment one: data reduction

The goal of the first experiment was to quantify the reduction in data transfer that could be possible using Lazy Grouping. It was expected that Lazy Grouping would provide significant benefit when applied to the RMBL data set. The experiment was performed using simulation. To achieve this, an off-line version of the Lazy Grouping algorithm was run over a real event stream gathered at RMBL. The off-line version of Lazy Grouping only simulated the creation of groups, not the requesting and transferring of data (this was simulated in the second experiment). In the RMBL data set seven nodes all one hop from the sink sent back any events that were triggered by their event detectors (regardless of whether they were actually marmot calls or not). For each detection that arrived at the control console, the sending node ID, global time of detection and time taken to transfer the data were recorded. The amount of data sent

per detection was 128 kB, rather than 32 kB. This was due to a misconfiguration of the system which was not possible to rectify during the deployment. This disparity in data size does not affect the analysis here as it is only the timing of the detection events that are of interest, not their size. The amount of detection data transferred during the actual in-situ experimentation was 44.125 MB (353 events). When the off-line Lazy Grouping algorithm was run over the data set (using a *fuzz factor* of 440 ms), and a minimum grouping of 3 detections to make a position estimate), the amount of data that would have been sent was reduced to 15.75 MB (126 detections). This represents a 64% reduction in data transferred—a clear validation of the data reduction capabilities of the Lazy Grouping algorithm. In all, 24 groups were made: 14 groups with 3 detections each, 16 groups with 4 detections each and 4 groups with 5 detections each. No groups larger than 5 were found.

Figure 6.11(a) on the next page shows a magnified window of the first 18 minutes of the RMBL data trace, with time since the first detection on the x axis, and node id on the y axis. Detection events are marked as dots and events which have been grouped together are marked with squares. To aid understanding, the mean timestamp of each group is shown as a vertical line. This shows the filtering effect of the Lazy Grouping approach and gives an indication of the number of detections that can be disregarded based on the temporal and identity consistency rules. Figure 6.11(b) shows a particular group of detections from Figure 6.11(a), highlighting how the temporal consistency rule rejects detections which are not within the *uncertainty factor* from the group's mean.

Experiment two: latency reduction

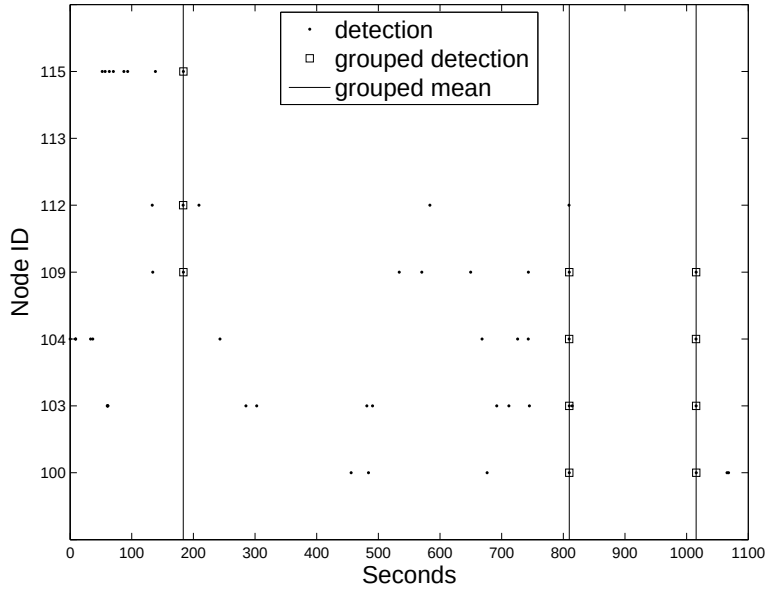
The goal of the first experiment was to demonstrate the potential reduction in the amount of data sent using Lazy Grouping. The goal of the second experiment was to simulate the data collection phase of Lazy Grouping. This demonstrates the potential reduction in latency that is possible when only the data classed as useful by the Lazy Grouping algorithm is gathered. The Lazy Grouping data collection phase consists of the following steps: (1) receive detection notifications from nodes, (2) send out requests for data and (3) receive raw data responses from nodes. More formally, assume that a group g_i containing $n(g_i)$ detections is created by the Lazy Grouping algorithm. Assume $\ell_g = \{g_i\}_{i=1}^{n(g_i)}$ is the set of latencies for collecting data from all nodes in a group g, where each element of ℓ_g is determined by

$$\ell_g = q_i \cdot d_{raw,i} + d_{det,i} + d_{req,i}) \tag{6.3}$$

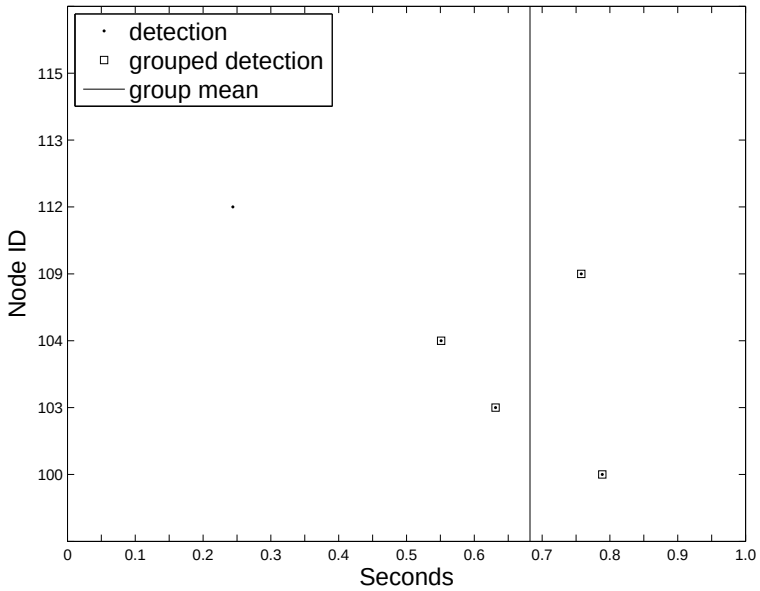
where q is the number of detections the ith node had waiting to be sent in its local queue before the current request, d_{det} is the time taken for the ith node to send a detection notification and d_{req} is the time taken for the sink to send a data request to the ith node. The overall latency $l(g_i)$ to gather the data from the group g_i is therefore

$$l(g_i) = \max_g \ell_g \tag{6.4}$$

The simulation of Lazy Grouping's data collection process as discussed above was implemented in Matlab. The simulation iterated through the list of detections belonging to the groups created by the Lazy Grouping algorithm in experiment one (Section 6.3.2). The list was ordered by detection time. For each detection, q_j was inferred by counting the number of detections node j had made as of the current



(a) A plot showing the result of grouping detections to decide which data should be sent to the sink by nodes. Each point on the plot is a detection, made by a node (shown on the y axis) at a certain time (shown on the x axis). Vertical lines show the mean of the group to which detections marked as squares belong to. All other detections are discarded.



(b) Rejection of detections that are not close enough to the mean of the event group. In this case, the detection from node 112 was more than 440ms away from the mean established by the detection timestamps from nodes 109, 104, 103 and 100.

Figure 6.11: The grouping of estimates based on temporal and identity consistency.

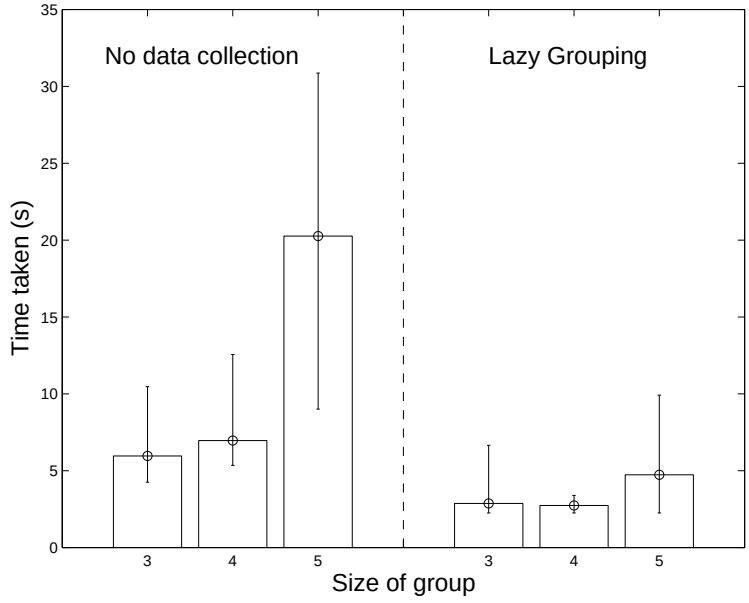


Figure 6.12: A graph showing the potential benefit of Lazy Grouping's data collection process in simulation. For comparison, the time taken for the data to arrive at the sink for the original data trace is shown. The error bars represent the min and max times for each transfer, and are ordered by group size.

position in the list and subtracting the number of detections the sink had received (this could be inferred from the reception time logged for detection).

The other parameters to calculate l_i (d_{raw}, d_{det} and d_{req}) were determined from empirically gathered data. Eight 128 kB and twenty-two 800 B simultaneous seven-node data transfers were requested during controlled experimentation in a one-hop network at RMBL. For each 128 kB or 800 B data request, the start of the first transmission and the arrival of the last transmission were recorded, giving the total time to gather 128 kB or 800 B from all seven nodes. The mean of the twenty-two 800 B transfer times were used to model both d_{det} and d_{req}.

For comparison, the time taken to gather the groups in the in-situ data trace were calculated, simulating what would happen if the on-line grouping algorithm was grouping data as it arrived at the sink, rather than requesting it. To do this, the earliest detection time and latest arrival time of the detections in each group were found. The results of the Lazy Grouping simulation and the latency from the normal event stream are shown in Figure 6.12. The bars represent mean time taken to collect the data for the group, with error bars representing the min and max times.

There is a clear improvement in the time taken to gather a group of data using Lazy Grouping. The best improvement in data collection time is seen when collecting data for groups of five detections: the average time goes from twenty seconds to five seconds, a reduction in latency of 75%. In general, the trend indicated by both sides of Figure 6.12 is that larger clusters take longer to gather. This is particularly notable in the non-Lazy Grouping simulation. However, the latency increase is most likely

partially caused by a high frequency of detections being triggered and transferred during periods when larger groups are likely to be formed. This is exactly the kind of situation that Lazy Grouping can address, and it does so by ensuring that only useful data is sent.

6.3.3 Discussion

There is an implicit assumption in temporal grouping that time is the only important factor in deciding whether a set of detected events constitute an actual marmot call (or other animal). However, it was observed experimentally that other events such as rainfall and wind could cause false positives from the event detector. A correlated false positive (where several nodes' event detectors are triggered by a non-marmot sound) could give the impression of a real event, even though the actual sound that caused it was not the source of interest. A partial solution to this problem was discussed in Chapter 5, where classification techniques could be used on-node to differentiate between actual marmot calls and false positives. However, this may require increased processing power or operate with an additional delay, especially when dealing with several detections triggered closely in time. By comparison, the temporal grouping algorithm is relatively lightweight, with minimal processing. The amount of data required to be sent to determine network wide events is also minimal.

Section 6.3.1 discussed Lazy Grouping, an algorithm to reduce latency and network traffic based on gathering data from nodes after their detection notifications have been grouped based on temporal correlation. The Lazy Grouping protocol adds an extra round of communication between sink and nodes, first to send detections, and then request data, followed by the actual data transfer. In terms of network traffic, the Lazy Grouping protocol requires that nodes send detection notifications (1 packet) instead of raw data upon receiving detections. It also requires that the sink sends out a small (1 packet) data request to every node that is part of the group for a given event. Nodes must also store the detected events, rather than just sending them as they are detected. Each VoxNet node maintains a 10 second ring buffer of data (see Chapter 5) from which the data can be queried and sent. It is likely that this would suffice for normal operation, as detection events are likely to be requested soon after the detection message has been sent to the sink (that is, within 1 or 2 seconds). The Lazy Grouping algorithm was validated by two experimental simulations, one which showed how the amount of data needing to be sent could be reduced, and the other showing how the time taken to gather data corresponding to a group could be reduced. In a realistic deployment, it is expected that Lazy Grouping would significantly reduce the amount of data transmitted over the network, and a marked difference in data transfer times would be seen. Of course, latency will still be seen from data transfers during periods of wireless loss.

6.4 Adaptation policies

In Section 6.3 two principles were proposed to reduce end-to-end latency in the acoustic localisation system. The first principle, *to only send data when it is useful for the network's overall aim*, was addressed by Lazy Grouping 6.3.1 on page 160. Section 6.4 addresses the second principle: to *process locally when it is advantageous to do so*. It has already been shown in Section 6.2 that local processing is a viable alternative to sending data over the network in some cases, but the difficulty is in predicting *when* a node should process locally. Two policies to predict when local processing is advantageous have

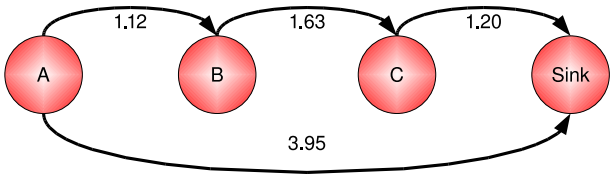


Figure 6.13: An example of how per-link ETX is aggregated to make path ETX. In this case, the path ETX is greater than 2.5, so the Static Thresholding algorithm dictates the node should process the AML algorithm locally.

been developed and implemented by the author: *static thresholding* and *Dynamic Estimation*, which are described in the rest of Section 6.4.

6.4.1 Static Thresholding

The basis of the Static Thresholding policy is to use an empirically determined threshold (Section 6.2) to aid a node in deciding when to process locally. It was previously observed that nodes between 2 and 3 three hops away from the sink would have been best suited to processing data locally. However, a node's hop count from the sink is an integer value, thus cannot represent a value between 2 and 3 hops. Instead, this intermediate 2 to 3 hops value can be roughly equated to the Expected number of transmissions to transmit one packet (ETX) (Couto et al. 2003). Using ETX in this way is reasonable because it gives an indication of the number of packets that must be sent over a link in order for one packet to be successfully received. The *path ETX* value is the sum of the ETX for each link in the path. An example is shown in Figure 6.13. This is the same idea as the example given in Section 6.1.3, where if a node is three hops away from the sink, it will take *at least* three transmitted packets for the packet to arrive.

Recalling that the original experimental observation was that a node should process locally if it was between two and three hops from the sink, this could be equated to a *minimum* ETX value of 2.5. VoxNet's implementation of the DSR routing algorithm (described in Chapter 4) uses ETX to decide the best routes in the network, thus the ETX value is available to nodes locally. The general Static Thresholding policy that is evaluated by a node for each detection triggered is

if $c(E) > \tau$, then process locally, Else send raw data to the sink.

where $c(E)$ gives the current ETX value at a node, and τ is the ETX value representing the decision threshold. The experimentation performed in Section 6.2 on page 157 indicated τ should be set to 2.5 (this value for τ is used in further evaluation in Section 7.1 on page 172.

Static Thresholding is a naïve approach to adaptation, in that does not take into account local queues on the node for either outgoing data, or local AML processing queues. However, this approach does have the advantage of being cheap in terms of required processing, as each node only requires knowledge of their current path ETX, given by $c(E)$. The value of 2.5 for τ is based on one set of experimental data transfers in Section 7.1 on page 172 so is not necessarily representative of larger networks, different routing topologies or networks that change routes over time. Therefore, it is expected that the threshold will not be a decision indicator for local processing in general. A more generally suitable approach would be to

dynamically estimate whether to process locally or not, based on parameters that can be measured about the state of the network as it is running. A policy called Dynamic Estimation, which uses a dynamic estimator based on recent network transfers is presented by the author in the remainder of Section 6.4.

6.4.2 Dynamic Estimation

Dynamic Estimation uses data that is available from the running system to make the choice to process locally, based on the evaluation of a dynamic estimator. The dynamic estimator requires that a node records the latency of all raw data transfers it makes: the latency of these previous transfers is used to estimate how long the next transfer will take. The policy is as follows:

> if $t_{aml} \cdot n(q_{aml}) < \hat{\ell}(d)$ then the node should process locally

where t_{aml} is the time to taken process an AML locally on a node, $n(q_{aml})$ is the number of raw detections waiting to be processed locally and $\hat{\ell}(d)$ is the estimated time that the transfer current detection will take to send over the network $\ell(d)$, provided by a dynamic estimator function (discussed next). The reason why only raw data transfers are used because they are the same size: using the information/log messages may give an incorrect estimation of goodput because they are so small, and do not suffer the effects of TCP and wireless loss to the same extent (see Section 6.1.2 on page 149).

Two different implementations of the dynamic estimator function to estimate $\ell(d)$ are presented in Section 6.4.2: $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$. Both of these implementations use the time taken for previous transfers to predict the next transfer. $\hat{\ell}_1(d)$ uses a goodput measurement, which is the measure of useful data received divided by the time taken to transmit. $\hat{\ell}_2(d)$ uses a *pseudo-goodput* measurement, which measures the time elapsed from the detection being placed on the outgoing queue until it was received at the sink. Both estimators for $\ell(d)$ determine their predictions as the amount of data that must be sent on the path from node to sink, divided by the goodput (or pseudo-goodput) estimate, which is calculated from previous transfers. The estimate of $\ell(d)$ using $\hat{\ell}_1(d)$ is

$$\hat{\ell}_1(d) = n(q_{send}) \cdot D/g_k \tag{6.5}$$

where $n(q_{send})$ is the number of detections queued to send (including the current detection), D is the amount of data to be sent, and g_k is the mean goodput of the last k transfers the node has made. The amount of data to be sent D is given by

$$D = h \cdot s(d_{raw}) \tag{6.6}$$

where h is the number of hops the node is from the sink and $s(d_{raw})$ is the size (in kB) of a raw detection, so that the forwarding of data over multiple hops is considered. The mean goodput g_k is calculated from the last k transfers on a *per-node* basis as

$$g_k = \frac{1}{k} \sum_{i=x-k}^{n} \frac{D_i}{t_i} \tag{6.7}$$

where x is the number of transfers sent so far, t_i is the time taken for the ith transfer and D_i is the total amount of data that was sent over the sent over the path between node and sink for a given transfer, as in Equation 6.6. This allows the node to use an arbitrary number of previous transfers to help in predicting the time that the next transfer will take.

The pseudo-goodput estimate $\hat{\ell}_2(d)$ for $\ell(d)$ is

$$\hat{\ell}_2(d) = D/p_k \tag{6.8}$$

where p_k is the mean pseudo-goodput derived from the past k raw data transfers. The pseudo-goodput differs from the mean goodput g in that it includes the time spent on the message queue q_i. The mean pseudo-goodput p_k is

$$p_k = \frac{1}{k} \sum_{i=x-k}^{n} \frac{D_i}{q_i + t_i} \tag{6.9}$$

where q_i is the time each detection spends on the message queue. The pseudo-goodput estimate $\hat{\ell}_2(d)$ may be easier for the higher level system to determine, because $q_i + t_i$ is determined by timestamping when the data was queued to send and when the transfer was completed, rather than when the data transmission actually began.

To allow the node to determine transfer times and evaluate the Adaptation policies locally, the sink provides the global time at which it received the node's transfer as part of the acknowledgement of message reception. Static and Dynamic Estimation are evaluated in Section 7.1 on page 172, where the in-situ and controlled experimentation performed at RMBL is augmented by controlled experiments using the same VoxNet nodes at the University of California, Los Angeles.

6.5 Summary

This chapter identified the main sources of end-to-end latency in the marmot localisation system. These sources were observed during an in-situ deployment of the system, as well as controlled experimentation, in-situ and in a laboratory environment. These sources of latency were shown to be either node-based, network-based or sink-based. In-situ observation of the system running established that network latency was by far the biggest contributor to overall latency in the system.

Transmission of data over the network was identified as an expensive resource, which needed to be used when it was necessary, rather than whenever possible. Based on this observation, the author proposed two axioms to be met for reduction of end-to-end latency of position estimation—*to only send data when it is useful for the networks overall aim* and *to process locally when it is advantageous to do so.*

To meet these axioms, two refinements and associated algorithms were presented: *Lazy Grouping* and *Adaptation.* Lazy Grouping was used to stop unnecessary raw data from being sent over the network and taking up valuable network bandwidth. This was achieved by requiring nodes to send small detection notifications to the sink (rather than full raw detection data) which were grouped. Data was requested only from nodes whose detections were part of a group. Simulation was used to evaluate both the grouping and data collection aspects of Lazy Grouping, using a real-life data trace. A 64% reduction in data transferred was observed, as well as a reduction in mean data collection time of 75% (five seconds

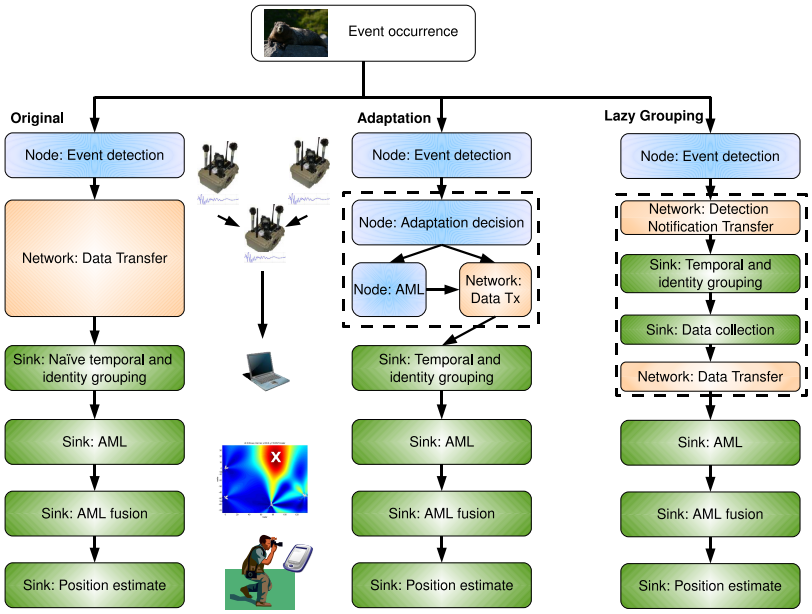


Figure 6.14: The flow of events in the original system (left), with adaptation (middle) and with Lazy Grouping (right) refinements.

versus twenty seconds on average for a group of five detections).

Two approaches to adaptation, *static* and *Dynamic Estimation* were presented. These are further evaluated in Section 7.1 on page 172. Static Thresholding used an empirically derived, static measure of when local processing was advantageous, and Dynamic Estimation used recently completed transfers of raw data to predict how long the next transfer would take (and hence whether to process locally or not).

Figure 6.14 shows how the Lazy Grouping and Adaptation components could be embedded into the original system flow (shown on the left of figure 6.14). The Lazy Grouping approach to data filtering and collection (the far right column in Figure 6.14) conceptually replaces both the data transfer and grouping components seen in the original system data flow. Whereas the flow of the original was one-way (nodes transmit raw data to the sink), Lazy Grouping requires a more involved, two-way interaction: nodes transfer detection notifications to the sink, which groups them. Subsequently, the sink sends data collection requests to the node, which reply with the relevant raw data.

The data flow for Adaptation is still conceptually one-way, however it can be either raw data or processed AML results which are sent by the node to the sink. Although the integration of these two components is not discussed here, it is considered in Chapter 7, where the implications of integration are considered, and a new data flow formed.

In Chapter 7, Adaptation is evaluated through the Static and Dynamic Thresholding algorithms (including both $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ latency estimators for Dynamic Estimation).

Chapter 7

Adaptation policy evaluation and general results

Chapter 6 presented two refinements, Lazy Grouping and Adaptation, for reducing the latency of data transfers in VoxNet based on application-specific information. Adaptation is the choice a node makes (when it triggers an event detection) as to whether it should process the raw data corresponding to a detection through the AML algorithm at the node, or send the raw detection data to the sink for AML processing.

Two on-node policies for Adaptation were presented in Section 6.4 on page 166, where the choice of where to carry out the data processing is based on an empirically derived static threshold or a dynamically calculated latency estimate. This chapter presents the evaluation of these two Adaptation algorithms. The evaluation was performed: (1) in simulation using data traces gathered from the VoxNet's deployment described in Chapter 5, (2) through controlled experiments performed in-situ and around the campus of the University of California, Los Angeles (UCLA). Controlled, in-situ experimentation was necessary to provide data traces that had environmental effects which are complex to simulate using theoretical models. Simulation using these controlled data traces was sufficient to provide proof-of-concept validation.

Dynamic Estimation was found to improve the correctness of Adaptation decisions by up to 30% over Static Thresholding. Dynamic Estimation has a further advantage over Static Thresholding in that the estimators it uses predict the latency of data transfers. The accuracy of latency estimates produced by the Dynamic Estimation policy's latency estimators is adequate for the application at hand. However, for general suitability, the estimators require further improvement.

The implications of integrating Lazy Grouping and Adaptation into the VoxNet system were considered. Several issues were raised regarding changes that would be required to Lazy Grouping, Adaptation to allow co-existence, as well as the integration-related modifications that would be needed at application and network level in VoxNet. The resulting data flow for marmot localisation is compared to other high data-rate systems, and in particular to the Lance framework for data collection. VoxNet shares some important characteristics with Lance, but also provides application specific dynamic processing (Adaptation) which is not suitable for Lance's generic approach.

The work presented in this chapter is a combination of practical experimentation, simulation and emulation aimed at evaluating Adaptation in terms of its effect on latency of data transmissions as part of on-line source localisation. The use of different experimental approaches enabled a thorough and complete understanding of the core issue in this chapter, namely improving the timeliness of the

end-to-end acoustic localisation system. The rest of this chapter is organised as follows: Section 7.1 describes the experimental work and simulation set-up used to evaluate the Adaptation policies, along with analysis of the results. Section 7.2 discusses the integration of both Lazy Grouping and Adaptation into the end-to-end acoustic localisation system. Sections 7.3 and 7.4 consider the generic elements of the system with respect to other localisation applications and high data-rate systems, and finally Section 7.5 summarises the contributions of this chapter.

7.1 Evaluation of Adaptation policies

To evaluate the performance of the two Adaptation policies presented in Section 6.4 (Static Thresholding and Dynamic Estimation), simulation was carried out using Matlab based on data traces gathered from in-situ experimentation. Section 7.1 describes the approach used to gather the data traces, how they were used in simulation, and the analysis of the results obtained.

The evaluation of the Adaptation policies through simulation had two goals. The first evaluation goal was to compare the performance of Static Thresholding and Dynamic Estimation policies on the basis of the correctness of their Adaptation choices. This is the only way the two policies could be compared side-by-side, since the Static Thresholding policy was based on observations from a fixed local processing time (for the AML) and a specific network topology.

The second evaluation goal was to determine the accuracy of the latency estimators $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ used by the Dynamic Estimation policy (Accuracy refers to how closely the predicted and actual data transfer latencies were). Accurate prediction of transfer latency allows the Dynamic Estimation policy to be generally suitable in making local processing decisions where the local processing operation takes an arbitrary amount of time. In addition to the latency prediction, the factors which could affect the accuracy of the transfer latency predictions were also evaluated: the amount of previous transfers to use in latency estimation, and the amount of time a local processing operation took.

This rest of Section 7.1 is divided into: the gathering of empirical data (Section 7.1.1), an outline of the simulation process (Section 7.1.2) and analysis of the simulation results. The analysis of results is organised into: comparing the performance of Static Thresholding and Dynamic Estimation (Section 7.1.3), the accuracy of the latency estimators (Section 7.1.4), the effects of varying previous data transfers on latency estimates (Section 7.1.5), and the effects of varying local processing operation time of latency estimates (Section 7.1.6).

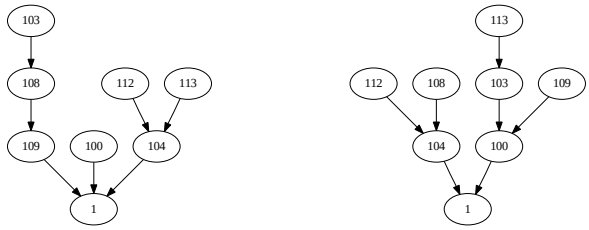
7.1.1 Gathering of empirical data

In total, three data traces were gathered of which one came from live system operation, with real event detectors running (taken at Rocky Mountain Biological Laboratory in 2007). The other traces were taken from controlled outdoor and indoor experiments, where requests were sent to nodes to obtain data back, emulating the effect of acoustic events occurring in the network.

All experiments used seven nodes, and a gateway with laptop attached to act as the sink. For each experiment, the network topology was known, and consistent for the experimental period (shown in Figure 7.1 on the facing page). Table 7.1 describes the data sets gathered. First, it should be noted that the RMBL data set was different to the controlled data sets in that the size of the detections was

Table 7.1: A description of the data sets gathered, including location, network topology and data source.

Experiment location	Topology	Data source	Per-detection size
RMBL, CO, 2007	single-hop	in-situ events	128 kB
Royce Hall, UCLA, 2008	*dynamic* multi-hop	request plan	32 kB
CENS Lab, UCLA, 2008	*fixed* multi-hop	request plan	32 kB



(a) Royce Hall, UCLA (b) CENS, UCLA

Figure 7.1: The multi-hop routing trees used in experimentation. Note that the RMBL, CO experimentation was single-hop, thus is not shown here.

larger (128 kB versus 32 kB) due to misconfiguration. Therefore, a direct comparison with a data set where 32 kB detections were gathered may be misleading; the transfer latency was greater, as would be expected. However, the analysis of this data set was included to allow for discussion on the general suitability of Dynamic Estimation's latency estimators $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$.

Second, the multi-hop topology used in the CENS Laboratory was created by using MAC address filtering at each node to enforce a particular multi-hop topology (that is the DSR component could *only* find the particular multi-hop topology enforced).

Experiments at Royce Hall and the CENS Lab made use of the WSH mechanism which allowed data requests to be sent to some or all of the nodes in the network, mimicking actual event occurrences (this functionality was described in greater detail in Chapter 4). Modifications were made to allow the sending of data requests to be automated, using a script denoted here as a *request plan*. This allowed for timings to be inserted in between each data request (that is: send a request, then wait x seconds, then send another), enabling repeatable request plans across different experiments. An example of a test plan is shown in Figure 7.2.

Two *request plans* were used. In one, data requests took place at 1 second intervals, and in the other, data requests took place at 5 second intervals. Both plans were logically divided into seven *phases*, each consisting of 21 data requests. In each phase, an increasing number of nodes were requested to simultaneously send data, starting with one node and ending with all seven. In each phase, each node in the network was requested to send data an equal number of times, so that no particular node could dominate the data set. A pause was inserted in between phases so that any delayed transfers could complete before the next phase started. For all experiments (RMBL, Royce and Lab), the following data

```
1000 send test nodes=112,113 size=32768
5000 send test nodes=100,103,104,108,109,112,113 size=32768
40000 send test nodes=100,103,104,108,109,112 size=32768
```

Figure 7.2: A few example lines from a request plan. The first field is the timeout (in milliseconds) before the command is issued. `send test` tells the WSH to send the command over the network, `nodes` describes which nodes should replay to the request, and `size` tells the node the amount of data to send back.

were recorded for each raw data transfer between node and sink that took place, representing the data trace:

- Node ID
- Event detection timestamp
- Detection arrival timestamp
- Timestamp when message was sent to outgoing queue
- Time taken to transmit data
- Detection data size
- ETX value when the detection data was queued to send
- Number of hops node was from sink
- Number of detection events in outgoing message queue
- Aggregate number of detection events in the network

These traces provided enough data to enable simulations of the Adaptation algorithms to be run off-line.

7.1.2 Simulation

The Static Thresholding algorithm was simulated by iterating through the list of data transfers, examining the ETX value and recording the outcome of the policy evaluation (to process locally or not). To simulate the Dynamic Estimation algorithm, the list of data transfers was copied into two lists: one sorted all the transfer records by event detection timestamp (to simulate the order in which they were detected), and the other sorted the transfer records by detection arrival timestamp (to simulate the order in which the transfers were received at the sink). This meant that for any event detection record, it was possible to determine all of the transfers that had occurred (for that particular node). Therefore, the k most recent transfer latencies could be computed. For the simulation $k = 3$ was used, as it was found to be the best trade-off between historical transfer latency and latency prediction accuracy across all data traces (this will be further discussed in Section 7.1.5). As with the Static Thresholding simulation, the detection list was iterated through, and for each data transfer record, both latency estimators $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ were used to predict the transfer latency. Since the goal of the experimentation was to evaluate the accuracy

of the latency estimators, it was not essential to include Lazy Grouping in the system or the concept of a local processing queue in the policy evaluation. The local processing queue is important to consider when addressing the integration of the Lazy Grouping protocol as a whole, which is discussed in Section 7.2.

It could also be argued that the decision to process locally means that the data transfer corresponding to that detection would not be sent, and thus not used for historical data. In the simulation all transfers are included. Selectively removing some transfers is difficult to simulate with an off-line data set, and is more conducive to a real system test (as discussed in Section 7.2.3 on page 185). However, the factors being considered by the experimentation are whether the latency estimators could predict correctly when to adapt, and whether the estimation was accurate given the data supplied.

The rest of Section 7.1 analyses the simulation results, starting with the relative performance of the Adaptation policies.

7.1.3 Performance of Adaptation policies

For each detection event that is triggered at a node, the goal of Adaptation is to correctly decide whether it will take longer to process raw detection data locally or to send it to the sink. Therefore, the best way to evaluate the Adaptation policies' relative performance is to compare the processing choices made by the algorithm with the choices that ensure minimum system latency (referred to as the *correct choices*). The Static Thresholding policy decided to process locally if the ETX at the time of event detection was exceeded the ETX value of the threshold τ (Section 6.4.1 on page 167). Based on experimentation in Section 6.2 on page 157 the ETX value of τ was 2.5 (which can be considered to equate to *between two and three hops*). The Dynamic Estimation policy chose to process locally if the estimated time taken to transfer $\hat{\ell}(d)$ was greater than the time taken to process the raw detection through the AML algorithm locally t_{aml} (assuming no AML processing queues). In Section 4.5.4, t_{aml} was determined to be 2.2s. Every choice made by a policy was classed as *correct* if it met one of the following criteria (otherwise it was classed as incorrect);

- The decision is to adapt and the actual transfer time was greater than t_{aml} seconds (correct because sending would have taken longer).
- The decision is *not* to adapt, and the actual transfer time was less than t_{aml} seconds (correct because sending would have taken less time).

The choices for all of the transfers were recorded according to these criteria, and compared to what would have been the correct choices (determined from the actual transfer latencies). It was expected that the Dynamic Estimation policy would make a higher percentage of correct decisions than the Static Thresholding policy because of it could potentially adapt to changes in network latency over time. Table 7.2 on the following page shows the percentage of correct choices made by both algorithms. It should be noted at this point that the ETX values (at time of sending) were not gathered in the RMBL experimentation, hence no results for Static Thresholding are reported. However, it is hypothesised that ETX would not have been a good indicator of latency because the data size sent was different (128 kB vs 32 kB) and the network topology was different (all nodes were one hop away from the sink). Link

Table 7.2: The relative performance comparison between an ETX based decision point for Adaptation and the $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ latency estimators. As previously discussed, the estimates were made using the mean of the last three received transfers for each node.

Experiment	ETX (%)	$\hat{\ell}_1(d)$ (%)	$\hat{\ell}_2(d)$ (%)
RMBL	n/a	74.79	70.54
Royce	64.9	94.07	96.3
Lab	86.05	97.87	97.02

quality would have to have been low for ETX to exceed 2.5 (Static Thresholding's Adaptation decision threshold) for any of the nodes.

Both of $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ performed better than the ETX value used by Static Thresholding. In the case of the Royce data set, the improvement is in excess of 30%. The Lab data set shows $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ performing approximately 11% better than the Static Thresholding. The ETX value used to compare against τ may not be suitable when there are items in a node's local queue. This is because the ETX represents the state of the network when the data was queued, not when it was actually sent. It is certainly possible that the ETX would have changed when the node actually transmitted the data. To test if queued ETX had an effect of the percentage of correct choices, only transfers initiated when a node had no queued detections were evaluated using the ETX value. This resulted in a change of less than 1% for both data sets (64.43% vs 64.9% for the Royce data set, 86.78% vs 86.05% for the Lab data), showing that the effects were minimal.

From Table 7.2, there is clearly a difference in the latency estimator performance that is dependent on the data set. For the $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ latency estimators, the correct choice percentage is high for both Royce and Lab data sets (between 94% and 97%), but around 20–25% lower for the RMBL data. On explanation for this may be the difference in detection data size for the RMBL experiments (128 kB for RMBL as opposed to 32 kB for both Royce and Lab data sets).

These results confirmed the expectation that the Dynamic Estimation policy would perform better than the Static Thresholding policy.

7.1.4 Accuracy of Dynamic Estimation latency estimators

Section 7.1.3 compared the relative performance of the two policies for Adaptation, Static Thresholding and Dynamic Estimation. Whilst Static Thresholding's adaptation evaluation was based on an empirical threshold τ, Dynamic Estimation made the choice to process locally based on estimates provided by one of two latency estimators: $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$.

Section 7.1.4 examines the accuracy of both $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ in predicting the latency of raw data transfers. The accuracy of latency estimates has a bearing on the generic suitability of Dynamic Estimation. If the estimates provided by $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ are sufficiently accurate, the Dynamic Estimation policy can be used to make Adaptation choices for arbitrary local processing times (rather than the specific t_{aml}). The metric M_{err} used to evaluate the accuracy of the latency estimates made by $\hat{\ell}_1(d)$ and

Table 7.3: The accuracy of transfer latency prediction. The columns show the percentage of relative error between estimated and observed latency, and each cell shows the percentage of predictions which fell within that interval. For example, 67.71% of transfers were within 50% of the actual value for the RMBL experiment.

	% of estimates within error bound		
Experiment	±50% error	±25% error	±10% error
RMBL ($\hat{\ell}_1(d)$)	66.86	46.46	27.20
RMBL ($\hat{\ell}_2(d)$)	67.71	49.01	28.33
Royce ($\hat{\ell}_1(d)$)	70.99	43.99	16.91
Royce ($\hat{\ell}_2(d)$)	75.88	47.76	18.75
Lab ($\hat{\ell}_1(d)$)	89.54	62.67	28.40
Lab ($\hat{\ell}_2(d)$)	90.14	63.52	28.66

$\hat{\ell}_2(d)$ compared to the actual transfer latencies $\ell(d)$ is given by

$$M_{i,err} = ((\hat{\ell}_i(d) - \ell_i(d))/\ell_i(d)) \cdot 100 \tag{7.1}$$

where $\hat{\ell}_i(d)$ is the estimated latency (estimated by $\hat{\ell}_1(d)$ or $\hat{\ell}_2(d)$) and $\ell(d)$ is the observed latency (from Section 6.4). This metric describes the estimate $\hat{\ell}_i(d)$ as a percentage of the actual latency. If $M_{i,err}$ is positive, the error is an over-estimate, and an under-estimate if negative. This allows for comparison of error across different latencies in a way that the absolute error would not be suited to. For example, a prediction of 0.01s seconds when the actual latency was 0.10s is only 0.09s error, but a prediction of 1s when the latency was actually 10s shows a 9s error. Using $M_{i,err}$ would show they have both been under-estimated by 90% (-0.9).

The evaluation was performed as follows: the results of applying $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ to the all of the detection transfer records in the RMBL, Royce and Lab data traces (as per the simulation in Section 7.1.2) were used with the actual latencies for each transfer to calculate $M_{i,err}$. Table 7.3 shows the results across each of the data traces (RMBL, Royce, Lab) with respect to three arbitrarily chosen accuracy bounds: ±50%, ±25% and ±10% of the actual latency. For each bound, the percentage of all latency estimates for a given experiment that were within these bounds are shown. The specific bounds were chosen to represent increasing degrees of accuracy: ±10% error in latency prediction could be considered *accurate*, ±25% considered *moderately accurate* and ±50% considered *not particularly accurate*.

Bearing these classifications in mind, the results in Table 7.3 do not show particularly accurate performance. An ideal result for the latency estimators would be to have a large percentage of the data transfers be classed as *accurate* (within ±10%) all of the time. However, for each of the three data traces, the latency estimates provided by $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$ were classed as *accurate* only 17%-29% of the time (depending on experiment). In fact, the latency estimates are for RMBL and Royce are only classed as *not accurate* (within ±50%) around 66%–75% of the time.

Overall, the accuracy of latency estimates was best for the Lab data set, where around 90% of all

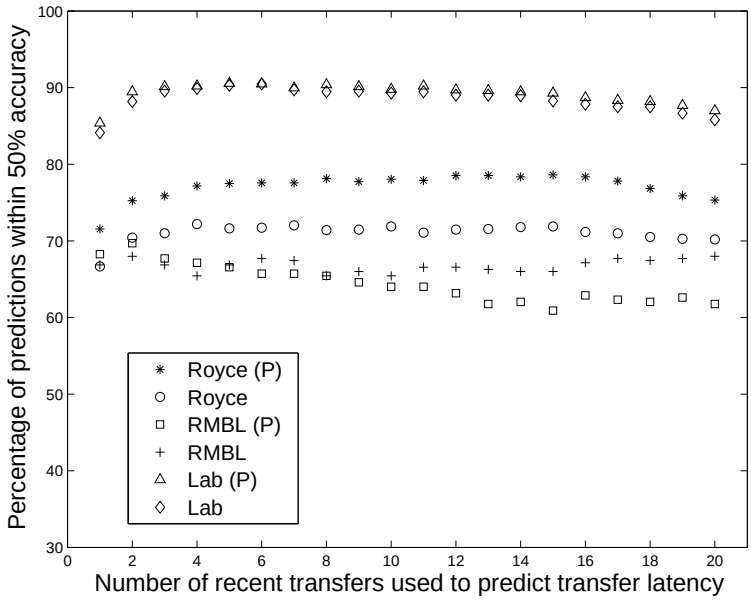


Figure 7.3: The trade-off of number of recent transfers to use in calculating the mean, for different data sets. Those noted with (P) on the legend used $\hat{\ell}_2(d)$, others used $\hat{\ell}_1(d)$.

transfers were at least within ±50% of the actual latency (although this is still classed as *not accurate*). The $\hat{\ell}_2(d)$ latency estimator shows better performance on the whole than the $\hat{\ell}_1(d)$ latency estimator, although they are never further than 5% apart in the different error bounds categories.

There is a clear gap between expected and achieved accuracy for both of the latency estimators. This has an effect on the general suitability of the estimators, particularly when considering arbitrary local processing times, as discussed in Section 7.1.6.

7.1.5 Effects of varying transfer history on latency estimation

The optimal number of previous transfers to be considered for predicting the latency of the next transfer was not immediately clear during initial simulation. To investigate this further, the Dynamic Estimation policy latency estimators were tested for a varying number of previous transfers. Both $\hat{\ell}_2(d)$ and $\hat{\ell}_1(d)$ were simulated with all data sets, using between one and twenty transfers in each latency estimate.

Figure 7.3 shows a graph of the number of transfers used in each latency estimate on the x axis versus the percentage of estimates within the ±50% accuracy bound on the y axis (as discussed in Section 7.1.4). The other error bounds (±25% and ±10%) showed similar trends, and are not presented here.

There is a drop in accuracy for the RMBL data set after 2 transfers used, indicating that using a small amount of history is sufficient. However, the Lab data set seems to stabilise after 3 transfers used, and the Royce data set continues to improve after as many as 8 previous transfers are used. The choice on how many previous transfers is unclear on these data sets, but the use of 3 seems to be an adequate trade-off, hence why this was used in the Dynamic Estimation policy simulation in Section 7.1.3 on page 175. Interestingly, there is a definite difference in the benefit gained by using previous transfers between $\hat{\ell}_2(d)$

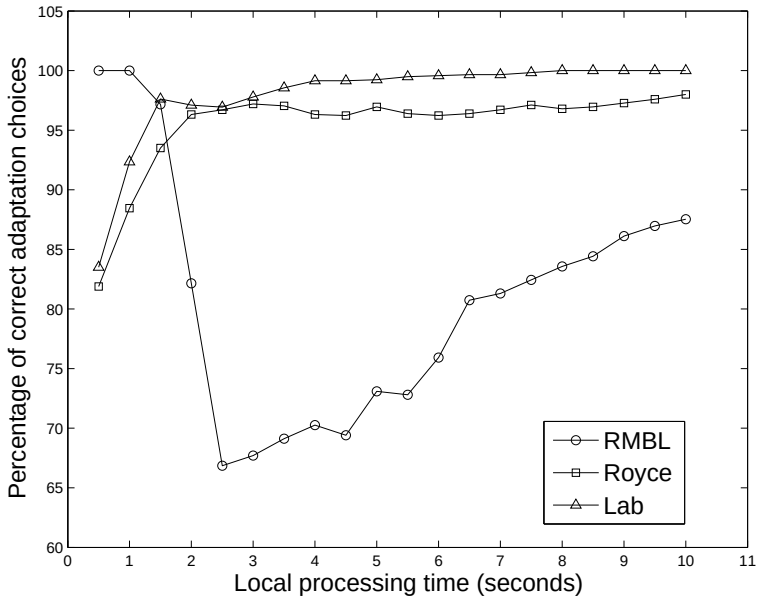


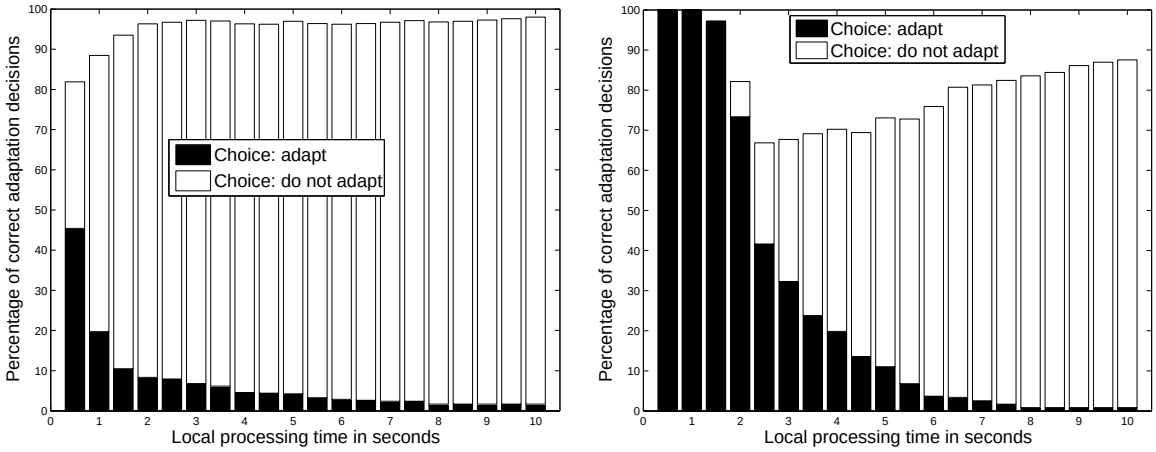
Figure 7.4: A graph showing different processing times vs correct percentage of Adaptation choices for the three different experimental data traces used.

and $\hat{\ell}_1(d)$. In general, $\hat{\ell}_2(d)$ seems to show better performance when using more than one recent transfer, most notably in the Royce data set, where it consistently outperforms $\hat{\ell}_1(d)$.

7.1.6 Effects of varying local processing time on correct choices

An important part of understanding the general suitability of the Dynamic Estimation policy is its performance with respect to different on-node processing latencies (rather than just the previously observed time to process the AML). To evaluate this, the Dynamic Estimation policy was simulated using the RMBL, Royce and Lab data sets, with the $\hat{\ell}_2(d)$ latency estimator. For all transfers in all data sets, twenty different local processing times were evaluated, ranging from 0.5s to 10.0s in 0.5s intervals. Because both latency estimators showed similar performance in the previous analyses, it was decided to use only the $\hat{\ell}_2(d)$ latency estimator in this analysis (for ease of presenting and analysing results).

Figure 7.4 shows a graph of the total correct decisions made by the Dynamic Estimation policy for each local processing time, and over each data trace. The x axis is local processing time and the y axis is the total percentage of correct choices the policy made. For the Royce and Lab data sets in Figure 7.4, Dynamic Estimation with the $\hat{\ell}_2(d)$ latency estimator never fell below 80% correct in making Adaptation decisions. Both Royce and Lab data sets see an increase in correct decision percentages as the local processing time increases. However, the RMBL data set shows a different pattern: the correct choice percentage starts high and reduces dramatically around the 2–2.5 second mark. In order to better understand the reason for the dip in correct Adaptation decision percentages, two bar graphs were plotted for both the Royce and RMBL data sets, showing the breakdown of total correct choices by choice type: either to process locally (adapt) or not (do not adapt). These are shown in Figure 7.5, where the x



(a) Breakdown of correct choices for Royce data set

(b) Breakdown of correct choices for RMBL data set

Figure 7.5: The relative breakdown of choices for each data set. Black shows correct decisions to adapt, and white shows correct decisions not to adapt.

axis represents the processing time, in 0.5s intervals, and the y axis represents the percentage of *correct* choices that Dynamic Estimation made with each latency estimator.

For the Royce data set in Figure 7.5(a), it can be seen that apart from the 0.5s local processing time, the relative breakdown of correct choices is more in favour of not processing locally (and the correct choice percentage increases as local processing time increases). This is because as the processing time increases, the network's latency is less than the local processing latency, hence more *do not adapt* choices are made.

For the RMBL data set in Figure 7.5(b), the correct Adaptation decision percentages are entirely dominated by choices to process locally between 0.5 and 1.5 seconds. As the local processing time increases, the total percentage of correct Adaptation decisions decreases and more choices to not process locally are seen. The total percentage of correct decisions is worst at 2.5 seconds, and increases gradually from then on (where a similar trend to the Royce data set is seen).

Taking the median transfer times across all transfers in each data set reveals a potential cause for the disparity in performance. The Royce data set and the Lab data set had median transfer latencies of 0.58s and 0.55s respectively, whereas the Royce data set had a median transfer latency of 2.98s. This disparity in median transfer latency was most likely caused by the different in raw detection data sizes that were transferred (32 kB for Royce and Lab, 128 kB for RMBL). If these median times are compared to the local processing times in Figure 7.4, it can be seen that the point at which each data set sees its worse correct decision percentage is around the point at which the median transfer time is located.

This shows that the Dynamic Estimation policy gets the least correct decisions around the point at which the median transfer time is roughly the same as the local processing time.

This makes sense based on the accuracy experiments: unless the latency estimators' predictions are accurate, they will perform worst at this point. It is be expected that a more accurate latency estimator

would improve the percentage of correct choices when the median transfer time is roughly the same as the local processing time.

7.1.7 Discussion

Section 7.1 has evaluated the Static Thresholding and Dynamic Estimation policies presented in Chapter 6. The aim of these policies is to help improve the end-to-end *timeliness* of the acoustic localisation system by allowing nodes to decide whether to process locally or not, based on observed network conditions. The first policy, Static Thresholding, used the ETX metric to decide whether a node should process locally or not. The ETX value used as a threshold was based on the observation that a node should process locally if it was between two and three hops from the sink. The second policy, Dynamic Estimation, made the decision to process locally or not based on the predicted latency of the transfer. This predicted latency was based on the recorded latency of previous transfers. Two separate latency estimators were proposed and evaluated: $\hat{\ell}_1(d)$ and $\hat{\ell}_2(d)$. The evaluation of the policies was performed in simulation, using data traces of network transfers gathered in-situ and through controlled laboratory experiments. The Dynamic Estimation approach is applicable to more general use, as it could be used for arbitrary data payload sizes, and arbitrary network sizes. This is because latency estimates are based on real data transfers that occur in the network, and are size-adjusted depending on the current number of hops the node is from the sink.

The latency estimators used in the Dynamic Estimation algorithm did not require direct consideration of lower-level layer specifics, such as TCP performance under high traffic, loss and congestion, and variable bit-rate transmissions. Instead, application level data was used: when a packet was sent (or queued to be sent), and when it arrived, as well as the number of hops a node was from the sink. This data was easier to gather and use in the latency estimators implemented.

In comparing the two Dynamic Estimation latency estimators, $\hat{\ell}_2(d)$ showed better accuracy at predicting transfer times over several data sets. It was also shown that the $\hat{\ell}_2(d)$ could work accurately against a variety of local processing times.

A problem for the Dynamic Estimation latency estimators is a lack of historical data: if relatively few transfers are made by each node, the latency predictions may not be accurate and therefore Adaptation decisions may not be correct. In practise however, it would be expected that the system would see data transfers until significant loss or congestion occurred, followed by periods of local processing, roughly echoing the real-time network state.

Adaptation was one of two refinements presented by the author to reduce latency, thus improving timeliness in the acoustic localisation system. The other refinement was Lazy Grouping which was validated as a proof of concept in Section 6.3.1. In Section 7.2, Lazy Grouping and Adaptation are considered in terms of their co-existence and integration in the acoustic localisation system. Potential problems and implementation points are discussed, as well as the general applicability of the approach used.

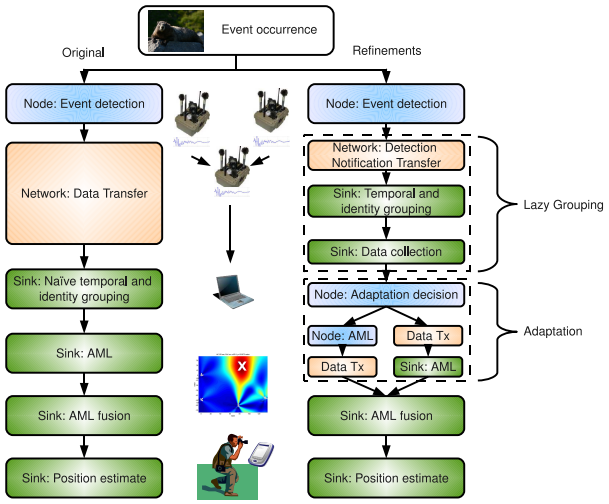


Figure 7.6: The comparison of the original processing chain and the refined processing chain, including both lazy grouping and Adaptation. The boxes indicate *where* the processing is taking place, followed by *what* is happening.

7.2 Adaptive system data flow and integration

Having presented the Adaptation and Lazy Grouping algorithms and evaluated them in isolation, it is important to understand how they would co-exist when integrated into the VoxNet system, and how the event flow would subsequently be modified. A comparison between the event flow in the original and the refined system (integrating both Lazy Grouping and Adaptation) is shown in Figure 7.6. The left hand side shows the original processing chain, and the right hand side shows the processing chain with both Lazy Grouping and Adaptation incorporated.

The refined system event flow is as follows (with reference to Figure 7.6): Nodes have detections triggered locally in response to the acoustic event of interest (Node: Event detection). The notifications of these events are sent to the sink (Network: Detection Notification Transfer), whilst the data corresponding to the event is kept in a ring buffer, from where it can be queried out at any time by the sink, using the node-specific sequence number to identify it. Upon receiving detection notifications, the sink attempts to group them using the online grouping algorithm described in Chapter 6 (Sink: Temporal and Identity Grouping). When it decides that there are enough events to process, it requests the data from nodes (Sink: Data Collection). Upon receipt of the request, the node then evaluates the Adaptation policy (Node: Evaluate Adaptation policy). If a node decides to process locally, it sends the result to the sink when finished (Node: AML, Data Tx). Otherwise, it sends the raw data associated with the detection (Data Tx, Sink: AML). When the sink receives an AML result back, it stores it in the record it holds for the group. When it receives raw detection data, it processes the AML immediately (as it knows already that the data is useful), and stores the result in the group record. The sink will not allow the group to be processed through the localisation algorithm until all requested data has been gathered and processed through the AML algorithm to produce DoA estimates. Following this, the AML results in the group are

passed on to the data fusion algorithm, from which the position estimate is determined and displayed to the user (Sink: AML Fusion, Sink: Position Estimate). Whilst conceptually the components fit together in the system flow, it remains to be seen whether the refinements would together improve the end-to-end latency or interfere with one another.

7.2.1 Integration issues

In order to enable the flow of this use-case, the system originally presented in Chapter 4 would need to be changed. Due to VoxNet's flexible architecture, both Lazy Grouping and Adaptation could be expressed in Wavescript, the language that VoxNet uses to describe applications. However, some changes would be necessary to lower level network components to support this.

In Wavescript, a program is defined as a directed graph, where data flow streams through processing operators towards an endpoint (as discussed in Chapter 4). In the original system's data flow graph streams of detections made by multiple nodes in the network flow toward the sink, where they are merged into a single stream, becoming a stream of AML results and then a stream of position estimates. However, Lazy Grouping implies a more complex data flow graph where data flows from nodes to the sink, then back out to the nodes, and back to the sink. In terms of a Wavescript program graph, the flow could be described as follows (assuming the flow starts at the event detection operator on each node): the output of the detection operator on each node is a stream of raw detections and a stream of detection notifications. The stream of detection notifications flows through a network operator (flowing to the sink). At the sink, the flows of detection notifications from each node are merged together by a *detection grouping* operator, which produces a stream of grouped data requests. This flows through a network operator, which pushes a stream of data requests to each node. The stream of data requests is merged with the stream of raw detections at each node, at a *data request* operator. The operator outputs a stream of raw detection data which passes through an *Adaptation* operator. The Adaptation operator produces either a networked stream of detection data (flowing to the sink), or a stream of data which passes through an AML operator, and then a network operator to the sink. At the sink, the incoming detection flows (either raw or processed detections) are merged into groups by an *Adaptation fusion* operator, which produces a stream of groups suitable for passing through the data fusion operator (to produce position estimates).

The event flow graph for Lazy Grouping and Adaptation as described above can largely be supported by application level implementation—Wavescript has existing support for event detection and network sending and receiving operators, for example. The operators described in italics in the above description can be expressed as user-defined Wavescript operators, the functionality of which is discussed below:

1. **Detection grouping operator**: collect all detections from nodes and merge them into a single stream of data requests, which flows back out to all nodes.

2. **Data request operator**: merge raw detection stream with the data request stream coming from the network, producing a stream of raw detections corresponding to the requests issued by the sink. In this way, no items flow from the raw detections stream to the sink until a request for an item can be merged with an item in the queue. This also implies that a policy would be required to evict

data from the input buffer if it has not been merged for some time.

3. **Adaptation operator**: takes as input a stream of raw detections, internally making a call to a lower level network function to get a prediction for a transfer. This produces either a stream of raw detections or a stream of processed AML results. This requires the use of a switch operator, which changes the path of the directed graph depending on evaluation of a switch condition—this type of behaviour is already available in Wavescript.

4. **Adaptation fusion operator**: merge raw detections and AML results coming from the network into correct groups to produce a stream of grouped, AML-processed data which can flow through the data fusion operator (to produce a position estimate).

In addition to user-defined network operators to support Adaptation, VoxNet's network stream layer (which sits under the Wavescript program and above the data transport layer) would need to be modified to record the time taken and data size for each transfer that had taken place. This change would be implemented for both the sink side and node side. The sink would need to keep a record of all transfers for all nodes, and each node would need to keep a record of its transfers to the sink (or the most recent transfers). Additionally, on the sink-side, a change would have to be made for the underlying stream mechanism, so that the sink sends back the time a transfer took as a field in the final data acknowledgement message (as discussed in Chapter 6, Dynamic Estimation). This would enable both the sink and node to track all transfers that have taken place. This transfer data would be made available to the application layer by means of Wave Script's foreign function interface. The function would query the mean goodput of the last x transfers from the stream layer.

In summary, Lazy Grouping and Adaptation can theoretically be expressed as a Wavescript graph, where most functionality can be implemented as user-defined operators to carry out the various tasks that make up the algorithms. However, some support is required from the network layer, via modifications to the networked stream layer. The next consideration for the refinements is how they may be improved to enable a self-organising, adaptive network which can provide the output required by the user without user intervention.

7.2.2 Further algorithmic refinements

Further refinements to both Adaptation and Lazy Grouping become apparent when considered in the context of the system as a whole. Assigning priority to data collection, important run-time parameters, data collection timeout and the data collection method are considered next.

In a network where many events are being triggered, a trade-off presents itself—is it more important to keep up with current events (ignoring older data), or is it more important to make sure all data that was requested gets collected? In the motivating end-to-end system use-case, *timeliness* is the goal—it is important that the position estimates arrive in enough time that they can be acted upon. In this case, the scientist may want to ignore events that have not been collected after 10 seconds, for example. In general, the *priority* of data may have different meanings depending on the relative importance of certain data to the application goal. Several examples are highlighted below, reflecting different data request scheduling techniques that could be used under network load.

- **Stack**: newest data requests are fulfilled first.
- **Collection time**: data requests could be ordered according to how long the sink estimates it will take to gather all of the data.
- **Size**: data requests could be prioritised according to the size of the groups the sink has made—for example, choose groups that have four observations before any others, or, always process the biggest groups first.
- **Queue with data age**: oldest data requests are fulfilled first, but if the *data age* limit is reached on outstanding data requests, they are cancelled.

The Dynamic Estimation latency estimates (made by either $\hat{\ell}_1(d)$ or $\hat{\ell}_2(d)$) could be used to dynamically calculate *data age* for each group created by the Lazy Grouping algorithm. The data age would be determined as the largest predicted latency in a group. If the data has not been received by this time, the sink would cancel the transfer of any data that had not yet been received and carry on with data fusion. The user may also benefit from having control over the minimum and maximum number of *detections per group*. This will affect how long it takes to gather all of the data for a position estimate. If it is assumed that more observations equates to a higher accuracy position estimate, then latency may be traded off with accuracy. This will have an effect on the *timeliness*.

The final adjustment proposed here is related to the data gathering approach. When the Lazy Grouping algorithm has formed a group, it requests data from all of these nodes simultaneously. It may make more sense to request data for a group on a node-by-node basis, or to at least stagger the requests. This may help reduce channel contention, and therefore TCP congestion. However, this request method was not evaluated, and is left for future work.

7.2.3 Discussion

Section 7.2 has highlighted some potential issues to be resolved when integrating Lazy Grouping and Adaptation into the VoxNet system, and the further algorithmic improvements that could be made. Some aspects have either not been evaluated in-situ, or have not been considered in the evaluation presented in this thesis. The reasons for this are discussed below.

It is clear that further experimental work is need to evaluate the impact of integrating Lazy Grouping and Adaptation. However, experimental design to support this could be potentially difficult. In Section 7.1, experimental data traces were used for evaluation. This allowed proof of concept simulations to be carried out. However, performing in-situ evaluations of the proposed components is a more complex task. The main problem lays with ensuring repeatability to allow for comparison between different approaches and integrated system versions. This is particularly problematic for the Adaptation component, which makes decisions based on the dynamic nature of the network. The experimental plan described in Section 7.1 on page 172 helps in terms of providing a repeatable simulated event generation scheme, but does not account for differences in the network behaviour. Therefore a full evaluation of the improvements proposed in this chapter require in-situ, parallel deployment of the respective system iterations for long periods of time. However, in reality this is difficult, and seldom seen in real sensor

network deployments, due to expense of equipment, and the physical difficulty of deploying two separate networks. This evaluation is left for future work.

The work reported in Chapter 6 and evaluated in this chapter focused mainly on the latency effects seen at the nodes in the network and their contribution to the end to end latency seen in estimating position of acoustic sources. This view is not complete however, as it does not consider the factor of sink-side processing. At the sink side, there are potential performance bottlenecks in the data fusion algorithm that stem from how the search space is defined and the number of detections used. Because of the comparatively limited local processing resources, the sink-side AML calculations are cheap compared to on-node calculation. The localisation algorithm is slightly more expensive, as it requires a search space to be defined in which the estimated position is expected to be (based on how the scientist has deployed the network). As discussed in Chapter 5, this can potentially provide a bottleneck if the search space is too large, or the resolution of the search space is too high. This will increase the end-to-end latency of the localisation system.

7.3 Generic adaptive acoustic localisation

The system that has been designed, evaluated and refined in this thesis is fundamentally an event-based system, exhibiting some level of adaptive/autonomous behaviour, rather than a basic sense and send system. The adaptive aspects were designed into the system for two reasons: (1) the data load on the network is too high to stream all of the data back to the sink over the network, and (2) the system is intended to be used in an on-line manner meaning it is not sufficient to gather and archive data, but also data must be processed whilst the system is running. Although the use-case presented to motivate the acoustic localisation system was specific, the issues raised and addressed are more generic. Thus, the marmot localisation application can be seen as an instantiation of a more generic acoustic localisation system data flow. From this viewpoint, the generic flow of an end-to-end acoustic localisation system as seen throughout this thesis is as follows:

1. **Distributed acoustic event detection**: events of interest are detected within the network
2. **Event grouping**: events of interest are grouped together according to certain criteria (for example, time).
3. **Data collection**: data is selectively collected from the network, based on its relevance to the overall application goal
4. **Data fusion**: relevant data is combined to provide the final observation
5. **Information representation**: the resulting observation is provided to the user in some form (such as visualisation)

In the marmot localisation system, the on-node event detectors instantiated the event detection stage, Lazy Grouping instantiated the event grouping and data collection stages, and the DoA-based pseudo-likelihood localisation algorithm instantiated the data fusion and information representation stages.

If each of the above steps is viewed as a modular component in the application flow, then adapting the flow to detect a different phenomena requires that the event detection component be substituted for a

different one (Chapter 5 discussed the implications of different event detectors). The other components—data grouping, data collection and data fusion would not need to be changed. However, other components could feasibly be used, such as different localisation algorithms. For example, the data gathered for position estimation might be time of flight distance measurements rather than angle of arrival measurements, and localisation may be performed using multilateration, for example.

7.4 VoxNet and other high data-rate systems

The data flow presented for the marmot localisation application in Section 7.3 indicates a generic data flow which can be used to describe a given on-line source localisation application.

In Section 2.3 on page 16, other high data-rate systems were surveyed. Like VoxNet, most of the systems discussed were custom, application-specific implementations which needed to deal with the constraints high data-rate applications, but used mote-class devices.

Using constrained devices meant that only limited processing was available on each node. Hence, the common characteristics with respect to on-node processing in the systems discussed were:

- Data filtering: send only useful data to the sink. This is manifested either in on-node event detection or sink-based data collection.
- Data compression: intelligently compress data streams such that they can be transferred more easily over the network.

The vast majority of data flows were linear, with the following pattern: data is sampled at the node level, potentially filtered, compressed and then sent to a higher tier microserver or sink (for high level inference). The only exception to this was the one formal, generic framework that was presented: Lance (Werner-Allen et al. 2008).

Of all of the systems presented in Chapter 2, Lance is the most similar to the instantiation of VoxNet. The authors of Lance recognised this fact, and indicated in their paper that the marmot localisation application could indeed be described in the Lance framework (Werner-Allen et al. 2008). The authors indicate that the event detector for the marmot localisation application could be implemented as a *triggered* summarisation function, which is fundamentally different to the continuous summarisations which the evaluated version of Lance uses.

However, VoxNet's data flow can be likened to the data flow in Lance: event detections are summaries, sent to the sink. The grouping algorithm in Lazy Grouping is a policy module which temporally correlates incoming detections. The data collection protocol of Lazy Grouping is similar to the scheduler, in that it sends requests when data is deemed to be useful. Whilst VoxNet does determine which data is useful at both node and network level (event detections and groups respectively), the scheduling constraints are slightly different. Lance is primarily configured to optimise the network lifetime, whereas VoxNet is focused on end-to-end timeliness as the main goal. The Adaptation policies of Static Thresholding and Dynamic Estimation were hence developed for VoxNet to support this.

Indeed, the feature which VoxNet provides that Lance does not is that of Adaptation: the ability for a node to decide whether it should process a part of the data before sending. The reason Lance does not cater for this component is Lance's generality—the framework is primarily for data collection, not

full application support. Hence, while VoxNet tries to optimise data transfer for end-to-end position estimation latency by performing some of the processing locally, Lance cannot assume that users want to do anything more than collect *relevant* data.

Like VoxNet, all of the other systems surveyed in Chapter 2 suggested either on node compression or filtering as local processing operations: NetSHM discusses two modes of operation, raw data collection and locally processed data collection. Similarly, VanGo offers a set of time-domain related filtering, event detection and compression libraries. However, none of the systems discussed were *individually adaptive* in their behaviour toward on-node processing: all nodes process one way or another. In fact, the authors of VanGo noted that individual Adaptation had not been needed in their application experience (Greenstein et al. 2006).

Therefore the contribution made by Adaptation is two-fold: the identification of a scenario where Adaptation could actually be useful, and the instantiation of Adaptation for a real application. In the specific marmot localisation application, only the AML algorithm was processed locally or at the sink. However, this represents the application-specific instantiation. It could be envisaged that the dynamic node-side processing component could be any application-specific data processing algorithm, rather than an AML computation. Certainly, the Dynamic Estimation policy with its latency estimators would be able to support this: all that is needed would be a profile of the time that the processing takes on a node.

It is of course noted that the hardware used for implementing VoxNet is considerably more capable than the motes used in all of the other systems discussed above and in Section 2.3, apart from the Toad Monitoring application (Hu et al. 2005). This actually strengthens a view that has been central to this thesis: the device should not dictate the application functionality in the early stages of development, but the application should. The toad monitoring application in particular holds testament to this: the original system implementation was not hindered by mote capability, and was able to perform the processing required by the application. However, when it came to transferring some of the processing to motes, it became impossible to keep up with the required data rates, resulting in 60% of data being lost. Subsequent solutions to this problem were unconvincing: scheduling sampling between nodes introduces further complexity but reduces the amount of data that can be gathered.

7.4.1 Summary

Section 7.3 has presented a generic data flow for acoustic source localisation based on the original marmot localisation application and the subsequent refinements to improve latency by gathering only useful data and processing locally wherever possible. It has been argued that the processing flow originally defined for on-line marmot localisation is potentially applicable to a variety of acoustic localisation applications. An important contribution here is the identification of two behaviours which characterise high-data rate applications: filtering behaviour and dynamic behaviour. These general behaviours can be used to classify a wide range of application-specific approaches to high data-rate, potentially on-line applications. Section 7.5 validates this by providing a comparison with two existing high-data rate frameworks and identifying instances of both filtering and dynamic behaviour.

7.5 Discussion

This chapter and the previous chapter have introduced and evaluated two significant contributions to improving the robustness and timeliness of a marmot localisation application built using the VoxNet platform: Lazy Grouping and Adaptation. The integration of these features into the original VoxNet system has been considered in detail, along with the issues raised by their implementation and co-existence. The evaluation of Lazy Grouping and Adaptation within the specific context of a marmot localisation application has provided a revised data flow which could be used for a variety of *on-line* acoustic localisation applications.

Whilst these contributions have improved the performance of the specific marmot localisation application and given insight into general localisation applications and high data rate applications, it is important to consider *why* they were implemented in the context of the application-centric view on the WSN design space.

Chapter 2 introduced three different views on the WSN design space: the network-centric, the device-centric and the application-centric. The design and development of the marmot localisation system was deliberately approached from an application-centric viewpoint. This meant that the specific application requirements defined the design and implementation of the system and dictated what components were necessary for the system to operate. It was convenient to approach the system development from an application-centric viewpoint given the computing resources available in the Acoustic ENSBox devices. With VoxNet, it was not necessary to be overly concerned with the device-centric view, as the ENSBox V2 platform is a highly capable, *microserver-class* platform. The high-bandwidth 802.11b radio links used by the V2 nodes provided their own problems (concerning bit-rates) when compared to low power fixed bit-rate standards like 802.15.4. However, they provided a vastly superior bandwidth by comparison, meaning large data files could be sent over the network at high rates using familiar TCP/IP services.

Only after the system was deployed and observed in-situ did the need for improvements related to robustness and reduction of end-to-end latency become apparent. The observed application-specific performance issues provided the motivation to implement adaptive, dynamic behaviour in the network.

From a network-centric view, one might ask whether the changes implemented were generally useful for arbitrary WSNs. From a network-centric viewpoint, Adaptation is a distributed, in-network processing solution. Adaptation allows individual nodes to dynamically react to changes they observe in the network to minimise latency. However, the approach is not collaborative per se as nodes do not explicitly coordinate with one another to achieve this goal. In contrast, Lazy Grouping is not distributed: it implements a centralised algorithm at the sink to coordinate data collection from nodes. From a network-centric viewpoint, the Lazy Grouping approach to data collection is not scalable (in the network-centric sense) as it has one point of failure (the sink) and will tend to put communication stress on nodes close to the sink.

A fully distributed approach to Lazy Grouping could be taken by attempting to perform all of the processing collaboratively, in the network, rather than relying on a centralised point to take care of elements that require inputs from multiple nodes. However, this approach requires a drastic increase in complexity: nodes would most likely have to form clusters and exchange observations to reach a consensus

on whether an event has been detected before negotiating further processing, such as local AMLs and a distributed version of the localisation algorithm.

It is not clear that this extra complexity would be required in this application specific case, or that it would be quicker than the approach that was adopted in this thesis. Indeed, a fully distributed approached to Lazy Grouping was not taken because it was not required when considering the application performance and the amount of nodes available to experiment with. If the network were to scale, then it is possible that a fully distributed approach *might* be needed.

Other desirable network-centric features were not considered because given the specifics of the marmot localisation application and the amount of hardware available. For example, because the nodes were only deployed for several hours at a time in attended deployments, energy management for network lifetime was not a concern.

Issues of scalability cannot be completely ignored in the general class of localisation applications. Instead, they should be considered from an application-centric context. Two reasons to scale an acoustic sensing system (of which an acoustic localisation system is an example) are to increase *density*, or improve *coverage*. Density is increased by deploying more nodes in a given area. This would be done to provide redundancy (hence ensure full functionality in case of failure) or potentially increase accuracy (of localisation estimates, for example) by adding more observations.

However, increasing the deployment density in a given area may provide a diminishing return, where the decrease in available network bandwidth as more nodes are added means receiving an end-to-end position estimate, for example, may actually end up taking longer. The implications of density and coverage are more complicated than this scaling-based treatment would indicate, and are discussed further in Chapter 8 as future work.

To summarise, the application-centric viewpoint was necessary to help identify the changes which would not impact positively the system performance. It was only through real-life evaluation that these observations could be made. Using embedded hardware that was not heavily resource constrained allowed the application-specific changes to be evaluated without being clouded by other issues, such as energy management.

The next chapter provides a conclusion to the thesis. The contributions presented by the thesis as a whole are summarised, and the answers to the research questions presented in Chapter 1 are provided, with appropriate conclusions.

Chapter 8

Conclusion

This chapter concludes the thesis. Firstly, the contributions of the thesis are listed. Secondly, answers to the research questions posed in Chapter 1 are provided, including cross-references to where the associated evidence is presented in the thesis. Next, future work is then discussed, and finally closing remarks are presented.

8.1 Thesis overview

Both self-localisation and source-localisation in WSNs were examined in this thesis. The motivating application was the source-localisation of marmots, which required: (1) accurate 3D node-localisation to provide a frame of reference for the animal localisation and (2) some form of in-network processing to filter the large amount of data that was generated so that it could be transferred over the wireless network in a timely manner.

A key theme in this thesis has been the WSN design space, and in particular taking an application-centric view on the WSN design space, where the application requirements are the primary influence on the systems' design. Chapter 2 discussed the advantages of taking an application-centric approach to WSN design over network or device centric views. Having an application which provided realistic requirements and sufficient reason to develop in-network processing was vitally important to the development, deployment, evaluation and refinement cycle which the system went through in this thesis.

The problems faced by the marmot localisation application are indicative of the issues faced by WSN-based high data-rate systems in general.

8.2 Contributions

This thesis contributes both to acoustic-based self- and source-localisation. These contributions are listed below, along with the chapter in which they appear.

The contributions to self-localisation are as follows:

- The establishment and justification of an evaluation cycle for self-localisation algorithms based on simulation, emulation and deployment (Chapter 2).
- Experimental characterisation and evaluation of three acoustic ranging mechanisms and two localisation algorithms on platforms with varying computational capabilities, in both indoor and outdoor environments (Chapter 3).
- Implementation and evaluation of a proof-of-concept platform for acoustic ranging, including hardware and software integration and implementation of a suitable ranging algorithm (Chapter 3).

The contributions to source-localisation are as follows:

- Two designed, implemented and deployed iterations of an end-to-end, on-line, marmot localisation system. This included in-situ deployment and micro-benchmarks of application specific and general aspects of the system: on-node processing, on-node data archiving, data transfer reliability (Chapter 4).

- The identification and analysis of deployment-related systems issues that are not easily resolved without in-situ operation (Chapter 5).

- The design, implementation and proof-of-concept evaluation of two refinements that increase robustness and timeliness of the marmot localisation system—*Adaptation* and *Lazy Grouping* (Chapters 6 and 7).

These contributions emerged as part of the process of answering the research questions posed in Chapter 1. Section 8.3 revisits these questions and answers them, providing supporting evidence.

8.3 Research questions

8.3.1 Can a class of WSN applications be identified which require 3D self-localisation?

Yes, a class of WSN applications which required 3D self-localisation are source localisation-motivated applications where the sensor network is deployed over an irregular terrain to localise a target based on characteristic signals it emits. This was discussed in Chapter 2. Source localisation applications were a motivating application for self-localisation in this thesis: the source localisation application was the localisation of marmots in their natural habitat.

8.3.2 Does the performance of existing 2D self-localisation algorithms change for these applications?

Yes, the performance of algorithms normally used for 2D self-localisation can change for applications requiring 3D localisation. This question was answered in Chapter 3, where lateration (a common primitive calculation for position estimation) was tested using Mica2 motes that estimated range between one another using a lightweight, but low-precision ranging mechanism. In 3D, lateration was found to be sensitive to both reference point (anchor) placement and measurement noise (in range estimates). Inaccurate range estimates we seen to produce error that was an order of magnitude greater than the actual distance between devices. Self-localisation in 2D can be used to approximate over 3D environments, but as the deployment surface becomes more irregular, the 2D approximation becomes a worse representation of the actual positions (this was discussed in Chapter 3).

8.3.3 Are there design trade-offs which can ensure adequate 3D self-localisation performance for a given application?

Yes, there are design trade-offs that can be made to ensure both adequate and accurate 3D self-localisation. Localisation accuracy is largely affected by the accuracy of the input data: reference positions, range estimates and angle estimates, meaning that providing accurate input data gives a better

chance of obtaining better positional estimates from the localisation algorithm. Therefore, the main trade-off in localisation performance is accuracy versus platform and computation cost.

In Chapter 3, the evaluation of three acoustic ranging mechanisms of varying computational complexity were presented on three different sensing platforms (with varying computational resources). It was found that the constrained, generic, commercial WSN device (the Mica2) with a lightweight ranging mechanism implementation provided inaccurate range estimation data, with an operational range of around 10m, a low precision and was strongly effected by multipath and echoes in the environment. Conversely, a resource-rich, custom-made platform (the Acoustic ENSBox) could provide accurate ranging with high precision (order of centimetres of error) and large operational range (at least 100m). However, it required highly accurate time synchronisation, intensive signal processing, a powered speaker and microphone array to achieve this. A proof-of-concept sensing platform based on the Gumstix platform and a ranging mechanism from the literature were implemented as a compromise between the accuracy of the custom-made platform's ranging mechanism and the lightweight ranging implementation used on the generic WSN platform. In ranging experiments, the platform was found to be able to reach an operational range of 30m with mean residual standard deviation of ±0.48cm (of samples taken at intervals over the operational range).

8.3.4 How is the design and integration of an acoustic marmot localisation system affected by real-time, interactive requirements?

In Chapter 4, two iterations of an on-line, interactive marmot localisation system were designed and implemented. The system was required to present position estimates of marmots to the user in a timely manner, so that they could interact. The localisation system's data flow was based on an existing localisation toolchain, used in an off-line context. In using these components in a source localisation system required to be used on-line and interactively, four design considerations were needed that would not be required for a purely off-line solution: (1) on-line event grouping, (2) reliable data transmission, (3) on-line interaction, and (4) real-time, on-node data archiving. These are now discussed in more detail.

For *on-line event grouping*, it was necessary to implement a policy that logically grouped detected events arriving at the sink (sent by nodes) such that they could be processed into position estimates in a timely manner. Section 4.3.2 provided an implementation of this.

Reliable data transmission was as important to consider as timely data transmission. To support reliable transmission, the network used a stream abstraction built on TCP to support applications running over multiple network hops streaming detection data to the sink (Section 4.4.4 on page 110).

To support *on-line interaction*, it was necessary to develop a visualiser (to display position results) and more importantly an interactive shell to allow the user to send commands to the nodes in the network, determine the status of nodes and issue new programs. The Wavescope Shell (WSH) was developed to support this, in Section 4.4.4 on page 112.

Real-time, on-node data archiving was important to support off-line analysis of data gathered during in-situ experimentation, in addition to the real-time localisation provided by the system as a whole. This was enabled through the *spill to disk* component, described in Section 4.4.4 on page 114.

8.3.5 What are the user and deployment related challenges to be overcome to ensure adoption of an end-to-end marmot localisation system?

As discussed in Chapter 5, the user and deployment related challenges that can help to ensure adoption of an on-line marmot localisation system fall into three categories: (1) Application specific support, (2) General support for on-line operation and (3) General support for off-line operation.

Application specific challenges were: correct grouping at the sink of incoming detection data from nodes, identifying false detections, and the correct choice of localisation algorithm. The application-specific challenges mainly related to the latency of data transfer, a factor which is important if the system is to be fit for purpose (that is, allow observations to be made in a timely manner to help augment the user's notes during deployment).

General support for on-line and off-line operation included: data logging, stream priorities, and data consolidation. In addition, specific user-interaction related challenges were raised: visualisation for arbitrary data streams (both on-line and off-line), intuitive data access (for off-line interaction), and automation for fault tolerance and robustness within the system.

8.3.6 How can in-network processing capabilities aid the robustness and timeliness of on-line acoustic localisation?

Timeliness refers to the end-to-end latency between detecting events within the network to presenting a position estimate to the user. *Robustness* refers to the ability of the network to dynamically adapt to environmental changes in the network which are detrimental to the system timeliness. These include changes in network connectivity and false detections (signals that trigger detections but are not created by the source of interest).

In-network processing aided both the robustness and timeliness of the on-line acoustic localisation system as an outcome of meeting two principles: (1) sending only data that is useful for the network's overall aim, and (2) only processing locally when it is advantageous to do so. Two approaches were presented and evaluated in Chapters 6 and 7 to address these principles: *Lazy Grouping* and *Adaptation*.

Lazy Grouping is a centralised algorithm which runs at the sink and controls what detection data nodes send to the sink, based on whether that data is *useful* or not. In this case, useful data are detections that were triggered within a small period of time (0.44s) by three or more nodes. Simulation of Lazy Grouping using a realistic data trace (Chapter 6) showed a 64% reduction in data transferred could be achieved, preserving precious network bandwidth. A simulation of data collection showed it was possible to reduce latency of collection of data by as much as 75% on average (five seconds vs twenty seconds to collect five detections from five nodes).

Adaptation is a policy that runs on each node in the network. It decides whether a node should process detection data locally and send the result to the sink, or pass the raw data to the sink for processing. The policy decision is taken according to either a static or dynamic predictor. The static predictor provides an empirically observed measure of estimated link quality—the ETX value. The dynamic predictor estimates the predicted latency of a data transfer based on previous transfer latencies. Predicted latency is calculated in one of two ways: the time taken since data was queued to send until it was received, or the time taken since the data left the node and arrived at the sink. Simulation of the Adaptation

policy for both Static Thresholding and Dynamic Estimation showed that Dynamic Estimation lead to correct local or sink processing decisions 70–97% of the time, and the Static Thresholding lead to correct decisions 65–85% of the time.

8.3.7 Are there data processing approaches used in the developed system that are shared by a class of on-line, high data rate WSN systems?

Yes. The data processing approaches used in the developed system were (1) an on-line marmot event detector (Chapters 2 and 4), (2) Lazy Grouping (Chapter 6), and (3) Adaptation (Chapters 6 and 7).

The on-line event detector, whilst not a contribution of this thesis, is an example of on-node data filtering (or transformation) in order to reduce the amount of data sent over the network. Generally, on-node data filtering is instantiated as data compression or event detection (or a combination of the two) in several high data-rate systems, such as Wisden (Xu et al. 2004), NetSHM (Chintalapudi et al. 2006), Toad Monitoring (Hu et al. 2005) and VanGo (Greenstein et al. 2006).

Lazy Grouping is an example of policy-based data collection, where data is collected according to a certain policy. In Lazy Grouping, the policy was based upon the number of temporally correlated detections. This data processing approach is also seen in Lance (Werner-Allen et al. 2008), a data collection framework for high data-rate systems.

Adaptation is an example of data processing not seen in other high data-rate systems in the literature. This type of data processing is on-node, dynamic data processing.

8.4 Future work

Several areas for future work have been identified: (1) characterisation and measurement (2) evaluation of refinements, (3) a new iteration of the VoxNet platform (4) smarter sensing (including automated census, echo detection and accuracy/density studies). Section 8.4 discusses these areas in detail.

8.4.1 Characterisation and measurement

The Gumstix platform presented in Chapter 3 was evaluated as a proof of concept to show that accurate ranging was possible with a modest increase in processing resources compared to the Mica2. Hence, a full evaluation of the platform was not necessary for the purposes of this thesis. If the platform were to be used in an application, consideration would have to be given to 2D/3D localisation-based evaluation as well as packaging.

Also in Chapter 3, experimentation was performed with the ENSBox V2 to determine if the speaker block was causing obstructions that could affect AoA measurements. It was found to have a negligible effect. A thorough characterisation of the AoA mechanism on the V2 ENSBox as per the V1 ENSBox (Girod 2005) was out of the scope of this thesis, but may be important for others using the V2 platform further.

Finally, the addition of a thermistor to compensate for temperature related changes in the speed of sound on both the Gumstix and ENSBox V2 platforms would be desirable, although did not affect the observations made in this thesis. A significant amount of research would be needed to determine how accurate the compensation needs to be, related to the observed accuracy of the ranging or AoA mechanisms.

8.4.2 The VoxNet platform—a new iteration

When implementing the on-line source localisation system, a variety of issues related to timeliness of data transfer became apparent. Two refinements where implemented to deal with this: Lazy Grouping and Adaptation. Both refinements were implemented and their effectiveness evaluated in simulation (using realistic data traces), as a proof of concept.

It was beyond the scope of the work in this thesis to produce a third iteration of the acoustic localisation system in order to evaluate the refinements, which is a significant undertaking (as discussed in Chapter 7). However, the VoxNet project will undergo a third hardware and software iteration before its completion, meaning that the changes presented here will be able to be integrated and evaluated.

Chapter 5 identified several aspects of VoxNet which were presented in design but not implemented in the evaluated system. They were related to the post-deployment phases of VoxNet's operation, where data is consolidated, archived and made available for off-line processing:

1. A unified data consolidation mechanism for collecting raw, processed and log data from nodes (and the control console) and archiving it post-deployment, for further off-line analysis.

2. Storage of said data in such a way that it can be easily and efficiently accessed.

3. Back-end and front-end support to query data from the archive and process it using VoxNet programs in an off-line manner.

These features were not a direct requirement of the motivating application (the on-line localisation of marmots), hence it was beyond the scope of this thesis to implement these aspects of VoxNet. Aspects of the functionality listed above will be developed as part of the VoxNet project as it continues.

8.4.3 Automated census

The work in this thesis resulted in an on-line, real-time acoustic source localisation of an animal (the marmot). This enables the study of movement patterns and behaviour of animals and birds around their natural habitat. When localisation is performed in an on-line manner, it allows new forms of user interaction that are not possible when data is collected for off-line analysis only: a picture of the calling animal can be taken (either automatically or manually) to augment observations, for example.

However, source localisation forms part of a larger goal for VoxNet: automated census of animals and birds in their natural habitat (as noted in Chapter 2 and Chapter 5). Census involves estimating the population of a certain species in an area. For automated acoustic census, it is important to be able to detect a species of animal or bird, and differentiate between individuals within that species. For this, localisation is potentially a tool that can be used to disambiguate simultaneous calls, or similar calls coming from different physical locations.

Being able to automatically classify not only a species, but individuals within a species is an ongoing research effort, both for VoxNet and other projects (Trifa et al. 2008).

Future work would expand the source localisation developed in this thesis to include classification, potentially demonstrating the use of localisation-based disambiguation.

8.4.4 Echo detection

In the off-line analysis of the data gathered in a deployment at Rocky Mountain Biological Laboratory in 2006, Ali et al. (2007) identified a problem in their source localisation results where reverberations appeared to cause correlated error in position estimation. The authors hypothesised that a single node was the source of this error, and removed it from the calculations. Subsequently, the mean and standard deviation of localisation results improved, validating the hypothesis.

This observation raises an interesting question: assuming the reverberence associated with immediate environment surrounding a node can affect network-wide localisation results, is it possible for nodes to automatically identify when they are in a reverberant position and take some form of remedial action?

An ideal solution in the context of automation would be if the node could autonomously determine how reverberant its environment is, and assign itself some indicative value which represents a measure of how reverberant the position it is placed in is. Then, for a given detection, its reverberance could be used to weight the node's contribution to a particular localisation computation. Similarly to the motivating observation, the node may just vote itself out of a particular localisation computation.

Additionally, this could help the user during the deployment phase. Nodes could raise hints to the scientist, that (1) the device placement may need to be reconsidered, or (2) using positions would induce a level of uncertainty in the position estimates. The user could then reconsider how important the current position of the node is compared to the uncertainty that it may introduce into a position estimate. It is possible that a node could perform a reverberance measure using its own speaker and recording facilities. It could record itself producing an acoustic signal, then look for echoes of the same signal, or even the direction that the signal came from.

8.4.5 Accuracy and density

Following on from the quality of observations made by nodes, it is also possible that the accuracy of a position estimate can be affected by how many node observations are used in the localisation computation.

Understanding and quantifying the trade-off between localisation accuracy and density of deployment is an interesting area for future work given its implications for both deployment and in-situ interaction.

Chapter 7 alluded to accuracy and density in the context of their relevance from an application-related standpoint. From a deployment standpoint, the implications of accuracy and density of deployments are compelling: understanding how the density of a deployment affects the resulting localisation accuracy is important to the system user, as it means the following questions can be posed:

1. What is the minimum number of nodes required to cover a given area to guarantee a certain localisation accuracy?
2. Given a certain number of nodes and a required accuracy threshold, what is the maximum area that can be covered?
3. What localisation accuracy can be expected from a given number of nodes over a given area?
4. Does increasing the density of nodes in an area increase the accuracy? If so, what is the limit of this accuracy?

Answers to these questions can help the user develop appropriate deployment strategies. It could also help users *zoom in* on an area during deployment, potentially by reducing the density in a different area (while perhaps still maintaining the same detection coverage). Common application deployment requirements are that the system covers a certain geographical area, or achieve a certain level of accuracy. The amount of nodes available, or feasible to deploy may be small, so understanding the effect of deployment density with respect to coverage and accuracy could be an important deployment-time hint to the application domain user.

8.5 Final thoughts

Localisation has been an enduring research area for wireless sensor networks. It has yielded many theoretical studies, and an increasing number of realistic deployments. It is clear that there is still no *one size fits all* solution to the localisation problem. It is the author's opinion that the general solution will lie in radio, as it previously has for the ubiquitous Global Positioning System. It is the only general solution which will allow sensors to utilise the medium which is present in any wireless embedded system to perform localisation. An enduring problem which may be solved by wireless communication for localisation based systems is dealing with Non Line-of-Sight conditions. UltraWide Band (UWB) solutions have promised to, and still may, provide a realistic solution to localisation in highly obstructed environments. However, it is not clear when transmitters and receivers will be widely available (due to the potential interference UWB can cause on the frequency spectrum).

This observation in no way invalidates the work in this thesis: wireless localisation represents an area still under research that cannot offer immediate, accurate solutions in the same way that acoustic localisation can. This thesis has shown that sufficient processing power is required to create accurate ranging, but that for applications where devices are performing acoustic, or other high data-rate sensing, this is not necessarily a restriction. Therefore, for high data-rate sensing systems that require 2D or 3D localisation up to several hundred metres, acoustic localisation provides a viable, accurate and immediate solution.

References

Aguayo, D., Bicket, J., Biswas, S., Judd, G. & Morris, R. (2004), Link-level measurements from an 802.11b mesh network, *in* 'SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications', ACM Press, New York, NY, USA, pp. 121–132.

Ali, A. M., Asgari, S., Collier, T., Allen, M., Girod, L., Hudson, R., Yao, K., Taylor, C. & Blumstein, D. T. (2008), 'An empirical study of collaborative acoustic source localization', *Journal of Signal Processing Systems* .

Ali, A. M., Collier, T., Girod, L., Yao, K., Taylor, C. & Blumstein, D. T. (2007), An empirical study of acoustic source localization, *in* 'Proceedings of the Sixth International Conference on Information Processing in Sensor Networks (IPSN '07)', pp. 41–50.

Allen, M., Girod, L. & Estrin, D. (2007), Acoustic laptops as a research enabler, *in* 'The Fourth Workshop on Embedded Networked Sensors (EmNets '07)', pp. 63–67.

Allen, M. P., Graham, E., Ahmadian, S., Ko, T., Yuen, E., Girod, L., Hamilton, M. & Estrin, D. (2008), Interactive environmental sensing: Signal and image processing challenges, *in* 'Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)', pp. 5161–5164.

Anonymous (2002), 'Lmc567 tone-decoder'.
URL: *http://www.national.com/pf/LM/LMC567.html*

Anonymous (2004), 'Cc1000 low power radio tranciever'.
URL: *http://www.chipcon.com/files/CC1000_Data_Sheet_2_1.pdf*

Anonymous (2007*a*), 'Airequalizer white paper'.
URL: *http://www.netequalizer.com/Hidden_Node_White_Paper.php*

Anonymous (2007*b*), 'Baudline'.
URL: *http://www.baudline.com/*

Anonymous (2007*c*), Seismogram transfer program (stp) reference manual v1.01, Technical report, Seismological Laboratory, California Institute of Technology.

Anonymous (2007*d*), 'Why wirelesshart?', HART Communication Foundation White Paper.
URL: *www.hartcomm2.org/hart_protocol/applications/white_papers/why_wirelesshart.html*

Anonymous (2008*a*), 'Archrock corporation'.
URL: *http://www.archrock.com*

Anonymous (2008*b*), 'Cornell lab of ornithology'.
URL: *http://www.birds.cornell.edu/Raven*

Anonymous (2008*c*), 'Engineering design'.
URL: *http://www.engdes.com/*

Anonymous (2008*d*), 'Sensinode'.
URL: *http://www.sensinode.com*

Anonymous (2009*a*), 'Alt-1000/4000 altimeter'.
URL: *http://www.rockwellcollins.com/ecat/BR/ALT-1000_4000.html*

Anonymous (2009*b*), 'Crossbow technology inc.'.
URL: *http://www.xbow.com*

Anonymous (2009*c*), 'Dust networks'.
URL: *http://www.dustnetworks.com*

Anonymous (2009*d*), 'Hilti pd38'.
URL: *http://www.us.hilti.com/*

Anonymous (2009*e*), 'Lv-maxsonar-ez4'.
URL: *http://www.maxbotix.com/uploads/LV-MaxSonar-EZ4-Datasheet.pdf*

Anonymous (2009*f*), 'Matlab'.
URL: *www.mathworks.com/products/matlab/*

Anonymous (2009*g*), 'Sentilla corporation'.
URL: *http://www.sentilla.com*

Arora, A., Ramnath, R., Ertin, E., Sinha, P., Bapat, S., Naik, V., Kulathumani, V., Zhang, H., Cao, H., Sridharan, M., Kumar, S., Seddon, N., Anderson, C., Herman, T., Trivedi, N., Zhang, C., Nesterenko, M., Shah, R., Kulkarni, S., Aramugam, M., Wang, L., Gouda, M., Choi, Y., Culler, D., Dutta, P., Sharp, C., Tolle, G., Grimmer, M., Ferriera, B. & Parker, K. (2005), Exscal: Elements of an extreme scale wireless sensor network, *in* 'The 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2005)', pp. 102–108.

Ash, J. N. & Moses, R. L. (2005), 'Acoustic time delay estimation and sensor network self-localization: Experimental results', *Journal of the Acoustical Society of America* **118**(2), 841–850.

Balogh, G., Ledeczi, A., Maroti, M. & Simon, G. (2005), Time of arrival data fusion for source localization, *in* 'Workshop on Information Fusion and Dissemination in Wireless Sensor Networks, (SENSORFUSION 2005)', Visegrad, Hungary.

Barton-Sweeney, A., Lymberopoulos, D. & Savvides, A. (2006), Sensor localization and camera calibration in distributed camera sensor networks, *in* 'The 3rd International Conference on Broadband Communications, Networks, and Systems (BROADNETS 2006)'.

Bentley, J. (1986), *Programming Pearls*, ACM, New York, NY, USA.

Birchfield, S. T. (2004), A unifying framework for acoustic localization, *in* 'in Proceedings of the 12th European Signal Processing Conference (EUSIPCO '04)'.

Bohn, D. A. (1988), 'Environmental effects on the speed of sound', *Journal of the Audio Engineering Society* **36**(4).

Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y. & Yu, H. (2000), 'Advances in network simulation', *IEEE Computer* **33**(5), 59–67.

Broxton, M. (2005), Localization and sensing applications in the pushpin computing network, Master's thesis, Massachusetts Institute of Technology.

Bulusu, N., Heidemann, J. & Estrin, D. (2000), 'GPS-less low cost outdoor localization for very small devices', *IEEE Personal Communications* **5**(5), 28–34.

Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S. & Srivastava, M. B. (2006), Participatory sensing, *in* 'World Sensor Web Workshop (WSW '06)'.

Capkun, S., Hamdi, M. & Hubaux, J.-P. (2001), Gps-free positioning in mobile ad-hoc networks, *in* 'Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS '01)', IEEE Computer Society, Washington, DC, USA, pp. 9008–9020.

Cerpa, A., Wong, J. L., Potkonjak, M. & Estrin, D. (2005), Statistical model of lossy links in wireless sensor networks, *in* 'Proceedings of the Fourth ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '05)', pp. 11–20.

Chen, J., Yao, K. & Hudson, R. (2002*a*), 'Maximum-likelihood source localization and unknown source localization estimation for wideband signals in the near-field', *IEEE Transactions on Signal Processing* **50**(8), 1843–1854.

Chen, J., Yao, K. & Hudson, R. (2002*b*), 'Source localization and beamforming', *IEEE Signal Processing Magazine* **19**(2), 30–39.

Chintalapudi, K., Fu, T., Paek, J., Kothari, N., Rangwala, S., Caffrey, J., Govindan, R., Johnson, E. & Masri, S. (2006), 'Monitoring civil structures with a wireless sensor network', *IEEE Internet Computing* **10**(2), 26–34.

Collier, T. (2008), Private communication, CENS, UCLA.

Couto, D. S. J. D., Aguayo, D., Bicket, J. & Morris, R. (2003), A high-throughput path metric for multi-hop wireless routing, *in* 'Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom '03)', ACM, New York, NY, USA, pp. 134–146.

Dana, P. H. (1994), 'Global positioning system overview'.
URL: *http://www.colorado.edu/geography/gcraft/notes/gps/gps.html*

Driscoll, D. A. (1998), 'Counts of calling males as estimates of population size in the endangered frogs geocrinia alba and g. vitellina', *Journal of Herpetology* **32**(4), 475–481.

Efrat, A., Erten, C., Forrester, D., Iyer, A. & Kobourov, S. (2006), Force-directed approaches to sensor localization, *in* 'In Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX)', pp. 108–118.

Eker, J., Janneck, J. W., Lee, E. A., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S. & Xiong, Y. (2003), 'Taming heterogeneity—the ptolemy approach', *Proceedings of the IEEE* **91**(2).

Elson, J. (2002), *FUSD: Framework for User Space Devices*, Center for Embedded Networked Sensing. **URL:** *http://www.circlemud.org/ jelson/software/fusd*

Elson, J., Girod, L. & Estrin, D. (2002), Fine-grained network time synchronization using reference broadcasts, *in* 'Proceedings of the 5th symposium on Operating systems design and implementation (OSDI '02)', Boston, MA, pp. 147–163.

Flanagan, B. P. (2007), Self localization using a modulated acoustic chirp, Technical report, The MITRE Corporation, VA.

Fu, Z., Zerfos, P., Luo, H., Lu, S., Zhang, L. & Gerla, M. (2003), The impact of multihop wireless channel on tcp throughput and loss, *in* 'Proceedings of the 22nd International Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM '03)', Vol. 4, pp. 1744–1753.

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E. & Culler, D. (2003), The nesc language: A holistic approach to networked embedded systems, *in* 'Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation (PLDI '03)', ACM Press, pp. 1–11.

George, J. C., Zeh, J., Suydam, R. & Clark., C. (2004), 'Abundance and population trend (1978-2001) of the western arctic bowhead whales surveyed near barrow, alaska', *Marine Mammal Science* **20**, 755–773.

Girod, L. (2005), A Self-Calibrating System of Distributed Acoustic Arrays, PhD thesis, Univerity of Caliornia at Los Angeles.

Girod, L. (2008), Private communication, CENS, UCLA.

Girod, L., Elson, J., Cerpa, A., Stathopoulos, T., Ramanathan, N. & Estrin, D. (2004), Emstar: a software environment for developing and deploying wireless sensor networks, *in* 'Proceedings of the 2004 USENIX Technical Conference', pp. 283–296.

Girod, L. & Estrin, D. (2001), Robust range estimation using acoustic and multimodal sensing, *in* 'International Conference on Intelligent Robots and Systems'.

Girod, L., Jamieson, K., Mei, Y., Newton, R., Rost, S., Thiagarajan, A., Balakrishnan, H. & Madden, S. (2007), The case for WaveScope: A signal-oriented data stream management system (position paper), *in* 'Proceedings of Third Biennial Conference on Innovative Data Systems Research (CIDR07)'.

Girod, L., Lukac, M., Trifa, V. & Estrin, D. (2006), The design and implementation of a self-calibrating distributed acoustic sensing platform, *in* 'Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)', pp. 71–84.

Girod, L., Mei, Y., Newton, R., Rost, S., Thiagarajan, A., Balakrishnan, H. & Madden, S. (2008), Xstream: A signal-oriented data stream management system, *in* 'Proceedings of the International Conference on Data Engineering (ICDE08)', pp. 1180–1189.

Girod, L., Ramanathan, N., Elson, J., Stathopoulos, T., Lukac, M. & Estrin, D. (2007), 'Emstar: a software environment for developing and deploying heterogeneous sensor-actuator networks', *ACM Transactions on Sensor Networks* **3**(3), 13.

Gnawali, O., Greenstein, B., Jang, K.-Y., Joki, A., Paek, J., Vieira, M., Estrin, D., Govindan, R. & Kohler, E. (2006), The tenet architecture for tiered sensor networks, *in* 'Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)', pp. 153–166.

Gotsman, C. & Koren, Y. (2004), Distributed graph layout for sensor networks, *in* 'Graph Drawing', pp. 273–284.

Greenstein, B., Kohler, E. & Estrin, D. (2004), A sensor network application construction kit (snack), *in* 'Proceedings of the Second International Conference on Embedded Networked Sensor Systems (SenSys '04)', ACM Press, pp. 69–80.

Greenstein, B., Mar, C., Pesterev, A., Farshchi, S., Kohler, E., Judy, J. & Estrin, D. (2006), Capturing high-frequency phenomena using a bandwidth-limited sensor network, *in* 'Proceedings of the 4th international conference on Embedded Networked Sensor Systems (SenSys '06)', pp. 279–292.

Gummadi, R., Gnawali, O. & Govindan, R. (2005), Macro-programming wireless sensor networks using kairos, *in* 'International Conference on Distributed Computing in Sensor Systems (DCOSS '05)', pp. 126–140.

Haenselmann, T., King, T., Busse, M., Effelsberg, W. & Fuchs, M. (2007), 'Scriptable sensor network based home-automation', *Emerging Directions in Embedded and Ubiquitous Computing* pp. 579–591.

He, T., Huang, C., Blum, B. M., Stankovic, J. A. & Abdelzaher, T. F. (2003), Range-free localization schemes for large scale sensor networks, *in* 'Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom '03)', pp. 81–95.

Hill, J. (2003), System Architecture for Wireless Sensor Networks, PhD thesis, University of California at Berkeley.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. & Pister, K. (2000), System architecture directions for networked sensors, *in* 'Proceedings of the Ninth International Conference on Arhitectural Support for Programming Languages and Operating Systems (ASPLOS-IX)', Cambridge, MA, USA, pp. 93–104.

Hobson, K. A., Rempel, R. S., Greenwood, H., Turnbull, B. & Wilgenburg, S. L. V. (2002), 'Acoustic surveys of birds using electronic recordings: New potential from an omnidirectional microphone system', *Wildlife Society Bulletin* **30**(3), 709–720.

Hosom, J., Cosi, P. & Cole, R. (1998), Evaluation and integration of neural-network training techniques for continuous digit recognition, *in* 'In Proceedings of the International Conference on Spoken Language Processing (ICSLP '98)', Vol. 3, pp. 731–734.

Hu, C., Xu, S. & Yang, X. (2002), 'A review on interval computation - software and applications', *International Journal of Computational and Numerical Analysis and Applications* **1**(2), 149–162.

Hu, W., Tran, V. N., Bulusu, N., tung Chou, C., Jha, S. & Taylor, A. (2005), The design and evaluation of a hybrid sensor network for cane-toad monitoring, *in* 'Proceedings of the 4th Information Processing in Sensor Netwotks (IPSN '05)', pp. 503–508.

Ji, X. & Zha, H. (2004), Sensor positioning in wireless ad-hoc networks with multidimensional scaling, *in* 'IEEE Conference on Computer Communications (INFOCOM '04)', pp. 2652–2661.

Johnson, D. B., Maltz, D. A. & Broch, J. (2001), *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*, Addison-Wesley, chapter 5, pp. 139–172.

Karp, R., Elson, J., Estrin, D. & Schenker, S. (2003), Optimal and global time synchronization in sensornets, Technical Report CENS-0012, Center for Embedded Networked Sensing.

Kendall, D. G. (1989), 'A survey of the statistical theory of shape', *Statistical Science* **4**(2), 87–99.

Khakpour, K. & Shenassa, M. (2008), Industrial control using wireless sensor networks, *in* 'In Proceedings of the 3rd Information and Communication Technologies: From Theory to Applications', pp. 1–5.

Kim, Y., Schmid, T., Charbiwala, Z., Friedman, J. & Srivastava, M. B. (2008), Nawms: nonintrusive autonomous water monitoring system, *in* 'Proceedings of the 6th International Conference on Embedded Networked Sensor Systems (SenSys '08)', pp. 309–322.

Kushwaha, M., Amundson, I., Volgyesi, P., Parvez, A., Simon, G., Koutsoukos, X., Ledeczi, A. & Sastry, S. (2008), Multi-modal target tracking using heterogeneous sensor networks, *in* '17th International Conference on Computer Communications and Networks (ICCCN 08)', St. Thomas, Virgin Islands (USA), pp. 1–8.

Kwon, O.-H. & Song, H.-J. (2008), 'Localization through map stitching in wireless sensor networks', *IEEE Transactions on Parallel Distributed System* **19**(1), 93–105.

Kwon, Y., Mechitov, K., Sundresh, S., Kim, W. & Agha, G. (2005), Resilient localization for sensor networks in outdoor environments, *in* 'IEEE International Conference on Distributed Computing Systems (ICDCS05)', pp. 643–652.

Langendoen, K., Baggio, A. & Visser, O. (2006), Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture, *in* 'IEEE International Parallel and Distributed Processing Symposium (IPDPS '06)'.

Langendoen, K. & Reijers, N. (2003), 'Distributed localization in wireless sensor networks: a quantitative comparison', *Computer Networks, special issue on Wireless Sensor Networks* **43**, 499–518.

Lanzisera, S., Lin, D. T. & Pister, K. S. (2006), Rf time of flight ranging for wireless sensor network localization, *in* 'Fourth Workshop on Intelligent Solutions in Embedded Systems', pp. 1–12.

Lanzisera, S. & Pister, K. S. (2008), Burst mode two-way ranging with cramer-rao bound noise performance, *in* 'IEEE Global Communications Conference (GLOBECOM)', pp. 1–5.

Ledeczi, A., Nadas, A., Volgyesi, P., Balogh, G., Kusy, B., Sallai, J., Pap, G., Dora, S., Molnar, K., Maroti, M. & Simon, G. (2005), 'Countersniper system for urban warfare', *ACM Transactions on Sensor Networks* **1**(2), 153–177.

Levis, P., Lee, N., Welsh, M. & Culler, D. (2003), Tossim: Accurate and scalable simulations of entire tinyos applications, *in* 'Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)', pp. 126–137.

Lim, Y.-s., Lim, S., Choi, J., Cho, S., Kim, C.-k., Hu, Y.-W. L. H., Zhang, H., Hu, H., Xu, B., Li, J. & Ma, A. (2007), A fire detection and rescue support framework with wireless sensor networks, *in* 'ICCIT '07: Proceedings of the 2007 International Conference on Convergence Information Technology', pp. 135–139.

Lukac, M., Davis, P., Clayton, R. & Estrin, D. (2009), Recovering temporal integrity with data driven time synchronization, *in* 'Proceedings of the 8th international conference on Information Processing in Sensor Networks (IPSN '09)'.

Lundgren, H., Nordström, E. & Tschudin, C. (2002), Coping with communication gray zones in ieee 802.11b based ad hoc networks, *in* 'WOWMOM '02: Proceedings of the 5th ACM international workshop on Wireless mobile multimedia', ACM, New York, NY, USA, pp. 49–55.

Luo, L., Cao, Q., Huang, C., Abdelzaher, T., Stankovic, J. A. & Ward, M. (2007), Enviromic: Towards cooperative storage and retrieval in audio sensor networks, *in* 'Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07)', p. 34.

Lymberopoulos, D., Lindsey, Q. & Savvides, A. (2006), An empirical characterization of radio signal strength variability in 3-d ieee 802.15.4 networks using monopole antennas, *in* 'Proceedings of the Third European Workshop on Wireless Sensor Networks, EWSN (EWSN '06)', pp. 326–341.

Madden, S. R., Franklin, M. J., Hellerstein, J. M. & Hong, W. (2005), 'Tinydb: an acquisitional query processing system for sensor networks', *ACM Transactions on Database Systems* **30**(1), 122–173.

Malan, D., Fulford-jones, T., Welsh, M. & Moulton, S. (2004), Codeblue: An ad hoc sensor network infrastructure for emergency medical care, *in* 'In International Workshop on Wearable and Implantable Body Sensor Networks'.

Maroti, M., Kusy, B., Simon, G. & Ledeczi, A. (2004), The flooding time synchronization protocol, *in* 'Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys04)', ACM, pp. 39–49.

Maroti, M., Volgyesi, P., Dora, S., Kusy, B., Nadas, A., Ledeczi, A., Balogh, G. & Molnar, K. (2005), Radio interfermetric geolocation, *in* 'Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05)', ACM, pp. 1–12.

Mautz, R., Ochiend, W. Y., Brodin, G. & Kemp, A. (2007), '3d wireless network localization from inconsistent distance observations', *Ad Hoc and Sensor Wireless Networks* **3**(2-3), 141–170.

May, A., Mitchell, V., Piper, J., Hanna, L., Hailes, S. & Koumpis, K. (2007), Opportunities and challenges for configurable sensor networks for enabling effective fire-in-tunnel response, *in* 'Proceedings of the International Emergency Management Society Conference', Trogir, Croatia.

Mazzoni, D. & Dannenberg, R. (2007), 'Audacity'.
URL: *http://audacity.sourceforge.net/*

McGregor, P. K., Peake, T. M. & Gilbert, G. (2000), Communication behavior and conservation, *in* L. M. Gosling & W. J. Sutherland, eds, 'Behaviour and conservation', Cambridge: Cambridge University Press, pp. 261–280.

McIntire, D. (2003), 'Energy benefits of 32-bit microprocessor wireless sensing systems', Sensoria Corporation White Paper.

Mellinger, D. K. (2001), Ishmael 1.0 user's guide, Technical report, NOAA Tech. Report OAR-PMEL-120.

Moh, W., Yao, D. & Makki, K. (1998), Wireless lan: study of hidden-terminal effect and multimedia support, *in* 'Proceedings of the 7th International Conference on Computer Communications and Networks', pp. 422–431.

Moore, D., Leonard, J., Rus, D. & Teller, S. (2004), Robust distributed network localization with noisy range measurements, *in* 'SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems', pp. 50–61.

Nagpal, R., Shrobe, H. E. & Bachrach, J. (2003), Organizing a global coordinate system from local information on an ad hoc sensor network, *in* 'Proceedings of the Second ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '03)', pp. 333–348.

Newton, R., Morrisett, G. & Welsh, M. (2007), The regiment macroprogramming system, *in* 'Proceedings of the 6th international conference on Information processing in sensor networks (IPSN '07)', pp. 489–498.

Niculescu, D. & Nath, B. (2001), Ad hoc positioning system (aps), *in* 'IEEE Global Communications Conference (GLOBECOM '01)', pp. 2926–2931.

Patwari, N., Ash, J., Kyperountas, S., III, A. O. H., Moses, R. & Correal, N. S. (2005), 'Locating the nodes: cooperative localization in wireless sensor networks', *Signal Processing Magazine, IEEE* **22**, 54–69.

Payne, K., Thompson, M. & Kramer, L. (2003), 'Elephant calling patterns as indicators of group size and composition: The basis for an acoustic monitoring system', *African Journal of Ecology* **41**(1), 99–107.

Peng, C., Shen, G., Zhang, Y., Li, Y. & Tan, K. (2007), Beepbeep: a high accuracy acoustic ranging system using cots mobile devices, *in* 'Proceedings of the 5th international conference on Embedded networked sensor systems (SenSys '07)', pp. 1–14.

Polastre, J., Szewczyk, R. & Culler, D. E. (2005), Telos: enabling ultra-low power wireless research., *in* 'Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)', pp. 364–369.

Pon, R., Batalin, M., Gordon, J., Kansal, A., Liu, D., Rahimi, M., Shirachi, L., Yu, Y., Hansen, M., Kaiser, W. J., Srivastava, M., Sukhatme, G. & Estrin, D. (2005), Networked infomechanical systems: a mobile embedded networked sensor platform, *in* 'Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)', p. 50.

Priyantha, N. B., Balakrishnan, H., Demaine, E. D. & Teller, S. J. (2003), Anchor-free distributed localization in sensor networks, *in* 'Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)', pp. 340–341.

Priyantha, N. B., Balakrishnan, H., Demaine, E. D. & Teller, S. J. (2005), Mobile-assisted localization in wireless sensor networks, *in* 'IEEE Conference on Computer Communications (INFOCOM '05)', pp. 172–183.

Priyantha, N. B., Chakraborty, A. & Balakrishnan, H. (2000), The cricket location support system, *in* 'Proceedings of the 6th annual international conference on Mobile Computing and Networking (MobiCom '00)', pp. 32–43.

Priyantha, N. B., Miu, A. K. L., Balakrishnan, H. & Teller, S. J. (2001), The cricket compass for context-aware mobile applications, *in* 'The 7th ACM Conference on Mobile Computing and Networking (MOBICOM '01)', pp. 1–14.

Raman, B. & Chebrolu, K. (2008), 'Censor networks: A critique of sensor networks from a systems perspective', *ACM SIGCOMM Computer Communication Review (CCR 2008)* pp. 75–78.

Ramanathan, N., Balzano, L., Burt, M., Estrin, D., Kohler, E., Harmon, T., Harvey, C., Jay, J., Rothenberg, S. & Srivastava, M. (2006), Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks, Technical Report 62, Center for Embedded Networked Sensing, the University of California, Los Angeles.

Reichenbach, F., Koch, M. & Timmermann, D. (2006), Closer to reality - simulating localization algorithms considering defective observations in wireless sensor networks, *in* 'Proceedings of the 3rd Workshop on Positioning, Navigation and Communication (WPNC'06)', pp. 59–65.

Romer, K. (2005), Time Synchronization and Localization in Sensor Networks, PhD thesis, ETH Zurich, Switzerland.

Sallai, J., Balogh, G., Maroti, M., Ledeczi, A. & Kusy, B. (2004), Acoustic ranging in resource-constrained sensor networks, Technical Report ISIS-04-504, Institute for Software Integrated Systems.

Savarese, C. & Rabaey, J. M. (2001), Locationing in distributed ad-hoc wireless sensor networks, *in* 'IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)', pp. 2037–2040.

Savarese, C., Rabaey, J. M. & Langendoen, K. (2002), Robust positioning algorithms for distributed ad-hoc wireless sensor networks, *in* 'USENIX Annual Technical Conference, General Track', pp. 317–327.

Savvides, A., Garber, W., Adlakha, S., Moses, R. & Srivastava, M. (2003), On the error characteristics of multihop node localization in ad-hoc sensor networks, *in* 'Proceedings of the Second ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '03)', pp. 317–332.

Savvides, A., Han, C.-C. & Strivastava, M. B. (2001), Dynamic fine-grained localization in ad-hoc networks of sensors, *in* 'Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom '01)', pp. 166–179.

Savvides, A., Park, H. & Srivastava, M. B. (2003), 'The n-hop multilateration primitive for node localization problems', *Mobile Networks and Applications* **4**(8), 443–451.

Savvides, A., Srivastava, M., Girod, L. & Estrin, D. (2004), *Wireless sensor networks*, Kluwer Academic Publishers, Norwell, MA, USA, chapter Localization in sensor networks, pp. 327–349.

Scherba, D. & Bajcsy, P. (2004), Depth estimation by fusing stereo and wireless sensor locations, Technical report, Automated Learning Group, University of Illinois at Urbana-Champaign.

Shang, Y. & Ruml, W. (2004), Improved mds-based localization, *in* 'IEEE Conference on Computer Communications (INFOCOM '04)', Vol. 4, pp. 2640–2651.

Shang, Y., Ruml, W., Zhang, Y. & Fromherz, M. (2003), Localization from mere connectivity, *in* 'Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (MobiHoc '03)', pp. 201–212.

Simon, G., Maróti, M., Ákos Lédeczi, Balogh, G., Kusy, B., Nádas, A., Pap, G., Sallai, J. & Frampton, K. (2004), Sensor network-based countersniper system, *in* 'Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)', pp. 1–12.

Slijepcevic, S., Megerian, S. & Potkonjak, M. (2002), 'Location errors in wireless embedded sensor networks: Sources, models, and effects on applications', *Mobile Computing and Communications Review* **6**(3), 67–78.

Stathopoulos, T. (2006), Exploiting Heterogeneity for Routing in Wireless Sensor Networks, PhD thesis, University of California at Los Angeles.

Stoianov, I., Nachman, L., Madden, S. & Tokmouline, T. (2007), Pipenet: a wireless sensor network for pipeline monitoring, *in* 'Proceedings of the 6th international conference on Information processing in sensor networks (IPSN '07)', pp. 264–273.

Stoleru, R., He, T. & Stankovic, J. A. (2004), Walking gps: A practical solution for localization in manually deployed wireless sensor networks, *in* 'The 29th Annual IEEE International Conference on Local Computer Networks (LCN '04)', pp. 480–489.

Titzer, B., Lee, D. K. & Palsberg, J. (2005), Avrora: scalable sensor network simulation with precise timing, *in* 'Proceedings of the 4th international conference on Information processing in sensor networks (IPSN '05)', pp. 477–482.

Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D. & Hong, W. (2005), A macroscope in the redwoods, *in* 'Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05)', pp. 51–63.

Trifa, V. (2006), A framework for bird songs detection, recognition and localization using acoustic sensor networks, Master's thesis, École Polytechnique Fédérale de Lausanne.

Trifa, V., Girod, L., Collier, T., Blumstein, D. T. & Taylor, C. E. (2007), Automated wildlife monitoring using self-configuring sensor networks deployed in natural habitats, *in* 'The 12th International Symposium on Artificial Life and Robotics (AROB)'.

Trifa, V., Kirschel, A. N. G., Taylor, C. E. & Vallejo, E. E. (2008), 'Automated species recognition of antbirds in a mexican rainforest using hidden markov models', *Journal of the Acoustical Society of America* **123**(4), 2424–2431.

Tung, T.-L., Yao, K., Chen, D., Hudson, R. & Reed, C. W. (1999), Source localization and spatial filtering using wideband music and maximum power beamforming for multimedia applications, *in* 'IEEE Workshop on Signal Processing Systems (SiPS '99)', pp. 625–634.

Varga, A. (2001), The omnet++ discrete event simulation system, *in* 'In the Proceedings of the European Simulation Multiconference (ESM'2001)', pp. 319–324.

Virone, G., Doan, T., Wood, A. & Stankovic, J. A. (2007), Dynamic privacy in assisted living and home health care, *in* 'Joint Workshop On High Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability'.

Volgyesi, P., Balogh, G., Nadas, A., Nash, C. & Ledeczi, A. (2007), Shooter localization and weapon classification with soldier wearable networked sensors, Technical Report TR-07-802, ISIS group, Vanderbilt University.

Ward, A., Jones, A. & Hopper, A. (1997), 'A new location technique for the active office', *IEEE Personal Communications* **4**, 42–47.

Wark, T., Crossman, C., Hu, W., Guo, Y., Valencia, P., Sikka, P., Corke, P. I., Lee, C., Henshall, J., Prayaga, K., O'Grady, J., Reed, M. & Fisher, A. (2007), The design and evaluation of a mobile sensor/actuator network for autonomous animal control, *in* 'Proceedings of the 6th international conference on Information processing in sensor networks (IPSN '07)', pp. 206–215.

Warneke, B., Last, M., Liebowitz, B. & Pister, K. S. J. (2001), 'Smart dust: Communicating with a cubic-millimeter computer', *Computer* **34**(1), 44–51.

Werner-Allen, G., Dawson-Haggerty, S. & Welsh, M. (2008), Lance: optimizing high-resolution signal collection in wireless sensor networks, *in* 'Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys '08)', pp. 169–182.

Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J. & Welsh, M. (2006), Fidelity and yield in a volcano monitoring sensor network, *in* '7th USENIX Symposium on Operating Systems Design and Implementation 2006', pp. 381–396.

Whitehouse, K. (2002), The design of calamari: an ad-hoc localization system for sensor networks, Master's thesis, The University of California, Berkeley.

Whitehouse, K. & Culler, D. E. (2006), A robustness analysis of multi-hop ranging-based localization approximations, *in* 'Proceedings of the Ffith ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '06)', pp. 317–325.

Whitehouse, K., Jiang, F., Karlof, C., Woo, A. & Culler, D. (2004), Sensor field localization: a deployment and empirical analysis, Technical report, The University of California, Berkeley.

Whitehouse, K., Karlof, C., Woo, A., Jiang, F. & Culler, D. E. (2005), The effects of ranging noise on multihop localization: an empirical study, *in* 'Proceedings of the Fourth ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '05)', pp. 73–80.

Wilson, J., Bhargava, V., Redfern, A. & Wright, P. (2007), A wireless sensor network and incident command interface for urban firefighting, *in* 'Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous '07)', pp. 1–7.

Wittenburg, G., Terfloth, K., Villafuerte, F. L., Naumowicz, T., Ritter, H. & Schiller, J. H. (2007), Fence monitoring - experimental evaluation of a use case for wireless sensor networks, *in* 'Proceedings of the Fourth European Workshop on Wireless Sensor Networks (EWSN 2007)', pp. 163–178.

Wu, W., Au, L., Jordan, B., Stathopoulos, T., Batalin, M., Kaiser, W., Vahdatpour, A., Sarrafzadeh, M., Fang, M. & Chodosh, J. (2008), The smartcane system: an assistive device for geriatrics, *in* 'BodyNets '08: Proceedings of the ICST 3rd international conference on Body area networks', ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, pp. 1–4.

Xu, N., Rangwala, S., Chintalapudi, K. K., Ganesan, D., Broad, A., Govindan, R. & Estrin, D. (2004), A wireless sensor network for structural monitoring, *in* 'Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04)', ACM, New York, NY, USA, pp. 13–24.

Yao, K., Hudson, R., Reed, C., Chen, D. & Lorenzelli, F. (1998), 'Blind beamforming on a randomly distributed sensor array system', *IEEE JSAC* **16**(8), 1555–1567.

Zhang, J., Yan, T., Stankovi, J. A. & Son, S. H. (2007), 'Thunder: towards practical, zero cost acoustic localization for outdoor wireless sensor networks', *SIGMOBILE Mob. Comput. Commun. Rev.* **11**(1), 15–28.

Zhang, Z. (1994), 'Iterative point matching for registration of free-form curves and surfaces', *Int. J. Comput. Vision* **13**(2), 119–152.

Glossary

ARM A 32-bit reduced instruction set architecture commonly used in embedded devices. 17, 58, 87, 108

beamforming A technique to combine multiple, co-located channels to increase the overall SNR of a signal. 28, 29, 35

EmStar A software framework for writing WSN applications. 6, 53, 54, 84, 89, 95, 99–103, 107, 110, 111, 115, 116, 119, 120, 124, 127, 134

Gumstix A single board ARM-based embedded computer with a 32-bit CPU. v, 58, 59, 78, 88–93, 111, 114, 117, 193, 195

Independent Basic Service Set An ad-hoc network of 802.11 compliant devices. 101

lateration A technique to estimate position of an unknown target using range estimates from three or more points with known location. 24, 28, 36, 39, 40, 43, 44, 59, 60, 63, 71–77, 79, 86, 91, 93, 102, 138, 187, 192

Mica2 An 8-bit microcontroller based platform, also referred to as a mote. v, 13, 14, 19, 24, 26, 35, 41, 58–63, 69–71, 76, 77, 79, 86, 88–93, 192, 193, 195

mote A generic term for describe small, resource constrained WSN platforms such as the Mica2. 9, 13, 17–19, 25, 26, 41, 60–62, 65, 66, 68–70, 72, 88, 187, 188, 192

Procrustes A set of techniques for shape analysis. 50, 85, 86

Pseudo-Independent Basic Service Set A version of Independent Basic Service Set which is specific to the Prism2 chipset. 101, 115

Slauson A 32-bit ARM-based embedded platform. 77, 117

VoxNet A hardware and software platform for distributed acoustic sensing. 4–7, 95, 104–112, 114–116, 119–121, 123–130, 133, 134, 138–143, 145, 151, 152, 155, 166, 167, 169, 171, 182–185, 187–189, 195, 196

Wavescope A stream processing engine for high rate WSNs. 4, 7, 95, 106–110, 112–114, 116, 119, 120, 124, 140

Wavescript The language used to write Wavescope programs. 7, 105–109, 114, 117, 119, 120, 124, 126, 183, 184

x86 A complex instruction set for hardware architectures compatible with the Intel 8086 architecture. 99, 108, 119, 157

Acronyms

ADC Analogue to Digital Converter. 60, 69

AML Approximated maximum likelihood. 29, 98, 99, 101–103, 108, 116, 119, 125, 131, 134, 135, 137–141, 146, 147, 151, 157–162, 167, 168, 170–172, 175, 179, 182–184, 186, 188, 190

ANN Artificial Neural Network. 134

AoA Angle of Arrival. 23, 25–30, 33–35, 43–47, 79, 81, 82, 86, 137, 195

BAR Boundary Alignment Ratio. 50

bps bits per second. 155

CDF Cumulative Density Function. 66, 156

CENS Centre for Embedded Networked Sensing. 5–7, 103, 173

CF Compact Flash. 140

COTS Commercial Off The Shelf. 10, 58, 92, 93

CPU Central Processing Unit. 14, 59, 87, 95, 116–118, 120

dB decibel. 63, 92

dBi decibel isotropic. 156

DCT Discrete Cosine Transform. 134

DFT Discrete Fourier Transform. 29, 134

DoA Direction of arrival. 26–29, 43, 44, 78, 86, 87, 98, 99, 102, 119, 135, 182, 186

DSR Dynamic Source Routing. 7, 115, 128, 167, 173

DTN Delay Tolerant Networking. 112

ENSBox Embedded Networked Sensing Box. v, 6, 28, 32, 33, 35, 43, 44, 50, 58, 59, 77–79, 81, 82, 85–93, 98, 99, 101, 102, 114, 117, 119, 120, 124, 126, 134, 189, 193, 195

ETX Expected number of transmissions to transmit one packet. 167, 174–176, 181

EWMA Exponentially Weighted Moving Average. 32

FFI Foreign Function Interface. 109, 112, 119

FFT Fast Fourier Transform. 31, 32

FPGA Field Programmable Gate Array. 17

FTSP Flooding Time Synchronisation Protocol. 12

FUSD Linux Framework for User-Space Devices. 100, 102

GB Gigabyte. 132

GDoP Geometric Dilution of Precision. 3, 38, 39, 44, 59, 74–77, 93

GER Global Energy Ratio. 49, 50, 84, 85

GHz Gigahertz. 34, 60, 119

GPS Global Positioning System. 1, 20, 24, 38, 40, 43, 58, 101

HMM Hidden Markov Model. 134

HTML Hypertext Markup Language. 103

HTTP Hypertest Transmission Protocol. 103

Hz Hertz. 2, 16, 34, 56

IBSS Independent Basic Service Set. 101

ICP Iterative Closest Points. 50

IP Internet Protocol. 101, 102, 110, 115, 118, 147, 149, 189

IPC Inter-Process Communication. 100, 109, 127

kB Kilobyte. 10, 58, 60, 93, 101, 102, 111, 117, 129, 133, 151–154, 156–159, 163, 165, 168, 173, 175, 176, 180

kbps Kilobits per second. 10, 60, 155

kHz Kilohertz. 3, 16, 17, 20, 34, 56, 60, 62, 69, 93, 101, 103, 114, 117, 132, 133

LAN Local Area Network. 115

LPC Linear Predictive Coding. 134

LS Least Squares. 21, 28, 79, 80

LSS Least Squares Scaling. 41

MAC Media Access Control. 10, 153, 155, 173

MAE Mean Absolute Error. 49, 50, 84, 85

MaxE Maxium Error. 49, 84, 85

MB Megabyte. 14, 58, 78, 87, 88, 118, 119, 163

Mbps Megabits per second. 155

MD Mean Density. 52, 85

MDS Multi-Dimensional Scaling. 41, 44

MFCC Mel-Frequency Cepstral Coefficient. 134

MHz Megahertz. 60, 77, 87, 88

ML Maximum Likelihood. 21, 28, 29, 45, 46

MRR Mean Range Residual. 50, 85

MTU Maximum Transmission Unit. 157

NLLS Non-linear least squares. 43

NLOS Non-line of sight. 36

OS Operating System. 14

PCMCIA Personal Computer Memory Card International Association. 78

PLL Phase locked loop. 61

PN Pseudo-random noise. 34, 35, 90

RAM Random Access Memory. 58–60, 87, 88, 93, 119

RBS Reference Broadcast Synchronisation. 101, 115

RBSH Remote Broadcast Shell. 113

RF Radio Frequency. 23, 24, 26, 34

RMBL Rocky Mountain Biological Laboratory. 123, 124, 139, 155, 158, 162, 163, 165, 169, 172, 173, 175–180

RMS Root Mean Square. 49

Acronyms

RSS Received Signal Strength. 23, 26

RSSI Received Signal Strength Indicator. 26

RToA Relative time of arrival. 23–25, 56, 58, 61–63, 69, 92

RTT Round trip time. 23, 25

SNR Signal to Noise Ratio. 35

TCP Transmission Control Protocol. 101, 102, 110–112, 116, 118, 140, 147, 149–151, 153, 155, 157, 158, 168, 181, 185, 189, 193

TDoA Time difference of arrival. 21, 23, 25, 27, 28, 33, 34, 44, 46, 47, 61

ToA Time of arrival. 23–25, 31–33, 44, 47, 58, 62, 79, 88, 90, 92

ToF Time of flight. 23–25, 30, 33, 35, 43, 58, 60, 61, 90

TWR Two-way ranging. 23, 25, 89, 92

TWTT Two-Way Time Transfer. 24

UDP User Datagram Protocol. 116, 151

WSH Wavescope Shell. 109, 112–114, 125, 131, 139, 152, 156, 173

WSN Wireless sensor network. vi, 1–5, 9–17, 19, 20, 24–26, 30, 35, 36, 39, 40, 45, 47, 48, 50, 52, 55, 57, 58, 60, 77, 78, 88, 95, 99, 100, 105, 120, 130, 133, 141, 145, 146, 189, 191–193