Agent-based decentralised data-acquisition and time-synchronisation in critical healthcare applications

Haghighi, M, Woznowski, P, Zhu, N, Tsimbalo, E, Hannuna, S, Burrows, A, Tan, B, Tao, L & Piechocki, R

Original citation & hyperlink:
Haghighi, M, Woznowski, P, Zhu, N, Tsimbalo, E, Hannuna, S, Burrows, A, Tan, B, Tao, L & Piechocki, R 2016, Agent-based decentralised data-acquisition and time-synchronisation in critical healthcare applications. in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT).*, 7389031, Institute of Electrical and Electronics Engineers Inc., pp. 81-86, 2nd IEEE World Forum on Internet of Things, Milan, Italy, 14-16 December.
https://dx.doi.org/10.1109/WF-IoT.2015.7389031

DOI 10.1109/WF-IoT.2015.7389031
ISBN 978-1-5090-0367-9
ISBN 978-1-5090-0365-5
ISBN 978-1-5090-0366-2

Publisher: IEEE

Agent-based Decentralised Data-acquisition and Time-synchronisation in Critical Healthcare Applications

Mo Haghighi, Pete Woznowski, Ni Zhu, Evgeny Tsimbalo, Sion Hannuna, Alison Burrows,
Bo Tan, Lili Tao, Rob Piechocki
Department of Electrical and Electronic Engineering
University of Bristol

***Abstract*— AAL platforms for health and care purposes are designed to be used by diverse stakeholders, such as household residents, carers and healthcare professionals, with equally diverse needs. Reducing the latency of such systems is key to achieving a positive user experience, as well as enabling appropriate responses to critical incidents. This paper introduces a distributed middleware solution that can improve data-acquisition, run computational algorithms locally and seamlessly deliver notifications to various subscribers. The paper further describes three time-synchronisation methods to achieve correct time-stamping of packets, using a combination of centralised and distributed packet processing.**

***Index Terms*—SPHERE, WSN, Agents, Synchronisation, Decentralised, Multi-threading, healthcare, Sensomax.**

I. Introduction

SPHERE (Sensor Platform for HEalthcare in Residential Environment) aims to develop a multi- purpose, multi-modal AAL (Ambient Assisted Living) platform of home sensors not only to extend the state of the art in a number of technology domains, but to engage with stakeholders across disciplines to illuminate the applications of current and new technologies to emerging health needs.

The aim of AAL technologies is to provide 24/7, reliable health-monitoring service. Health-monitoring covers a wide-range of features such as issuing alerts (medicine reminders), real-time monitoring of non-critical parameters (weight, blood pressure, temperature, etc.) and critical episodes e.g. fall detection, hearth attacks, etc. Therefore, it is crucial that these solutions are not subject to a single point of failure [1].

The SPHERE system architecture in Fig. 1 is made up of three sensor networks: Body, Environmental and Video sensor networks. Each sensor network communicates with the central SPHERE Home Gateway (SHG) via its own dedicated gateway, which we refer to as Access Point throughout this paper. Each part of the network is NTP-synchronised to ensure correct time-stamping of individual data items. The IP-based communication between sensor gateways and the SHG is based on the MQTT protocol. The SHG acts as an MQTT broker, collecting and distributing information captured by individual sensors and relayed by SBG, SVG and SEG gateways (MQTT clients) – detailed system description in [1-2]. To avoid confusion, hereafter SBG and SEG are referred to as the Access Points (AP).

Such architecture is fairly simple to understand and implement, yet has its drawbacks. The bottleneck and a single point of failure (SPOF) is the MQTT broker without which in-network communication and communication with the outside World is not possible.

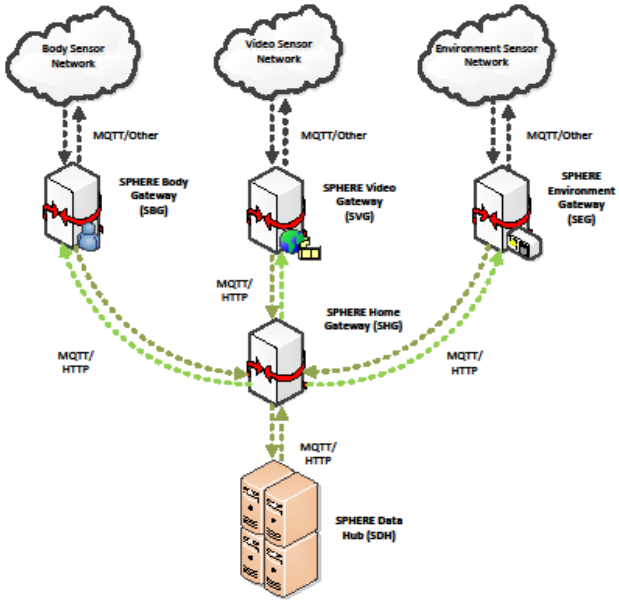

Figure 1: An overview of the SPHERE system architecture.

The MQTT broker (SHG) in the SPHERE system architecture is a single point of failure. Reasons for this can be multiple: insufficient processing power, network bandwidth or simply hardware failure. Since the danger of hardware failures is unpredictable and cannot be influenced let's leave it out of the equation. However, the other two causes can be minimised with a different communication model applied. From the data consumer point of view, the MQTT communication model is on-demand and dynamic. MQTT clients in this pub/sub model can subscribe and unsubscribe to/from a topic of interest and have the information pushed to them whenever new data is available. On the other side, the data producers have to publish their data onto a pre-defined topic to the known (IP and port) MQTT broker. Therefore, this part of the network is static and defined at deployment – although can be re-configured if a need arises. Hence, the MQTT broker is constantly bombarded with data coming from the three sensor networks, even if the data is not 'consumed' by any party. A better-suited communication model would be one, which reacts to the consumers' needs. Why should some information be streamed (and temporarily stored by the MQTT broker) if nobody raised any interest in it? Such approach wastes hardware and network resources, where traffic should be only generated on demand. Reduction of unnecessary traffic offloads the broker and network.

Another improvement applicable to the analysed SPHERE system architecture is to eliminate a single point of failure. Since in the SPHERE architecture all the traffic passes through a single MQTT broker (SHG), a danger exists that if

978-1-5090-0366-2/15/$31.00 ©2015 IEEE

this machine fails, the entire system fails. Since the entire home sensing system is made up of low-power processing units (Raspberry Pies and the NUC), the SHG may become unavailable due to very heavy traffic or may end up working with 100% CPU utilisation and impose delays in serving the real-time sensor data. However, this problem can be addressed since the home sensing infrastructure is made up of multiple processing units (SBG, SVG and SEG), which we refer to as the Access Points, with IP capability and able to serve sensor data.

Access Points (APs) run Sensomax, which is an agent-based WSN middleware, capable of executing multiple concurrent applications based on their required operational paradigm. Its agent-based architecture features:

- Conveying subscribers' requirements to the entities of interest
- Aggregate data on the APs in order to track events of interest
- Issue notifications to the local and external subscribers
- Facilitates decentralized operation amongst all APs in case of SHG failure
- Synchronizing clocks on the APs and time-stamping the packets

As we explained in the beginning of this section, AAL technologies need to be reliable in order to issue alerts of critical parameters. Therefore, it is crucial that these solutions are not subject to a single point of failure. In Sensomax, once the interest is registered (subscribed), the Access Points/individual sensor gateways (SBG and SEG) can push (publish) the required information directly to the interested party, bypassing the SHG. Distribution of roles, where each network segment is responsible for delivering its own information, reduces network traffic and offloads a SPOF machine. In the case of hardware failure, only a given segment is lost and not the whole system.

In Sensomax there exist two types of subscriptions and publications model: *External* and *Internal*.

External model refers to publishing critical healthcare notifications to interested parties, located outside the SPHERE home network, bypassing the SHG when unavailable.

Internal applies to the in-network communication e.g. for the purpose of sensor cueing - where video cameras are to be switched on upon activation of a PIR sensor in the same location. This information can be exchanged between the environmental sensor network and the video sensor network without the involvement of the SHG, thus reducing its load.

Here the environmental sensor network and video monitoring system are described briefly in order to provide a better understanding of how in-network communication between the two through their Access Points can improve their functionalities.

The SPHERE environment sensor system includes ambient temperature, humidity, luminosity, PIR, door contacts, dust particle, noise level sensors, cold/hot water consumption and electricity consumption at each appliance level. As a sub-system of the SPHERE sensor platform, the environment sensor system employs wireless sensor networks (WSN) to network 30+ sensor nodes in mesh topology, generating 90+ data points, distributed in all rooms of a full-instrumented resident house – SPHERE demo house [2].

Sensor nodes provide several fundamental functions such as multiple-channel signal acquisition, on-board signal conditioning, data aggregation etc. Sensed data from the sensor nodes are collected, pre-processed, structured and transmitted to a SHG.

Messages containing the data are sent from the sensor nodes via the WSN in either an event-driven or a time-driven manner. Event-driven messages are sent immediately while sensor detects event or sensor reading passes a designed threshold. time-driven messages, embedding a group of sensor data, are emitted in a required frequency. All sensor data are tagged with sensor name schema, sensor node ID, sensor node location, UTC timestamp and message number to build up context and temporal relationships amongst the entire datasets.

The video monitoring system is currently comprised of 3 Asus XtionPro RGB and depth sensors recording at 30 frames per second. Each is connected via a powered USB 2 cable to a dedicated bus on the SVG machine. These cameras are used to track human movements and return real-time 3d coordinates of bounding boxes encapsulating residents as they go about their daily lives. The use of video helps to provide several advantages compare with other types of sensing devices: they are less intrusive, less expensive, provides rich information of the environments and easier to integrate into already existing buildings, and they are able to simultaneously detect multiple events [9].

In order to save power and reduce the computational burden of the video system it is preferable to only activate the cameras when an individual is within their field of view. To this end, the ability to exploit the reduced latency provided by using Access Points, other than the SHG, offers the opportunity to maximise how much of an individual's actions are captured as they enter a room. A higher latency might mean the system misses the beginning of users' activities and then goes onto record needless extra footage after they have left a particular camera's field of view.

In the Evaluation section, we will measure the communication latency for the local and external subscribes using agent-based model.

In the next section we will describe how using smart agents in Sensomax can help facilitating in-network communication, computation and time-synchronisation between different subsystems.

II. METHODOLOGY

Sensomax has been described in detail in previous publications [3-7]. Sensomax is a component-based, multi-threaded, and dynamic WSN middleware, which is capable of running atop various Java-enabled embedded devices, including Raspberry Pi. Sensomax provides an end-to-end software solution for programming and updating various types of WSN applications onto large-scale distributed networks. It also offers a reliable mechanism for capturing data from multiple subsections of the network independently.

Sensomax features agent-based communications in a multi-clustered fashion in order to abstract network resources and optimise the dataflow. These features are intended to satisfy a number of objectives, including:

- Distributed execution of multiple concurrent applications/data aggregation;
- Adaptive localised behaviour through shifting multiple operational paradigms;
- Centralised/decentralised task distribution and data gathering;
- Time Synchronisation
- Dynamic runtime reorganisation of the network;
- Autonomous decision-making and data aggregation through distributed live computational algorithms.

The fundamental software architecture of Sensomax is modular, meaning that the overall functionalities of the system are broken down into multiple modules, each with a set of unique functionalities. The interaction of these modules creates a dynamic execution environment in which new modules can be created as either standalone components or as a combination of existing ones. Modules' activities are highly embedded, which means that they have no effect on the operations of other independent modules. However, in order to efficiently minimise the overheads of such operations and maximise their performance, they require minimal regulatory strategies, such as controlling their communication domains through a set of agent interaction patterns, as well as abstracting the underling resources according to those patterns, which are imposed by Sensomax.

The primary communication backbone of Sensomax is based conceptually on mobile agents, and nearly all of the operations and processing within the network and the nodes is carried out by 'intelligent' agents. Mobile agents convey data, tasks, and different configuration policies throughout the network in order to inject or execute them in the appropriate nodes. In Sensomax, as with other WSNs, nodes are required to specify their required functionalities to the agents; this is one of the major differences between agent-based approaches in conventional Internet-Protocol-based networks and WSNs. The mobile agent approach originated from a client-server communication paradigm that was developed in conventional computer networking, where servers provide a number of services to their clients. Sensomax, as its first preference, therefore, partitions the network into several clusters to implement a server-client paradigm. Another important motivation behind such an approach is decentralising task-allocation and data-distribution in order to perform asynchronous operations. Utilisation of agents in a clustered fashion helps toward the decoupling of functions and data, which results in accelerated data-distribution and task-allocation amongst network entities.

Sensomax incorporates concurrency in a multi-clustered fashion in which every application is designated with a logical cluster in an abstract form. The application itself resides in a single node, known as the Cluster-Head, where all the top-level executions happen. The Cluster-Head splits the application's requirements according to the above-mentioned categories, and distributes them amongst its members. In order to increase reusability and decrease redundancy, every node can simultaneously maintain multiple roles (operational modes) for different applications, either as a Cluster-Head or a Cluster-Member. Nodes can then switch their operational mode depending on the application being executed. This multi-clustered scheme creates a collaborative execution model with potentially multiple overlapping clusters, and that results in a massive amount of data being generated, either locally in each node, or in a Cluster-Head when data are captured from its several members. On the application side, Sensomax refines application demands into four major categories (Event, Time, Query, and Data requirements), whilst switching the Operational Paradigm (OP) on the node side into one of the Event-Driven, Time-Driven, Query-Driven and Data-Driven paradigms respectively. This distinction between different OPs allows each requirement to be identified and executed in its own paradigm-specific environment with other requirements of the same type. The concurrency model in Sensomax also enables application demands to be split based on their OP and distributed amongst others. This type of distribution can happen in a Cluster-Head when subtasks get allocated to its members. It is worth noting that in every node agents are still executed in their own application spaces within each OP.

Sensomax also abstracts both agents and available resources into three major categories: Global, Local, and System. This process safeguards exclusive interactions of agents and resources, where each type of agent is only privileged to access the resources of its own type. This mechanism benefited Sensomax enormously, resulting in more rapid runtime updates and better scalability. Because of this, multiple applications run simultaneously on both network and node levels whilst injecting their updates at runtime, without affecting others' processing, and they can potentially use and/or reuse the same set of resources.

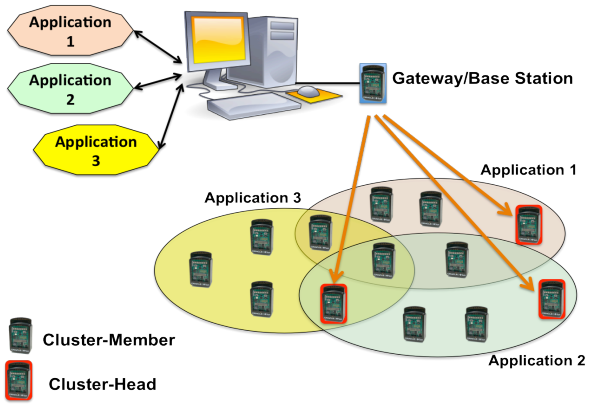

Figure 2: Multiple overlapping clusters for concurrent applications

Figure 2 shows how multiple applications run simultaneously in different cluster-heads whilst sharing the same cluster-members, acting as a cluster-head of one application and a cluster-member of another. In this figure applications are synonymous with subscriber's request, which can run on a cluster-head or in this case on an AP.

Having implemented these adaptive behavioural reactions to application requirements at initial deployment, the next requirement is to ensure that the network is able to dynamically reorganise itself according to the results of updates received from the pre-deployed applications. The decentralised execution modality of Sensomax, using a multi-agent approach, and the isolated executing environment of individual applications, facilitates an exclusive hierarchical agent processing in nodes and clusters.

Updates to pre-deployed applications are received from the end-users and distributed throughout the network in the form of System Agents, which are processed in an

asynchronous fashion in each Cluster-Head. The isolated executing environment of each application guarantees the receipt of updates by the addressees only, and the updates are pushed onto the application subtasks without affecting the normal operations of the nodes. Once all updates have been applied, the network will gradually adapt to the new changes, due to its decentralised interactive nature.

Another distinct benefit of such decentralised adaptive reorganisations is seamless data aggregation at multiple network levels. Algorithmic operations can be carried out on captured data in the same way that the updates are applied dynamically, but with one key difference: algorithms are embedded as computational components within the middleware's architecture. The isolated processing of agents facilitates data aggregation by distributing the collected data in the form of agents and applying the computational algorithms onto them either at Cluster-Heads or right where they are captured, at nodes. This decentralised methodology allows decoupled computational operations in nodes, whilst higher-level aggregation can be done independently in the associated Cluster-Heads. This scheme not only enables aggregation but also is the main principle behind implementing autonomous behaviours in Sensomax.

Examples of computational operations that can be seamlessly incorporated into the Sensomax architecture include: data aggregation algorithms, packet error correction using CRC, energy management strategies, statistical analysis of captured data, and any arithmetical operations on the captured data [6-7, 10]

SBH Access Point, uses BLE transceivers to interact with on-body sensors via USB port. Therefore it is necessary to process the received packets for error correcting.

In line with many communication protocols, BLE uses Cyclic Redundancy Check (CRC) codes to detect communication errors. CRC error correction would directly improve the packet reception rate at the receiver and decrease the amount of energy that is spent on retransmissions at the transmitter [8].

Prior to transmission, a CRC encoder processes the packet. On the receiver side, the received packet is forwarded to the higher layers of stack only if a CRC check is successful, otherwise the packet is considered corrupted and is dropped.

Considering that performing error correction requires additional processing at the receiver, this approach is essentially offering a way to decrease the retransmissions and unburden the transmitter by moving some of these costs to the receiver. Therefore, CRC error correction is a compelling choice for applications where an energy-constraint transmitter communicates with a constraint-free infrastructure, such as wearable sensors streaming data to a smart infrastructure [8].

This process can be implemented by the APs in a distributed and centralised manners using Sensomax agent-based model. Such a process requires to be multi-threading process on the central unit (as well as every access point) in which received data on 3 different channels are passed to a multi-step CRC-checking algorithm by parsing packet's properties including Antenna polarization, Channel Number, Relative Time stamp, Packet Sequence number and RSSI.

According to figure 3, for the purpose of CRC-checking and subscribing local and externals subscribers, Sensomax models the network in three modes. That is done in order to distribute agents, fetching notifications and run computations locally in clusters.

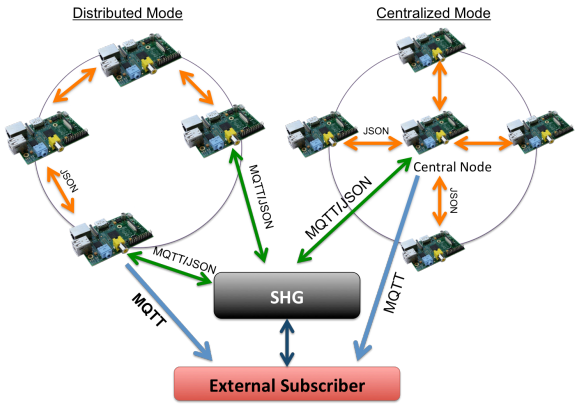

Figure 3: Distributed and Centralised agent-based model.

Distributed: Each access point receives data from the packet sniffer and transmits them in JSON format to the SHG over MQTT. This transmission mode operates regardless of other access points operations.

Centralised: One of the access point acts as a central unit and all other access points connect with the central unit in Star topology. Each access point sends its received data to the central unit in JSON format. The central unit opens an MQTT connection to home gateway and transmit data.

Dual: In this mode, both modes centralised and distributed modes are implemented simultaneously.

Checking and correcting received packets on 3 Channels from 4 APs facilitates more functionalities including:

1. Synchronizing real time clocks (RTC) of all access points;
2. Optional Time-stamping of received data;
3. Parsing Sniffer's received packets to/from JSON; format by adding UTC timestamps as requested;

In the Centralised approach, the clock of each AP is checked, and its time-stamping of packets is corrected by the central node in real-time. The central node maintains an active TCP connection (synchronous) with the SHG's NTP time server. In the Distributed mode, each AP corrects its time-stamping with the SHG with respect to the link latency. In the Dual mode however, both approaches are done sequentially starting with Distributed mode. The next section will evaluate all three synchronisation methods and measure the drift in the time-stamping over time.

In the next section we will also evaluate subscribing local and external subscribers using the centralised and distributed modes. Latencies in both communication methods, and their overhead on the APs' processor and memory will also be measured.

III. CASE STUDIES AND EVALUATION

In this section a number of experiments have been conducted to measure the latencies for delivering the notifications from the source of events to the local and external subscribers for both time-driven and event-driven applications. In the second part of this section we will evaluate different synchronisation mechanisms in terms of the time-drift in access points' clocks over 120 hours.

For the purpose of this experiment, a network of 4 Raspberry Pi devices acting as the access points were set up in 2 topologies of Centralised and Distributed, which have been described in the previous section.

For simplicity, the following applications are devised:

Table 1: Event-driven and Time-driven applications

Application	Operational Paradigm	Parameter	Frequency/ Threshold
A	Event-driven	PIR	200ms
B	Time-driven	Temperature	Every 5 seconds

For the first experiment, a cluster of 4 APs in star topology has been set up, with one AP acting as the central node (cluster-head). Applications A contains event-driven agents, in which any movement detected by the PIR sensor will be delivered to the central AP and flagged to be relayed to the relevant subscriber. The central AP will then decide whether to send the notification to a local subscriber or an external one. The same scenario applies to the application B, in which time-driven agents monitor temperature value at a 5 second interval and relay the data to the central node.

Based on SPHERE Requirements Specification Document, all packets are in JSON format following SenML mark-up language. Packets differ in terms of size and contents. Here is a sample packet for application A and B:

{'uid':'S2_H','bt':'2-45-41','mc':'128','dt':'2015-07-07T16:44:29.513Z','e':[{'n':'LT','v':90.811},{'n':'NS','v':69.0},{'n':'DT','v':0.143}]}

In addition to the above applications, by default the central AP runs the clock-synchronisation service at all time. It time-stamps the incoming packets by taking into account the clock-offset value of the associated node, and the NTP timeserver running on the SPHERE Home Gateway.

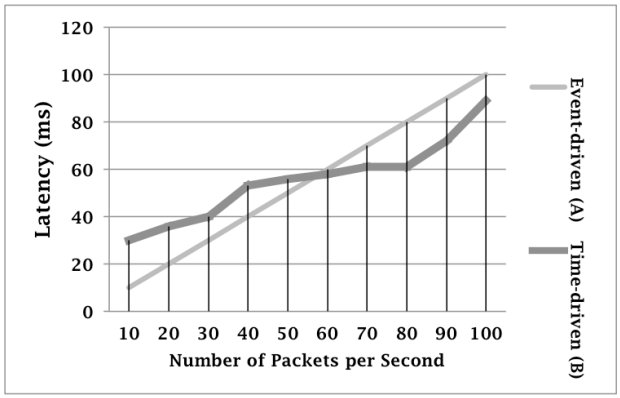

Figure 4: Latencies associated with Event-driven and Time-driven applications delivered to Local subscriber via a central node.

Figure 4 shows the latencies associated with event-driven and time-driven applications delivered to the local subscribers via the central node.

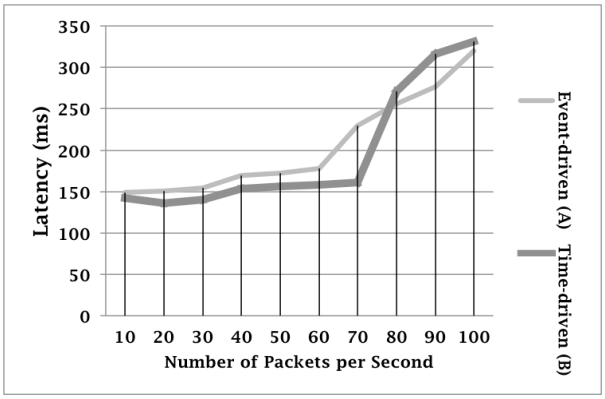

Figure 5: Latencies associated with Event-driven and Time-driven applications delivered to External subscriber via a central node.

Figure 5 shows the latencies associated with event-driven and time-driven applications delivered to external subscribers via the central node using MQTT.

As figure 4 and 5 show, the latency in the delivery of the notifications is proportionate to the number of packets sent to the central AP per second. Therefore connection traffic has a huge impact to the latency of data transfer. As this figure shows, Time-driven application has lower latency, since time-driven agents are expected at regular intervals (in this case every 5 seconds). Event-driven agents however, are unpredictable and only generated when an event of interest is detected (in this case, movement detected by PIR sensor).

The other important factor is the fact that, higher number of packets processed by the central AP imposes a massive overhead on the processor and the memory. Figure 6 shows the processor and memory overheads on the Raspberry Pi version 2 model B with Quad-core 900 MHz Cortex-A7 Processor and 1GB of RAM.

For the purpose of this experiment, Sensomax is the only application running on the raspberry pi with active UDP connections from 4 APs. Sensomax with no data flow occupies 19-20% of the processor when the AP boots up.

As this figure displays, 750 packets per second is the maximum the processor can handle. The system adds significant delay to this process or other system processes once the processor goes past this figure. It will also impose huge pressure on the CPU, which results in overheating and system failure.

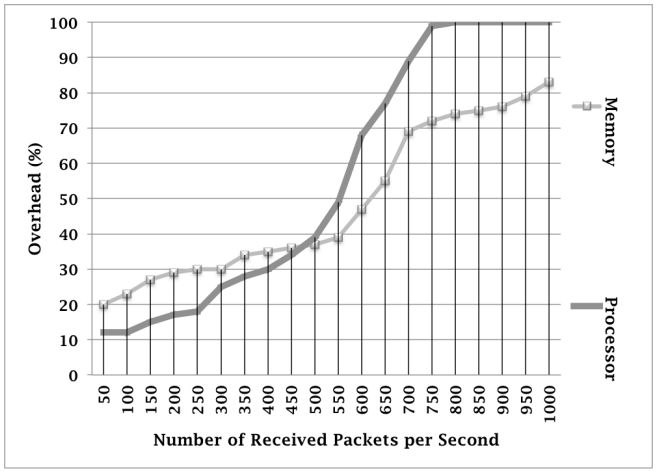

Figure 6:Processor and Memory overheads based on the number of packets received per second.

As this figure shows, the amount of occupied RAM based on the maximum number of packets is around 80%, which is within the healthy range, given the significant amount of JSON parsing done by the AP on the incoming packets. It is worth mentioning that SPHERE Body Gateway configuration requires the APs to run additional Python software, which looks after packet sniffer application provided by Nordic Semiconductor. This software is essential in order to interface with the Nordic USB Receiver. Nordic software normally takes between 30-50% of the processor based on the given load. Based on the above figure, processing more than 550 packets per second in addition to the Nordic library could lead to overheating of the CPU and failure of Raspberry Pi board. Therefore for the purpose of this experiment, Nordic software has been removed from the APs.

Next section repeats the previous experiment, this time in a distributed fashion without the central node. In this scenario APs deliver the notifications to the local subscriber directly (Figure 7) and to the external subscriber via the SHG or directly to a remote IP address via MQTT (Figure 8).

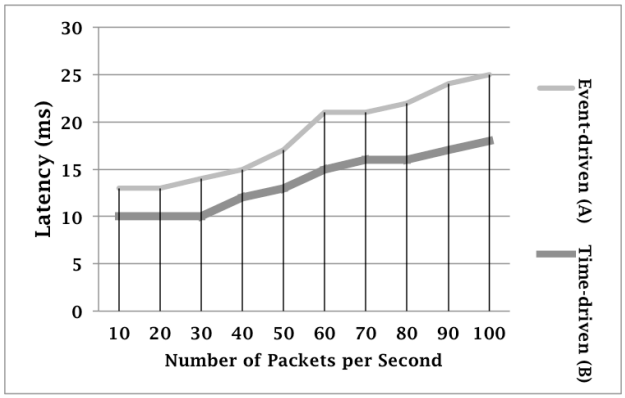

Figure 7: Latencies associated with Event-driven and Time-driven applications delivered to Local subscribers bypassing the SHG

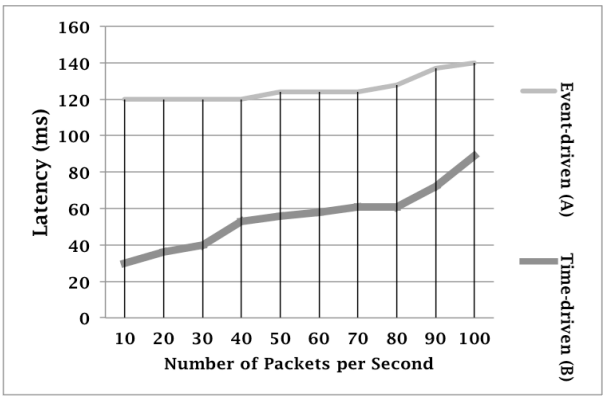

Figure 8: Latencies associated with Event-driven and Time-driven applications delivered to External subscribers via the SHG.

Like the previous experiment, both Figures 7 and 8 show that time-driven applications have lower latency compared to the event-driven ones. However, latency in a distributed mode is significantly lower. The omission of central node notably improves the timely delivery of notifications. However, each AP has to synchronise its clock with the SHG directly and time-stamping is only done on the APs in a distributed fashion.

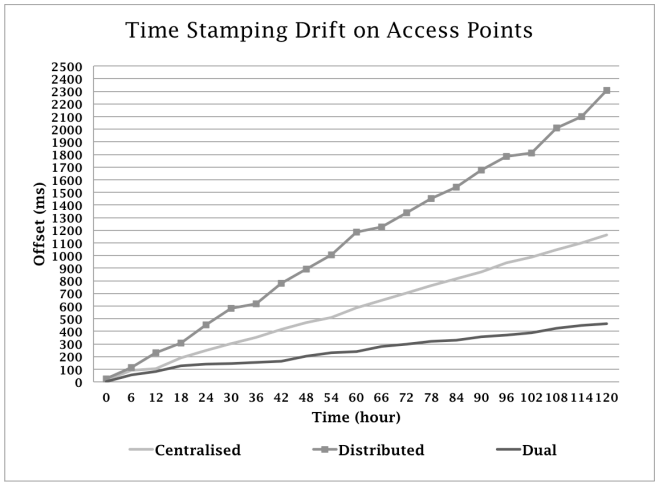

Figure 9: Drift in the time-stamping of the packets using Centralised, Distributed and the Dual modes.

It is also worth noting that this approach eliminates the advantage of decentralised operation of APs in case of SHG failure. As we mentioned in the previous section, another approach for clock-synchronisation is to use both centralised and distributed modes for adjusting the offset in the APs' time-stampings. Figure 9 reports the drift in the time-stamping of the packets over 120 hours using centralised, distributed and the dual mode, which is a combination of both approaches.

As figure 9 shows the drift in the timestamp of distributed mode introduces around 2300ms over 120 hours, which is equal to nearly 2 seconds over 5 days. The offset by the centralised mode is nearly half the distributed mode. The combined mode however, offers the best result; with only 500ms over a 5-day period.

IV. Conclusion

In this paper we have described how distributed middleware can improve data-acquisition and delivering notifications to the subscribers for critical healthcare applications and running local computational process. We also described three time-synchronisation methods in order to correct time-stamping of packets using a combination of centralised and distributed packet processing.

Acknowledgment

This work was performed under the SPHERE IRC, funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant EP/K031910/1.

References

[1] Woznowski, P., Fafoutis, X., Song, T., Hannuna, S., Camplani, M., Tao, L., Paiement, A., Mellios, E., Haghighi, M., Zhu, N., Hilton, G., Damen, D., Burghardt, T., Mirmehdi, M., Piechocki, R. J., Kaleshi, D., Craddock, I., "A multi-modal sensor infrastructure for healthcare in a residential environment." In: In IEEE Workshop on ICT-enabled services and technologies for eHealth and Ambient Assisted Living, London, UK, June 2015.

[2] Zhu. N, Diethe. T, camplani. M, Tao. L, Burrows. A, Twomey. N, Kaleshi. D, Mirmehdi. M, Flach. P, Craddock. I, "Bridging eHealth and the Internet of Things: The SPHERE Project", IEEE Intelligent Systems, special issue May/June 2015.

[3] M. Haghighi, D. Cliff, "Sensomax: An Agent-Based Middleware For Decentralized Dynamic Data-Gathering in Wireless Sensor Networks", The 2013 IEEE International Conference on Collaboration Technologies and Systems, San Diego, May 2013.

[4] M. Haghighi, M. Bocian, O. Oddbjornsson, J.H.G Macdonald, J.F. Burn, "Synchronous Data Acquisition from Large-scale Clustered Wireless Sensor Networks", 10th IEEE Vehicular Technology Society APWCS, Seoul, South Korea, August 2013.

[5] M. Haghighi, "An End-to-End Middleware Solution With Multiple Concurrent Applications Support for Wireless Body Area Networks", The 6th IEEE International Conference on Computational Intelligence and Applications, July 2013, Hiroshima, Japan.

[6] M. Haghighi, C.J. Musselle, "Dynamic Collaborative Change Point Detection in Wireless Sensor Networks", International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Beijing, China, October 2013.

[7] M. Haghighi, "Market-based Resource Allocation For Energy-efficient Execution of Multiple Concurrent Applications In Wireless Sensor Networks", LNEE, Springer, July 2013.

[8] Tsimbalo, E., Fafoutis, X., Piechocki, R. "Fix It, Don't Bin It! - CRC Error Correction in Bluetooth Low Energy", submitted to the 2nd IEEE world forum in IoT.

[9] A. Paiement, L. Tao, S. Hannuna, M. Camplani, D. Damen, and M. Mirmehdi, "Online quality assessment of human movement from skeleton data," in British Machine Vision Conference, 2014.

[10] M. Haghighi, K. Maraslis, T. Tryfonas, G. Oikonomou, "Game Theoretic Approach Towards Energy-Efficient Task Distribution in Wireless Sensor Networks", IEEE SENSORS 2015, Busan, South Korea, November 2015.