An Experimental Study of Congestion Control in Wireless Sensor Networks Using a Combination of Resource and Traffic Control

Thesis submitted for the degree of

Doctor of Philosophy

at the University of Leicester

by

Tarik KHALEL

Department of Engineering

University of Leicester, UK

March 2018

Abstract

Wireless Sensor Networks (WSNs) consist of a number of spatially distributed autonomous sensor nodes that are configured to gather physical or environmental information in an area. A sensor node can be subject to many constraints including limited power, low processing and sensing capabilities, small memory, small bandwidth, and low communication range. In WSNs, when an event occurs, a sudden burst of network traffic will be generated which may lead to congestion and hence to lost or delayed data packets, decreased reliability, and increased power consumption. One approach that controls the congestion in WSNs is the resource control method, in which congested nodes are avoided by sending the packets through alternative paths.

To test the existing methods of congestion control e.g. the Hierarchical Tree Alternative Path (HTAP) algorithm and to develop a new method, an experimental facility consisting of 40 nodes (Waspmotes) was deployed. In order to accurately measure end to end delay and to ensure nodes wake up at the same time, the node clocks were synchronized to an accuracy of approximately 40μsec between nodes and to 5ms between nodes and external time, using a method applied experimentally for the first time.

The experiments demonstrated that HTAP has better performance (Packet Delivery Ratio - PDR, throughput, and End to End delay) than a network without congestion control. However, in some network topologies where alternative paths do not exist, in the event of congestion, the network performance will be degraded. Therefore, traffic control, which reduces the packet rate, has also been employed to mitigate congestion. The location of the single path affects the network performance with poorer performance when it is close to the sink than to the source. Based on the results, using both resource control and traffic control leads to the best network performance.

Acknowledgements

First and foremost, all praises and thanks are due to almighty ALLAH, the most Gracious and the most Merciful who has created everything and taught the creation which they knew not and may ALLAH's peace and blessings be upon the Prophet Mohammed.

Then, I am very grateful to my supervisor Dr. Alan Stocker who was abundantly helpful and offered invaluable assistance, support and guidance. I would also thanks Prof. Mike Warrington, and Dr. David Siddle for their numerous help and support.

I am also thankful to Iraqi Ministry of Higher Education and Scientific Research (MOHESR) and University of Mosul for funding and supporting my research.

Also, I thank my colleagues at the University of Leicester, especially Radio Systems Laboratory Research group: Dr. Zaid, Dr. Hasanin, Dr. Mohammed Abdullah, Saddam, and Ameer for their support and beneficial conversations during my study.

Deepest gratitude is also due to Bilal Haveliwala, Dominic Kent, and Michelle Pryce for their help during the period of my study.

The most important thanks are for my family, especially for My wife for her support and patience throughout the period of my study.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

A

ADC	Analogue to Digital Converter
AIMD	Additive Increase Multiplicative Decrease
API	Application Programming Interface
ARC	Adaptive Rate Control
ATS	Average TimeSynch

C

CADA	Congestion Avoidance, Detection and Alleviation
CAR	Congestion-Aware Routing
CC	Congestion Control
CCF	Congestion Control and Fairness
CDU	Congestion Detection Unit
CL-APCC	Cross-Layer Active Predictive Congestion
CNU	Congestion Notification Unit
CODA	COngestion Detection and Avoidance
COMUT	COngestion control for MUlti-class Traffic
CONSISE	CONgestion control from SInk to SEnsors
CSMA/CA	Carrier Sense Multiple Accesses /Collision Avoidance
CTCP	Collaborative Transport Control Protocol

D

DBL	Drop-Burst Length

E

ECODA	Enhanced COngestion Detection and Avoidance
EECC	Energy Efficient Congestion Control
EEPROM	Electrically Erasable Programmable Read-Only Memory
ESRT	Event to Sink Reliable Transport

F

FACC	Fairness-Aware Congestion Control
Flock-CC	Flock-based Congestion Control

G

GCS	Global Clock Synchronization

H

HOCA	Healthcare Aware Optimized Congestion Avoidance
HP	High Priority
HTAP	Hierarchical Tree Alternative Path

I

ICD	Intelligent Congestion Detection
ICN	Implicit Congestion Notification
IEEE	Institute of Electrical and Electronics Engineers
IFRC	Interference-aware Fair Rate Control
ISM	Industrial, Scientific, and Medical

L

LACAS	Learning Automata-based Congestion Avoidance Scheme
LAN	Local Area Network
LMST	Local Minimum Spanning Tree
LP	Low Priority
LTS	Lightweight Time Synchronization

M

MAN	Metropolitan Area Network
MEMS	Micro-Electro-Mechanical System
MFR	Mac Footer
MHR	Mac Header
MSDU	Mac Service Data Unit
MST	Minimum Spanning Tree

N

NCC	No Congestion Control
NIST	National Institute Standards and Technology
NTP	Network Time Protocol

O

OSI	Open System Interconnection

P

PAN	Personal Area Network
PCCP	Priority-based Congestion Control Protocol
PDR	Packet Delivery Ratio
PPDU	Physical Protocol Data Unit
PRA	Priority-based Rate Adjustment
PSDU	Physical Service Data Unit

Q

QCCP-PS	Queue based Congestion Control Protocol with Priority Support
QoS	Quality of Service

R

RAM	Random Access Memory
RAU	Rate Adjustment Unit
RBS	Reference Broadcast Synchronization
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RTC	Real Time Clock
RTT	Round Trip Time

S

SD	Standard Deviation
SFD	Start of Frame Delimiter
SHIMMER	Sensing Health with Intelligence, Modularity, Mobility, and Experimental Reusability
SHR	Synchronization Header
SMA	Sub-Miniature Version A
SNR	Signal to Noise Ratio
SRAM	Static Random-Access Memory
SSID	Service Set Identifier
STCP	Sensor Transmission Control Protocol

T

TADR	Traffic-Aware Dynamic Routing protocol
TARA	Topology Aware Resource Adaptation
TDMA	Time Division Multiple Access

U

UTC	Universal Coordinated Time

W

WAN	Wide Area Network
WRAN	Wireless Regional Area Network
WSNs	Wireless Sensor Networks

Chapter 1: Introduction

The aim of this thesis is to address the problem of congestion in Wireless Sensor Networks (WSNs), which can have a significant effect on the performance of WSNs. The overall objective is to test and improve a WSN protocol in order to enhance the Quality of Service (QoS). The improvement consists of combination two control methods: resource control and rate control. The test of the simulated protocol and the suggested improvement is achieved experimentally on a deployed WSN.

This chapter starts with the background of WSN and its applications, followed by the background of the problem, the statement of the research problems, aims of this study, and the outline of this thesis.

1.1 Background

Technological advances in the area of Micro-Electro-Mechanical System (MEMS), wireless communication, and the underlying semiconductor technology have enabled researchers to build low-cost and small size wireless sensor node (Akyildiz *et al.*, 2002; Matin and Islam, 2012; Roseline *et al.*, 2013). A Wireless Sensor Network (WSN) is defined as a group of devices called sensor nodes that work together to form a wireless network (Figure 1.1).

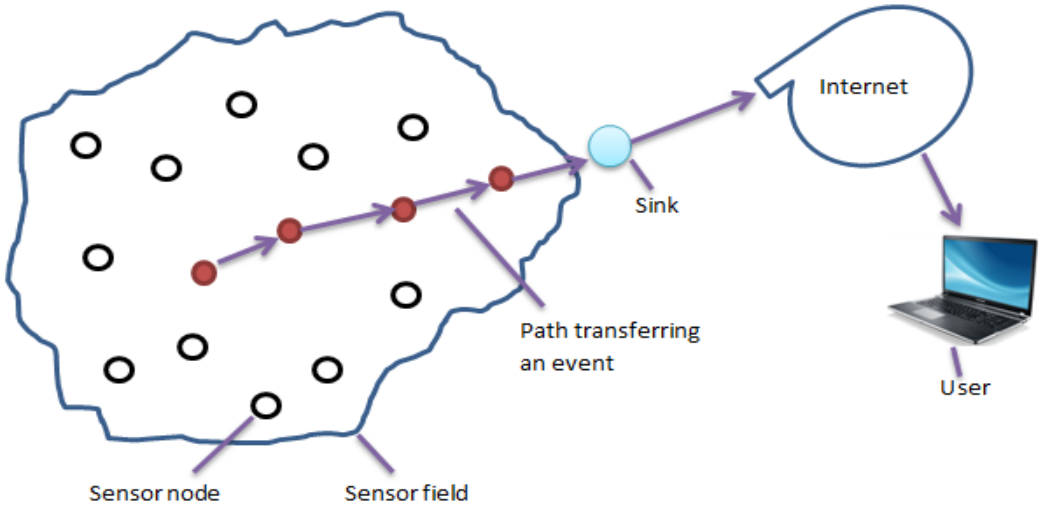


Figure 1 - 1: Wireless Sensor Network (after Yang and Mohammed 2000).

These nodes transfer information (e.g. temperature) from the monitored area to the sink in one or multiple hops through communication over wireless links. The data can be used locally at the sink or forwarded to other networks (e.g., the Internet) through a gateway connection as shown in Figure 1.1. Sensor nodes can be mobile or static, know their locations or not, and be homogeneous or heterogeneous (Buratti *et al*., 2009).

Sensor networks are subjected to tight communication, computation, and memory constraints. These constraints are due to size and cost limitations, since most WSN applications require tiny and cheap sensors (Mahapatro and Khilar 2011). This type of network differs from conventional ad-hoc networks in that typically the number of deployment nodes is higher, the nodes more densely distributed and usually in harsh environment, the nodes have short life-time, the topology may change frequently, and a WSN works with limited power and range resources (Alkhatib *et al*., 2013).

1.1.1 WSN applications

From above mentioned features of sensor nodes, and because there are different types of sensors like temperature sensor, humidity sensor, acoustic sensor,…etc, WSNs have been used in large number of applications such as environment monitoring, agriculture monitoring, health sector, home applications, and military applications (Ahmad *et al*., 2016). WSNs applications can be broadly classified into 3 types: i) continuous monitoring, ii) on demand, and iii) event driven (Borges *et al*., 2014). Some of the possible applications are listed in the following sections:

i) **Continuous Monitoring (periodic intervals) based application:** are where the data obtained at each node (e.g. temperature) is sent periodically to the sink (Rivero-Angeles & Bouabdallah, 2009). Some examples of continuous monitoring based applications:

- Military applications: To support operations and surveillance (Ahmad *et al*., 2016). For these applications, data need to be accurate and the network sufficiently long to support the military requirements (Khakestani & Balochian, 2015). WSNs can be used in battlefield surveillance, monitoring friendly forces, and the detection of chemical, biological, or nuclear weapons (Chatterjee & Pandey, 2014).
- Road traffic applications where high density of sensors are deployed along roads to monitor the traffic. The sensors can help in decreasing traffic accidents and help drivers find safe routing (Micek & Karpis, 2010).

ii) **On-demand based applications:** the sink sends requests for data to the sensors in the network. A query is only sent when measurements are required, and this action helps to save energy. Some examples of this type:

- Smart parking can help to solve vehicle parking problems especially in major cities. Web-servers collect the availability of parking spaces in real time from the target car parks, so that this information can be conveyed to users (Yang *et al*., 2012).
- In a water quality monitoring system, the sensors are deployed under water to monitor the quality of the water. Then, based on the collected data the central system will detect problems with water quality, and the required actions can then be taken (Pule *et al*., 2018).

iii) **Event driven based applications:** are those where an event of interest occurs at the sensor nodes is then report of to the sink. The collected data will be processed at the sink then a suitable decision will be taken (Alkhatib *et al*., 2013). Some examples of this type are listed below:

- To detect forest fires, a pre-deployment of a number of sensors is required to gather different types of raw data such as temperature, humidity, pressure, and position. After that, the collected data will be sent wirelessly to the sink, then from the sink to a control centre via a transport network such as GSM. For the collected data to be useful, the data must be delivered with low latency (Bouabdellah *et al*., 2013).
- For security and surveillance applications the nodes are equipped with embedded devices such as acoustic sensors or cameras. When an event occurs the data will be sent directly to central unit, then required action will be taken to protect the environments (Okediran, 2014).

In this thesis, the work focuses on event driven WSNs.

1.1.2 Sensor node components

A typical sensor network is limited in power, communication bandwidth, sensing capability, memory, signal processing and task scheduling (Zhao and Guibas 2004). Four main components make up the sensor node (Figure 1.2): the power, sensing, processing, and communication (transceiver) units. Depending on the application of the WSN the node may have additional components such as a GPS receiver or mobilizer (which is used in cases where a sensor node has to move from one location to another) (Akyildiz *et al*., 2002).

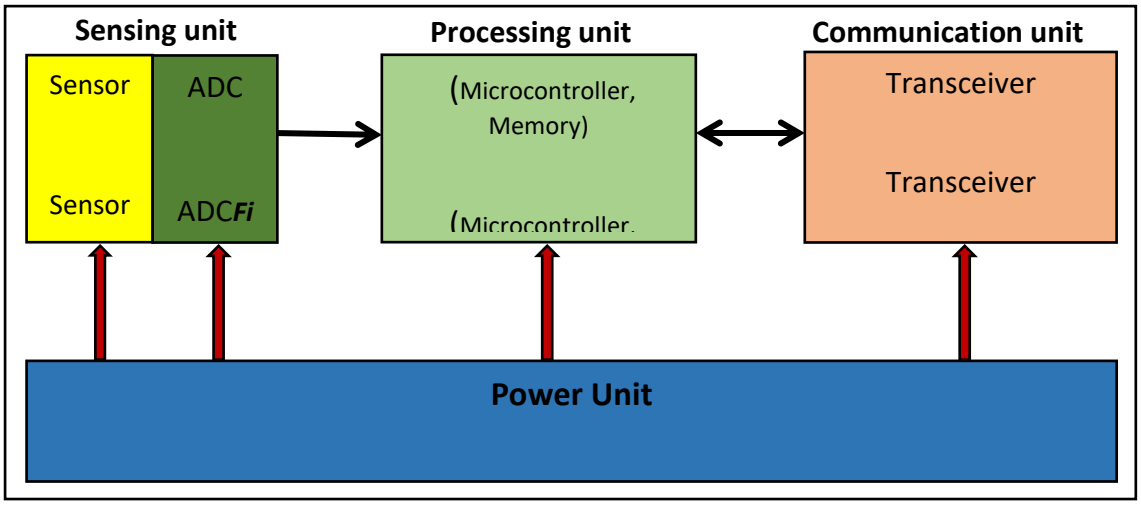


Figure 1 - 2: Structure of sensor node (after Akyildiz et al., 2002).

- The power unit normally consists of a battery to supply power for the other sensor node components, and in many cases the application needs the network to work at least 1 year (Perillo and Heinzelman 2004). Unfortunately, in many cases, battery recharging or replacement is unfeasible especially in large scale deployments with thousands of nodes or if they are placed in a harsh environment, therefore, the efficient usage of the energy is important. Preserving sensor energy is possible through three operational modes for the processor; Active, Idle, and sleep mode (Akyildiz *et al*., 2002).
- The sensing unit is considered as the main component of the wireless sensor node. Sensors convert physical measurements (such as humidity) to analogue electrical signals which are in turn converted to digital domain using Analogue to Digital Converter (ADC) for further processing and transmission. The digital data may be processed locally or transmitted by the communication unit to the sink for additional processing(Xia, 2009).
- The processing unit consists of two sub-units; the microcontroller and the storage unit (memory), and "manages the procedures that make the sensor node collaborate with the other nodes to carry out the assigned sensing tasks" (Akyildiz *et al*., 2002).
- Communication unit enables the sensor nodes to communicate with other nodes and a sink. Short range radio (10 m to 100 m) is often used (Zareei *et al*., 2011). Radio is ideal for sensor node, due to its non-line of sight behaviour (in contrast to optical or infrared which require line of sight) and it is possible to implement low power radio transceivers (of orders of milli-watts, e.g. for Xbee transceiver RF module, the transmitted power is 1-63 mW in case of 2.4GHz). The transceiver commonly uses

Industrial, Scientific, and Medical (ISM) unlicensed bands. Most current nodes take IEEE 802.15.4 as a standard for their radio chips (Healy *et al*., 2008).

1.1.3 Classifications of WSNs

Different criteria can be used to classify WSNs:

- **Static or Mobile Network**: in many applications, the sensor nodes are static; however, some applications need mobile nodes. Mobile nodes can be deployed in any scenario and it can cope with rapid changes in network topology. Moreover, the energy dissipation can be improved by using mobile sink compared to static sink, because the selection of convenient radius for the mobile sink and choosing proper duty cycle for nodes will achieve higher energy efficiency (Khan *et al*., 2013).
- **Deterministic or non-deterministic Network**: this class depends on the deployment of the sensor nodes. In deterministic networks, the nodes positions are pre-planned and are fixed once deployed (Figure 1.3-a). On the other hand, nondeterministic networks the deployment is not pre-planned; the reason is the harsh or hostile environments, (Figure 1.3-b). Non-deterministic is more scalable and flexible since it doesn't need planning work but leads to increased complexity because the nodes require the ability to organize themselves in the network. The deterministic sensor placement is often referred to as controlled placement (Damuut, 2012).

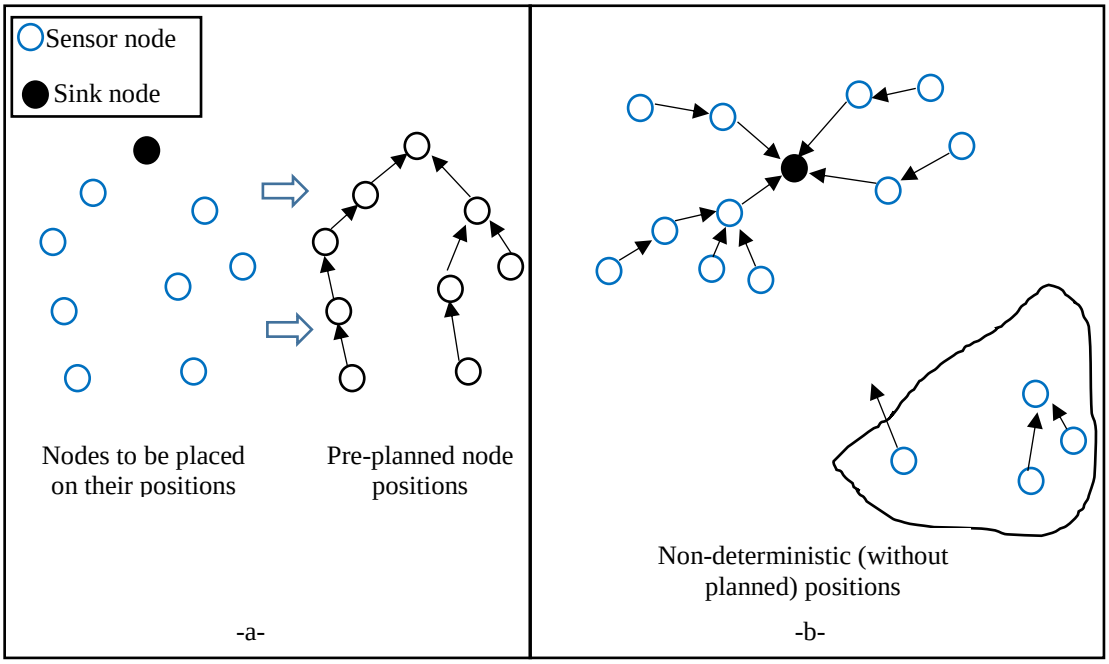


Figure 1 - 3: a- Deterministic network, b- non-Deterministic network.

- **Single-sink and Multi-sink network:** in a single-sink network, sensor nodes send their sensed data to one sink which is close to or inside the sensed region, while in multi-sink network the sensor nodes send their data to the closest sink which can make the traffic more reliable and balanced, since it can reduce the number of hops, (Figure 1.4) (Zheng, 2009).

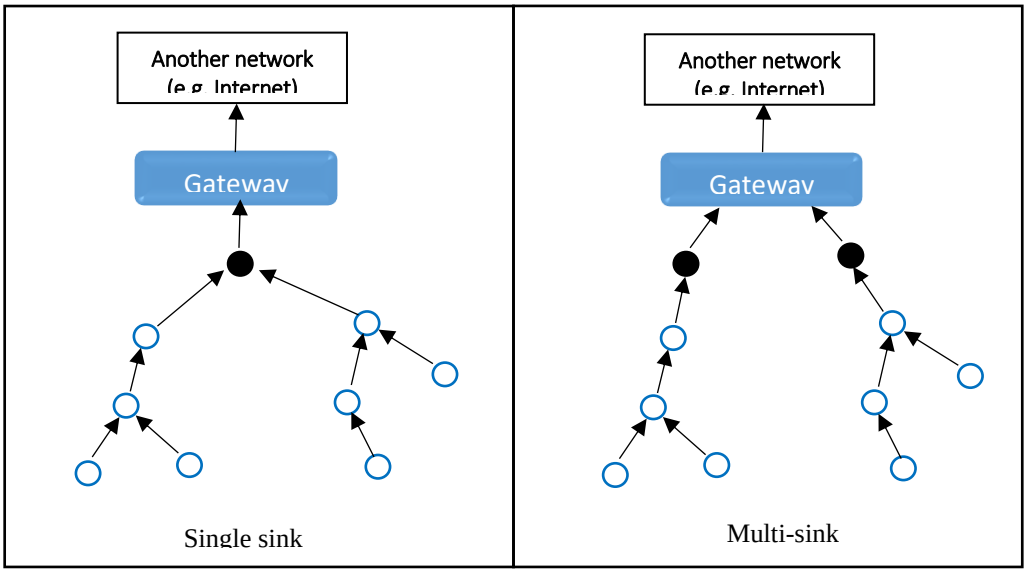


Figure 1 - 4: Single sink and multi-sink network.

- **Single-hop and multi-hop network:** in a single-hop network, the nodes send their sensed data directly to the sink in one hop. This is easy to design but it may consume more energy because the radio range is large and may increase the number of packet collisions at the sink because nodes may try to send data at the same time. In a multi-hop network, the nodes transmit data to the sink through many different intermediate nodes. These intermediate nodes route the data to the sink and may perform data processing to eliminate redundant data, and hence reduce the quantity of the data traffic and thereby improve the power efficiency of the network. Single-hop is more suitable for applications in small sensing areas with small number of nodes, while multi-hop can be used for a wide range of applications where the number of nodes is large (Figure 1.5) (Zheng, 2009).

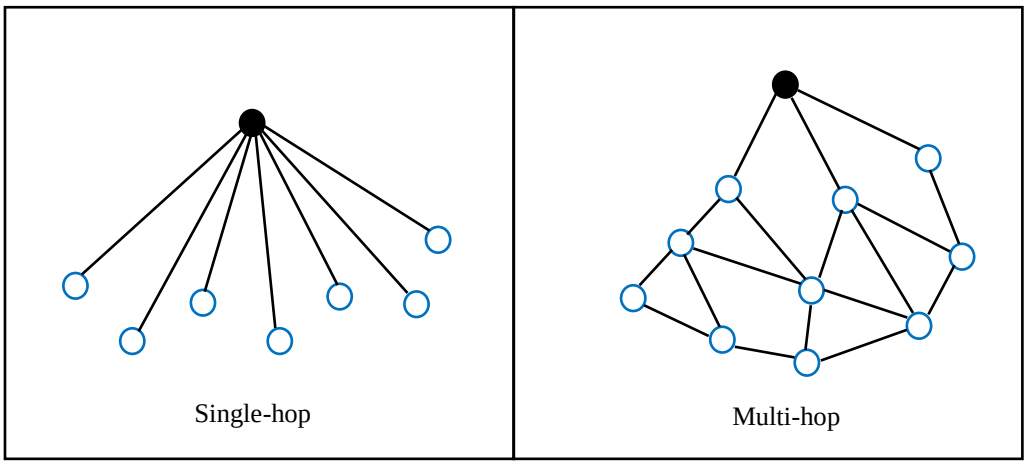


Figure 1 - 5: Single-hop and multi-hop network.

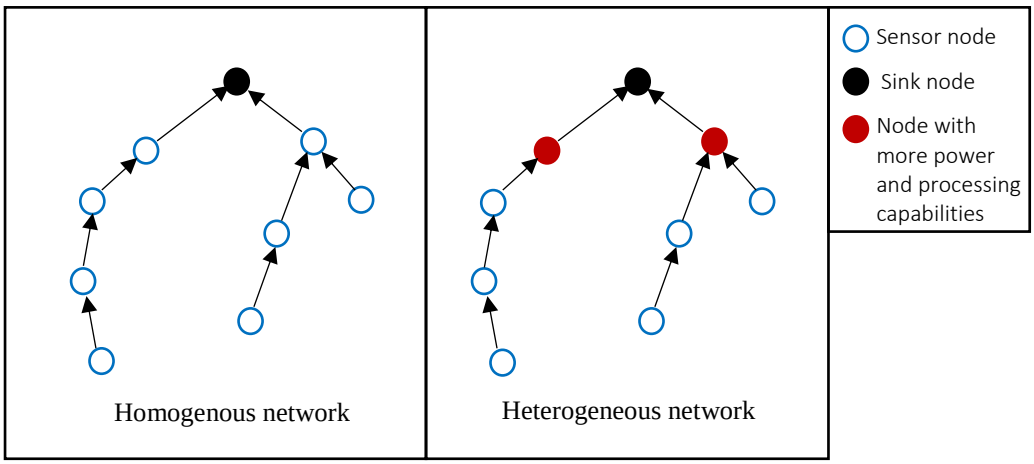


Figure 1 - 6: Homogenous and Heterogenous network.

- **Homogeneous and Heterogeneous Network:** in a homogeneous network, the sensor nodes have the same resources (energy, processing, and communication), while in a heterogeneous network the nodes have different capabilities, e.g. some nodes might have a more sophisticated design or ability, e.g. more energy, communication capability, or processing ability which can lead to enhanced network energy or lifetime, (Figure 1.6) (Uplap & Sharma, 2014; Zheng, 2009).

1.1.4 Design of WSNs

Factors that can influence the design of WSN are listed below (Akyildiz *et al*. 2002):

a. Network Fault tolerance is defined as the ability to keep the sensor network operating without interruption when some nodes fail. The level of fault tolerance required by the

sensor network depends on the application; e.g. if the sensor nodes are deployed in a house to monitor the humidity and temperature levels, the requirement (level) of the fault tolerance is low and flexible, since the sensor nodes in this type of applications are not easily susceptible to damage or noise interference. However, in some other applications such as battlefield surveillance the nodes are more susceptible to being destroyed by hostile actions, therefore, in such cases the level of fault tolerance should be high.

b. Scalability is the ability of the network to change the number of the nodes. The new designed scheme should be able to work and utilize the high density (which is indicated by the number of nodes in a region) nature of WSNs; the density can range from few sensor nodes up to few hundreds. In addition, the density of the nodes depends on the application in which the sensor nodes are deployed.

c. Production cost, the cost of the node should be minimized, since sensor networks can consist of up to thousands of nodes so a small extra charge for each node will be a large cost for the network.

d. Hardware constraints, as mentioned previously, the sensor node is comprised of four subunits: sensing, processing, transceiver, and power. Since the size of sensor nodes (depending on the application) range from shoe-box to match-box size (Poslad 2009), and may be in some cases smaller than even a cubic centimeter (Lu *et al*. 2014), all the subunits need to fit into a relatively small size.

e. Topology, the deployment of high numbers of nodes requires extremely high care of topology maintenance because the nodes are prone to frequent failures which makes the maintenance a challenging task. Node topology depends on the steps of node deployment, in the pre-deployment step, the topology is affected by the deployment method, e.g. dropping from a plane, or by placing one by one either by human or robot. In the post-deployment step, the topology changes are due to changes in the position of the nodes, their residual energy, or malfunctions. The topology will also change when new nodes are deployed to replace the failed nodes or due to changes in task dynamics.

f. Power consumption, nodes usually have a limited power source (e.g. <0.5 - 3 Ah, 1.2 – 3.7V) for example, wireless micro-sensor node in health applications uses less than 500mAh and 1.2V, and Waspmote sensor node uses 23mAh and 3.7V, and in some applications the replacement or recharging of the battery might be impossible due to

harsh environment (Akyildiz *et al*., 2002). In a multi-hop network, the failure of a node will affect the network topology and might require a re-routing of packets and a re-organization of the network. Hence, these conditions increase the importance of power conservation and power management. For these reasons researchers have designed new protocols and algorithms to save the power of the node. The consumption of the power is distributed between sensing the data, local computation and processing, transmitting the data to base station. The sensing and processing operations usually consume extremely low power, while the transceiver chip is normally consuming more power (in transmit and receive), therefore, most protocols concentrate on decreasing the power consumption during transmission (Akyildiz & Vuran 2010; Alkhatib *et al*. 2013).

1.1.5 WSN topologies

WSNs can be deployed and communicate in different ways. A brief description of some example topologies is outlined and presented below (Matin and Islam, 2012):

- *Star topology* has a single base station, with a single hop between the remote nodes and the base station (Figure 1.7). Communication between the remote nodes is not permitted. Star topology is simple and can minimise power consumption at the remote nodes, and results in low latency communications between base station and remote nodes. On the other hand, the drawback of this topology is that the base station must be within the radio range of all the nodes.

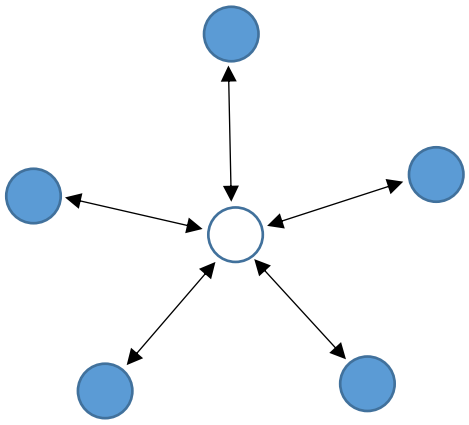


Figure 1 - 7: Star topology of WSN.

- *Mesh topology* the node transmits data to other nodes within its communication range, (Figure 1.8). This is called multi-hop communication, which help to transmit (forward)

data from one node to a node out of its range by using intermediate node. The advantages of this topology are: a) in the case of node failure, the remote node is still able to transmit its data by using another node in its range to forward the data to the desired location. b) the network can be extended simply by adding new nodes. The disadvantages are: a) the power consumption of intermediate nodes that perform the multi-hop operation is higher than the nodes don't have this capability. b) Due to extra hops the time to deliver the message to the destination delay is increased.

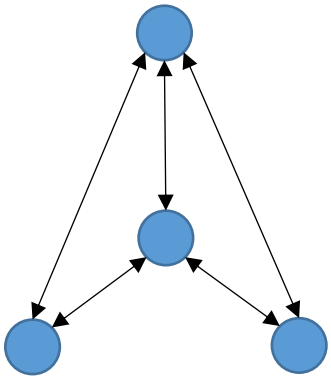


Figure 1 - 8: A Mesh topology of WSN.

- *Hybrid Star-Mesh topology* combines star and mesh networks to provide a robust and versatile communication network; also keep a minimum power consumption of the sensor nodes, Figure 1.9. The ability to forward messages by the nodes with lowest power is not enabled in this topology, then the minimal power consumption will be maintained. However, multi-hop capability with some nodes will allow the messages to be forwarded through the low power nodes to the sink node (base station).

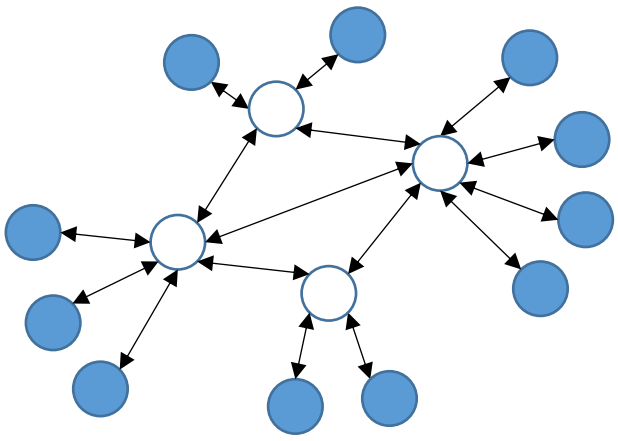


Figure 1 - 9: A Hybrid Star-Mesh WSN topology.

1.1.6 WSN Standards

The Institute of Electrical and Electronics Engineers (IEEE) 802 standards, which are the most important standards for wireless networks, include a family of networking standards dealing with local area networks and metropolitan networks. This standard allows networks to carry variable-size packets. The Physical and Data link layer and the higher layers of the Open System Interconnection (OSI) model are the operation area of IEEE802(IEEE, 2002). The committee of IEEE802 has developed many related standards ranging from IEEE.802.1 to IEEE802.25. Figure 1.10 shows how some of these standards relate to difference wireless networks sizes (Arzubi *et al*., 2013).

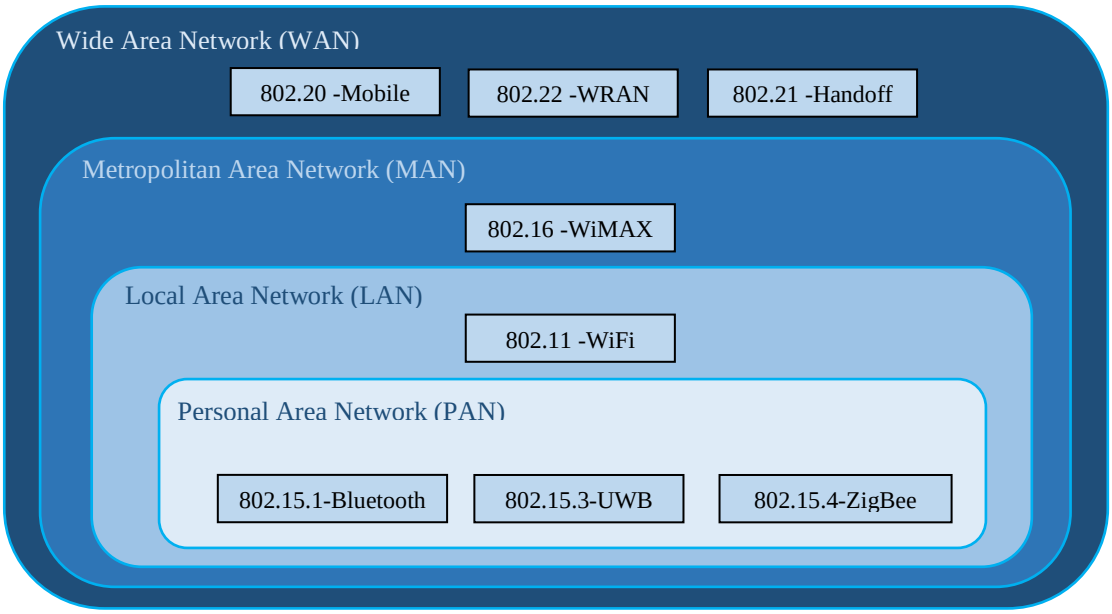


Figure 1 - 10: IEEE802 standards in different wireless networks technologies

The IEEE802.11 WiFi (Wireless Fidelity) standard is a technology for LAN using the 2.4GHz and 5.8GHz radio bands. Any device conforming to this standard can exchange and transfer data wirelessly over the network giving the rise to high speed internet connections(Banerji & Chowdhury, 2013).

The Personal Area Network (PAN) is a network for devices (computer, telephone, and personal digital assistants) that are close to one person (with ranges of meters). Examples of technologies for this topology are IEEE802.15.1 (Bluetooth), IEEE802.15.3a (Ultra-Wideband), and IEEE802.15.4 like ZigBee. These standards are those that are typically used in WSNs.

The proposed standard for wireless sensor networks is IEEE 802.15.4, which focuses on the low cost of deployment, low complexity, and low power consumption. IEEE802.15.4 defines the characteristics of the two lower OSI layers (physical layer and MAC layer). The IEEE802.15.4 physical layer supports different frequency bands for communication: 2.4GHz (worldwide), 915MHz (USA), and 868MHz (Europe). Table 1 summarizes some of the IEEE802.15.4 physical layer characteristics (Libelium, 2014). In Table 1.1, TX power is the required power for transmission, Sensitivity is the smallest signal power that the node is able to detect, and channels is the number of channels available. The MAC layer control accesses the radio channel using a Carrier Sense Multiple Accesses /Collision Avoidance (CSMA/CA) mechanism (Suriyachai *et al*., 2012). The MAC layer is also responsible for frame delivery, network synchronization, network access, and secure services (Yick *et al*., 2008).

Table 1.1: IEEE 802.15.4 Physical layer characteristics.

Frequency band	**TX power (**mW)	**Sensitivity (**dBm)	**Channels**
2.4 - 2.4835GHz (IEEE802.15.4)	63.1	-100	16
2.40 – 2.70GHz (ZigBee)	50	-102	14
902.0 - 928.0MHz	50	-100	10
868.0-868.6MHz	315	-112	1

The ZigBee standard is built on IEEE802.15.4 and defines the higher layer protocols. A Mesh topology is used for ZigBee devices to form networks of hundreds to thousands of devices. There are three types of nodes in the ZigBee standard: ZigBee coordinator, which is the device that initiates the network and stores information, and can bridge networks, ZigBee router which is used to link groups of nodes together and provide multi-hop communication, End device that consists of sensors which communicates with the router to transmit information to the coordinator.

1.2 Background of the problem

In event-driven applications of WSN (Section 1.1.1) such as fire detection, chemical explosion detection, leaking gas detection, target tracking, and structural health monitoring, when an event occurs a large amount of data can be generated which needs

to be transmitted reliably as widely as possible to the base station. Due to the constraints of WSNs (Section 1.1), the high generation rate of the burst of data which is usually uncontrolled, may exceed the available capacity at points in the network and this will lead to congestion. “Congestion in WSNs, is created at two levels: node-level congestion (or buffer overflow) and link-level congestion” (Ghaffari, 2015). In node level congestion this generally occurs at nodes that are closer to the sink when the packet arrival rate is higher than the packet service rate. Thus, this eventually leads to buffer overflow and the loss of packets. On the other hand, congestion at link-level (i.e. the link between two nodes) happens when large amounts of packets are sent on the link, this will result in competition, collision and bit error.

1.3 Problem statement

There is a need to develop an algorithm to mitigate the consequences of congestion in event driven WSNs. Network congestion causes data packets to be corrupt and increases the time for the packet to travel from the source to the sink (end to end delay). In addition, the retransmission of the lost packets consumes more power. Dropped packets are a major defect in WSNs, because they lead to wasted energy. This can impact on the longevity of the network: the most significant resource as it can be impossible to recharge or replace the battery due to deployment in a harsh environment (Adinya, 2012). If congestion is not controlled, the congested nodes will quickly lose their power and a dead path will appear (Sergiou, 2012).

1.4 Aims of the study

1. To test the congestion control algorithm, Hierarchical Tree Alternative Path (HTAP) experimentally. The HTAP algorithm has been reported in the literature (Sergiou et al., 2013) but the performance has only been simulated and there is a need to test this on a hardware implementation.
2. To improve the performance of the HTAP algorithm in the case of the existence of single path in the network topology. The improvement is by applying the technique of combining the resource control (alternative path) and the rate control to mitigate the congestion in WSN.

Chapter 2: Literature Review

To verify the experimental implementation of the congestion algorithm and its improvement efficiently in the WSNs of this research, this chapter contains a comprehensive review of the literature from a general discussion of general protocols for routing and traffic-shaping then to the field of congestion protocols of WSNs. Firstly, the WSNs routing protocols are introduced with some details, then some protocols of traffic shaping will be illustrated. Secondly, the congestion problem and its types in WSN are presented in detail. Finally, the congestion protocols of WSNs are investigated sequentially to identify the addressed knowledge gap of this research.

2.1 Routing and traffic shaping WSNs protocols

Different applications of WSNs employ routing protocols to improve their performance demands. Also, in this thesis the concentration is on mitigating WSN congestion by using alternative path at the event of congestion, therefore, a review of important WSN routing protocols is a good starting point. In addition, in this section some of important traffic shaping protocols will be presented with some details.

2.1.1 Routing protocols

Routing protocols are to route data from source to destination in reliable and efficient energy manner in order to increase the lifetime and improve the overall performance of the network. Routing protocols os WSNs have the following features:

- Sensor nodes do not have Internet protocol (IP) addresses as in the traditional networks; therefore, IP-based routing protocols cannot be applied to WSNs (Kumar *et al*., 2017).
- The design of protocols needs to be scalable.
- It should manage the communication among many nodes and propagate the data to destination.
- The protocol should meet the resources constraints of the node (limit power, low-bandwidth, low storage).
- Issues such as efficiency, fault tolerance, fairness, and security, should be achieved under the design of the network protocols (Villalba *et al.*, 2009).

The most important of the existing routing protocols are presented in some detail:

An Ad Hoc On-Demand Distance Vector (AODV) is a routing protocol designed for wireless and mobile ad hoc networks. The routes are established by this protocol to destinations on demand and supports both unicast and multicast routing. In this protocol, a route from source to destination is found when the source broadcasts a Route REQuest packet (RREQ). The RREQ will be relayed from neighbor to neighbor until the RREQ reaches the destination or to find a new route information from another intermediate node. At the time of first copy of RREQ is received by the destination or the intermediate node, a Route REply Packet (RREP) will be sent to the source node along the path from which first copy is sent. The route in AODV is built on demand and is not updated until the route breaks of times out (Tang & Zhang, 2004).

Marina & Das propose Ad-hoc On-demand Multipath Distance Vector Routing (AOMDV) which is an extension to AODV. AOMDV protocol is computing multiple loop-free and link disjoint paths. Computing the multiple paths during route discovery is the main idea of AOMDV protocol. AOMDV is designed to solve the issue of link failures and route breaks that occur frequently in highly dynamic ad hoc networks. The inefficiency of route discovery that is needed in response to every route bread can be avoided by the availability of redundant paths, and the new route discovery is only needed when all paths to the destination are broken. Since AOMDV is a multipath routing protocol which needs more flooding message and then more message overheads during route discovery (Marina & Das, 2001).

Optimized Link State Routing (OLSR) is a proactive routing protocol for mobile ad hoc networks, and it is best suitable for large and dense ad hoc networks. The link state information is discovered by OLSR by using hello and topology control messages and then the link state information disseminates throughout the mobile ad hoc network. The OLSR employs periodic exchange of messages to maintain topology information of the network at each node. Multipoint relaying technique are used by OLSR for efficient and economic flooding of the control messages (Jacquet *et al*., 2001).

Greedy Perimeter Stateless Routing (GPSR) a routing protocol that make the packet forwarding decisions by using the positions of routers and a packet's destination. In the network topology the information about a router's immediate neighbours are used by GPSR to make greedy forwarding decisions. If a greedy forwarding is impossible in a region then the algorithm recovers by routing around the perimeter of the region. GPSR uses local topology information to find correct new routes quickly in the case of topology changes of mobile networks (Karp & Kung, 2000).

Previous Hop Routing (PHR) is a protocol that makes the forwarding decision on a packet-by-packet basis based on exploiting previous hop information and distance to destination. This protocol is used in highly dynamic network where the life time of a hop-by-hop path between source and destination nodes is short. PHR employs probabilistic forwarding to save network resources in the event of high network loads (Ali., 2018).

2.1.2 Traffic shaping protocols

Traffic shaping is a technique that manages the network bandwidth which delays some or all packets to bring them into compliance with a desired traffic profile. The traffic shaping technique is used to:

- Optimize or guarantee performance
- Improve latency
- Increase usable bandwidth for some kinds of packets by delaying other kinds.

Some of important traffic shaping protocols used in WSNs are illustrated in this sections with some details:

Kazemian (2009) proposes an intelligent video streaming which is a video transmission mechanism that works over ZigBee and Bluetooth. The unnecessary excess of MPEG multimedia stream over the ZigBee channel was avoided by a developed traffic shaping queue. A Neural-Fuzzy (NF) system was introduced by adjusting traffic shaping queue to remove intolerable delay of the VBR encoded multimedia and to confirm the data to the token bucket's convention prior inflowing to the ZigBee channel (Kazemian, 2009).

Alam *et.al.* (2000) proposed Traffic shaping for MPEG video transmission over the next generation Internet. Based on a token bucket mechanism the authors developed a systematic method of the traffic shaper which is studied the transmission specifications of MPEG compressed video streams over the guaranteed service (GS) (Alam *et al.*, 2000).

Lie et.al. (2008) proposed Evalvid-RA: trace driven simulation of rate adaptive MPEG-4 VBR video. A solution for generating real adaptive MPEG-4 streaming traffic is presented by the authors; this solution is used the quantized scale for regulating the transmission rate. During the simulation run-time, a feedback based VBR rate controller is used which supported TCP-Friendly Rate Control (TFRC) and a proprietary congestion control system named P-AQM (Lie & Klaue, 2008).

2.2 Congestion in WSNs

WSN is simply defined as a group of devices called sensor nodes that work together to form a wireless network. These nodes transfer environmental information from the monitored area to the sink in one or multiple hops through wireless links communication. However, due to the resource constraints of WSN, the number of deployed nodes, event-driven nature of WSNs, many to one communication, and the high traffic of sensor nodes, WSNs in some applications like event driven application face significant challenges due to huge amount of transmitted data from the nodes in the event place, this large amount of data might lead to network congestion, which can cause a plethora of malfunctions such as: decrease in the network throughput and energy efficiency, packet loss, increase delays, and reduced network lifetime. The dropped packets need to be retransmitted which in turn more energy will be consumed, these mentioned problems will impact the QoS (Flora *et al*., 2011; Ghaffari, 2015).

Generally, WSN congestion is classified in two major categories:

- **First category:** is based on how the packets are lost, which is divided into 2 types (Kang *et al*., 2007): The node- level congestion (buffer overflow) and the link-level congestion, Figure 2.1.

*1-Node level congestion***:** When the packets, that need transmission, overflow the buffer of a particular node the congestion of node level happens. The overflow that causes the congestion is happened when the packet arrival rate is higher than packet service time (Ghaffari, 2015). The impacts of this type of congestion are: packet loss, increased queuing delay. The packet loss need retransmission which in turn consumes additional energy, and hence decreasing the network lifetime (Flora *et al*., 2011).

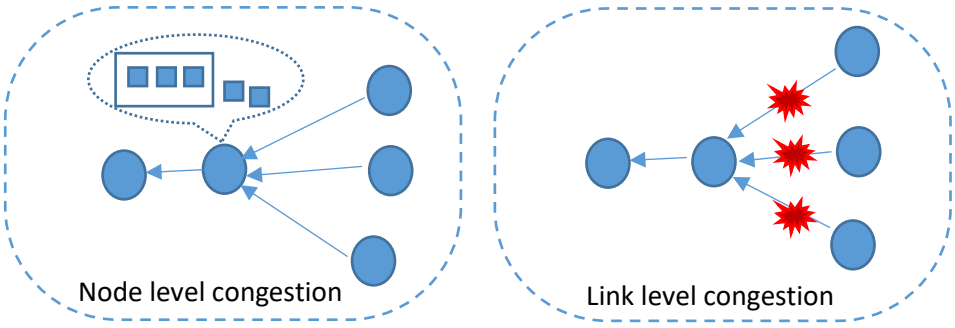


Figure 2- 1: presents common congestion in WSNs, (after Wang & Li, 2007).

2-Link level congestion: If multiple active sensor nodes share the same wireless channel and each one of them tries to send packets in the same transmission medium, the collision might occur and the transmitted packets fail to reach next hop because of the occurred collision between the packets, this known as the link level congestion. The

impact of the packets collision is the increase in the packet service time and a decrease in the link utilization. Medium Access control techniques such as Carrier Sense Multiple Access (CSMA) and Time Division Multiple Access (TDMA) are used to prevent the collisions (Michopoulos, 2012; Wang & Li, 2007; Zhang *et al*., 2012).

- **Second category:** is based on the congestion place (hotspot) in the network, there are 3 typical hotspot scenarios, these hotspots are listed below (Fang *et al*., 2010; Sergiou, 2012):

  1- Source hotspot: this hotspot is created around the occurred event when the event is detected by densely deployed sensors whose sensing range cover the event spot, Figure 2.2. Sending control messages from congested nodes to source nodes would be effective to control the traffic rate and to change the sending route.

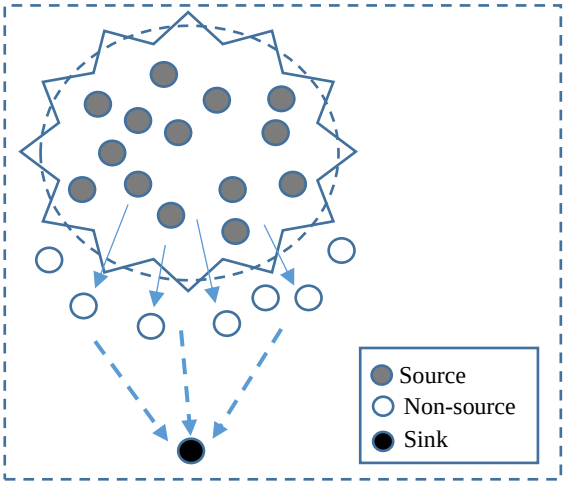


Figure 2- 2: Hotspot near sources generating event, (after Fang et al., 2010).

  2- Hotspot near the sink: the traffic intensity of the data packets is normally increased near the sink, this because of many-to-one and multi-hop nature of data transmission in WSN, Figure 2.3. This increment in the traffic convergence near the sink leads to network congestion. The combination of packet dropping and sending control messages would be an effective way of mitigating this hotspot congestion. Another effective way is the use of multiple sinks; therefore, the load would be balanced between the uniformly scattered sinks.

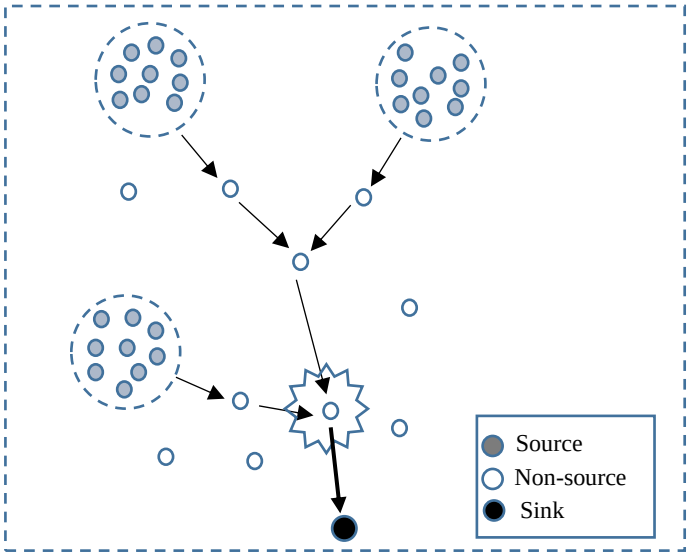


Figure 2- 3: Hotspot near sink - convergence hotspot, (after Fang et al., 2010).

3- Intersection hotspot: the traffic intersection can happen in the network because of the presence of multiple sinks, Figure 2.4. The intersection nodes can become congested hotspot due to overflows of the aggregate traffic load (Das & Dutta, 2005).

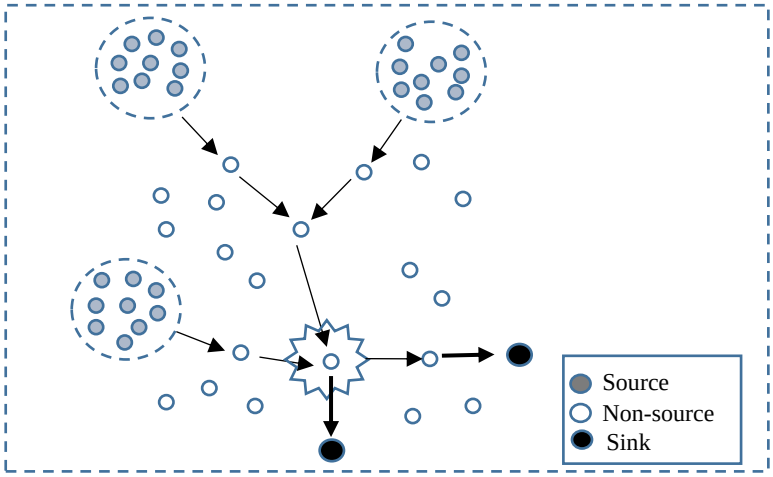


Figure 2- 4: Intersection hotspot with the presence of multiple sinks, (after Fang et al., 2010).

The mentioned types of congestion have negative effect on network performance, such as decreasing network throughput, increasing time delay and energy consumption. Therefore, to enhance the network performance an appropriate congestion algorithm should be designed to control the occurred congestion. The design of congestion control is based on two aspects: congestion avoidance and congestion control. Avoiding congestion can be occurred through many ways: the nodes should be deployed properly, reasonable network topology, network application should be chosen suitable to the network technology, and the network resources should be provided sufficiently (Yuan *et*

al., 2014). In this context, congestion control to decrease congestion in WSN, consists of three phases: 1) Congestion detection, 2) Congestion notification, and 3) Congestion mitigation (Ghaffari, 2015).

2.2.1 Congestion detection

The existence and location of congestion in WSN are detected through the process of congestion detection. In this process, detecting WSN congestion is by using different parameters in congestion control protocols: i) Buffer occupancy, ii) Channel load, iii) combination of buffer occupancy and channel load, iv) Packet service time (M. Chatterjee et al., 2010; Lee & Chung, 2010; Paek & Govindan, 2010).

i) **Buffer occupancy**: this parameter also known as queue length, the incoming data packets are usually stored in sensor node buffer. Thus, WSN congestion is indicated by the occupancy of this buffer. In many algorithms, congestion is detected when buffer occupancy of sensor node exceeds a constant threshold for queue length (Antoniou & Pitsillides, 2010; Chen & Yang, 2006; Jenn-Yue Teo *et al*., 2008).
ii) **Channel load**: this parameter is used by some protocols when the time for transmission of a data packet in the wireless medium exceeds a predefined threshold, then congestion is detected (Vedantham *et al*., 2009; Yin *et al*., 2009).
iii) **Buffer occupancy and wireless channel load**: the congestion is detected either in the buffer or in the wireless medium (Hull *et al*., 2004; Cy Wan *et al*., 2003).
iv) **Packet service time**: packet service time and packet inter-arrival time or the combination of both of them are used by some protocols to detect congestion in WSNs (Ee & Bajcsy, 2004; Sohraby & Lawrence, 2006).

Table 2. 1: Congestion detection parameters and their mechanisms.

Detection parameter	Detection mechanism
Buffer occupancy	Buffer occupancy exceeds a constant threshold
Channel load	Transmission time exceeds a predefined threshold
Buffer occupancy + channel load	Same as above
Packet service time	Time difference between packet arrival and the transmission time.

2.2.2 Congestion notification

To inform upstream nodes (the nodes those sending packets to congested node) about the detected congestion, the congested node is notifying them either explicitly or implicitly. The notification might be as small as a single bit in the packet header known as Congestion Notification (CN) bit (Enigo, 2009; Hull *et al*., 2004; Iyer *et al*., 2005; Yogesh Sankarasubramaniam *et al*., 2003).

- **Explicit congestion notification:** The notification in this type is by transmitting control packets from the congested node to upstream nodes (Tezcan & Wang, 2007; Cy Wan *et al*., 2003) to inform them about occurred congestion to take the required action. Since the medium is already congested, therefore sending control packets as notification messages well add more load to the congested wireless medium. Based on this fact, the explicit congestion notification has not employed by many congestion algorithms.
- **Implicit congestion notification:** in this method, no extra packets are added to the already congested network. The congested node is piggybacking the notification bit in the packet header (or in ACK packets) to inform other neighbour nodes about its congestion state. Due to avoid using further data for notification in this method, a large number of protocols employed this method in their congestion control design (Akan & Akyildiz, 2005; Kang *et al*., 2007; Sohraby & Lawrence, 2006; Tao & Yu, 2010).

2.2.3 Congestion mitigation

The proposed congestion control protocols of WSNs are different in the way they detect, notify, and mitigate congestion. These congestion control protocols are classified into 4 categories:

- **Traffic control:** in this category, when the data load that is injected in the network is near or exceeds the network capacity then congestion has occurred, and the mechanism of traffic control will be used to alleviate network congestion. Traffic control techniques are classified into two categories: Additive Increase Multiplicative Decrease (AIMD) and rate-based technique. In the former technique of AIMD, the sender increases its traffic rate linearly (by growing its rate 1packet in each Round Trip Time (RTT)) until the congestion takes place (the packets start to drop) the sender will decrease its rate exponentially, this reduces the congestion

multiplicatively and which makes for next transmission that the possibility of congestion is pretty less and the overall congestion in the network is sustainable (Akan & Akyildiz, 2005; K. Li *et al*., 2013; Vuran *et al*., 2010). On the other hand, the latter technique of rate-based technique is mitigating congestion by decreasing the rate of source nodes. However, reducing the traffic rate can have an impact on the valuable data packets carrying information in event-based applications, therefore, for event-based application this is not an efficient technique for mitigating congestion.

- **Resource control:** as mentioned previously, that reducing the data rate might eliminate valuable data, particularly in even-based application or critical-mission applications which need every packet to be delivered to network sink. As a result, many congestion control protocols use alternative technique known as resource control approach, which is based on using uncongested or idle paths to send the data through it to the sink when congestion is occurred (Antoniou *et al*., 2013; Kang *et al*., 2007). Sending data packets by using alternative path at the event of congestion to the sink in this technique, is improving the packet delivery rate comparted to first technique of traffic control.
- **Priority-aware congestion control scheme:** in this scheme, the protocols use the MAC techniques to give priority for congested node to access the used channel, then the congested node by taking this priority will alleviate its congestion by sending its data to the sink (Ghaffari, 2015).
- **Queue-assisted technique:** in this technique, the queue length of the node is used to mitigate the occurred congestion. The protocols employed this technique observe the queue length of the nodes and then reduce the rate such as using AIMD to decrease the queue length lowest possible length (Ghaffari, 2015).

Some of common metrics should be checked in each protocol to appraise the performance of protocol. The most common metrics are listed below:

- **Packet Delivery Ratio:** this parameter is mostly used metrics in WSNs, which is the ratio of the delivered packets to the generated packets. The algorithm is efficient if the packet delivery ratio is nearly 100%.

Table 2. 2: Congestion mitigation categories.

Mitigation category	Mitigation mechanism
Traffic control	Decreases the data rate at the event of congestion
Resource control	Uses idle paths or uncongested paths at the event of congestion
Priority-aware congestion control	Gives priority for congested nodes to access the channel
Queue-assisted technique	Observes the queue length and reduce the rate to decrease the queue length to mitigate occurred congestion

Drop-Burst Length (DBL): is a novel measurement of the probability of drop a consecutive number of packets in each route. It is the distribution of the lengths of the packet group drops. The packet drop impact on the network performance is measured with this proposed metric. There are two types of DBL as in Figure 2.5, short DBL such as one or two consecutive packets which can be coped with some applications. The second type of DBL is the long DBL, this type has bad influence on application performance (Ali *et al*., 2017).

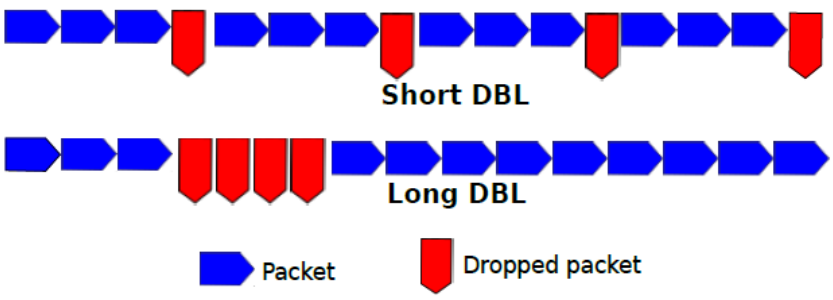


Figure 2- 5: Short and Long DBL (Ali et al., 2017).

Throughput: is the number of successfully received packets by the sink per unit time, for efficient algorithm this parameter should be high.

Packet loss rate: is the rate of lost or dropped packets due to buffer overflow divided by the total number of packets delivered to the network sink.

Fairness: is the degree of variation in data sending rate. The desirable algorithm is achieving fairness in all the source nodes that transmitting packets.

End-to-End delay: is the time taken by the packet from it was generated by the source to the time it reaches the sink.

Energy consumption: it is the measurement of consumed energy by the sensor nodes, when the nodes send, receive, and forward packets. The algorithm is efficient when the energy consumption is low.

2.2.3.1 Resource control protocols

The important and popular protocols of resource (using alternative path) control mechanism of congestion control in WSNs are listed below with some detail:

Kang *et al*. (2007) proposed Topology Aware Resource Adaptation (TARA), with the aim of adapting additional resources of the network and in the case of congestion TARA alleviate intersection hot spots. Detecting congestion in TARA is by observing the buffer occupancy and the channel load. To alleviate congestion, TARA is using two types of nodes: distributor and merger. The former is distributing the traffic originating from the hotspot between the original path and the detour path, while the latter merges the two flows. The traffic is deflected from the hotspot through the distributor node along the detour and reaches the merger node in the case of congestion is occurred (Kang *et al*., 2007).

Flock-based Congestion Control (Flock-CC) is proposed by Antoniou *et al*. (2013) to control congestion based on birds' behaviour, this design is robust, scalable and self-adaptive congestion control for WSNs. The swarm intelligence paradigm that inspired by the collective behaviour of bird flocks is used by Flock-CC. To avoid congestion (obstacles), the packets (birds) are guided form flocks and flow towards the sink (global attractor, as in Figure 2.6.

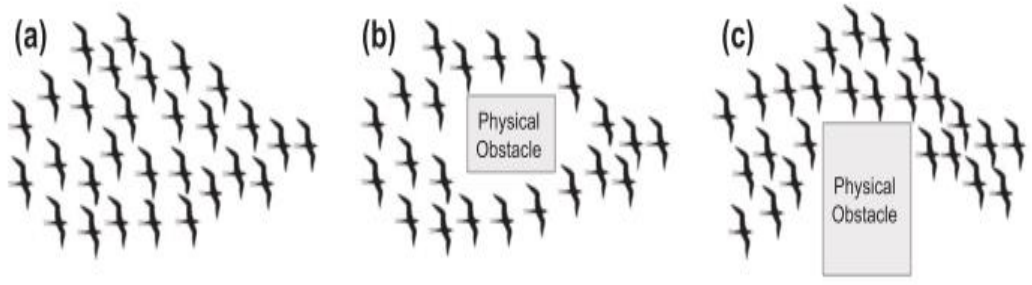


Figure 2- 6: The behaviour of birds avoiding obstacles: a) a flock of migrating birds moves through an obstacle-free environment, b) 2 sub-flocks split due to existence of obstacle, c) the flock reformed along the obstacle (Antoniou et al., 2013).

The Flock-CC scheme balances the existing load by moving the packet to the sink and by exploiting the available network resources. Furthermore, Flock-CC is easy to implement at individual nodes (Antoniou *et al*., 2013).

Tian He *et al*. (2003) proposed SPEED protocol which is a real-time communication protocol for sensor networks. SPEED protocol provides 3 services of the real-time communication which are they: 1) real-time unicast, 2) real-time area-multicast, and 3) real-time area-anycast. By diverting the traffic through multiple routes and regulating the sending rate the end-to-end packet delay becomes proportional to the distance between the source and destination. The congestion indication is estimated by using the single hop sender delay (Tian He et al., 2003).

Traffic-Aware Dynamic Routing protocol (TADR) is proposed by He *et al*. (2008) to remove congestion and to enhance the throughput, the TADR uses the idle or under-loaded nodes to alleviate the occurred congestion. By using multiple paths, the TADR routes the packets around the congestion areas. The congestion is detected by employing the queue length. TADR protocol choses the shortest paths when there is no congestion, but when there is congestion, the multiple paths will be picked by TADR, and the excess of packets will be forwarded through these paths (T. He et al., 2008).

Kumar *et al*. (2008) proposed Congestion-Aware Routing (CAR) protocol, CAR discovers the congested zone between sources and the sink, in this zone the high-priority traffic is forwarded towards the sink. Multipath forwarding is used by CAR to forward the High Priority (HP) and the Low Priority (LP). The short path is used for routing HP

traffic, whereas the longer path with uncongested nodes is used for routing the LP traffic (R. Kumar et al., 2008).

Rahman *et al*. (2008) proposed Quality of Service adaptive cross-layer protocol, in which the sink assigns different priorities to the sensed data according to their priorities. By measuring the QoS requirement of the packet the nodes in this protocol send their data packets to appropriate next hop, and each node considers a primary route and at least one alternative route, also based on the differences of the acquired event the priorities is disposed. The congestion indication in this protocol is the ratio of the average packet service rate and packet scheduling rate. Implicit congestion notification is used by overhearing, then the real time-traffic will be split between the routes to alleviate the occurred congestion (Rahman *et al*., 2008).

Siphon proposed by Chieh-yih Wan *et al*. (2005) is a set of fully distributed algorithms that control the congestion in overload conditions. The typical assumption considers that all nodes are equal except the sink in reacting to the onset of congestion, the application fidelity measured at the sink degrades because the congestion control at the sources and intermediate forwarding nodes is to rate control the traffic or even drop event packets during periods of transient and persistent congestion. Therefore, to address this problem a small number of all-wireless multi-radio virtual sinks are randomly or selectively placed (e.g. on the edge of the network to siphon off the data events from regions of the sensor fields that are beginning to show signs of overload) are capable of offering overload traffic management services to the existing low-power sensor network. The proposed algorithms support: 1) discover and select the virtual sink from the dual-radio virtual sinks, 2) detect congestion, and 3) redirect the traffic in the sensor network. At the event of congestion, the virtual sink forms a secondary ad-hoc network and the congested route will be directed to the physical sink to maintain the rate of events and to avoid the throttling or deleting packets. The notification of congestion is achieved by the node which enables a redirection bit in the packet header (Chieh-yih Wan *et al*., 2005).

The TALONet scheme that proposed by Huang *et al*. (2009) is a power efficient grid based congestion avoidance scheme. This scheme consists of three phases: 1) Network formation, 2) Data dissemination, and 3) Framework update. During the first phase the sink sends to each sensor node information about the sinks location and the square grid

topology, then the nodes can then operate as TALON or normal nodes. In the second phase, the packets are forwarded by normal nodes to their closest TALON nodes until these packets reach the sink. In last phase, to preserve energy, the control message will be broadcasted periodically to every node in the network then a new network formation will be resulted (J. M. Huang et al., 2009).

In Table 2.2, a comparison of the above-mentioned protocols related to resource control.

Table 2. 3: comparison of resource control protocols.

Protocol	Congestion detection	Congestion notification	Application type	Evaluation type
TARA (Kang *et al*., 2007)	Queue length + channel load	Feedback messages	Continuous	Simulation
Flock-CC	Queue length	Feedback messages	event	Emulation
SPEED (Tian He *et al*., 2003)	Packet delay of single hop	MAC layer feedback	Real time	Simulation + experimentation
TADR (T. He et al., 2008)	Queue length		Event	Simulation
CAR (R. Kumar et al., 2008)	Competition between HP and LP packets		Event	Simulation
QOS adaptive cross layer (Rahman *et al*., 2008)	Packet service ratio	Information in header	Continuous	Simulation
SIPHON	Queue length + channel load	Bit in the header	Event	Simulation + experimentation
TALONET	Queue length	Information in header	Continuous	Simulation

2.2.3.2 Traffic control protocols

In the case of traffic demand is near or exceeds the capacity of network resources, this leads the congestion to be occurred, thus a traffic control mechanism is required to control the injected data load. Two approaches are related to traffic control: window-based and rate-based. In first approach, the sender probes for the available network bandwidth by slowly increasing a congestion window, and if the congestion is detected the protocol will reduce the congestion window greatly. To avoid network collapse, an essential rapid reduction of the window size in response to congestion is needed. In second approach, the rate-based attempt to estimate the available network bandwidth explicitly. The rate adjustment is achieved by using throughput formula or empirically derived directives.

In this section, the review of most important traffic control protocols is presented in detail, with the advantage and disadvantage of each mentioned protocol.

The Adaptive Rate Control (ARC) protocol is proposed by Woo & Culler, (2001), the aim of this protocol is to monitor the injected packet to traffic stream as well as route-through traffic. In ARC, the increase or decrease of the intermediate node rate depends on the received feedback, and if the feedback reveals that the previously sent packet are forwarded to the sink then the intermediate node will increase its rate by a factor α wherw $\alpha > 1$, otherwise the intermediate node will decrease its rate by a factor β, where $0 < \beta < 1$. The ARC protocol doesn't employ the explicit notification for congestion, also congestion is detected implicitly, thus the ARC protocol avoids using control messages. One of the ARC disadvantages is the effect of employing the rate adjustment which introduces packet loss.

Wan *et al*. (2003) proposed COngestion Detection and Avoidance (CODA) protocol for controlling congestion in WSNs. In this protocol, an open-loop hop-by-hop backpressure mechanism and a closed-loop multi-source regulation mechanism are implemented to control congestion in event-driven WSNs. A backpressure message is broadcasted from congested node toward source nodes to reduce their rates in the event of transient congestion, this achieved in open-loop (control message is sent from congested node) hop-by-hop procedure. For persistent congestion, the second mechanism of closed-loop (control message is sent from sink) multi-source regulation is used to control congestion over many source nodes, when the sink transmits ACK packet towards nodes to regulate their transmission rate. However, in CODA protocol additional energy is consumed because of the use of ACK from the sink and the use of backpressure messages. CODA protocol uses AIMD mechanism to adjust the rate, and it is often leads to packet loss (Cy Wan *et al*., 2003).

Tao & Yu (2010) proposed Enhanced COngestion Detection and Avoidance (ECODA), which incorporates three strategies for detecting congestion, first strategy: dual buffer thresholds and weighted buffer difference are used for congestion detection. The second strategy, the ECODA has a flexible queue scheduler, and based on the priority the packets are scheduled, the packets are filtered according to channel load and the occurred congestion. Finally, in case of persistent congestion, ECODA uses the scheme of bottleneck node-based source transmission rate as the third strategy (Tao & Yu, 2010).

The protocol of Congestion Control and Fairness (CCF) that proposed by Ee & Bajcsy (2004) is detecting congestion by packet service time. In this protocol, the data is routed from source to sink based on tree routing structure. A fair and efficient transmission rate to each node is assigned by CCF protocol. In media access control, a back-off is a widely used mechanism to reduce contention. In case that the channel is busy the node adding a back-off time, the idea of back-off is to restrain a node from accessing the channel for a period of time and when the channel is free the node starts to send its data. CCF proposed two methods to eliminate congestion: 1) to add a small delay in the data-link layer, which will decrease the probability of collision during simultaneous transmissions from nodes. 2) Monitoring the queue occupancy, and when the occupancy exceeds a predefined threshold, the congested node will inform the upstream node about the congestion and then the rate will be reduced (Ee & Bajcsy, 2004).

Yin *et al*. (2009) proposed Fairness-Aware Congestion Control (FACC), in this protocol the intermediate nodes are categorized according to their location from the sink. Nodes that are close to source are named near-source nodes, and nodes which are close to sink are called near-sink nodes. In this protocol a near-sink node can have more traffic compared to a near-source node. Near-source nodes are responsible for adjusting the flow which is the number of packets per second (these intermediate nodes store the incoming data for a certain time to be able to identify packet flows, therefore, the nodes can control the flow by increasing or decreasing it), and allocate an approximately fair rate to each passing flow to share the bandwidth by comparing the incoming rate of each flow and the fair bandwidth. Near sink nodes do not need to maintain per flow state and it just generate a control message to warn near-source node, depending on this message near sink node will share fair rate to each flow, and this will control the congestion (Yin *et al*., 2009).

Vedantham *et al*., (2007) proposed a congestion control method name as CONgestion control from SInk to SEnsors (CONSISE) which adjusts the downstream (from sink to sensors) sending rate at each of the sensor nodes. The congestion control in this type of WSN is clearly motivated by the nature of the application, for example, tracking an object in military application, needs the sink to transmit a query and the data associated with the query, such as photos of the all the enemies being targeted. Then, this would make the message size associated with these queries substantially large. Also, a reliable delivery of these messages to the sensors should be achieved in the least possible

time, the sending of these type of large messages might create congestion. Then, with depending on the congestion level in the local environment, the available network bandwidth will be utilized according to this adjustment. The advantage of this method, is its easy implementation, efficient use of resources with minimal overheads, and the approach is scalable (Vedantham *et al*., 2007).

Sankarasubramaniam *et al*. (2003) developed Event to Sink Reliable Transport (ESRT) which a transport solution to achieve reliable event detection with efficient energy consumption and a functional resolution to congestion problem. When the node buffer is nearly to overflow (like during next period) the ESRT will set a congestion notification (CN) bit in the packet header, then the sink based on the reliability measurement and the congestion notification bits and also the previous reporting rates will compute next new reporting rate. One of the disadvantages of ESRT, there is no congestion control mechanism at the intermediate nodes and all the rate adjustments is the sink responsibility. In addition, the weakness of ESRT protocol is the sink should be powerful sink to be able to broadcast to all the nodes in the network. Another disadvantage of this protocol is that the rate reduction will be applied to all nodes even for uncongested path (Yogesh Sankarasubramaniam *et al*., 2003).

Iyer *et al*. (2005) propose Sensor Transmission Control Protocol (STCP) which is the transport layer with scalability and reliability. In this protocol, the most functionalities of this protocol are implemented at the base station. The congestion detection and avoidance and the controlled variable reliability are supported by STCP. With the use of session initiation packets each node will establish an association with the sink. STCP adopts congestion notification explicitly for congestion detection. Two thresholds points are maintained in the buffer of each sensor node, these are (T_low, T_high). The first congestion notification is set in the packet header with a certain probability when the reaches T_low. On the other hand, when the buffer reaches T_high, the congestion notification bit is set to every forwarded packet by the node. Then when the sink receives these packet, the sink by using the notification bit in the ACKs will inform the source of the congested path. On receiving the congestion notification the source will slow down the transmission rate (Iyer *et al*., 2005).

A strategy called FUSION, which is proposed by Hull *et al*. (2004), is detecting congestion by measuring the queue length. The node responsible for detecting congestion will set the congestion notification bit in the header of each forwarded packet in the case of congestion is occurred, then the neighbouring nodes once overhear the congestion notification will stop sending packets to congested node (Hull *et al*., 2004).

Fang *et al*. (2010) proposed Congestion Avoidance, Detection and Alleviation (CADA) protocol. This protocol in case of congestion the data transmission is ensured to be high and the energy consumption is optimized. With the combination of buffer occupancy and the channel utilization the CADA protocol will detect WSN congestion. CADA alleviate congestion dynamically using traffic control and source control as a result the network throughput, energy consumption, and end-to-end delay will be better (Fang *et al*., 2010).

Enigo, (2009) designed Energy Efficient Congestion Control (EECC) protocol which is a source rate congestion control. Any intermediate node adds to the received packet its current weight (the weight defined as the product of channel busyness and the buffer occupancy) then this packet will be passed to next hop node. The sum of such weights is used and when the higher threshold is reached then the node will set the congestion notification bit in every data packet sent. The sink according to collected data will partition the nodes to clusters and each cluster will have its own sending rate and data similarity (Enigo, 2009).

Karenos *et al*. (2007) proposed COngestion control for MUlti-class Traffic (COMUT) framework that supports multiple classes for WSN, this frame work consists of scalable and distribute cluster-based mechanisms. Three mechanisms are combined COMUT: 1) cluster formation, in this state, the sensor nodes are grouped in clusters and each cluster elect a cluster head, the cluster is structured with the help of zone routing protocol. 2) traffic intensity estimation, in this mechanism of COMUT, the congestion level of each cluster is estimated based on the traffic intensity within and across multiple clusters, and 3) rate regulation, in this mechanism the communication between the cluster head and the source will specify the rate of the source. COMUT has another feature, which is the differentiation between the flows in the congested path, this achieved by

making the flows with low importance to decrease their flow in the existence of flows with higher importance (Karenos *et al*., 2007).

A Collaborative Transport Control Protocol (CTCP) is proposed by Giancoli *et al.* (2008) to guarantee reliable packet transmissions between the base station and the source, and also in CTCP the energy is saved by implementing two reliability profiles. In the event of congestion, a CTCP node will broadcast a stop message when the queue occupancy exceed a predefined threshold, then when the nodes receive the stop messages, they will stop transmitting data packets to the congested node until the reception of a start message (Giancoli *et al*., 2008).

Yaghmaee & Adjeroh (2008) proposed a new Queue based Congestion Control Protocol with Priority Support (QCCP-PS) which uses a queue length as indication of congestion degree. This protocol consists of three parts: 1) Congestion Detection Unit (CDU) that uses the queue length as the congestion indicator and it creates a congestion index which is a number between 0 and 1. 2) Rate Adjustment Unit (RAU) which calculates the traffic rate of each sources as well as its local traffic source, this calculation is based on the current congestion index and the source traffic priority, then the new rate will be sent to next unit, which is 3) Congestion Notification Unit (CNU), then the CNU will notify all the nodes about the child nodes about the new rate (Yaghmaee & Adjeroh, 2008).

The Learning Automata-based Congestion Avoidance Scheme (LACAS) that proposed by Tiwari *et al*. (2009) is an adaptive learning solution for congestion avoidance. In this work, to avoid congestion of intermediate nodes, their data rate is controlled by implementing in each node a code capable of taking intelligent actions (called automata). The action of automata code in each node is the adjustment of the data rate of the node, the adjustment is based on the probability of how many packets are likely to be dropped if the data rate in the node remains the same. Automata allocates accurate data rate based on its learning from past behaviour (Tiwari *et al*., 2009).

Akyildiz *et al*. (2006) proposed XLM which is a cross-layer protocol that fusing communication layers in to a single protocol. The aim of XLM protocol is: minimizing the energy consumption, adapt communication decisions, and avoid congestion. In XLM

protocol, the transmission is initiated from the node by broadcasting an RTS (Request to Send) with its location. Then, each node closer to sink decides its participation based on the RTS Signal to Noise Ratio (SNR), its remaining energy, and its available buffer space. If because of the occurred congestion no CTS are received then the node will multiplicatively decrease its generation rate. Otherwise, for each received ACK the generated rate will increase linearly. As each transmission at every hop must be preceded by a handshake message exchange then the overhead of this approach is high.

In Table 2.1, a comparison of the above-mentioned protocols related to traffic control.

2.2.3.3 Priority-aware congestion control protocols

Some protocols of priority-aware congestion control are presented with some detail:

Priority-based Congestion Control Protocol (PCCP), this protocol consists of three components a) Intelligent Congestion Detection (ICD), b) Implicit Congestion Notification (ICN), and c) Priority-based Rate Adjustment (PRA). ICD uses the ratio of packet inter-arrival time (time of received packet at the node) over packet service time (needed time for processing packets at the node) to detect the congestion, this ratio also named congestion degree. In ICN, congestion notification is given through setting a bit in the header of data packet. Since each sensor node might have different priority depending on their function or location, a node priority index (assigned depending on the node traffic) is introduced in this protocol to reflect the importance of each sensor node. The bandwidth is determined on a novel priority-based algorithm employed in each sensor; the node with higher priority index gets more bandwidth, the nodes with same index get same bandwidth, and more bandwidth is given to node with sufficient traffic compared to node generates less traffic (Sohraby & Lawrence, 2006).

Table 2. 4: comparison of traffic control protocols.

Protocol	Congestion detection	Congestion notification	Application type	Evaluation type
ARC (Woo & Culler, 2001)	Packet loss	-	Event and periodic	Implementation
CODA (Cy Wan *et al*., 2003)	Queue length + channel load	Back-pressure message	Event	Simulation, experimentation
ECODA (Tao & Yu, 2010)	Buffer occupancy +Queue length + transmission rate	Information in header	periodic	Simulation NS2
CCF (Ee & Bajcsy, 2004)	Packet service time +queue length	Information in header	Event	Simulation, implementation
FACC (Yin *et al*., 2009)	Channel busyness + queue length	Feedback message	Continuous	Simulation NS2
CONSISE (Vedantham *et al*., 2007)	Periodic rate control	Information in header	Query	Simulation NS2
ESRT (Yogesh Sankarasubramaniam *et al*., 2003)	Queue length	Bit in header	Event	Simulation
STCP (Iyer *et al*., 2005)	Buffer occupancy	Bit in header	Event	Implementation
FUSION (Hull *et al*., 2004)	Queue length	Bit in the header	Hybrid	Experimentation
CADA (Fang *et al*., 2010)	Buffer occupancy + channel utilization	-	Event	Simulation
EECC (Enigo, 2009)	Buffer occupancy + channel busyness	Bit in header	Event	Simulation
COMUT (Karenos *et al*., 2007)	Queue size	Backpressure message	Event	Simulation
CTCP (Giancoli *et al*., 2008)	Buffer overflow	Control message	Event	Implementation
QCCP-PS (Yaghmaee & Adjeroh, 2008)	Queue length	Information in header	Multimedia	Simulation
LACAS (Tiwari *et al*., 2009)	Drop packets + queue delay	Messages from sink to sources	Continuous	Analytical, simulation
XLM (Ian Akyildiz *et al*., 2006)	CTS packet loss	-	Event	Analytical, simulation

Wan *et al*. (2009) proposed Cross-Layer Active Predictive Congestion (CL-APCC) to improve the network performance. The data flows of a single-node according to its memory status is examined by employing the queuing theory, also the average occupied size of local networks is analysed in this scheme. The current data change trends of local networks is investigated to forecast and adjust the sending rate of the node in next period (J. Wan et al., 2009).

2.2.3.4 Queue-assisted technique protocols

Some of the queue-assisted protocols that control WSN congestion are listed below in some detail:

Healthcare Aware Optimized Congestion Avoidance (HOCA), this protocol is designed for healthcare application and needs minimum delay and high energy efficiency (to prolong battery lifetime). A cross layer function between transport and network layers is used to control the congestion. At the network layer, the data are classified as: sensitive data, non-sensitive data, and control data. The sensitive data need low delay and should arrive the sink as soon as possible when an event occurs. HOCA protocol controls the congestion using Active Queue Management (it maintain the length of a queue and prevents it become full), and if congestion occurs then a control message informs the sources to adjust the data rate (Rezaee *et al*., 2014).

Rangwala *et al*. (2006) propose a low-overhead congestion sharing mechanism called Interference-aware Fair Rate Control (IFRC). With use of a distributed rate adaptation technique, each node converges to a fair and efficient rate for the flows. To achieve fair and efficient rate, the node shares the congestion information with the potential interferers (a node is potential interferer if a flow originating from first node uses a link that interferes with the link between second node and its parent). Three inter-related components are combined the IFRC: 1) measuring the node level congestion, 2) sharing congestion information with potential interferers, 3) using AIMD control to adapt the transmission rate. When the queue size of a node exceeds an upper threshold then the congestion is occurred and the node is considered to be congested and the rate is reduced according to AIMD rate adaptation scheme. Each outgoing packet is attached in its packet header the average queue size and the current transmission rate (Rangwala *et al*., 2006).

2.3 Addressed research gap

Hierarchical Tree Alternative Path (HTAP), this protocol uses dynamic alternative paths to the sink in order to alleviate congestion in a simple, efficient, and scalable resource control. HTAP has four steps: 1) Topology control in which each node creates a table of neighbouring nodes, 2) Hierarchical tree creation, here each node that is becoming a source node (by sensing an event) is self-assigned as level 0, and sends level discovery through its neighbours, nodes that receives this message are considered as children to the source node and set as level 1, this operation is repeated until all nodes are assigned a level and stops when the level discovery reach the sink, 3) Alternative path creation, in which the transmitter node after notifying about the congestion from the congested node, it searches in its neighbour table and choose the next available path in the same level to send through it, 4) Powerless nodes handling, if a node's power is depleted then it sends a message to inform other nodes about its power, then the neighbouring nodes update their paths (Sergiou *et al*., 2013).

The simple design of HTAP is the major advantage of HTAP, since it adds minimal overhead to the already heavy loaded networks that is intended to operate onto.

From previous section, we notice that the majority of these algorithms employs traffic control for congestion mitigation in WSNs, and the traffic control can produce in sufficient performance in many cases such as the critical applications, this is because reducing the traffic rate might delay the required data that need to arrive quickly in critical application. Therefore, the design of HTAP is depending on the resource control, also the resource control algorithms can assist in the uniform energy utilization of network which leads to the extend network's lifetime. On the other hand, since the existing resource control protocols do not provide strict performance bounds, the HTAP protocol assurances comprise a critical factor for the design of congestion control and avoidance algorithms.

HTAP algorithm is designed to avoid congestion in the deployed nodes by employing alternative paths, using source-based trees in the network. Buffer occupancy is the first indication of congestion occurrence and then employs the ratio of out/in data rate to find alternative path.

The first part of this research is to test the performance of the HTAP algorithm experimentally (only simulations are reported in the literature). A number of 40 Waspmote nodes is initially deployed randomly, but uniformly in the space of

engineering lab. In this network there is one sink is located at known position, and there are many source nodes deployed in the network, since an event may occur at any point in the network. Also, in this network each node knows it position and the position of the sink. In our network deployment all nodes except sink are identical and CSMA/CA is employed as the MAC protocol.

In second part of this research, the HTAP algorithm suggests that each node should be connected to at least two higher-level nodes, to make two paths towards network sink. However, in some cases the intermediate node might be connected only to one node at the higher-level, then in this case only one path is available. Therefore, to alleviate congestion in this case, the improved HTAP algorithm will use the traffic control (rate control) to alleviate the occurred congestion. As a result, the improved HTAP algorithm in this research, is combining both resource control and traffic control to alleviate WSN congestion.

Chapter 3: Research Methodology

3.1 Introduction

As stated in previous chapters, when an event occurs in an event-driven application, a potentially large number of generated packets need to be delivered to sink reliably and on-time. High network data traffic may lead to congestion and hence lost data packets, decreased reliability, and increased power consumption. Congestion can be controlled using resource aware methods, e.g. the use of alternative routes to send the generated packets to the sink. As mentioned in section 1.3, HTAP has been simulated, but in this research, HTAP is implemented in real hardware, then from the found results it is able to verify that HTAP is able to control the occurred congestion practically. To investigate resource aware methods (HTAP algorithm), various experiments were carried out between November 2015 and March 2017. The experiments took place in the Engineering departmental labs of University of Leicester. The Waspmote sensor node was used to perform the experiments of this project. The important advantage of using the Waspmote node as the sensor network node in all experiments is that it is relatively easy to program, has good manufacturer support, and using XBee transceiver can provide different communication ranges. In this project, the congestion is detected by using the parameter of buffer occupancy for the Waspmote node. The actual Waspmote buffer size is 8 Kbytes.

3.2 Network requirements

The common wireless sensor network topology that is used by most sensor nodes deployments is illustrated in Figure 3.1 [Yang and Mohammed 2000]. The suggested experiments will be done for an indoor application such as fire detection, the indoor applied experiments do not need the precise alignment to an external clock.

A survey of available wireless sensor nodes will be introduced in the following section with more details about Waspmote nodes.

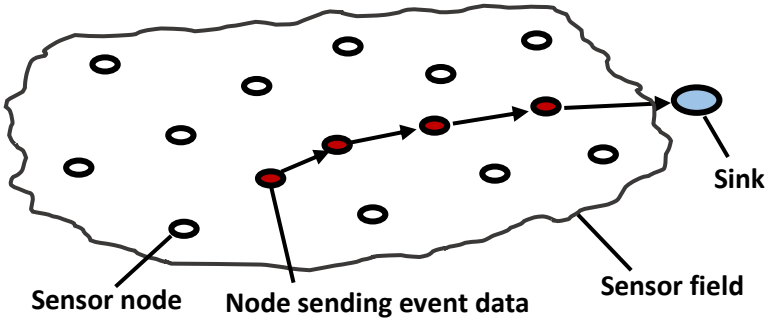


Figure 3- 1: Basic wireless sensor network, mesh topology.

3.2.1 A comparison of currently available sensor nodes (motes)

There are many types of nodes can be chosen to build a wireless sensor network, however, the experiments that study the control congestion in this project require sensor node with the following features:

- Can be deployed in different places in any busy indoor environment, which mean the node can be adapted to have different communication ranges.
- Easy and supported programming language from the manufacturer.
- Long-life battery.
- Commercially available and has available spare parts.
- Common interface availability (USB port).
- Flexible control to the node's memory such as SRAM, Flash memory, and EEPROM. Also, has an expandable memory.

Some of current nodes will be discussed briefly, and the candidate node for my intended research will be discussed in more detail:

a. **TelosB:** is an open source platform, works on the standard IEEE 802.15.4 for low power consumption (long battery life) as well as fast wakeup from sleep. TelosB has the operating system TinyOS, which is open source, small, and energy-efficient software. This operating system is a full package for lab studies that includes USB programming and it is using NesC language which is an extending C language, and the programming is under Linux environment. TelosB can use wide range of sensors that can be added through expansion board, (Memsic, 2013)

b. **SHIMMER:** (Sensing Health with Intelligence, Modularity, Mobility, and Experimental Reusability) is the smallest and most robust wearable wireless sensor, and is originally designed for education, remote sensing and clinical researches to integrate wireless body sensor into a wide range of application areas.

c. **MicaZ:** It is compatible with the IEEE 802.15.4 standard and uses TinyOS and NesC for programming. Like the TelosB it has built in transceiver chip, and has some built in sensors.

d. **IRIS:** - latest wireless sensor network module from Crossbow Technologies. It includes several improvements (e.g. increased transmission range) over the Mica2/MicaZ family of products. This node is for low power wireless sensor networks. IRIS provides users with wide variety of custom sensing applications (Johnson *et al*., 2009)

e. **Waspmote:** is an open source wireless sensor platform, specially focused on the implementation of low-consumption modes which allows the sensor nodes ("motes") to be completely autonomous, i.e. battery powered. Hence, depending on the duty cycle and the radio used this design will offer a lifetime for the node of between 1 and 5 years (Libelium 2014). Waspmote is an Arduino based node, uses XBee transceiver from Digi Company, and this is a flexible transceiver choice, which can provide many standards including IEEE802.15.4 and ZigBee. Also, the XBee transceiver can be adapted to have different communication ranges by connecting attenuator between the antenna and the XBee module. The main components of the Waspmote board are illustrated in Figure 3.2.

There are two radio sockets (Radio Socket 0 and Radio Socket 1) on the Waspmote board. Socket 0 is used for connecting the XBee transceiver, while Socket 1 can be used to connect another communication module such Wi-Fi module.

The USB port is used to upload the microcontroller's program from the computer to Waspmote board (Libelium, 2014).

Table 3.1 shows the technical capabilities of the nodes discussed above nodes.

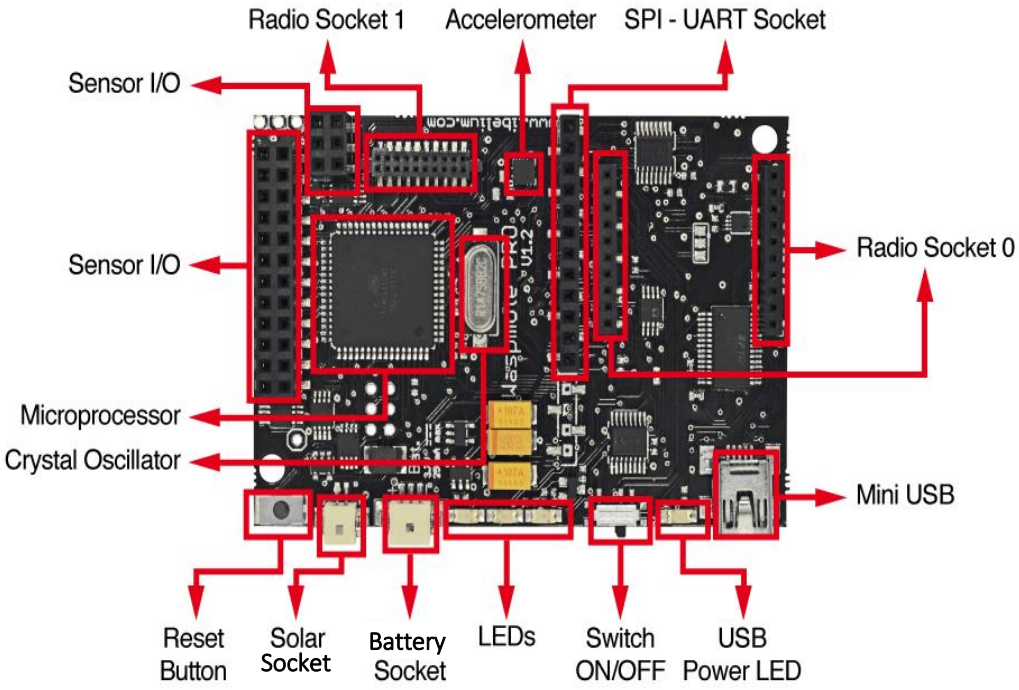


Figure 3- 2: Main components of Waspmote board (Libelium, 2014).

Table 3. 1: Selected technical specifications of Sensor nodes.

Name of Node	Micro-controller	RAM (kB)	External memory (KB)	Transceiver	Frequency range (GHz)	Data rate (kbps)	Transmitter power dBm	Max. outdoor range (m)	Operating system
TelosB	TI MSP430	10	48	TI CC2420	2.4-2.483	250	-24 to 0	75-100	TINY OS
SHIMMER	TI MSP430F1611	10	48	SHIMMER SR7 (TI CC2420)	2.4-2.483	250	-24 to 0	~100	TINY OS
MicaZ	ATMEGA128	4	128	TI Chipcon CC2420	2.4	250	-24 to 0	75-100	TINY OS
IRIS	ATMEGA128	8	128	AT86RF230	2.4	250	3	>300	TINY OS
Waspmote	ATmega1281	8	128	Xbee-802.15.4-PRO	2.4-2.465	250	10-18	7000 LOS	-*

*to write C code, and doesn't have operating system.

Table 3. 2: Waspmote specifications.

Microcontroller	ATmega1281
Frequency	14.7456 MHz
SRAM (buffer)	8 kbytes
EEPROM	4 kbytes
FLASH memory	128 kbytes
SD Card	2 Gbytes
Norrmal operation: ON	Consume: 17 mA
Sleep mode	Consume: 30µA
Deep sleep mode	Consume: 33µA
Hibernate	Consume: 7µA

Another good feature for Waspmote compared to other available nodes, is that Waspmote is Arduino based, and since Arduino has a wide range of previous applications (with written codes). These Arduino applications can be used for comparison from Waspmote new experiments.

Waspmote uses C++ language for programming, the programs with this language are fast written, need small memory footprint, and they small file-size (Spring, 2017). Waspmote can use other radio modules such as (Wi-Fi module, GSM/GPRS module, 3G+GPS module, RFID module, and Bluetooth module).

By using expansion radio board, it is possible to connect two radios at the same time. This means a lot of different combinations are now possible using any of the 10 radios available for Waspmote: 802.15.4, ZigBee, Bluetooth, RFID, RFID/NFC, Wi-Fi, GSM/GPRS, 3G/ GPRS, 868 and 900. A possible combination in this project is using XBee module and Wi-Fi module at same time for clock synchronization.

Another useful feature of the Waspmote is the ability to add an SD memory card up to 2GB (because it uses FAT 16 allocation table) which can be used to store large amount data such as images. The application development is carried out using through software called Waspmote-IDE, it is an open source easy to install and easy to use software.

In general, Waspmote specifications is listed in Table 3.2 (Libelium, 2014). An important part of this research is to study the communication protocols; therefore, the following sections will concentrate on some details of the transceiver (XBee) chip in Waspmote.

3.2.2 Waspmote-IDE

The Waspmote-IDE is Waspmote's software development kit (Figure 3.3). Writing Waspmote code and uploading it to Waspmote node is done through Waspmote-IDE. The Waspmote codes are written in the text editor and called sketches. Waspmote code is combined from three parts, variable declaration part, basic *setup ( )* function part, and *Loop ( )* function part. In first part, it is necessary to include required libraries and declare global variables at the beginning of the code.

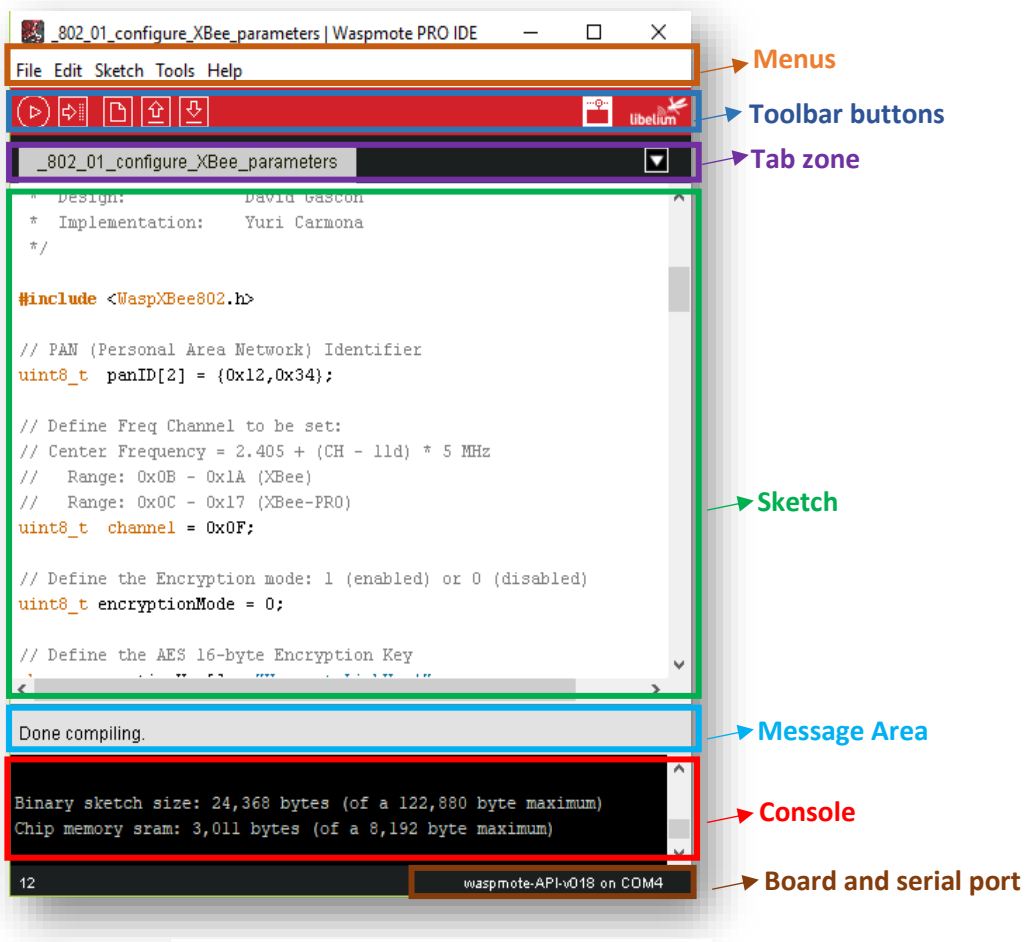


Figure 3- 3: Waspmote IDE sections.

Waspmote code skeleton:

```
  // 1. # include Libraries
  // 2. Variables declaration and required definitions
void setup ( ) {
  // 3. Modules initialization
  // 4. Required setting operation
  }
void loop ( )  {
  // 5. measure
  // 6. send and receive data
  // 7. Sleep mode
  }
```

Setup ( ) function part runs once only each time Waspmote starts running (including reset operation or reboot). It's advised to initialize Waspmote attached modules such as XBee 802.15.4 or Zigbee or GPRS, etc. in this part. Also *setup ( )* function includes some important parts of the codes that needed to be run at Waspmote starting such as Waspmote time setting. The last part of *Loop ( )* function runs continuously while power remains. In this function, the data can be sent, received, measured, and the node can also be put into sleep mode to decrease the node energy consumption.

After pressing upload button from the toolbar buttons, the console will display an error message if the code has any mistakes. The console shown in Figure 3.9, displays:

Binary sketch size: 30,526 bytes (of a 122,880 bytes maximum)

30,526 bytes represents the code size. The sketch is saved in Waspmote flash memory. From Tools menu, the Waspmote board can be chosen and then the correct serial port.

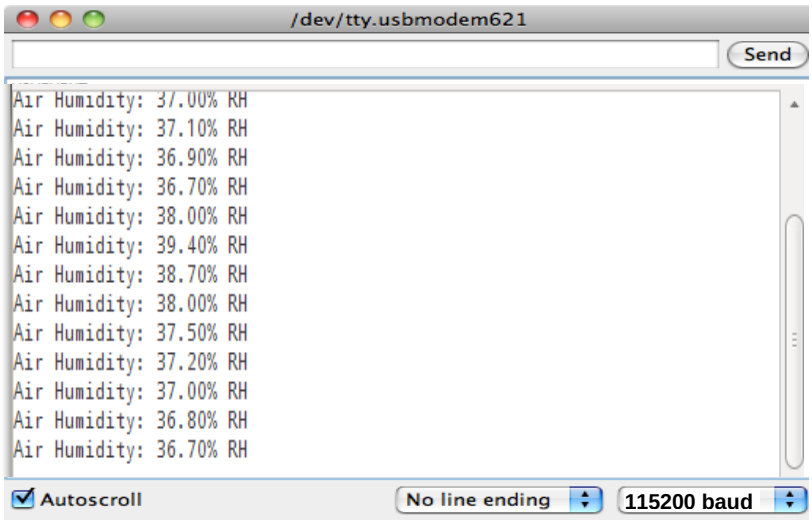


Figure 3- 4: Serial monitor to display Waspmote data (Libelium, 2014).

3.2.3 Serial Monitor

To display the sent or received data by Waspmote, serial monitor (Figure 3.4) should be set to use baud rate of 115200bps for Waspmote v12. The text box in serial monitor is used to send data in broadcasting mode to network nodes.

Figure 3- 5: XBee 802.15.4 PRO with RPSMA connector and whip antenna.

3.2.4 XBee-802.15.4 PRO

The candidate transceiver to do this research is XBee-802.15.4 PRO (Figure 3.5). The XBee 802.15.4 PRO module conforms to the IEEE802.15.4 standard which defines the physical and Mac layers. Compliance with the IEEE802.15.4 standard provides reliable delivery of data between devices. The unique needs of low cost, low power wireless sensor networks are also supported by XBee-PRO. This module is designed to be mounted into Socket 0 (see Figure 3.2). XBee 802.15.4 module communicates with the microcontroller using the UART0 at 115200bps.

3.2.4.1 XBee-PRO specifications

Table 3.3 shows XBee 802.15.4-PRO specifications.

Table 3. 3: XBee-802.15.4 PRO specifications.

Specification	**XBee-502.15.4 PRO**
Protocol	802.15.4
Operating Frequency	2.4 GHz
txPower	63.1mW
Sensitivity	-100dBm
Indoor range	60m
Outdoor range*	1600m
RF Data rate	250kbps
Antenna options	Integrated whip, chip or UFL connector, RPSMA connector
Number of channels	12

* Line of sight and outdoor

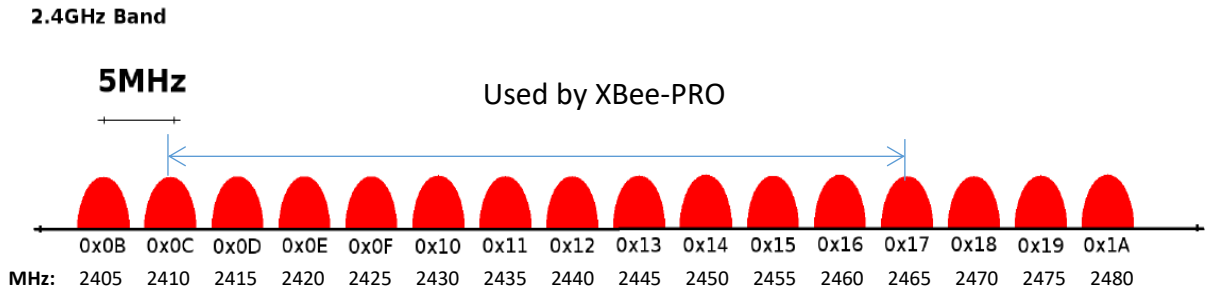


Figure 3- 6: Frequency channels in 2.4 GHz band, starting from centre of 2.405GHz up to 2.480GHz, and each channel is 3MHz wide, centred around the indicated frequency (for XBee-PRO 12 channels started from 0X0C: centre frequency 2.410GHz to 0X17:2.465GHz) (Libelium, 2014).

XBee-PRO module operates within the free band of 2.4GHz frequency which known as Industrial Scientific Medical (ISM) band. XBee-PRO uses 12 channels each with 5 MHz bandwidth from the 16 channels of 802.15.4 protocol (Figure 3.6).

The transmission power can be adjusted to 5 different levels, from 10dBm to 18dBm in 2dBm steps (labelled as levels 0 to 4). The power level can be set either through XCTU (section 3.2.5.1) software or API instruction. Another technique to set the power level of XBee-PRO is by using over the air setting. Which is done by sending a setting message through a gateway (connected to PC) to the XBee-PRO. The letter P in the control message shown in Figure 3.7, refers that this message is XBee-PRO power setting message. The setting message may contain other XBee setting such as CSMA threshold.

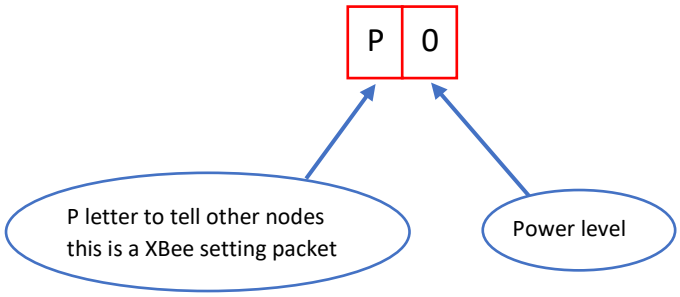


Figure 3- 7: XBee-PRO power setting message construction.

3.2.4.2 XBee-PRO networking

Some parameters of XBee-PRO should be set to prepare Waspmote node for working in networks. XBee parameters setting can be done either by using API instruction or by using XCTU software (section 3.2.5.1). These parameters are:

- PAN ID: is a 16-bit number that identifies the network. The PAN ID must be unique to differentiate a network.
- Node Identifier: is a string of up to 20 characters, which identifies Waspmote node in the network.
- Network address: is a 16-bit network address, used to send data to the node in 16-bit unicast transmissions.
- Channel: defines the frequency used by XBee module to receive and transmit. XBee 802.15.4 PRO work in 12 different channels (Figure 3.6).

Another important XBee parameter is the Mac address. This address is a 64-bit unique identifier used inside the network for unicast transmission. Mac address has two groups of 32bits High and Low. This address cannot be modified and is given by the manufacturer.

To establish a Waspmote network, every network attached to Waspmote must have the same PAN ID, Channel, and Encryption configuration.

The sent data packet contains in its header a source address and destination address field. XBee module complies with the 802.15.4 specification and supports both 16-bit and 64-bit addresses. The transmission can be done either in Unicast or Broadcast mode.

- **Unicast transmission mode:** in this mode after sending a data packet from the source, an acknowledgment will be sent from the receiver to the sender. If the sender doesn't receive the acknowledgment after 200ms, the data packet will be sent up to 3 times. Unicast mode supports both 16-bit and 64-bit addresses.
- **Broadcast transmission mode:** it is used for sending packets to all modules in the same network. The receive address used for broadcast mode is 0x000000000000FFFF. In this mode, the sent packet will be received by all modules within the sender module range. Broadcast mode doesn't use Acknowledgement. However, to ensure the reception of the sent packet the packet will automatically be transmitted 4 times.

The multiple packets sent during broadcast transmission might affect the network performance by increasing the number of collisions. Therefore, to avoid packet collision while using broadcast mode, each router will adopt a random delay before relaying the

broadcast message. On the other hand, employing unicast transmissions in a dense network may increase packet loss due to the large number of acknowledgements.

3.2.4.3 XBee-PRO communication range

Antennas that can be used with XBee-PRO include a whip antenna or a PCB antenna. The actual performance of XBee-PRO with whip antenna depends on many factors in the indoor environment. These factors include: antenna orientation, antenna height, walls, interference, mounting the antenna near metal object which cause an inefficient antenna operation; etc. For the last factor, it is better to separate the antenna from the metal object by several centimeters. Table 3.4 and Figure 3.8 show the specification and radiation pattern of the antenna respectively. The indoor range test for the XBee-PRO with whip antenna was carried out in the Engineering Lab. The maximum range achieved was 40 meters, this value is measured between 2 Waspmote nodes (sender and receiver), and by setting the power level of the XBee transceivers to highest level (level 4 which is equivalent to 18dB) also, the achieved PDR at the receiver side is 100%.

Table 3. 4: XBee-PRO whip antenna specifications.

Specification	Description
Type	SMAM-RP whip antenna with right angle
Operating frequency	2.4GHz
Gain	2dBi
Polarization	Vertical
Radiation pattern	Omnidirectional
Impedance	50 ohms
Electrical length	¼ λ
Dimension	5 cm

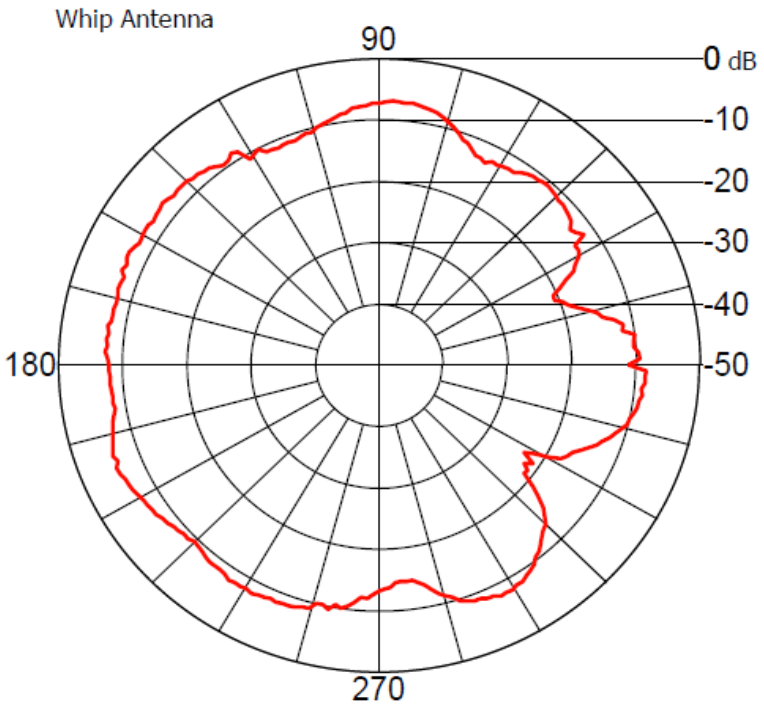


Figure 3- 8: Radiation pattern of whip antenna connected to an XBee-PRO, (Digi., 2012).

In the network space, the important issue for any deployed node is the communication with its neighbour nodes (in one hop), therefore, the nodes' positions in the physical space can be changed to give the same network layout. Since XBee-PRO has a large communication range, so the XBee-PRO needs to be adapted to be used in small designated area to combine WSN with multiple hops. Therefore, in this project to use Waspmote nodes with XBee-PRO in Engineering lab, the XBee-PRO indoor range should be decreased by using Sub-Miniature Version A (SMA) attenuators Figure 3.9. Three different SMA attenuators (10dB, 20dB, and 30dB) were used with two different antennas (2dB and 5dB) to adapt the experiments.

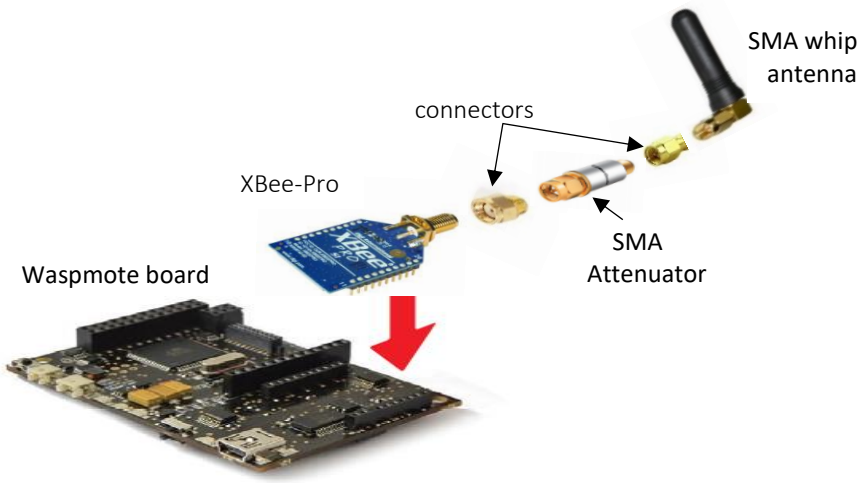


Figure 3- 9: Adding SMA attenuator between XBee module and the antenna.

The Received Signal Strength Indicator (RSSI) reports the signal strength of last received RF data packet. The value is for the last hop and not for earlier hops in a multi-hop link.

Figures 3.10, shows signal strength as a function of distance for 4 different antenna configurations: without attenuator, 10dB attenuator, 20dB attenuator, and 30 dB attenuator respectively. The power level for XBee module was set to highest value which is 4 (18dBm). The measurements were carried out in the Engineering lab. In these measurements, one node (transmitter) was fixed in position and the receiver node was moved to different locations. As the receiver node drifting away from the transmitter node the RSSI value is decreased, then the point of the minimum measured RSSI value gives the maximum coverage area of the transmitter node. The results of these measurements were used for experiments nodes deployments in Engineering lab. For instance, to achieve connectivity between any node and its neighbour nodes, the distance between the nodes can be taken from the results of Figure 3.8. However, these results may not applicable to be used totally in different indoor area, since the propagation will be different in different places.

In Figure 3.9, the connection of a 10dB attenuator to both transmitter and receiver decreases the RSSI at the receiver by about 20dB for both antennas. Thus, the communication range of the sender is minimised to the value of module sensitivity which is (-100dB). It can be seen from Figure 3.10 the distance of the range decreased from 40m to around 12m with 2dBi whereas the range for 5dBi antenna decreased from 45m to 13m.

Using attenuator of 20dB will decrease the value of RSSI at the receiver to around (-80dB) as illustrated in Figure 3.10. The maximum communication range can be covered for 5dBi antenna is 11m with (-101dB), and for 2dBi antenna is 9m.

For last attenuator of 30dB, the maximum communication range is 6m for 5dBi antenna whereas for 2dBi antenna the communication range is 5m. These small values of communication ranges may be used for busy indoor environment to combine WSN with multiple hops without interference. The measured RSSI with the use of 30 dB in both transmitter and receiver at 0m is -80, while it should be around -100dBm, this discrepancy is coming from the fact that near field range is affecting the measured value of RSSI to be -80dBm.

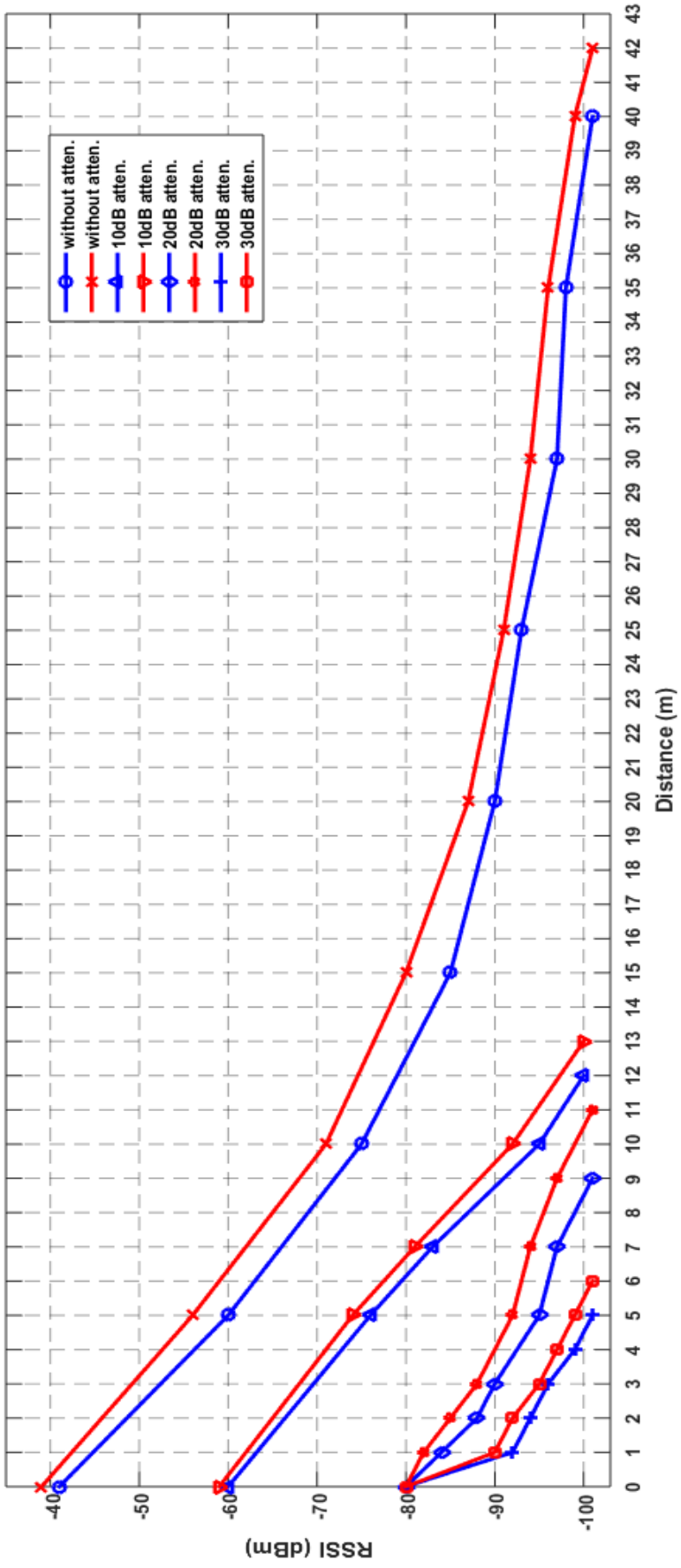


Figure 3- 10: Measured RSSI values versus distance for 4 possible combination of the attenuator (without atten., 10dB, 20dB, and 30dB) with XBee module. (red line is for 5dBi Antenna, blue line is for 2dBi Antenna).

3.2.5 Programming and XBee setting software

Waspmote nodes can be programmed using Waspmote-Integrated Development Environment (Waspmote-IDE) software (Libelium,2014). This is fully described in (Libelium, 2014), but there follows an overview of the process required to develop code for the device. The received data can be displayed and stored in a text file by using number of available software, such as: serial monitor, Hyperterminal, and XCTU software.

3.2.5.1 XCTU tool

It is a conventional tool from Digi International for programming the XBee module (Figure 3.11). With XCTU software, it is easy to update XBee parameters, upgrade XBee firmware, read XBee specifications such as Mac address, and perform communication testing.

To optimize the XBee module, XBee parameters such as channel, node ID, CSMA threshold, baud rate, XBee power level, etc. can be updated after connecting the XBee module to USB serial port.

Baud rate is an important parameter that needs modification. The transfer rate is between 9600 to 115200 bps. Since the RF data rate of XBee module is 250kbps it is safe to set XBee to the maximum baud rate of 115200. XBee- 802.15.4 PRO transmission power level values are listed in Table 3.4, can be set in XCTU by choosing the power level from 0 to 4 as explained in section 3.2.4.1.

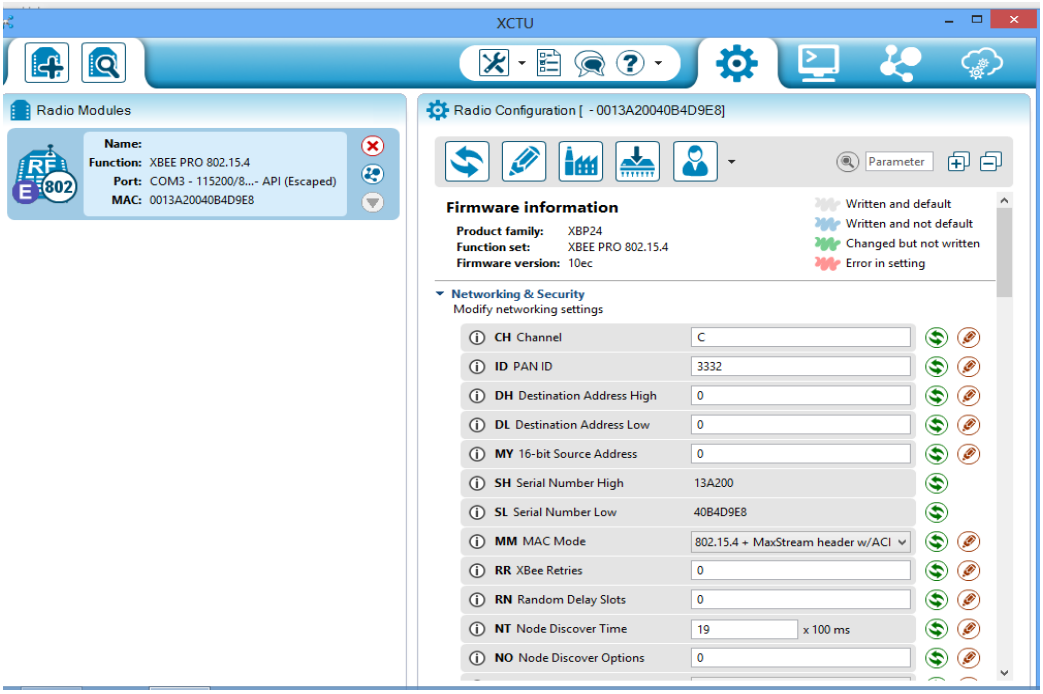


Figure 3- 11: XCTU software window, presenting XBee networking setting.

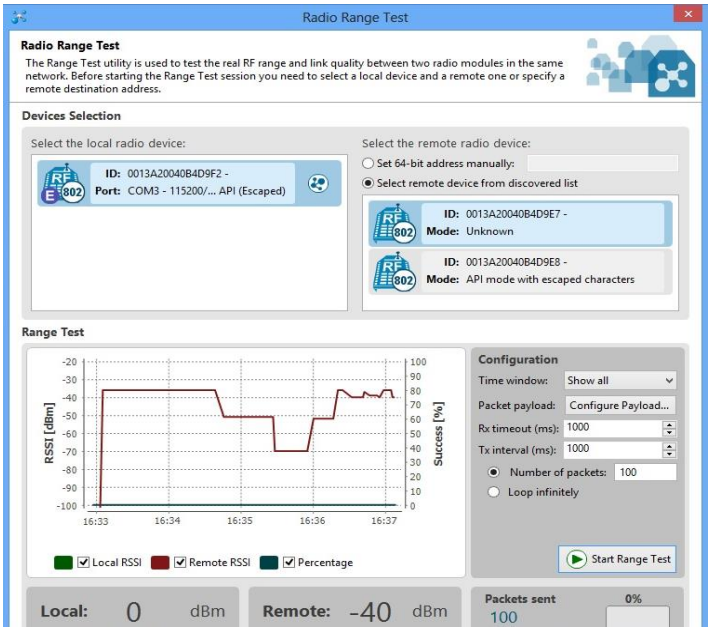


Figure 3- 12: XCTU window, displaying the RSSI node on a mobile node, it also shows the ID of each node and its MAC address.

XBee power level has a great influence in decreasing signal interference inside wireless sensor networks, as will be discussed in next sections. XBee firmware also can be upgraded from 802.15.4 to Digi mesh and vice versa.

XCTU displays the received data, and can show frame header and payload in Hexadecimal format. These received packets can be saved in a text file. Another good feature of XCTU is presenting overall network topology which gives useful information about the active path and the number of network nodes.

Link quality and communication range can be measured between two XBee modules in the same network by using range test utility in XCTU tool. Range test needs to select a local device and a remote device before starting the session.

The range test of a mobile node is illustrated in Figure 3.12, it shows the value of RSSI of a mobile node (indoor mobility) on XCTU window. Also, displaying RSSI vs. Time, gives an indication of the power of the signal from specified node.

3.2.6. Network setup

In order to test the existing methods of congestion control (Sergiou *et al*., 2013) and to develop new methods, an experimental facility consisting of up to 40 nodes (Waspmotes) was deployed in an indoor area of 2250m^2 at the Engineering labs of Leicester University. All nodes, except the sink, are identical, and CSMA/CA is employed as the MAC protocol.

The combined sensor network has one sink located at a known position, and several sources deployed in different positions through the network, because the event might occur at any point in the network. Since this project is studying the congestion of networks, and it's not possible for the node to use GPS indoors each node will be given its position and the position of the sink. The network space will be considered as on x-y plane, then from this plane each node has its x-y coordinates.

3.2.6.1 Distance calculation

For the collected data to be sent to the sink, sensor nodes need to send their data in a multi-hop style in the direction of network sink. However, for any node to know the direction of the sink two phases should be completed: node scan phase and distance calculation phase.

After the scan phase, which will be explained in more detail in next chapter, each node will know its neighbour nodes. The node through this phase will collect the node identification, Mac address, and the RSSI of each neighbour node. Then the node will collect the neighbour nodes positions by sending control message to each neighbour node to send its x-y points.

During the second phase, each node calculates the distance between its position and the sink position as in Figure 3.13, this distance known as the reference distance. Additionally, the distance between each neighbour node and the sink also will be calculated. Based on these distances the neighbour nodes will be classified into two groups: sink direction nodes and opposite sink direction nodes. Sink direction group consists of nodes whose distance from the sink is less than the reference distance, and vice versa for the second group.

To illustrate the distance calculation phase in more detail, consider the following simple network in Figure 3.13. The network is represented in x-y plane, and each node with its x-y points. For instance, Node 4 knows its position and sink position. Let us consider that after scan phase, Node 4 finds the following neighbour nodes: 1,2,3,5, and 6. However, from the distance calculations the neighbour nodes of sink direction group are nodes 5 and 6.

The distance is calculated by using Euclidean distance, the following analysis and Table 3.5 show the two groups of nodes.

- Distance reference (D-ref): between Node 4(7,6) and the Sink (19,7): 12.04

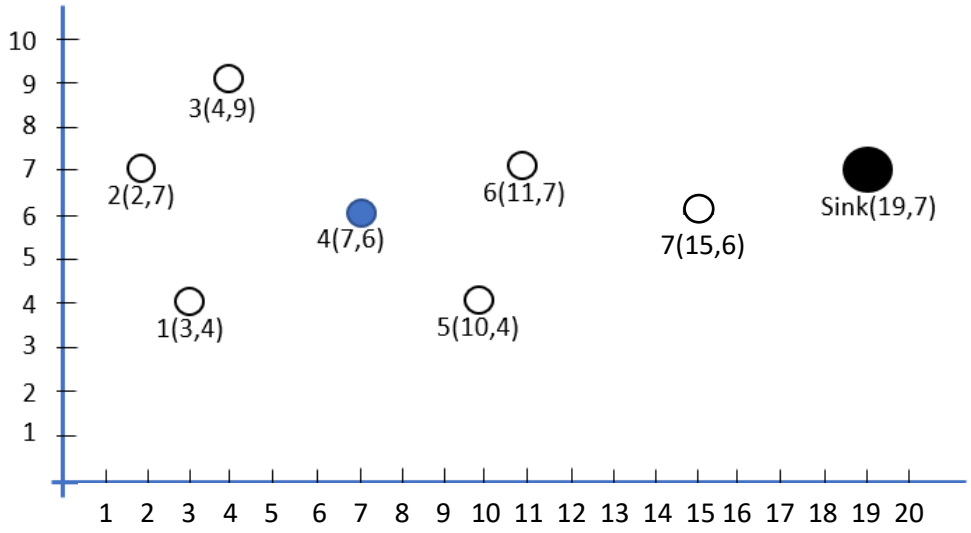


Figure 3- 13: Simple WSN represented in x-y plane.

Table 3. 5: Results of distance calculation and neighbour nodes groups for Node 4.

Node x	Distance between node x-sink (m)	D-ref	Opposite sink direction group (> D-ref)	sink direction group (< D-ref)
Node 1	16.27	12.04	Node 1	-
Node 2	17.00	12.04	Node 2	-
Node 3	15.13	12.04	Node 3	-
Node 5	9.48	12.04	-	Node 5
Node 6	8.00	12.04	-	Node 6

3.2.6.2 Coverage and connectivity of nodes

An event-driven application that monitors critical conditions of a physical environment needs full coverage with connectivity for any location in the monitored field. A location is considered to be covered if it is within the sensing range of any sensor node. To ensure good connectivity between network nodes, each node should be in the communication range of its neighbour nodes (Xin Liu, 2006) and at least some nodes should be within the communication range of a sink.

Physical obstructions are likely to negatively affect the nodes connectivity. Wireless sensor networks using the IEEE 802.15.4 protocol have low power signals. Therefore, to keep good signal strength between the transmitter and the receiver the direct path between them should be as clear as possible.

Waspmote locations in an Engineering laboratory illustrated in Figure 3.14. The Waspmote nodes located on tables in the lab.

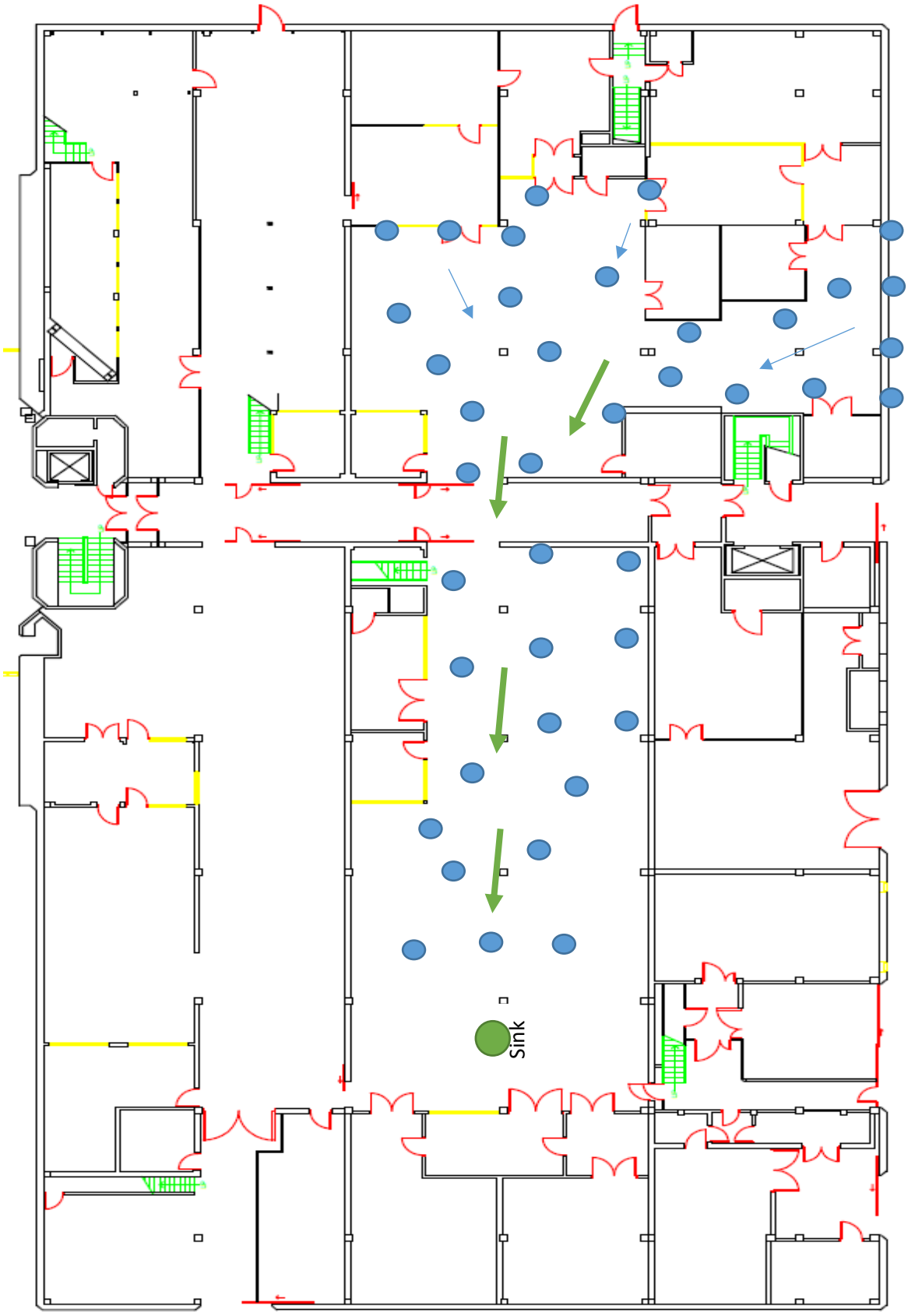


Figure 3- 14: Node deployment (blue dots) in Engineering department lab. Arrows represent the intended data flow toward sink.

3.2.7. Network interference

The 2.400 GHz to 2.485 GHz ISM band is a shared and unlicensed frequency band that can be utilised by multiple wireless networks at the same time. These networks such as 802.11, Bluetooth, WSNs, and cordless telephones, may interfere with each other and lead to intermittent network connectivity, low throughput, high energy consumption, etc (E. & Terzis, 2008). However, interference can happen between nodes in the same network, for instance, if two transmitters can hear the receiver, but not hear each other, this causes interference when they send to the receiver at same time.

In this experimental work, there is an external interference coming from 802.11 Wi-Fi networks (Ahmed *et al.*, 2010). As in Figure 3.15, the 802.11 channels are each 22 MHz wide and overlap with 4 802.15.4 channels. Three non-overlapping channels of 802.11 are normally used by the Wi-Fi routers, which are (1,6, and 11) and they can be observed to avoid interference with 802.15.4 channels. This is achieved by choosing an 802.15.4 channel away from the used Wi-Fi channel. For instance, if the existed Wi-Fi channel in the place is channel 6, then the selected 802.15.4 channel will be far from the frequency of the Wi-Fi channel, such as channel 0C of 802.15.4 channels. Therefore, prior to each experiment the dominant Wi-Fi channel in the area was found by using Vistumbler software, then a broadcast message was sent from the gateway to set the Waspmote nodes to select an appropriate channel to avoid interference with Wi-Fi network.

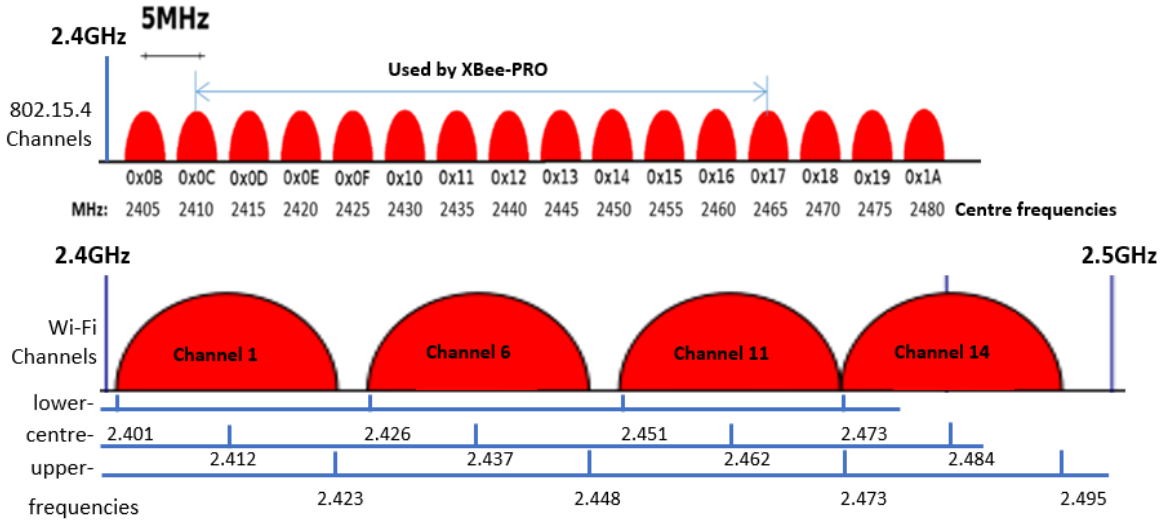


Figure 3- 15: Frequency ranges of IEEE802.15.4 and Wi-Fi channels.

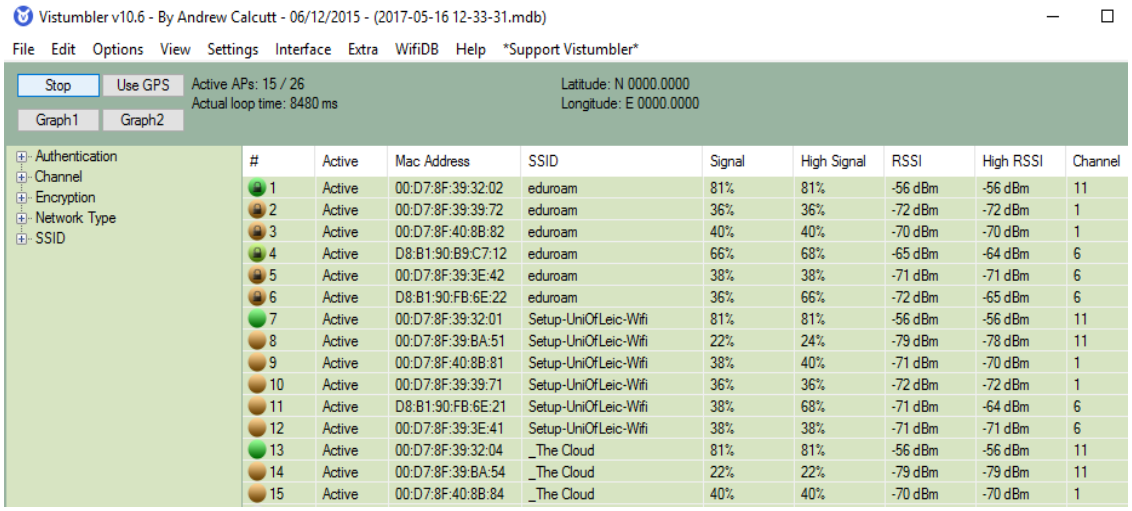


Figure 3- 16: Vistumbler software window, presenting a sample of running Wi-Fi channels in Engineering Lab.

Vistumbler software (Figure 3.16), is an open source tool that scans for nearby Wi-Fi networks. After scan session, Vistumbler will display the networks RSSI, Mac address, network channel, Service Set Identifier (SSID), encryption being used, and more. Vistumbler scanning always finds 1, 6, and 11 Wi-Fi channels as the dominant channels in Engineering Lab, as listed in Table 3.6. Unfortunately, it is not possible to use channel 0x19 and 0x1A in the network of Waspmote sensor nodes, since Waspmote node uses XBee-PRO transceiver which has only 12 channels as illustrated in Figure 3.3. It can be seen from Table 3.6, that channel 11 is the most dominant channel through the scanned Wi-Fi channels, and channel 1 has the lowest signal, therefore, at the first run of the network, a setting message from gateway will be broadcasted to set the whole Waspmote network nodes to channel 0C (which is far from channel 11 of Wi-Fi channels) to avoid Wi-Fi interference.

Table 3. 6: The dominant Wi-Fi channels in the Engineering lab.

#	Wi-Fi channel	Status	High signal	RSSI	SSID
1	1	active	40%	-70dBm	eduroam
2	6	Active	68%	-64dBm	eduroam
3	11	Active	81%	-56dBm	eduroam

3.2.8. Congestion criteria

The main purpose of this project is to control the network congestion to achieve better network performance. Hence, it is necessary to measure the buffer space of any route node in each program loop to notify other nodes about the probable congestion.

As mentioned in Table 3.2, the Waspmote board has 3 memories: Flash memory, EEPROM, and SRAM. The execution code can read or write to SRAM, so in this context, SRAM is segmented for several purposes by running code, (Figure 3.17). These segments are:

- Static data segment is a reserved space by the program code for all the global and static variables used in the code.
- Heap segment, which is for used dynamically allocated data items. Hence, the allocation of data will make the heap to grow up from the top of static data segment. In this segment, the received packet is stored before processing or sending it to another node.
- The Stack segment is used for storing local variables, and may be used by a running function for adding some state data to the top of the stack; removing these data from the stack is the responsibility of the function.

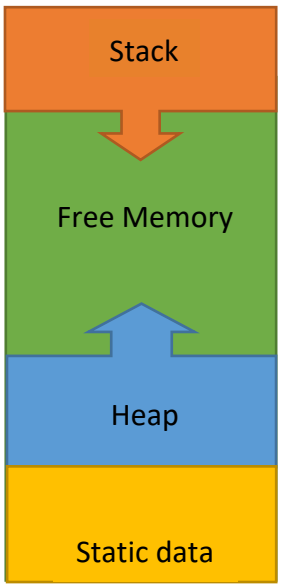


Figure 3- 17: SRAM segments after code execution.

The space between the heap and the stack is the free memory space. The API function to measure the free memory space of Waspmote node is: freeMemory( ), this function returns the number of free buffer (RAM) bytes.

To minimize node congestion, the free memory space should be as large as possible. In this project, a new technique has been used to get large free memory space. This technique is through variable declaration. Since, the Waspmote code is combined from three parts as illustrated in section 3.2.2, declaration part, Setup function part, and Loop function part. The technique suggests that any variable used only in either function setup or Loop should be declared in this function, and any variable used in both functions should be declared in the declaration part of the code as a global variable, this is because the memory size measurement doesn't count the size of variables declared in setup or loop function, but it counts the size of variable declared in declaration part of the code. Therefore, the measured memory size will be larger, and this will mitigate the node congestion.

Waspmote SRAM buffer size is 8kbytes, using freeMemory( ) function will measure the free memory space after the execution of the code. Figure 3.18 shows the free memory space values of a node under congestion.

```
Data: <=>€#394780769#Node_01#0#STR:XBee frame#BAT:88#
free Memory:4837
Data: <=>€#394779126# Node_02#0#STR:NODE2#BAT:94#
free Memory:4692
Data: <=>€#394780769# Node_01#1# STR:XBee frame#BAT:88#
free Memory:4547
Data: <=>€#394779126# Node_02#1#STR:NODE2#BAT:94#
free Memory:4402
```

Figure 3- 18: Sample of received data in a node under congestion, presenting the available space memory after each received packet.

3.3 Synchronizing the node clocks

The requirements for precise time by many protocols and WSN application make the time synchronization very important. For example, in the application of environmental monitoring, the synchronized clock is necessary to give the user exact overview of what is happening in the monitoring area at the same time. Also, to give the right direction of a moving car, the two different times of detecting the moving car should be compared meaningfully. Clock synchronization is required in TDMA configuration to avoid interference between sensor nodes. Additionally, for efficient control of using the limited energy resource in the sensor node, WSNs typically subject to periodic sleep-awake

cycles, these cycles need an accurate time synchronization (Benzaïd et al., 2017; J. He et al., 2017; G. Huang et al., 2012).

There is no global clock or common memory in distributed systems, and there is a particular internal clock and time for each processor. These clocks can easily drift seconds per day, accumulating significant errors over time. Also, these clocks may not remain synchronized although they might be synchronized when they start, this might happen because different clocks tick at different rates. Serious problems can happen to applications that depend on a synchronized time.

In this project, clock synchronization has been achieved using two methods: 1. relative synchronization in which each node synchronizes to other nodes without the need of exact real-world time, this method is the required synchronization for the experiments of this research. 2. Synchronization to real world time. In second method, the synchronization is achieved in two phases: external synchronization and internal synchronization. For external synchronization, we synchronized a reference Waspmote node to an external time like Universal Coordinated Time (UTC). In the internal synchronization, the reference node will broadcast packets with an explicit timestamp to synchronize all the other nodes.

The normal tool for providing time information in many existing synchronization protocols is GPS. But as mentioned in section 3.2.6 the use of GPS is not possible indoors, so in this project the external synchronization of the Waspmote reference clock is done by using a WiFi-PRO module as a second communication module on the Waspmote. The WiFi-PRO module allows the Wasmote's RTC to be synchronized to Network Time Protocol (NTP) servers.

3.3.1 Review of relevant clock synchronization protocols

Much has been published about the clock synchronization protocols and algorithms. In this section, some synchronization algorithms relevant to the intended method in this project will be reviewed.

The present clock synchronization protocols for WSNs are divided into the following two groups:

- Internal clock synchronization: in this group, the important requirement is that the nodes in the WSN should be connected. Nodes synchronization is achieved either by a synchronization to a reference node or the nodes synchronized with each other in peer to peer style.

- External clock synchronization: in some applications like weather monitoring and environment monitoring, there are physical environment constraints that divide the WSN into several connected components. In this case, only the external synchronization can be used to synchronize the entire network, because it's infeasible to synchronize the whole network by applying the internal synchronization in each component. External synchronization is achieved when one or more nodes are synchronized to external standard time like UTC, then these nodes will be used to synchronize the remaining nodes of the network (Swain & Hansdah, 2011).

Clock synchronization protocols can be classified into another two categories:

- Continuous synchronization: the clock always remained synchronized by these protocols. So, whenever an event occurs, it can be timestamped with the synchronized clock. However, if events only occur occasionally, continuous clock synchronization dissipates the limited energy of the nodes particularly through idle times.
- On-demand synchronization: the clocks synchronization takes effect only at the start of the event, whereas at other times the clocks run unsynchronized with the power-saving mode switched on (G. Huang et al., 2012).

To better understand the synchronization mechanism, four distinct components related to packet delay need to be defined, Figure 3.19:

- Send Time is the required time to construct the timestamp packet by the sender. This time is composed of: algorithm processing time, operating system delay, and the transfer time of the constructed packet from the node to its network interface.
- Access Time is the waiting time the packet spent it in the communication part (transceiver part like the XBee transceiver in Waspmote node) before accessing the transmit channel. This time is specific to MAC protocol. Channel access can be achieved either by CSMA technique, for which the packet will spend a random time, or by using TDMA in which the sender waits for its slot before transmitting.
- Propagation Time is the travel time of the message from leaving the sender node to arriving at the receiver node. This time can be very small if the sender and receiver share access to the same physical media i.e. they are neighbours in an ad-hoc wireless network. On the other hand, the propagation time might dominate the delay in wide-area networks.
- Receive Time: the time spent by the receiver to process the message.

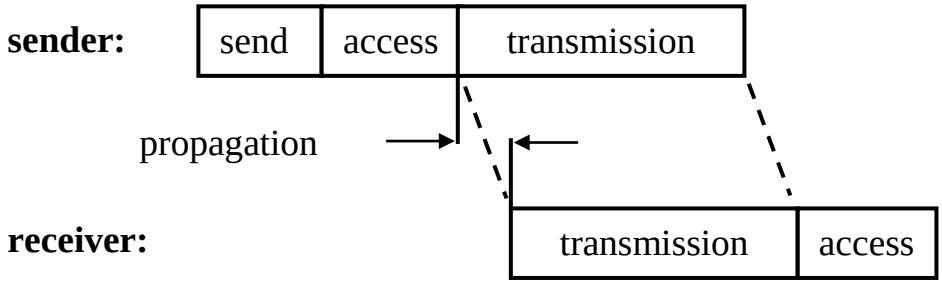


Figure 3- 19: Packet delay components over a wireless channel (Maroti et al., 2004).

In addition to the above components of delay, other clock parameters need to be defined. The *Clock rate* is defined as the rate at which a clock advances. The difference between the local times of two nodes is known as the *Clock Offset* parameter. The *Clock skew* is the difference in the frequencies of two clocks. The clocks may run at slightly different speeds and this leads to a drift among node clocks and drift from the ideal clock, this is known as the *Clock drift*.

Mills, (1989) proposed Network Time Protocol (NTP), this protocol is being used frequently in the domain of the internet. The NTP servers will be used for time synchronization from the clients in the order of milliseconds. The time servers are synchronised to external sources such as GPS.

In the rest of this section, the following synchronization algorithms will be described:

- Reference Broadcast Synchronization (RBS) was proposed by Elson *et al*., (2002). In RBS a beacon message is broadcast from the reference node to synchronize a set of receivers located within single-hop range. Each receiver records the local time as soon as the message has arrived. Afterwards, the receivers exchange the time of arrival between each other and compensate their own clocks. The technique in this protocol is opposite to traditional synchronization which synchronizes the sender with its receiver as in Figure 3.20 (left). Therefore, the advantage of RBS is the removing of the send time and access time from the critical path, as illustrated in Figure 3.20 (right), the synchronization starts after the send time and access time. Figure 3.20 shows that the only shared times between the nodes are the transmission time and access time (which are illustrated in the critical path- on the right) then the opportunity of precise synchronization will increase by decreasing the shared critical path.

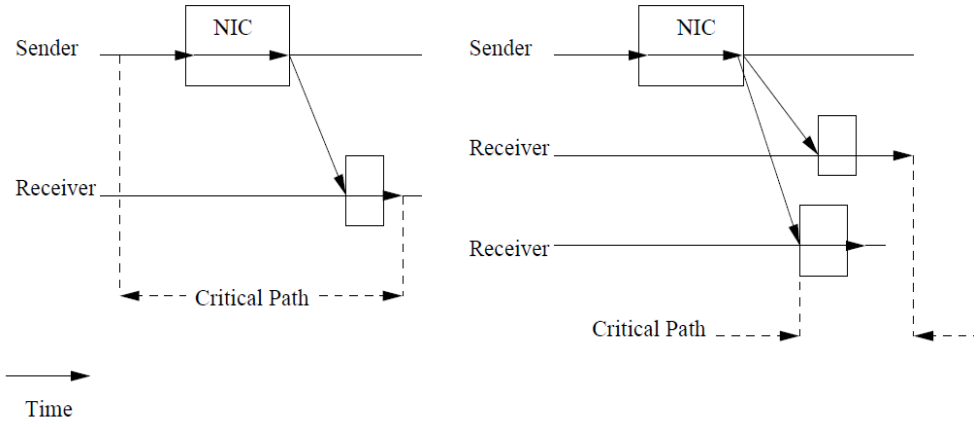


Figure 3- 20: (left) A critical path analysis for traditional time synchronization, (right) RBS protocol (Elson et al., 2002).

However, the exchange of time of arrival between the receivers endure a large overhead which is considered as a possible drawback to this technique (Elson *et al*., 2002).

- Lightweight Time Synchronization (LTS): this protocol uses two schemes: First, a synchronization with single hop will synchronize two pair-wise nodes, second, this scheme is expanded into multi-hop synchronization which consists of pair-wise synchronizations implemented on the spanning tree (a subnetwork which connects all the nodes using the main number of edges). In the pair-wise synchronization, two nodes can synchronize their local time through exchanging two packets between them. The transmit and receive times will be stored in each node. To achieve good result of synchronization time, the delay components that mentioned previously should be known and the protocol should be implemented identically for all nodes. At the end of the packets circulation between the pair nodes, the nodes will be synchronized after the subtraction of the unknown transmission time. In the extended scheme of multi-hop synchronization, after constructing a spanning tree structure, a pair-wise synchronization will be performed along the tree edges only (van Greunen & Rabaey, 2003).
- Average TimeSynch (ATS): a cascade of two consensus algorithms which is the base of this algorithm. This cascade is used for the tuning of the compensation parameters and converge nodes to a steady state virtual clock. To estimate the clock skew rate between the nodes, each node will broadcast the local timestamp in first stage. Secondly, virtual clock skew rate estimation will be broadcasted from each node, and

the receive node will combine this with the relative skew estimation to adjust its virtual clock estimation. The offset errors will be removed based on same principle (Schenato & Fiorentin, 2011).

3.3.2 Required hardware for clock synchronization

In this project, because of the high cost of a WiFi-PRO module, it's difficult to afford WiFi module for each Waspmote node for synchronizing the network to external time. Therefore, for the second synchronization method in this project a Waspmote node with two communication modules: WiFi-PRO and XBee-PRO is used as a reference node. The reference node has a Real Time Clock (RTC); this RTC through the use of WiFi-PRO module will be synchronized to external clock such as UTC in first phase. In second phase, the reference node will broadcast time beacons to synchronize the whole network. To cover the entire network, the XBee-PRO on the reference node will be set to high level transmission power and will use an antenna of 5dBi gain.

The WiFi-PRO and Waspmote RTC clock will be introduced in the following two sections.

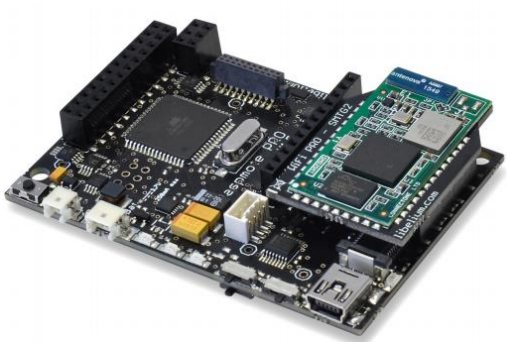


Figure 3- 21: Waspmote board with WiFi-PRO connected to socket0 (Libelium, 2017).

3.3.2.1 WiFi-PRO module

WiFi-PRO communication module (Figure 3.21) for the Waspmote platform enables the Waspmote node to make direct communication with any WiFi router. WiFi-PRO module conforms with IEEE802.11b/g/n protocols, and can be connected to socket0 or socket1 of the Waspmote board. In our case the connection will be to socket1, while socket0 will be used for connecting XBee-PRO. However, to connect WiFi module to the socket1, an expansion radio board must be used. The WiFi-PRO specifications are listed in Table 3.7.

Table 3. 7: WiFi-PRO module specifications.

Specification	WiFi-PRO module
Standard	IEEE 802.11 b/g/n
Operating Frequency	2.412-2.472 GHz
Number of channels	1 to 11
txPower	802.11b: 17 dBm, 802.11g: 14 dBm, 802.11n: 12 dBm
Sensitivity	-70dBm to - 94 dBm
Internet protocols	ICMP, IP, TCP, DGCP, FTP, HTTP, NTP
Antenna	UFL connector, On-chip antenna

3.3.2.2 Waspmote's RTC

An internal Real Time Clock (RTC) is built in to the Waspmote.

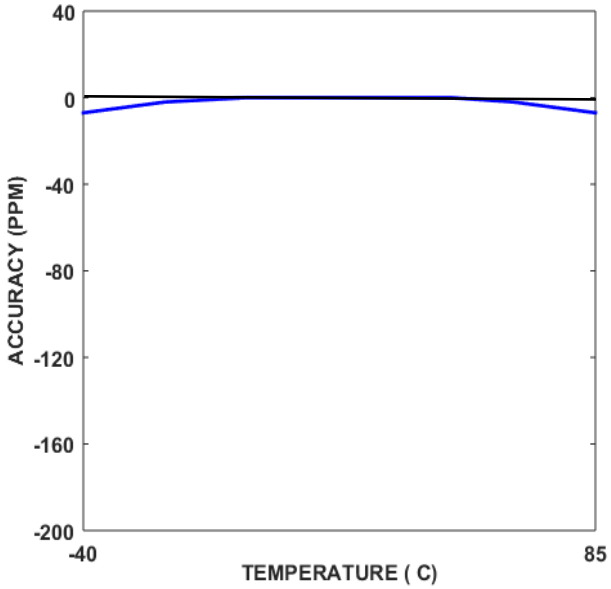


Figure 3- 22: Waspmote's RTC accuracy vs. temperature, there are minimal accuracy variations at the ends of temperature scale whereas the variation is zero at room temperature. (https://www.maximintegrated.com/en/app-notes/index.mvp/id/3566).

The Maxim DS3231SN is the used RTC in the Waspmote node. The DS3231SN operating frequency is 32,768Hz, and its operating temperature range of -40°C to +85°C. It has a loss of ± 2ppm (parts per million), which leads to variation of 0.173s per day (1.0512 min/year). Waspmote's RTC has an internal compensation mechanism which makes the accuracy variations with temperature close to zero at the ends of temperature range, Figure 3.22. The RTC is powered by Waspmote's battery, consequently the time data will be lost when the battery is removed or out of power. For this reason, most of the use of RTC time as a relative time not absolute time (Libelium, 2014; Maxim Integrated,

2015). After a long run of Waspmote node (around 72hrs) and the battery level dropped to 20% (which is the minimum required level for normal operation) the clock synchronization is remained the same as for full battery charge without any noticed change.

3.3.3 Algorithm description

In this project, clock synchronization is achieved with two methods: relative method and real-time method.

3.3.3.1 Relative synchronization

Some networks don't require exact real time at which the event occurred, but require only the order in which the events occurred. In this method, a homogeneous network of 40 uniform Waspmote nodes were deployed in the Engineering labs, and a Waspmote node called timer node broadcast virtual time packets to synchronize all network nodes. Figure 3.23 shows a picture of part of the indoor testbed. The timer node was set to maximum transmission power level and has an antenna gain of 5dBi.

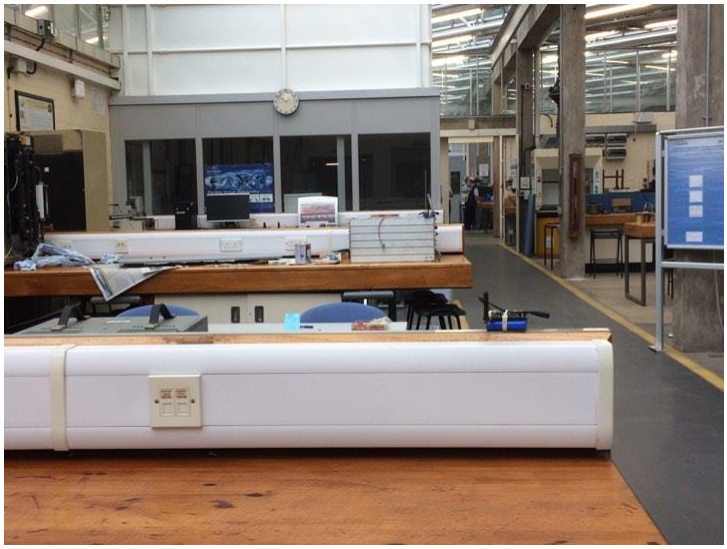


Figure 3- 23: picture of indoor environment where 40 Waspmote nodes have deployed (Engineering lab – University of Leicester).

To adjust each Waspmote's RTC, after switch on the node waits to receive the time packet from the timer node. The time packet from the timer node consists from 21 bytes as shown in Figure 3.24. Its first byte is a special letter ('Q') to inform all waiting nodes that this packet is a time packet.

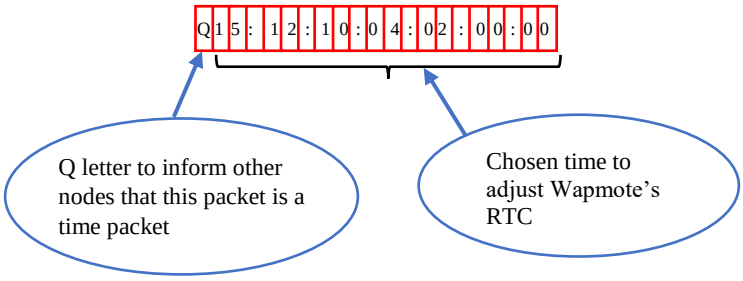


Figure 3- 24: time packet broadcast from timer node.

step1- START
step2- Switch ON Waspmote node
step3- Waspmote node waits until receiving data packet
step4- check:
if: 1st byte of the packet is "Q" *, then
Adjust Waspmote's RTC from the time value in remaining 20bytes
else: got to step3
end;
step5- STOP

***** *special letter used to inform the receiver that this a time packet.*

Figure 3- 25: Algorithm for relative time synchronization of sensor nodes.

Each node that receives the time packet will adjust its RTC according to the time data included in the packet using the algorithm in Figure 3.25.

The time taken for the packet sent from timer node to arrive at the receiver consists of several components: send time, access time, propagation time, and receive time. However, for this case the send time and access time are the same for all nodes since the packet is from the same sender. On the other hand, the propagation time and receive time may not be the same for all nodes. As it propagates over the wireless channel the packet may be corrupted or affected by different propagation interactions such as reflection, diffraction, …etc. also, the propagation time depends on the node place. In addition, for the receive time the time from receiving the packet to processing it in the node may be different from node to node. Consequently, the time path to reach each node may be different for different nodes. Figure 3.26 shows the critical-time path of the time packet from the sender to receiver side with all delay components.

The analysis of the time nodes after the RTC adjustment show that the average difference between clock nodes in the order of microseconds as explained in section 3.3.4.

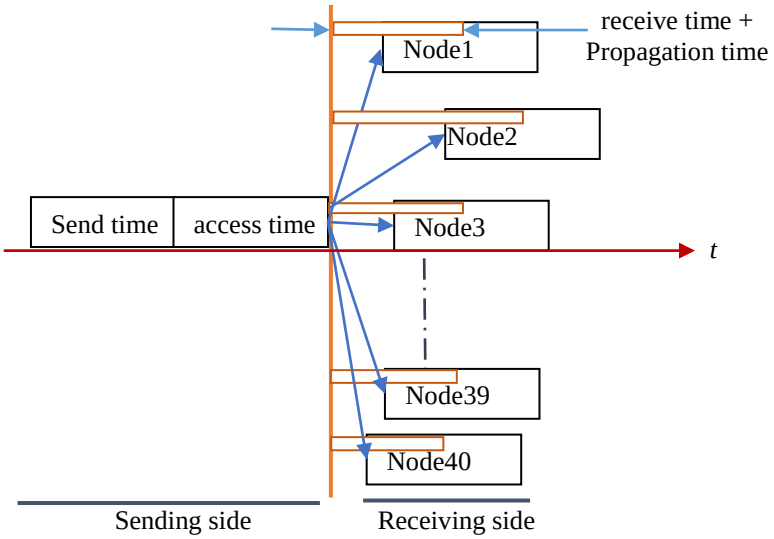


Figure 3- 26: Critical-time path of the sent time packet with decomposition of the delay components.

3.3.3.2 Real time algorithm

The second algorithm for synchronizing Waspmote nodes in this project consists of two phases, an external synchronization phase followed by an internal synchronization phase (Figure 3.27).

External synchronization phase: In this phase, a Waspmote node called the reference node employs two communication modules (WiFi-PRO and XBee-PRO) and will synchronize its RTC to external source of time such as UTC signal. The synchronization will be achieved through the communication of WiFi module with one of NTP time servers, as mentioned in section 3.3.2.1. The steps of updating the Waspmote's RTC setting to internet time are illustrated in Figure 3.28.

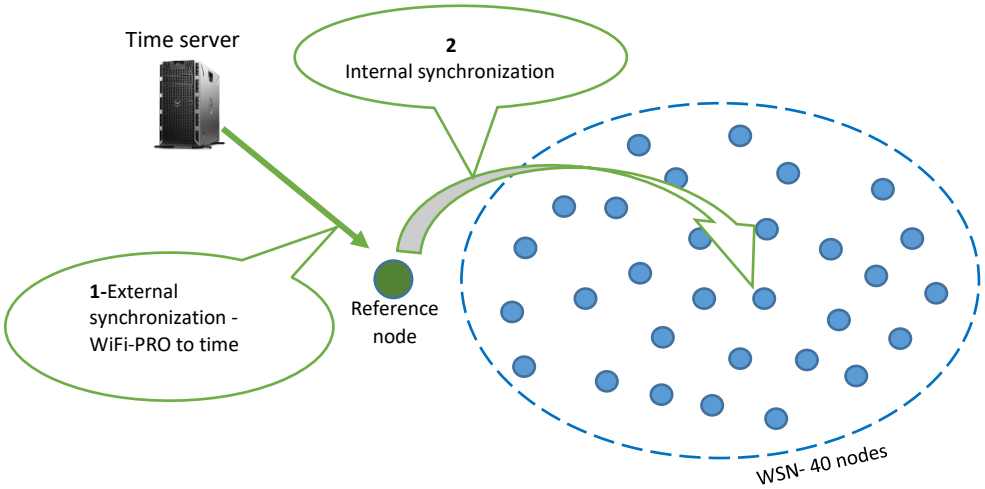


Figure 3- 27: Real time synchronization diagram (Internal and External synchronization).

Internal synchronization phase: After synchronizing the reference node to external UTC time through WiFi-PRO module, the reference node broadcasts beacons consisting of packets contain real time value to synchronize all network nodes. In this algorithm, after the reference node synchronizes its RTC to the external clock and before it broadcasts its synchronization packets, some steps should be followed to prepare network nodes for accurate real-time synchronization.

Step1: All Waspmote nodes including the reference node should be restarted to run with same time. The restart will be done through broadcasting a restart packet from the reference node, then all nodes will be restarted at the same time.

Step2: Waspmote node has an API function "milli( )", which counts the number of milliseconds the current code has been running since the last reset. The reference node will insert the number of milliseconds and the real-time in the broadcast time packet.

Step3: At the receive node, the inserted number of milliseconds time will be subtracted from the number of milliseconds of the receiver node, this calculation will give the time in milliseconds for the journey of the broadcasted packet from reference node until arriving at the receive node. However, to find the average required time for the packet to be sent from the sender node to the receive node, a Waspmote node sends 1000 packets to a Waspmote receive node, then the required time for each packet is calculated as explained above. The intensive measurements of the time journey of the broadcast packet is around 185ms with a Standard Deviation (SD) of 3.457ms, this means it is less than a second, this 185ms can be subtracted from the 1 second to give 815ms which can be used as a delay time adjustment as illustrated in *step 4*.

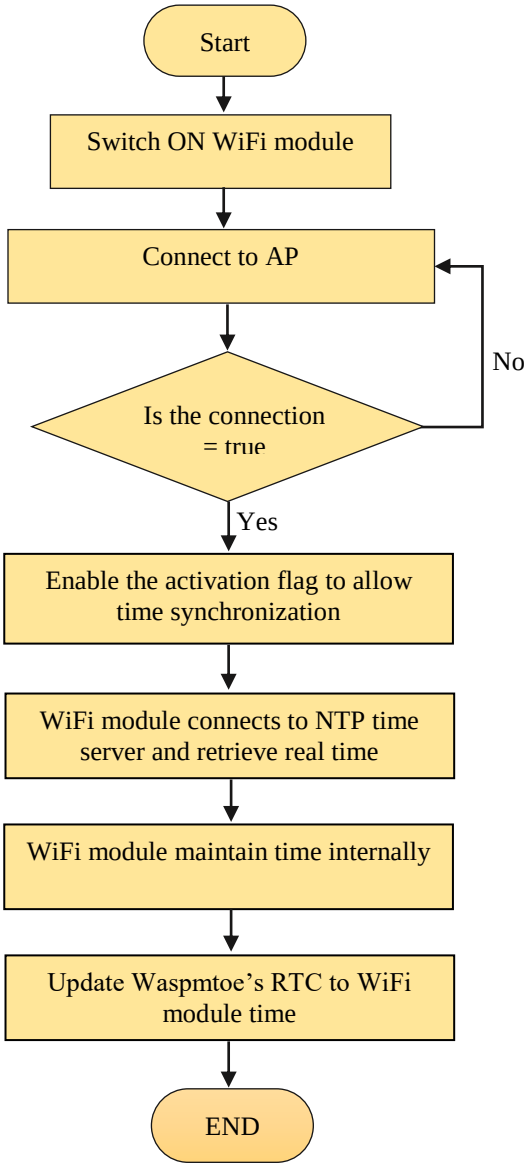


Figure 3- 28: Flowchart of synchronizing Waspmote reference node to NTP time server.

***Step4*:** At the receive node, the seconds part of the received real-time value will be increased by 1. Then the delay time value will be:

delay in milliseconds = 1sec - number of milliseconds of journey packet

This delay will be used by the receive node as a complement to 1sec, then when the delay ends the adjustment of RTC will be to the exact real-time value.

3.3.4 Synchronization results

To test the time accuracy of the RTC, a Tektronix oscilloscope (Figure 3.29) has been used to measure the time difference between two nodes after synchronization. A

Waspmote board has sensor I/O connector pins and the digital pins can be configured as an output pin. That was connected to one of oscilloscope channels.

For relative synchronization, a random selection of pairs of Waspmote nodes was used in testing the differences between their clock time. All the tested pairs show differences in the order of microseconds. For example, Figure 3.30 shows the difference in time for two square waves generated at same time from two Waspmote nodes, the difference is 40µsec. This level of accuracy in setting the RTC is acceptable for the measurements presented in this and later chapters.

Figure 3- 29: (left) Testbed used to test the difference between RTC of the synchronized Waspmote nodes, (right) Description of sensor connector pins.

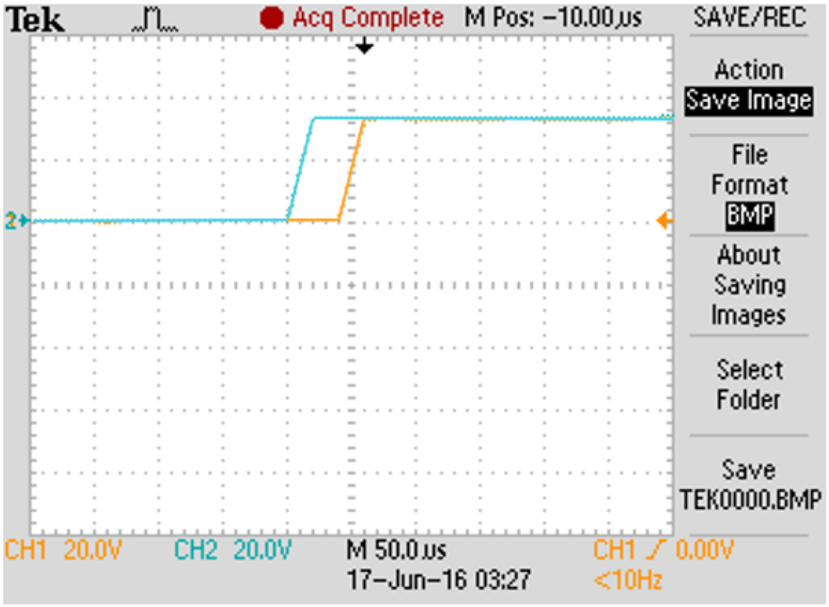


Figure 3- 30: The difference in RTC between two Waspmote nodes under relative synchronization.

While the results of real time synchronization method show that the differences between the nodes under internal synchronization in the order of microseconds, the difference of each node with respect to reference node is in the order of milliseconds (around 5ms) as in Figure 3.31. This difference in milliseconds stems primarily from the difficulty of calculating the exact time required to insert the real time in the packet and send it to other nodes.

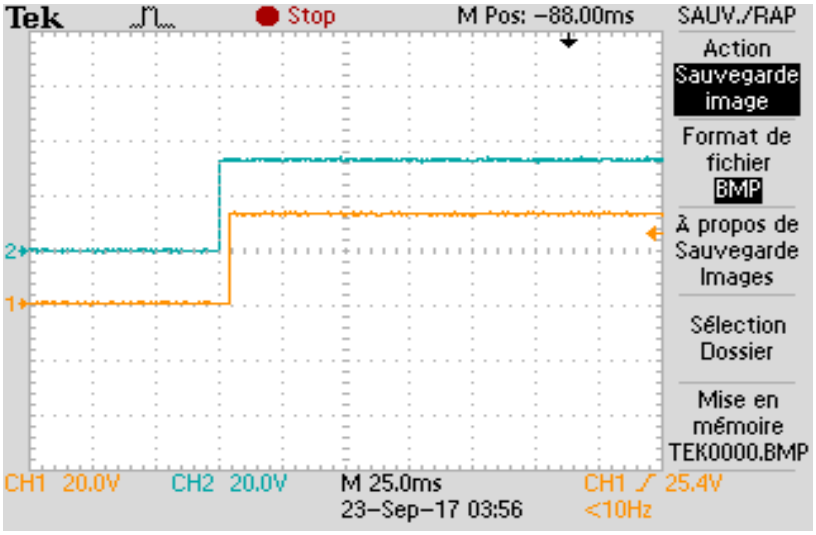


Figure 3- 31: The difference between synchronized node and reference node in milliseconds.

3.4 Using EEPROM

Waspmote board has a non-volatile memory called Electrically Erasable Programmable Read-Only Memory (EEPROM) of 4kbytes size. Waspmote nodes reserve the addresses from 0 to 1023 for saving important data, so these addresses must not be over-written. Therefore, only the last 3kbytes from 1024 to 4095 are available for data storage. EEPROM can be used to store the variables and network setup information, because these values might be lost when the node is reset or there is no battery. In this project the EEPROM has been used in two cases.

First, sometimes Waspmote node faces heavy congestion and its buffer will be affected by memory leak, and consequently, the node cannot store more packets which then leads to large packet loss (see memory management section). To solve the memory leak problem the node must be restarted, and therefore the node then needs the setup information from EEPROM.

Second, in this project the analysis of the results need to check every node setup, and because it is not practicable to connect a laptop to observe each node setup, setup information written in EEPROM can be retrieved at the end of the experiment.

3.5 Memory management

In this project a node is said to face congestion, when the free memory fells below 25% it will notify the nodes that are sending it data to change their route. The reason to choose the value of 25% for congestion notification is to give the nodes enough time to choose their route before the congestion starts to affect the network performance. The congested node will send its remaining stored packets on the same route and check the free memory after the sending of each packet. However, in some cases due to heavy congestion, the congested node cannot free the buffer space between the heap and the stack and the value of free memory space will remain under the value of 25% of the buffer space of the first code execution. In this case, the node will check both the number of stored packet and the free memory space at the same time, and if the number of stored packet reaches zero but the free memory space is still under the value of 25%, this indicates a memory leak, which happens when the code consumes memory but is unable to release it back to be used again and the memory full of fragmented places as in Figure 3.32. To solve this problem, a technique of suing memory pool can be used to remove the effect of memory leak. The memory pool is a memory block which can be got from system and use some unit of it to replace the system call of allocating memory (by using new/delete instructions). This method of creating memory pools might increase the node code size and then decrease the already small node buffer. The second solution for removing the memory leak is that the node will be rebooted to refresh the memory. After restart (the reboot operation takes around 1 sec) the node will upload its set up setting from the EEPROM and will then be ready to work as normal, Figure 3.33 shows the flowchart of the algorithm solving the memory leak problem (Amir, 2000; Ferres, 2010; Last & Edt, 2016; Rafkind *et al*., 2009).

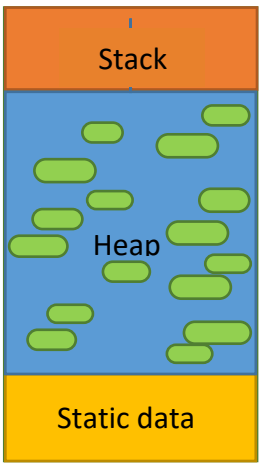


Figure 3- 32: Memory leak and fragmented Heap memory.

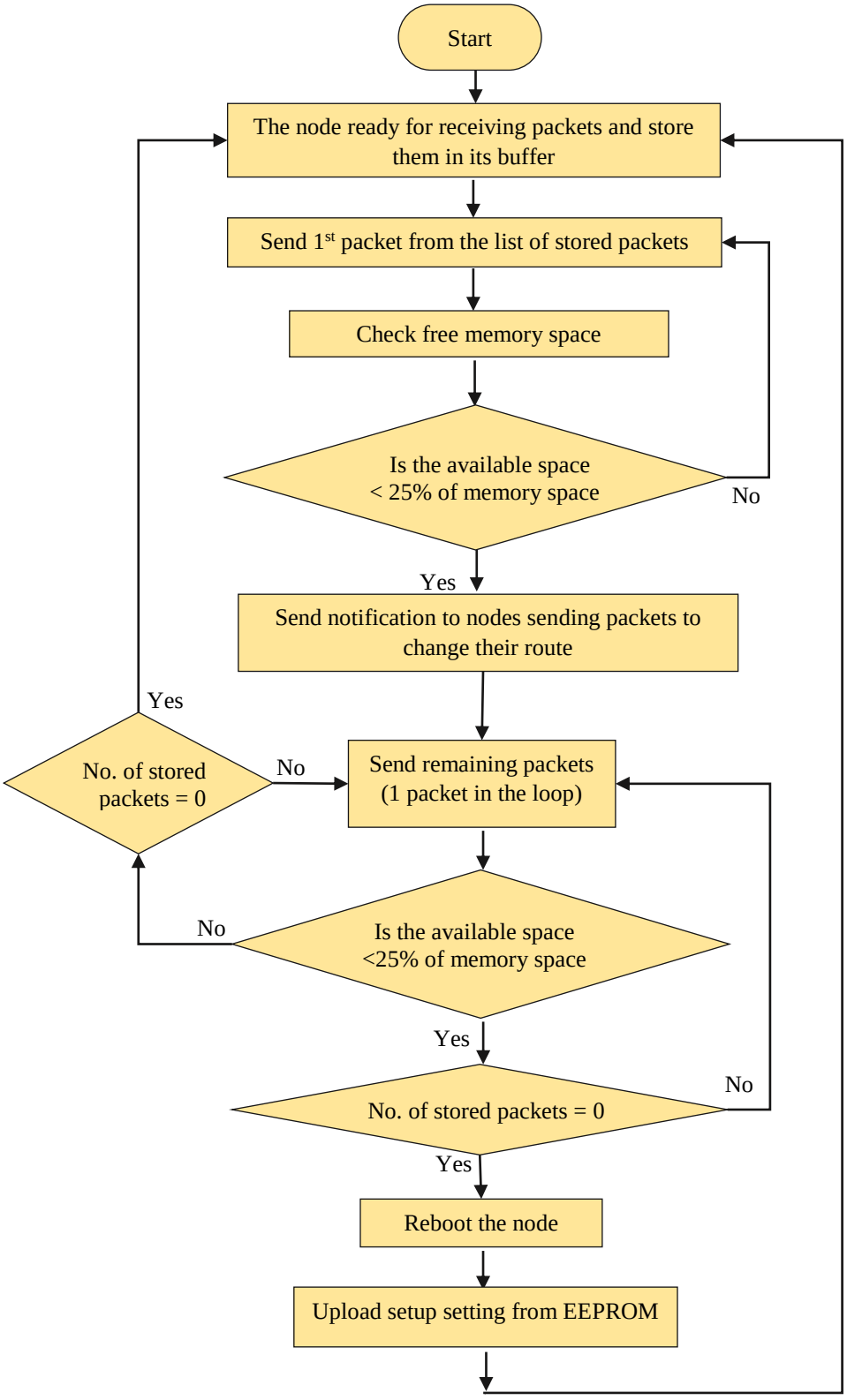


Figure 3- 33: Flowchart of solving memory leak problem in Waspmote board.

3.6 Including necessary information in data packet

In order to diagnose the functioning of the network, some diagnostic information is included in the data packet: the generation time of the packet, and the node IDs of the route from source to the sink.

3.6.1 Packet generation time

At the time of generating the frame in the source node, the node reserves 6 bytes (2bytes hrs, 2 bytes minutes, 2 bytes seconds) at the beginning of frame payload, as in Figure 3.34a. Although the generation time gives advantage in data analysis, the generation time consumes 6 bytes from frame payload. In that case, the generation time can be inserted in the frame header instead of node serial number, Figure 3.34b.

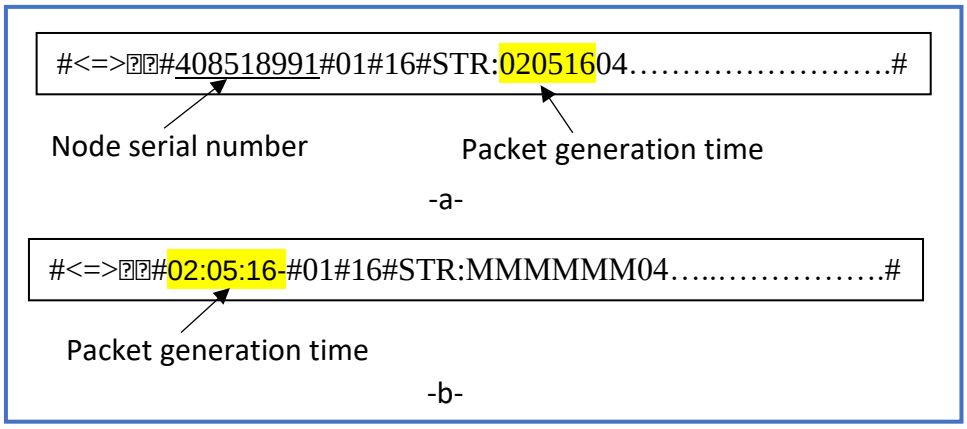


Figure 3- 34: a- generation time in frame header, b- generation time instead of node serial number.

3.6.2 Route node IDs

The node IDs of the route from source to sink can be added to frame payload. At each node, the packet route is updated by adding the current node ID, which is started from 1st hop. Figure 3.35 shows a route from source to sink with node IDs.

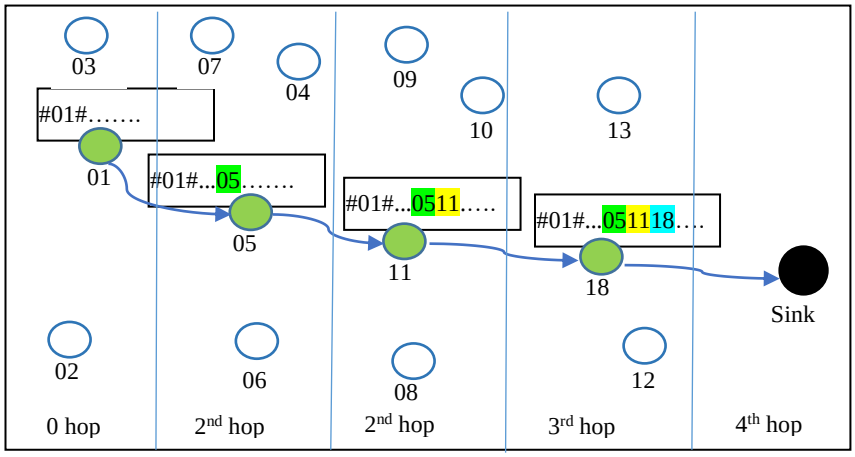


Figure 3- 35: Node IDs inserted in the frame payload from source to sink.

3.6.3 Effects of including diagnostic information in data packet

Including diagnostic information in frame payload will consume number of bytes from the payload as well as adding processing delay time. To decrease the effect of the extra bytes a coding technique has been used to reduce the number of used bytes by half. To verify the effects of including the diagnostic information in the data packet. The time required to send the data packet from the source to sink has been measured. For this verification, a WSN with four hops was used as in Figure 3.35.

The coding technique is achieved by the following steps (Figure 3.36):

step1- START

step2- Convert both bytes from char type to int type

step3- Shift the bit list of 1st byte to left 4 times (<<4)

step4- OR operation between the bits of the two bytes to generate coded byte

step5- Reconvert the resulted byte to char type, ready for sending the packet

Figure 3- 36: steps of coding two bytes into one byte.

Figure 3.37 shows a packet before and after coding operation, the yellow shaded (time) is minimised from 6 bytes to 3 bytes, and the green shaded (node IDs) for 3 nodes is minimised from 6 bytes to 3 bytes.

Before coding <=>€ #408520689#01#142#STR: 234912051118……#

After coding <=>€ #408520689#01#142#STR: SyB5AH……...……#

Figure 3- 37: data packet before and after coding technique, the yellow colour is for the time before and after coding, and the green colour is for the nodes route before and after coding.

As an example, let's take the two bytes of the hour (first 2bytes of the time) which is 23, the coding result is (S) and in after coding packet. This coding operation is illustrated in Figure 3.38, the ASCII table is in Figure 3.39.

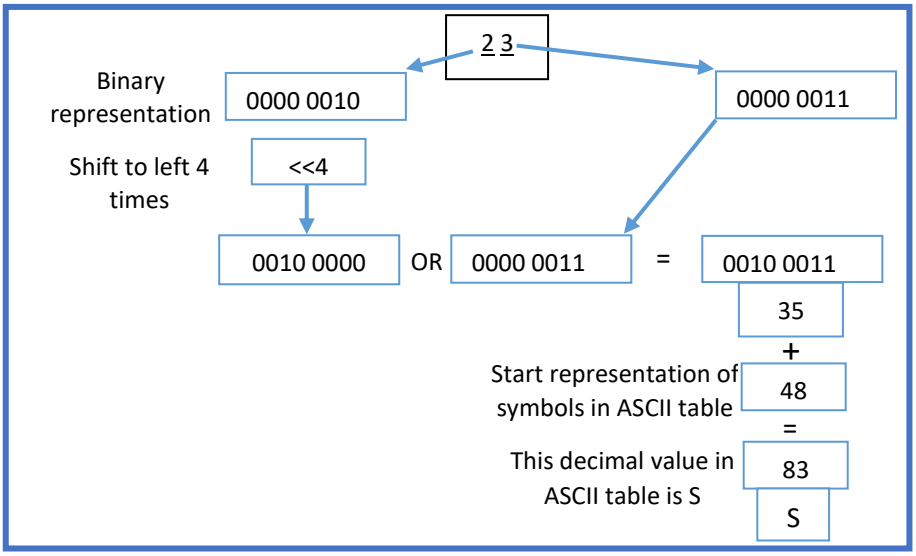


Figure 3- 38: Representing the steps of coding two bytes according to ASCII table.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Figure 3- 39: ASCII table (https://simple.wikipedia.org/wiki/ASCII).

The decoding operation of the results in analysis stage is illustrated in Figure 3.40.

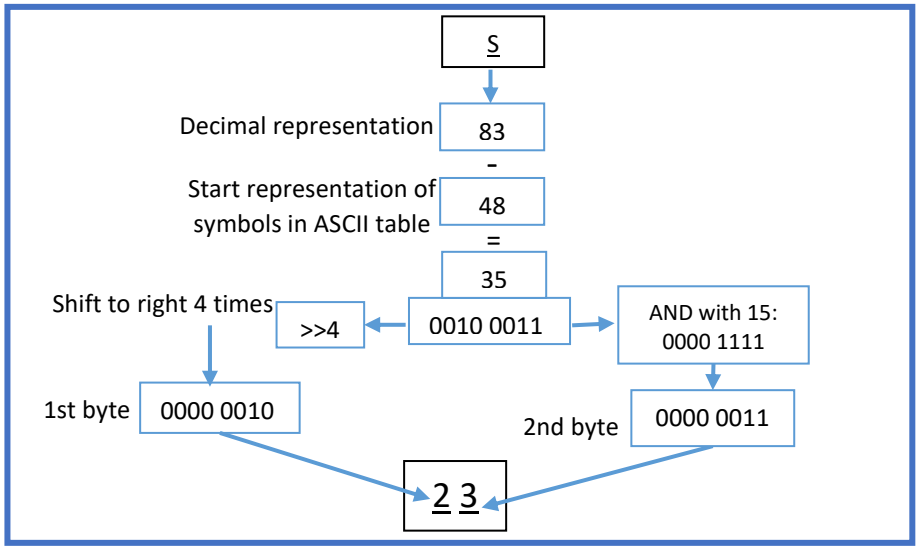


Figure 3- 40: Flowchart of decoding operation of the received coded byte into two bytes.

Three different formats of diagnostic information were sent for this verification and the required sending time from source to sink has been measured, Figure 3.41.

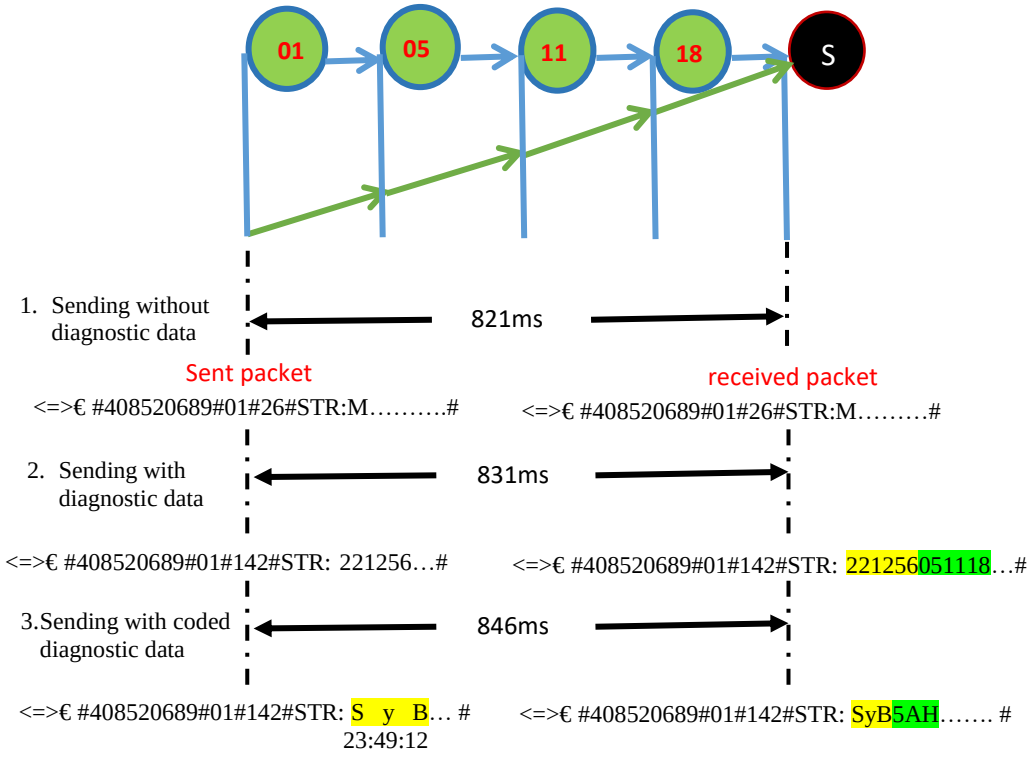


Figure 3- 41: Time for data packet to travel from source to sink.

1) Sending data packet without any diagnostic data: the average required time from source to sink is 821ms with Standard Deviation (SD) of 4.92236.
2) Sending with diagnostic data: the included information is 6 bytes for generated time and the ID of each node of the route from source to node. The average required time is 831ms with SD of 5.3607.
3) Sending with coded diagnostic data: the number of used bytes for generated time and node IDs is decreased to half by coding each 2bytes to 1byte. The average required time is 846ms with SD of 3.57413.

The normal distribution of the three scenarios for sending packet data from source to sink is illustrated in Figure 3.42. For 1st scenario sending without diagnostic data, the mean value is 821ms and the SD is 4.92236. Next scenario is sending with diagnostic data, the mean value is 831ms and the SD is 5.3607. Last used scenario is sending with coded diagnostic data, the mean value is 846ms and the SD is 3.57413.

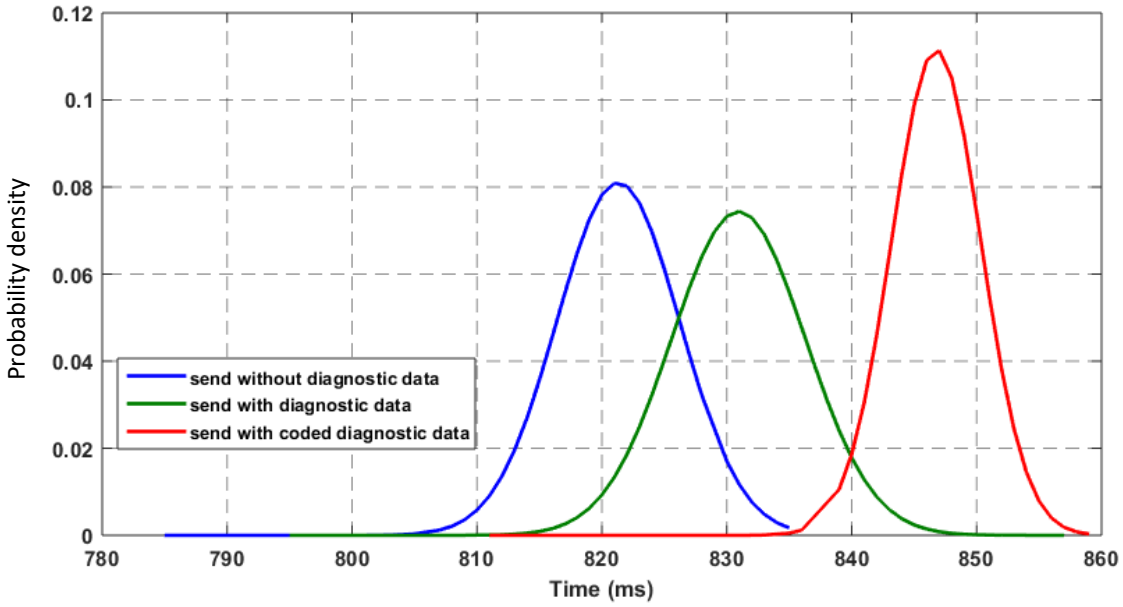


Figure 3- 42: Normal distribution of the required sending time of the three sending scenarios.

In conclusion, the mean value of the exact time required to include coded data 15ms. It is a trade-off between decreasing the number of used bytes for investigation and between the additional delay to the required time from source to sink.

Chapter 4: Congestion Control Using Alternative Path

4.1 Introduction

As introduced in chapter 1 and 3, the aim of this project is to control the congestion in WSN with a combination of resource control (alternative path) and traffic control. In this chapter, the resource control algorithm Hierarchical Tree Alternative Path algorithm (HTAP) will be investigated.

In addition, the preliminary results of experiments studying congestion control using alternative path are presented in detail. Finally, the collected data from these experiments were analysed and studied to view the network performance after applying the HTAP algorithm of congestion control.

4.2 Congestion control algorithm

Node level congestion in WSN may be controlled by utilizing resource control (using alternative path) or by traffic control. The resource control method used here is HTAP (Sergiou *et al*., 2013). The HTAP algorithm attempts to solve congestion in WSNs by employing of alternative paths to avoid congested areas. The alternative path is created from unused nodes in the initial path from source to sink.

In order to test HTAP algorithm of congestion control and to develop new methods an experimental facility consisting of up to 40 Waspmote nodes was deployed in an area of 2250m^2 (labs of Engineering Department-University of Leicester) as illustrated in Section 3.2.6.

The experimental work assumptions are as follows:

- Waspmote nodes are deployed densely in the network.
- There is one sink is located at one side of the lab, and the number of sources is variable.
- Each Waspmote node knows its position and the position of the sink.
- All nodes are considered to be identical (same buffer size, same battery,… etc.).
- CSMA/CA is the MAC protocol of all nodes.

The aim of experimental work is to test the four different schemes within the HTAP algorithm. These schemes are: **Topology control, Hierarchical Tree Creation, Alternative Path, and Handling of Powerless and/or Failed Nodes**.

4.2.1 Topology control

The random location of sensor nodes in the initial deployment may produce some very dense areas which have a high number of redundant nodes. Message collisions will be increased in this area. Nearby nodes will provide several copies of the same information. As a result, the network lifetime will be reduced. However, the topology of the network can be changed by modifying some parameters of the nodes, e.g. transmission power, sleep/active duty cycle…etc.

Topology Control is a technique used to alter the underlying network by reorganizing and managing the node parameters and the modes of operation (active or sleep) periodically. These factors will extend the network lifetime, maintain sensing connectivity, and coverage (M. Li *et al*., 2013).

Topology control is divided into: topology construction, and topology maintenance.

- **Topology construction**: concerned with initial topology by reducing the number of active nodes and managing node parameters.
- **Topology maintenance**: is responsible for preserving node coverage and connectivity, particularly when there is a dead node or congested node topology control recreates a path from unused nodes. An example of this technique is the Minimum Spanning Tree (MST).

Since the focus in this project is to control the congestion of the node in WSN, and the control is achieved by using an alternative path to avoid the congestion, it is necessary to employ unused nodes for creating extra paths.

4.2.1.1 Local Minimum Spanning Tree (LMST)

In general, connecting all vertices (nodes) of any graph and represents it in a subgraph is called a spanning tree, and the graph may have many spanning trees. The spanning tree cannot be disconnected and it does not have cycles. Therefore, there is no spanning tree in the disconnected graph because all its vertices cannot be spanned. In Figure 4.1- a, there are three different possible combinations of the spanning trees.

The Graph is considered to be directed if each edge (the connection between vertices) has orientation, see Figure 4.1 b and c. In this context, the minimum spanning tree is the spanning tree with minimum sum of edge weights among all possible spanning trees in the graph. The spanning tree is only for the sub-network from source to sink and not the whole network. The weight can be the length of a route, the capacity of a line, the energy needed to move from a point to another point along a route, …etc (Graphs, 2017).

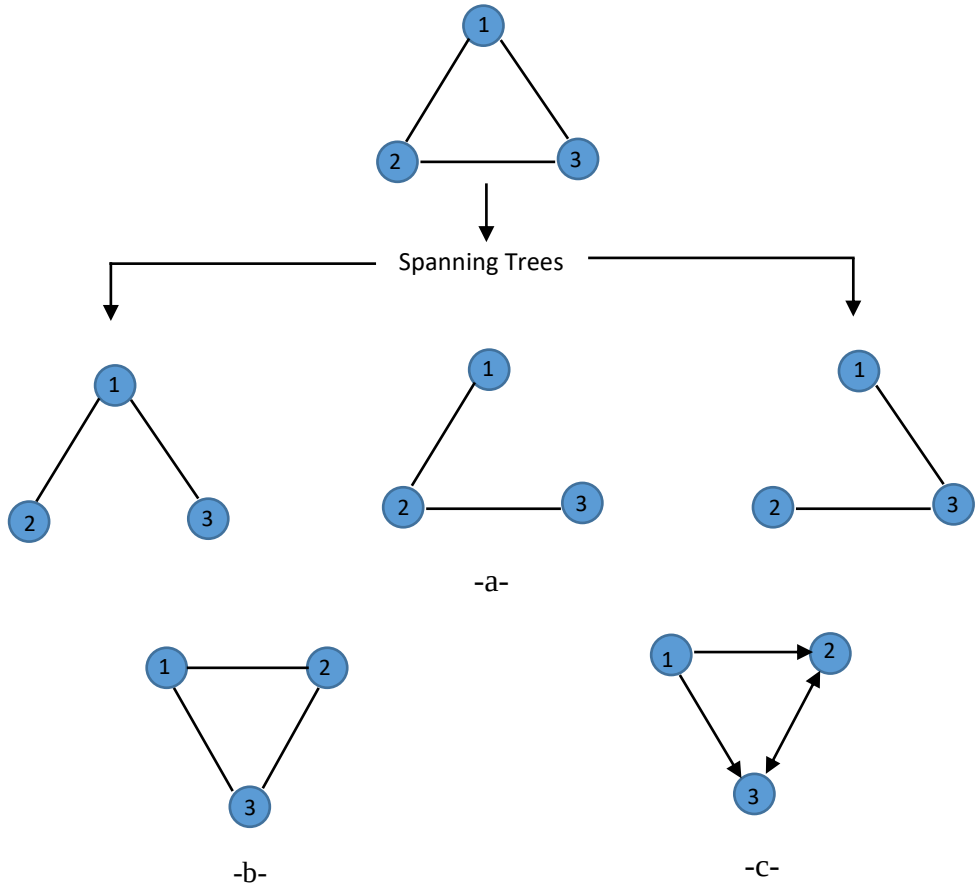


Figure 4- 1: a- possible combinations of spanning trees, b-undirected graph with 3 vertices, c- directed graph.

HTAP uses the Local Minimum Spanning Tree (LMST) for its initial topology control *(*Li *et al., 2003;* Sergiou *et al*., 2013). The LMST find the shortest path from source to sink. For our multi-hop WSN, each node of the network will build its own local minimum spanning tree by using Prim's (Prim, 1957) algorithm and only keeps on-tree nodes that are one-hop away as its neighbours in the final topology.

Prim's algorithm that finds the minimum spanning tree for a weighted undirected graph as follows:

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, which is here the edge length.
3. Repeat step 2 (until all vertices are in the tree).

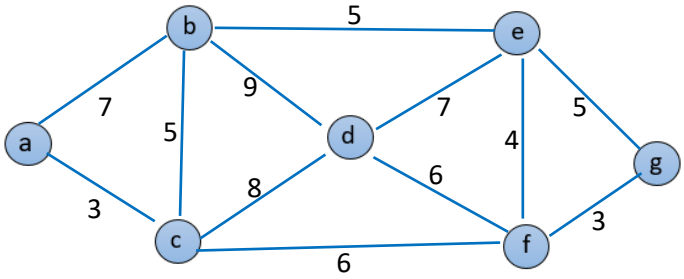


Figure 4- 2: Connected graph of 7 vertices (nodes), the numbers represent the weights of the edges.

In Figure 4.2, if we start with:

- step 0 at node a, then the lightest edge is **a-c** which is 3 (it represents edge length).
- Step 1, is **c-b**, which is 5.
- Step 2, is **b-e**, which is 5.
- Step 3, is **e-f**, which is 4.
- Step 4, is **f-g**, which is 3.

Then the last shape for the minimum spanning tree as in figure 4.3, the resulted tree is for Node **a** when it builds its local minimum spanning tree independently and keeps on the tree only neighbouring nodes which are one hop away.

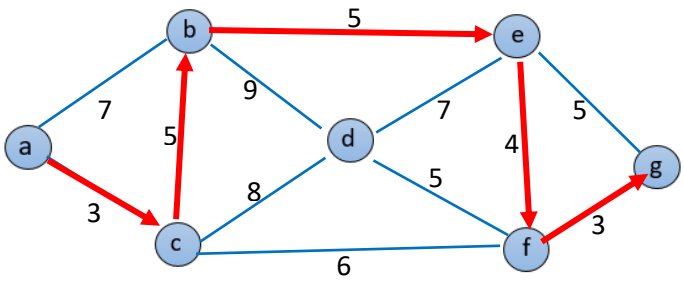


Figure 4- 3: Minimum spanning tree, using Prim's algorithm.

There is a unique spanning tree can be created from the sink, while there are many spanning trees can be created from the sources (one for each source) depending on which subset of nodes lie on the path from source to sink as in example of Figure 4-3, the nodes are a c b e f g and d is excluded. During the setup phase each source will build its own local minimum spanning tree from source to sink. The minimum tree is not absolute minimum, but a comprise between simplicity and optimization. Therefore, the reason that HTAP employs source-based trees instead of sink-based tree is that the first provides numerous alternative paths from source to sink (Sergiou, 2012).

LMST consists of the following phases:

1) Information collection, 2) Topology construction, 3) Determination of transmission power, 4) Construction of topology with only bi-directional edges.

1) **Information collection:** in this phase Waspmote uses the scanNetwork() function, to discover neighbouring nodes. The following information is collected for each node:

- Mac Address
- Node Identifier
- RSSI

For the experimental deployment (see Figure 3.14), each node must start the scan phase to find neighbour nodes, storing the collected data in the Waspmote's buffer. The collected data of the scanning phase (Mac address, Node Identifier, and RSSI) will be written in EEPROM for analyses, then at the end of the experiment and from the stored information scanning data it's easy to know each node's neighbours with their Mac addresses.

A section of the experimental WSN show the scanning phase of Node 07 is presented in Figure 4.4. The scanning phase data (Mac address and Node identifier) of Node 07 (derived from the data in EEPROM) is presented in Figure 4.5 with 6 nodes founds.

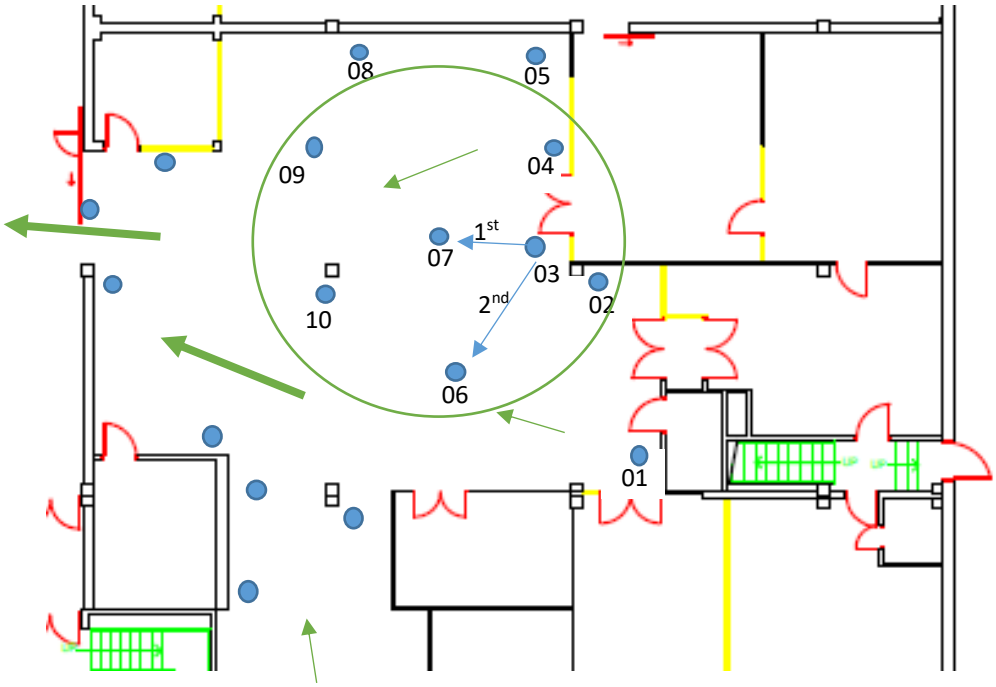


Figure 4- 4: Section of WSN with Node 07 scanning phase.

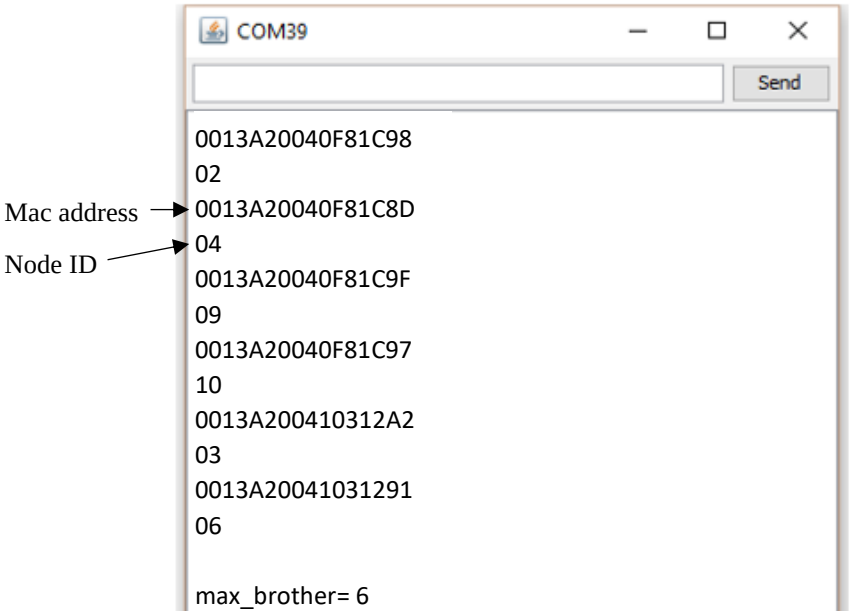


Figure 4- 5: Scanning phase data of Node 07, shows Mac address, Node identifier, and maximum number of scanned nodes.

2) **Topology construction:** Each node after scanning neighbour nodes builds its own LMST by using Prim's algorithm.

Three parameters should be considered before constructing source-based tree:

1) Each node must know it's (x, y) position in relation to the sink.
2) The tree should keep only the nodes, that each one of them has a connection to at least one node at a higher level (node levels will be explained in Section 4.2.2).
3) Maximum number of neighbour nodes is six.

Nodes know their own position (x, y) relative to the sink position. They will also collect the neighbour nodes positions by sending control messages to each neighbour node to send its x-y points.

The distance between the node and each of its neighbours will be calculated as explained in detail in Section 3.2.6.1. By considering the distance between the node and its neighbour node as an edge, and then using the minimum distance (minimum edge) for creating the spanning tree, this assumption contributes in building LMST. Employing LMST as a routing path from the source to sink is a power efficient technique, because longer paths than the minimum will increase the power consumption of WSN in general.

To illustrate the creation of LMST, Node 03 in Figure 4.4 will be considered as a source. From EEPROM of Node 03, the next hop nodes for Node 03 are Node 06 and Node 07. Node 03 will calculate the distance to both nodes (06 and 07) after receiving their position (x, y) points. Node 03 will arrange in a table the nodes of its next hop

starting from its nearest node, which in this experiment and from Node 03 EEPROM is Node 07 (as marked in the Figure by 1st and blue arrow), whereas the second node in the table is Node 06 (marked by 2nd). With the same method Node 07 will calculate the Euclidean distance with its next hop nodes to select the nearest node for building the next branch of LMST which is started at Node 03, and so on so forth up to network sink.

3) **Determination of transmission power:** After the scanning phase each node has the Received Signal Strength Indication (RSSI) of all its neighbour nodes. In case that any source or routing node needs to send a packet to any neighbour node it will use the RSSI related to specified neighbour node. Based on the value of RSSI the sender node will adjust its required power level so that the sent packet can reach the farthest neighbour. In the experiments of this project, the XBee power levels (see Section 3.2.4.1) related to RSSI values are illustrated in Table 4.1.

Table 4. 1: XBee power level related to the value of neighbour RSSI value.

Neighbour RSSI value	XBee power level
> = -50 dBm	0
-51 dBm to -60 dBm	1
-61 dBm to -70 dBm	2
-71 dBm to -80 dBm	3
< -90 dBm	4

4) **Construction of topology with only bi-directional edges:** the network consists of only bi-directional links which is important for link level acknowledgments and for the medium access control mechanism such as RTS/CTS.

4.2.2 Hierarchical Tree Creation

Each Waspmote node becomes source when it senses an event i.e. a sensor value becomes a typical, then it starts to create its Hierarchical Tree from itself to the sink.

At the end of topology construction phase any node is able to connect to a maximum of six nodes which are one hop away from itself. The reason for a maximum number of six nodes is analytically explained in N. Li *et al*., (2003), and also, as mentioned in Section 4.2.1, small node degree is desirable for minimum MAC-level contention.

Creating the path from source node to sink, includes constructing levels for the nodes of this path.

- *Level Construction:* For level construction, two assumptions have been made for the experiments presented in this thesis:
  1): the 1st line of deployed nodes (farthest from the sink) will be assigned as level 0, and the nodes of this level will ignore any level discovery message.
  2): the level discovery message will be ignored if it is coming from a node that is closer to the sink than the receiver node since the tree should be created in sink direction.

The levels are constructed as follows:

- Source node considers itself as level 0.
- Broadcasts level_discovery message (this message includes the node ID of the source and the level value which is level 1 for 1st broadcast) to neighbour nodes.
- The nodes that receives level discovery message will be assigned to level value included in the message which is here is level 1, the level of each one of these nodes is related to this source node.
- The level 1 nodes are considered as children to source node.
- In the same order, level_discovery message will be updated (by increasing the level value by 1) and broadcasted from each node of level 1 to allocate other nodes in new levels, and this process is repeated to assign each node in a level up to the sink.
- A node might receive different level discovery messages from different nodes, in which case the node will select a message that places it at a lower level.

At the end of level construction phase each node will be assigned to a level, for instance, if Node 03 in Figure 4.4 becomes a source it will create its hierarchical tree up to network sink, during this phase Node 03 starts to construct levels for its hierarchical tree, and it will start by assigning itself to level 0, then it broadcast a level discovery message to nodes that is under its communication range, these nodes are: 02, 04, 06, and 07. Nodes 02 and 04 will ignore the level discovery message because they are assigned as level 0 initially since they are in the 1st line deployment according to 1st assumption above. Then nodes 02 or 04 will be removed from the hierarchical tree of Node 03. As a result, nodes (06 and 07) will be assigned as level 1. Then each one of them will broadcast level discovery to create next level up to sink.

All farthest nodes are sources as illustrated with green colour in Figure 4.6, while after the experiments, the level of each node is taken from its EEPROM, with this indicated in Figure 4.6.

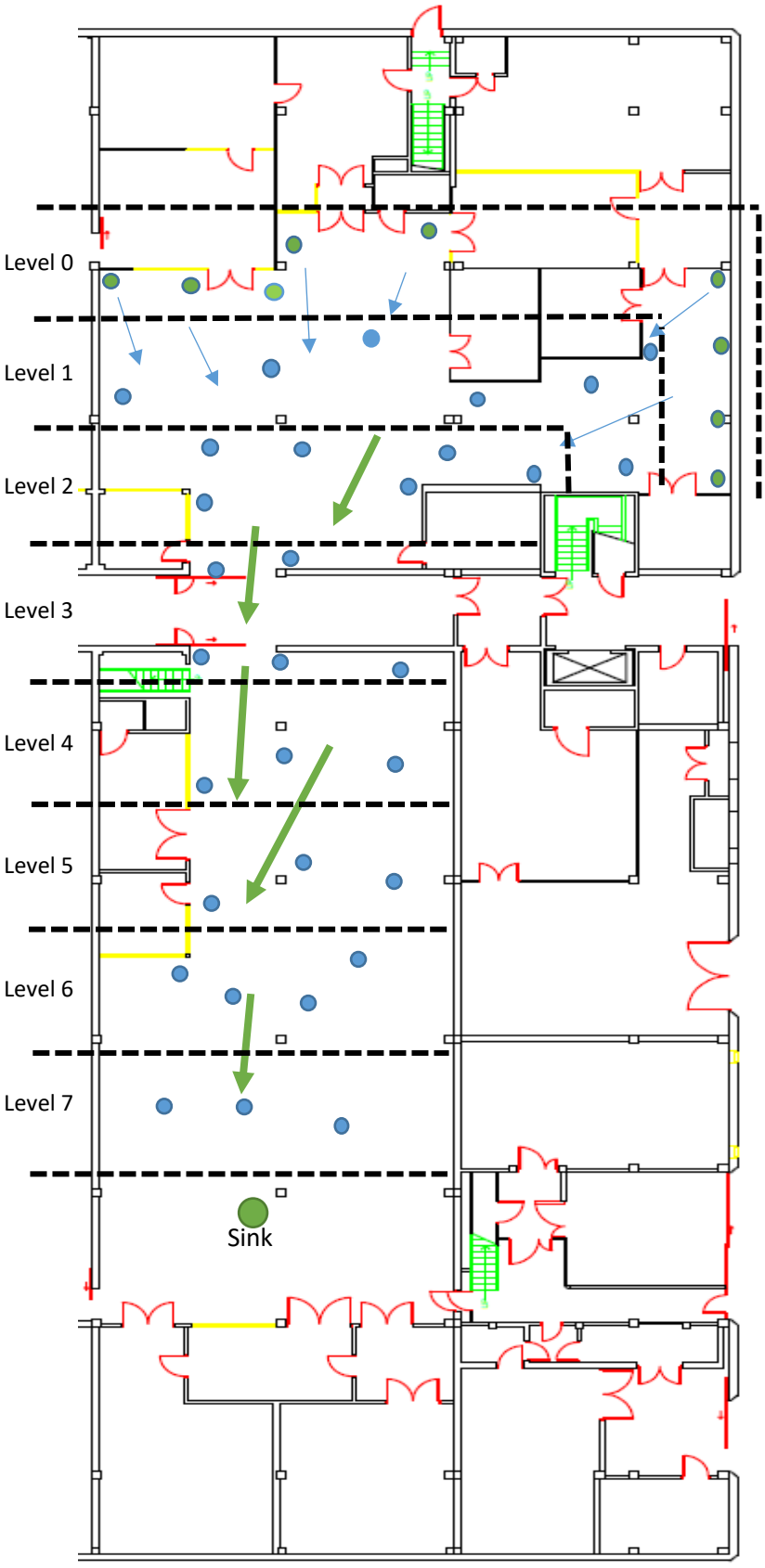


Figure 4- 6: WSN with nodes levels after assuming all farthest nodes as sources.

After level construction phase, each Waspmote node will classify its neighbour nodes into two groups (as explained in Section 3.2.6.1): a sink direction group and an anti-sink direction group. The out of the sink direction group, the node will only keep the nodes that are at a higher level. For example, Node 07 in Figure 4.7 has 6 scanned nodes (02, 03, 04, 06, 09, and 10) presented in Figure 4.5. The anti-sink direction group consists of nodes 02,03, and 04 and the sink direction group of nodes 09 and 10 which are in higher level, whereas Node 06 is removed from this group because it is at the same level as Node 07. These two groups can be read from the EEPROM of Node 07 as in Figure 4.8. The blue arrows in Figure 4.7 indicate the first part of the hierarchical tree of Node 03 in the event of it being a source. The indications of ('1st ' and '2nd ') represent the order of the nodes that the sender will send to them, so Node 03 will send firstly to Node 07, then if Node 07 congested, Node 03 will send to Node 06.

When Node 07 is a router node for data coming from nodes 02, 03, or 04, Node 07 routes the data firstly to Node 10. If Node 07 is nearly congested, it will send control message to the anti-sink direction group to change their route.

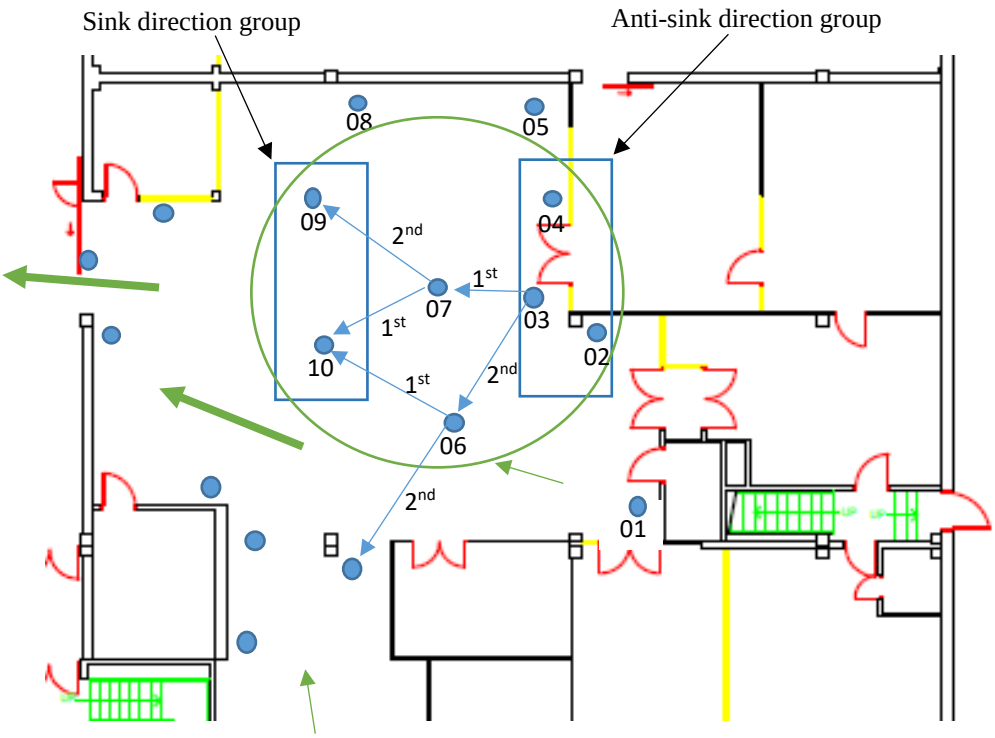


Figure 4- 7: Section of WSN with Node 07 scanning phase.

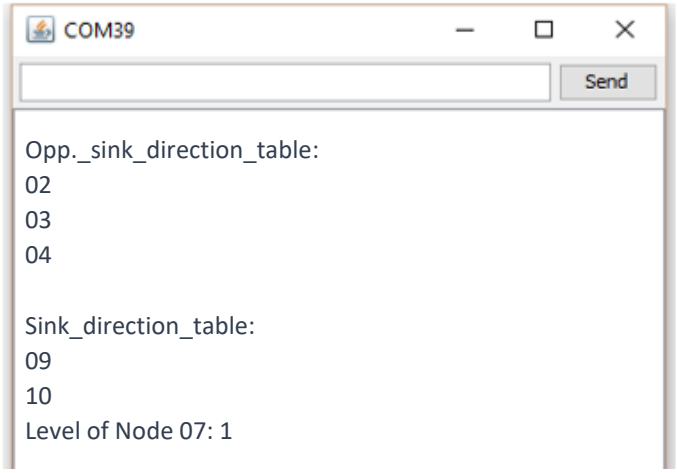


Figure 4- 8: Sink direction group and anti-sink direction group for Node 07.

4.2.3 Alternative Path Creation

In this project, any node is considered to be congested when it receives packets at a higher rate than it can transmit. When this occurs the Alternative Creation Path algorithm will start to run congestion control in current node.

To study the performance of the Alternative Path Creation scheme, the transmission rate of the node and the buffer need to be found.

The maximum number of packets per second that the nodes can transmit is clarified in next section. While the effect of the buffer node congestion will be explained in Section 4.2.3.2.

4.2.3.1 Transmission rate

The maximum packet rate per second the Waspmote node can achieve is determined by measuring the time it takes to send 100 packets (each of 100 bytes) to a gateway (the used address of the gateway is the Mac address which is 64-bit address) connected to a computer. Dividing the number of packets by the time in seconds will give the number of packets per second. By repeating this operation 100 times an accurate average calculation for the number of packets per second will be acquired. The algorithm for this measurement is presented in Figure 4.9.

step1- take Waspmote run time by using milli( ) function and store it in a variable.
step2- send 100 packets to gateway connected to a computer.
step3- take run time by using milli( ) function at the end of sending.
step4- divide the number of sent packets (100) on (the value of subtraction the first run time from last run time) and print the result packets /sec.
step5- repeat step1 to step4 for 100 times.
step6- find the average.

Figure 4- 9: Steps of finding the maximum number of transmitted packets per second.

However, this number of maximum packets per second the Waspmote node can send, might be decreased by inserting a delay time at the end of each packet transmission procedure, as in Figure 4.10. As shown in the Figure, the packet rate (when the inserted delay is 0) is 5.5 packets per second, and when inserting a delay after each packet transmission the number of packets per second is decreased as illustrated in the Figure. Inserting the delay in the Waspmote code adjusts the required Waspmote rate. For instance, the delay of 5000ms makes the Waspmote node to send in the packet rate of 0.2 packet per second, as in Figure 4.10.

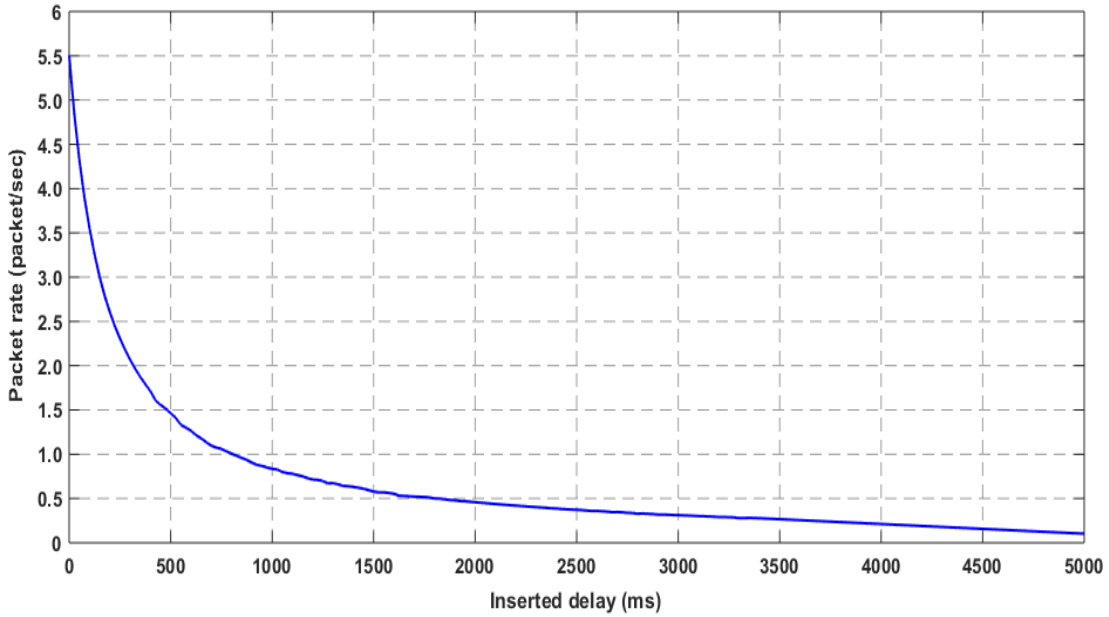


Figure 4- 10: Observed number of packets/second received by sink.

4.2.3.2 Waspmote buffer

As explained in Section 3.2.8, the working space of Waspmote RAM used for storing the generated and received data packets is the space between the heap and the stack. The fraction of occupancy of this space can be used to determine whether congestion exists. The primary free buffer space is measured at the beginning of running code and this value (stored as a global variable) will be used as a reference. The API function used for checking the free space buffer is: freeMemory( ) function, which returns the space value in bytes.

The transmission of data from events that have occurred in different places of WSN might introduce heavy traffic, and hence the buffer of each node of the transmitting path will start to be filled by the transmitted packets. When the level of occupation of a buffer in any node reaches a threshold point, the node congestion detection algorithm starts to run to mitigate the node congestion.

The congestion detection algorithm by identifying the node ID of each packet header can determine the number of nodes that send their packets through this node. In the experiments reported here the packet rate of the WSN assigned at the beginning of first run, the algorithm in each node can compare the number of received packets with the number of sent packets and if the ratio is larger than a certain percentage then the node needs to send a control message to the nodes that transmit packets through it to change their route and find alternative path for sending their packets.

The nodes that receive the control messages indicating that a particular node is congested are in a lower level than the level of congested node, so the notified nodes will search in the level of the congested node for next candidate node that can receive the sent data packets. When the congested node becomes available for receiving data packets, it informs the anti-sink direction group that is available.

The control message is represented in this project with a key letter to inform the nodes that this message is a congestion control message. Also, the control message contains the node ID of the congested node so nodes are aware about which node is congested and it is also used for analysis purposes. In Figure 4.11 a, is the congestion control message which informs other nodes about the occurred congestion in the node, and Figure 4.11 b, presents the availability control message, which informs other nodes that the current node is without congestion and it is available to receive packets.

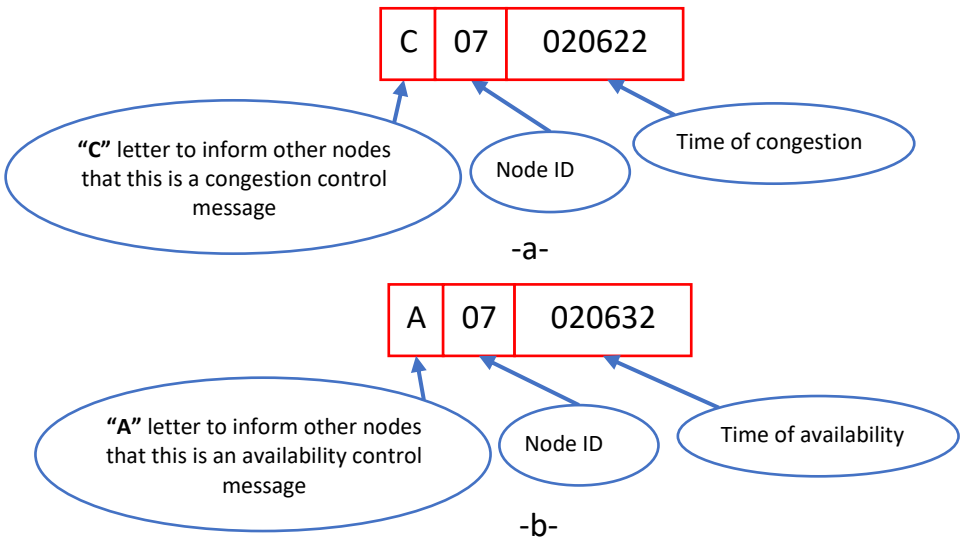


Figure 4- 11: a- Congestion control message, b- Availability control message.

The time when the congestion occurred is also included in the control message. The availability control message has the similar format to the congestion control message, (see Figure 4.11).

The control messages in Figure 4.11, are messages taken from an experiment. The congestion control message was sent from Node 07 at the time 02:06:22.

Whether the stored packets are sent depend on the application of the WSN, if every packet is required, the stored packets will be sent to allow the node to be available again. On the other hand, if the application (e.g. collecting weather temperature data) doesn't need every packet, the stored packets may be deleted and the node will be available immediately. For the example in Figure 4.11, the node became available again sending the stored packets which took about 10 sec.

Based on the first suggestion of HTAP for the congestion (the comparison between the received and transmitted packets mentioned above), sometimes, the node might receive a burst quantity of packets in a short time and then it keeps receiving packets at lower rate, in this case the node is considered to be congested in terms of occupied buffer (but this overload is temporary), therefore, the threshold with duration of receiving the burst quantity number of packets. Consequently, HTAP suggests that if the period of congestion is short, the node does not need to take any action. However, if the congestion lasts for a long period of time then control messages will be sent and alternative path found. Thus, two factors are important to determine when the node is congested: the threshold of congestion and the duration of the burst period (excess transmission).

HTAP uses the adaptive method for better control to WSN congestion, in this method, the node checks whether it is congested as follows:

- Is the buffer occupancy > 50% of the initial buffer space?
- If so, the node will count the number of nodes that have sent packets by checking the node ID in the header of received packets.
- The node will use the packet rate to check the time required for the buffer to be occupied by 75% of initial value, which is an increase of 25%.
- At the end of the required time if the occupation of the buffer is between 70% and 75%, the node will consider itself as a congested node. Then, the node will send congestion control message to let other nodes find an alternative path from their routing tables.
- If the occupation at the end of this time is less than 70%, the node will adjust the first threshold point from 50% to 60%.

For instance, in one of the experiments of WSN in this project, the size of used data packet was 100 bytes, and the transmission rate was 2 packet/sec (200 bytes/sec) for each node. The measurement of initial buffer was 4014 bytes. For Node 07 (Figure 4.7). There were 3 nodes sending packets to it, so the required time for Node 07 to be occupied from 50% to 75% (25% of buffer occupation) is calculated as follows:

t (required time) = 25% of initial buffer space / (no. of nodes x packet rate)

t = 1003 bytes / (3 x 200 bytes/sec) = 1.67 sec

This is the required time for checking that 75% of the buffer is occupied after the initial 50% of buffer occupation. So, 1.67sec after 50% occupancy, the node will check if the buffer is around 75% of occupation. If so, then the node will send a congestion control message. Then the congested node will send its stored packet until the buffer is empty, the node will send a control message for informing other nodes that it's available to receiver packets. The reason to indicate the 0% of the buffer occupation as an available node to receive packets not like 30% or 20%, is because when the route is changed in the event of congestion, the congested node should have a higher degree of confirmation that the node is uncongested which is when it reaches 0% of occupation (because less than 50% occupation might be temporary).

4.2.3.3 Accurate notification of the congestion

To control the congestion at any node, the occupation of the node buffer will be measured to check if the buffer is around 75% of the initial buffer, the node will send a

control message to nodes that sending data packets in the lower level (previous hop) to change their route by sending their data through another node.

However, in some cases, the congested node might not know all the nodes in the lower level. This problem can happen during the information collection of neighbour nodes, since sometimes, due to interference problem nodes may not find all neighbour nodes. Because of this asymmetry, some sending nodes in the previous levels will keep sending packets to congested node, since they don't receive the control message to change their route. To solve this problem, a definition message will be sent from each node to the nodes in its neighbour table to define itself to them, and any node that doesn't have the sending node in its neighbour table will add it to its table.

For instance, in the example network of Figure 4.12, there are six nodes, Nodes 1, 2, and 3 have Nodes 4 and 5 in their next level nodes table, whereas Node 4 only has Nodes 1 and 2 in its previous level nodes table.

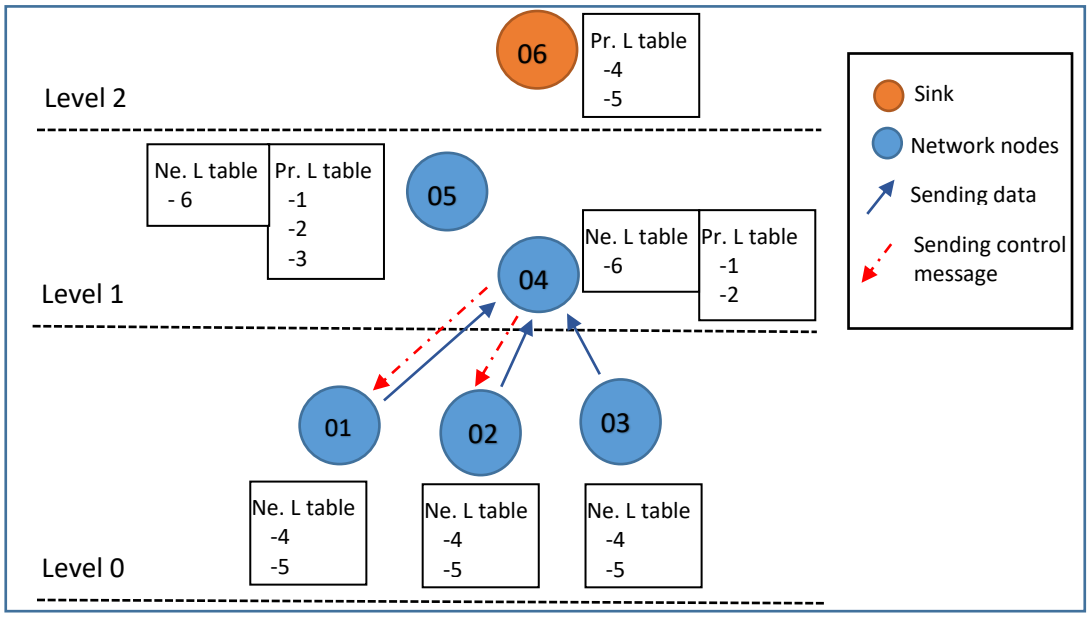


Figure 4- 12: Wireless sensor nodes with their tables for lower and next levels nodes.

In this case Node 4 doesn't know Node 3, but at the same time Node 3 knows Node 4. Therefore, when Node 4 faces congestion and starts sending a control message to its lower level nodes to change their route, it won't send a control message to Node 3. Then, Node 3 will keep sending data to Node 4, and Node 4 will not become available to receive data since it keeps receiving data packets (Figure 4.13).

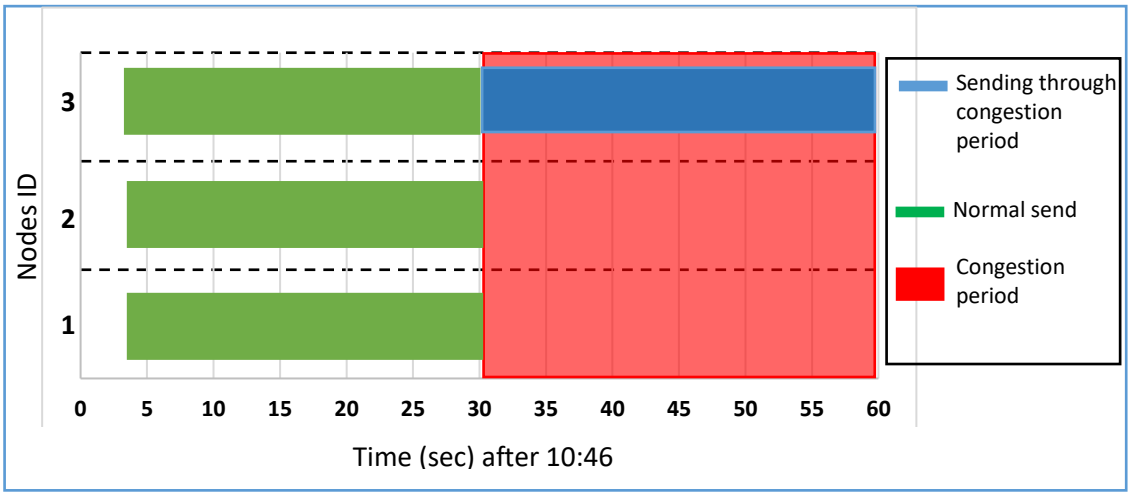
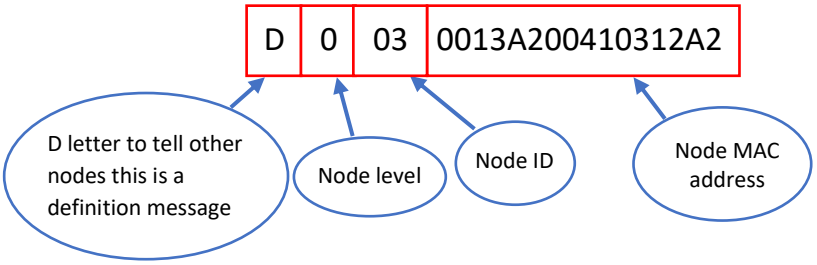
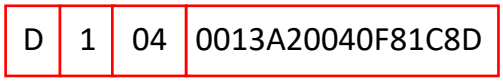


Figure 4- 13: Data received at Node 4 from nodes 1,2, and 3.

In a suggested solution to this problem, each node will send a definition message to the nodes in its next and previous nodes table, this message contains node: level, ID, and MAC address, as shown in Figure 4.14. In our example, after the information collection phase (when the node ID, MAC address, and the RSSI of each neighbour node is found) a definition phase will start, and each node will send a definition message (in unicast mode) to the nodes in its tables. After this phase, each node should know all the nodes that know it. After the definition phase, Node 4 will add Node 3 to its previous hop nodes table, then when Node 4 experiences congestion it will send control message about congestion to Nodes: 1, 2, and 3, then these three nodes will change their route to Node5.



-a-



-b-

Figure 4- 14: a- definition message construction of node 3, b- node 4 definition message construction.

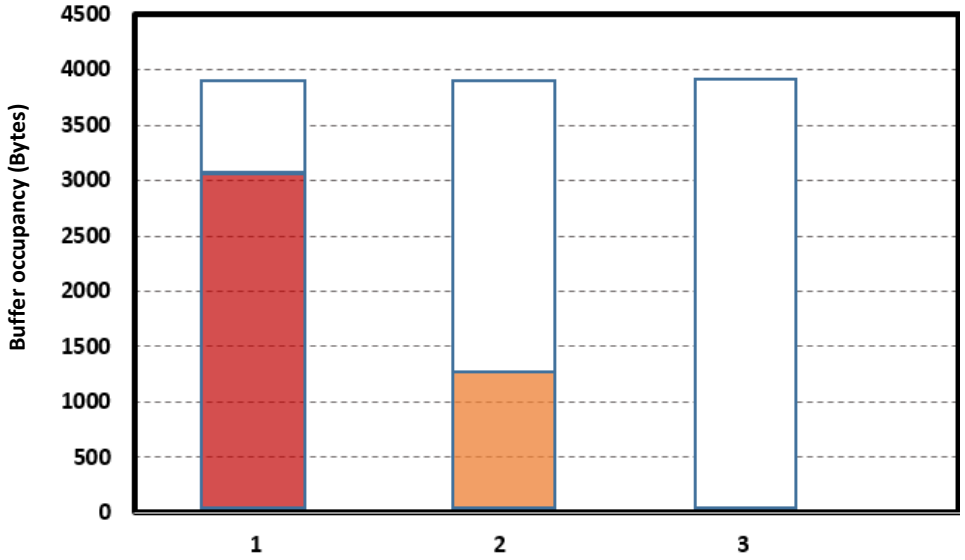


Figure 4- 15: Buffer at node 4, 1: buffer at congestion, 2: buffer before using definition phase, 3: buffer after using definition phase.

Figure 4.15, shows Node 4 buffer at: 1) congestion stage, when Node 4 buffer occupancy/level is more than 3000 bytes, 2) before using definition phase, node 4 will not reach availability phase since node 3 still sending data, and 3) after using definition phase, Node 4 sent all its stored packets and its buffer now available to receive new data packets.

4.2.4 Handling of Powerless and/or Failed Nodes

To test the last scheme of the congestion algorithm in this project, two techniques were used to handle power-exhausted node and the failure of nodes, these are:

1) Battery observation 2) Checking neighbour nodes running

For first technique, each node will measure the remaining power through an API function. (PWR.getBatteryLevel( )). The value of battery level will be used to notify the neighbour nodes about imminent battery exhaustion, the exhausted level in this project is 20% of the full charge. The battery level can also be inserted in the payload of the sent packet for analysis purposes at the sink.

The technique to detect node failure is as follows:

- Firstly, if in two consecutive attempts to communicate with it, the node doesn't send acknowledgment, then:
- The sender node will increase its power level to highest level.

- “Hello” message will be sent from sender node to receiver node and waits for the reply.
- Then if no reply the node will be considered as a failed node.
- The sender node will inform the neighbour nodes about the failure of the node, and the neighbour nodes will remove the failed node from their tables.

4.3 Performance of congestion algorithm

In this section, the performance of HTAP which controls the congestion by employing a resource control algorithm, is evaluated experimentally by deploying the nodes in testbed in the Engineering labs. The results from these experiments were compared with the results of running the WSN in the same testbed but without employing congestion control.

The congestion control code which was uploaded to the Waspmote node is illustrated in Figure 4.16. The algorithm without congestion control is illustrated with the red dotted arrow in the flowchart.

Forty Waspmote nodes were deployed randomly in the Engineering labs. The deployed WSN has 1 sink at the end of the lab as in Figure 4.6, and the first line of the deployed nodes were sources.

Before the experiment, the coverage of each Waspmote node was tested to confirm that had full connectivity with its neighbour nodes.

At the beginning of each experiment, a fixed generated packet rate (as network load) was assigned to each source node. The used packet rate ranged from 0.2 Packet/sec to 5.5 packets/sec, while the size of data packet in these experiments was 100bytes. The WSN was run for 2-3 hrs to obtain the results of resource control algorithm in each experiment. Twelve data rates (network load) were selected for the experiments of testing HTAP algorithm. These data rates were: 0.2, 0.4, 0.8, 1, 1.3, 1.7, 2, 2.6, 3.5, 4.6, 5, and 5.5 packets/sec. Each experiment had two sessions, one for running the network with HTAP algorithm and the other for running the network without congestion control. Running the network without congestion control is by notifying all Waspmote nodes to bypass the congestion control part of the code, as illustrated in Figure 4.16.

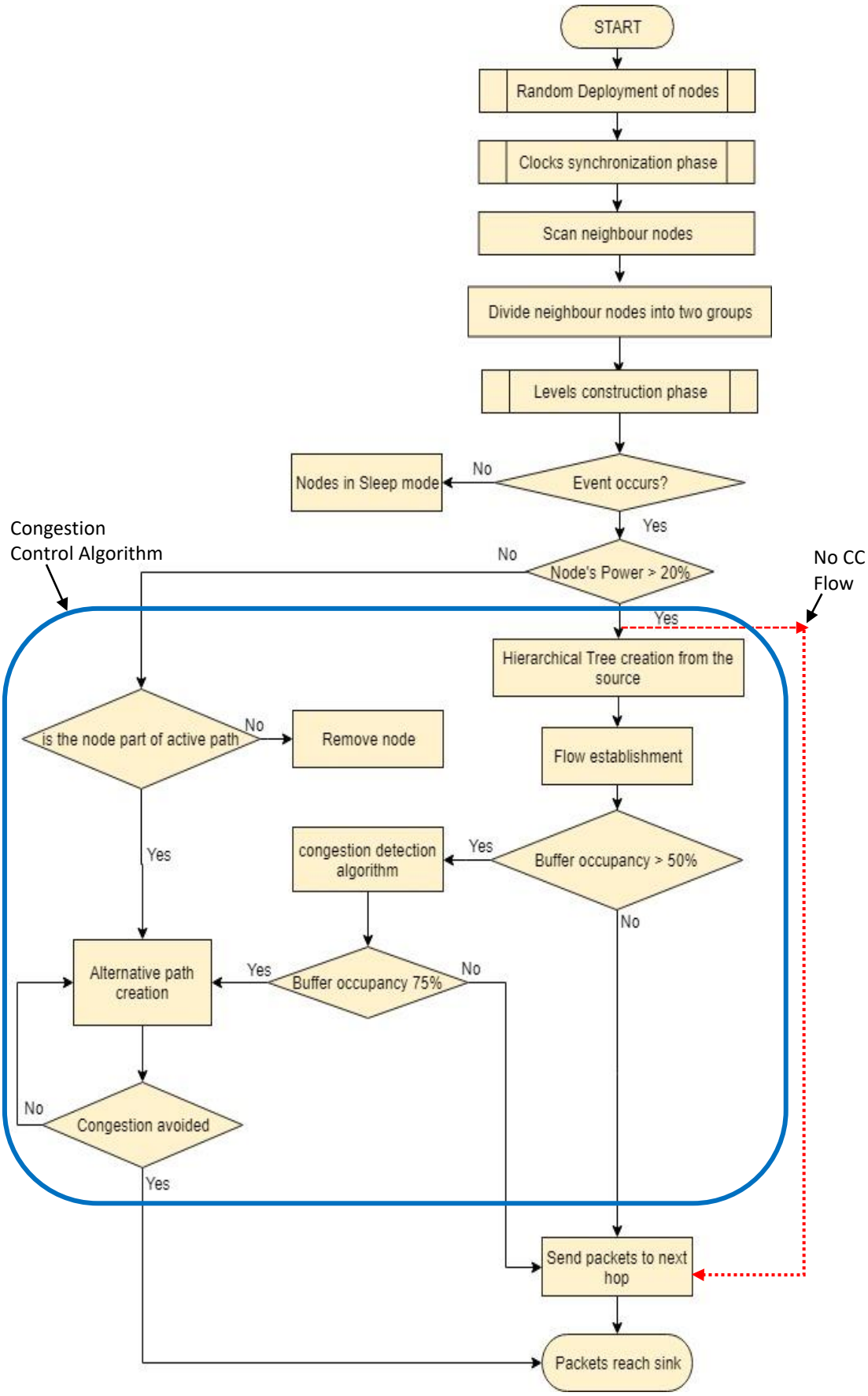


Figure 4- 16: Flowchart of congestion control algorithm, no congestion control flow is illustrated by red dotted arrow.

Several metrics were used for determining the network performance, these being packet delivery ratio (section 4.3.1), throughput (section 4.3.2), and end-to-end delay (section 4.3.3).

4.3.1 Packet Delivery Ratio (PDR)

A widely used metric in evaluating the performance of WSNs protocols is PDR. The PDR is defined as the ratio of the total number of data packets successfully received at the sink to the number of data packets sent from the sources.

$$Packet\ Delivery\ Ratio = \frac{Total\ Packets\ received\ by\ the\ sink}{Total\ Packets\ sent\ from\ source\ nodes} \times 100$$

In the experiments, the duplicated packets (same node ID and same generated time) will be removed from the calculation of the PDR. The duplication might occur because of interference, since in some cases this can prevent the acknowledgments from being received by the sender nodes. Which will then send the packet again.

For critical applications, ideally, every packet should be delivered to the sink. Obviously, higher values of PDR are better because it means a large number of packets is delivered to the sink in comparison to lost packets. The collected data from the experiments were analysed and the PDR as a function of packet rate of HTAP algorithm and algorithm without congestion control is presented in Figure 4.17.

It can be seen from Figure 4.17, that the effect of increasing the packet rate will decrease the PDR value of HTAP algorithm gradually from 93.9% at 0.2 packet/sec to the value of 69% at 5.5 packets/sec of the network load.

On the other hand, a significant decrease from PDR value of 70 % at a packet rate of 1 packet/sec to 5% at the packet rate of 5.5 packets/sec occurs for without congestion control. The increments in the data rate of the network load has the following effects: increasing the likelihood of packets collisions in the wireless link, increasing the channel contention between the sender nodes, and the faster filling for node buffers. Therefore, these effects will increase the number of dropped packets and hence the lower PDR will be resulted.

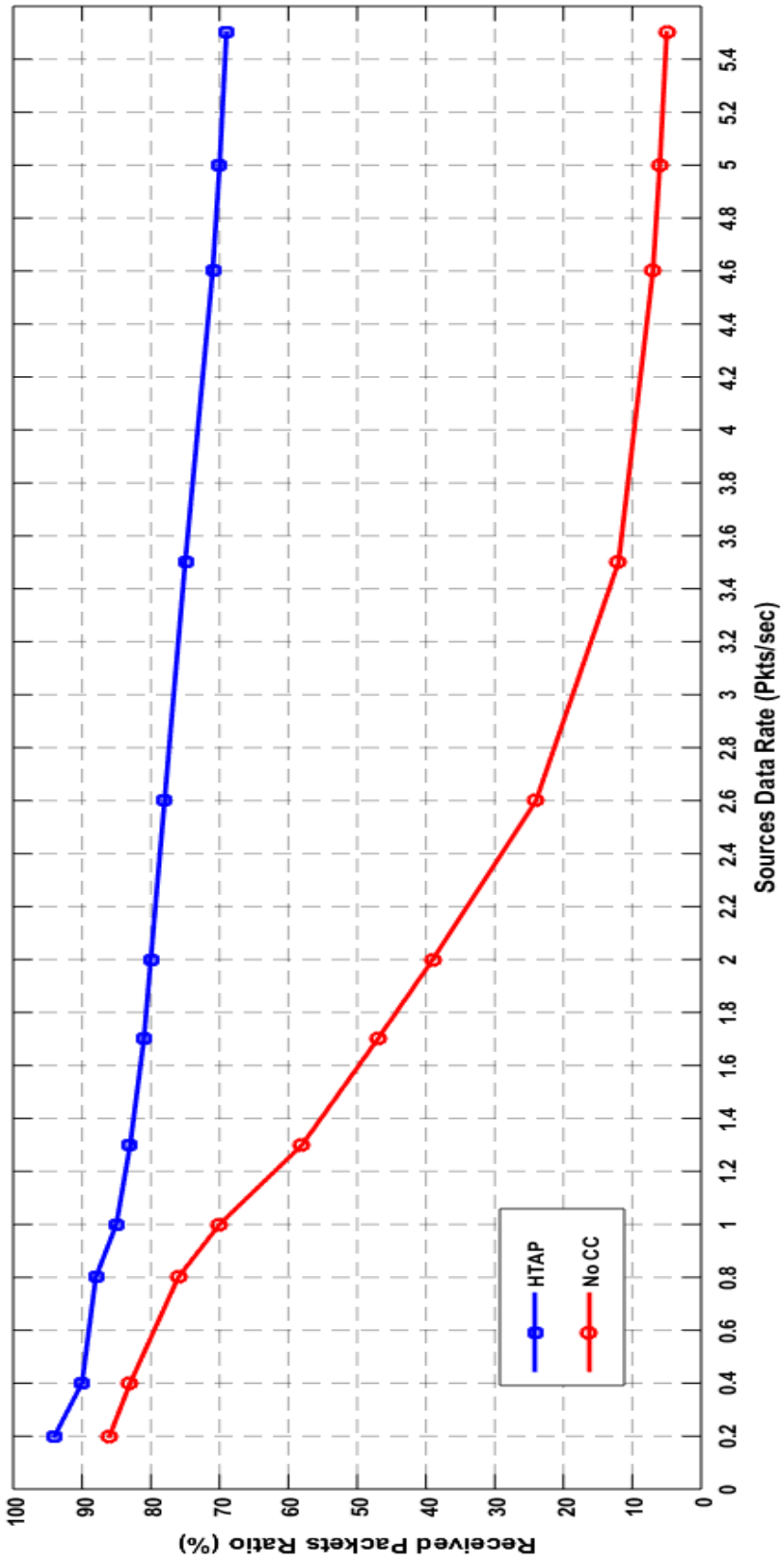


Figure 4- 17: PDR as a function of source data rate for the HTAP algorithm and the algorithm without congestion control.

Additionally, for the network of no congestion control, when the buffer of Waspmote node is filling due to high data rate the RAM of Waspmote node might be affected by the memory leak (as explained in previous chapter) then the Waspmote node needs a reboot. The reboot operation will increase the number of dropped packets.

There is some variation in the measured PDR as a function of time. Figure 4.18 to Figure 4.22 display graphs of PDR as a function of time with the mean value of the PDR, for source data rate of 0.2, 1, 2.6, 3.5, and 5.5 packets/sec.

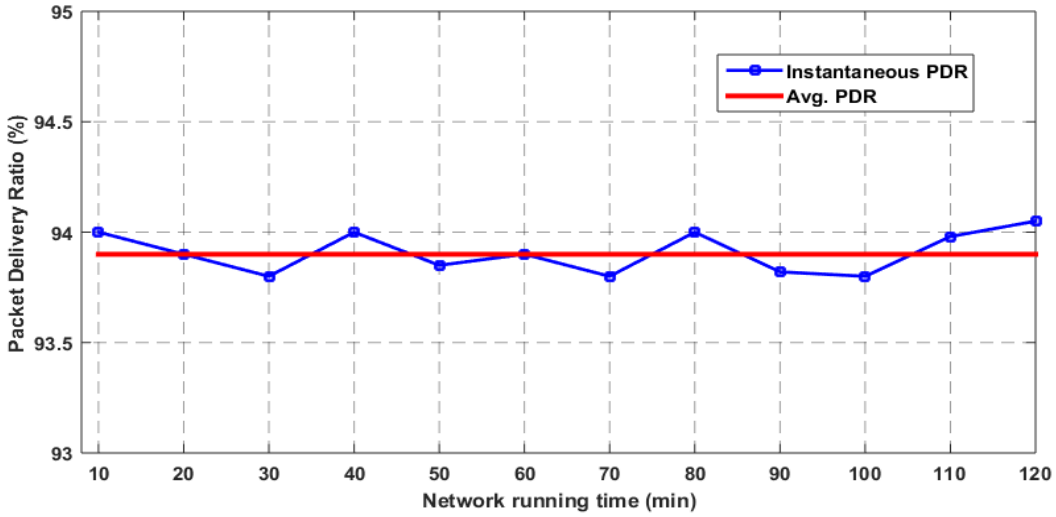


Figure 4- 18: PDR as a function of time over the period of first 10min to 120min of network running time, the network load is 0.2 packets/sec.

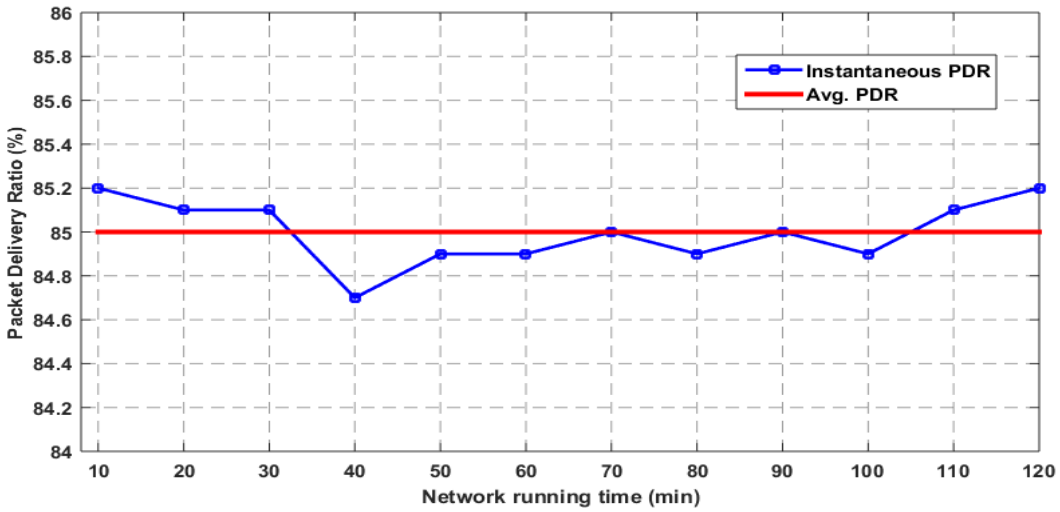


Figure 4- 19: PDR as a function of time over the period of first 10min to 120min of network running time, the network load is 1 packets/sec.

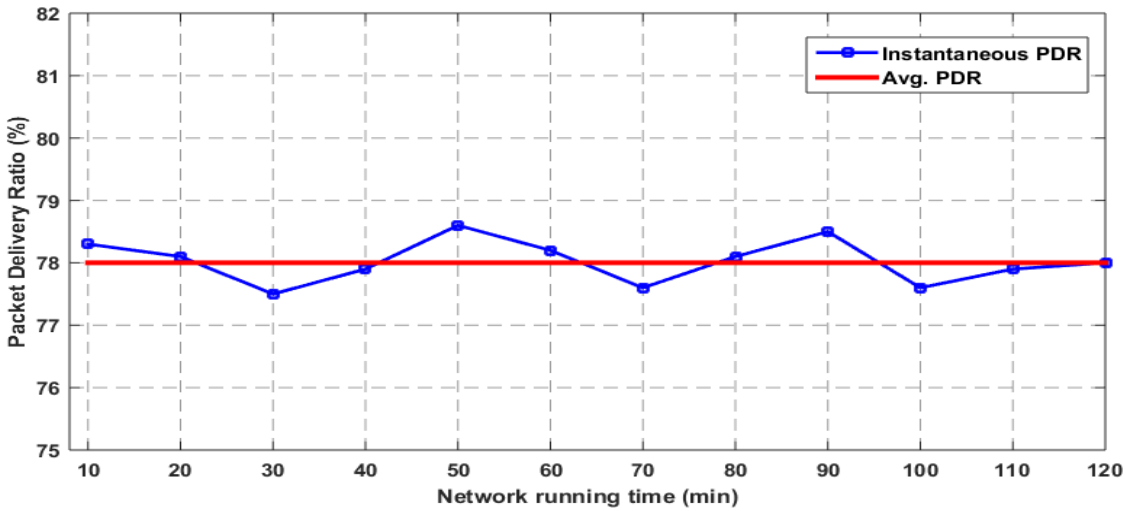


Figure 4- 20: PDR as a function of time over the period of first 10min to 120min of network running time, the network load is 2.6 packets/sec.

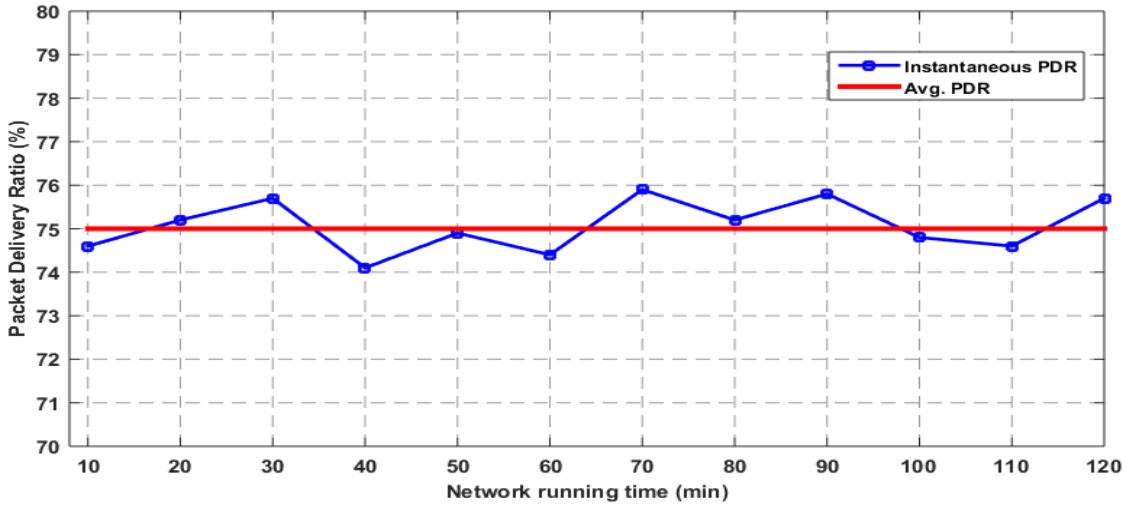


Figure 4- 21: PDR as a function of time over the period of first 10min to 120min of network running time, the network load is 3.5 packets/sec.

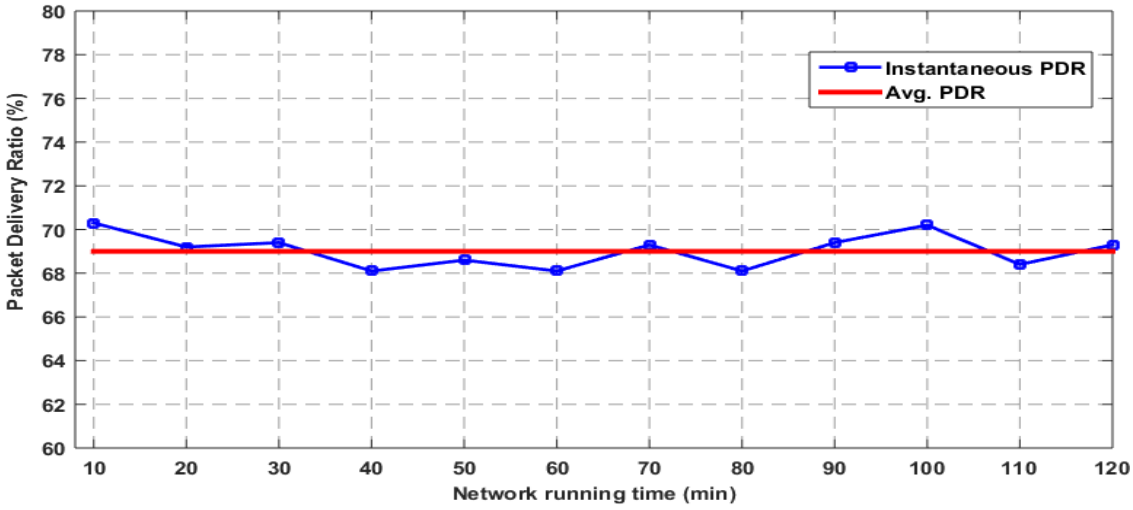


Figure 4- 22: PDR as a function of time over the period of first 10min to 120min of network running time, the network load is 5.5 packets/sec.

Table 4. 2: Statistics of the PDR results values for the period of 2hrs from the first 10 min to the end of 120 min.

Data rate (packets/sec)	Avg. PDR	SD
0.2	93.9	0.1
1	85	0.2
2.6	78	0.3
3.5	75	0.6
5.5	69	0.8

It can be seen from the above figures, that with the variation of PDR, the Standard Deviation (SD) is not a large value. However, the SD of the observed PDR increases as the data rate increases (Table 4.2). The higher SD for the PDR of high data rate network is because the network under heavy load, and hence the number of dropped packets is increased as mentioned above in this section. As a result, the instantaneous PDR depends on different values of dropped packets, the number of dropped packets might be large at sometimes, and small number at another time.

For the simulated HTAP (Sergiou *et al*., 2013), the average value of PDR at data rate of 72 packets/sec is 100%. However, for the experimental verification of HTAP with Waspmote node doesn't achieve average value of 100% for PDR, even with very low data rate (0.2 packet/sec), this is because the simulated HTAP assumed a large buffer size of 512kbytes which is very large compared to the buffer size of 8kbytes, then the node takes long time to be congested, and hence better PDR will be achieved.

4.3.1.1 Drop-Burst Length

The HTAP algorithm results analysed to find the values of DBL under three different packets rate (1.7, 3.5, and 5.5 packets/sec), see Figure 4.23. We can see that under 1.7 packets the probability of DBL is much lower than other higher data rate, particularly for the 1 packet DBL its probability is 0.088, but for other tested data rates (3.5, and 5.5 packets/sec) is 0.144 and 0.19 respectively. Also, from these results, the behaviour of DBL under 1.7 packets/sec data rate is lower than the other data rates, which gives an indication that HTAP can recover the broken route under low data rate better than higher data rate.

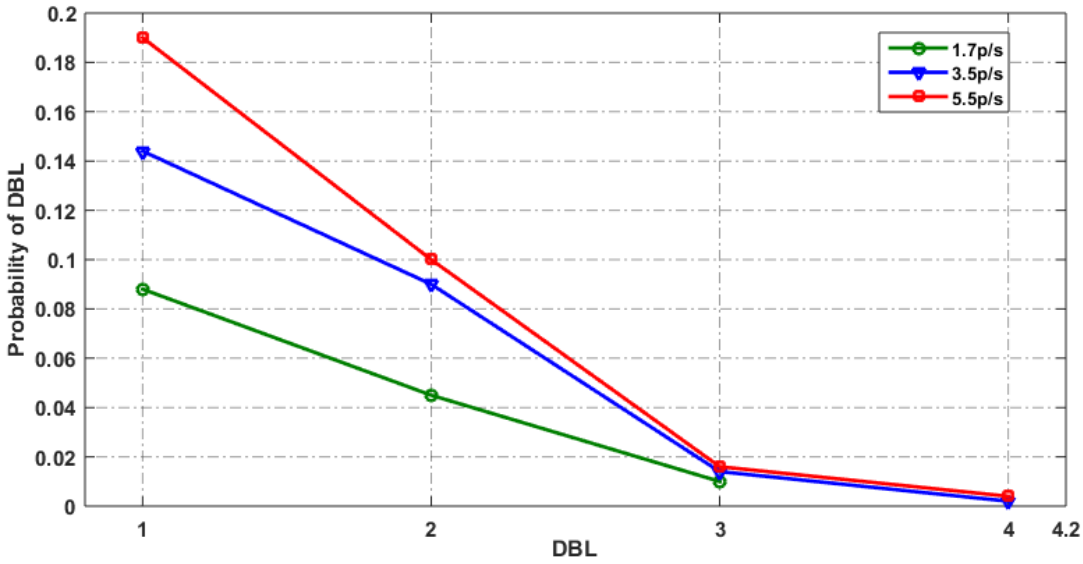


Figure 4- 23: Short Drop Burst of HTAP under different data rates.

4.3.2 Network Throughput

Network throughput has also been used to measure how well the HTAP algorithm control the congestion. This is defined as how much amount of data is sent or received during a defined period of time, the throughput is expressed in packets/sec or kbps... *etc* (Alazzawi & Elkateeb, 2008; Piyare *et al*., 2013).

$$Network\ throughput = \frac{no.of\ received\ packets\ by\ the\ sink}{network\ running\ time\ (\text{sec})}$$

The packets collected at the sink will be calculated during the 2hour period the network was operated. In Figure 4.24 the network throughput is presented as a function of source data rate.

At low data rates (< 0.4 packets/sec) both the HTAP and no congestion control have similar performance and are close to ideal case. For data rates greater than 0.4 packet/sec the difference starts to enlarge, until at a data rate of 2 packets/sec starts the performance of no congestion control becomes very poor. For HTAP, the throughput continues to increase as the source rate increase, but the deviation from the ideal case does increase.

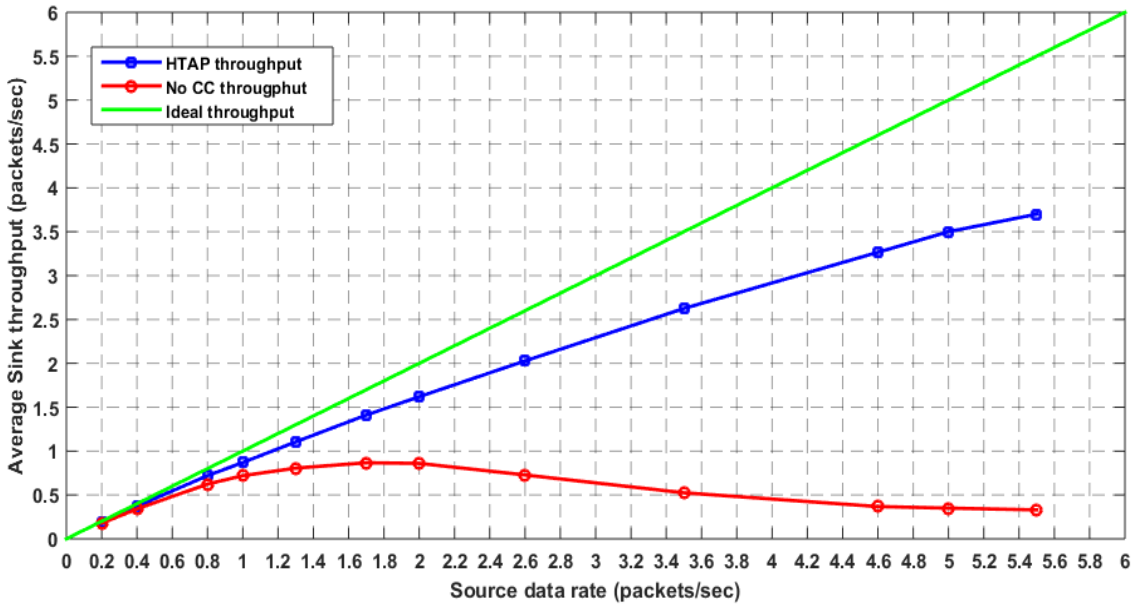


Figure 4- 24: Average Sink throughput as a function of source data rate for HTAP, no congestion control, and ideal conditions.

The performance of HTAP has a throughput value of 3.7 packets/sec at the source data rate of 5.5 packets/sec, this value gives a difference of 3.37 from the value of no congestion control at the same data rate. From this difference and at the high data rate, it's clear that the resource control algorithm enhances the performance of the network by making the throughput value closer to the ideal throughput response.

To study the behaviour of network throughput with different data rates, 3 values were chosen to study the behaviour of network throughput as a function of time, these values of data rates are: 0.2, 2, and 5.5 packets/sec. Figure 4.25 to Figure 4.27 show the sink throughput as a function of time over the period 10 min to 120 min.

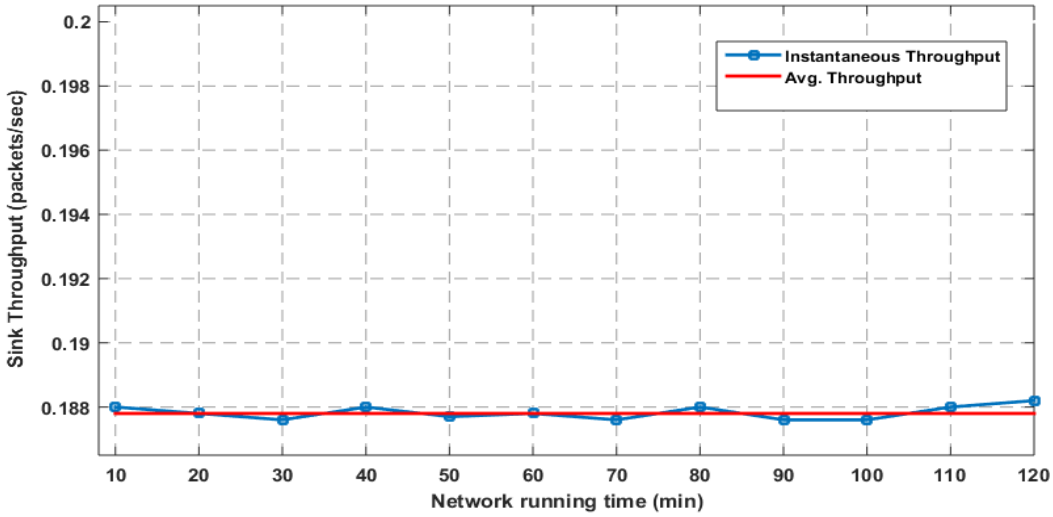


Figure 4- 25: Sink throughput with source data rate of 0.2 packet/sec.

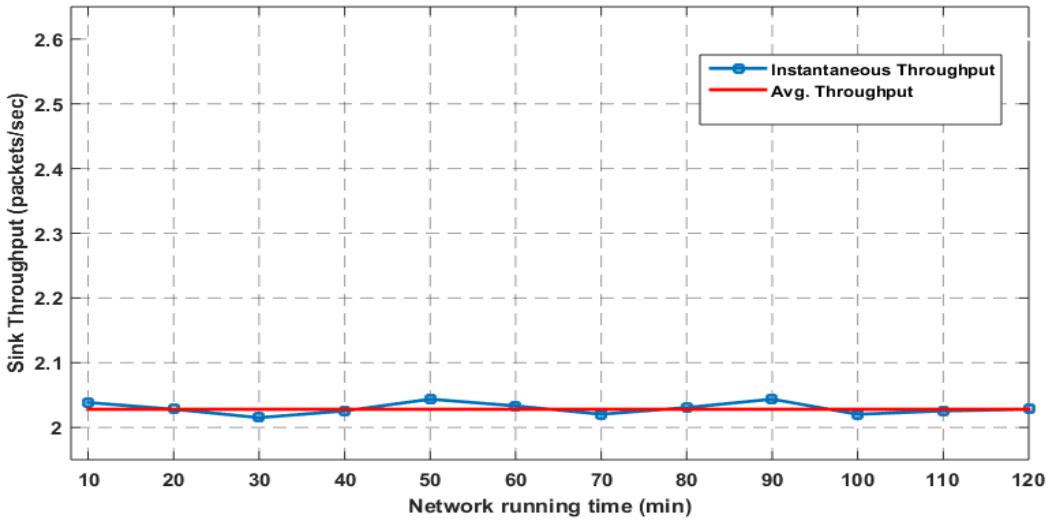


Figure 4- 26: Sink throughput with source data rate of 2.6 packets/sec.

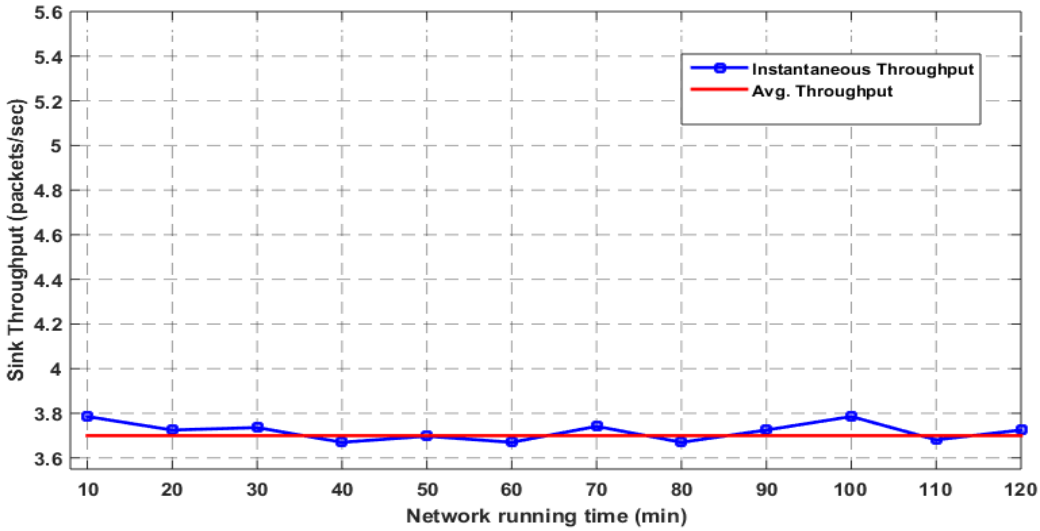


Figure 4- 27: Sink throughput with source data rate of 5.5 packets/sec.

Table 4. 3: Sink throughput of 3 different data rates.

Source data rate (packets/sec)	Avg. Throughput	SD
0.2	0.189	0.005
2.6	2.03	0.01
5.5	3.7	0.04

Table 4.3 shows the statistics of the 3 previous figures. Also, in the table the increase in the data rate yields to more difference between the average throughput and the ideal difference, also with considerable standard deviation. The standard deviation of throughput at low data rate is very low compared to high data rate, and this is a normal result because at low data rate the network is more stable, since there is a small chance of collision as well as low channel contention.

4.3.3 Average End to End delay

Average delay a third significant metric for testing whether a resource control algorithm has a strong ability to control the network congestion. The delay is defined as the time taken by the packet to be sent from the source until the entire packet is received by the sink.

Since in this experiment the Waspmote network is synchronized at the beginning of network running, a timestamp was inserted in each generated packet. The timestamp insertion is by using millis( ) function. The packet can then be computed at the sink by subtracting the source time stamp from the sink timestamp. The collected time delays at the sink will be averaged to find the End to End delay. The packet delay to the HTAP algorithm and with no congestion control are presented in Figure 4.28 as a function of source data rate. At the low data rates, the behaviour of HTAP is nearly the same as the no congestion control mechanism, but as the data rates increases the End to End delay of the no congestion control increases at about twice the rate as for HTAP, e.g. at 5.5 packet/sec the delay is 15 secs for the no congestion control and 8 secs for HTAP

The reason for the large value of delay with no congestion control is because that the node buffers will suffer from memory leak under congestion and need to be rebooted.

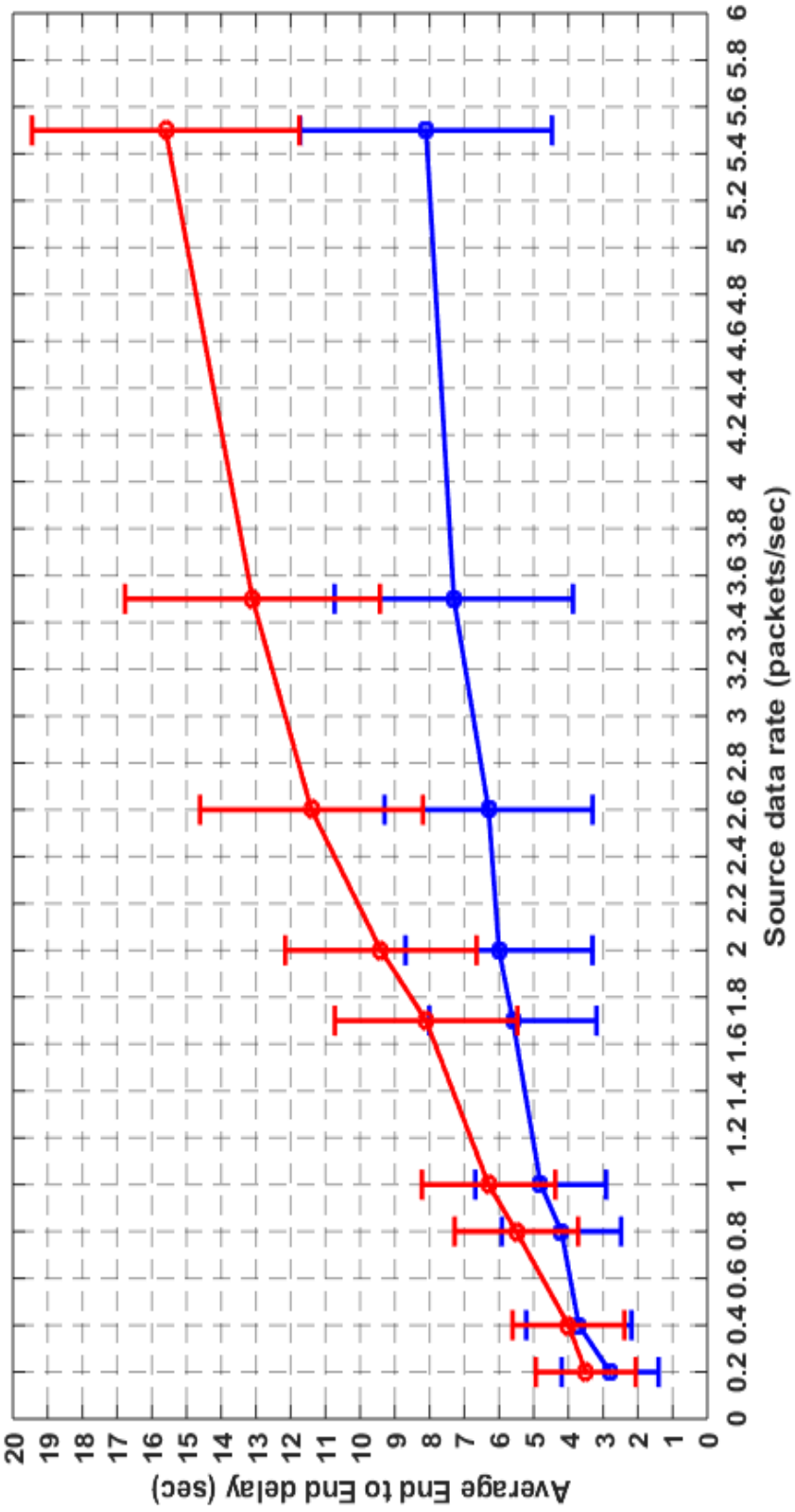


Figure 4- 28: Average End to End delay of HTAP algorithm and with no congestion control.

The large presented SD in Figure 4.28, can be clarified that some of transmitted packets might be delayed in intermediate nodes because of the occurred congestion, also these delayed packets probably to get through congested node many times during their journey from the source to the destination (Sink). On the other hand, some of transmitted packets arrive the sink in short journey compared to other delayed packets. The network with this kind of large SD might be imagined in the critical application which needs every single generated packet.

To clarify the behaviour of the data delay of Figure 4.28, 3 different data rates were chosen to view the end to end delay in Figure 4.29 to Figure 4.31, these rates 1, 1.7, and 5.5 packets/s.

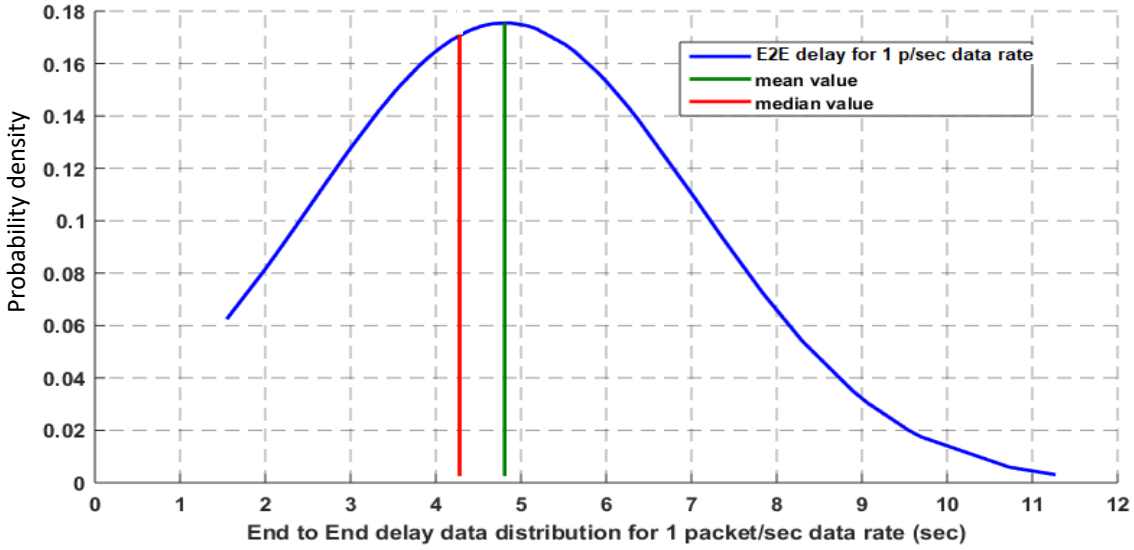


Figure 4- 29: Data of End to End delay distribution for the data rate of 1 packet/s.

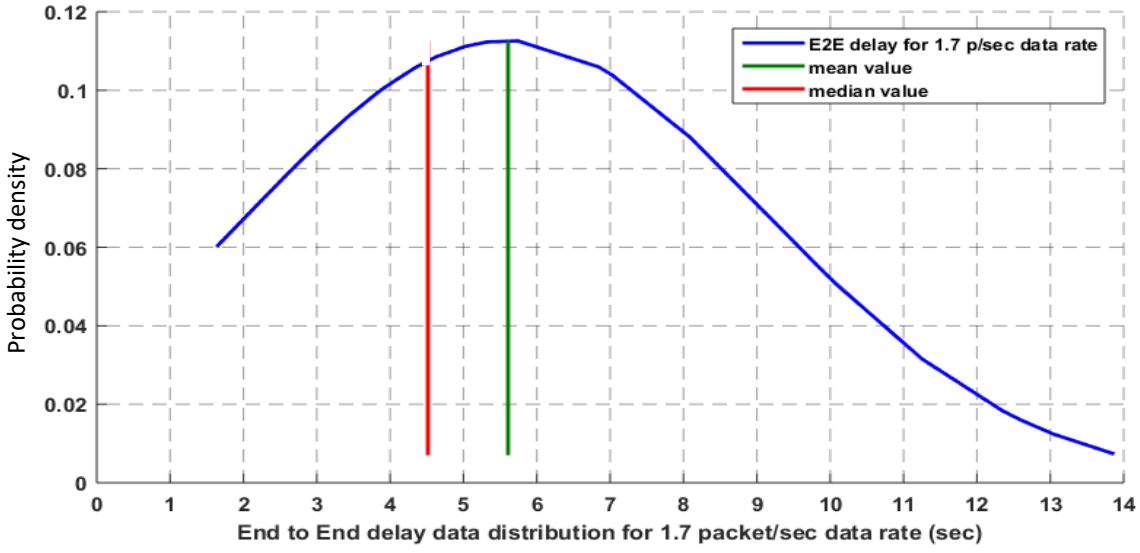


Figure 4- 30: Data of End to End delay distribution for the data rate of 1.7 packet/s.

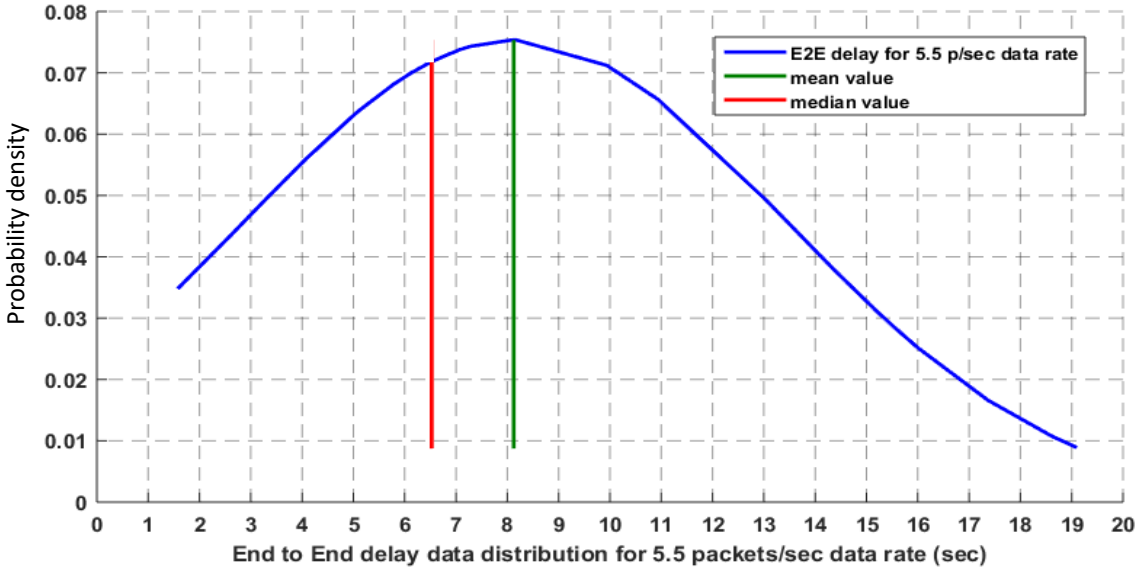


Figure 4- 31: Data of End to End delay distribution for the data rate of 5.5 packets/s.

The graphs show that each distribution is positively skewed and the median is smaller than the mean in all presented distribution. The mean value of End to End delay increase proportional to the increase in the data rate.

Table 4. 4: Mean, median, and standard deviation of the End to End delay for 3 different data rates.

Source data rate (packets/sec)	Mean value	Median value	SD
1	4.813	4.28	2.272
1.7	5.601	4.514	3.541
5.5	8.142	6.526	5.318

As explained previously in this section, that due to the delay of some packets because of the congestion, the average delay is increased to large values, these delays are known outliers. These outliers increase the mean value of each data rate. Therefore, all the distribution of above figures is skewed to the right.

The large average value of the End to End delay of HTAP algorithm can be interpreted as follows:

- In the event of congestion and as mentioned in Section 4.2.3.2, after sending the control message about the congestion the congested node will send the stored packets in its buffer using the same route to the sink.

- Sending or dropping the stored packets after the congestion depends on the type of WSN application. Critical WSN application, as considered here, needs every single packet to be received by the sink, whereas in some applications receiving some data is enough for the environment application (such as temperature, humidity), so in this kind of applications the stored packets can be dropped.
- Each data packet has timestamps indicating when it was sent from the source and received by the sink. From the data analysis, it's observed that some packets have large difference between the transmitting and receiving timestamps. This large difference for some packets arises because these packets queue up in the node buffer for long time because of the node congestion. This situation might occur to the same packets in a second or third congested node resulting in more End to End delay. This large quantity of the delay time will increase the overall average of the End to End delay in the network.

The difference in performance between sending the stored packets or dropping the stored packets for HTAP is presented, Figure 4.32.

The delay after dropping the stored packets decreased at the data rate of 5.5 packets/sec from 8 secs to 5 secs. However, of course dropping the stored packets has a negative effect on other network performance such as the network throughput, therefore this technique of dropping stored packets is a trade-off between the End to End delay and the network throughput.

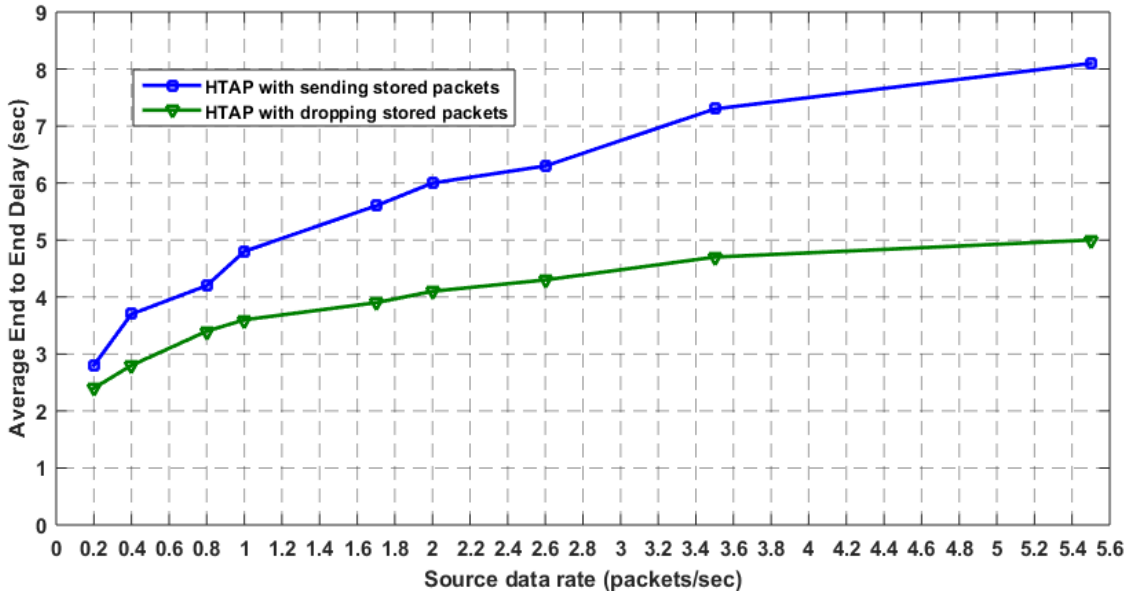


Figure 4- 32: End to End delay for HTAP algorithm with sending or dropping the stored packets after congestion.

In a wider scale WSN, the end to end delay affected by the increase in the number of nodes in WSN, the increase either in same deployment area or in a larger area. So, for increase the number of nodes in same area, the end to end delay might be decreased because the transmitted packet can find more than alternative path at the event of congestion. On the other hand, the increased in the number of nodes with the increase in the deployment area might increase the end to end delay, because of the increase in the number of required hops for the transmitted packet to reach the sink.

The link-level congestion in the experiments of this research is affected negatively at higher data rate, this is because the increase of data rate will increase the number of existed packets in the link with in turn increases the packet collision.

The network designer must choose a packet rate that delivers the required data as fast as possible while avoiding the extra packet loss and delay caused by too high a packet rate. Therefore, for a packet rate where data completion is important (e.g. investigating pipe leaks) a low data rate is best. However, for applications where is less important to receive all the transmitted packets (such as collecting a temperature on a farm), the data can be sent with a higher data rate allowing dropped packets in the event of congestion.

4.4 HTAP algorithm overhead estimation.

The HTAP algorithm overhead can be estimated into two different categories which appear during the network running. The first category, the HTAP uses metadata, network routing information and the congestion control message sent by an application, these types of information are using a portion of the available bandwidth of HTAP algorithm. The second category is the other probable HTAP overhead is the required time during the HTAP setup and the HTAP mechanism of sending the stored packets. The previous two categories of HTAP overhead estimation can be illustrating in the following detail:

- *First category*: in the information overhead category, HTAP algorithm adds in the frame header the following information:
    1. The time and day of generated packets are represented in the frame header by 11 bytes in the frame header.
    2. Two bytes represent the node ID in the frame header.
    3. Three bytes represent the packet sequence in the frame header.
    4. The packet adds 2 bytes for each node (in a hop) it transmitted through it, so, for network of hops from source to sink the packets will take 10 bytes from

the frame payload. So, 26 bytes are used by HTAP as packet and routing information.

5. Nine bytes are used by congestion control message, which is sharing the available bandwidth.

- *Second category*: the spent time by HTAP to set the network, which can be illustrated:

1. The neighbour scanning phase required 20 seconds during the network setup phase.
2. Assigning each node to a level is around 10 seconds.
3. Dividing the neighbour nodes to two groups, required around 1 second. So, the required time for HTAP during setup phase is around 32 seconds.

4.5 Summary

The performance of HTAP algorithm is evaluated experimentally by deploying 40 Waspmote nodes in the Engineering lab. Three metrics of networks performance were checked: PDR, throughput, and End to End delay. The performance of HTAP is compared with same deployed nodes but without congestion control algorithm. Four schemes were tested to evaluate the performance of HTAP, these schemes are: Topology control, Hierarchical Tree Creation, Alternative Path, and Handling of Powerless and/or Failed Nodes. The HTAP algorithm used in its topology control the Local Minimum Spanning Tree (LMST).

From figure 4.17, the improvement of occurred to the value of PDR of HTAP compared to the value of PDR of NCC is presented clearly, the ratio of PDR of HTAP to the PDR of NCC is increasing as the data rate become higher, it can be seen that at lower data rate (1 packet/sec) the PDR value for both (HTAP and NCC) is close to each other, whereas at higher data rate (5.5 packet/sec) the ratio increased to become around 13 times for the value of PDR of HTAP compare to PDR of NCC.

Additionally, in Figure 4.27, the ratio of end to end delay of HTAP to the value of end to end delay of NCC is nearly the same at lower data rates, but this ratio is decreased at higher data rate which gives an indication that HTAP improves the performance of the network compared to the running of the network without congestion control.

Chapter 5: Factors affecting network congestion

5.1 Introduction

In this chapter, important factors related to network congestion are investigated experimentally. Firstly, the adaptive method of congestion threshold is introduced and compared with different congestion threshold values. Then, the effect of the number of nodes is presented. As explained in Chapter 1, a WSN has three components: source (sensor) nodes, forwarding (router) nodes, and sink (base station). The number of nodes in the experimental network will be fixed to 40 nodes while studying the effect of changing the number of sources, routes, and hops. In other experiments, the number of nodes will be changed while keeping fixed number of sources fixed. Finally, the effect of packet size on network congestion is measured. Also, the essential principles of 802.15.4 packet configuration are introduced.

5.2 Effect of using different congestion threshold values

One of the main restrictions of the sensor node is its small memory size, which is considered as the main cause of node-level congestion type of network congestion (Ghaffari, 2015; Kafi *et al*., 2014). A sensor node that receives more data packets than it can forward, buffers the excess data packets. Therefore, due to limited buffer space node-level congestion will happen, and because of the buffer overflow the data packets will be dropped from the buffer.

In section 4.2.3.2, the mechanism of congestion detection in HTAP was explained, and in this section the impact of employing different congestion buffer size threshold values on the network performance will be investigated.

The Waspmote SRAM size is 8kbytes (8192bytes), while the buffer size that can be used for storing the received packets is the free memory space between the heap and the stack, as explained in Section 3.2.8. The HTAP algorithm is tested with buffer size congestion threshold values of: 25%, 50%, 75% of the buffer, and on adaptive method. The used packet size was 100 bytes and source data rates of 1.7 and 5.5 packets/sec used.

At the first running of the uploaded code, the buffer space will be measured by using freeMemory( ) function. For the adaptive method a 50% of the measured buffer size will

be set as the first congestion threshold with following steps illustrated in Figure 4.16. For the fixed threshold values the adaptive method will be ignored in the code.

Figure 5.1 shows the measured packet delivery ratio (PDR) at the two data rates. In this figure, the value of PDR of each data rate is decreased with increasing congestion threshold value. Also, at each threshold value the value of PDR is decreased with an increase in the data rate.

The improved performance at lower threshold is when the nodes trigger the alternative path early the network avoids congestion. The drop in performance with increase of packet rate occurs when the received packets exceed the network capacity causing the packets to be lost.

The PDR value of the adaptive method is better than the fixed threshold values, because the control message selecting an alternative path will not be sent immediately after the first threshold congestion, but will be sent after the satisfaction of second threshold value.

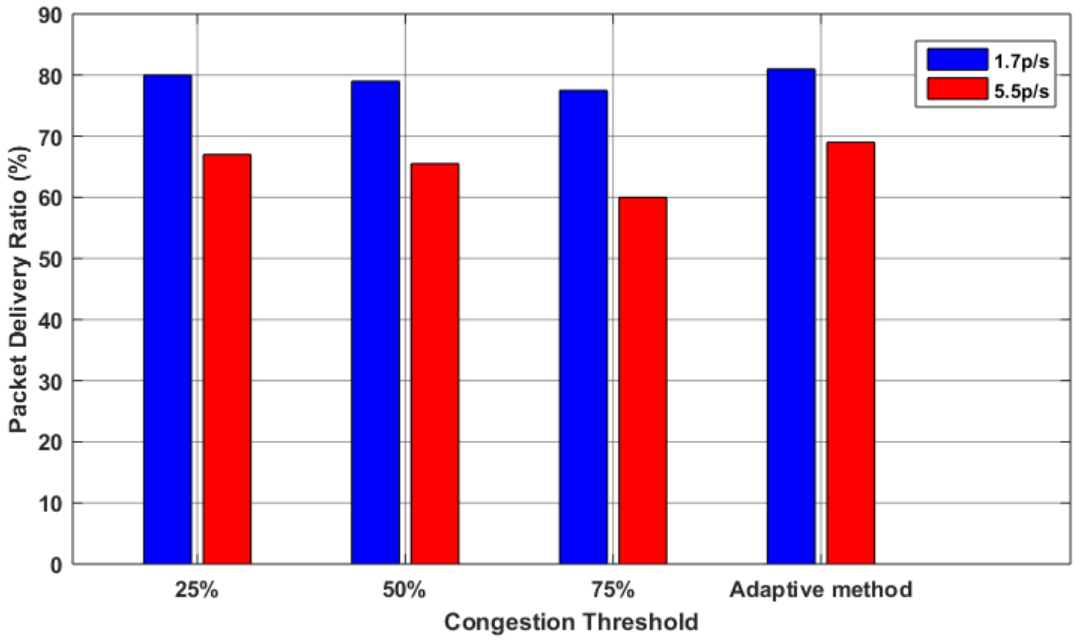


Figure 5- 1: Packet delivery ratio of HTAP algorithm with different congestion threshold values.

5.3 Effect of the number of nodes and sources in WSN

Increasing the number of nodes in WSN is another factors that can lead to network congestion (Alduais *et al.*, 2016). The probability of finding neighbour nodes increases for any node as the number of nodes increases in the network. In consequence, the number

of routes will be increased proportionally. Another effect on the network congestion is studied in this section, this effect is increasing the number of sources.

5.3.1 Possible routes for nodes

To find out the number of possible routes for each node to send its data through them, a simple WSN consists of 1 source node positioned in level 0 (as starting point for sending), then next level (Level 1 which is the 1st hop) consists of 6 nodes, and the last level 2 is a gateway connected to laptop, as in Figure 5.2.

In this experiment, the starting node sends packets in 5 phases each with a different data rates (1, 2, 3, 4, and 5 packets/sec). Also, all combination of level 0 nodes and level 1 nodes are tested in this experiment.

In each phase, the nodes in Level 1 send packets to Level 2 with sending rate starting from 1packet/sec to 5 packets/sec with 0.5 step. For example, in first phase the source node sends with 1 packet/sec of data rate and the node in Level 1 starts sending in data rate of 0.5 packet/sec for a period of time then in 1 packet/sec up to 5 packets/sec.

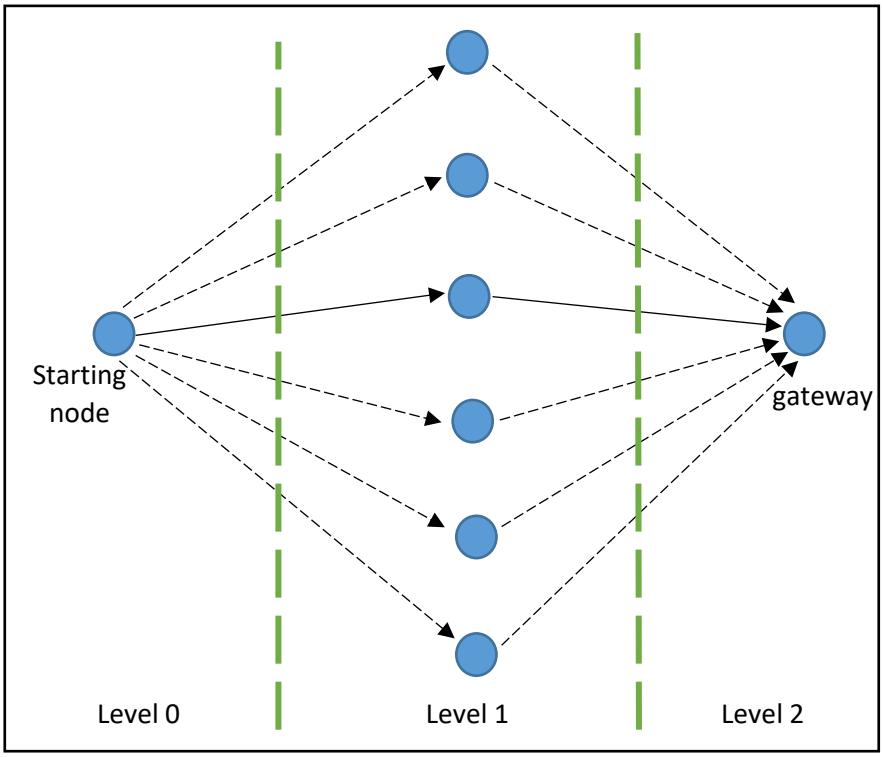


Figure 5- 2: WSN diagram of the experiment for calculating the number of required routes.

The period of 30 minutes for nodes of Level 1 to send in each data rate (for 0.5 packet/sec 30 minutes, and for 1 packet/sec 30, … etc), in this period the nodes of level

1 send packets to gateway with same data rate. Finally, the gateway is connected to laptop to save the received data from Level 1.

To determine the required number of routes for sending packets from starting Waspmote node to Level 1 nodes, firstly the time of congestion and time of node availability (the node is available to receive data after congestion) of Level 1 nodes will be calculated theoretically.

The starting Waspmote node sends its packets to first node (with shortest path) from Level 1, then after the congestion of receiving node will be sent to starting Waspmote node to change the route to next Waspmote node in level 1, and so on so forth until the first congested node becomes available to start receiving packets from starting node.

To calculate the time (seconds) required for the receiving node of level 1 to be congested, in each second after sending and receiving packets the occupation of the received node buffer will be checked until the buffer is congested, then the congestion time represents the required seconds for the buffer to be congested. For the first second, based on the data rate of both starting node and received node the buffer occupation will be determined, the following equation gives the buffer occupation in packets in the required second:

$$B_{pkts} = (T_{\text{sec}} \times S_{s_r}) - R_{sr} \times (T_{\text{sec}} - 1) \quad \ldots\ldots\ldots\ldots \quad (5.1)$$

Where: B_{pkts} is buffer occupation in packets

T_{sec} is time in seconds

S_{s_r} is the starting node sending rate in packets/sec

R_{sr} is the Received node (in Level 1) sending rate in packets/sec

The number of packets for the buffer to be congested is 75% of buffer size in packets. By substituting the number of required packets for congestion in Equation 5.1, the time of congestion can be calculated from Equation 5.2:

$$T_C = \frac{B_C - R_{sr} \times d}{S_{sr} - R_{sr}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (5.2)$$

Where: T_C is the time (in seconds) until congestion

B_C is the number of packets to congest the buffer

d is delay, and it is 1 sec in this particular case

The time for the congested node to become available is calculated by:

$$T_A = \frac{B_C}{R_{s_r}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (5.3)$$

T_A is the time (in seconds) for the node to become available

Based on the above calculations, the number of required routes for the starting variable node to send its data packets is calculated by the integer part of the result:

$$No.of\ Routes = \frac{T_A}{T_C} + 2 \quad \ldots\ldots\ldots\ldots\ldots\ldots (5.4)$$

Equation 5.4 is designed based on empirical attempts to fit the acquired results rather than theoretical basement.

The experimental measurements of congestion time, available time, and the number of required routes are presented for different router packet rates in Figure 5.3 – Figure 5.7:

The presented results in the above figures are for the buffer size of 4000 bytes or 40 packets (each packet is 100 bytes), then the 75% of buffer size is 30 packets is used in the theoretical calculations.

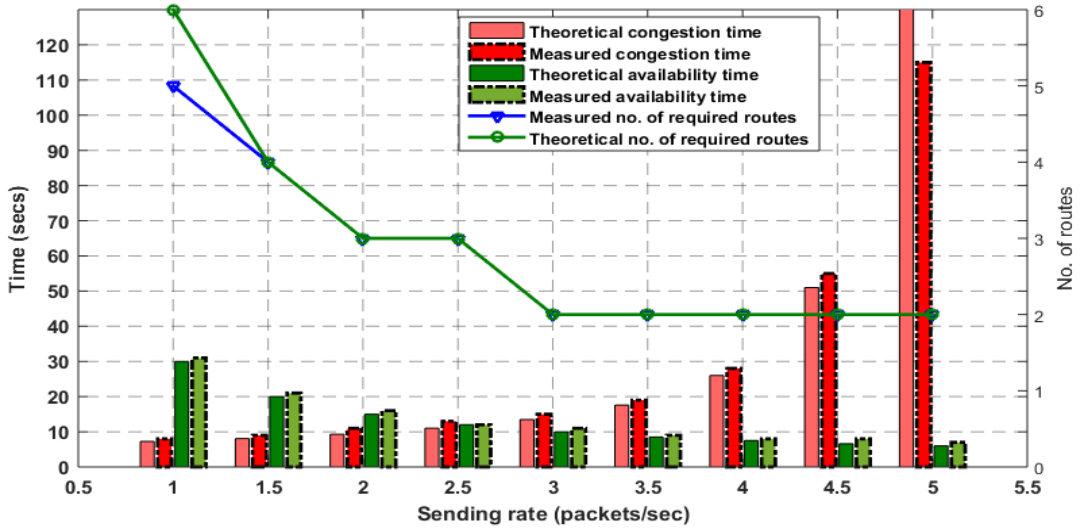


Figure 5- 3: Number of available routes with congestion and availability time for received node in response to 5 packets/sec for the starting node sending rate.

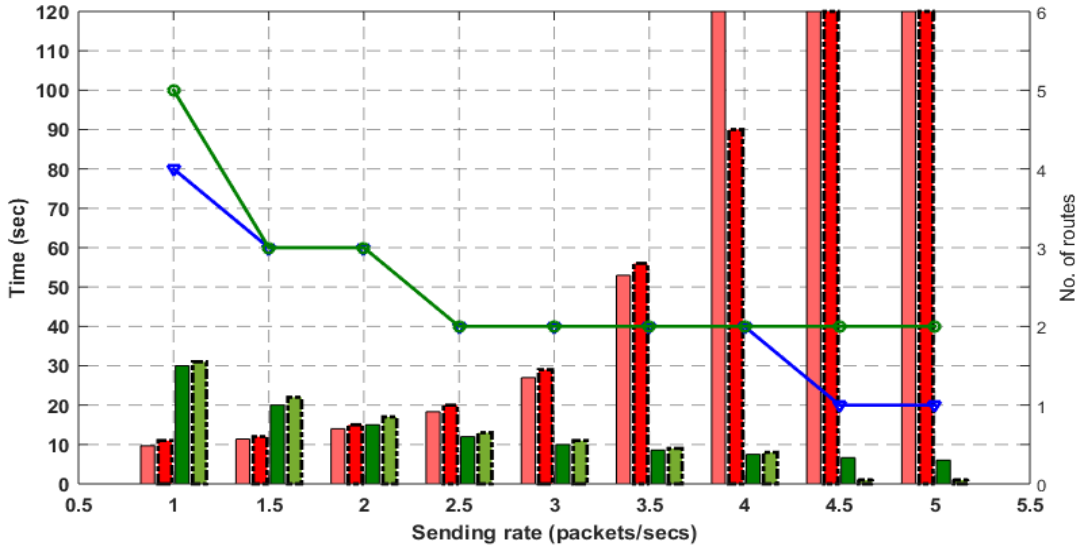


Figure 5- 4: Number of available routes with congestion and availability time for received node in response to 4 packets/sec for the starting node sending rate, legend as in Figure 5.3.

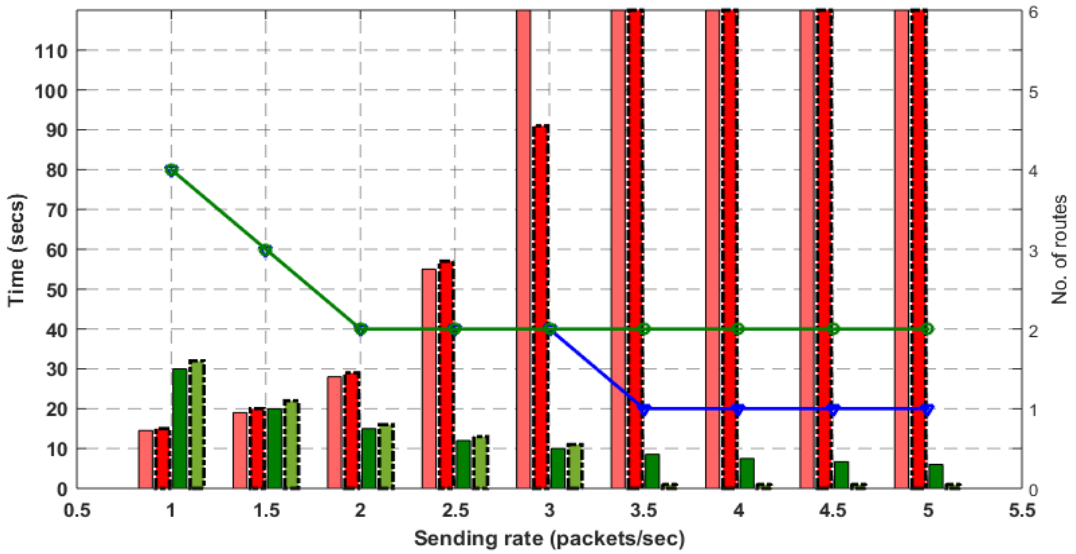


Figure 5- 5: Number of available routes with congestion and availability time for received node in response to 3 packets/sec for the starting node sending rate, legend as in Figure 5.3.

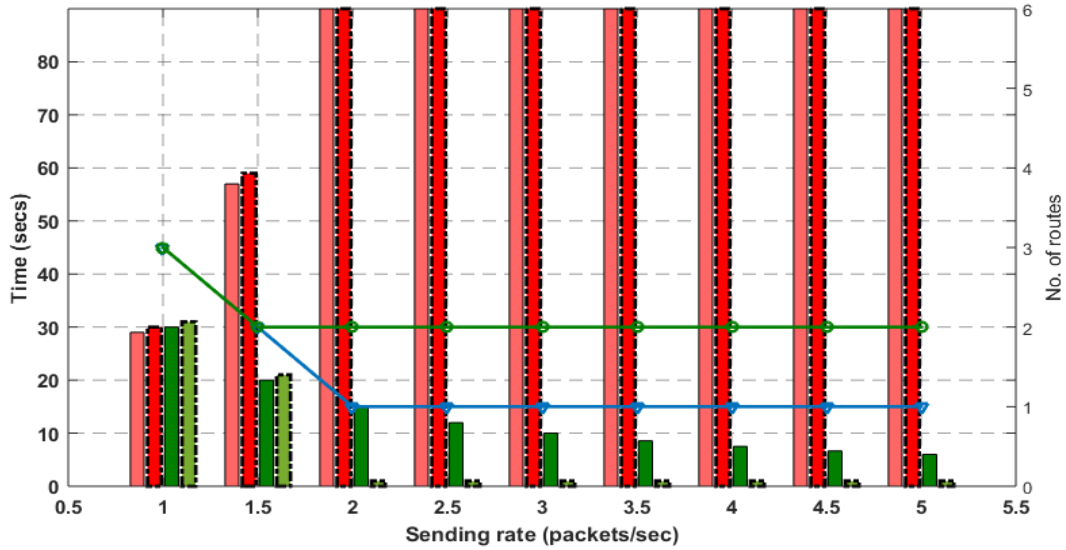


Figure 5- 6: Number of available routes with congestion and availability time for received node in response to 2 packets/sec for the starting node sending rate, legend as in Figure 5.3.

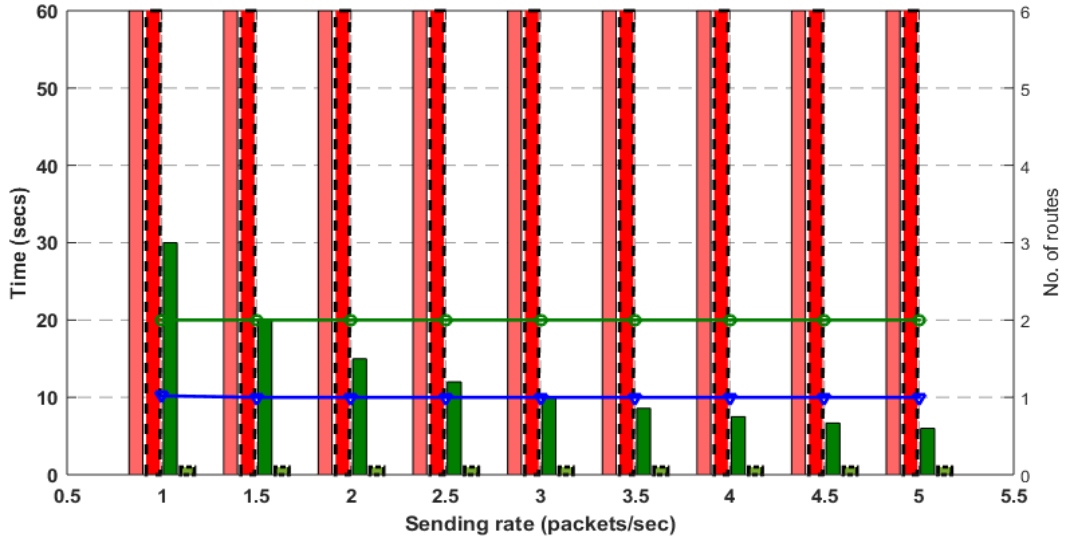


Figure 5- 7: Number of available routes with congestion and availability time for received node in response to 1 packet/sec for the starting node sending rate, legend as in Figure 5.3.

From above figures, the theoretical results and experimental results are nearly the same in most values, except for last case in Figure 5.7.

The results show that the increase in the data rate of the level 1 node will increase the required time for the node to be congested, and the availability time will be decreased. At low sending rate, when the sending rate of the receiving node equal to the sending rate of the starting node, the receiving node will not be congested so is always available, whereas at the high data rate the node might be congested even when the theoretical value predicts no congestion. The reason for that, is at high data rate the link will be busy, and hence packet collision can occur, which will increase the retransmission rate, and as a result increase the buffer congestion.

The number of available routes required for the starting node to send its packets, is decreased with the increase in the data rate of the receiving node. On the other hand, the number of required routes is directly proportional to the increased value of starting node sending rate.

Additionally, from these figures, all the experimental values (congestion and availability times) are larger than the theoretical values. This is because the experimental values of congestion and availability may be subject to collision and retransmission conditions that are not accounted for in the theory.

5.3.2 Effect of number of nodes on network congestion

To study the effect of number of the nodes, several experiments with up to 30 nodes were conducted. The number of sources was fixed to 6 nodes, while the number of nodes was changed from 15 to 30. In a second experiment, the number of nodes was fixed at 30, while the number of sources was changed from 1 to 6. For both experiments a data rate of 1 packet/sec was used in these experiments.

The experiments were conducted in a computer lab as illustrated in Figure 5.8. The sources were deployed in the first line of nodes. Increasing the number of nodes in the fixed area (same area) represents an increase in the node density, which for fixed radio range means increasing the number of neighbour nodes for each node. From the acquired results, the number of levels remains the same which is 5 levels for all combinations of number of nodes. For 30 nodes network, in level 0 there were 6 nodes, level 1 has 9 nodes, level 2 has 6 nodes, level 3 has 5 nodes, and level 4 has 4 nodes. All the number of nodes for each level is illustrated in Table 5.1 which is for increasing the number of nodes.

Table 5. 1: Number of nodes in each level for the deployment of nodes from 15 to 30 nodes.

No. of nodes in the network	Level 0	Level 1	Level 2	Level 3	Level 4
15	3	4	3	3	2
20	4	6	4	3	3
25	5	7	5	4	4
30	6	9	6	5	4

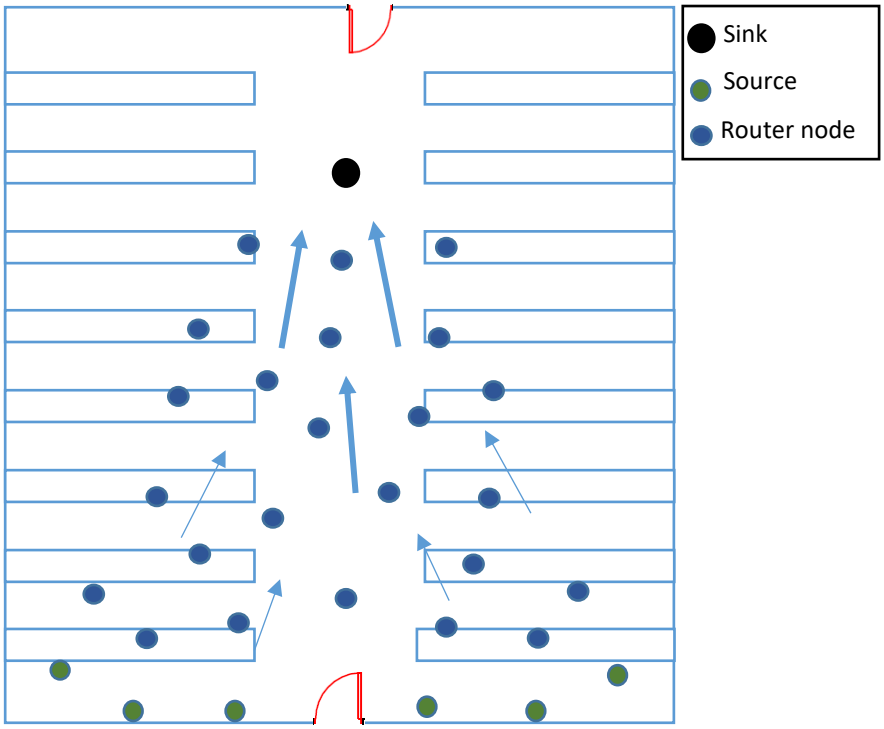


Figure 5- 8: Computer lab in R-block of Engineering department, with deployment of nodes.

Figure 5.9 and Figure 5.10, show the effect of increasing the number of nodes on PDR and throughput for HTAP and no congestion control. PDR increased with the number of nodes because this provides more neighbour nodes and consequently more routes for resource control of the network congestion.

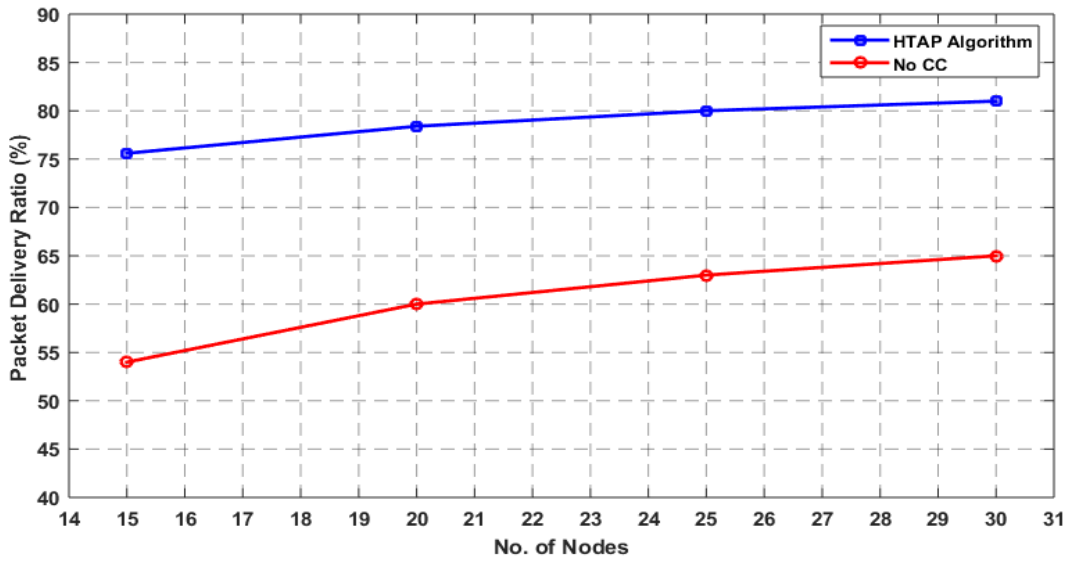


Figure 5- 9: PDR as a function of number of nodes with fixed number of 6 sources.

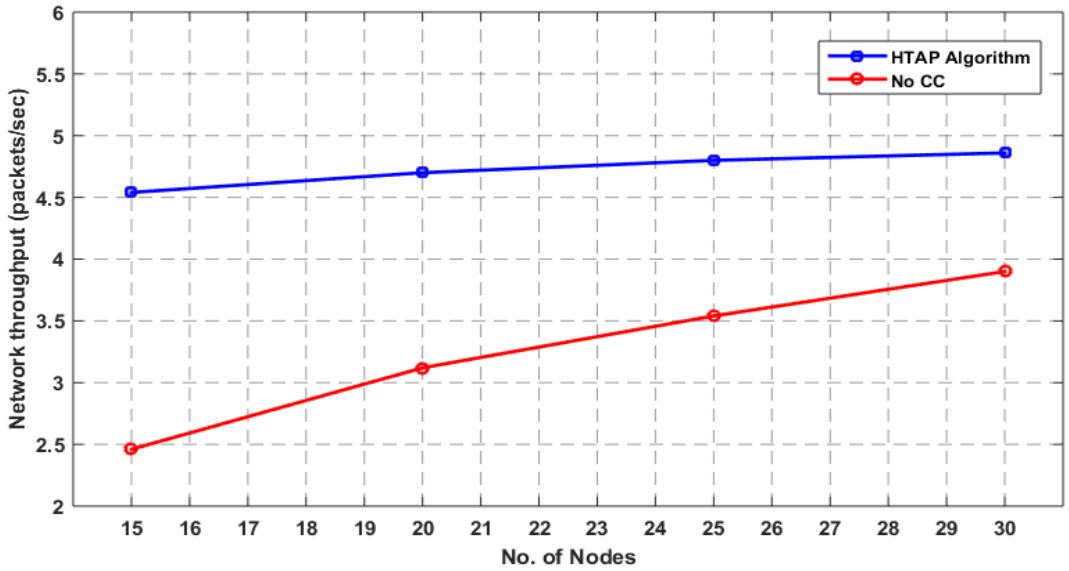


Figure 5- 10: Network throughput for the 6 sources as a function of number of nodes.

Network throughput also increases with the number of nodes because the resource control algorithm will be able to deliver more packets to sink with the increase in the number nodes, since each congested node will have more neighbour nodes in its neighbour table. In Figure 5.10, the increase in the throughput of no CC is higher than HTAP, this is because more paths will be available for the no CC algorithm in the running of the experiment.

In the second experiment, the number of nodes was fixed at 30 nodes, but the number of sources was increased from 1 to 6. Figure 5.11 and Figure 5.12 show the PDR and throughput as a function of the number of sources respectively.

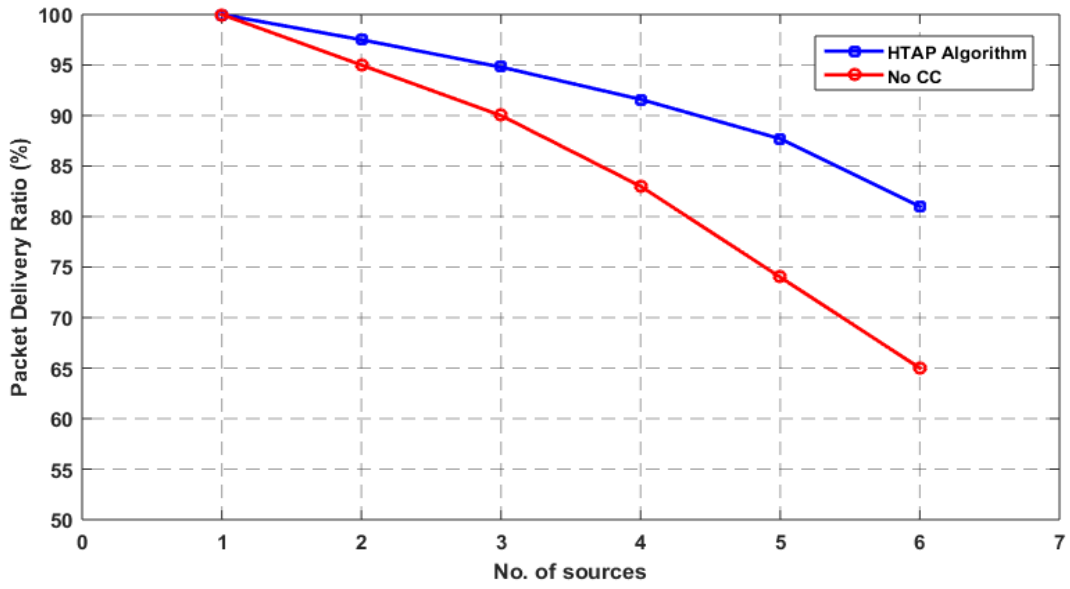


Figure 5- 11: PDR as a function of no. of sources for the network of 30 nodes.

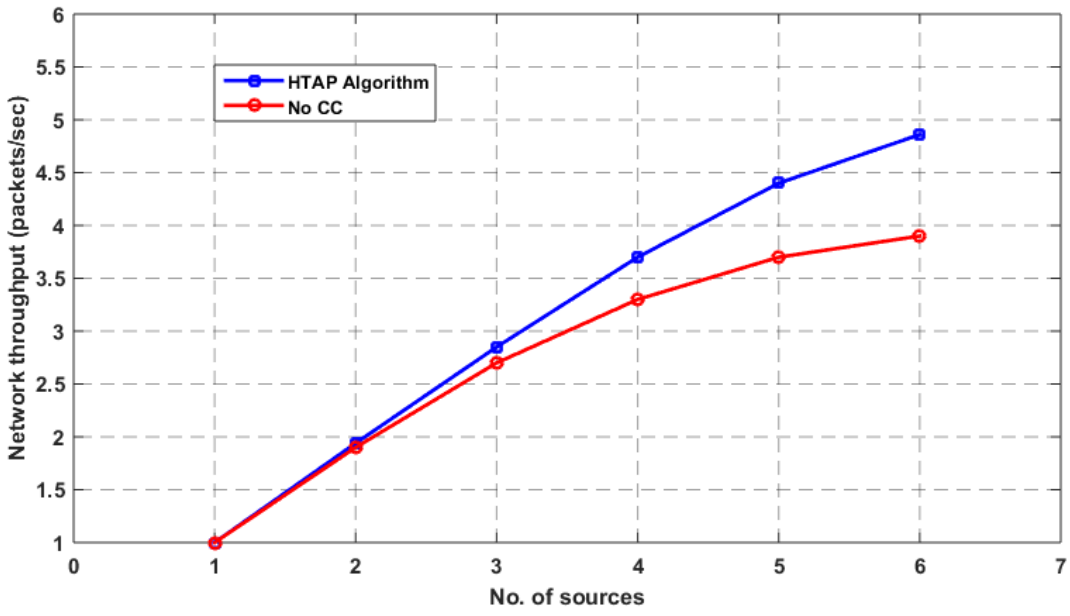


Figure 5- 12: Network throughput as a function of no. of sources for the network of 30 nodes.

Both Figure 5.11 and Figure 5.12 indicate that the increase in the number of sources decrease the performance of the network from the ideal, this because the increased number of packets generated by sources will increase the packets collisions and node level congestion. In Figure 5.11, when there is 1 source in the network the PDR is 100% since all the packets sent from the source are delivered to the sink, but when the number

of sources increased the number of packets also increased and the network congestion is likely to occur. The resource control algorithm will mitigate the effect of congestion and hence better network performance will be achieved compared to network without congestion control.

5.4 Effect of packet size on network congestion

As stated in Chapter 1 and Chapter 2, when an event occurs in critical application, a sudden surge of data packets will be transmitted from the sources to the sink. Congestion can result in the node buffer or the high traffic of data packets may cause severe contention between transmitted packets in the wireless link. Congestion causes packet drops and hence low service quality will result. In addition, the overall network delay and the energy consumption will increase due to the retransmission of dropped packets. The effect of packet size on congestion, channel contention, and on network performance, is a subject that has been studied by many researchers e.g. (Sankarasubramaniam *et al*., 2003; Yaakob *et al*., 2010). Employing a large packet size in a WSN can cause data corruption which means packet retransmission will increase. On the other hand, using short packet size may increase data transmission reliability, but the short packet size may not be efficient in terms of the fraction of payload data to total data (i.e. payload and header) which means more data transmission is required which leads to higher energy consumption.

The performance, of a WSN with 30 Waspmote nodes was tested with different frame sizes (from 30 to 100 bytes). The effect of different data rates (1.7, 2.6,3.5, and 5.5 packets/sec) was also tested.

In this section we will study the effect of different packet sizes on the network performance while using congestion control (HTAP). Before presenting the experimental results (5.4.2) the 802.15.4 packet configuration will be explained.

5.4.1 Configuration of an 802.15.4 packet

The physical layer of the IEEE802.15.4 standard defines 4 different frames: Data frame, Acknowledgement frame, Beacon frame, and MAC command frame. Figure 5.13 shows the IEEE802.15.4 data frame (Adams, 2006). The Physical Protocol Data Unit (PPDU) is enabled for the transmission and reception across the physical radio channel by the physical layer data service. The leftmost field of PPDU is transmitted or received first.

The PPDU consists of the following parts:

- Synchronization Header (SHR) contains a preamble sequence which is 4 bytes and a further 1 byte for the Start of Frame Delimiter (SFD). The receiver acquires and synchronizes the incoming data from the preamble sequence, and the SFD indicates the end of synchronization and the start of the packet data.
- Physical Header (PHR) which specifies the frame length in bytes for the PSDU.
- Physical Service Data Unit (PSDU), contains the payload of the physical packet. Also, the PSDU represents the Mac Protocol Data Unit (MPDU).
- Mac Header (MHR), which consists of 2 bytes of frame control, 1 byte for data sequence number, and 4 to 20 bytes of address data.
- Mac Service Data Unit (MSDU) represents the Mac packet payload and has a maximum number of 100 bytes of data.
- Mac Footer (MFR) is a 16-bit of frame check sequence.

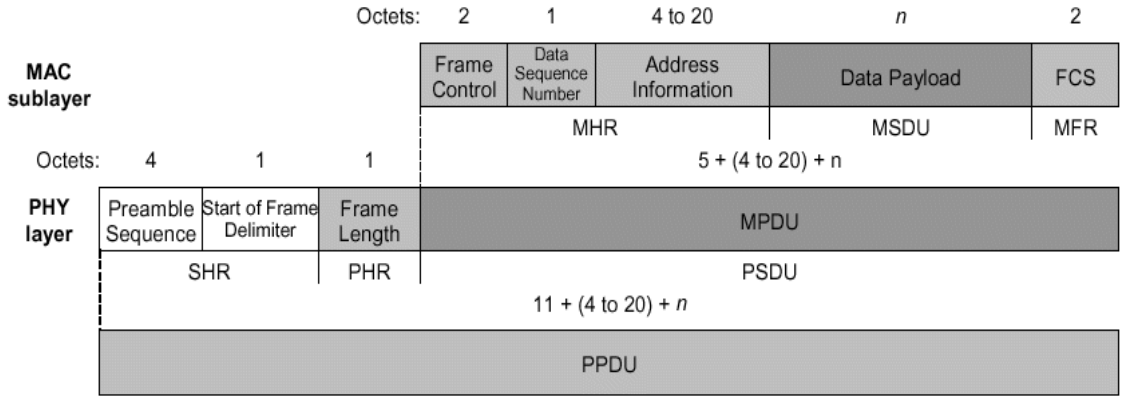


Figure 5- 13: IEEE802.15.4 data frame. The number of bytes is indicated above each part of frame (Adams, 2006).

HEADER										PAYLOAD						
<=>	Frame Type	Num Fields	#	Serial ID	#	Waspmote ID	#	Sequence	#	Sensor_1	#	Sensor_2	#	...	Sensor_n	#

Figure 5- 14: ASCII frame structure (Libelium, 2013).

The Mac payload (MSDU) is called the API frame, which can be monitored using the serial monitor (Section 3.2.3). There are two types of the API frames: ASCII and Binary. The ASCII frame structure is illustrated as in Figure 5.14, and its fields are defined as follows (Libelium, 2013):

The ASCII Header consists of:

- Start Delimiter (3 bytes), it is composed by three characters: "<=>", these 3 bytes identifying frame starting.
- 1 byte Frame type, with this byte the frame is identified either ASCII or Binary, also the aim of this frame such as event frame or alarm frame, will be identified by this byte.
- 1 byte Number of Fields, specifies the number of sensor fields sent in the frame.
- 1 byte separator #.
- 10 bytes serial ID, this serial provides a different identifier for each Waspmote device.
- 0-16 bytes for Waspmote node ID, has a variable size between 0 and 16 bytes, the node ID is a string to identify each Waspmote node inside the WSN.
- 1-3 bytes Frame sequence, this is an 8-bit frame counter that count from 0 to 255 then it reset to 0. The number is converted to a string of 3 bytes. Also, the use of this sequence number gives indication of the received and lost packets.

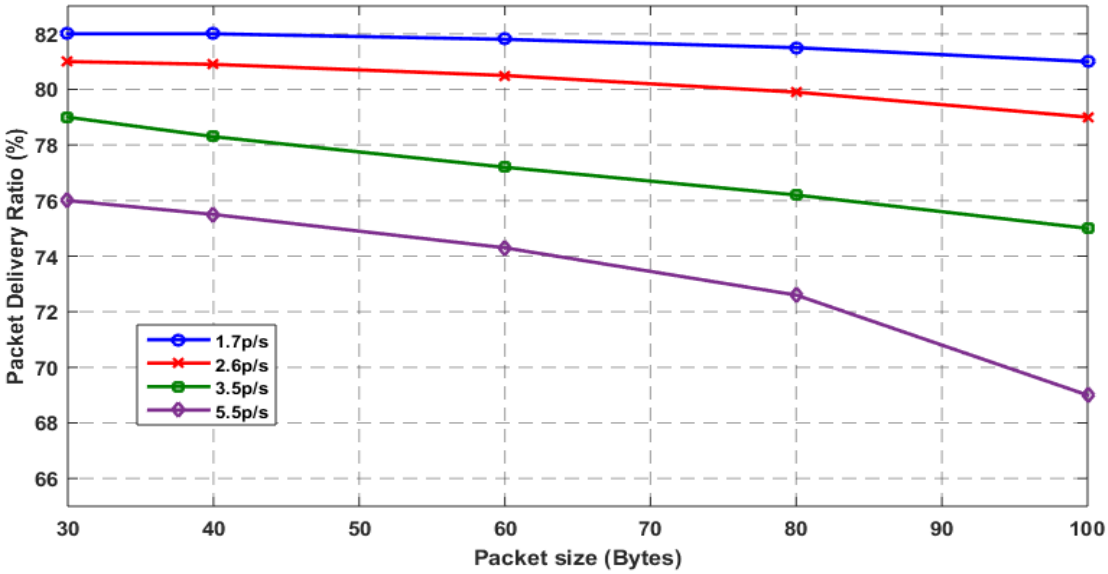


Figure 5- 15: PDR as a function of packet size for different data rates.

The ASCII payload is the remaining part of the 100 bytes after considering the frame header (which is 17 – 35 bytes).

5.4.2. Effect of packet size on PDR

The variation of PDR with different packet sizes and with different data rates is presented in Figure 5.15, PDR decreases with increasing packet size and packet rate. When the size of the packet is small, bit corruption might occasionally happen then the transmitted packets are more likely to be received by the sink, as a result the PDR is larger. As the packet size is large then the possibility of packets collisions will be

increased then number of dropped packets increases and hence PDR will be decreased (Leghari *et al*., 2013). Also in Figure 5.15, the value of PDR is decreased as the value of data rate is increased, this is because when the data rate is low the number of generated packets is small, then this leads to lower possibility of collision, and hence higher number of received packets then higher PDR (Leghari *et al*., 2013; S *et al*., 2015). Whereas a higher data rate leads to more packets more likely packet collisions and hence more dropped packets and low PDR.

Furthermore, from this figure, the standard deviation for the PDR values (76, 79, 81, and 82) at the packet size of 30 bytes is 2.65, whereas the standard deviation of PDR values (69, 75, 79, and 81) at 100 bytes is 5.29. This is because the shorter packet size decreases the chance of packet collisions, also the number of transmitted packets and received packets remains nearly the same, which in turn produces approximate values of PDR.

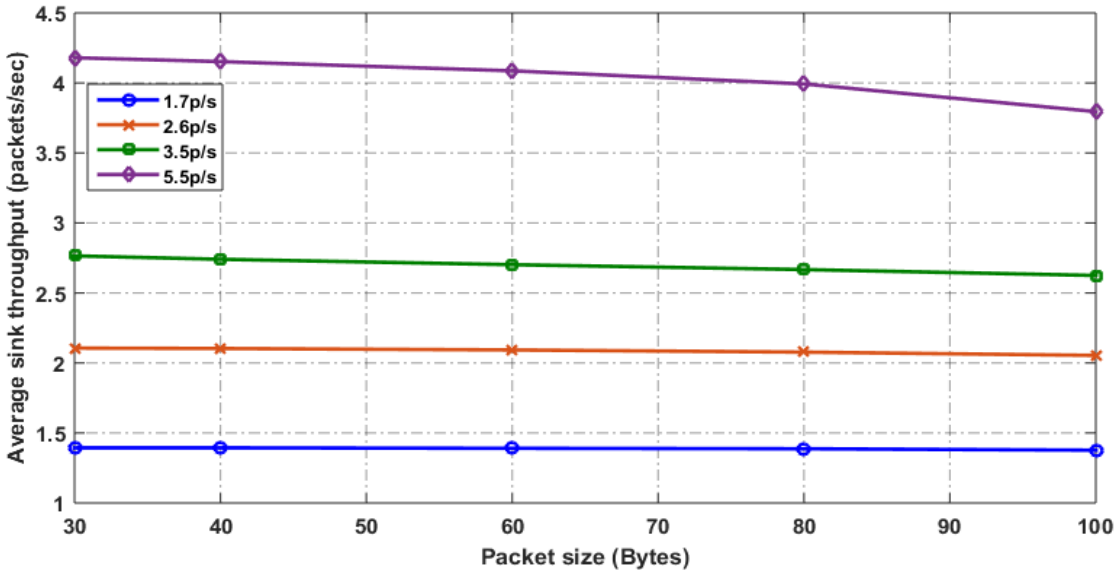


Figure 5- 16: Throughput as a function of packet size for different data rates.

Figure 5.16 presents the throughput as a function of packet size, it can be seen that throughput at low data rate nearly the same value for the change of packet size from 30 to 100bytes. As the data rate increases the throughput starts to be affected, then at the higher data rate of 5.5 packet/sec the throughput decreased sharply at 80 to 100 bytes.

5.4.3. Effect of packet size on End to End delay

The second investigated metric was the average End to End delay.

In this section, for the 802.15.4 packet size the required-on air time (from the sender transceiver to receiver transceiver) to transmit a data packet from one XBee802.15.4 module to another XBee module is calculated as follows:

The accurate calculation for the Time on Air, the following references of XBee 802.15.4 with the frequency of 2.4GHz should be taken into consideration:

- Max. payload=100 bytes
- RF data rate 250kbps
- Time to send 1 Byte= (1 / 250 kbps) x 8 = 32 µs
- From Figure 5.14, Max. packet size = Phy. Header + Mac Header + Mac Footer + Mac payload

  = 6 + 23 + 2 +100 = 31 + 100 =131 bytes

The Time on Air for the transmitted packet can be calculated as follows:

The Time on the Air to send 1 Byte payload:

T_{air} (1 Byte) = (31 + 1) x 32 µs = 1.024 ms

The required time to send maximum packet size (131bytes) is:

T_{air} (100 Bytes) = (31 + 100) x 32 µs = 4.192 ms

Table 5.2 presents the time on air for the frame size started from 30 to 100 bytes.

However, for our experiment the calculated time is computed from the time of generating the packet until it received by the sink. Figure 5.16 shows the average End to End delay (sender processing time +access time + on air time + receiving time + receiver processing time) for different packets sizes of the same selected data rates.

Table 5. 2: Required time on air for different packet sizes.

Frame size (Mac payload) (Bytes)	Full packet size (Bytes)	T_{air} (msec)
30	61	1.952
40	71	2.272
60	91	2.912
80	111	3.552
100	131	4.192

From Figure 5.17, the End to End delay increases with increasing in the packet size, this is because the longer packet size requires more time, also the longer packet size increases the packet collisions then increases the number of retransmission which in turn increases the End to End delay.

Also, as the node data rate increases, the number of generated packets will be increased and this increases the possibility of collisions, then the number of retransmissions increases which also leads to a higher value of End to End delay.

In addition to above discussion, also from Figure 5.16, the calculated standard deviation for the values of End to End delay (4.1, 4.2, 4.6, and 5) at the packet size of 30 bytes is 0.411, and this value is less than the 0.793 standard deviation of End to End delay values (4.6, 5.2, 6.1, and 4.3) for 100 bytes. The reason for that is because the shorter packet size means lesser packet collisions and in turn less number of retransmission, which makes the values of End to End delay closer to each other.

From the above discussion, it can be concluded that the packet size has direct effect on network performance. Employing long packet size in a WSN increases the data bits corruption and leads to increase the packet retransmission, and because the transmission of packets toward the sink is consumed the most power of WSN therefore, the longer packet size will decrease the efficiency of WSN in overall. On the other side, using short packet size may produce better PDR and less End to End delay, but in the terms of the capacity of the payload it is not efficient (Leghari *et al*., 2013).

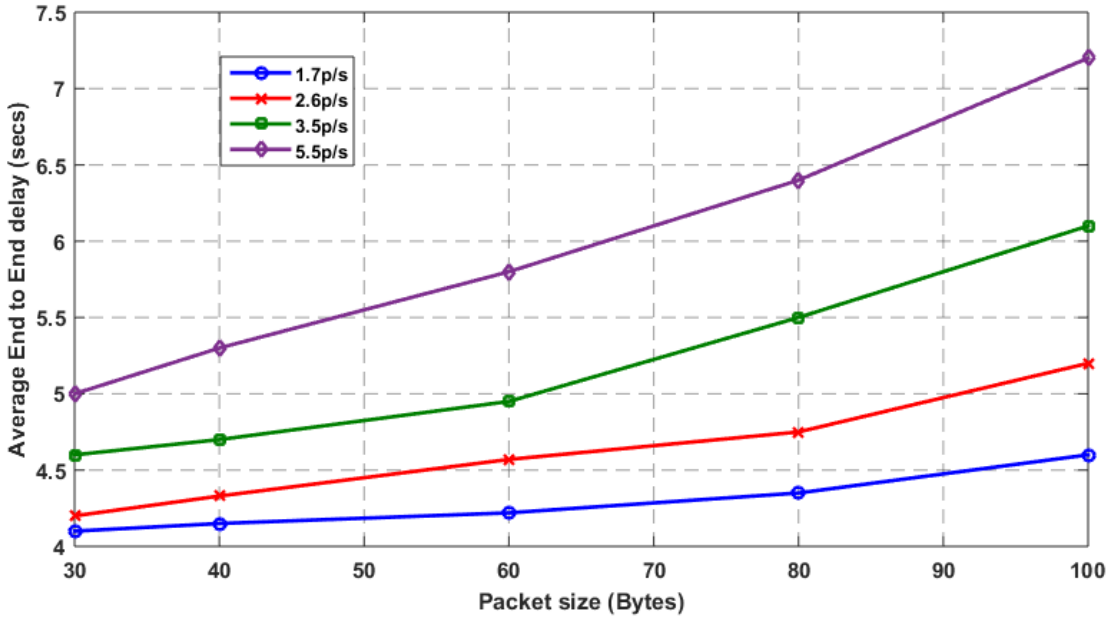


Figure 5- 17: Average End to End delay as a function of packet size for different data rates.

5.5 Summary

This chapter investigates some factors that affect the network congestion. First introduced factor is the use of different congestion threshold values. The buffer size congestion threshold values of 25%, 50%, 75% full of the buffer, and the adaptive method are presented with the use of packet size 100 bytes and for two data rates: 1.7 and 5.5 packets/sec. It was found that the fixed threshold values have lower performance than the adaptive method. The proportion of PDR values at 1.7 packet/sec to the value at 5.5 packets/sec are increased with the increase in congestion threshold and this is a sign that the PDR with high data rate is affected negatively with the increase in the value of congestion threshold.

The effect of the number of nodes in the WSN on the congestion is discussed through this chapter. The PDR values are increased with the increase in the number of nodes, and the HTAP algorithm has higher PDR values compare to no congestion control algorithm. In Figure 5.9, the ratio of the value of PDR of HTAP to PDR of NCC is decreased with the increase in the number of nodes.

Also, the number of sources was investigated, and found that the increase in the number of sources affects negatively to the performance of HTAP algorithm in both the PDR and the network throughput. From Figure 5.11, the ratio of PDR of HTAP to the PDR of NCC decreases with the increase in the number of sources.

The last investigated factor is the effect of packet size. It is found that employing short packet size will increase the PDR values and decrease the End to End delay, but also the short packet size is not efficient in the terms of the capacity of the payload. Using longer packet size will increase the packet retransmission because of the data bits corruption occurred because of the use of large packet size, as well as the increase in the power consumption due to increase in the retransmission, since the most power consumption happens in the data transmission. Also, the ratio of the value of PDR at low data rate (1.7 packet/sec) to the value of PDR at higher data rate (5.5 packets/sec) is increased with the increase in the packet size, and this is an indication that the behaviour of PDR is getting worse with the increase in the packet size.

From figure 5.17, the values of the ratio of end to end delay at 1.7 packet/sec to the value of end to end delay at 5.5 packets/sec are decreased with the increase in the packet size, and this is an indication that the increase in the packet size with low data rate has less end to end delay than it has with higher data rate.

Chapter 6: Congestion control using a combination of resource control and traffic rate control

6.1 Introduction

Measurements of the effect of a congestion control technique that combines the resource control (i.e. establishing an alternative path between the transmitted nodes) and the traffic rate control (changing the transmission rate) on network performance presented in this chapter.

The performance of the HTAP algorithm in mitigating the effect of congestion has been measured in Chapter 4 and 5. The network performance (e.g. PDR and throughput) is improved when utilizing HTAP algorithm compared to a network without congestion control.

However, in some cases the network performance might be reduced even with the use of HTAP algorithm, which is occurred in the exist of single path. In more detail, in HTAP algorithm, any node after the scan phase finds its neighbour nodes, then it divides the neighbours into 2 groups: one group closer to the sink and the second group closer to source (see section 4.2.1). However, sometimes the sink ward group will only contain one node. So, in this case there is only one path to send packets to the sink (i.e. no alternative path). Therefore, in the event of congestion, the transmitter nodes don't have an alternative path, and hence the congested node will start to drop incoming packets which leads to a decline in the network performance. Also, the single path might exist in the case of 2 paths and one of the paths is congested, then the remaining is only single path, which can be transmit through it. To address this problem, the rate control technique can be used to mitigate the node congestion in the same time with HTAP algorithm.

In this chapter, the traffic rate control technique is introduced together with related protocols, then the intended improvement of the HTAP algorithm is implemented and experimentally tested.

6.2 Traffic rate control technique

As stated in chapter 2, the mechanisms for controlling the congestion in WSNs are composed of 3 stages: 1) Congestion detection, 2) Congestion notification, and 3) Congestion control.

In the first stage, the congestion is detected by using buffer occupancy, channel load, or the combination of channel load and buffer occupancy. Buffer occupancy has been

employed in this work for reasons discussed in Chapter 2. In stage 2 when congestion occurs a control message is sent to the nodes that are transmitting the packets to change their route. Finally, in stage 3, the congestion is controlled either by changing the route (which is known as resource control) as in HTAP algorithm or by changing the transmission rate (traffic rate control) as in CCF (Chatterjee *et al*., 2010).

Congestion control using the traffic rate control technique is employed by many protocols in the literature. In the following paragraphs, some of the traffic rate adjustments are presented briefly:

In Adaptive Rate Control (ARC) protocol, the traffic rate is adjusted by the relay node (intermediate node), which increases its rate by a constant α, and it can decrease its rate by multiplying the rate with a factor β ($0 < \beta < 1$) (Woo & Culler, 2001).

The Congestion Detection and Avoidance (CODA) protocol (Cy Wan *et al*., 2003) adjusts the traffic rate by using the Additive-Increase/Multiplicative-Decrease (AIMD) algorithm. In this algorithm the sender increases its traffic rate linearly (by growing its rate by 1 packet in each Round Trip Time - RTT) until the congestion occurs (the packets start to drop) when the sender will decrease its rate exponentially (Li *et al*., 2013).

FUSION was introduced by Hull *et al*. (2004) to mitigate the congestion. FUSION decreases the random back-off timer for the CSMA of the congested node, then the congested node has the priority in the channel compared to uncongested nodes.

In this project, the congested node computes the number of nodes that transmitted data, then according to this number the traffic rate will be adjusted. The congested node will send a control message with this number to the transmit nodes, after receiving the control message each node will decrease its traffic rate to the equivalent of dividing the traffic rate by the number included in the control message.

6.3 The effect of network topology on the performance of HTAP

The experimental results presented in Chapter 4, verified the results of simulated HTAP algorithm in its behaviour for the PDR, throughput, and the End to End delay. To this end, the quality of service metrics for the WSN running HTAP algorithm might be diminished by some types of network topology.

The HTAP algorithm employs a source-based tree scheme for its topology control, (see Chapter 4). This scheme suggests that each node should be connected to at least two higher-level nodes, and if it is only connected to one node, then the closest nodes to sink in the same level of the node, should be added to the neighbour table of this node. If there

are no such nodes in the same level, the node should not be considered as a good neighbour for data forwarding (Sergiou, 2012). However, this is likely to affect network performance because omitting the node for data forwarding might lead to the node being neglected, and hence a waste of network resource. More importantly, if this node is the sole connection between a group of nodes in the network and the sink, then data from this part will be lost.

Figure 6.1 presents a possible network topology after the random deployment, which has a single node in the higher-level nodes. After running the HTAP algorithm as explained in Section 3.2.6.1, each sensor node discovers its neighbour nodes. The neighbour nodes will be classified into 2 groups according to their distances from the sink position: sink direction group and anti-sink direction group. The nodes (03,04, and 05) in level 0, have only one node (Node07) in their next hop (level 1). Since in the original HTAP, when Node 07 is almost congested it sends control messages to nodes 03,04, and 05 to change their route (alternative path) to mitigate the congestion. In this topology, nodes 03,04, and 05 don't have an alternative path to send their data, therefore the packets start to be dropped, and hence the quality of service will be degraded. Consequently, in order for these nodes 03,04, and 05 to keep sending their data along the same route Node 07, the improved HTAP employs the traffic rate control technique in Nodes 03,04, and 05 to mitigate the congestion in Node 07, and then the packets transmission will be continued albeit at slower rate.

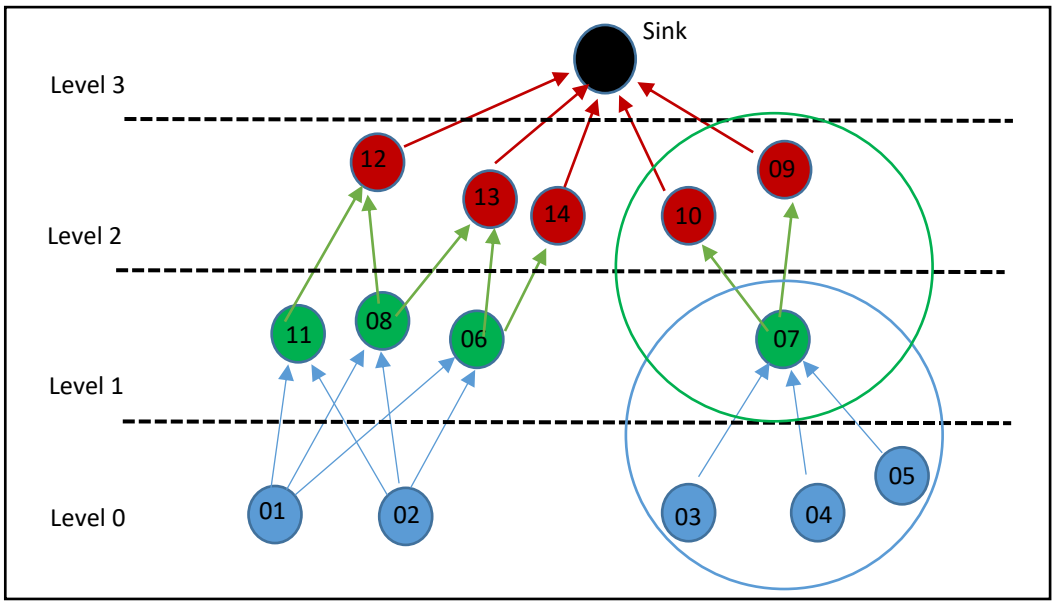


Figure 6- 1: WSN diagram presents single transmission path.

Based on the problem described above, the improved algorithm for HTAP uses the combination of the resource control and traffic rate control techniques. The improvement also suggests for each relay node faces congestion (even it has multipath to send its data) to include in its control message the number of transmitting nodes (in its lower-level nodes). This number will be used by the transmitting nodes (lower level nodes) to reduce their transmission rate, as illustrated in Figure 6.2, which lists the steps of employing the traffic rate control at the congested node.

step1- relay node starts to receive packets
step2- from the received packets headers, relay node determines the number of transmitting nodes, and store the number in variable n
step3- if relay node starts to be congested:
- relay node inserts the variable n in its control messages
step4- if the transmitting node receives control message, it will:
- reduce its transmission rate (r_origin) to: r_new = r_origin / n
step5- if the relay node become available, it will send availability message to transmitting node to retrieve their original transmission rate (r_origin)
step6- if the congestion happens again, this procedure will be repeated again from step 1

Figure 6- 2: Steps of employing traffic rate control in the improved HTAP algorithm.

Another place where a single path may exist is on the edge of the network. An example is given in Figure 6.1, where for Node 11 there is only a single path to send its data to the sink. Node 11 sends its data to Node 12 only, whereas Node 12 receives data from two nodes in its lower-level (Node 08 and Node 11). Therefore, in the event of congestion in Node 12, Node 11 will reduce its transmission rate and continue to send data packets to Node 12. While Node 08 will change its path to Node 13. Moreover, in this example of the existence of Node 11 on the edge of the network, the advantage of inserting the number of transmitting nodes in the control message of any congested node is illustrated, because Node 11 without the number of transmitting nodes will not recognize that there is a single path (only Node 12), and therefore in the event of

congestion Node 11 when it receives the control message will reduce its transmission rate immediately.

The flowchart for the code of the improved HTAP algorithm is presented in Figure 6.3.

The code has two parts, one part for normal HTAP algorithm and the second part for the improvement.

The improved HTAP algorithm has a second enhancement on the HTAP algorithm, which is related to power exhaustion. In the case where relay node has failed due to battery exhaustion, the transmitting nodes will update their table of neighbour nodes. If the updated table of neighbour nodes has only 1 relay node, in the event of congestion the transmitting node will use traffic rate control instead of the resource algorithm.

6.4 Effects of the existence of a single path on the network performance

To investigate the effects of a single path on the network performance using HTAP algorithm, two different topologies were used. The first topology had a single path close to the sources (Figure 6.4), and the second topology had a single path close to the sink (Figure 6.5). Table 6.1 and Table 6.2 presents the ID of single path in both topologies. Each topology was tested with both the normal HTAP algorithm and the improved HTAP algorithm.

To ensure the existence of a single path a 30-dB attenuator was used with the nodes in the path to decrease the Waspmote range, while the power level of XBee transceiver was adjusted to create the required connectivity between the nodes without interference with other hops. The source data rate was varied from 1 to 5 packets/sec.

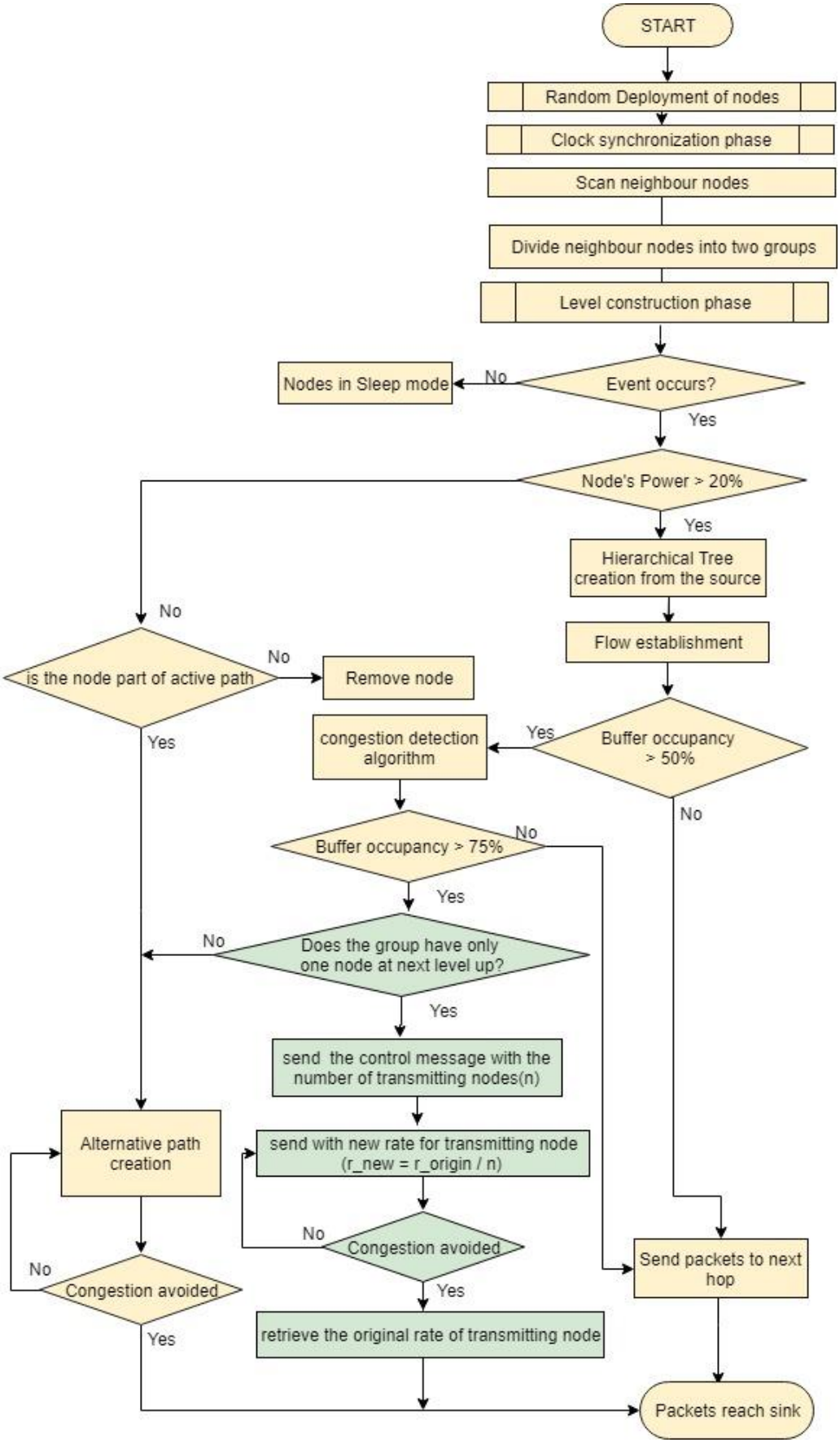


Figure 6- 3: Flowchart of the improved HTAP algorithm (the green parts).

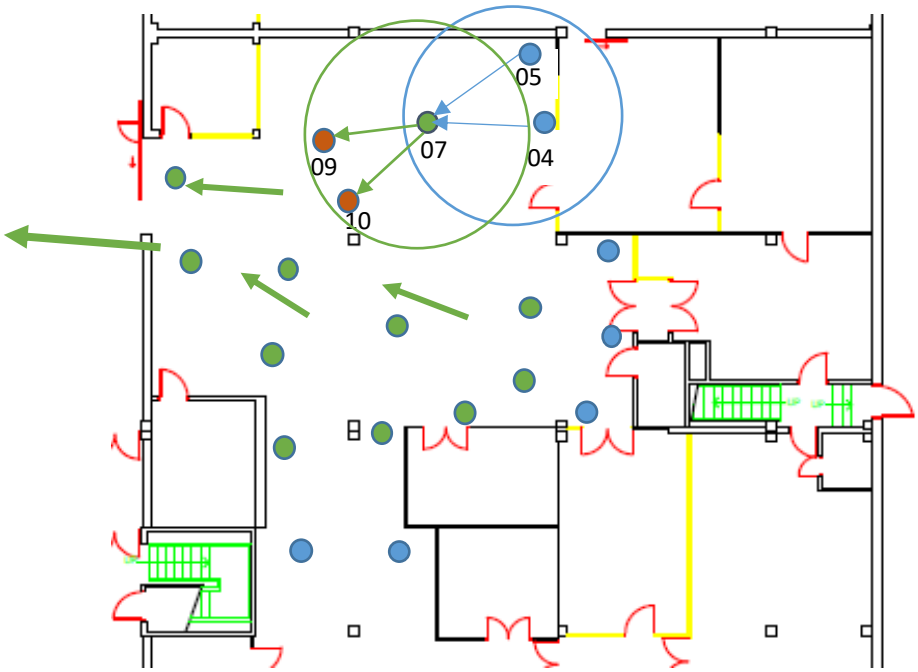


Figure 6- 4: Section of the experimental WSN with a single path close to the sources (blue nodes).

Table 6. 1: The nodes of the single path close to the sources.

Single Path- close to sources	Lower level nodes	Higher level nodes
First single path with node - 07	04, 05 (sources)	09, 10

Table 6. 2: The nodes of the single path close to sink.

Single Path- close to sink	Lower level nodes	Higher level nodes
First single path with node - 34	31, 32 (relay nodes)	Sink

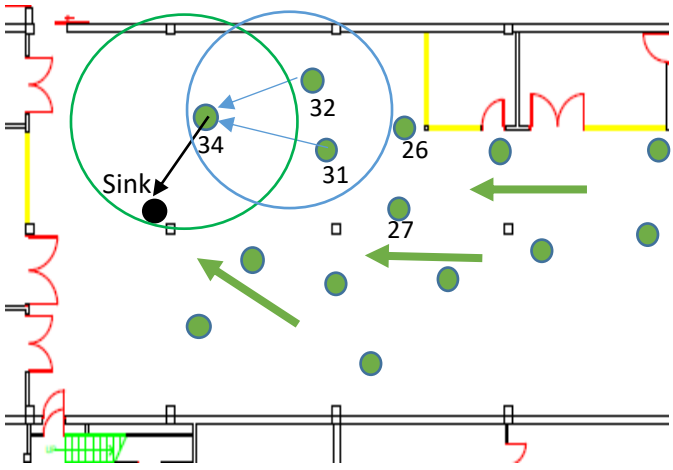


Figure 6- 5: Section of the experimental WSN with a single path close to the sink.

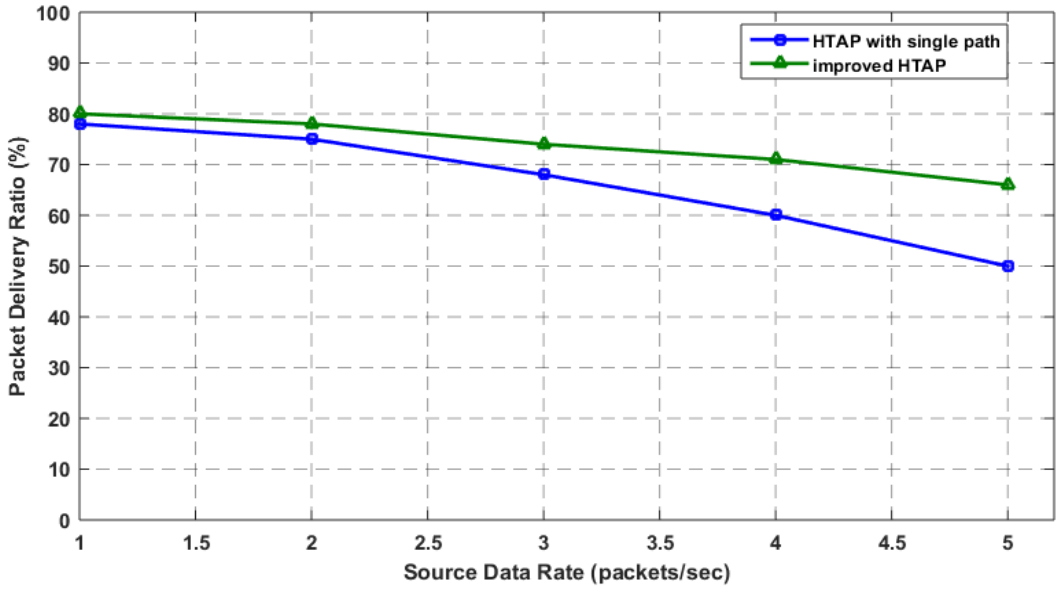


Figure 6- 6: PDR as a function of source data rate for normal HTAP algorithm and the improved HTAP algorithm where the single path is close to the sources.

The PDR measured when the single path is close to the network sources is presented in Figure 6.6, and when the single path is close the sink is presented in Figure 6.7.

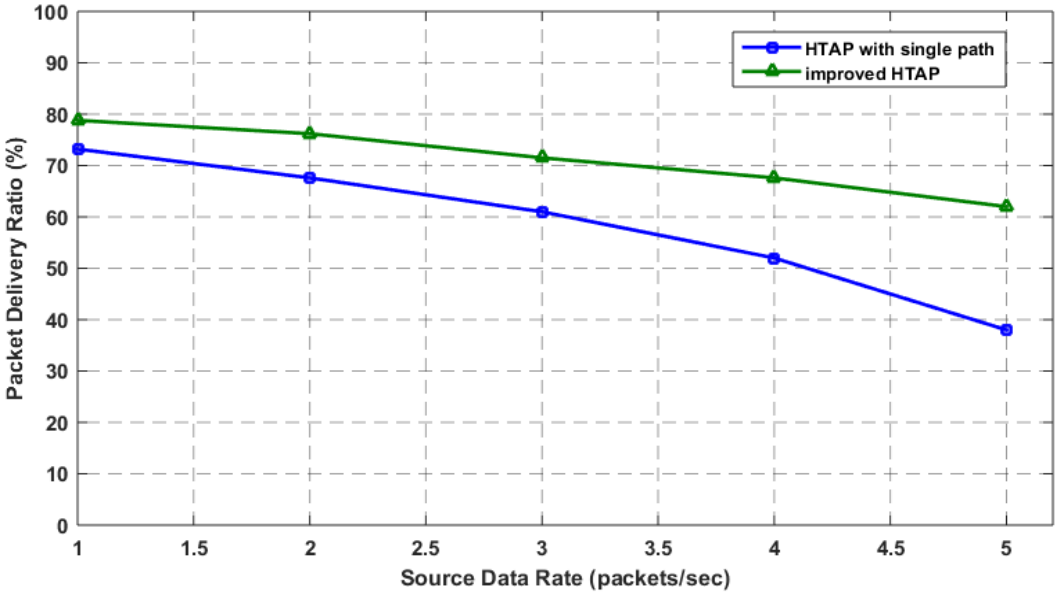


Figure 6- 7: PDR as a function of source data rate for both: normal HTAP algorithm and the improved HTAP algorithm in the 2nd topology of single path closes to sink.

For the experiments of applying normal HTAP algorithm, it's noticed that the existence of the single path in the network (close to sources) makes the PDR values decrease with the increase in the source data rate. This is because when Node 07 (as in Figure 6.4) is congested, it doesn't have alternative path to send its data, so the congestion leads to degradation in network performance. On the other hand, applying the improved HTAP algorithm (using the technique of combining resource control and rate control) will enhance the PDR of the network as presented in Figure 6.6.

From the measurements in Figures 6.6 and 6.7 the improvement to the HTAP algorithm (by using the combination of resource control and rate control) has improved the PDR for both topologies.

However, the PDR results of both topologies as in Figure 6.6 and Figure 6.7, show that the existence of a single path close to sink has more effects on the PDR values compared to the existing of a single path close to sources. This is because Node 34 (in Figure 6.5) is more likely to be congested since it's in the neck of the network (close to sink), and the network neck means that all the network data should pass this neck to sink, then more data need will pass through Node 34, which in turn the congestion is more likely to happen, and hence more packets drop.

Furthermore, from Figure 6.8 the throughput values of improved HTAP with single path close to source is better than with single path close to sink, and this can be interpreted in the 2nd topology with the running of improved HTAP as follows:

- In Figure 6.5 (2nd topology), when Node 34 is nearly to be congested, it will send control messages to nodes (31,32) about congestion
- Nodes (31,32) will decrease their transmission rate (rate control technique)
- But, these two nodes (31,32) will face congestion, this is because the incoming data from nodes (26,27).
- Nodes (31,32) will send control message to nodes (26,27) to change their path (resource control), then data will be directed to other side of the neck, and hence more data will be passed from it then the congestion is more likely to happen.
- Therefore, the PDR is less with single path close to sink than with single path close to sources.

The throughput for these experiments is shown in Figure 6.8 and Figure 6.9.

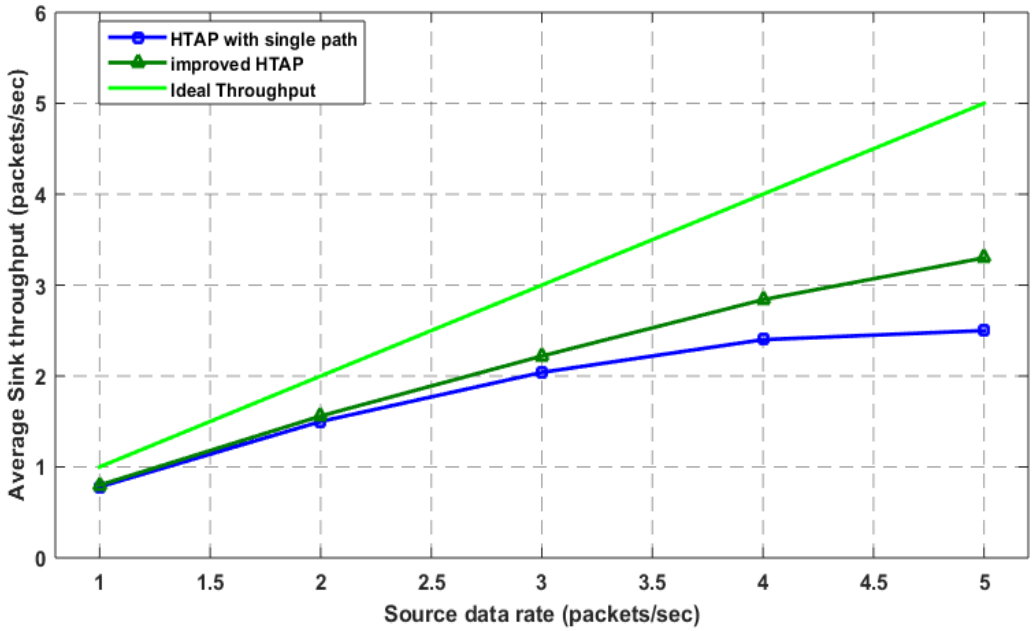


Figure 6- 8: Average sink throughput for a single path close to the sources.

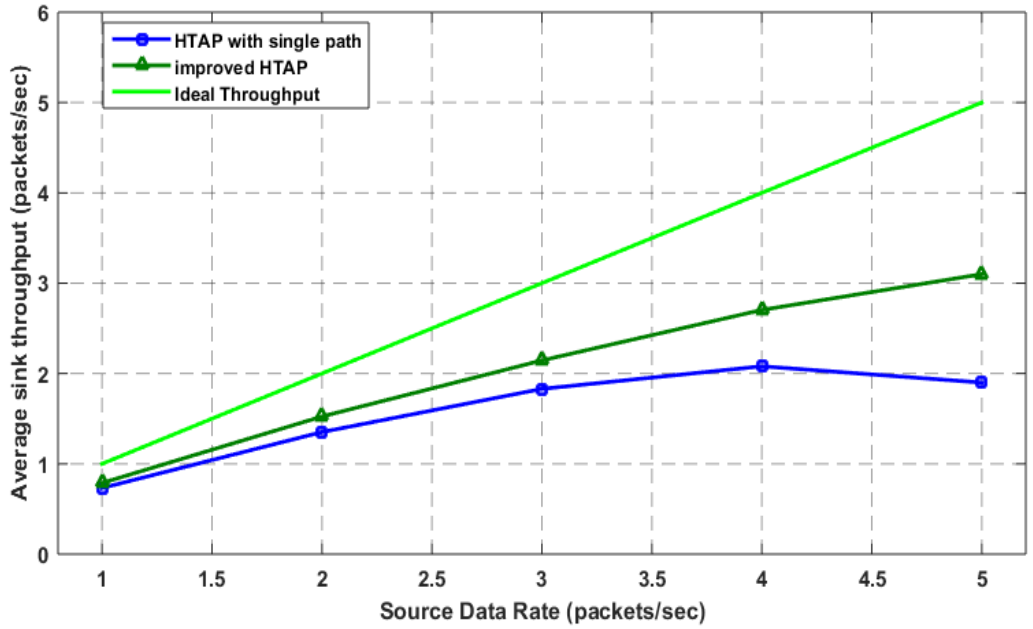


Figure 6- 9: Average sink throughput for a single path close to the sink.

These two figures show that the increase in the data rate affect both algorithms, but in both figures the improved HTAP algorithm has a higher throughput. Additionally, as expected the performance when a single path is close to the sources is better when it is close to the sink.

Figure 6.10 and Figure 6.11 present the End to End delay for both topologies.

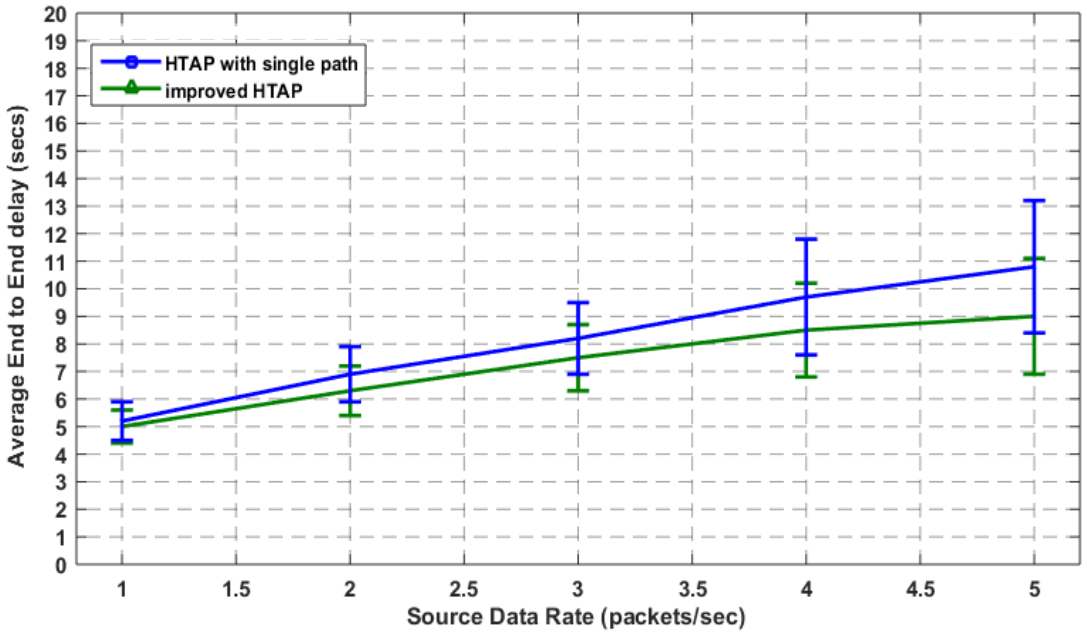


Figure 6- 10: End to End delay for the topology with single path close sources.

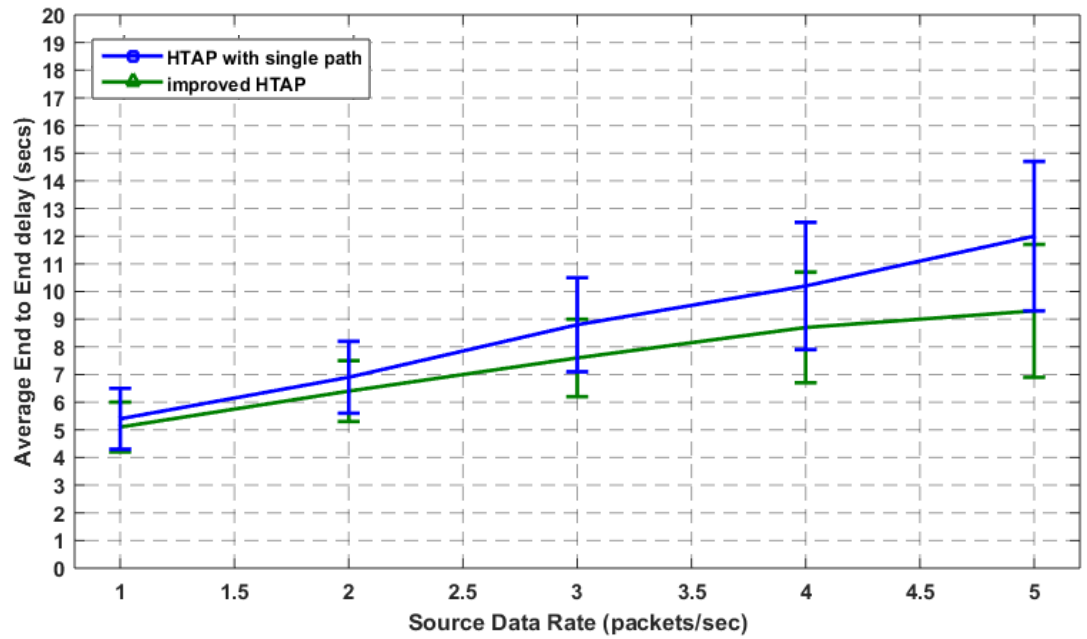


Figure 6- 11: End to End delay for a single path close to the sink.

The results for End to End delay in Figure 6.10 and Figure 6.11 indicate that the improved HTAP algorithm has decreased the delay compared to the normal HTAP algorithm, also the differences between the delay values of these two algorithms increase with the increase in the source data rate, particularly at the highest data rate. These differences are because at high data rate the congestion has higher opportunity to occur and hence more delay faces the packet trip from the source to sink.

This is related to the existence of the single path close to sink and in both algorithms, the congested node of the single path adds more delay to the incoming data that already have delay in their route from the source to the sink. For instance, Node 34 in Figure 6.5 when it's facing congestion in the normal HTAP algorithm, the lower level nodes 31 and 32 will keep sending their data to Node 34 since there is no other path. This leads to packet drops and then retransmission which in turn adds more delayed to already delay packet. Also, for improved HTAP algorithm, when the congestion occurs at Node 34, the nodes 31 and 32 in its lower level will decrease their data rate which means more delay added to the packets from these two nodes to Node 34.

In summary, from the experimental measurements it is clear that presence of a single path in WSN has effects on the network performance. In order to minimize these effects a combination of resource control and traffic rate control has been used. Furthermore, when there is a single path in the network this affects performance with the existence of single path close to the sink having more negative effect on the performance of the network compared to when the single path is close to the sources. This is because the presence of single path close to the sink has high traffic compared to other areas and also the incoming data packets to this area have already been delayed because of the trip from source to sink or may be some of the incoming data have faced congestion in previous hops. There are fewer paths when the network getting closer to sink, so, the network congestion is inevitable. The existing of single path in WSN is considered as one of limiting factors for the running of HTAP algorithm, and the traffic control can work actively in the existing of single path.

In terms of the ratio of PDR values of improved HTAP to normal HTAP, the ratio of PDR values with the existing of single path are increased with the increase in the data rate, the ratio values of PDR for improved HTAP to the PDR of normal HTAP at data rate of 1 packet/sec are (1.02, 1.08) whereas the ratio at data rate of 5 packets/sec are (1.32, 1.63) for single path close to sources and close to sink respectively.

The ratio values of end to end delay for improved HTAP to normal HTAP are decreased with the increase in data rate. The ratio values at low data rate 1packet/sec are (0.94, 0.91), and the values at 5 packets/sec are (0.77, 0.74) for single path closes to sources and closes to sink respectively.

Chapter 7: Conclusions and Future work

The focus of the research in this thesis has been on congestion control in WSNs using resource control and traffic control. The network performance achieved using different methods has been measured experimentally with a network of 40 nodes.

In event driven WSNs, the network load might be unpredictable and highly variable. During periods when the traffic is high and network congestion can occur, sensor nodes that are close to the sink, which receive more data than those close to the sources are more likely to be affected by congestion.

7.1 Node synchronization

In order to measure the end to end delay and to let the nodes enter wake or sleep mode at same time the nodes must be synchronized, the required synchronization accuracy is in tens of microseconds. In a novel method, the synchronization of the node clocks was achieved by two approaches, the relative method and the real-time method.

In the relative method, the nodes are not synchronized to absolute time, but such that the RTC of all nodes have the same clock time. In this method, after starting up the nodes wait for time packets from a timer node and then adjust their RTC (Real Time Clock) to the data time included in the packet. The results show a difference between the RTC on different nodes of approximately 40µsec.

In the real-time method, the synchronization is achieved in 2 phases: an external phase and an internal phase. In the external phase, the reference node synchronizes its RTC by using WiFi-PRO module to an external clock such as UTC. Then, in internal phase, the reference node broadcasts packets containing the real-time value to synchronize all network nodes. Following this process, the clocks on the network nodes are synchronized to within around 5msec of the external clock.

7.2 HTAP algorithm

The performance of the Hierarchical Tree Alternative Path (HTAP) (Sergiou *et al*., 2013) has been measured experimentally and the results compared to network with NCC (No Congestion Control) algorithm (see Chapter 4).

The experimental work tested the schemes that form the HTAP algorithm, these schemes are: Topology control, Hierarchical tree creation, and alternative path.

In Chapter 4, the performance of the HTAP algorithm was compared experimentally with that with NCC for different packet rates, number of nodes, and number of sources. These results are summarised in (Table 7.1).

In Chapter 5, the PDR and throughput of HTAP is studied for different congestion threshold (i.e. 25%, 50%, 75%, and adaptive method).

Also, the PDR and throughput are studied versus the number of network nodes (up to 30 nodes) with fixed number of sources (6 sources), and number of sources (up to 6 sources) with fixed number of nodes (30 nodes), 1 packet/sec the used data rate in these two experiments. The results of HTAP for these metrics are compared with NCC network.

The effect of packet size on network congestion is studied in Chapter 5 as well, the PDR and end to end delay metrics of HTAP are compared with NCC (Table 7.2).

From Table 7.1, the proportion value of PDR for the network with HTAP algorithm to the value of PDR for NCC network is increased with the increase in the data rate.

Table 7. 1: The proportion of PDR and end to end delay for the network with HTAP algorithm and without any congestion control.

		PDR_{HTAP} / PDR_{NCC}	T-E2E $_{HTAP}$ / T-E2E $_{NCC}$
Data rate (packet/sec) For 40 nodes and 8 sources	**1**	1.21	0.77
	5.5	13.8	0.51
no. of nodes for fixed 6 sources	**15**	1.39	
	20	1.30	
	25	1.27	
	30	1.24	
no. of sources for fixed 30 nodes	**1**	1	
	2	1.03	
	3	1.05	
	4	1.11	
	5	1.18	
	6	1.24	

The PDR_{HTAP} / PDR_{NCC} is increased with the increase in the data rate, at 1 packet/sec data rate, the PDR is nearly the same for both HTAP and NCC, and this ratio is increased at data rate of 5.5 packets/sec to over 13. This is an indication that HTAP improves the network performance compared to NCC network. The T-E2E $_{HTAP}$ / T-E2E $_{NCC}$ proportion is also nearly 1 for 1 packet/sec same for both HTAP and NCC at lower data rate, whereas at high data rate of 5.5 packets/sec was decreased, and the T-E2E of HTAP is nearly the half delay of NCC, then this represents an enhancement for HTAP compared to NCC.

Increasing the number of nodes on the network performance has positive effect on both the performance of the network with HTAP and without congestion control, this is because increasing the number of nodes is increasing the number of paths toward the network sink, and hence mitigating the effect of congestion in WSN. Also, from Table 7.1, the proportion of PDR for network with HTAP to NCC network is decreased with the increase number of nodes, this is because increasing the number of paths also enhance the performance of NCC network. PDR decreases with an increase in the number of sources for both HTAP and NCC, However, the HTAP algorithm improves the PDR compared to NCC network.

Table 7.2 presents the effect of congestion threshold value on PDR for two different data rates (1.7 and 5.5 packet/sec).

Table 7. 2: the effects of congestion threshold values and packet size on the PDR and end to end delay for the network with HTAP at different data rates.

		$PDR_{1.7p/s}$ / $PDR_{5.5p/s}$	**End-to-End delay $_{1.7p/s}$ / End-to-End delay $_{5.5p/s}$**
Congestion threshold values	**25%**	1.19	
	50%	1.22	
	75%	1.3	
	Adaptive method	1.17	
Packet size (bytes)	**30**	1.07	0.82
	40	1.08	0.78
	60	1.10	0.73
	80	1.12	0.68
	100	1.17	0.64

The proportion values of PDR for the two different data rates are increased with the increase in congestion threshold and this is a sign that the PDR with high data rate is affected negatively with the increase in the value of congestion threshold. The PDR proportion for the adaptive method shows low value compared to the defined congestion threshold values, and this is an indication that the adaptive method enhanced the PDR behaviour for both data rates.

Also, Table 7.2 shows the effect of packet size on the performance of network with HTAP, it can be seen that the increase in the packet size is increasing the proportion of the PDR with two different data rates (1.7 and 5.5 packet/sec), and this is an indication that the PDR at low data rate (1.7 packet/sec) has better behaviour than it has at high data rate (5.5 packet/sec) for different packet sizes. The effect of packet size on end to end delay is presented also in Table 7.2, increasing the packet size with low data rate has less delay than it has with higher data rate.

Chapter 5 presents the number of possible routes for any sending node, it is concluded that the number of required routes are decreased with the increased in the data rates. From the results of HTAP algorithm, it is found that the nodes closer to sink are more susceptible to congestion since all the nodes paths are narrowing as they reach the sink and hence the data will be concentrated in this area.

7.3 Improved HTAP

In an effort to further improve the performance of the HTAP algorithm, a resource control and traffic control were combined (Chapter 6). Traffic control is employed in HTAP when the congested node only has one path to send its data packets (there is no alternative path). Two topologies were tested, the first has the single path close to source, and the second its single path close to sink.

From table 7.3, the proportion of PDR for both the improved and normal HTAP algorithm increases with increasing data rate. The topology with the single path close to the sink the effect of the improved HTAP has more positive effect compared to normal HTAP algorithm. Also, the existence of the single path close to the source has smaller effect on the network performance compared to the existence of a single path close to the sink.

Table 7. 3: The behaviour of PDR and end to end delay for the improved and normal HTAP algorithm.

		PDR $_{\text{improved HTAP}}$ / PDR $_{\text{HTAP}}$		**End-to-End delay $_{\text{improved HTAP}}$ / End-to-End delay $_{\text{HTAP}}$**	
		Single path closes to source	Single path closes to sink	Single path closes to source	Single path closes to sink
Data rate (packet/sec)	**1**	1.02	1.08	0.94	0.91
	2	1.04	1.12	0.92	0.88
	3	1.09	1.17	0.86	0.84
	4	1.18	1.30	0.85	0.82
	5	1.32	1.63	0.77	0.74

The proportion of end to end delay for both the improved and normal HTAP algorithm is decreased with the increase in the data rate as illustrated in Table 7.3, and this is an indication that the improved HTAP enhanced the network performance with existence of single path for different data rates. In addition, the existence of the single path closes to sink increased the end to end delay compared to the existence of the single path close to source.

7.4 Future work

The following are the potential points of future work:

- Using a 2GB SD card in Waspmote nodes to store the less important packets in the case of congestion. These stored packets can be sent when the network is less busy.
- The metrics studied in this research (i.e. PDR, throughput, and the End to End delay) concentrate on the data part of the network performance. However, there is a need to study the effect of congestion and congestion control on the power usage in WSNs. On full battery charge the required time for each experiment would be around 72 hrs in order to ensure a significant, measurable, power loss.
- When a single path exists in the network topology, the inclusion of mobile nodes to create alternative paths for the transmitted data packets can help alleviate the

congestion in the original path (Madhumathy & Sivakumar, 2014), the effect of mobile speed, this work should be studied.

- Packet priority transmissions, where packets will be sent based on their importance might be used to control congestion, and establishing which are the important packet would be an important aspect of this work.
- The technique of combining resource control with traffic control to mitigate the congestion might be applied in testing and improving a WSN congestion protocol with traffic control only, then improving this protocol by employing the combination of both the resource control and the traffic control.

References

Adams, J. T. J. (2006). An introduction to IEEE STD 802.15.4. *2006 IEEE Aerospace Conference*, 1–8. doi:10.1109/AERO.2006.1655947.

Ahmad, I., Shah, K., Ullah, S., Wali khan University Mardan, A., & Pakhtoonkhwa, K. (2016). Military Applications using Wireless Sensor Networks: A survey. *International Journal of Engineering Science and Computing*, *6*(6), 7039–7043.

Ahmed, N., Salil, S. K., & Jha, S. (2010). Mitigating the effect of interference in Wireless Sensor Networks. *Proceedings - Conference on Local Computer Networks, LCN*, 160–167.

Akan, Ö. B., & Akyildiz, I. F. (2005). Event-to-sink reliable transport in wireless sensor networks. *IEEE/ACM Transactions on Networking*, *13*(5), 1003–1016.

Akyildiz, I., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, *38*, 393–422.

Akyildiz, I., Vuran, M., & Akan, O. (2006). A Cross-Layer Protocol for Wireless Sensor Networks. *2006 40th Annual Conference on Information Sciences and Systems*, 1102–1107.

Alam, M. F., Atiquzzaman, M., & Karim, M. a. (2000). Traffic shaping for MPEG video transmission over the next generation Internet. *Computer Communications*, *23*(14), 1336–1348.

Alazzawi, L., & Elkateeb, a. (2008). Performance Evaluation of the WSN Routing Protocols Scalability. *Journal of Computer Systems, Networks, and Communications*, *2008*, 1–9.

Alduais, N. a. M., Audah, L., Jamil, a., & Abdullah, J. (2016). Study the effect of number of nodes in large-scale Wireless Sensor Networks with design GUI support tools. *ARPN Journal of Engineering and Applied Sciences*, *11*(22).

Ali, A. K., Phillips, I., & Yang, H. (2017). Drop-Burst Length Evaluation of Urban VANETs. *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, *6*(2), 1–6.

Alkhatib, A. a a, Baicher, G. S., & Darwish, W. K. (2013). Wireless Sensor Network-An Advanced Survey. *IJEIT_International Journal of Engineering and Innovative Technology (IJEIT)*, *2*(7), 355–369.

Amir, Y. (2000). Memory Management. Retrieved from http://www.cs.jhu.edu/~yairamir/cs418/os5/sld001.htm [Accessed August 22, 2017].

Antoniou, P., & Pitsillides, A. (2010). A bio-inspired approach for streaming applications in wireless sensor networks based on the Lotka-Volterra competition model. *Computer Communications*, *33*(17), 2039–2047.

Antoniou, P., Pitsillides, A., Blackwell, T., Engelbrecht, A., & Michael, L. (2013). Congestion control in wireless sensor networks based on bird flocking behavior. *Computer Networks*, *57*(5), 1167–1191.

Arzubi, a A., Lechtaler, a C., Foti, a, & Fusario, R. (2013). Analysis of Radio Communication Solutions in Small and Isolated Communities under the IEEE 802 . 22 Standard, (October), 1107–1118.

Banerji, S., & Chowdhury, R. S. (2013). Wi-Fi \& WiMAX : A Comparative Study. *Indian Jounal of Engineering*, *2*(5), 1–5.

Benzaïd, C., Bagaa, M., & Younis, M. (2017). Efficient clock synchronization for clustered wireless sensor networks. *Ad Hoc Networks*, *56*, 13–27.

Borges, L., Velez, F., & Lebres, A. (2014). Survey on the Characterization and Classification of Wireless Sensor Networks Applications. *IEEE Communications Surveys & Tutorials*, *XX*(X), 1–1.

Bouabdellah, K., Noureddine, H., & Larbi, S. (2013). Using Wireless Sensor Networks for Reliable Forest Fires Detection. *Procedia Computer Science*, *19*(Seit), 794–801.

Buratti, C., Conti, A., Dardari, D., & Verdone, R. (2009). An overview on wireless sensor networks technology and evolution. *Sensors*, *9*, 6869–6896.

Chatterjee, A., & Pandey, M. (2014). Practical Applications Of Wireless Sensor Network Based On Military, Environmental, Health And Home Applications: A Survey. *International Journal of Scientific & Engineering Research*, *5*(1), 1043–1050.

Chatterjee, M., Kwiat, K., & Brahma, S. (2010). Congestion Control and Fairness in Wireless Sensor Networks, 8th IEEE International conference on Pervasive Computing and Communcations, 413–418.

Chen, S., & Yang, N. (2006). Congestion Avoidance based on light weight buffer management in Sensor Networks. *IEEE Transaction*, *17*, 934–946.

Damuut, L. P. (2012). A Survey of Deterministic Vs . Non-Deterministic Node Placement Schemes in WSNs. *The Sixth International Conference on Sensor Technologies and Applications*, (c), 154–158.

Das, A., & Dutta, D. (2005). Data acquisition in multiple-sink sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, *9*(3), 82.

Digi International. (2009). XBee ® /XBee-PRO ® RF Modules. *Product Manual v1.xEx-802.15.4 Protocol*, 1–69. Retrieved from http://www.digi.com/products/wireless-wired-embedded-solutions/ [Accessed 4 September 2017].

E., R. M., & Terzis, A. (2008). Minimising the effect of WiFi interference in 802.15.4 wireless sensor networks. *International Journal of Sensor Networks*, *3*(1), 43.

Ee, C. T., & Bajcsy, R. (2004). Congestion Control and Fairness for Many-to-One Routing in Sensor Networks. *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems - SenSys '04*, 148.

Elson, J., Girod, L., & Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, *36*(SI), 147.

Enigo, V. S. F. (2009). An Energy Efficient Congestion Control Protocol for Wireless Sensor Networks. *Computing*, March 2009, 6–7.

Fang, W., Chen, J., Shu, L., Chu, T., & Qian, D. (2010). Congestion avoidance, detection and alleviation in wireless sensor networks. *Journal of Zhejiang University SCIENCE C*, *11*(1), 63–73.

Ferres, L. (2010). Memory management in C: The heap and the stack. *Department of Computer Science, Universidad de Concepcion*, 1–7.

Flora, D. F. J., Kavitha, V., & Muthuselvi, M. (2011). A survey on congestion control techniques in Wireless Sensor Networks. *2011 International Conference on Emerging Trends in Electrical and Computer Technology*, 1146–1149.

Ghaffari, A. (2015). Congestion control mechanisms in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, *52*, 101–115.

Giancoli, E., Jabour, F., & Pedroza, A. (2008). Collaborative Transport Control Protocol for Sensor Networks. *2008 International Conference on Computer and Electrical Engineering*, 373–377.

Graphs -Coursehero, W. (2017). Chapter 7 Weighted Graphs. Available at: https://www.coursehero.com/file/17378075/Chapter-7-Weighted-Graphs/ [Accessed 5 October 2017].

He, J., Duan, X., Cheng, P., Shi, L., & Cai, L. (2017). Accurate clock synchronization in wireless sensor networks with bounded noise. *Automatica*, *81*, 350–358.

He, T., Ren, F., Lin, C., & Das, S. (2008). Alleviating congestion using traffic-aware dynamic routing in wireless sensor networks. *2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON*, 233–241.

Healy, M., Newe, T., & Lewis, E. (2008). Wireless sensor node hardware: A review. *Proceedings of IEEE Sensors*, 621–624.

Huang, G., Zomaya, A. Y., Delicato, F. C., & Pires, P. F. (2012). An accurate on-demand time synchronization protocol for wireless sensor networks. *Journal of Parallel and Distributed Computing*, *72*(10), 1332–1346.

Huang, J. M., Li, C. Y., & Chen, K. H. (2009). TALONet: A power-efficient grid-based congestion avoidance scheme using multi-detouring technique in wireless sensor networks. *2009 Wireless Telecommunications Symposium, WTS 2009*, (Llc), 1–6.

Hull, B., Jamieson, K., & Balakrishnan, H. (2004). Mitigating congestion in wireless sensor networks. *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems - SenSys '04*, 134.

Iyer, Y. G., Gandham, S., & Venkatesan, S. (2005). STCP: a generic transport layer protocol for wireless sensor networks. *Proceedings of 14th International Conference on Computer Communications and Networks, (ICCCN'05)*, 449–454.

Jacquet, P., Mühlethaler, P., Clausen, T. H., Laouiti, a., Qayyum, a., & Viennot, L. (2001). Optimized link state routing protocol for ad hoc networks. *Proceedings of the IEEE International Multi Topic Conferences*, 62–68.

Jenn-Yue Teo, Yajun Ha, & Chen-Khong Tham. (2008). Interference-Minimized Multipath Routing with Congestion Control in Wireless Sensor Network for High-Rate Streaming. *IEEE Transactions on Mobile Computing*, *7*(9), 1124–1137.

Johnson, M., Healy, M., Van De Ven, P., Hayes, M. J., Nelson, J., Newe, T., & Lewis, E. (2009). A comparative review of wireless sensor network mote technologies. *Proceedings of IEEE Sensors*, 1439–1442.

Kafi, M. A., Djenouri, D., Ben-Othman, J., & Badache, N. (2014). Congestion control protocols in wireless sensor networks: a survey. *Communications Surveys & Tutorials, IEEE*, *16*(3), 1369–1390.

Kang, J., Zhang, Y., & Nath, B. (2007). TARA: Topology-aware resource adaptation to alleviate congestion in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, *18*(7), 919–931.

Karenos, K., Kalogeraki, V., & Krishnamurthy, S. V. (2007). Cluster – based Congestion Control for Sensor. *Engineering*, *V*(November), 1–31.

Karp, B., & Kung, H. (2000). GPSR: Greedy Perimeter Stateless Routing for wireless networks. *ACM MobiCom*, (MobiCom), 243–254.

Kazemian, H. B. (2009). An intelligent video streaming technique in ZigBee wireless. *IEEE International Conference on Fuzzy Systems*, 121–126.

Khakestani, F., & Balochian, S. (2015). A Survey of Military Application of Wireless Sensor Networks For Soldiers. *International Journal Of Engineering And Computer Science*, *4*(4), 13205–13210.

Khan, M. I., Gansterer, W. N., & Haring, G. (2013). Static vs. mobile sink: The influence of basic parameters on energy efficiency in wireless sensor networks. *Computer Communications*, *36*(9), 965–978.

Kumar, A., Shwe, H. Y., Wong, K. J., & Chong, P. H. J. (2017). Location-Based Routing Protocols for Wireless Sensor Networks: A Survey. *Wireless Sensor Network*, *09*(01), 25–72.

Kumar, R., Crepaldi, R., Rowaihy, H., Harris, A. F., Cao, G., Zorzi, M., & La Porta, T. F. (2008). Mitigating performance degradation in congested sensor networks. *IEEE Transactions on Mobile Computing*, *7*(6), 682–696.

Last, B. E., & Edt, P. M. (2016). Memories of an Arduino. Available at: https://learn.adafruit.com/memories-of-an-arduino?view=all [Accessed on 12 September 2017].

Lee, D., & Chung, K. (2010). Adaptive duty-cycle based congestion control for home automation networks. *IEEE Transactions on Consumer Electronics*, *56*(1), 42–47.

Leghari, M., Abbasi, S., & Dhomeja, L. (2013). Survey on Packet Size Optimization Techniques in Wireless Sensor Networks. *Icwsn*, 1–8.

Li, K., Nikolaidis, I., & Harms, J. (2013). The analysis of the additive-increase multiplicative-decrease MAC protocol. *2013 10th Annual Conference on Wireless On-Demand Network Systems and Services, WONS 2013*, 122–124.

Li, M., Li, Z., & Vasilakos, A. V. (2013). A survey on topology control in wireless sensor networks: Taxonomy, comparative study, and open issues. *Proceedings of the IEEE*, *101*(12), 2538–2557.

Li, N., Hou, J., & Sha, L. (2003). Design and analysis of an MST-based topology control algorithm. *IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the*

IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), *3*(C), 1702–1712.

Libelium. (2013). Waspmote Data Frame, 18. Available at: http://www.libelium.com/downloads/documentation/data_frame_guide.pdf [Accessed 19 August 2017].

Libelium. (2018). Waspmote Technical Guide Waspmote. Available at: http://www.libelium.com/downloads/documentation/waspmote technical guide.pdf [Accessed 24 Feb. 2018].

Lie, A., & Klaue, J. (2008). Evalvid-RA: Trace driven simulation of rate adaptive MPEG-4 VBR video. *Multimedia Systems*, *14*(1), 33–50.

M.A. Matin and M.M. Islam. (2012). Overview of Wireless Sensor Network. *Wireless Sensor Networks - Technology and Protocols*, 320.

Madhumathy, P., & Sivakumar, D. (2014). Mobile Sink Based Reliable And Energy Efficient Data Gathering Technique For Wsn, *61*(1), 1–9.

Mahapatro, A., & Khilar, P. M. (2011). SDDP : Scalable Distributed Diagnosis Protocol for Wireless Sensor Networks. In Aluru S. et al. (eds) Contemporary Coputing. IC3 2011. Communications in Computer and Information Science, vol 168. Springer, Berlin Heidelberg, 1–12.

Mahmood, M. A., & Seah, W. (2015). Reliability in Wireless Sensor Networks : Survey and Challenges Ahead. Computer Networks. vol 79. 166-187. *Elsevier*.

Marina, M. K., & Das, S. R. (2001). On-demand Multipath Distance Vector Routing in Ad hoc Networks. *Proceedings of the 9th IEEE International Conference on Network Protocols*, 14–23.

Maxim Integrated. (2015). DS 3231 RTC General Description. *Data Sheet*, 20. Retrieved from https://datasheets.maximintegrated.com/en/ds/DS3231.pdf [Accessed 17 September 2017].

Micek, J., & Karpis, O. (2010). Wireless sensor networks for road traffic monitoring. *Komunikacie*, *12*(3 A), 80–85. Available at: http://www.scopus. com/inward/ record. [Accessed 13 Jan. 2018].

Michopoulos, V. (2012). Congestion and Medium Access Control in 6LoWPAN WSN Doctor of Philosophy. Thesis. University of Loughbourgh.

Mills, D. L. (1989). Internet time synchronization the Network Time Protocol, *39*(October), 1482–1493.

Okediran, O. O. (2014). Design and Development of a Security Surveillance System based on Wireless Sensor Network, *1*(4), 283–291.

Paek, J., & Govindan, R. (2010). Rcrt. *ACM Transactions on Sensor Networks*, *7*(3),1–45.

Perillo, M., & Heinzelman, W. (2004). DAPR: a protocol for wireless sensor networks utilizing an application-based routing cost. *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*, *3*, 1540–1545.

Piyare, R., Lee, S., & Korea, S. (2013). Performance Analysis of XBee ZB Module Based Wireless Sensor Networks. *International Journal of Scientific & Engineering Research*, *4*(4), 1615–1621.

Prim, R. C. (1957). Shortest Connection Networks And Some Generalizations. *Bell System Technical Journal*, *36*(6), 1389–1401.

Pule, M., Yahya, A., & Chuma, J. (2018). Wireless sensor networks: A survey on monitoring water quality. *Journal of Applied Research and Technology*, *15*(6), 562–570.

Rafkind, J., Wick, A., Regehr, J., & Flatt, M. (2009). Precise garbage collection for C. *Proceedings of the 2009 International Symposium on Memory Management - ISMM '09*, 39.

Rahman, O., Monowar, M. M., & Hong, C. S. (2008). A QoS Adaptive Congestion Control Sensor Network in Wireless Ficl whre notrion Alof rioute sensorIN, 941–946.

Rangwala, S., Gummadi, R., Govindan, R., & Psounis, K. (2006). Interference-aware fair rate control in wireless sensor networks. *ACM SIGCOMM Computer Communication Review*, *36*(4), 63.

Rezaee, A. A., Yaghmaee, M. H., Rahmani, A. M., & Mohajerzadeh, A. H. (2014). HOCA: Healthcare aware optimized congestion avoidance and control protocol for wireless sensor networks. *Journal of Network and Computer Applications*, *37*, 216–228.

Rivero-Angeles, M. E., & Bouabdallah, N. (2009). Event reporting on continuous monitoring wireless sensor networks. *2009 IEEE Global Telecommunications Conference, GLOBECOM 2009*.

Roseline, R. a, Devapriya, M., & Sumathi, P. (2013). Pollution Monitoring using Sensors and Wireless Sensor Networks : A Survey, *2*(7), 119–124.

S, K. J., Viswanatha Rao, S., & Pillai, S. S. (2015). Impact of IEEE 802.11 MAC Packet Size on Performance of Wireless Sensor Networks. *IOSR Journal of Electronics and Communication Engineering Ver. IV*, *10*(3), 2278–2834.

Sankarasubramaniam, Y., Akan, Ö. B., & Akyildiz, I. F. (2003). ESRT : Event-to-Sink Reliable Transport in Wireless Sensor. *MobiHoc*, 177–188.

Sankarasubramaniam, Y., Akyildiz, I. F., & McLaughlin, S. W. (2003). Energy efficiency based packet size optimization in wireless sensor networks. *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, 1–8.

Schenato, L., & Fiorentin, F. (2011). Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica*, *47*(9), 1878–1886.

Sergiou, C. (2012). *Performance-Aware Congestion Control in Wireless Sensor Network Using Resource Control*. Thesis. University of Cyprus.

Sergiou, C., Vassiliou, V., & Paphitis, A. (2013). Hierarchical Tree Alternative Path (HTAP) algorithm for congestion control in wireless sensor networks. *Ad Hoc Networks*, *11*(1), 257–272.

Sohraby, K., & Lawrence, V. (2006). Priority-based Congestion Control in Wireless Sensor Networks. *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC'06)*, *1*, 22–31.

Spring, T. (2017). A Programming Language for the Internet of Things. Thesis. University of Oslo.

Suriyachai, P., Roedig, U., & Scott, A. (2012). A Survey of MAC Protocols for Mission-Critical Applications in Wireless Sensor Networks, *14*(2), 240–264.

Swain, A. R., & Hansdah, R. C. (2011). A Weighted Average Based External Clock Synchronization Protocol for Wireless Sensor Networks. *2011 31st International Conference on Distributed Computing Systems Workshops*, 218–229.

Tang, S., & Zhang, B. (2004). A robust AODV protocol with local update. *APCC/MDMC '04. The 2004 Joint Conference of the 10th Asia-Pacific Conference on Communications and the 5th International Symposium on Multi-Dimensional Mobile Communications Proceeding*, (January), 418–422.

Tao, L. Q., & Yu, F. Q. (2010). ECODA: Enhanced congestion detection and avoidance for multiple class of traffic in sensor networks. *IEEE Transactions on Consumer Electronics*, *56*(3), 1387–1394.

Tezcan, N., & Wang, W. (2007). ART : An Asymmetric and Reliable Transport Mechanism for Wireless Sensor Networks. *International Journal of Sensor Networks*, *2*(3-4), 188–200.

Tian He, Stankovic, J. a., Chenyang Lu, & Abdelzaher, T. (2003). SPEED: a stateless protocol for real-time communication in sensor networks. *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, 46–55.

Tiwari, V., Misra, S., & Obaidat, M. (2009). Lacas: Learning automata-based congestion avoidance scheme for healthcare wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, *27*(4), 466–479.

Uplap, P., & Sharma, P. (2014). Review of Heterogeneous/Homogeneous Wireless Sensor Networks and Intrusion Detection System Techniques.

Van Greunen, J., & Rabaey, J. (2003). Lightweight time synchronization for sensor networks. *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications - WSNA '03*, 11.

Vedantham, R., Sivakumar, R., & Park, S. J. (2007). Sink-to-sensors congestion control. *Ad Hoc Networks*, *5*(4), 462–485.

Villalba, L. J. G., Orozco, A. L. S., Cabrera, A. T., & Abbas, C. J. B. (2009). Routing protocols in wireless sensor networks. *Sensors (Basel, Switzerland)*, *9*(2), 8399–421.

Vuran, M. C., Akyildiz, I. F., & Fellow, I. F. A. (2010). XLP : A Cross-Layer Protocol for Efficient Communication in Wireless Sensor Networks XLP : A Cross-Layer Protocol for Efficient Communication in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing vol. 9. issue 11*.

Wan, C., Eisenman, S. B., Campbell, A. T., & Crowcroft, J. (2005). Siphon: Overload Traffic Management using Multi-Radio Virtual Sinks in Sensor Networks. *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems - SenSys '05*, *3*, 116.

Wan, C., Eisenman, S., & Campbell, A. (2003). CODA: congestion detection and avoidance in sensor networks. *2nd International Conference on Embedded Networked Sensor …*, 266–279.

Wan, J., Xu, X., Feng, R., & Wu, Y. (2009). Cross-layer active predictive congestion control protocol for wireless sensor networks. *Sensors*, *9*(10), 8278–8310.

Wang, C., & Li, B. (2007). Upstream congestion control in wireless sensor networks through cross-layer optimization. *Selected Areas*, *25*(4), 786–795.

Woo, A., & Culler, D. E. (2001). A Transmission Control Scheme for Media Access in Sensor Networks. *Acm Sigmobile*, 1–15.

Xia, F. (2009). Wireless sensor technologies and applications. *Sensors*, *9*(11), 8824–8830.

Yaakob, N., Khalil, I., & Hu, J. (2010). Performance Analysis of Optimal Packet Size for Congestion Control in Wireless Sensor Networks. *2010 Ninth IEEE International Symposium on Network Computing and Applications*, 210–213.

Yaghmaee, M. H., & Adjeroh, D. (2008). A New Priority Based Congestion Control Protocol for Wireless Multimedia Sensor Networks Lane Department of Computer Science and Electrical Engineering ,.

Yang, J., Portilla, J., & Riesgo, T. (2012). Smart parking service based on Wireless Sensor Networks. *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 6029–6034.

Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks*, *52*, 2292–2330.

Yin, X., Zhou, X., Huang, R., Fang, Y., & Li, S. (2009). A fairness-aware congestion control scheme in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, *58*(9), 5225–5234.

Yuan, H., Yugang, N., & Fenghao, G. (2014). Congestion control for wireless sensor networks: A survey. *The 26th Chinese Control and Decision Conference (2014 CCDC)*, 4853–4858.

Zareei, M., Zarei, A., Budiarto, R., & Omar, M. A. (2011). A Comparative Study of Short Range Wireless Sensor Network on High Density Networks. *17th Asia-Pacific Conference on Communications (APCC)*, (October), 247–252.

Zhang, M., Cai, W., & Zhou, L. (2012). Hop-to-hop congestion feedback mechanism for sink bottleneck problem in WSNs. *Proceedings - 5th International Conference on Intelligent Networks and Intelligent Systems, ICINIS 2012*, 142–145.

Zhao, F., & Guibas, L. J. (2004). Wireless sensor networks: an information processing approach, 358.

Zheng, J. (2009). Network architectures and protocol stack, Wireless Sensor Networks: A Networking Perspective. Wiley Online Library. 19–33.