Wireless Sensor Network Softwarization: Towards WSN Adaptive QoS

Syarifah Ezdiani, Indrajit S. Acharyya, Sivaramakrishnan Sivakumar, and Adnan Al-Anbuky

***Abstract*—Connecting wireless sensor networks (WSNs) to remote servers or clouds through the Internet of Things inspires the formation of a highly complex system. This has significant potential to encourage WSN softwarization. Such a network enables interaction between the physical and virtual sensor network, allowing for flexibility within the physical system to adapt with the dynamics of the service requirements. An architecture of a test environment with adaptive quality of service (AQoS) has been presented. The concept provides an avenue for a flexible system that is capable of reacting to dynamic changes of process demands. Physical network performance can be predicted by analyzing the historical data in the background on a network simulator or virtual network. This in turn allows for estimation of the necessary adjustments needed to improve the network performance without disturbing the physical system. This paper reports the experimentation and applicability of user-driven AQoS models on the system. Early stages of testing the organization, taking into account the system behavior and subsequent reaction to necessary adjustment of the WSN operational configuration, has shown encouraging results.**

***Index Terms*—Adaptive quality of service (AQoS), Internet of Things (IoT), quality of service (QoS) provisioning, wireless sensor network (WSN), WSN virtualization.**

I. Introduction

THE INTEGRATION of the wireless sensor network (WSN) to the cloud and through the Internet has become one of the prime technologies that bring the Internet of Things (IoT) to reality. Recently, the emerging applications of IoT have brought forward various research efforts within academia and the industry. One of the main challenges in integrating WSNs to the Internet is to ensure reliable traffic flows between both networks, to offer an end-to-end quality of service (QoS).

In many WSN applications, the occurrence of an important event can suddenly generate a burst of data traffic. This is demonstrated in various WSN applications such as intruder detection systems [1] and fire hazards monitoring applications [2]. Typically, the network should make more effort in delivering higher priority packets. This highlights a need for having an adaptive WSN network performance matrix that could follow the demand of the physical process.

Manuscript received February 6, 2017; revised May 5, 2017 and July 27, 2017; accepted August 4, 2017. Date of publication August 29, 2017; date of current version October 9, 2017. *(Corresponding author: Syarifah Ezdiani.)*

The authors are with the Sensor Network and Smart Environment Research Centre, Auckland University of Technology, Auckland 1142, New Zealand (e-mail: ssyednor@aut.ac.nz; iarchary@aut.ac.nz; ssivakum@aut.ac.nz; adnan.anbuky@aut.ac.nz).

Digital Object Identifier 10.1109/JIOT.2017.2740423

For instance, a service differentiation scheme may be featured with adaptive capabilities, as it reacts to process dynamics. It is also important to gain real-time and continuous assessment to the current QoS conditions to ensure the required application-specific QoS is always met. With knowledge of the QoS performance such as queuing delay and traffic drop, informed adaptations can consequently be made to the network. This can be done potentially through reconfigurations of node attributes such as buffer size, service rates (SRs), and bandwidth allocations.

Physical WSN-IoT testbeds [3] offer great benefits to researchers in evaluating WSN design as they gather real-life data from physical settings. However, while testing new algorithms and protocols, the process of experimenting with different scenarios and performing comparisons amongst these scenarios can be quite challenging. In this case, simulators offer capabilities and features that make them favorable for design and tests of new protocols. It is envisaged that better analysis of WSN applications can be facilitated by exploiting the advantages of both virtualization and real-life testbeds. Therefore, we propose an IoT-based WSN test environment which offers interactions between the behavior of the physical environment as it interacts with the phenomenon and the necessary analysis in a virtual remote environment.

An overview of a test environment architecture is presented in [4]. The primary idea is to allow for reconfiguration of the physical sensor nodes in a flexible way, to adapt to the network's QoS condition. The reconfiguration is based on the QoS evaluation by the software which resides in the server, which analyzes the recent historical data. The architecture allows user-driven QoS-related experimentation, and work on a case-by-case basis. As part of our continued investigation, we are looking at the applicability of the models by testing with application-specific WSN QoS parameters. Therefore, in this paper the implementation of a proposed adaptive QoS (AQoS) model which serves as a use case on the test environment is presented.

In Section II that follows, a literature review on adaptive QoS and service differentiation is presented. Next, Section III presents an overview of the AQoS operational architecture of the proposed test environment. This is followed by a discussion on the implementation of the system in Section IV. The experiment cases and results are presented in Section V. Finally, Section VI concludes this paper and highlights on-going and future work.

II. Related Work

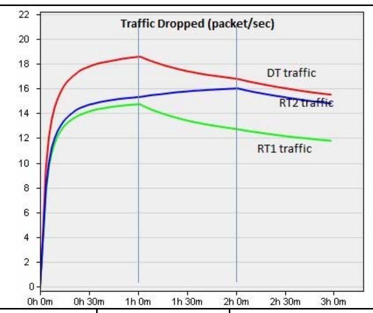
This section begins with a literature review on WSN QoS and the pertaining dynamic aspects which affect WSN QoS. A review of sensor network virtualization is presented next. Finally, the impact of interaction of the WSN with the dynamics of the phenomenon that it monitors is discussed.

In general, traffic priority is categorized by two main domains: 1) timeliness and 2) reliability [5]. Packets with time constraint can only tolerate minimal delay, or may be strictly end-to-end delay bound. On the other hand, packets which need to be sent in a reliable manner can only tolerate a small percentage of loss, or even zero loss.

In a service differentiation [6], [7] approach, adaptive QoS may be used to maintain QoS relative to the dynamics of the sensor network. A related work on an adaptive service differentiation is demonstrated in [8], by adopting congestion control [9] and rate adjustment solutions. The work shows that, using a weighted priority-based rate control scheme, congestion can be controlled by adjusting transmission rates relative to various data types. Node-level congestion (as opposed to link-level congestion) is caused by buffer overflow in the node which will lead to packet loss or an increase in queuing delay. Consequently, packet loss will degrade the application's reliability, reduce link utilization, and waste limited node energy. Basically, buffer overflow may happen as the packet arrival rate (AR) exceeds the packet SR. The overflow is more likely to occur at the sensor nodes close to the sink, or at the sink itself, which carry the collective upstream traffic. Note that in an IoT-WSN application, the sink may also take the role of an IoT access point.

An adaptive system for service differentiation through a fuzzy logic controller has been proposed [10]. It employs a fuzzy logical controller for a traffic load parameter with a priority-based rate in the network. The work shows that an adaptive QoS system can prolong the system lifetime as adjustments to the network can be performed to enhance system performance.

The implementation of virtual sensor networks (VSNs) as a software replicated image of the corresponding physical sensor network (PSN) is demonstrated by Barbato *et al.* [11]. The VSN contains all the metadata of the PSN. Data processing takes place within the virtualized network to cater for the different requirements of the users/clients. This covers for the need for any data storage, processing, or computation on PSN hardware. Large scale IoT testbed implementation of SmartSantander [12] reaps the benefits of virtualization hosted within a cloud infrastructure and provides experimentation and testing facilities to end users. Therefore, an extended function of specialized network modeling tools, such as Riverbed Modeler, within the cloud may facilitate an environment for hosting the QoS controller models on the virtualized network. It serves as a testing environment, i.e., to observe the impact of modifying virtual sensor node parameters such as data rate, SR, buffer size, and other functionalities of network protocol on network performance, prior to implementation on PSN hardware.



Traffic Class	Weightage	Buffer size (packet)	Data rate (kbps)	
			Initial rate	Adjusted rate
RT1	0.5	20	32	16, 8
RT2	0.3	20	32	19.2, 12.8
DT	0.2	30	32	16, 12.8

Fig. 1. Example of AQoS outcome within three adaptive cycles.

The virtualization level within the Lysis platform [13] comprises of abstractions of the functionalities of the real world objects/devices such as smartphones, as well as their social capability. The virtualization layer present within the iCore [14] architecture consists of virtual objects and facilitates for exposure of a virtual interface from the real object (either sensor or actuator). The first approach considerably reduces latency, whereas the second approach results in decoupling of the real objects, thus allowing for reuse of the virtual interface hardware implementation undergoing alteration.

A continuous QoS monitoring calls for a more involved IoT-based sensor network infrastructure that enables the virtualization of the real physical world. With a test environment infrastructure, the AQoS algorithm may be extended to allow informed adjustments on the IoT access point, which is highly desirable to ensure seamless interconnection with the Internet and to provide better balance to the AQoS strategies. With a software-based network modeling that provides virtualization of the real physical world, important performance parameters related to the application QoS requirements can be considered without having to make major changes to the physical setting.

Example test results of an AQoS concept that has been conducted as part of this paper are illustrated by Fig. 1. To demonstrate the continuous monitoring of the QoS condition, an AQoS concept is implemented to react to the data flow dynamics close to WSN applications in a physical world. Fig. 1 shows the performance of three source variables with different priority levels and different QoS requirements.

In this example, RT1, RT2, and DT represent real-time traffic (RT) with high priority, RT with low priority, and delay-tolerant traffic (DT), respectively. The figure depicts the traffic performance during three QoS monitoring cycles. In each cycle, the performance indicator is given by average traffic drop. Based on the traffic drop status, the system's QoS controller reacts by setting a new traffic source data rate. As shown in the figure, although the traffic drop of all traffic types increased in the first cycle, the rate adjustments reduce the traffic drop of DT traffic and RT traffic in the second and third cycle. In summary, the test shows that continuous

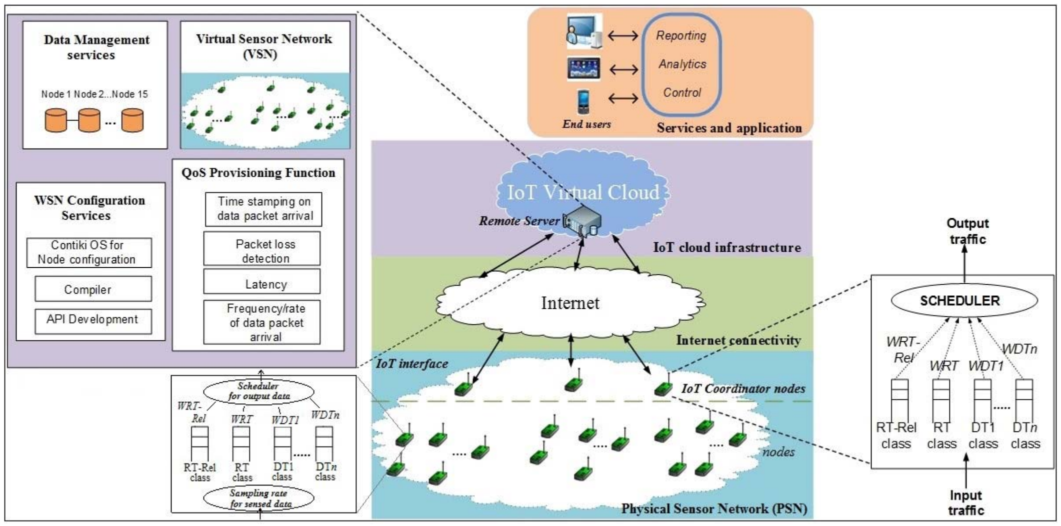


Fig. 2. AQoS testbed architecture.

network improvement and traffic reliability can be achieved through the adaptive approach. The test highlights the impact of the interaction of the WSN with the dynamics of the phenomenon being monitored. Moreover, network virtualization and pretesting features in an IoT-based sensor network can be further exploited with the feature of the software-based [10] QoS controller.

III. AQoS Operational Architecture

An environment for enabling the software adaptation of the IoT-based-WSN is introduced in this section. The environment provides continuous monitoring of a WSN-IoT network performance through assessment of the QoS and makes necessary adjustments to the physical network's configuration. The approach involves modular organization that allows implementation and analysis of QoS for WSN based on historical sensor data captured from the physical network. Results of the analysis may recommend reconfiguration of the PSN parameters that allow maintaining the required QoS.

Fig. 2 depicts the proposed architecture [4], which is established with the PSN, IoT interface, and remote server or cloud support. The PSN may be organized in multiple physical sensor cloud (PSC) formations. Each PSC has an IoT coordinator which serves as the access point between the PSC and the Internet. The remote server or the cloud host the data storage and necessary environment for intelligence, network virtualization, and other processing that support decision making for managing the PSN operation.

Sensor data received by the IoT coordinator is pushed to the cloud server, which accommodates the database for storing related historical data, network simulation tools for modeling the VSN, and a mathematical modeling tool for hosting QoS evaluation models. Furthermore, a QoS controller program, which runs the QoS algorithm is part of the QoS provisioning function. It facilitates analytical activities and suggests adjustments to be made on the sensor nodes' operational parameters to maintain the network's QoS condition. The adjustment is done by means of remote configurations of the physical leaf nodes.

WSN-IoT service models are categorized based on three factors: 1) interactivity; 2) delay; and 3) criticality of the WSN applications [15], [16]. The delay factors are categorized by the nature of the application traffic, namely non real-time, soft real-time (SRT), and hard real-time (HRT). Our use case of adaptive QoS in this paper is part of the many applications that utilizes the *complete service model*, which involves continuous data flow of SRT or HRT traffic and is mission-critical.

Fig. 2 also illustrates a queuing model for regulating the buffer for data packets whilst in queue to get transmitted at the IoT coordinator. Upon arrival at the IoT coordinator the data will be allocated to different buffer queues based on the service differentiation which runs within the IoT coordinator. As shown by the figure, different traffic types are buffered in separate queues in the IoT coordinator.

An example of an AQoS algorithm is shown in Table I. The objective of the QoS model is to avoid congestion in the network by determining the traffic's data rate which is appropriate to the network condition. For instance, congestion detection in WSN has been proposed using indicators such as buffer occupancy, queue length [17], [18], service time of packets [19], as well as the ratio of service time to interarrival time of the packets between the intermediary devices [20].

Without losing generality, the algorithm given in Table I supports two types of traffic classes: 1) RT and 2) DT. In the initialization phase, the algorithm identifies the data rate of the traffic sources and their QoS requirements, namely the RT packet's tolerable delay and the DT packet's tolerable packet loss. The algorithm also estimates the required buffer size [21] of different traffic types based on this information. The required buffer size is estimated to ensure that the reliability-constraint DT packets can take up more space in the IoT coordinator's buffer. Therefore, in the QoS model, more bandwidth is allocated to RT traffic to meet low delay. In addition, more buffer storage is allocated to DT traffic to avoid buffer overflow which may result in higher traffic drop.

TABLE I
AQoS ALGORITHM EXAMPLE

```
Define QoS requirement:  RT packet delay bound, DT packet loss tolerance

Initialization:         Identify data rate RT_datarate, DT_datarate,
                        Set required buffer size RT_buffer, DT_buffer

While TRUE {

Data transmission:      Transmit data array to IoT coordinator

Queue Algorithm…  Arrival and departure of packets in IoT coordinator
buffer using weighted priority queuing scheduling policy

QoS Condition Monitoring and Adjustment (Every T interval)

  - Congestion Control Unit
                        Calculate congestion index
                        congestion_index = service_rate/arrival_rate

  - Rate Adjustment Unit:
                        Adjust new RT data rate based on congestion index
                        if congestion_index>1
                            RT_datarate = new_RT_datarate($_{priority, datarate}$)
                        else if congestion_index <1
                            RT_datarate = new_RT_datarate($_{priority,datarate}$)
                        else
                        Adjust new DT data rate based on congestion index
                         if congestion_index>1
                            DT_datarate = new_DT_datarate($_{priority, datarate}$)
                         else if congestion index <1
                            DT_datarate = new_DT_datarate($_{priority,datarate}$)
                         else

  - Buffer Adjustment Unit:
                         Adjust buffer size based on new data rate
                            RT_buffer ← min (size$_{RT}$)
                            DT_buffer ←min(size$_{DT}$)

OUTPUT:
Write new value      In c header file (to be read by OS for reconfiguration)
                            RT_datarate, DT_datarate
                            RT_buffer, DT_buffer
}
```

The rate adjustment unit calculates the new data rate of the traffic source, using the results of the congestion index [20] and the predefined source traffic priority [8]. Then, the adjustment parameters, i.e., data rate and buffer size, are passed to the WSN configuration services, to perform the physical nodes reconfiguration. This approach will ensure that the QoS requirements for different traffic classes are met while maintaining steady queues in the IoT coordinator buffer. Another aim is to ensure that allocated buffer resources are utilized efficiently.

The VSN residing within the virtualization environment offered by the network simulator is an exact replication of the nodes within the PSN and mimics the network dataflow occurring within the actual physical network. Such cloud level virtualization offers an environment for testing the performance of the network when subjected to different QoS parameters generated by the QoS algorithms residing within the QoS provisioning function, in an iterative fashion.

In addition to this, the past real-life PSN data accumulated within the remote server can be reused on the VSN for simulation purposes to increase the level of accuracy of the obtained simulation results. This in turn aids the process of identifying and converging upon the required QoS parameters in an adaptive manner via closing the loop between PSN and VSN.

In the proposed AQoS concept [22], PSN data within the database acts as the input for the QoS model residing within the network and QoS model simulators. SQL queries are performed by the simulation tool to feed the PSN historical data to the application layer belonging to the simulation model. In addition, the associated time stamps of the historical data can be used by the simulator to compute the service time or other performance measures. Then, the network performance derived from the network simulator is passed to the analytical tool for QoS analysis.

Riverbed Modeler is a network modeler that strives for closer representation of real network devices. The control mechanism and decisions to support AQoS are computed using MATLAB functions. The mathematical analysis tool also supports high level analysis, which makes future network complexity more manageable. MATLAB offers many mathematical functions that ease the computation. MATLAB MX allows use of MATLAB functions in C programs [23]. This in turn makes more efficient executable functions at either the Cloud or the network edge.

The co-simulation approach may also ensure QoS for the virtualized environments, especially when the PSN application workload increases [24]. Hence this structure has great potential for analysis and solving complexity while it acts as part of the WSN system operational management.

The performance of the network is monitored continuously, and the QoS condition is determined using the QoS provisioning algorithm. After an adjustment decision has been made, the adjustment parameters (e.g., data rate and buffer size given in Table I) are written in a C header file. This header file will be used by the operating system to reconfigure the PSN. Note that the parameters are not limited to the examples given. Other parameters related to other potential WSN applications may be selected. Thus, such a closed loop conceptual framework aids the architecture in imposing the necessary dynamic changes on the PSN adaptively.

IV. SYSTEM IMPLEMENTATION

This section presents the system implementation of the proposed architecture. The hardware and software tools that have been selected to implement the system are presented.

A. PSN

In the PSN setup, CC2538 controllers are deployed on a functional basis and at different locations within the Sensor Network and Smart Environment Research Centre (SeNSe) Laboratory, as shown by Fig. 3. There is a total of 16 nodes within this network, 15 of which act as end devices and one node acts as an IoT coordinator. The end devices offer light, temperature, and received signal strength indicator (RSSI) sensing. The sensed values are forwarded to the upper tier of the server database over the Internet.

In the proposed architectural organization, each of the three sensor data captured by the end devices represents different

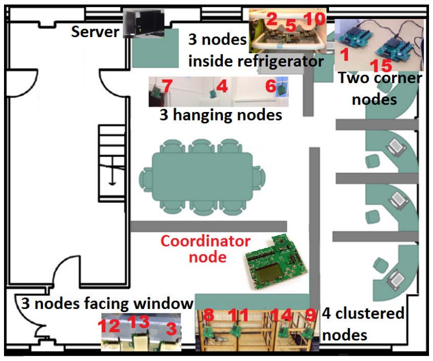


Fig. 3. SeNSe laboratory plan showing deployment of the nodes on a functional basis.

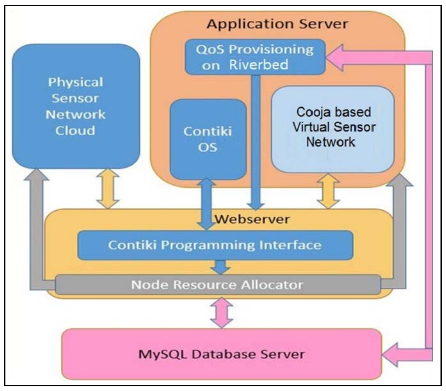

Application Server
QoS Provisioning on Riverbed
Physical Sensor Network Cloud
Contiki OS
Cooja based Virtual Sensor Network
Webserver
Contiki Programming Interface
Node Resource Allocator
MySQL Database Server

Fig. 4. Remote server architecture—inter-relationship between application server, Web server. and database server.

traffic types with different priorities by means of packet remarking. PSN data is transmitted to the IoT coordinator, which in turn relays the data to the cloud server over the Internet wherein the Data Management Services Unit stores and organizes the incoming data in its database.

The nodes' MAC protocol is implemented with polling based on time division multiple access. The end devices transmit their data to the IoT coordinator only when polled. In the current setup, the server database is automatically updated with new incoming data every 90 s. Pseudo codes for both the IoT coordinator as well as the end devices are shown in Table II.

Grouping of nodes on a functional basis renders a clear demarcation between the different sets of groups of nodes. This results in separate and easily distinguishable data plots on the webpage. During the course of our physical experimentation (through data logging), grouping of the nodes in separate geographical locations emerged as a viable option.

B. Server Implementation

The server has been implemented to accommodate the major functions as discussed in Section III. It consists of an application server, a database server, and a Web server. Fig. 4 shows the interworking of the three servers.

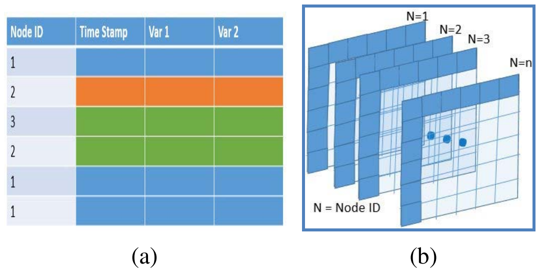
The application server is installed with Contiki OS, mainly to carry out the physical and virtual node configuration. Contiki OS compiles the scheduler to perform network operations to the target node. Riverbed Modeler also resides within

TABLE II
ALGORITHM PSEUDO CODES FOR OPERATION OF IoT COORDINATOR AND END DEVICE

```
Pseudo code for IoT coordinator node
While TRUE {
Initialization:      Set and initialize data storage arrays of size n
                       light[n] = {$0_1,0_2,\ldots 0_n$}
                       temp[n] = {$0_1,0_2,\ldots 0_n$}
                       rssi[n] = {$0_1,0_2,\ldots 0_n$}

Polling request:     For each 'n' end devices: Transmit polling counter
                       messages (1, 2,….., n-1,n) to all the end devices.
INPUT:
Reception:           Keep storing received node data to arrays defined for the 3
                       sensor variables

                     Repeat 'n' times{
                       light[n] = {$l_1,l_2,\ldots l_n$}
                       temp[n] = {$t_1,t_2,\ldots t_n$}
                       rssi[n] = {$r_1,r_2,\ldots r_n$}
                     }
OUTPUT:
Serial data string:  Send sensor data received via serial output
                     For each 'n' end devices
                     Node ID: light[l], temperature[t], rssi[r];

Delay:               Wait for 45 seconds;
}

Pseudo code for end node
While TRUE {
- Sampling:          Sense the 3 sensor values – light, temp and rssi

- Reception:         Receive polling counter messages i.e. 1,2,….,n-1,n

- Check condition:   Check if the condition (Counter = Node ID) is satisfied

        If Transmit flag is set {
                     Transmit Flag = TRUE;

- Transmission:      Transmit data array to IoT coordinator;
                       light[n] = {$l_1,l_2,\ldots l_n$}
                       temp[n] = {$t_1,t_2,\ldots t_n$}
                       rssi[n] = {$r_1,r_2,\ldots r_n$}

                     Transmit Flag = FALSE;
                   }
}
```

the application server for QoS modeling and provisioning purposes. Network simulators such as the Riverbed Modeler allow replication of a PSN to form a VSN which serves as abstraction levels for testing applications or protocols prior to execution on real hardware devices. Riverbed Modeler also allows support for simulation of entire networks and access to model parameters, hence having a significant effect on the simulation accuracy.

MATLAB is also installed in the Application Server, to run the QoS Provisioning module. Therefore, the co-simulation between Riverbed Modeler and MATLAB to support the QoS Provisioning takes place in the Application Server. While users may benefit from Riverbed Modeler as a tool that strives for closer representation to network devices, the control mechanism, and decisions to support AQoS is happening in MATLAB. MATLAB also supports high level analysis, which makes future network complexity more manageable. The co-simulation approach ensures QoS for the virtualized environments, especially when the PSN application workload increases [24]. Hence this structure offers the environment for analysis and solving complex computational requirements.



Node ID	Time Stamp	Var 1	Var 2
1			
2			
3			
2			
1			
1			

Fig. 5. (a) Single table database with Node ID and variables. (b) Node ID as database tables.

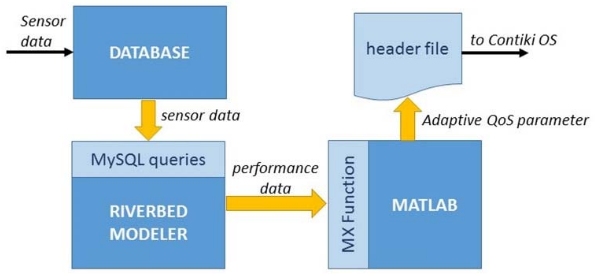


Fig. 6. AQoS concept implementation.

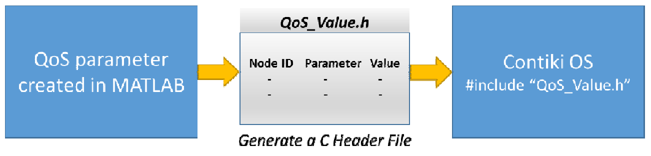


Fig. 7. Passing QoS parameter to Contiki OS.

The application server also contains a Contiki-based Cooja simulator. This is a useful tool for the development of Contiki-based sensor network implementations. It serves as a testing environment for code testing [25] and observes the impact of modifying network parameters such as data rate and network protocol on network performance and power consumption prior to implementation on PSN hardware.

The Web server is the point of entry for the sensor data. It receives data from the IoT coordinator using a REST API. The REST API establishes communication with the users and exchanges information between the application server and the database. The user selects the nodes using GET and POST. The server uses PHP for scripting. The client side uses Python for collecting sensor data. The choice of these specific scripting languages is due to a minimal required learning curve for the researchers. The client forward data to the Web server. The Web server forward the data to the MySQL database for processing and determining sensor node statistics using REST APIs. The QoS parameters predicted by the QoS provisioning module are inputs to the Web server that use the Contiki programming interface to update the physical and virtual nodes with the QoS parameters. In the following, the implementations of the main functional blocks of the server are discussed.

1) Data Management Services: MySQL is implemented within the database server as the Relational Database Management System. The database queries retrieve the PSN data from the incoming data packet. The database is organized such that each node has its own table. The SQL queries query the table to get the node information.

The collected data packets are time stamped and stored in rows within the database. As the size of the database increases, the latency of data retrieval increases proportionally. This spatio-temporal increase has a performance implication. Our findings [22] show that data retrieval can be maintained at a minimal rate. This reflects the independence of retrieval rate on the number of nodes when individual tables for every sensor node are used.

A database table as shown in Fig. 5(b) resulted in $O(n\log(n))$ complexity for data retrieval. It required two operations: a search operation of the node and next a read operation to fetch the data. The temporal requirement for the search operation was improved from $O(\log(n))$ to $O(1)$ by splitting the single table into multiple tables as illustrated in Fig. 5(b). As shown in the figure, each table represents a specific physical or virtual node. As the complete node data is encapsulated in individual tables, the overall retrieval time is reduced from $O(n\log(n))$ to $O(n)$. This has also supported the one-to-one representation of the nodes within the PSN.

2) Adaptive QoS Provisioning—Conceptual Model and Analytics: Fig. 6 illustrates the implementation of the proposed AQoS concept [22]. PSN data within the database acts as the input for the QoS model residing within the Riverbed Modeler. SQL queries are performed by the simulation tool to feed PSN historical data to the application layer belonging to the simulation model. In addition, the associated time stamps of the historical data can be used by the simulator to compute the service time or other performance measures.

Then, the network performance derived from the model hosted by the Riverbed Modeler is passed to the QoS analysis model hosted by MATLAB. MATLAB uses the performance data to analyze the QoS condition and to identify the corresponding AQoS parameters. In the system implementation, the co-simulation between Riverbed Modeler and MATLAB is set up using MATLAB MX functions and Riverbed Modeler APIs [23], as shown by Fig. 6. The QoS algorithm shown in Table I which has been discussed in Section III is implemented as a use case.

3) WSN Configuration Services Using Contiki OS: In the system implementation, the low power wireless transceivers are subjected to many reconfigurations over the Internet. In addition, a self-configuring architecture [26] which requires no human intervention for the reconfiguration and deployment of IoT applications [27] is a desirable approach. To deal with such stringent requirements of an IoT-based environment, Contiki OS offers advantages such as flexibility, multitasking, and concurrency [28].

As shown in Fig. 7, the QoS parameters predicted by the QoS provisioning module are inputs to the Web server that uses the flexible and open source Contiki OS residing within the SeNSe server to update the physical and virtual nodes with the QoS parameters. Contiki OS is specifically compiled for the physical nodes and the code is in C. The reconfiguration header file generated by the QoS provisioning module is

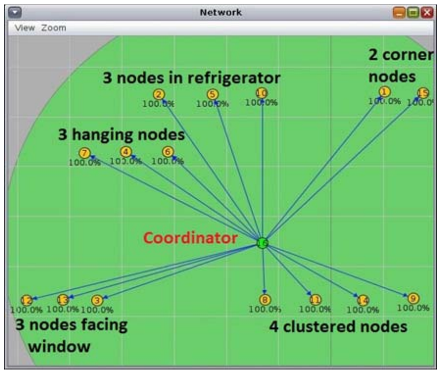


Fig. 8. VSN implementation in Cooja simulator during running conditions.

integrated into the directory that is compiled into the hex file for the nodes. With the current implementation, the reconfiguration file is generated as a C header file that is created in the directory where the Contiki OS code resides. Currently, the reprogramming of the nodes with the new reconfiguration settings is done manually. However, part of our on-going work involves a fully automated system, whereby the VSN would be automatically reconfigured based on the information it received from the physical network.

4) Virtual Sensor Network Implementation: Contiki OS residing within the SeNSe server consists of an in-built Cooja simulator. Cooja is used as a test ground before actual deployment of the PSN. The implementation of VSN in Cooja is a representation of the actual PSN deployment within the laboratory. The Contiki OS code used to program the PSN nodes is cross-compiled for the VSN nodes, both at the end device and node IoT levels. This allows the simulation to be a closer representation of the actual hardware, precisely mimicking the actual processing and flow of data within the VSN.

Fig. 8 depicts the VSN in running condition within the Cooja simulator. The virtual end devices have been clustered in a similar way as those of the real-life functional deployment positions of the PSC setup within the lab. However, owing to the approximation of their positions with respect to the actual deployment, the RSSI signal representation emanating out of this virtual setup is not entirely precise.

As compared to recent IoT architectures such as Lysis and iCore which consist of abstractions of real world objects functionalities [13] and virtual interfaces from real world objects [14], which result in advantages of reduced delay [13] and reuse of the persisting virtual interfaces [14], our idea herein was to merely have a virtual representation of the physical sensor nodes within the cloud. Such a virtual network composed of Cooja nodes would accurately reflect/represent the logical (software code) facet of the physically deployed sensor network operation and thus, aid securing an accurate and precise logical correlation between the physical and virtual WSN environments.

V. Tests and Case Study

To ensure a virtualization with the closest replication of the PSN with the Riverbed Modeler simulation tool, comparisons

TABLE III
Experiment Parameters

Test Cases		Service Rate (SR), pkt/s	
		RT (Light)	DT (Temperature)
Case 1	SR = AR	8	5
Case 2	SR > AR	16	10
Case 3	SR > AR	9	7
Case 4	SR < AR	4	2
Case 5	SR < AR	7	4

between PSN and simulation behavior were conducted. The QoS performance of the network's heterogeneous data traffic was studied via implementation and simulation of similar models.

The communication between one IoT coordinator and the end devices in the PSN testbed has been established as unicast on a single channel using Contiki OS. The polling-based network protocol was successfully implemented for the Contiki-based PSN setup. In the simulation, two types of traffic have been defined. RT and DT traffic flows are fed into the IoT coordinator buffer, which in turn serves the packets based on a predefined queuing policy.

Furthermore, the queuing mechanism in the IoT coordinator is observed, and the QoS performance of the traffic is studied. The interoperability among system components is also discussed through representation of sensor data from the database and the related database performance.

A. Effect of SR

To ensure the comparability of both physical and simulated environments, an experiment with a single node communicating with a single IoT coordinator was done first. The aim is to identify the impact of SR to network performance. Five cases have been set up—all of which are differentiated by the values of packet AR, the IoT coordinator's SR, and buffer size. AR and SR are defined as the number of packets that arrived at the buffer and are served by the buffer per second, respectively. The AR in both the PSN and simulation is set by defining the sampling rate of the traffic sources. This is also indicated in Table II within the pseudo code for the end nodes.

As shown in Table III, it is assumed that RT data is light sensory data [29], whereas DT data is represented by temperature sensory data. For all cases, the AR of RT and DT are constant at 8 packets per second (pkt/s) and 5 pkt/s, respectively. Furthermore, a relatively small buffer size of 20 packets is also kept constant for each of both RT and DT buffers. The small buffer size was set to allow seeing the results of different AR and SR more easily. The experiment was conducted for a duration of 10 min.

In both experiments, it was observed that RT traffic and DT traffic are served according to the assigned serviced rates of the different queues in the buffer. As shown by Fig. 9, the trend of serviced packets to the function of received packets at the buffer for all cases shows a good match. The difference between both environments was always smaller than 3% for cases 1, 4, and 5. Furthermore, as shown in Fig. 9(b),

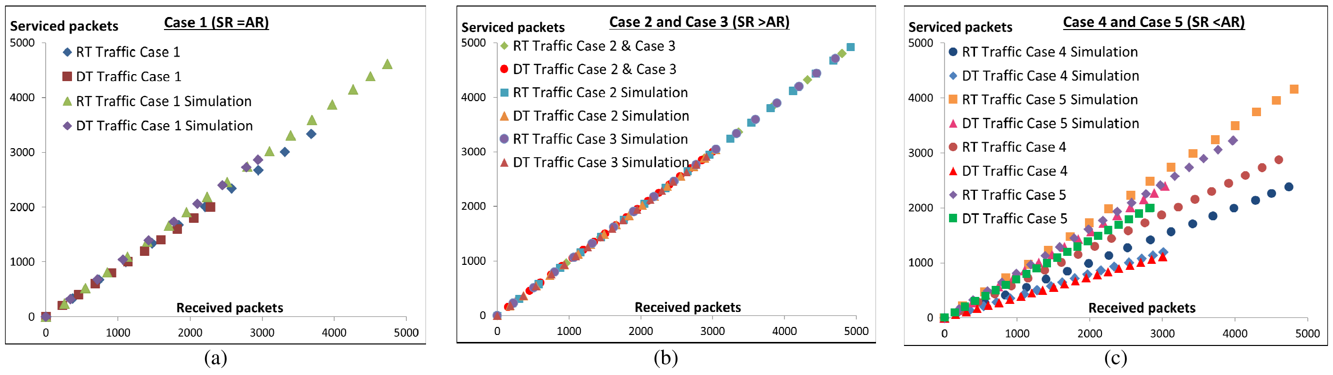
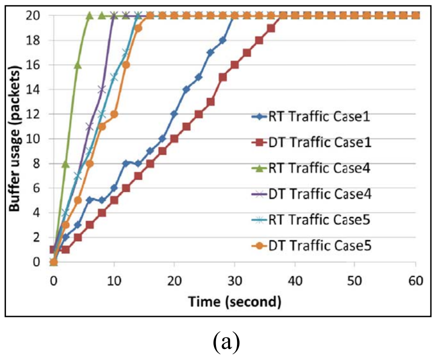


Fig. 9. Comparison between PSN and simulation serviced packets versus received packets for (a) SR = AR, (b) SR > AR, and (c) SR < AR.



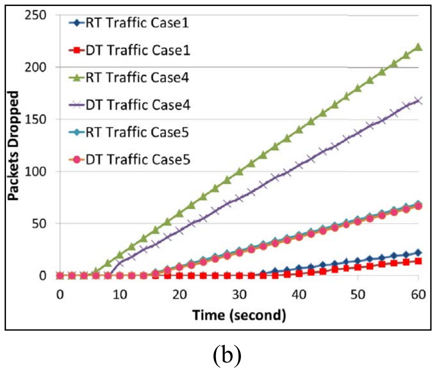


Fig. 10. (a) Buffer usage over 1-min experiment. (b) Packets dropped over 1-min experiment.

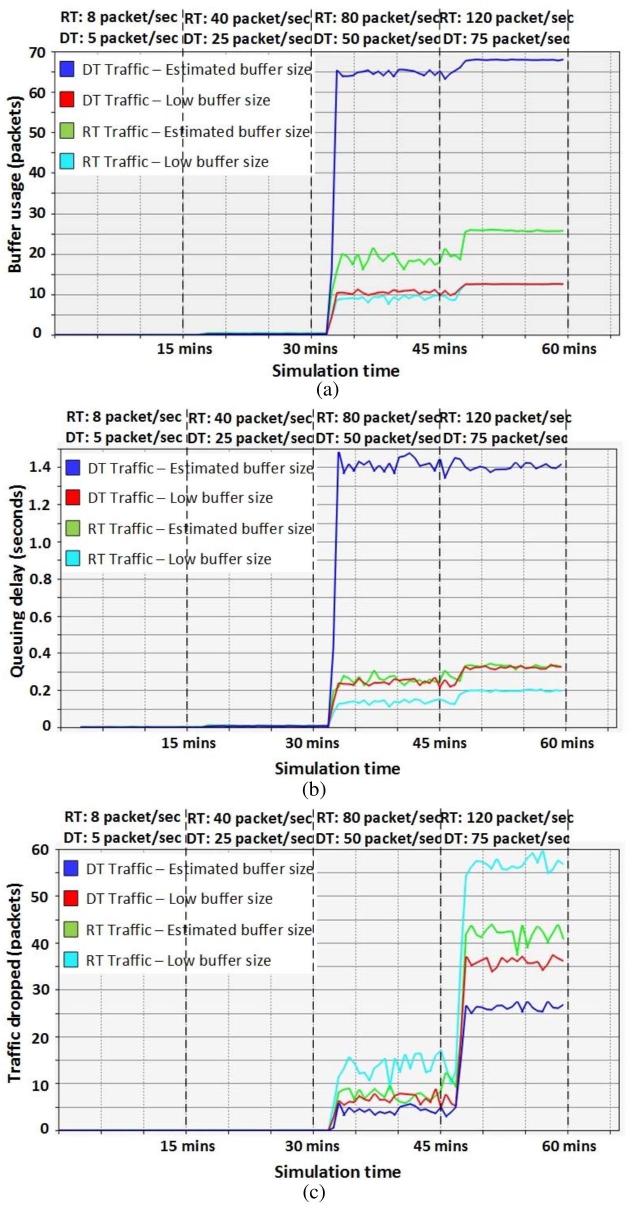


Fig. 11. (a) Buffer usage (packets). (b) Queuing delay (s). (c) Traffic drop (packets) versus simulation time.

when SR>AR, there is very significant similarity in both PSN experiment and simulation.

In the PSN experiment, a small deviation occurs in terms of total packets received in the IoT coordinator. Meanwhile, the traffic generated in the simulation is always ideal based on theoretically calculated values. Fig. 9(a) shows that the total RT packets received at the buffer was only 3680 packets in the PSN experiment, whereas the actual number of incoming packets shown by the simulation is 4800. The deviation of the packets received in the IoT coordinator may be due to the communication channel's stability between the sensor node and the IoT coordinator. However, the overall rate at which traffic is served in the RT and DT buffer queues through the course of both environments showed a good match. This validates the fact that the behavior of the queue and node model in the simulation closely resembles the performance characteristics of an actual IoT coordinator.

The buffer usage and traffic drop also showed a similar trend between both PSN experiments and simulation. Both cases 2 and 3, i.e., when SR > AR, have 100% delivery

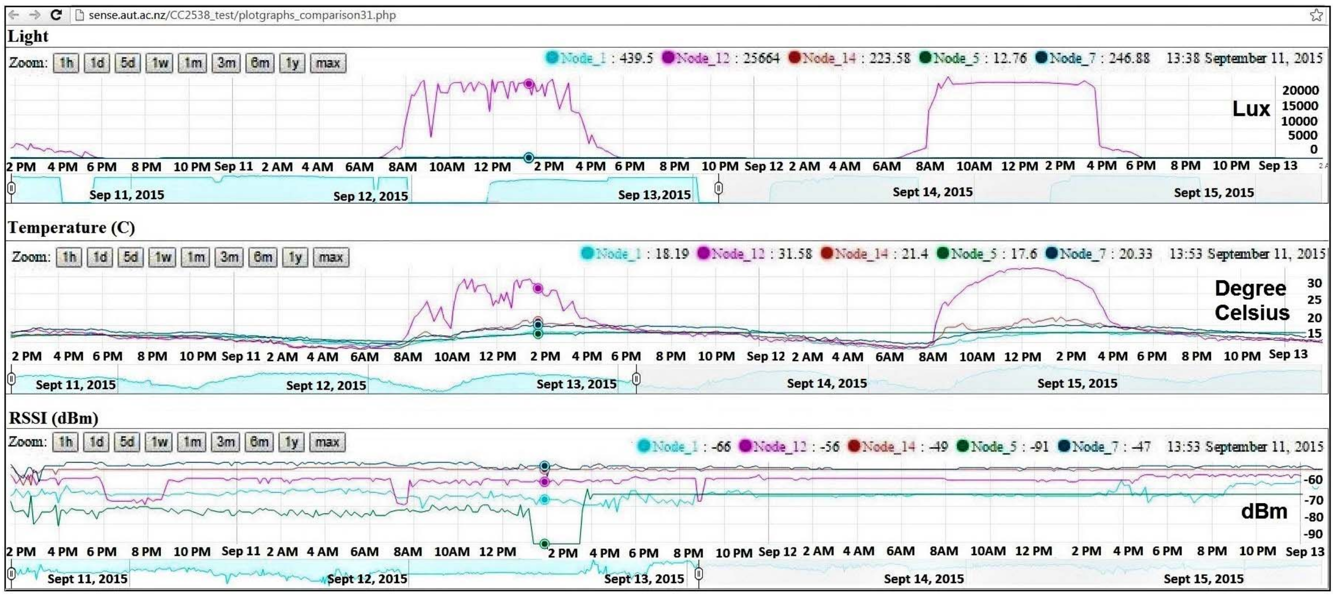


Fig. 12. Sensor data representation from the database.

rate and zero buffer usage at all time. This indicates the traffic drop can be avoided when packets are served at a higher rate than the AR. Fig. 10(a) and (b) depicts the buffer usage and packet drop for the remaining cases, i.e., cases 1, 4, and 5. Fig. 10(a) shows that when the packet rate for traffic source nodes is increased, the buffer of the IoT coordinator grows steadily. Fig. 10(b) shows that once the buffer usage has reached its maximum capacity, packets start to be dropped. Consequently, more incoming packets will be dropped due to buffer overflow. This can be solved by adjusting the buffer size to match the requirements of the RT and DT traffic.

B. Effect of Buffer Size and AR

With the conforming queuing behavior between both PSN and the simulated environment, it was deemed sufficient to study the effect of different IoT coordinator's buffer sizes in a network with multiple numbers of nodes in a simulation environment. A simulation on Riverbed Modeler involving 15 nodes was conducted to investigate the behavior of the network under varying data rates and buffer sizes. Service differentiation among RT and DT traffic was introduced by implementing a weighted fair queuing (WFQ) policy. In WFQ policy, each traffic source can be assigned a weight. Traffic source with higher weight receive more bandwidth than those with less weight [8]. For this purpose, the normalized weight assigned to RT and DT traffic classes are set to 0.7 and 0.3, respectively. In addition, a low buffer size of 30 packets and estimated buffer size were allocated for RT and DT traffic, respectively.

The idea was to allocate sufficient buffer storage to DT traffic to avoid buffer overflow which may result in high traffic drop and to allocate more bandwidth to RT traffic to meet low bound delay. This will accommodate the timeliness and reliability QoS requirements associated with RT and DT traffic. Over the period of a 1 h simulation, RT and DT packets' data rates were increased every 15 min.

Fig. 11(a)–(c) shows the performance for varying RT and DT data rates, in terms of buffer usage, queuing delay, and traffic drop, respectively. If the AR is increased, the estimated buffer should be able to accommodate for the burst traffic, even when the SR on the IoT coordinator remains constant. Insufficient buffer size will result in packet drop when the AR is greater than the SR. The results confirm that by introducing service differentiation among the traffic as well as estimating the required buffer size, both RT and DT traffic met their QoS requirements. As shown in Fig. 11(b), by setting higher weights to the RT traffic, hence more bandwidth is allocated, the low bound delay requirement of RT traffic is met. Furthermore, as shown in Fig. 11(c), by estimating sufficient buffer size to DT traffic, traffic drop can be minimized.

C. Test Environment—Sensing Data

The graphs in Fig. 12 show samples of the signal trends of data collected, i.e., ambient light, temperature, and the RSSI values, for some of the nodes used in the test. The sensory data trends contain the information associated with the layout of the nodes deployment, as shown by Fig. 3. The figure shows the readings for three consecutive days. From the light sensor reading, it is observed that the light values are high when it is sunny during the day, particularly on the nodes facing the window. A lower light reading is observed from the nodes facing the lab. It is also shown that the temperature goes up during the day for those nodes facing the window, reflecting the internal heat inside the controller chips. The RSSI data trend indicates some activities during the day due to people movement within the lab, whereas more stable RSSI values can be seen during the night.

VI. Conclusion

A conceptual organization targeting AQoS is presented in this paper. The architecture offers a system for interactions between the behavior of a PSN and the necessary analysis in a virtual remote environment. The main contribution of the architecture is its capability of identifying the required adjustment for the PSN in order to enhance the QoS performance of the WSN applications. An adaptive service differentiation QoS algorithm which is implemented as a target application of the system is presented. This represents a use case of an AQoS algorithm which can provide requested QoS for different traffic classes.

The concept of virtualization has been presented and the performance of both PSN and computer simulation under the QoS model has been distinguished. The virtualization has been demonstrated as the network performance shows similarity between the developed model in the simulation and the PSN experiment. The techniques for interoperability among the system components, namely PSN and a remote database, VSN and QoS provisioning unit, as well as Contiki OS and network reconfiguration, are presented in this paper. With the encouraging preliminary outcomes between pairing components, the proposed architecture can be used as a test environment for experimentation involving custom AQoS parameters and real-time analysis of network performance. The co-simulation concept between network simulator and MATLAB is viable and this opens up more future research possibilities when more complexity takes place at the business intelligence level. Ongoing effort includes generating incremental outcomes from the modular organization and to achieve a completely automated system.

Our future research includes extending the experimental sensor network, and comparing the performances of the proposed adaptive QoS with other related work. The latter may potentially be the assessment of this paper in contrast to the QoS algorithm through cross-layer optimization approach such as the one proposed in [20], and distinguishing the QoS performance against other related QoS middleware architecture. Our future work also includes accommodating the learning process of predicting the network parameters for a given QoS and depositing this as part of an event warehouse. The system would be designed in such a way that as new scenarios are experienced by the network, the event warehouse database would be updated accordingly, so that the information could be used to predict future QoS parameters as the system evolves. This would lead to the creation of an unsupervised learning tool for controlling the PSNs. We envision that the approach will overcome the repetition of adjustment calculation for the AQoS, hence contributing improvement to the system's latency. The challenges of the learning process and addressing the latency during reconfiguration activities will be part of our future research.

References

[1] N. A. Alrajeh, S. Khan, and B. Shams, "Intrusion detection systems in wireless sensor networks: A review," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 5, Jan. 2013, Art. no. 167575.

[2] J.-F. Martinez *et al.*, "Modelling QoS for wireless sensor networks," in *Wireless Sensor and Actor Networks* (IFIP), vol. 248. Boston, MA, USA: Springer, 2007, pp. 143–154.

[3] A. Gluhak *et al.*, "A survey on facilities for experimental Internet of Things research," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 58–66, Nov. 2011.

[4] S. Ezdiani, I. S. Acharyya, S. Sivakumar, and A. Al-Anbuky, "An architectural concept for sensor cloud QoSaaS testbed," in *Proc. 6th Workshop Real World Wireless Sensor Netw. (RealWSN)*, Seoul, South Korea, 2015, pp. 15–18.

[5] B. Bhuyan, H. K. D. Sarma, N. Sarma, A. Kar, and R. Mall, "Quality of service (QoS) provisions in wireless sensor networks and related challenges," *Wireless Sensor Netw.*, vol. 2, no. 11, pp. 861–868, Nov. 2010.

[6] B. Nefzi and Y.-Q. Song, "QoS for wireless sensor networks: Enabling service differentiation at the MAC sub-layer using CoSenS," *Ad Hoc Netw.*, vol. 10, no. 4, pp. 680–695, 2012.

[7] F. Safaei, H. Mahzoon, and M. S. Talebi, "A simple priority-based scheme for delay-sensitive data transmission over wireless sensor networks," *Int. J. Wireless Mobile Netw.*, vol. 4, no. 1, pp. 165–181, 2012.

[8] M. H. Yaghmaee and D. A. Adjeroh, "Priority-based rate control for service differentiation and congestion control in wireless multimedia sensor networks," *Comput. Netw.*, vol. 53, no. 11, pp. 1798–1811, 2009.

[9] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion control protocols in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1369–1390, 3rd Quart., 2014.

[10] Y.-L. Chen and H.-P. Lai, "A fuzzy logical controller for traffic load parameter with priority-based rate in wireless multimedia sensor networks," *Appl. Soft Comput.*, vol. 14, pp. 594–602, Jan. 2014.

[11] A. Barbato, M. Barrano, A. Capone, and N. Figiani, "Resource oriented and energy efficient routing protocol for IPv6 wireless sensor networks," in *Proc. IEEE Online Conf. Green Commun. (GreenCom)*, Piscataway, NJ, USA, 2013, pp. 163–168.

[12] *Santander on Fire—Future Internet Research & Experimentation*. Accessed: Jul. 27, 2017. [Online]. Available: http://www.smartsantander.eu

[13] R. Girau, S. Martis, and L. Atzori, "Lysis: A platform for IoT distributed applications over socially connected objects," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 40–51, Feb. 2017.

[14] R. Giaffreda, "iCore: A cognitive management framework for the Internet of Things," in *The Future Internet: Future Internet Assembly 2013: Validated Results and New Horizons*, A. Galis and A. Gavras, Eds. Heidelberg, Germany: Springer, 2013, pp. 350–352.

[15] M.-A. Nef, S. Karagiorgou, G. I. Stamoulis, and P. K. Kikiras, "Supporting service differentiation in wireless sensor networks," in *Proc. 15th Panhellenic Conf. Informat. (PCI)*, Kastoria, Greece, 2011, pp. 127–133.

[16] M.-A. Nef, L. Perlepes, S. Karagiorgou, G. I. Stamoulis, and P. K. Kikiras, "Enabling QoS in the Internet of Things," in *Proc. 5th Int. Conf. Commun. Theory Rel. Qual. Service (CTRQ)*, Chamonix, France, 2012, pp. 33–38.

[17] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, Baltimore, MD, USA, 2004, pp. 134–147.

[18] Y. G. Iyer, S. Gandham, and S. Venkatesan, "STCP: A generic transport layer protocol for wireless sensor networks," in *Proc. 14th Int. Conf. Comput. Commun. Netw. (ICCCN)*, San Diego, CA, USA, 2005, pp. 449–454.

[19] C. T. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 148–161.

[20] C. Wang, B. Li, K. Sohraby, M. Daneshmand, and Y. Hu, "Upstream congestion control in wireless sensor networks through cross-layer optimization," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 4, pp. 786–795, May 2007.

[21] H. Liu, A. Srinivasan, K. Whitehouse, and J. A. Stankovic, "Melange: Supporting heterogeneous QoS requirements in delay tolerant sensor networks," in *Proc. 7th Int. Conf. Netw. Sens. Syst. (INSS)*, Kassel, Germany, 2010, pp. 93–96.

[22] S. Ezdiani, I. S. Acharyya, S. Sivakumar, and A. Al-Anbuky, "An IoT environment for WSN adaptive QoS," in *Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst.*, Sydney, NSW, Australia, 2015, pp. 586–593.

[23] S. Sivakumar, "Energy efficient opportunistic connectivity for wireless sensor network," Ph.D. dissertation, School Eng., Auckland Univ. Technol., Auckland, New Zealand, 2013.

[24] J. Liu, Y. Zhang, Y. Zhou, D. Zhang, and H. Liu, “Aggressive resource provisioning for ensuring QoS in virtualized environments,” *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 119–131, Apr./Jun. 2015.

[25] I. Khan *et al.*, “A data annotation architecture for semantic applications in virtualized wireless sensor networks,” in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ottawa, ON, Canada, 2015, pp. 27–35.

[26] S. Cirani *et al.*, “A scalable and self-configuring architecture for service discovery in the Internet of Things,” *IEEE Internet Things J.*, vol. 1, no. 5, pp. 508–521, Oct. 2014.

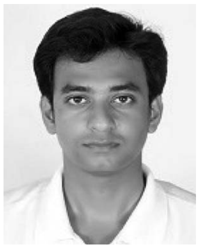
[27] R. Rajkamal and P. V. Ranjan, “Packet classification for network processors in WSN traffic using ANN,” in *Proc. 6th IEEE Int. Conf. Ind. Informat. (INDIN)*, Daejeon, South Korea, 2008, pp. 707–710.

[28] A. Dunkels, B. Grönvall, and T. Voigt, “Contiki—A lightweight and flexible operating system for tiny networked sensors,” in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Tampa, FL, USA, 2004, pp. 455–462.

[29] Y. Li and L. E. Parker, “Intruder detection using a wireless sensor network with an intelligent mobile robot response,” in *Proc. IEEE Southeastcon*, Huntsville, AL, USA, 2008, pp. 37–42.

Syarifah Ezdiani received the B.Eng. degree in computer and communication systems and M.Sc. degree in computer systems engineering from Universiti Putra Malaysia, Seri Kembangan, Malaysia. She is currently pursuing the Ph.D. degree at the Auckland University of Technology, Auckland, New Zealand.

She is with the Sensor Network and Smart Environment Research Centre, Auckland University of Technology, where she is involved with research on quality of service (QoS) provisioning of integrated wireless sensor network (WSN) and Internet systems. Her current research interests include WSN QoS and computational intelligence as applied to QoS for IoT-based WSN.



Indrajit S. Acharyya received the M.Tech. degree in embedded systems and technology from SRM University, Chennai, India, in 2013, and the Master of Engineering degree from the Auckland University of Technology (AUT), Auckland, New Zealand, in 2014, where he is currently pursuing the Ph.D. degree.

He is involved with research with the SeNSe Research Center, AUT. His current research interests include WSN, IoT, and network function virtualization.

Sivaramakrishnan Sivakumar received the B.Tech. degree from Dr. A. P. J. Abdul Kalam Technical University, Lucknow, India, in 2006, the M.Sc. (Engg.) degree from Coventry University, Coventry, U.K., in 2008, and the Ph.D. degree from the Auckland University of Technology, Auckland, New Zealand, in 2013, where he was focused on energy efficient opportunistic connectivity for wireless sensor networks (WSNs).

From 2014 to 2016, he was a part of the SeNSe Laboratory, Auckland University of Technology, Auckland, as a Post-Doctoral Research Fellow and has been involved in the study of sensor networks for cold storage logistics. His research focuses mainly on WSNs. His current research interests include Internet of Things, WSNs, big data visualization, and sensor network configuration through cloud and emulation.

Adnan Al-Anbuky is a Full Professor with the Auckland University of Technology, Auckland, New Zealand, where he directs the SeNSe Research Laboratory. His current research interests include dynamic interaction of the WSN with the physical phenomenon in capturing the critical events taking into consideration the IoT and cloud support. Recently, involvement in projects like public space ambient intelligence PSAmI and traceability of food condition during logistics has encouraged the contribution toward the concepts of cyber-physical systems and smart cities. They both involve federated sensor networks, IoT and Big-Data management.

Mr. Al-Anbuky is an editorial board member of a number of international journals and scientific groups. He is actively contributing to the organization or operation of numerous local and international events and conferences. He has delivered a number of keynote talks and has numerous conference and journal peer reviewed publications.