Routing Protocols for Next Generation Mobile Wireless Sensor Networks

T. Hayes

A Thesis Submitted for the Degree of Doctor of Philosophy

School of Engineering and Informatics

University of Sussex

February 2016

DECLARATION

I hereby declare that this thesis has not been and will not be submitted, in whole or in part, to another University for the award of any other degree and that the work presented here is my own unless stated otherwise.

Signature: ……

T. Hayes

Date: __/__/__

University of Sussex

Thesis submitted in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

Routing Protocols for Next Generation Mobile Wireless Sensor Networks

T. Hayes

SUMMARY

The recent research interest in wireless sensor networks has caused the development of many new applications and subsequently, these emerging applications have ever increasing requirements. One such requirement is that of mobility, which has inspired an entirely new array of applications in the form of mobile wireless sensor networks (MWSNs). In terms of communications, MWSNs present a challenging environment due to the high rate at which the topology may be changing. As such, the motivation of this work is to investigate potential communications solutions, in order to satisfy the performance demands of new and future MWSN applications. As such this work begins by characterising and evaluating the requirement of a large variety of these emerging applications.

This thesis focuses on the area of routing, which is concerned with the reliable and timely delivery of data from multiple, mobile sensor nodes to a data sink. For this purpose the technique of gradient routing was identified as a suitable solution, since data can quickly be passed down a known gradient that is anchored at the sink. However, in a mobile network, keeping the gradient up-to-date is a key issue. This work proposes the novel use of a global time division multiple access (GTDMA) MAC as a solution to this problem, which mitigates the need for regularly flooding the network. Additionally, the concept of blind forwarding is utilised for its low overhead and high reliability through its inherent route diversity.

The key contribution of this thesis is in three novel routing protocols, which use the afore mentioned principles. The first protocol, PHASeR, uses a hop-count metric and encapsulates data from multiple nodes in its packets. The hop-count metric was chosen because it is simple and requires no additional hardware. The inclusion of encapsulation is intended to enable the protocol to cope with network congestion. The second protocol, LASeR, utilises location awareness to maintain a gradient and performs no encapsulation. Since many applications require location awareness, the communications systems may also take advantage of this readily available information and it can be used as a gradient metric. This protocol uses no encapsulation in order to reduce delay times. The third protocol, RASeR, uses the hop-count metric as a gradient and also does not perform encapsulation. The reduced delay time and the relaxed requirement for any existing method of location awareness makes this the most widely applicable of the three protocols. In addition to analytical expressions being derived, all three protocols are thoroughly tested through simulation. Results show the protocols to improve on the state-of-the-art and yield excellent performance over varying speeds, node numbers and data generation rates. LASeR shows the lowest overhead and delay, which comes from the advantage of having available location information. Alternatively, at the expense of increased overhead, RASeR gives comparatively high performance metrics without the need for location information.

Overall, RASeR can be suitably deployed in the widest range of applications, which is taken further by including four additional modes of operation. These include a supersede mode for applications in which the timely delivery of the most recent data is prioritised. A reverse flooding mechanism, to enable the sink to broadcast control messages to the sensor nodes. An energy saving mode, which uses sleep cycles to reduce the networks power consumption, and finally a pseudo acknowledgement scheme to increase the reliability of the protocol. These additions enable RASeR to satisfy the needs of some of the most demanding MWSN applications.

In order to assess the practicality of implementation, RASeR was also evaluated using a small testbed of mobile nodes. The successful results display the protocols feasibility to be implemented on commercially available hardware and its potential to be deployed in the real world. Furthermore, a key issue in the real world deployment of networks, is security and for this reason a fourth routing protocol was designed called RASeR-S. RASeR-S is based on RASeR, but introduces the use of

encryption and suggests a security framework that should be followed in order to significantly reduce the possibility of a security threat.

Whilst the main focus of this work is routing, alternative MAC layers are assessed for LASeR. Unlike the other two protocols, LASeR uses available location information to determine its gradient and as such, it is not reliant on the GTDMA MAC. For this reason several MAC layers are tested and the novel idea of dedicated sensing slots is introduced, as well as a network division multiple access scheme. The selected and proposed MACs are simulated and the GTDMA and two proposed protocols are shown to give the best results in certain scenarios.

This work demonstrates the high levels of performance that can be achieved using gradient orientated routing in a mobile network. It has also shown that the use of a GTDMA MAC is an efficient solution to the gradient maintenance problem. The high impact of this work comes from the versatility and reliability of the presented routing protocols, which means they are able to meet the requirements of a large number of MWSN applications. Additionally, given the importance of security, RASeR-S has been designed to provide a secure and adaptable routing solution for vulnerable or sensitive applications.

ACKNOWLEDGEMENTS

I would firstly like to thank my family and especially parents for their constant support and unfaltering belief in me. I also wish to thank Paloma for taking care of me and sharing her persistent work ethic, which continues to encourage me.

I would like to extend significant gratitude to my supervisor, Dr Falah Ali, without whom I would not have had the opportunity to undertake this challenge. I also want to mention my lab mates, James, Fei, Murtala, Ibrahim and Andreas, who have all helped me along the way.

I also wish to acknowledge the university and the EPSRC, who funded this work (grant num. EP/K503198/1).

PUBLICATIONS

T. Hayes and F. H. Ali, "Mobile Wireless Sensor Networks: Applications and Routing Protocols," *Handbook of Research on Next Generation Mobile Communication Systems*, IGI Global, pp. 256-292, Sept. 2015.

T. Hayes and F. H. Ali, "Proactive Highly Ambulatory Sensor Routing (PHASeR) Protocol for Mobile Wireless Sensor Networks," *Elsevier Pervasive and Mobile Computing*, vol. 21, pp. 47-61, Aug. 2015.

T. Hayes and F. H. Ali, "Location Aware Sensor Routing (LASeR) Protocol for Mobile Wireless Sensor Networks," *IET Wireless Sensor Systems*, vol. 6, no. 2, pp. 49-57, April 2016.

T. Hayes and F. H. Ali, "Robust Ad-hoc Sensor Routing (RASeR) Protocol for Mobile Wireless Sensor Networks," *Elsevier Ad Hoc Networks*, vol. 50, pp. 128-144, Nov. 2016.

T. Hayes and F. H. Ali, "Medium Access Control Schemes for Flat Mobile Wireless Sensor Networks," *Submitted to IET Wireless Sensor Systems 27/7/16.*

T. Hayes and F. H. Ali, "Secure Robust Ad-hoc Sensor Routing (RASeR-S) for Mobile Wireless Sensor Networks," *To be submitted to IEEE Transactions on Mobile Computing.*

T. Hayes and F. H. Ali, "Testbed Implementation of the RASeR MWSN Routing Protocol," *To be submitted to IEEE Int'l Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC).*

CONTENTS

ABBREVIATIONS

ACK	:	Acknowledgement
AODV	:	Ad-hoc On-demand Distance Vector
AWGN	:	Additive White Gaussian Noise
BER	:	Bit Error Rate
CA	:	Collision Avoidance
CBC	:	Cipher Block Chaining
CCA	:	Clear Channel Assessment
CFB	:	Cipher Feedback
COTS	:	Commercial-Off-The-Shelf
CRC	:	Cyclic Redundancy Check
CSMA	:	Carrier Sense Multiple Access
CSS	:	Channel Sensing Slot
CTR	:	Counter
CTS	:	Clear to Send
DCF	:	Distributed Coordination Function
DoS	:	Denial of Service
DS	:	Dedicated CSS Slots
DTN	:	Delay Tolerant Network
ECB	:	Electronic Codebook
EE	:	Enter Executives
FCFS	:	First-Come First Serve
FEC	:	Forward Error Correction
GPS	:	Global Positioning System
GTDMA	:	Global Time Division Multiple Access
IO	:	Input/Output
ISM	:	Industrial, Scientific and Medical
IV	:	Initialisation Vector
LASeR	:	Location Aware Sensor Routing
MAC	:	Medium Access Control
MACRO	:	Mobility Adaptive Cross-layer Routing
MANET	:	Mobile Ad-hoc Network
MWSN	:	Mobile Wireless Sensor Network
NDMA	:	Network Division Multiple Access
OFB	:	Output Feedback
OLSR	:	Optimised Link State Routing
OPNET	:	Optimised Network Engineering Tool
OQPSK	:	Offset Quadrature Phase Shift Keying
OSI	:	Open Systems Interconnection
OTAP	:	Over-the-Air Programming
pACK	:	Pseudo Acknowledgements
PCBC	:	Propagating Cipher Block Chaining
PDR	:	Packet Delivery Ratio
PHASeR	:	Proactive Highly Ambulatory Sensor Routing

PHY	:	Physical Layer
PLR	:	Packet Loss Ratio
RASeR	:	Robust Ad-hoc Sensor Routing
RASeR-S	:	RASeR - Secure
RC5	:	Rivest Cipher 5
RSSI	:	Received Signal Strength Indicator
RTS	:	Request to Send
SA	:	Slotted ALOHA
SAR	:	Search and Rescue
SNR	:	Signal to Noise Ratio
TDMA	:	Time Division Multiple Access
TI	:	Texas Instruments
TTL	:	Time To Live
UAV	:	Unmanned Air Vehicles
UGV	:	Unmanned Ground Vehicles
USV	:	Unmanned Underwater Vehicles
WSN	:	Wireless Sensor Network
XE	:	Exit Executives

FIGURE LIST

TABLE LIST

CHAPTER 1

INTRODUCTION

The introduction of small, low cost sensor motes, with the ability to communicate wirelessly has inspired a vast array of applications [1.1]. These wireless sensor networks (WSNs) have shown a lot of potential for large-scale data acquisition. However, it's only within the last decade that mobility has been used in conjunction with the traditional static WSN. This new paradigm of mobile wireless sensor networks (MWSNs) has enabled applications to be conceived that hadn't been considered before and will be a key enabling technology in the future of ubiquitous sensing [1.2].

The wide applicability of sensor networks in areas such as safety, research and military have made them a popular topic in the research community. Their ability to sense phenomena without human presence, in potential harsh or hostile environments, make them an invaluable resource. Research topics such as MAC (medium access control) protocols, localisation techniques, synchronisation methods and routing protocols have all been studied in some detail within the scope of static sensor networks. However, the addition of mobility gives rise to new constraints and challenges, which calls for novel approaches to these problems.

1.1 RESEARCH MOTIVATIONS

This work is motivated by the vast number of potential and emerging MWSN applications, each of which have their own individual requirements. These applications range from extra-terrestrial terrain mapping to search and rescue drones and as such the use of a reliable routing protocol could enhance scientific study or even save lives.

This unique style of network presents a challenging environment in which to provide a routing solution. The potentially high levels of mobility make determining a path to the sink an ongoing challenge. Some applications may also have strict resource limitations, which also need to be considered. Additionally, the application requirements may set high performance targets in terms of reliability and delay in data delivery.

1.2 Aims and Objectives

The primary aim of this thesis is to develop routing solutions that are able to deliver data from many sensors to a single sink in highly challenging scenarios. The protocols should be able to adapt to fast and frequent changes in topology, high traffic levels and be scalable to the requirements of various applications. Additionally, a high level of performance should be maintained in terms of successful packet delivery and low latency.

Whilst the highly diverse and changing applications for this technology means that there is never going to be a single solution that will satisfy the requirements of every MWSN application, the solutions presented here are designed to meet the needs of a high number of applications and scenarios as well as allow for future advances in the field.

These aims will be achieved by designing novel routing protocols and evaluating them using mathematical analysis, extensive simulation and testbed implementations. In this way the suggested solutions can be investigated and comparisons drawn against existing protocols. More importantly, the performance levels of the protocols can be determined, which will clearly indicate whether they are fit for purpose. Additionally, a testbed implementation will serve as a proof of concept, illustrating the feasibility of a real world deployment.

1.3 Contributions

The main contributions of this thesis to the area of communications in MWSNs are outlined in the following points.

Firstly, the novel routing protocols presented here are based on the concept of gradient routing. Previously, this technique had not been used in the case of a MWSN without location awareness, mostly because of the gradient maintenance problem. This is the problem that, in a mobile environment, the gradient value held by each node may change and the network will require some method of keeping the gradient metric up-to-date with the varying topology. So, in order to utilise the advantages of gradient routing, the gradient metric used would need to be reliably maintained across all the nodes in the network. For this reason, the novel use of a global time division multiple access (GTDMA) MAC protocol is proposed, which allows nodes to keep track of their position in the network and maintain the gradient metric. The advantages of this particular technique is that it's low in overhead and can cope with very high levels of mobility.

In a mobile network it is important that the protocols are adaptive to change, and for this reason the use of blind forwarding is suggested. Blind forwarding dictates that the decision to pass on a packet is made by the receiving node. In other words, the transmitting node broadcasts the packet to all of its neighbours. The neighbours then determine whether they are closer to the

destination than the transmitting node. If so, then they may decide to forward the packet. This technique has a few major advantages in that it requires no overhead or handshakes to pass on a message, it is highly adaptable to frequently varying topologies and it inherently creates multiple routes for packets, which improves the protocols reliability.

The culmination of this work uniquely combines the solution to the gradient maintenance problem and the use of blind forwarding to create two novel routing protocols, PHASeR and RASeR, which are the first ever protocols to use a gradient based routing technique using a hop count metric in a MWSN. Mathematical expressions were derived in order to describe the characteristics of the proposed and simulation models were all coded, debugged and validated in a network simulation package. Subsequently, this innovative approach to routing is validated through extensive modelling, simulation and mathematical analysis. RASeR is also used to evaluate the effect of a fading channel on the protocols performance through further simulation. Additionally, RASeR is deployed on a testbed, which is used to evaluate the feasibility of a real-world deployment.

RASeR is also provided with energy saving features and a method of sending messages up-stream from the sink to the sensors, in order to satisfy the requirements of potential applications. The protocol is further evaluated in a testbed experiment and its use assessed in terms of the application requirements of UAV aided search and rescue, which is a potential future use of this technology. In addition to this, an improvement to the protocols reliability is also proposed and simulated. This takes the form of a pseudo acknowledgement, which allows nodes to verify the successful reception of a transmitted packet without any additional overhead.

A third routing protocol, LASeR, is also presented here, which uses blind forwarding but adopts location awareness in order to maintain the gradient metric. This alleviates the burden of maintaining the gradient metric from the communications system, which in turn allows alternative MAC layers to be considered. As alternatives, a location aware MAC is suggested and the novel concept of dedicated channel assessment slots are proposed. The potential MACs are assessed by thorough simulation and evaluated based on their performance when used with LASeR.

Furthermore, in a variety of scenarios, all three protocols, PHASeR, LASeR and RASeR, are compared with each other as well as an up-to-date state-of-the-art MWSN routing protocol from the literature.

A common concern in communications systems is security, which has also been addressed in this work. RASeR-S is an adaptation of RASeR that uses the up-to-date symmetric key cipher, RC5, with a dual keying scheme and a novel R4 framework to secure the protocol. The framework takes advantage of the recharge cycles required by MWSN nodes, to provide a

solution to the key distribution problem. A threat analysis is used to evaluate the protocol based on common types of attack.

In addition to the above contributions, the application evaluation and research methodology chapters give a good foundational knowledge with some original analysis included. The majority of these two chapters has been published as a book chapter, which will help to spread and advance the topic by giving a well-grounded introduction to those that are new to the subject.

1.4 Thesis Structure

The next chapter will firstly describe MWSNs in relation to WSNs and mobile ad hoc networks (MANETs) and continue by discussing the current, future and emerging applications that motivate the advancement of the subject, in terms of their requirements and characteristics. Chapter three will focus on the communications aspects of MWSN by first identifying the roles of the key layers in the OSI model, identifying their difficulties and limitations, as well as giving thorough literature reviews of both MAC and routing protocols designed for MWSNs. The fourth chapter will address the various methodologies used in evaluating the work presented in this thesis.

Chapter five describes the design considerations and introduces the key routing principles that arise from the literature review. This chapter then gives a technical description and derives mathematical expressions for three novel protocols, PHASeR, LASeR and RASeR. Consequently, in chapter six the modelling of the three protocols is described and subsequently evaluated and compared in both simulation and mathematical analysis. Results are gathered for five metrics over varying levels of mobility, scalability and congestion. Additionally, the three proposed protocols are compared with another MWSN protocol and two MANET protocols, which demonstrates the superiority of the presented solutions.

The design of LASeR allows it to be used in conjunction with multiple different MAC layers, as such, chapter seven evaluates the reviewed literature and selects several suitable MAC protocols. From this, some novel protocols are presented, which are designed specifically for LASeR. Extensive simulations are carried out to compare the selected MAC protocols and the results are used to determine the best option.

The eighth chapter presents four additional modes of operation for RASeR, which are designed to make the protocol more applicable to the growing needs of future MWSN applications. These include simulation results to verify their enhancement to the original protocol. Additionally, the applicability of RASeR to a specific case study is evaluated using the example application of UAV aided search and rescue.

Chapter nine presents a testbed evaluation of RASeR, in order to test its validity in a real world scenario. This involves the implementation and verification of the protocol on the selected hardware as well as gathering results from simulation as well as the testbed.

Finally, the tenth chapter considers security and introduces some additional measures to RASeR, to create RASeR-S. A thorough threat analysis is performed and simulation results are gathered to evaluate the effect of the security framework on the performance on the underlying routing protocol.

Finally, the thesis in concluded and future work is suggested in chapter eleven.

CHAPTER 2

MWSN APPLICATION EVALUATION

2.1 INTRODUCTION

As one of the main motivations behind this study of MWSNs is the large number of emerging applications that it makes possible, this chapter will categorise and evaluate the potential areas that MWSNs can be applied to, in terms of their characteristics and requirements. However, firstly this chapter will highlight the key differences between MWSNs and the closely related network types of WSNs and MANETs. In this way, the remit of a MWSN with regard to its applications can be well defined.

2.2 COMPARISON OF NETWORK TYPES

The two styles of network that are the most closely related and have had the most influence over MWSNs are MANETs [2.1] and, of course, WSNs [2.2]. MANETs are general purpose mobile networks, in which each node is required to be able to communicate with any other node in the network. The nodes are often considered to be personal computers that are able to enter the network and leave whenever they want. This requires MANETs to provide methods of allowing any two nodes to communicate over a changing topology, with a dynamic number of nodes.

Contrastingly, WSNs are usually made up of small devices dedicated to gathering sensory data. All of the data in the network should then be delivered to the sink, where it can be stored, further analysed or passed over the internet to another location. The static nature of WSNs allows the nodes to establish efficient routes and medium access in an initial phase before the data is transmitted. However, these nodes are prone to failure from harsh conditions or battery depletion, which often means that the network is reinitialised periodically.

MWSNs generally use the many-to-one communication style, in which data is gathered from the sensors and sent to the sink. The mobility of the network can cause frequent topology changes, which makes the routing of data difficult. Medium access is also a challenge since the number of nodes within transmission range will vary with time. However, the total number of nodes in the network is usually fixed and less likely to suffer node failure.

In terms of network scales, WSNs can be made up of hundreds of nodes, whereas MANETs are often at most ten or twenty. MWSN network sizes can vary based on the application, but are usually in the order of ten to a hundred. The scale of deployment is mostly due to cost, for which WSN nodes are the cheapest. They also have the least processing power and the smallest memory, battery and physical size. These limitations require WSN nodes to be very power efficient; since they are expected to have a long lifetime and may never have a chance to recharge. MANET nodes are usually more expensive and therefore have larger memory size and higher processing power. Since they are mobile, it is also assumed that they may recharge when necessary so energy efficiency isn't a major concern. MWSN nodes can be expensive depending on the application, their processing capability and memory resources are often reasonably good. Also, large batteries may be required to provide mobility, which will lessen the demand for energy efficiency in the communications system, as will the nodes' ability to travel to a power supply for recharging.

In addition to these factors, all three network types are decentralised and distributed in an ad hoc manner, which will require self-organisation amongst the nodes. Also, they will all need to overcome the challenges inherent in wireless transmission.

Overall, MWSNs share commonalities with both WSNs and MANETs, however the main attributes that define this type of network are its mobility, the use of a dedicated sink and each network often has a specific task. As MWSNs are a unique network type, they will require specific solutions to the research problems they create. These problems include MAC and routing protocols, localisation techniques, security, physical layer transmission, resource management, quality of service and many more.

2.3 APPLICATIONS

The fact that nodes may be allowed to move freely, means they can be attached to a variety of mobile platforms without compromising performance.

2.3.1 Mobile Platforms

There are four main categories of platforms that can enable the mobility of sensors, namely people, animals, manned vehicles and unmanned vehicles. These are described below along with example applications.

2.3.1.1 People

People are a primary source of mobility for both wearable sensors and sensors in objects. For example, in a smart environment [2.3] items such as smart phones and tablets can act as mobile sinks, which will be moved by people. They could gather data from devices such as a refrigerator, oven, thermostat, security cameras or television. A smart environment could also

monitor health and fitness from a user wearing a sensor. Since the mobile devices considered will be battery powered, energy consumption will need to be kept low. Additionally, as they are carried by people, the life time of the devices should also account for the time between available places to recharge. Contrastingly, any fixed devices in a smart environment, such as a smart home, are likely to be permanently connected to a power source. These types of application are also likely to only experience low speeds in small network areas.

2.3.1.2 Animals

Sensors attached to animals may be used to gather information such as location, body temperature, heart rate and ambient temperature [2.4]. This information can then be reported back to the sink for analysis by researchers to determine migration and feeding habits, as well as the health of the animals, as illustrated in figure 2.1. Similarly to devices carried by people, these nodes will be battery powered, however they may be expected to last for months or years. As such, applications using this platform are the most power restrictive so devices and protocols used will need to be very energy aware. Additionally, these applications will need large amounts of memory, but the delay requirements are less stringent. This is because it could be periods of weeks before the animals come into contact with a sink that can collect the recorded data. In contrast to having sensors attached to people, the increased speed of movement in this type of application can make route discovery difficult and nodes becoming disconnected is also more likely since the network area will be much larger.

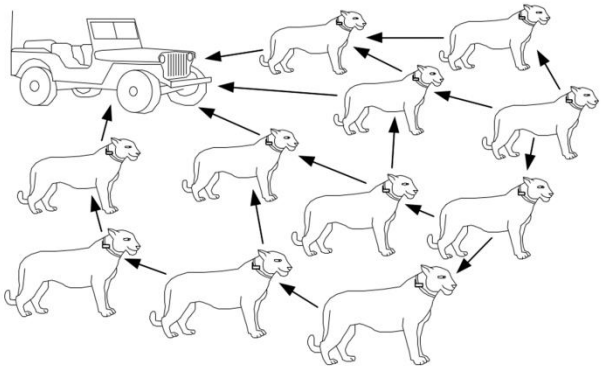


Fig. 2.1. Animals are equipped with sensor nodes that transmit data back to a sink.

2.3.1.3 Manned Vehicles

Sensors may also be attached to vehicles such as cars, bicycles or motorbikes. In the case of road vehicles, the sensors could be tracked by a central authority, to determine traffic patterns and areas of congestion [2.5]. This information can be used to route users around areas with potentially long delays and alleviate congestion. In the same way the data can also aid the emergency services in responding faster to call outs. Generally, vehicles have a large on-board power supply, which allows the node more freedom in terms of power consumption. However,

the frequency of topology change is likely to be quite high due to the potential speeds of the vehicles, which can create difficulties for routing protocols.

2.3.1.4 Unmanned Vehicles

The use of unmanned vehicles is a very promising emerging technology for a vast array of applications based predominantly on drones and robots. One example of which is terrain mapping [2.6], whereby some unmanned vehicles are equipped with sensors to measure location, altitude and distance from the ground. In this way a map may be constructed from the data, which could be utilised by the military in hostile environments as well as mapping of the sea floor and exploration of extra-terrestrial landscapes as shown in figure 2.2. Similarly to the manned vehicles, an on-board power supply is usually available; however, it's less likely to be as large. Although in the case of extra-terrestrial exploration, the use of solar power is a commonly utilised resource. It also shares the similarity of having high maximum speeds, though if a swarm of drones were deployed for a specific task it is likely that they would move in a more regular pattern than a city of individually controlled cars.

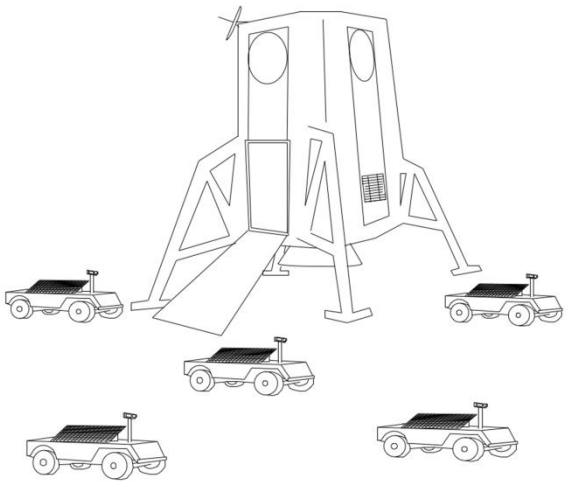


Fig. 2.2. Unmanned ground vehicles map the terrain of a planet.

2.3.2 Application Types

Applications for MWSNs can broadly be split in to four categories; constant mapping, event mapping, constant monitoring and event monitoring. Constant applications are time driven so data is recorded and transmitted periodically after a certain time interval, whereas event driven applications only report data when an event occurs. Mapping applications often perform their task once to map a phenomena, whereas monitoring applications are regularly repeating measurements over a period of time.

2.3.2.1 Constant Mapping

Constant mapping applications sense the phenomena once and report data periodically after some constant time interval. An example of this time driven sensing is in the mapping of

radiation levels in an irradiated building [2.7]; the radiation prevents people entering the building so unmanned vehicles may be sent in to map the radiation levels throughout the building. This requires the vehicles to navigate the building and sample a sensor periodically to report the radiation levels back to a sink, along with some geographical data. In this way, a picture may be built up of the radiation levels throughout the building. So, each area of the building is only sensed once and the data is reported periodically. In this type of application the frequent reporting of data to the sink can create a high level of traffic, especially if the data size is large or there are a high number of nodes. This can cause congestion, which in turn can cause latency if the networks bandwidth is insufficient.

2.3.2.2 Event Mapping

Event mapping applications sense a phenomena once, but only report data when a specified event occurs. Unmanned aerial vehicle (UAV) aided search and rescue [2.8] is an example of this, in which multiple UAVs are deployed to search an area for a victim as depicted in figure 2.3. In this case, once the target has been found its location is reported back to the sink. So each area is only sensed once and data is only sent back when an event occurs, such as the finding of the target. This also implies that the bandwidth requirement is less than that of constant mapping, since traffic is created less frequently. However, there is often a higher demand for the timely delivery of data as events usually necessitate some kind of responsive action.

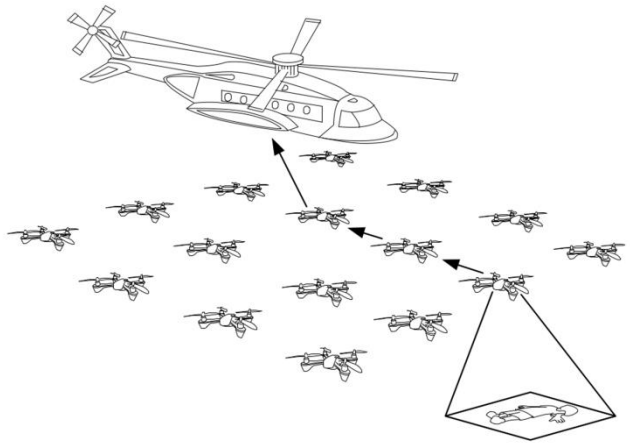


Fig. 2.3. Unmanned aerial vehicles report the location of a victim to the manned helicopter.

2.3.2.3 Constant Monitoring

Constant monitoring applications repeatedly sense the same phenomena and report data periodically after some constant time interval. An example of this would be tracking objects [2.9] or people, which could be for the covert pursuing of a suspect. Alternatively, it could be used for the safety of hikers or researchers in potentially dangerous environments, such as the rainforest. In this way, if there is an emergency they can be found quickly. In this case the location of the person being tracked would be sampled repeatedly and reported to the sink

regularly. This also implies that a low delay time will be required as well as a high bandwidth to cope with the frequency of data, making this one of the more demanding application types.

2.3.2.4 Event Monitoring

Event monitoring applications repeatedly sense the same phenomena but only report data when a specified event occurs. In terms of an example, wireless heart rate monitoring of patients in hospitals [2.10] would enable the wearers to move around freely, whilst keeping the staff informed of any irregular activity. This application would require the repeated sensing of the wearers heart rate, then if an event occurs, such as the heart rate exceeding certain predefined thresholds, a message can be sent to the sink in order to alert staff. Since data is only sent when an event occurs it is likely that there will be less traffic than constant monitoring applications. Though, similarly to event mapping applications, there is a high demand for the real time delivery of data to enable a timely response to the reported event.

2.3.3 Architecture

There are two basic types of architecture; flat and hierarchical. A hierarchical network may consist of multiple tiers, whereas a flat network will have only one.

2.3.3.1 Flat

Flat architecture implies that all the sensor nodes in the network have the same role; to both gather data and route it towards the sink. For example, military target acquisition [2.11] could utilise a small swarm of UAVs to fly ahead of an armed aircraft or ground vehicle in order to perform reconnaissance. Once the UAVs have identified the target they can relay its location back to the aircraft or ground vehicles as in figure 2.4. In this case all of the UAVs have the same role and resources. Combined with the fact that flat networks don't require any infrastructure, they are very easy and flexible to deploy in any scenario. Additionally, by equipping the UAVs with number plate or facial recognition software, the same type of network could be used in an urban environment to aid law enforcement.

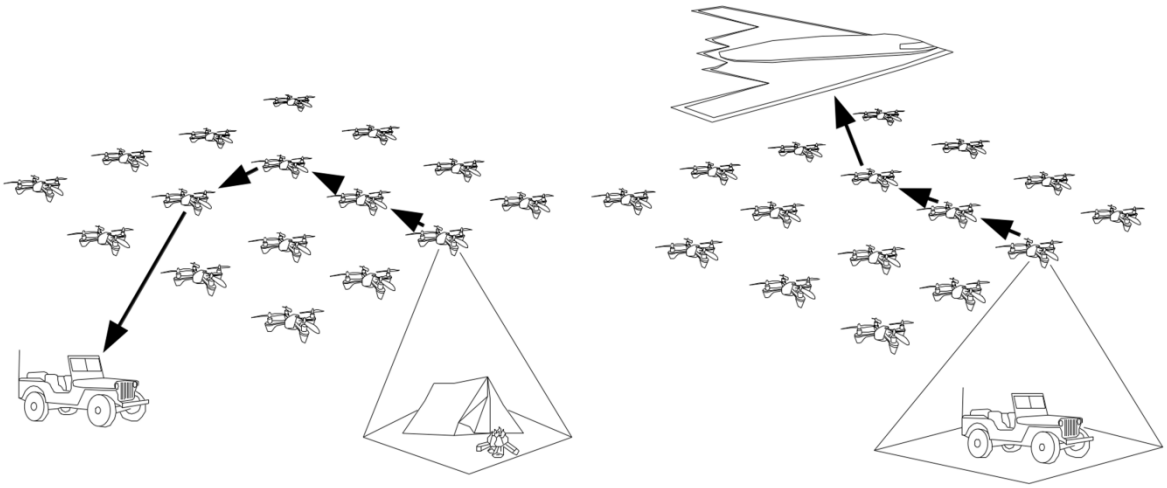


Fig. 2.4. UAVs acquire a target and report it back to the sink.

2.3.3.2 Hierarchical

Contrastingly, hierarchical networks assign different roles to different nodes, which can be done before deployment or dynamically whilst the network is being used. An example of this is in providing quality of service information to users of public transport. A city wide bus service may equip its buses with some form of positioning sensor, the data from which can be regularly reported back to a central control location. From this point, arrival times for services can be estimated and made available to users. Since the network is fixed to the city area, intermediate nodes can be fixed in strategic positions around the city. In this way, location data from the buses can be passed to the relay stations, who in turn forward it to the controller. The use of two types of node, the mobile bus sensors and the fixed relays, makes this a two-tier hierarchical network. In general this type of approach is good for energy limited scenarios, since fixed relays will often be connected directly to a power supply. Additionally, if the higher tier nodes are not static, many protocols will dynamically elect them based on residual power levels, which can prolong the lifetime of the network.

2.3.4 Heterogeneity

Sensor networks can be classified as either homogeneous or heterogeneous depending on whether they are designed to sense single or multiple phenomena.

2.3.4.1 Homogeneous

In homogeneous networks all sensor nodes monitor the same phenomena, such as a contamination cloud [2.12]. In this example a swarm of UAVs may be used to monitor the boundary of a cloud of contaminated air as illustrated in figure 2.5. Each UAV would be equipped with a sensor for determining air quality and its current location and data from both would be periodically reported to the sink. Since all of the sensor nodes are dedicated to monitoring the contamination cloud, the network is homogeneous. The requirements for this

type of network are usually very specific and will depend on the application type and mobility platform used.

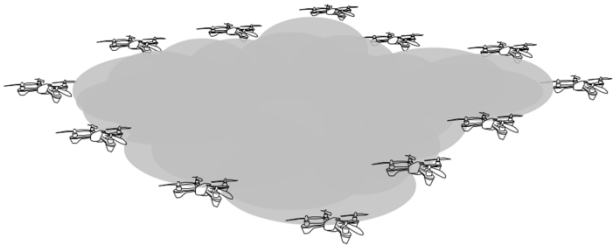


Fig. 2.5. UAVs can map or monitor the boundaries of a contamination cloud.

2.3.4.2 Heterogeneous

A heterogeneous sensor network is defined as one that is able to report on multiple phenomena. This may be by using different nodes each dedicated to different phenomena or by having multiple sensors on each node. In the case of fire rescue teams it would be beneficial for each firefighter to have wearable sensors reporting the heart rate and blood pressure of the wearer as well as the temperature, humidity and smoke density at their location [2.13]. In this way if the firefighters heart rate or blood pressure exceed predefined limits the sensors can alert the incident commander, who can recall the firefighter. Alternatively, if the firefighter is not responding another firefighter can be sent in and the location information can be used to enable the swift rescue of their teammate. Simultaneously, the temperature, humidity and smoke density information can be periodically reported and mapped to enable the incident commander to make critical operational decisions based on real time information. In this situation, the constant mapping of the environment and the event monitoring of the firefighter are being done at the same time. It's likely that the requirements for heterogeneous networks will be higher than that of homogeneous networks as multiple requirements for parameters will have to be accommodated. The combination of different nodes with potentially varying traffic generating frequencies, delay requirements and mobility platforms makes heterogeneous sensing one of the most challenging types of MWSN application.

2.3.5 Application Requirements

Each unique application will have its own set of requirements, based on a wide variety of parameters. These may be imposed by the nature of the application or by external bodies, such as the manufacturer, designer or end user. Since assumptions cannot be made on the limitations driven by external bodies, such as cost or design choices, this section will focus on the demands created by the applications themselves.

2.3.5.1 Prerequisites

The type of application will dictate certain prerequisites for the functionality of the application. For example, a health monitoring network would require a low end-to-end delay between the sensor detecting an emergency and the sink reporting it. This would also need a high packet delivery reliability so that the packet isn't lost and the emergency can be responded to as quickly as possible. This application would also imply that the task lifetime should be relatively long as the patient may be wearing the sensor for some time.

2.3.5.2 Characteristics

Each different application will also have certain characteristics; for example terrain mapping using UAVs to periodically take measurements as they explore the landscape. This will generate a frequent number of transmissions with potentially a high quantity of topography data, which will lead to a high level of traffic. The number of nodes required for a full scale mapping is likely to be large, however in some cases, such as the mapping of extra-terrestrial terrain using unmanned ground vehicles (UGVs), it's possible that the cost of this would severely limit the number of nodes used. The lifetime of this task is likely to be fairly short, using a high amount of energy for movement and a high amount of energy for the periodic transmissions, which will all be powered from a medium sized battery. The area to be mapped is likely to be large giving a low density of high speed nodes.

2.3.5.3 Requirements

Knowing the prerequisite needs and characteristics of an application, its requirements for transmission range, bandwidth and power efficiency can be determined. In the case of health monitoring, a high number of patients in a building would require only a small transmission range. In terms of power, the infrequent traffic and long lifetime would necessitate an average power efficiency. Additionally, the short transmission range and small battery capacity would also require an average power efficiency. Even though this application generates a low level of traffic, the need for a short end-to-end delay, with high reliability could require a medium bandwidth to avoid any chance of congestion or packet loss.

On the other hand, the constant mapping with UAVs application is not so strict on the delay and reliability criteria, but will be generating a lot of traffic and as such it will require a high bandwidth. The high traffic level can cause large energy consumption; over a short task lifetime this implies that an average level of power efficiency is necessary. However, the low density of nodes over a potentially large network area will require a long transmission range, which will increase its energy requirements. Combining this with the power needed for movement and the medium sized battery capacity, a high demand for power efficiency can be inferred.

2.3.6 Comparison

This section will draw conclusions from the requirements and implications of certain categories of MWSN applications, these are summarised in table 2.1. It should be noted that these are only guidelines for assessing applications based on typical scenarios and each application should be evaluated individually to determine its exact requirements and implications.

	Prerequisites			Characteristics							Requirements		
Application Type	End-to-End Delay	Packet Delivery Reliability	Task Lifetime	Frequency of Traffic Generation	Number of Nodes	Energy Consumption	Network Size	Speed of Movement	Node Density	Battery Size	Transmission Range	Bandwidth	Power Efficiency
Constant Mapping	Medium	Medium	Short	Frequent	N/A	High	N/A	N/A	N/A	N/A	N/A	Large	Average
Event Mapping	Short	High	Short	Irregular	N/A	Low	N/A	N/A	N/A	N/A	N/A	Medium	Low
Constant Monitoring	Medium	Medium	Long	Frequent	N/A	High	N/A	N/A	N/A	N/A	N/A	Large	High
Event Monitoring	Short	High	Long	Irregular	N/A	Low	N/A	N/A	N/A	N/A	N/A	Medium	Average
Mobility Platform													
People	N/A	N/A	N/A	N/A	Medium	None	Small	Low	High	Small	Short	N/A	Average
Animals	N/A	N/A	N/A	N/A	High	None	Large	Medium	Medium	Small	Medium	N/A	High
Manned Vehicles	N/A	N/A	N/A	N/A	Medium	High	Large	High	Low	Large	Medium	N/A	Average
Unmanned Vehicles	N/A	N/A	N/A	N/A	Low	High	Large	High	Low	Medium	Long	N/A	High

Table 2.1. Table of the qualitative analysis of different MWSN application categories. The results are given in relative comparison to each other rather than absolute terms.

2.3.6.1 Application Type

Table 2.1 shows that application type can give a good indication of a networks requirements. In general event driven applications require a timely response, so delay should be low and reliability high. Additionally, the irregular data generation is usually of a low rate, whereas applications that constantly generate data at regular time intervals often experience a more consistent high level of traffic, which is reflected in the bandwidth requirement. These time driven applications are also less sensitive to packet loss; since if a packet is lost another one will be generated within a defined time period. However, in event based applications, the trigger may only be reported once, which gives a high weighting to reliability. In general, the task lifetime of a mapping application is only the time it takes for the nodes to document the phenomena and then it has fulfilled its task. In the case of monitoring applications, the phenomena could need to be observed for an indefinite period of time, so it's expected task lifetime is much longer. Since more traffic requires more power being used to transmit packets, the constant monitoring and constant mapping applications will have a higher energy consumption than the event monitoring and event mapping applications. Subsequently, the power efficiency implications are drawn from a combination of the task lifetime and the energy consumption. Constant monitoring applications generate a large amount of traffic and will be required to do so for a long time, as such over the task lifetime they will consume the highest amount of power and as such require the highest power efficiency. On the other hand, event mapping applications will use up less energy from the transmitting of data and also has a lower expected task lifetime, so it will have the lowest power consumption over the task lifetime and

therefore has the lowest need for power efficiency. The constant mapping and event monitoring applications generate lots of data over a short time and little data over a long time respectively. This gives them both a medium level of energy consumption over the task lifetime and an average need for power efficiency.

2.3.6.2 Mobility Platform

One of the key implications that can be drawn from the mobility platforms is that of battery size. Since vehicles often have large batteries anyway it is feasible to assume that this resource can be used to power a sensor node, whereas a person or an animal would have to carry the additional weight and so their battery size is considered to be small. Though, it is also true that the energy consumption due to the powering of vehicles will be high, whereas no additional power is required for mobility from people or animals. Applications made mobile by people are usually based in small areas such as buildings, with a relatively average number of nodes. This leads to a high density of sensors which only needs a small transmission range. Alternatively, large numbers of sensors made mobile by animals are often spread over a large outdoor area giving them a medium node density, which requires a medium distance transmission range. Contrastingly, unmanned vehicle applications generally use a small number of nodes, due to cost and complexity, but they can be spread out over large areas. This causes a low density of nodes and demands a large transmission radius. Additionally, manned vehicle applications also use a large network area, but with a medium amount of nodes. This creates a low density of nodes, but due to the large area would require a medium transmission range. From this point the expected power efficiency can be determined. Nodes made mobile by animals have a medium transmission range but only a small battery, so have a high demand for power efficiency. Nodes made mobile by people also have a small battery, but only require a short transmission range, so need a relatively average level of power efficiency. Unmanned vehicle applications have medium sized batteries and require power for movement as well as having a long transmission range, so demand a high level of power efficiency. Manned vehicle applications have large batteries but only require a medium distance transmission range, however they also need a large amount of power for movement and as such have an average demand for power efficiency.

2.3.6.3 Architecture and Heterogeneity

In comparison to application type and mobility platform, the architecture and heterogeneity cannot contribute as much in terms of conclusions. This is because the architecture is often chosen based on the desired implementation of the application. If the network is homogeneous, then the qualitative analysis in table 2.1 can be applied directly based on the categories it falls into. However, if the network is heterogeneous then it may overlap into multiple categories, which generally means that it will require more resources, higher power efficiency and higher bandwidth.

2.4 CONCLUSION

This chapter firstly defined a MWSN in relation to both WSNs and MANETs before focusing on the vast multitude of applications for which MWSNs are suitable. By categorising these applications in terms of mobility platform, type, architecture and heterogeneity, their prerequisites and characteristics can be identified. Furthermore, based on this, conclusions about the applications requirements can be drawn. The next chapter will explore the currently available literature on the topic of MWSNs, specifically concerning that of MAC and routing.

CHAPTER 3

LITERATURE REVIEW

3.1 INTRODUCTION

This work is focused on the communications aspects of MWSNs, which is a major part of any sensor network, since their functionality relies on the ability to transport the gathered data from the source to the sink. In general, communications systems are split into functional layers based on the open systems interconnection (OSI) model. The OSI model defines seven layers, of which three are of key importance in a MWSN; the network layer, the data link layer and the physical layer. This chapter will take each of these key layers in turn and identify their main function and a thorough review of the major MAC and routing literature for MWSNs will also be considered.

3.2 PHYSICAL LAYER

The physical layer (PHY) [3.1] is responsible for the actual transmission of messages from one node to another. In MWSNs this is essentially the radio used for the wireless transmission of packets. The PHY is required to transmit data reliably and quickly, whilst minimising power consumption and bandwidth. The fundamental issues for this layer are common to most wireless systems, such as noise, path loss and interference, which are likely to introduce errors. Additionally, in a mobile environment Doppler shift can cause fast fading. Some of the techniques used to overcome these issues include forward error correction (FEC) coding, frequency spreading and modulation selection. In general the main restriction is cost, so sensor nodes will often be equipped with a low power, low bandwidth, single frequency, simplex radio. The power restrictions reduce the transmission radius available to the node, making multihop routing the only way to transmit across long distances. Also, the simplex radio means that a node cannot transmit and receive at the same time, combined with the fact that it's using a shared medium, there is the possibility of collisions.

There is no current standard for MWSNs, but WSNs tend to favour the 802.15.4 standard, which has a maximum data rate of 250kbps. It uses direct sequence spread spectrum (DSSS) to tolerate noisy channels, with offset quadrature phase shift keying (O-QPSK) modulation [3.2].

This is designed as a low power, short range solution and as such are generally intended to be operated in the tens-of-metres range.

3.3 Data Link Layer

In a MWSN the data link layer is responsible for the avoidance of these collisions, whereby two nodes transmit at the same time, in close proximity to each other. This will cause their packets to collide in the air and can prevent other nodes from receiving the packet correctly. Collision avoidance is performed by a MAC protocol. There are many types of MAC protocol, which can be based on either time, frequency or code.

3.3.1 Code and Frequency Based Multiple Access

CDMA (Code Division Multiple Access) [3.3] uses orthogonal codes to allow multiple nodes to transmit simultaneously, although the computational complexity required for these operations is potentially too great for resource constrained sensor nodes. Similarly, FDMA (Frequency Division Multiple Access) [3.4] is unsuitable for the same reason as it would require nodes to have expensive radios that could send and receive on multiple channels.

3.3.2 Time Based Multiple Access

Generally, MWSN MAC layers use some form of time division to allow multiple nodes to share the medium. This means that a single transceiver can be used on one frequency, but the trade-off for this is the additional delay incurred whilst waiting for the medium to become free. In general, there are two basic schemes upon which the vast majority of MWSN MACs are based.

The first of these is TDMA (Time Division Multiple Access) [3.5], which often uses a central controller to allocate time slots to nodes on demand. This ensures that bandwidth is not wasted as nodes are only allocated the time they need, which can be very efficient. This is particularly applicable in hierarchically structured routing protocols, in which cluster heads can assign time slots appropriately. However, in direct relevance to this work, is the global TDMA (GTDMA) MAC [3.6], in which nodes are allocated a single time slot before deployment. The time slots cycle continuously with each node only transmitting in its own slot, making the network collision free. Although, bandwidth can be wasted if a node has no data to transmit in its slot and in large networks a node may have to wait for a long time before it gets a chance to transmit, which may cause significant delays. TDMA protocols also rely on the accurate synchronisation of nodes. These MACs are contention free since nodes aren’t having to compete for access to the channel.

3.3.3 Contention Based Multiple Access

The second most commonly used MAC layer is carrier sense multiple access with collision avoidance (CSMA/CA) [3.7]. This protocol is contention based, since each node is in direct competition with the others to gain channel access. The protocol requires each node to listen to the medium before transmitting, then if the channel is clear the message may be sent. It may be the case that two nodes perform this action at the same time, both hear that the medium is free and then both transmit, causing a collision. To detect this it is common for an acknowledgement (ACK) to be sent from the receiver to the transmitter as a receipt of the packet. If no ACK is received the nodes will both wait for a random amount of time before trying to transmit again.

Other issues with the CSMA/CA MAC are the hidden and exposed node problems [3.8]. The hidden node problem occurs when a node listens to the medium but cannot hear another node that is transmitting because they are out of range. This is illustrated in figure 3.1 part 1, in which both nodes A and C wish to transmit to node B. Node A is already transmitting to node B, but since node C is out of range it may then decide that the channel is clear and begin to transmit, causing a collision. The exposed node problem occurs when a node is transmitting and another node overhears it and defers sending, even though the receiving node is beyond the second nodes transmission range. This is shown in figure 3.1 part 2, where node B wants to transmit to node A and node C wants to transmit to node D. Node B is already transmitting, which causes node C to conclude that the medium is in use. This means that node C will wait until node B has finished sending before transmitting to node D, even though both transmissions could occur simultaneously without collision. The hidden node problem has been addressed by the introduction of a request to send (RTS), clear to send (CTS) handshake. The process begins with the transmitting node sensing the medium to be free and sending a RTS packet, the receiver then has the option to deny the request or accept it by replying with a CTS packet.

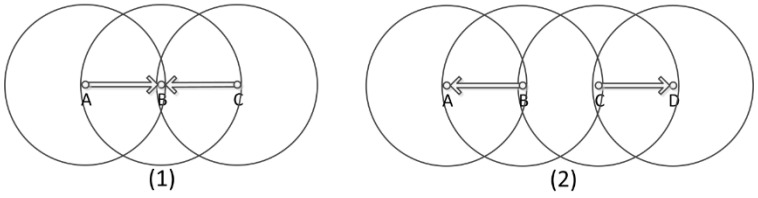


Fig. 3.1. The hidden node problem and the exposed node problem.

3.3.4 MWSN MAC Protocols

In addition to those mention above, there have also been a number of MAC protocols proposed, which are specifically designed for MWSNs.

One such as MS-MAC (Mobility Aware Sensor MAC) [3.9]. MS-MAC is an extension of S-MAC (Sensor MAC) [3.10], which is based on the CSMA/CA. S-MAC is designed to limit energy consumption with the use of periodic duty cycles, in which means that nodes are only awake and listening at certain intervals. Nodes share their sleep schedules every two minutes in

a synchronisation period; this means that they will know when to listen for broadcasts by neighbours. Some nodes may be in range of two nodes that have different sleep schedules, in which case it will be expected to adhere to both. Nodes that share a schedule form a virtual cluster, the nodes that are following more than one schedule link up these clusters and are called border nodes. In a MWSN nodes may become disconnected from one virtual cluster, they will then have to wait for up to 2 minutes before a synchronisation period occurs and they can join a new cluster. MS-MAC aims to reduce this latency by adjusting the frequency of synchronisation periods according to the mobility of the cluster. So, each node measures the signal strength from each neighbour and if it changes then the nodes are assumed to be moving relative to one another. From this information the speed is estimated, stored and then broadcast in the synchronisation period, which then allows everyone to adjust the frequency of synchronisation. This adaptability improves energy consumption in low mobility networks but also reduces packet losses and delay in high mobility.

Another protocol based on SMAC is MOBMAC [3.11], which attempts to improve the performance of SMAC by reducing the amount of frame loss due to the Doppler shift in transmissions from a mobile source. This is done by varying the frame size, such that in bad channel conditions only small frames are transmitted and in good channel conditions larger frames are permitted. This means that in bad channel conditions any losses are kept to a minimum and good channel conditions are taken advantage of by transmitting more data.

MS-SMAC [3.12] is similar to MS-MAC in its scheduling, however it requires the use of known static nodes to transmit messages to the mobile nodes, such that the mobile nodes can determine their speed more accurately. In this way it allows mobile nodes to indicate to its neighbours when its speed has changed and then update the synchronisation frequency accordingly. Each node also records how long it's expecting to be able to talk to each of its neighbours.

Similarly, MobiSense [3.13] also uses static nodes; in this case the static nodes are intentionally placed cluster heads which form the backbone of the network. Adjacent clusters are required to communicate in different channels, such that they can reduce inter-cluster interference and schedules can be efficiently designed locally by the cluster head. MobiSense also uses regular broadcasts from cluster heads with information about their cluster including the channel number, such that nodes can move between clusters easily.

Alternatively, MMAC (Mobility-Adaptive Collision-Free MAC) [3.14] is based on TRAMA (Traffic-Adaptive Medium Access) [3.15]. TRAMA works by splitting frames into a random access setup phase and a scheduled access data transmitting phase. The setup phase allows nodes to share topology information and traffic information, such as the rate of data

being generated and the intended receiving nodes. Also, the setup time can be used by new nodes, to join the network. Once the setup phase is complete the nodes will compute priorities for each time slot based on the shared information giving contention free access to the highest priority node. If a node has priority but no data to send then it may give up the slot to another node. MMAC uses location awareness to attempt to predict node mobility with an autoregressive model. The predicted mobility is then used to dynamically adjust the frame size, such that a highly mobile network will have a smaller frame size, meaning that topology and traffic information is update more frequently. In order to keep the entire network synchronised and using the same frame length, nodes will periodically transmit their expected next positions to a cluster head, which will in turn broadcast the information to each node in the cluster. The nodes will then individually calculate the new frame time and transmit it back to the cluster head, which then averages the results. The averaged frame times from each cluster head are then sent to a single second tier cluster head, which then takes a global average and disseminates this to the rest of the network.

Another MAC based on time synchronised scheduling is M_TDMA (Mobility-Aware TDMA) [3.16], which adapts a classical TDMA MAC for use in mobile scenarios. It does this by adding a control phase at the beginning of each round, in which a cluster head will broadcast the cluster information to its neighbour nodes. When a node hears this, it will know whether it has moved clusters, is in the same cluster or if it is out of range of any cluster heads. If the node has moved clusters, it will then announce its presence to the cluster head, who will subsequently broadcast new time slot allocations. In the case of multiple nodes joining a new cluster, their announcements may collide, in which case they should initiate a random back-off timer and attempt to join the cluster in a later round.

Contrastingly, MA-MAC [3.17] is a low duty cycle protocol, which uses the short preamble of X-MAC [3.18], such that nodes can save time and energy. The short preamble contains the ID of the destination node and is strobed. This means that when a node wakes up and hears the preamble, if it is the intended destination then it can transmit an early acknowledgement before the preamble is retransmitted. The transmitting node can then follow this up by sending the data packet. MA-MAC improves on this system by including a handover mechanism, which reduces packet loss due to link breaks. During transmission, if the transmitter detects that the receiver is beyond a certain threshold, it will begin to look for potential relay nodes, which may then be used to pass on data to its intended recipient.

As an alternative to the schedule based MACs, MA-CSMA/CA (Mobility Adaptive Carrier Sense Multiple Access with Collision Avoidance) [3.19] takes the 802.15.4 MAC [3.20] for static WSNs and improves it for mobility. It does this by speeding up the process by which mobile nodes may switch clusters. Nodes measure the signal strength of all the cluster heads in

range and if one becomes stronger it is assumed that the node is moving towards it. At this point the node will request a time slot in the new cluster via its current cluster head. So the request is passed to the nodes cluster head, who then passes it to the cluster head that the node is moving towards. Once the new cluster head has received the request it allocates a contention free time slot for the node, which may then be used for association. After the time slot has expired the node reverts to the standard method of contending for the medium.

Similarly, from the same authors, CFMA (Collision Free Mobility Adaptive) [3.21] is also based on the 802.15.4 MAC. CFMA dictates that the cluster heads allocate a delay time to nodes based on their priority. This allocated delay is used instead of the random backoff that is normally performed in order to reduce collisions. Nodes that are moving clusters are given high priority and therefore a shorter delay time.

3.3.5 Conclusion

Overall, the majority of the MACs designed for MWSNs are based on WSN MACs and the focus is on enabling them to cope with a dynamic topology. Furthermore, the majority of MWSN MACs are adaptations of the classic CSMA/CA or a dynamic TDMA, which may not be suitable in all situations. Additionally, many of the protocols are designed for hierarchical routing protocols, which may not be the ideal routing technique for a mobile network.

3.4 NETWORK LAYER

The network layer is responsible for the end-to-end delivery of data between two nodes, which will often require multiple hops. This is done with the use of a routing protocol, which presents one of the main challenges in MWSNs. The routing protocol defines how a packet should move through the network, instructing each node on how to treat a received message.

3.4.1 Routing Requirements

Routing protocols may have many requirements, the major one being packet delivery reliability. A high packet delivery reliability means that the majority of packets created by the nodes are successfully delivered to the sink. This is always a priority in any application; however emergency or health orientated networks may have much lower tolerances for packet loss. End-to-end delay is also a concern in many applications, since some networks rely on the timely delivery of data so that it can be acted upon quickly. Other MWSNs may accept large amounts of latency between the gathering of the data and its reception by the sink, these kinds of networks are referred to as delay tolerant networks (DTNs). Often these networks will be particularly sparse, with nodes being spread over large distances. Another concern is security, which considers whether the information being transmitted can be kept confidential as well as

whether the network is vulnerable to an attack. These requirements may be critical in certain applications, especially those that involve sensitive data.

In addition, for these protocols to function properly in an ad hoc environment they must be self-organising. This will allow them to be randomly distributed in any situation and still perform well. They are also expected to be scalable to potentially hundreds of nodes, though most MWSN applications currently require less than this. Resource efficiency is also a factor since nodes will be limited in terms of energy, bandwidth, memory and computational power. As such, a good routing protocol is expected to have low overhead, which will reduce bandwidth requirements. Moreover, as the nodes radio is often a major component of energy consumption, a low overhead will reduce this as well. The memory and computational power limitations mean routing protocols should be kept as simple as possible.

3.4.2 Routing Challenges

The main challenge for a routing protocol in a MWSN is the potentially frequently changing topology. This means that it can be difficult to determine a route from the source to the sink and even once a route has been discovered it may not exist for very long. There is also the issue of nodes becoming disconnected from the sink, in which case no route will exist. Additionally, channel conditions may make this worse, as in a fading channel link reliability may be low. Unexpected node failures can also occur due to power failure or hardware malfunction, which can impede the protocols route discovery. This constantly changing environment requires highly adaptive protocols to find the most efficient and reliable routes through a network.

Another common issue is the sink-hole problem [3.22], in which the nodes closest to the sink have to pass on the most information. This leads to them using up more power than nodes on the edge of the network, which can cause them to fail faster. Though, this is more of a concern in static sensor networks, it should be taken into account in slower moving networks.

Additionally, some protocols require localisation, such that each node is aware of its position in the network. This is commonly achieved with the use of GPS [3.23], however GPS consumes a lot of power and is not always accurate. Other techniques have been proposed using a subset of anchor nodes [3.24]; these nodes are either equipped with GPS or have a fixed, known location. Anchor node methods are usually based on measuring one aspect of a received signal to determine distance. This metrics are usually either time-of-arrival, angle-of-arrival or received signal strength. In general, once a nodes distance from three or more anchor nodes has been established, then its relative coordinates can be determined.

Another requirement of some protocols is the strict synchronisation of nodes, which can be difficult in an ad hoc network. Some proposed techniques include the use of a third party

transmitting a broadcast pulse to synchronise with, such as in RBS (Reference Broadcast Synchronisation) [3.25] and RIP (Reference Interpolation Protocol) [3.26]. Alternatively, a dynamically established hierarchy can be used to propagate timing information, such as in TPSN (Timing-Sync Protocol for Sensor Networks) [3.27], or if GPS is used each node can synchronise to the received time from the satellites [3.28]. Some alternative methods are also surveyed in [3.29]. Additionally, [3.30] describes a working implementation of a TDMA based sensor network that uses an out-of-band AM transmitter to maintain global synchronisation.

3.4.3 Routing Techniques

The ad hoc nature of a MWSN favours a distributed approach to routing, which is a commonality amongst nearly all MWSN routing protocols. There are a number of techniques in the literature for handling the challenges mentioned above, whist also trying to fulfil the requirements made by the demanding applications of MWSNs.

3.4.3.1 Hierarchical and Flat

In WSNs the general dichotomy of protocols is split between flat and hierarchical. The hierarchical approach assigns different tasks to different nodes; generally groups of sensors are split into clusters and one of them will be elected as a cluster head. It's then the role of the cluster head to receive data from the sensors and relay it to the sink. This can be seen in protocols such as LEACH-M (Low Energy Adaptive Clustering Hierarchy - Mobile) [3.31] and LEACH-ME (LEACH-M Enhanced) [3.32], which are based on the popular WSN protocol, LEACH (Low Energy Adaptive Clustering Hierarchy) [3.33]. The LEACH protocol consists of two phases; the setup phase and the steady state phase. Initially, in the setup phase nodes will randomly nominate themselves to be cluster heads, then the remaining nodes will join clusters by becoming associated with the cluster heads. It's then the role of the cluster head to derive a schedule, such that each of the nodes associated with it are allocated a time slot. Then in the steady state phase, nodes will transmit data to the cluster head in their own time slot, where it will be aggregated and passed on to the sink.

LEACH-M adapts this protocol for mobility by using a timeout mechanism, allowing nodes to realise when they have become disconnected from their cluster head, in which case they should try to associate themselves with another cluster. LEACH-ME takes this further by adding another metric for cluster head election. This protocol will select cluster heads based on each nodes level of mobility, in this way the least mobile nodes will become clusters heads, making the network more stable.

CBR Mobile-WSN (Cluster Based Routing for Mobile WSNs) [3.34] works in a similar way to the above, however each cluster head includes additional empty time slots in their schedule. These empty time slots can be used by nodes that have become disconnected from

their own cluster, which helps to reduce packet loss. ECBR-MWSN (Enhanced Cluster Based Routing for MWSNs) [3.35] took this and included a metric for cluster head selection that took in to account a nodes residual energy and distance from the sink. This means that the nodes with the most energy should become cluster heads, which will improve the lifetime of the network. MBC (Mobility Based Clustering) [3.36] on the other hand, uses a cluster head selection metric based on estimated connection time, residual energy, as well as a cluster heads node degree and distance from the sink.

Alternatively to a hierarchical structure, a flat architecture may be used, in which each node has the same status. In other words there are no cluster heads and every node behaves in the same way, such as in the WSN protocols SPIN (Sensor Protocol for Infusion via Negotiation) [3.37] and DD (Directed Diffusion) [3.38]. With regards to MWSN routing protocols, DD was adapted into DCBM (Data Centric Braided Multipath) [3.39], which is an event driven, reactive protocol for MWSNs. Essentially, routes are created when an event occurs, and in this case the event is a query made by the sink. The sink floods the network with the query and each intermediate node caches information about the nodes that it received the query from. Then, once the intended node receives the query it may then start to transmit data back down the path that was taken by the query. DCBM also allows nodes at each hop, to send the packet down multiple paths towards the sink, in order to reduce the chances of a break in a path to cause packet loss. The use of multipath routing is a common method of improving packet delivery reliability in mobile networks. It's beneficial since, if one route is lost and subsequently loses the packet, there will be another still active to successfully deliver the data. The downside to this technique is the added redundancy uses extra resources, such as bandwidth and power.

Another method used to increase packet delivery reliability is node cooperation, RRP (Robust cooperative Routing Protocol) [3.40] is a flat protocol that aims to create robust paths through the network by allowing nodes to relay failed transmissions. If a node overhears a failed transmission and it's within range of the intended recipient, then it can relay the packet. This mechanism can decrease the chances of packet loss and also remove added delay created by route maintenance or route error messages.

Often WSN protocols are adapted for MWSNs by adding functionality to allow them to cope with the mobility of nodes. The hierarchical protocols have an inherent delay in setting up clusters, but benefit from low power consumption. They will often also use data aggregation to combine multiple packets of similar data into one message, which saves bandwidth. However, high mobility can cause nodes to switch clusters regularly, which introduces delay from the overhead created in a node joining a new cluster. In extreme cases this can cause instability in cluster based protocols, which may results in long delays and packet loss. Often flat routing is preferred in mobile networks due to their ability to cope with frequent topology changes.

Additionally, since all nodes are equal at all times algorithmic complexity is reduced. However, they are generally not as scalable as hierarchical protocols.

3.4.3.2 Proactive and Reactive

Contrastingly to WSNs, MANET protocols are usually flat and are then characterised as either proactive or reactive. Proactive protocols determine routes pre-emptively, such that there's always a route ready when a node needs it. This method of gathering topology information can cause high overhead from large routing tables being disseminated throughout the whole network. In highly mobile scenarios this information will need to be updated regularly causing congestion, which can create in large delays and packet loss. Alternatively, if routing tables aren't updated regularly enough then packets may be sent down routes that no longer exist. Reactive protocols tend to perform better in highly mobile situations since they aim to only discover routes when they're needed. This does mean that there is a small delay for route discovery between when a node has a packet to send and actually sends it. It also can cause issues in high traffic applications from the network becoming congested with route discovery overhead.

DSR (Dynamic Source Routing) [3.41] is a flat, reactive routing protocol for MANETs, which discovers a route by first flooding the network with a route request, which records the route is has taken. Once the destination node has received the route request it will respond with a route reply back along the path taken by the route request. AODV (Ad-hoc On-demand Distance Vector) [3.42] uses a similar technique, however each intermediate node stores its next hop for a given destination. Additionally, as mentioned before, multipath routing is a common choice for improving a protocols reliability. This can be seen in AOMDV (Ad-Hoc on Demand Multipath Distance Vector) [3.43], which takes AODV and adds a multipath element.

AODV++ [3.44] is designed for MWSNs and based on AODV, however it makes a more informed choice on routes influenced by link reliability, node energy levels and traffic rates. In this way the protocol attempts to prolong the network lifetime, whilst also trying to find the quickest and most reliable route to the sink. AODV-PSR (AODV with Pre-emptive Self Repair) [3.45] is another MWSN protocol also based on AODV, which tries to predict link breaks and then provide alternate routes.

One recently explored technique is cross-layer routing, which has also been incorporated in MWSN protocols; MACRO (Mobility Adaptive Cross-layer Routing) [3.46] exploits the interaction between the layers by sharing information such as average speed and RSSI data. Similarly to AODV, MACRO discovers routes using the common route request/route reply technique, however it limits the flooding by minimizing the number of nodes that forward the requests. In order to improve the reliability of packet delivery, routes are given an expiry time

based on the mobility of the node and the link quality. This ensures that only the most reliable routes are used. Energy is saved by using an adaptive duty-cycle in the MAC layer and also adjusting the transmission power of the nodes radio.

In contrast to the proactive routing style, MANET protocols such as OLSR (Optimised Link State Routing) [3.47] and DSDV (Destination Sequenced Distance Vector) [3.48], opt for a proactive approach, in which routes to all other destinations are recorded. DSDV nodes each keep a record of every possible destination node, their distance and which node is the next hop. In OLSR the frequency at which the routes are refreshed is dependent on how often the topology changes. As such, highly mobile networks tend to avoid these kind of protocols as this would require frequent refreshing of routes, which causes a high amount of overhead.

3.4.3.3 Opportunistic

Opportunistic protocols tend to perform well in MWSNs as they take advantage of the topology presented to them. Instead of defining a path through the network, a set of potential forwarding nodes are defined. In this way the best forwarding neighbour from the defined set can be chosen in an opportunistic way. For example Ex-OR (Extremely Opportunistic Routing) [3.49] is designed for ad hoc networks and attempts to minimise the number of transmissions between the source and the destination. It does this by collecting information about the other nodes in the network, which enables the source node to compile a list of potential forwarding nodes. The packet is then broadcast with this list of forwarding nodes, which are in order with the node closest to the destination listed first. Any nodes that overhear the transmission and are listed in the packet, wait for a certain amount of time and then transmit the packet if a node higher on the list hasn't already done so.

The MWSN protocol OR-RSSI (Opportunistic Routing - RSSI) [3.50] is based on Ex-OR and uses a high power beacon from the sink to allow each node to determine a relative distance gradient based on the RSSI. In a similar way GOR (Geographically Opportunistic Routing) [3.51] assumes each node knows its own location in the network as well as the sinks. GOR also splits up the network into grids and the source will first try to transmit to any node in the cell which is closest to the sink and still within its transmission range. If this fails the node will try increasingly closer cells until the transmission is received.

3.4.3.4 Location Aware

The use of location awareness has become a popular choice with many devices already utilising services such as GPS. Additionally, there are many applications, such as tracking and mapping, which require location information to be gathered. For these reasons it may be advantageous for a routing protocol to take advantage of this knowledge, which is already available. GPSR (Greedy Perimeter Stateless Routing) [3.52] is a protocol designed for mobile

networks, which incorporates the use of location information as a gradient metric. Each nodes periodically broadcasts its current position to its neighbours in a proactive manner, such that each node can maintain a list of neighbours and their positions. This list can then be used to greedily select the closest node to the indented destination to be the forwarding neighbour. The MWSN protocol GPSR-MS (GPSR with Mobile Sensors) [3.53] is based on GPSR but instead of only using the location information as a gradient metric, it adds speed and direction.

The use of location information can give rise to the dead-end problem [3.54], whereby a transmitting node forwards its packet to a locally maximal neighbour. In other words if a transmitting node chooses to forward a packet to another node that's closer to the destination, but this node has no suitable forwarding neighbours of its own, then the packet has reached a dead-end. Figure 3.2 shows this case, where the labelled node is a dead end, since it has no neighbours that are closer to the sink. In GPSR and GPSR-MS this problem is solved using the right-hand rule, which simply selects the forwarding neighbour as the next node counter-clockwise from the neighbour that transmitted the packet. PAGER-M (Partial-Partition Avoiding Geographic Routing - Mobile) [3.55] is specifically designed for MWSNs with the intention of identifying the dead end nodes and increasing their gradient such that packets are routed around the problem. This is done in two phases, the first identifies the locally maximal nodes and any other nodes which will ultimately forward data to them. The initial gradient of a node is set to be that nodes Euclidean distance from the sink, the second phase increases the gradient of the nodes identified in the first phase to ensure that packets are routed away from the dead-end.

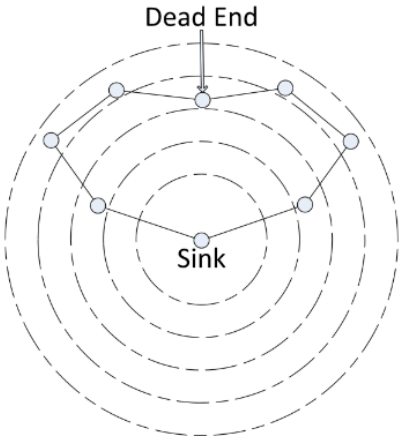


Fig. 3.2. The dead end problem.

ADSR (Angle-based DSR) [3.56] has been adapted from the flat reactive MANET protocol, DSR, and also uses location information. The geographic information is used at each node to determine the angle between itself, the fixed position sink and a potential forwarding node. This information is then used to select a forwarding node, which ensures that the data is always transmitted towards the sink in a greedy manner.

The proposed mechanism in [3.57] allows nodes with data to transmit, to request location information from their neighbours. The authors suggests that disseminating location information through the use of beacons may cause nodes to forward data based on out-of-date information. So, by allowing location information to be retrieved on-demand, nodes will make better choices.

Location information has also been used in hierarchical protocols designed for MWSNs, such as ZBR (Zone Based Routing) [3.58]. ZBR splits the network area into predefined zones, such that each node knows which zone it's in by its geographical location. Each zone has a zone head, which is elected based on a mobility factor, so that the least mobile node will become the zone head. Once zone heads have been established they discover routes to the sink using a route request and reply mechanism similar to that of AODV. Essentially, data gathered by a node is passed to its zone head, which then forwards it through other zone heads to the sink.

LFCP-MWSN (Location aware Fault-tolerant Clustering Protocol for MWSNs) [3.59] is another hierarchical protocol that uses location information to create clusters and then cluster heads are selected based on mobility and residual energy. This setup is performed periodically to refresh the cluster heads. A time out scheme similar to LEACH-M is used to determine when a node has been lost from a cluster and should try to join another. LFCP-MWSN also saves energy by only enabling GPS on a subset of the nodes in the network, called anchor nodes. Other nodes may then listen to the broadcasts of nearby anchor nodes and determine their own position.

3.4.3.5 Delay Tolerant

The unique MWSN case of DTNs poses the additional challenge of sparsity, from having nodes spread thinly over a potentially very large network area. Also, the fact that a path from a sensor to the sink occurs infrequently, requires very different protocols to those that have been discussed previously. RED (Replication-Based Efficient Data Delivery Scheme) [3.60] is a MWSN protocol designed for these situations, which uses a delivery probability as a gradient metric. The delivery probability is based on how successful a node is at passing messages to other nodes; if a node successfully transmits a message, its delivery probability increases, however if after a certain time it hasn't completed a transmission then its delivery probability decreases. RED also uses erasure coding to split the message into smaller blocks, where the number of blocks is based on the delivery probability. In this way, the sink may recover the complete message from receiving a subset of the blocks and the number of blocks created increase their probability of making it to the sink. FAD (Fault Tolerance-Based Adaptive Data Delivery Scheme) [3.60] is similar to RED as it uses the same delivery probability gradient. It also reintroduces a packet to the queue after it has been transmitted based in order to increase redundancy. FAD uses a fault tolerance metric for each packet to determine which packet should be transmitted next and which should be dropped if the queue is full. The protocol

proposed in [3.61] takes RED and adds a measure of direction, such that if the node is moving away from the sink, its probability gradient is significantly reduced.

3.4.3.6 Gradient Routing and Blind Forwarding

In general, the literature suggests that the most suitable protocols for mobile networks are those with the least overhead, which makes gradient routing a particularly applicable technique. As mentioned previously, GPSR uses location awareness to establish a gradient metric, however one drawback to this is in the reliability and power consumption required to obtain the geographic information and the dead end problem.

The static WSN protocol GBR (Gradient Based Routing) [3.62] initially establishes a hop-count metric at each node, such that data can be forwarded down the gradient to the sink. This is done in a setup phase by flooding the network, and then nodes share their hop-counts with their neighbours so nodes can choose the most appropriate next hop. In MWSNs the movement of nodes would require the network to be flooded periodically to keep the gradient updated, which would cause significant network congestion.

Additionally, it should be noted that gradient routing has also been adapted for MANETs using landmark nodes. HIR (Hop ID Routing) [3.63] maintains a hop-count gradient from each node to a set of landmark nodes, which then acts as an addressing scheme. This is similar to the gradient based schemes used in delay tolerant networks as described above, although, they use a probability metric instead of a hop-count.

However, the issue of maintaining an up-to-date gradient is still a concern, especially when low overhead is desired and no location information is available.

In static WSNs, DBO (Directed Broadcast with Overhearing) [3.64], uses gradient routing but with the technique of blind forwarding. Blind forwarding is a method of forwarding data in a gradient based routing scheme, in which a transmitting node will not select a single forwarding neighbour, but blindly broadcast its packet to all neighbours within range. Each of the receiving nodes can then compare their own gradient to that in the received packet and decide whether they should forward the packet or not. The technique of blind forwarding has been used in multiple static WSN protocols [3.64, 3.65, 3.66, 3.67] and can be seen as far back as 1991 [3.68], however is has previously never been implemented in a MWSN.

This style of routing is somewhat opportunistic in its nature, however in opportunistic routing subsets of forwarding nodes are defined, either at the source or on a hop-by-hop basis. Then the forwarding node is selected through a coordination process usually based around an acknowledgement (ACK) scheme or request-to-send/clear-to-send (RTS/CTS) handshake [3.69]. Contrastingly, blind forwarding considers all nodes to be potential forwarding nodes so there is no forwarding node set determination. Also, the technique doesn't require any

forwarding node coordination, which reduces overhead by eliminating any need for ACKs or RTS/CTS communication.

3.4.3.7 Secure Routing

Network security is always a concern when implementing a new system as well as during its deployment as threats can change and adapt rapidly. One of the first static sensor network routing protocols that took security into consideration was SPINS (Security Protocol Optimised for Sensor Networks) [3.70], which consists of two parts; μTESLA and SNEP. SNEP uses RC5 in counter (CTR) mode to provide confidentiality as well as a synchronised counter to give semantic security, replay protection and weak freshness. It also uses the well-known CBC (cipher block chaining) message authentication code[1] for authentication. μTESLA is an adaptation of TESLA [3.71] for resource constrained hardware, which provides authenticated broadcasts for routing information.

INSENS (Intrusion Tolerant Routing Protocol for Wireless Sensor Networks) [3.72] is another secure routing protocol for static WSNs and also chooses RC5 as its encryption method. INSENS initially performs route discovery, in which one way functions and the standard CBC message authentication code is used to verify the authenticity of the routes. INSENS records multiple routes from the sensors to the sink, in order to be able to route around security threats.

TinySec [3.73] is a secure routing protocol designed to be used with the TinyOS [3.74] framework for sensor motes. The protocol also uses the common CBC message authentication code, but contrastingly, Skipjack encryption is chosen over RC5, favouring lower memory over higher speed. TinySec has two modes; the first provides authentication only, whereas the second performs both authentication and encryption.

SIGF (Secure Implicit Geographic Forwarding) [3.75] is also designed for static sensor networks and is based on IGF (Implicit Geographic Forwarding) [3.76]. IGF uses location information to opportunistically forward data to a node closer to the sink, within a fixed angular range. SIGF provides three levels of security on top of this, which can then be selected based on the application. The first level simply lets the forwarding node be chosen randomly within the angular range, which lessens the probability of choosing a malicious node. The second level introduces a reputation metric, which is designed to identify nodes that are behaving in an unusual way. The metric is calculated individually by each node, about every other node, based on the overheard transmissions from a particular node. This includes recording the number of messages sent, number of messages forwarded, its claimed location and the delay between

[1] It is common in the literature to abbreviate ‘message authentication code’ to MAC, however, because of the ambiguity between this and ‘medium access control’, ‘message authentication code’ will always be written out in full and never abbreviated.

receiving and forwarding a message. The nodes will use this information to calculate forwarding success, fairness, consistency and performance, each of which are weighted and used to create a reputation value. Subsequently, if a nodes neighbour is calculated to have a reputation value below a set reputation threshold, then it is disregarded as a potential forwarding node. The third security level adds some encryption, message authentication code and sequence numbers to further secure the protocol.

mGKE (Mobile Group-based Key Establishment) [3.77] is a key distribution scheme for static sensor networks with mobile collectors, which uses unique private keys for every pair of node wishing to communicate. The sensors are arranged into groups, where each node is given a key for every other node in its group. Additionally, some nodes are given keys to allow them to communicate with nodes in other groups. In this way, two nodes in different groups that don't share a key, will have to use two other nodes that can communicate across the groups, as intermediaries in order to agree on a unique key. In the limited memory version of this protocol, the mobile collectors are also preloaded with a number of keys for nodes in different groups. They may then agree on a key for other nodes in that group, using the holder of the known key as an intermediary.

The security scheme proposed in [3.78] also considers the situation of a static sensor network with mobile sinks and utilises the key agreement method proposed in [3.79]. The key agreement method relies on knowing a shared polynomial. As such the protocol defines two sets of polynomials, those for stationary sensor nodes and those for mobile nodes. The mobile polynomials are distributed among the mobile sinks and selected static access nodes. The stationary polynomials are distributed between the fixed position sensor and access nodes. As such, stationary nodes will be able to communicate using the stationary polynomials to agree on keys, whereas mobile sinks will request data from a sensor node, via an access node using the mobile polynomials.

In terms of secure routing protocols designed to cope with the dynamic topology of a MWSN, [3.80] proposes a hierarchical protocol based on SHIVA [3.81]. Initially, nodes are assigned to a cluster based on their geographical location and then the base station nominates cluster heads based on each nodes distance to the sink, residual energy level, degree and number of times nominated previously. Once established, sensor nodes in the cluster transmit data to their cluster head, who then forwards the aggregated data from the cluster to the sink via other cluster heads if necessary. The routes from the cluster heads to the sink are determined by the base station, who can then instigate the changing of routes to increase energy efficiency. Cluster heads are in charge of TDMA slot allocation to the sensor nodes in their cluster. For the purposes of security, this protocol defines two set of keys; one for use between sensor nodes and their cluster heads and one for use between the cluster heads and the sink. These keys are

generated and distributed by the base station. The proposed protocol adopts the technique of threshold key cryptography [3.82], which means that a node must receive a certain number of keys before it can decrypt anything. Subsequently, any potential attackers must also receive a certain number of keys. Also, by allowing the base station to regularly change the routes for each sensor node, any malicious nodes that become part of a route will not remain in the route indefinitely, so its impact is limited.

Also for MWSNs [3.83] proposes a key distribution technique using nodes' post-deployment location information. Initially, each node is given a subset of the complete set of key units, which are made up of a key and a unique geographic location associated with that key. Then, once the nodes are deployed, they compare their current location with the location associated with each of the known keys. The distances are then used to prioritise key units with locations closest to the nodes actual location. Neighbouring nodes then identify the key units that they both share by swapping the location information associated with key, such that the key itself is not revealed. Subsequently, the common key units, the locations of both nodes and their IDs are combined, and using a one-way hash function, both nodes generate a new unique shared key allowing them to communicate securely.

The MWSN security architecture proposed in [3.84] utilises the pairwise key agreement algorithm from [3.79], which enables any pair of nodes to establish a private key. However, in this case a certificate authority is required. The establishment of a key occurs with a nodes neighbours upon deployment and when a node receives a message from a node for which it does not have a key. This architecture selects the Serpent [3.85] cipher for its encryption, in CTR mode, and for a CBC message authentication code.

3.4.3.8 Conclusions

The literature concerning routing is vast, even when narrowed to only that which concerns MWSNs. Overall, the inhibiting factor of many proposed protocols is in the large amounts of overhead causing congestion. This congestion can cause significant delays, which can accumulate in high traffic scenarios. These large delays not only prevent the timely delivery of data, but can cause packet loss and create routing errors from out-of-date information. In hierarchical protocols, additional overhead comes from nodes changing clusters and having to become re-associated with a new cluster head. In proactive protocols it comes from the large amounts of data that frequently needs to be flooded in order to keep routes up to date. Reactive protocols tend to be preferred, however they still require some overhead in route discovery.

Gradient routing seems to be a technique that is highly suitable to the many-to-one nature of a sensor network and in combination with blind forwarding, it may yield a particularly low level of overhead from the fact that no handshakes are required. Additionally, one of the crucial

features of blind forwarding is in its ability to adapt to topology changes; as no route is defined, the nodes can simply choose the most appropriate next hop in a completely ad hoc manner. The technique also creates multiple routes, which will improve reliability. These unique features make blind forwarding one of the best techniques for routing data in a MWSN. However, it still requires that the issue of gradient maintenance is solved and does not incur further overhead so as to nullify the advantages of blind forwarding.

With regard to security, the literature shows that there has been some work on securing sensor networks, however it has mostly been focused on static WSNs. The work on MWSNs that does exist, tends to use key agreement schemes to establish private keys between each pair of nodes that wish to communicate. Contrastingly, [3.80] uses a shared key between clusters of nodes, which reduces the memory needed for storing numerous keys and reduces the time and overhead involved in key agreement protocols. In general, the vast topic of information security in the unique application of MWSNs has only received a small amount of attention and further investigation will be needed.

3.5 CONCLUSION

This chapter has given a thorough review of the existing literature relating to MWSNs, primarily in terms of MAC and routing protocols. The routing protocols were also categorised and conclusions were drawn in terms of the most suitable routing styles. The next chapter will identify the techniques used to evaluate routing protocols, with a specific focus on MWSNs.

CHAPTER 4

RESEARCH METHODOLOGY

4.1 INTRODUCTION

When designing and testing routing protocols for MWSNs, it's important to choose the most useful measures of performance to describe their characteristics. It is also necessary to evaluate these protocols with appropriate techniques, which is often by mathematical analysis, simulation or experimental testbed. Each of these methods have their own advantages and disadvantages in terms of cost, evaluation time and accuracy. In this chapter performance metrics are identified before the three evaluation methods are considered and discussed in detail.

4.2 PERFORMANCE METRICS

A good performance metric should be clear and well defined, in order to avoid ambiguity and confusion. They should also characterise a single aspect of the protocol, in order to give useful and transferable information. One of the most important characteristics of a system is the packet delivery reliability, which is commonly described using packet delivery ratio (PDR). PDR is generally well defined and accepted in the literature as the ratio between the number of data packets received at the sink and the number of data packets generated by the sensors,

$$PDR = \frac{P_{rx}}{P_{tx}} \tag{1}$$

where P_{rx} is the number of packets received at the sink and P_{tx} is the number of packets introduced into the network. Losses may occur from events such as collisions, buffer overflow or node drop out. This metric gives a clear indication as to the quality of the protocols used and its unambiguity makes it a good point of comparison between other protocols.

Average end-to-end delay is also another important measure and is commonly defined as the average time between a packet being introduced into the network and it being received by the sink. This means it will take into account queuing delay, transmission delay, overhead amongst others. It's described as

$$Average\ \ End - to - End\ \ Delay\ = \frac{\sum_{i=0}^{P_{rx}} d_i}{P_{rx}} \tag{2}$$

where d_i is the end-to-end delay of packet i. In many systems this is considered to be a highly important measure, since long delays can cause the usefulness of the data to expire. Also, real-time systems will have very strict timing requirements and may additionally need measures of maximum end-to-end delay, in order to guarantee performance.

Routing overhead can often give a good indication of a protocols performance, since a large amount of overhead can increase delays, as well as use up a lot of energy and bandwidth. There are two main types of overhead; control overhead and packet overhead.

Control overhead can be defined as the fraction of time the network is passing control messages:

$$Control\ Overhead = \frac{t_{con}}{t_{con} + t_{data}} \tag{3}$$

where t_{con} is the time spent transmitting control messages and t_{data} is the time spent transmitting data packets. In reactive protocols the control overhead will mostly be from route discovery and route maintenance, whereas in proactive protocols it will mostly come from the distribution of topology data.

Packet overhead is generally defined as the fraction of bits in a packet that aren't data, so

$$Packet\ Overhead = \frac{L_P - L_{data}}{L_P} \tag{4}$$

where L_{data} is the number of data bits in each packet and L_P is the total size of the packet. Packet overhead is usually made up of node and packet IDs, routing lists and other information.

Throughput is a popular metric, however its definition tends to vary, which can make it misleading and difficult to compare across protocols. The expression given here defines it as the rate at which data bits are received by the sink;

$$Throughput = \frac{\sum_{i=0}^{P_{rx}} L_i}{t_{total}} \tag{5}$$

where L_i is the number of data bits in the i^{th} delivered packet and t_{total} is the total amount of time over which the data was collected. Other definitions have measured the rate of the total number of bits delivered to the sink, or alternatively the rate of packets delivered to the sink. Similarly goodput suffers from the same ambiguity, as such, if used, they should be clearly defined in text.

The throughput can also be deceptive; for example, taking two protocols with the same throughput, one may have a high PDR but a low end-to-end delay, whereas the other may have a low PDR but a high end-to-end delay. For this reason, it is often more useful to consider PDR and average end-to-end delay than throughput.

In many systems energy is a limited resource and so some applications may have strict power consumption requirements. In terms of average power consumption per second, energy efficiency can be given as

$$Average\ Power\ Consumption\ per\ Second = \frac{\sum_{s=0}^{S_{max}} t_s \cdot W_s}{t_{total}} \quad (6)$$

where t_s is the time spend in state s, W_s is the power consumed per second whilst in state s and S_{max} is the maximum number of states. Examples of states could be transmitting data, receiving data, carrier sensing or sleep. The energy efficiency can also be described as power per bit or per packet by replacing t_{total} with the total number of successfully received bits or packets respectively. Due to varying hardware power consumption it is a very difficult parameter to evaluate and compare. Another potential metric would be network lifetime, which is usually measured by waiting for a set number of nodes to fail. When the fraction of failed nodes has exceeded the predetermined threshold, the network is then considered to be dead. This metric is highly correlated with the systems energy efficiency, but is much more scenario specific.

4.3 Evaluation Techniques

There are three main techniques used to evaluate routing protocol performance, namely mathematical analysis, simulation and experimental testbeds. Mathematical analysis can be complex to derive, but expressions are usually easy to evaluate with the right software. Also, a large number of assumptions are often required, which can limit the accuracy of the analysis. Simulation is often more accurate, but requires specialised software to run. Also, the time taken to evaluate a simulation may be longer than a mathematical analysis, but will be considerably less than real-time. Coding protocols is often a lot easier in a simulation environment than when working with hardware in a testbed. Testbeds give the most accurate results since it is a real implementation, however the hardware and space to run the experiment can be expensive. Evaluation time is also much longer, since the use of hardware provides many more opportunities for bugs to arise, also experiments must be run in real time.

4.3.1 Mathematical Analysis

There are many mathematical modelling techniques that can be applied to MWSNs in order to predict protocol performance. The characteristics of the system that an expression describes can be split into two categories; protocol analysis and scenario analysis. Protocol analysis looks at aspects of the protocol such as PDR and delay, whereas scenario analysis focuses on the environment in which the protocols are deployed.

4.3.1.1 Scenario Analysis

The scenario in which a protocol is deployed has a large impact on the performance of a system. So, these expressions can generally be used to assess parameters of the scenario with regard to their effect on protocol performance. Additionally, they can be used to design scenarios in which to test or deploy MWSNs. A key metric is node order [4.1], which is defined as a nodes expected number of one hop neighbours, N_n, and for a uniform distribution is given as

$$N_n = \left(\frac{\pi r^2}{L^2}\right)(n-1) \tag{7}$$

where r is the transmission radius of the nodes, L is the length of one side of a square network area and n is the total number of nodes in the network. Additionally, this can also give the probability that a node has no neighbours [4.2], N_e:

$$N_e = \left[1 - \left(\frac{\pi r^2}{L^2}\right)\right]^{(n-1)} \tag{8}$$

From this it can be determined whether a network is well connected, so there is a low chance of a node becoming disconnected, or sparse, where there is a high chance of a node being disconnected. The expected number of available single hop communication links between nodes in the network at any time [3.5] can be derived as, $E(N_C)$:

$$E(N_C) = \left(\frac{\pi r^2}{L^2}\right)\binom{n}{2} \tag{9}$$

Link lifetime is a valuable metric since it describes how quickly the topology is changing. The average link lifetime, t_{av}, is the amount of time a link between two nodes is expected to last and is given by

$$t_{av} = \frac{4r}{\pi V_{max}} \tag{10}$$

where V_{max} is the maximum speed of the nodes. [4.3] considers the length of time that two nodes will be within transmission range of each other as

$$t_{link} = \frac{D_{link}}{|\vec{v}|} \tag{11}$$

If we consider two nodes, *A* and *B*, then D_{link} is the distance which node *A* travels whilst in range of node B. $|\vec{v}|$ is their relative velocity, which gives the duration of the link, t_{link}, as above. If the nodes are stationary or moving at the same speed in the same direction, then their relative velocity will be zero. At the other extreme, if they are both moving away from each other at

their maximum speed, then their relative velocity will be $2v_{max}$. Assuming that their speeds are uniformly selected then the average relative velocity will be v_{max}.

For simplicity it is assumed that any node passing through the transmission radius of another node, travels along a fixed trajectory at a fixed speed as shown in figure 4.1. Using these assumptions D_{link} can be calculated as $2r|cos(\alpha)|$ and if it's assumed that α is uniformly distributed, then the expected value of D_{link} is $^{4r}/_{\pi}$. Taking the average value of $|\vec{v}|$ gives v_{max}, and combining it with the expected value of D_{link} gives t_{av} as above. However, since nodes don't just travel in a straight line at a set velocity, this expression will slightly underestimate the average link lifetime.

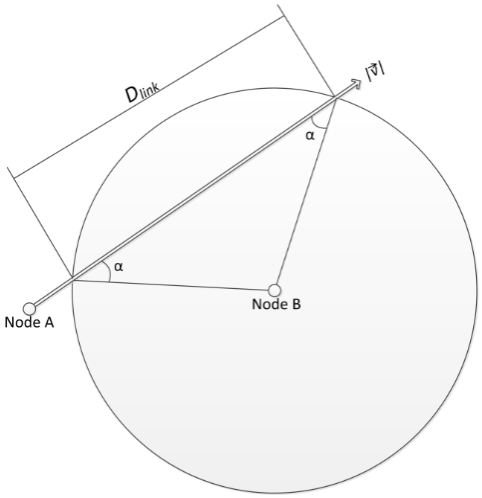


Fig. 4.1. Node A's trajectory through the transmission radius of node B.

[4.1] states that the expected distance between any two nodes, d_{av}, on a square plane is

$$d_{av} = L\frac{\sqrt{2}}{3} \tag{12}$$

Then to determine the expected distance of a single hop using a greedy algorithm, the transmitting nodes transmission area is split in to N_n segments, such that there is one potential forwarding node per segment. This is shown in figure 4.2 for an N_n of three. The forwarding node in the segment facing the sink is expected to have an angle of $\pi/(2N_n)$ and an expected distance of $(r\sqrt{2})/2$. This will give an expected one hop distance, d_{hop}, of

$$d_{hop} = \frac{r\sqrt{2}}{2}\cos\left(\frac{\pi}{2N_n}\right) \tag{13}$$

Using this, the expected number of hops, h, is given as

$$h \geq \frac{d_{av}}{d_{hop}} = \frac{2L}{3r \cdot \cos\left(\frac{\pi}{2N_n}\right)} \tag{14}$$

In [4.1] it is also stated that this expression is most accurate when $N_n \geq 5$.

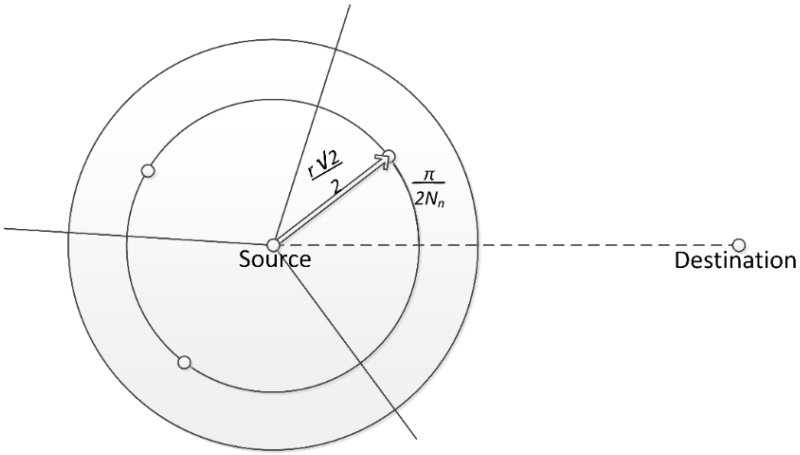


Fig. 4.2. Expected distance of a single hop.

4.3.1.2 Protocol Analysis

Markov models [4.4] are a useful tool in mathematical modelling and MWSN systems using a flat protocol can be described using the simple model shown in figure 4.3. This model assumes that a packet can have three states; in an intermediate sensor node, successfully delivered to the sink or lost. Since all of the nodes in a flat protocol are performing the same function, they can all be represented by a single state, which packets can be fed back into. The sink and lost states are absorbing, since packets that enter these states are never reintroduced to the network.

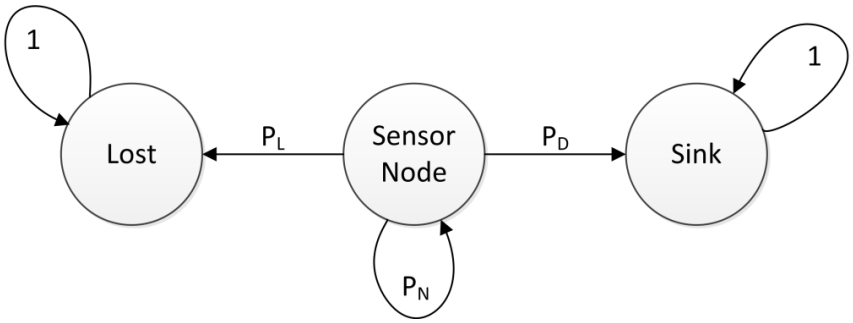


Fig. 4.3. Simple Markov model.

The probability of a packet being lost is labelled P_L, the probability of a packet being delivered to the sink is labelled P_D and P_N is the probability that the packet is passed to another node. P_L will be the sum of all potential causes of packet loss, such as channel errors, collisions or node failure. There may also be losses from a queuing buffer overflow, a location aware protocol reaching a dead-end, a routing protocol being unable to find a path or the packets TTL (time to live) expiring.

The Markov model can be described as a transitional matrix, P_M, in canonical form:

$$P_M = \begin{bmatrix} P_N & P_D & P_L \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Taking P_M to the power k, will give the probabilities of the packet being in those states after k transitions, which is given as

$$P_M{}^k = \begin{bmatrix} P_N{}^k & \frac{P_D\left(P_N{}^k - 1\right)}{P_N - 1} & \frac{P_L\left(P_N{}^k - 1\right)}{P_N - 1} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

Since the model is expected to reach a steady state, where all packets are either lost or delivered, the limit of $P_M{}^k$ should be computed. Knowing that $P_N < 1$, means that as k tends to infinity, $P_N{}^k$ will tend to zero. Also, since $P_N+P_D+P_L=1$, it is true that

$$\lim_{k\to\infty}\left(P_M{}^k\right) = \begin{bmatrix} 0 & \frac{P_D}{P_D + P_L} & \frac{P_L}{P_D + P_L} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & PDR & PLR \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

This leaves final expressions for both the PDR and packet loss ratio (PLR), which intuitively state that all packets will be split proportionally between the sink state and the lost state, based on the ratio of their probabilities.

$$PDR = \frac{P_{rx}}{P_{tx}} = \frac{P_D}{P_D + P_L} \tag{18}$$

$$PLR = 1 - \frac{P_{rx}}{P_{tx}} = \frac{P_L}{P_D + P_L} \tag{19}$$

Additionally, the discrete time steps of the Markov chain are analogous to the hop delay, D_h. D_h is defined as the length of time between a node receiving a packet and then transmitting it again. This takes into account things such as the queuing delay, route discovery and processing time.

Overall, end-to-end delay, D_{av}, can be calculated as

$$D_{av} = D_h \cdot h \tag{20}$$

Often, the largest delay at a node is the queuing delay, which occurs when a node has multiple packets to service. From queuing theory [4.4], assuming a the packets arrive at a node conforming to a Poisson distribution, each node can be modelled as a M/G/1 first come first serve (FCFS) queue. In this way the average queuing delay, T_q is given by the Pollaczek-Khinchine formula [4.5]

$$T_q = \frac{\lambda_s \mathrm{E}\left(T_s{}^2\right)}{2(1 - \lambda_s T_s)} + T_s \tag{21}$$

where λ_s is the arrival rate of packets to the node and T_s is the packet service time. T_s is directly related to the MAC protocol used, since a packet cannot be transmitted until the MAC layer

permits it. $E(T_s^2)$ is defined as the second moment of the service time, so for an M/G/1 queue this would be equal to $T_s^2+Var(T_s)$. Accordingly, for an M/M/1 queue $E(T_s^2)= 2T_s^2$ and for an M/D/1 queue $E(T_s^2)= T_s^2$.

Network load can also be a valuable parameter in determining whether a network is likely to become congested or not. N_L is the number of packets per second generated by the network:

$$N_L = f_p \cdot n_s \tag{22}$$

where f_p is the rate per second at which a single node produces packets and n_s is the number of nodes able to generate data. Also, assuming that firstly the network has a single sink, that all generated packets are delivered successfully and that the load is split equally amongst the sinks neighbours, then

$$L_{sn} = \frac{N_L}{N_n} \tag{23}$$

where L_{sn} is the number of packets needing to be serviced by a single sink neighbour per second. By knowing the service time for a single packet, T_s, a measure of network stability, ρ_N, can be derived

$$\rho_N = L_{sn} \cdot T_s \tag{24}$$

Thus, if ρ_N is greater than one, the sinks neighbouring nodes are likely to be receiving more packets than they can service, making the system unstable.

Since the extreme lower bound of a routing protocols PDR is zero and the extreme upper bound on end-to-end delay is infinite, it may be useful to analyse an unintelligent protocol in order to give some expected performance bounds. A simple protocol with perfect channel conditions and a collision free MAC will be looked at in order to determine expected worse case performance bounds for MWSN routing protocols. The protocol simply delivers the packet when the node is within range of the sink, otherwise it will select a random neighbour to forward it to. If the node finds itself disconnected from the network, meaning it has no neighbours, then the packet will be dropped.

Firstly, the probability that a node is in range of the sink will become P_D and is given as

$$P_D = \frac{\pi r^2}{L^2} \tag{25}$$

The probability of a node being disconnected, is the probability of packet loss, which is the same as N_e, but removing the possibility of the node being connected to the sink:

$$P_L = \left[1 - \left(\frac{\pi r^2}{L^2}\right)\right]^{(n-2)} \tag{26}$$

This leaves P_N as

$$P_N = 1 - P_D - P_L \tag{27}$$

As an intelligent protocol is anticipated to be better than this, by taking the formula for PDR an expected minimum bound on the successful delivery of packets can be determined. With the simple protocol, the expected number of hops is equal to the expected number of transitions before ending up in the absorbing sink state:

$$h = \frac{1}{P_D} = \frac{L^2}{\pi r^2} \tag{28}$$

Since this protocol is transmitting packets to all neighbours and a packet is only delivered when the transmitting node happens to be in range of the sink, it is expected that any intelligent protocol will have a lower average hop count. The intelligent protocol is presumed to send packets down a more direct route to the sink, making the number of hops before delivery better than chance. This expression differs to the equation for *h* given previously, since the previous expression assumes a greedy forwarding method. To determine bounds for end-to-end delay, the expected number of hops can be used with the expression for D_{av}.

4.3.2 Simulation

One of the most popular ways of evaluating protocols is through using a software based simulation tool. There are multiple available software packages, which are generally based on the use of discrete event simulation [4.6]. In this way events are dealt with sequentially in the order in which they are scheduled to occur. An event may be a packet being created, transmitted, received or the movement of nodes. Generally, simulation tools will provide a selection of protocols, which have already been implemented, however currently it is rare to find any available protocols designed for MWSNs being freely distributed since there is no standard and there are very few commercial deployments of MWSNs using these protocols.

4.3.2.1 Simulators

Potentially the most popular simulation tool is ns-2 (Network Simulator 2) [4.7]. ns-2 uses C++ to model nodes and then oTcl scripts to define other simulation parameters, such as topology and mobility. It is open source, has quite a large footprint, limited GUI and can be difficult for beginners. Recently the entire ns-2 suite has been refurbished to give ns-3 [4.8],

which removes the use of the oTcl scripts and improves performance in terms of computation time and memory usage. It should also be noted that models designed for ns-2 are not directly compatible with ns-3 and will have to be ported manually.

Another available simulator is OPNET modeller (Optimised Network Engineering Tools Modeller) [4.9], which is commercially available or free for academic use. Models are written in C or C++ and designed within an easy to use GUI. QualNet [4.10] is also a commercially available simulator, which uses a C based language developed by UCLA called Parsec. The open source predecessor of QualNet is GloMoSim (Global Mobile Information System) [4.11], which has reduced scalability and a more limited GUI.

Alternative to the C based simulators, J-Sim (Java Simulator) [4.12] is written in Java and models everything as components, which are then linked through ports. JiST (Java in Simulation Time) [4.13] is also a Java based simulator, however it doesn't provide mobility support. Consequentially, JiST is often used together with SWANS (Scalable Wireless Ad-hoc Network Simulator) [4.14], which is designed to simulate MANETs through JiST. This setup has been shown to give excellent computation time but at the expense of significant memory usage [4.15].

Though this section is not encompassing of every available simulation tool, it has mentioned some of the most commonly used both commercially and in research communities.

4.3.2.2 Modelling

The modelling of a MWSN consists of two main areas; the modelling of the protocols used and the designing of the scenario in which the network is deployed. The modelling of a protocol requires the accurate translation of the concept into a language understood by the simulation software, or hardware in the case of a testbed. This can present problems if the initial concept is not thoroughly thought through or if some aspects have been overlooked. Overall, each model is very specific to the protocol being simulated and the software being used.

Also, the choice of scenario parameters is often specific to the application, which will define the nature of the scenario in which the system will be implemented and affect the performance of the protocol.

Network density will alter how well connected a network is, which can be measured by the node order, using the equation given in the previous section. Essentially, this is based on the number of nodes, the network size and the transmission radius. A sparse network will make it difficult for nodes to deliver packets since the chances of having a good forwarding neighbour is reduced. Contrastingly, a very dense network will mean that a lot of nodes will be contending for the same medium, which can cause long delays.

The mobility of a network is used to describe how frequently the networks topology will change. It can be characterised by the expression for average link lifetime from the previous section, which is a function of the speed of nodes as well as the transmission radius. If the mobility is very high then paths will not last for very long, making routing difficult with packets reaching dead-ends and being dropped. Also, topology changes can create overhead from new route discovery or proactive topology information dissemination, which can cause congestion, creating large delays and causing large amounts of overhead and potentially packet loss.

The network load, as described in the previous section, is also a critical element of the scenario and is made up of the number of nodes generating data and how frequently they are generated, the size of these packets should also be taken into account. A high load will cause congestion if there is insufficient bandwidth, which will cause long end-to-end delays and potentially packet loss from memory saturation or timeout events.

The effects of these parameters hold true for any deployment, whether in a simulation environment or not, however in a testbed they are often determined by the cost, availability and limitations of the hardware. Also, the space in which the nodes are deployed and any application that is implemented, will contribute to the actual values for each of these parameters.

4.3.3 Experimental Testbed

One of the most accurate ways to evaluate the performance of a MWSN system is to implement it within an experimental testbed. Testbeds are made up of a number of mobile nodes deployed in a controlled environment. Often the choice of equipment is based on the cost, available space and any intended application.

4.3.3.1 Node Architecture

Generally, nodes consist of at least a microcontroller, power source, sensors, radio transceiver and mobility platform as depicted in figure 4.4. The microcontroller is responsible for the execution of the code, control of the radio and sometimes the mobility as well. The microcontroller used will impose restrictions on computational speed as well as program and data memory. The power source is often from a battery, which will restrict the available lifetime of the node. However, it is likely that mobile nodes will be able to seek out a power source for recharging or, alternatively, they may have an on-board generator utilising something like solar, motion or vibration power to recharge the batteries. The size of the battery is usually limited by the weight allowed by the mobility platform. In terms of sensors, the options are vast with most nodes offering ambient light and temperature as standard. It's also common for nodes to have additional, unused ports with analogue to digital converter capabilities, such that any type of sensor can be incorporated into the device. The radio transceiver is controlled by the microcontroller and provides the link between nodes. The choice of transceiver will create

limitations in terms of bandwidth and transmission range, also, depending on the mobility platform, the transceiver may be the largest consumer of power. Alternatives to RF include infrared and ultrasound, however these have some propagation limitations and often require a line-of-sight connection. The mobility platform is responsible for transporting the node and in a testbed may be from robots moving in predefined patterns within a network area. Testbeds have also been known to use cars or pedestrians to carry nodes and follow set patterns of movement. Contrary to the mote orientated hardware, it is also common for nodes to be implemented on personal computers, providing they are equipped with the necessary resources.

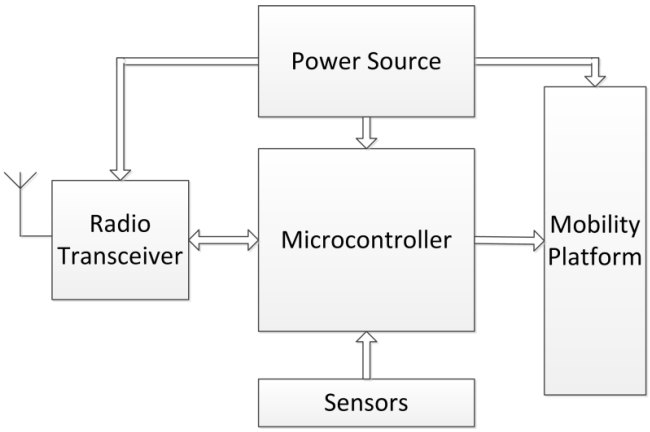


Fig. 4.4. Overview of node architecture.

4.3.3.2 Hardware

There are a multitude of commercial hardware options available from various vendors including the Texas Instruments (TI) eZ430-RF2500 development tool [4.16]. It's one of the cheapest modules available, consisting of a TI MSP430F2274 microcontroller and a TI CC2500 transceiver. The module is not designed specifically for WSNs and is USB based, which means that each node will need to be attached to a computer. In the context of a real sensor network deployment, this is not an ideal solution, however for a testbed it may be useful. For example in the APE (Ad hoc Protocol Evaluation) testbed [4.17] volunteers were enlisted to manually carry laptops, which then timestamp and log traffic. Though the eZ430-RF2500 doesn't come with any sensors the target board has 15 digital IO pins, 10 of which may be used as analogue inputs to the MSP430F2274s 10-bit analogue to digital converters. The microcontroller also has UART, SPI and I^2C capabilities as well as 32kbytes of flash memory and 1kbyte of RAM. The low power transceiver operates in the 2.4GHz ISM band and has a data rate of up to 500kbps. The transmitter will consume 21.2mA whilst transmitting at 0dBm and a minimum of 13.3mA when receiving. The microcontroller will typically consume 270µA whilst active and 0.7µA whilst in standby mode.

The popular TelosB mote [4.18] from Memsic is more expensive than the eZ430-RF2500, but is designed to be used as a standalone sensor node. It's controlled by a TI microcontroller

from their MSP430 range and offers 48kbytes of program flash memory, 1024kbytes of measurement flash memory and 10kbytes of RAM. The mote uses an IEEE 802.15.4 compliant radio, which is based in the 2.4GHz range and has a data rate of 250kbps with an outdoor range of up to 100m. It's powered by two AA batteries, without which it weighs 23 grams making it one of the heavier nodes. It also has the option of including integrated light, temperature and humidity sensors. In terms of power consumption, the microcontroller will consume 1.8mA whilst active and 5.1µA whilst in sleep mode. The transceiver will consume 23mA whilst in receive mode, 21µA whilst in idle mode and 1µA whilst in sleep mode.

Contrastingly to the previous two nodes, Memsics MicaZ mote [4.19] is based on the Atmel ATmega128L microcontroller, which provides 128kbytes of program flash memory and 512kbytes of measurement flash memory. Similarly so the TelosB mote, the MicaZ has an IEEE 802.15.4 compliant radio, with a data rate of 250kbps with an outdoor range of up to 100m. It also uses two AA batteries but weighs less the TelosB mote, at 18 grams. The option of a purpose built sensor board is also available, which can provide light, temperature, humidity, barometric pressure, seismic or acoustic sensors as well as GPS and a magnetometer. Additionally, the microcontroller consumes 8mA whilst in active mode and less than 15µA whilst in sleep mode. The transceiver consumes 19.7mA whilst receiving, 17.4mA when it's transmitting at 0dBm, 20µA in idle mode and 1µA in sleep mode.

Another popular choice of sensor node from Memsic is the IRIS mote [4.20], which uses the Atmel ATmega1281 microprocessor. This gives the mote 128kbytes of program flash memory, 512kbytes of measurement flash memory and 8kbytes of RAM. As with the TelosB and MicaZ, the IRIS mote also has a 2.4GHz, 802.15.4 compliant radio with a data rate of 250kbps, however its outdoor range is more than 300m. Like the MicaZ, the IRIS weighs only 18 grams, takes two AA batteries and has the same sensors available to it. As for power consumption the Atmel microcontroller consumes 8mA whilst active and 8µA whilst in sleep mode. The transceiver consumes 17mA whilst transmitting at 3dBm and 16mA when receiving.

The SHIMMER wireless sensor unit [4.21] is a complete integrated solution, designed for wearable applications. It is based on a TI microcontroller from the MSP430 range, offering 48kbytes of flash memory and 10kbytes of RAM. The radio is 802.15.4 compliant and the device also includes integrated 3 axis accelerometer, tilt/vibration sensor and Li-ion battery. It's also the smallest complete node at 53mm by 32mm by 15mm.

Overall, the eZ430-RF2500 is a cheap way to turn a set of laptops into sensor nodes, but would not be suitable for a real deployment. The SHIMMER nodes are ideal for wearable applications; however they're not quite as flexible as the Memsic nodes, whihch can be adapted to almost any application.

It should also be noted that none of the nodes come with any method of mobility. There are many options for this, including people on foot or in cars carrying the nodes around in a choreographed manner. Alternatively, quadcopters or remote controlled ground vehicles can have sensors attached and then be individually controlled by people. Another option would be to implement a system such as the Ardupilot Mega [4.22] on the remote controlled vehicles. Ardupilot Mega is an Arduino based system that implements an autopilot on remote controlled vehicles. In this way each sensor node can have their route preprogramed and the autopilot will guide the vehicle along the defined path.

4.3.3.3 Software

The most popular firmware designed specifically for sensor nodes is TinyOS [4.23]; of the motes mentioned above, TinyOS is utilised by the TelosB, MicaZ, IRIS and the SHIMMER. TinyOS is a free open source embedded operating system which uses a C-based programming language called nesC. The structure is based on allowing concurrency in an event triggered environment. Essentially, programs are split into multiple components, which can be either commands, events or tasks. Tasks are called by event handlers and commands, but are not necessarily executed immediately. Instead tasks are added to a stack and executed in a first-in first-out (FIFO) manner.

Another option is Contiki [4.24], which is also open source but, in contrast to TinyOS, it provides concurrency through the use of threading rather than a stack. Execution occurs as the interrupts are triggered based on a priority system. Contiki is modular in its architecture and is written in C. Lite-OS [4.25] is also modular, written in C and supports multithreading. However execution is done by scheduling each task in turn. Comparatively, MANTIS (Multimodal System for Networks of In-Situ Wireless Sensors) [4.26], is based on a layered architecture and is written in C. It uses threading and execution is done based on a priority system.

4.4 CONCLUSION

This chapter has focused on the various methods used in evaluating the performance of routing protocols in MWSNs. Descriptions of popular hardware, firmware and simulators have been given, as well as a discussion of the literature pertaining to existing test beds. Common and frequently used mathematical expressions have also been taken from the literature with some minor additions. The main contributions of this thesis will begin in the next chapter with the presentation of three novel routing protocols.

CHAPTER 5

PROPOSED ROUTING PRINCIPLES AND PROTOCOLS

5.1 INTRODUCTION

To contextualise the presented routing protocols and justify their design choices, this chapter firstly discusses some primary considerations relating to MWSNs. Following this, the key routing principles developed in this work are detailed. Based on this, three novel routing protocols are presented and described. Each protocol is introduced with a technical and operational description, as well as mathematical analysis. This chapter also draws comparisons between the designs in terms of their advantages and disadvantages.

5.2 DESIGN CONSIDERATIONS

Whilst the application of this work is wide and encompasses a large number of scenarios, for the purposes of making design decisions, some of the key considerations and their motivations will be described. The environment in which MWSNs may be deployed are widely varied and depend strongly on the application they are being used for. Primarily, the nodes are considered to be significantly more expensive than the traditional disposable motes used in a static WSN. An example of this is in the use of unmanned, autonomous vehicles, such as unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGVs) and unmanned underwater vehicles (UUVs). This type of platform can be used in applications such as extra-terrestrial environment mapping, UAV aided search and rescue and military reconnaissance. The added expense of these drones and the necessity for high quality in these state-of-the-art sensing platforms, means that they are expected to be reusable over multiple missions. This is unlike the traditional static WSN, in which nodes are cheap, disposable and often used only deployed once. Additionally, WSN nodes are prone to failure, however, the expensive, precision-made MWSN nodes are not expected to suffer from faults and should be able to last for the duration of the mission. Also, due to the cost, it is anticipated that fewer nodes will be used and, to avoid losing nodes, their power capacity should be sufficient to complete the mission and return to the base station to be recharged and redeployed. This view of expensive MWSN nodes, makes it likely that the equipment in a single network is likely to be owned and deployed by singular entities, who will then have full control of their entire network. These

institutions and private owners, are probably going to deploy these networks for a specific task. As such, it is not anticipated that foreign nodes will be required to join the network on an ad hoc basis and nodes are unlikely to leave the network before the mission is complete. This is in direct contrast to the case of MANETs, in which the majority of people own a compatible device and so nodes are expected to be frequently joining and leaving the network. Given these expectations, it is assumed that the number of nodes in the network will remain constant for the length of their deployment.

5.3 Routing Principles

The routing protocols presented in this section are based on some common concepts, which will be outlined here.

5.3.1 Gradient Routing

One of the key areas in choice of routing protocol for mobile scenarios, is that of overhead. In general, routing protocols with high overhead suffer in highly mobile environments due to the large amounts of communication required to discover routes or maintain up-to-date topology information. As such, low overhead solutions, such as opportunistic routing is preferable. One technique of opportunistic routing uses a gradient metric, which requires every node to maintain a value that indicates its distance from the sink. In this way, data may be forwarded down the gradient towards the sink. The critical issue with this technique is in the maintenance of the gradient metric in a mobile environment. If the metric cannot be kept up-to-date then the data will be forwarded incorrectly and may be lost. The most common solution to this problem is flooding the network, however this creates a large amount of overhead and will need to be done frequently to keep up with the changing topology. Consequently, if gradient routing is to be used in a mobile environment a better method of gradient maintenance will need to be developed.

5.3.2 Global Time Division Multiple Access

One key issue identified by the literature is the gradient maintenance problem, to which solutions often incur significant overhead or the use of location information. Given the assumptions above, specifically the use of a fixed number of nodes, it is possible to implement a GTDMA MAC layer. This means that each node in turn is allowed to transmit in a predefined order. So, all nodes are allocated their own timeslot in a cycle, where the timeslot length is the duration required to transmit a packet and the cycle is long enough that every node has a chance to transmit. This is illustrated in figure 5.1, which shows a single cycle of *n* timeslots, *n* being the number of nodes in the network. As the number of nodes is static, each node may be allocated its timeslot before the network is deployed, which means that it can be optimised for

every mission. This negates the need for additional overhead from a central controller allocating timeslots, as in a traditional dynamic TDMA MAC.

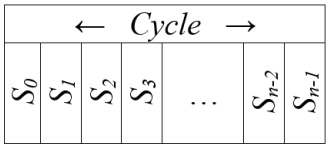


Fig. 5.1. GTDMA cycle structure, showing how a cycle is made up of n time slots. Slot S_0 will be assigned to the sink, S_1 will be assigned to the first sensor node, S_2 will be assigned to the second sensor node.

The main advantage of the GTDMA MAC in a mobile network is that each node will have the opportunity to overhear every other node and in this way a hop-count gradient may be maintained. If every node transmits its current hop-count in its timeslot, then a node may determine its own hop-count at the start of its timeslot. Given that by the time a nodes timeslot comes around it will have heard transmissions from all of its neighbours, so it will conclude that its own hop-count is the lowest overheard hop-count, plus one. For example, figure 5.2 shows a sink and 8 sensor nodes where all but one is labelled with its hop-count from the sink. Initially node *'A'* has just arrived to this part of the network and still has its previous hop-count. The GTDMA structure means that each node will transmit in turn so after listening to the broadcasts of its neighbours, node *'A'* will have heard the hop-counts 3, 4, 3, 2, and 1. From these values node *'A'* can take the lowest value and add one to determine its own hop count of 2, as shown in the second part of the figure. The third part shows how, on the next cycle, the node which originally had a hop-count of 4 has updated the value to 3, which reflects the new state of the network.

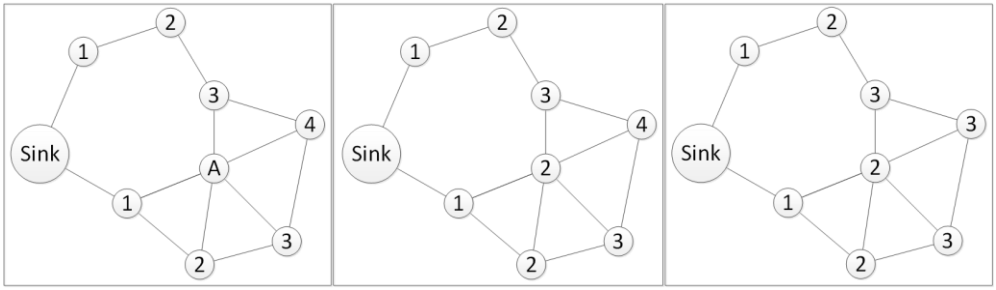


Fig. 5.2. Hop-count determination, in which node *'A'* has just arrived with an unknown hop-count. Then, from the broadcasts of its neighbours it derives its own hop-count and the nodes around it subsequently adjust their own hop-counts as well.

This particular MAC layer requires network-wide synchronisation, for which a number of techniques have been mentioned in chapter 3. One such method is RBS (Reference Broadcast Synchronisation) [3.25], which uses a third party to broadcast a high powered beacon that can be heard network-wide. Nodes may then synchronise themselves to the arrival time of the reference beacon packet. Within the GTDMA framework, the sink could be used to transmit this reference beacon, which could then regularly keep the network synchronised.

The novel use of the GTDMA approach allows nodes to regularly determine their hop-count by taking advantage of the cyclic nature of the GTDMA MAC. Consequently the issue of gradient maintenance is mitigated and, by only sharing local topology information, flooding the network is not required.

In the even that a node becomes disconnected from the network, it will not receive any messages from the other nodes. In this case, if a node doesn't receive any transmissions in a cycle, it will assume it is disconnected and stop transmitting packets until it becomes connected again. This both saves energy and reduces packet loss. One additional phenomena that occurs with this method of gradient updating, is when two or more connected nodes become disconnected from the rest of the network. In this situation, since the gradient is not being pinned down by the sink, the hop count of the other nodes will continue to increase indefinitely. Essentially, using a two node example, the first node broadcasts its hop-count, the second node will update its hop-count as one plus this value, then the second node will broadcast its new hop count and the first node update its hop-count to one plus this value. Subsequently, this process will repeat and the nodes hop-counts will continue to increase. For this reason a threshold is put on the hop-count, such that if the hop-counts increase too much, then the nodes will realise they have become disconnected and respond accordingly.

5.3.3 Blind Forwarding

By solving the gradient maintenance problem, hop-count based gradient routing solutions may be applied to MWSNs. One of the methods utilising the gradient metric is blind forwarding, which also has very low overhead, which makes it highly suitable to high mobility applications. As such, all of the protocols presented here use the blind forwarding technique, where transmitting nodes blindly broadcast the packet to their neighbours. Then, it's the responsibility of the receiving node to decide whether to forward the packet, which is done by using the hop-count gradient to determine whether the transmission came from a node that's closer or further from the sink. If the receiving node sees that it is closer to the sink that the transmitting node, then it will forward the packet. If the packet came from a node that is closer to the sink, then the packet will be dropped. As the choice to forward packets is made on a hop-by-hop basis, this technique is very suitable to highly dynamic topologies and the lack of handshaking reduces overhead.

5.3.4 Route Diversity

One key aspect of blind forwarding is route diversity, in which a packet may take multiple paths from its source to its destination. Since the blind forwarding technique doesn't allow nodes to select a single forwarding neighbour, multiple neighbours may decide to forward the packet. This will inherently create route diversity in the network. The main advantage of this is

in the protocols increased reliability, since if a packet is lost on one path, there is a chance that it may be delivered by another route. This makes blind forwarding a highly robust technique that is resilient to topology changes, link failure and node failure.

5.3.5 Priorities

By the nature of blind forwarding, packets may take multiple routes, which will increase the redundancy in the network. In order to control this nodes are only allowed to transmit a packet once. Additionally, to ensure that the correct packets are given priority, the use of a status bit is introduced. The status bit indicates a packets priority and designates it as either *priority data* or *diversity data*. When a node receives a packet it stores it in a queue, so before a nodes time slot it must decide which packet to transmit; packets with priority status are given precedence over those with diversity status.

The use of diversity packets is designed to increase the route diversity of the protocol without impeding the delay times of the priority packets. So, the diversity packets will increase the number of paths a piece of data may take to the sink, but priority packets will always be transmitted first. Additionally, the use of blind forwarding means that individual pieces of data are treated separately and may take multiple different paths through the network, which further increases robustness.

Based on the state of the status bit, the oldest pieces of data with priority status will be transmitted first, followed by the oldest diversity packets.

The determination of priorities is as follows:

- Data is initially given priority status by the node that generates it.
- If a node receives the data from a sender with a higher gradient, then it should be forwarded with the same priority as it was received.
- If a node receives the data from a sender with a lower gradient, then it should be dropped regardless of its priority.
- If a node receives priority data from a sender with the same gradient, then it should be forwarded with diversity status.
- If a node receives diversity data from a sender with the same gradient, then it should be dropped.

As an example, the use of priorities can be seen in figure 5.3, in which the eight nodes are labelled from *A* to *H*. In the first part of the figure, node *A* generates a packet and broadcasts it to its neighbours with a hop-count of 3 and priority status. Node *B* has a hop-count of 4 and as such drops the packet. Node *D* has a hop-count of 2 and stores the packet for forwarding with priority status. Node *C* has a hop-count of 3, which is the same as node *A*, so it stores the packet

with diversity status. In the second part of the figure, node *D* is shown to broadcast the packet, which will be dropped by node *A* and stored with priority status by node *F*. The third part of the figure shows how node *F* will also broadcast the packet, which is subsequently received by the sink. On the other path, node *C* will broadcast the packet with diversity status, which will be dropped by node *A* and stored for forwarding by node *E*. Node *E* will then broadcast the packet with diversity status, which will be dropped by node *C* and stored by node *G*. The final part of the figure illustrates node *G* broadcasting the packet with diversity status, which will be dropped by node *E* and received by the sink. Node *H* will also hear the transmission from node *G*, however as both nodes have the same hop-count and the packet has diversity status, node *H* will drop the packet.

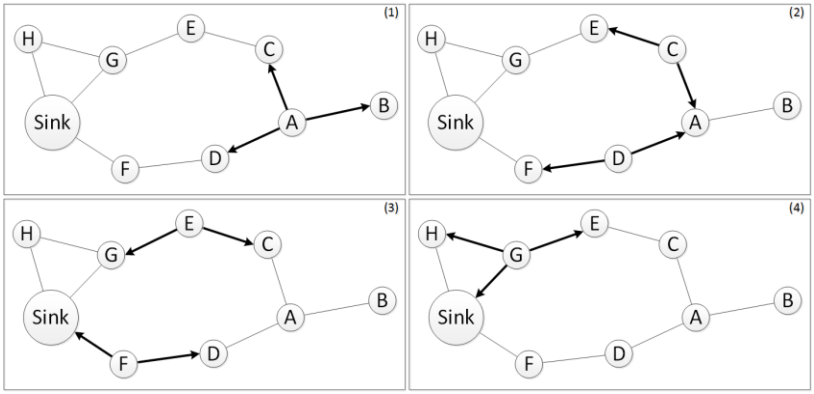


Fig. 5.3. Forwarding data with priorities; (1) Data is generated at node *A* and broadcast to its neighbours. (2) Based on the hop count gradient, only nodes *C* and *D* forward the packet. (3) Again due to the gradient, only nodes *E* and *F* forward the data. (4) Here, due to the use of priorities, only node *G* forwards the data.

5.4 PROACTIVE HIGHLY AMBULATORY SENSOR ROUTING (PHASER)

PHASeR (Proactive Highly Ambulatory Sensor Routing) is the first protocol presented in this thesis.

5.4.1 Protocol Description

PHASeR is designed for MWSNs in which the nodes are regularly generating data. In other words each sensor is assumed to be generating data periodically, with a fixed time interval. The protocol uses the GTDMA gradient maintenance and blind forwarding techniques, as described in section 5.3. As such, the nodes sampling frequency should dictate the frequency at which the timeslot cycle occurs. In this way a node will sample its sensor and transmit in its every timeslot, once every cycle.

Nodes will always be transmitting their own latest data in their timeslot, so in order to allow a node to forward data received from other nodes, a packet will have a variable capacity. The capacity of a packet will be expressed in *frames*, where the number of frames is the number of available data fields in a packet. The packet structure in figure 5.4 shows a two frame packet. The first frame, type 1, is always for the transmitting nodes data and protocol overhead and consists of four fields; the transmitting nodes ID, current hop-count and generated data, as well as an indication of which other nodes data is present.

Field Name	Node ID	Hop-Count	Generated Data	Forwarding Node IDs	Frame Data	Frame Priority	Frame Time Stamp
Size (bits)	$\lceil \log_2 n \rceil$	$\lceil \log_2 n \rceil$	L_{data}	n	L_{data}	1	$\lceil \log_2 n \rceil$
	|----------Frame 1----------|				|----------Frame 2----------|		
Total Size	$L_p = \lceil \log_2 n \rceil + F(L_{data} + 1 + \lceil \log_2 n \rceil) - 1 + n$						

Fig. 5.4. PHASeR packet structure, where n is the number of nodes, F is the number of frames and L_{data} is the required size of the data field for the application. F dictates how many frames from other nodes can be forwarded at a time. L_{data} dictates how much data is in each frame and may be sized to accommodate information from multiple sensors, geographical coordinates or the health of the node, depending on the application.

The second frame shown in figure 5.4, type 2, is made up of only three fields, the data, priority and time stamp. Depending on the number of frames required, a node may repeat the structure of the type 2 frame as needed. For example, if a node has the data from four nodes to forward it will fill the type 1 frame with its own data and then repeat the type 2 frame structure four times; once for each additional piece of data to be transmitted. The forwarding node IDs field indicates which nodes data is present in the packet, this is done using a single bit to represent each node. So, if the third and sixth bits were set high then it would indicate that there are two additional frames in the packet, containing data from nodes three and six respectively.

Since nodes are able to transmit data from multiple node in a single packet, future work may also look at using aggregation techniques to combine the received data and reduce the size of transmissions.

In order to calculate the minimum allowable timeslot length it is necessary to put an upper limit on the number of frames a node may transmit. This is non-trivial since nodes closer to the sink will require more space, whereas nodes further from it will need less. However, the varying topology means that a nodes requirement may change over time. As such, if the maximum number of frames is too low, the bottleneck effect will cause data to be lost. In contrast, if the maximum packet size is too large then there may be wasted bandwidth. Although, a large slot length will reduce the frequency of transmissions and save energy.

In order to determine an appropriate maximum for the number of frames, a metric, α is derived in *appendix A* and given as:

$$\alpha = 2^{1-n} \cdot \sum_{c=n-F}^{n-1} \binom{n-1}{c} \qquad (29)$$

where n is the total number of nodes in the network and F is the maximum frame capacity of a packet. The value, α, is the fraction of possible topologies that will not suffer from the bottleneck effect for a given n and F.

For example, in a network of 5 nodes there are 1024 possible topologies. In other words, assuming that all five nodes have at least one link, there are 1024 possible arrangements. By allowing each node to forward its own data and two other nodes' data, would give a frame capacity of 3. Out of the 1024 possible topologies there are 704 that would not suffer from the bottleneck effect. This means that there are 320 possible topologies in which the frame capacity of 3 would not be sufficient. This example gives α as 0.6875, which is equal to $^{704}/_{1024}$. Subsequently we could say that in a network of 5 nodes with a frame capacity of 3, 68.75% of topologies would suffer no losses due to the bottleneck effect. In this way (29) can be used to select a suitable value of F based on the number of nodes in the network and a given acceptable α threshold.

Each frame has a status, which is indicated by its priority bit. The status can be either priority or diversity and is assigned and used as described in section 5.3. When a node is compiling a packet for transmission it will first fill available frames with priority data, then any empty frames will be filled with diversity data. This allows nodes to utilise more of the available packet capacity. The blind forwarding treats individual frames separately, which allows individual pieces of date to transverse the network on its own path based on its priority.

In order to allow the protocol to keep memory requirements to a minimum and cope with high levels of traffic, superseded frames are dropped. In other words, if a node receives two pieces of data which originated from the same node, the older piece will be disregarded in favour of the newer data. Alternatively, if a node receives a piece of data that is older than a

piece of data already in the queue, the received data will be considered out-of-date and be dropped.

5.4.2 Operational Description

Figure 5.5 shows the general operation of PHASeR. Since the protocol uses the GTDMA MAC layer, the first step is to determine whether the current slot is the dedicated slot for this node. If it is not, then the nodes will listen for a transmission. If a transmission can be heard then the packet is received and the relevant information is extracted, such as the data, priority, source node and the transmitting nodes gradient metric. The received hop count is also used to update that nodes gradient metric. If no transmission can be heard, the node will sleep until the next time slot. When the node is in its own time slot, PHASeR will simply start to compile the data with the highest priority, as well as its own, into a single packet. Once the packet has been compiled it can be broadcast to the nodes neighbours. If there is no data to transmit the protocol simply sleep until the next time slot.

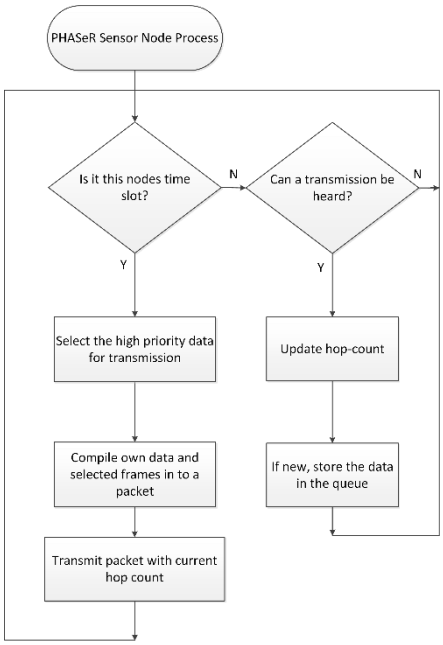


Fig. 5.5. Flow chart of PHASeRs basic operation.

As an illustrative example of frame capacity, figure 5.6 shows five nodes, where the frame capacity is limited to three. The five nodes, A, B, C, D and E have the hop counts 2, 2, 2, 1 and 1 respectively. The transmitted packets are described in frames; so, $F_1(A)$ is the 1st frame of the packet and it contains data from node A. The first part of the figure shows the first transmission from node A, which consists of only one frame with node A's own data inside. This is received by nodes B, C and D. The second part of the figure shows a transmission from node B, which contains both its own data and node A's data. This transmission is received by both nodes A and D, however node A has a greater hop count then node B, so the packet is not forwarded. The

second part of the figure also shows node C's transmission, which contains its own data as well as node A's. This is received by nodes A, D and E. Since node A's hop count is greater than node C's, the data is not forwarded. The third part of the figure shows node D's transmission. At this point node D has received data from nodes A, B and C, however the packet only has enough space for its own data and two other frames. So, in this case, node C's data is dropped and node D forwards node A's data, node B's data and its own data. The third part of the figure also shows node E transmitting a full packet containing node A's data, node C's data and its own. This example also illustrates how blind forwarding is used to pass data along multiple paths to create redundancy and how the frame capacity limits the number of duplicate packets so that the network doesn't become saturated.

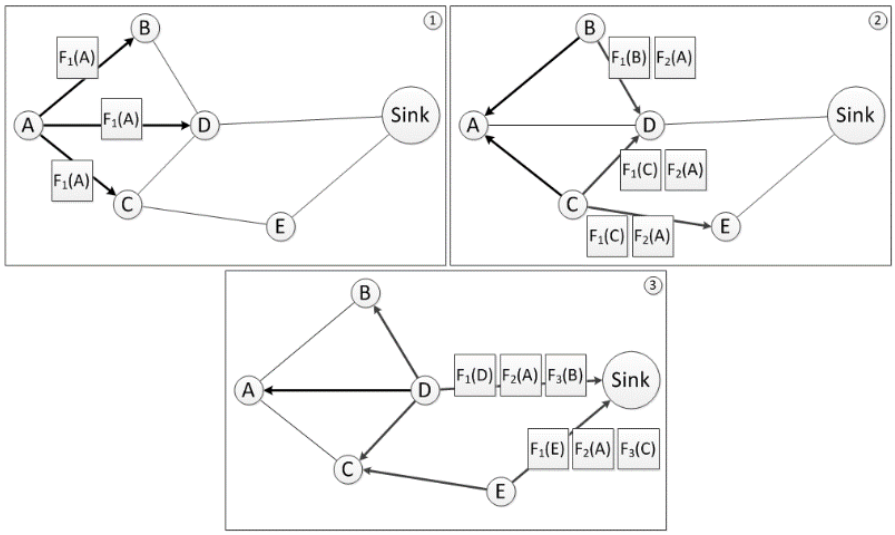


Fig. 5.6. Diagram of a five node network, showing how frames are propagated towards the sink.

5.4.3 Mathematical Analysis

This section will present a mathematical analysis of PHASeR, designed to characterise the protocols performance for varying parameters. The metrics analysed are average end-to-end delay, packet delivery ratio, throughput, overhead and energy consumption, all of which are as defined in chapter 4. Since the analysis is designed to characterise the general trend of the protocol, it will provide a generalised approximate results, which can be compared with the subsequent simulation results.

5.4.3.1 Average End-to-End Delay

The first metric tackled is average end-to-end delay, D_{av}, and is the average between a packet being created and being delivered to the sink as given in (20).

The average delay in a TDMA multihop based protocol depends greatly on the order of the allocated time slots of the forwarding nodes. The best case scenario is that the forwarding nodes are in sequence, which is shown in figure 5.7.

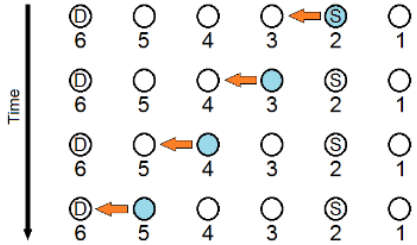


Fig. 5.7. Minimum packet delay based on the order of the time slot assignment in the forwarding nodes.

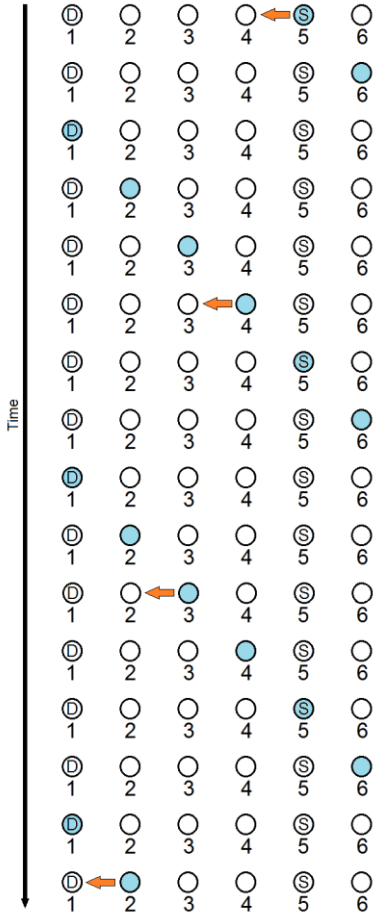


Fig. 5.8. Maximum packet delay based on the order of the time slot assignment in the forwarding nodes.

Figure 5.8 shows the source of the data, *S*, transmit the data to the destination, *D*. The nodes are shown at each timeslot, where the transmitting node is highlighted, and the progress of the data can be seen by the arrows. The timeslot in which a node should transmit is given underneath the respective nodes. The expression to calculate the delay in this scenario is:

$$T_{q_{min}} = h \cdot \Delta \qquad (30)$$

where Δ is the length of a single timeslot and h comes from (14).

However if the nodes allocated time slots are in the reverse order then the delay will be considerably longer, as shown in figure 5.8.

The expression for this worst case scenario is given as:

$$T_{q_{max}} = \big((h-1)(n-1)+1\big) \cdot \Delta \qquad (31)$$

Assuming that the time slots are uniformly distributed, then the average end-to-end delay is given by:

$$T_{q_{av}} = T_q = \left(\frac{n \cdot (h-1)}{2} + 1\right) \cdot \Delta \qquad (32)$$

This equation shows that as the number of nodes in the network increases, so will the delay, which is unsurprising considering the use of a TDMA MAC layer. The increase in nodes will increase the *n* term as well as the packet length, which will in turn increase the time slot length causing the delay times to increase quickly.

5.4.3.2 Packet Delivery Ratio

The second metric is PDR, which is the number of packets successfully received, per number of packets transmitted, as given in (1). As the TDMA MAC scheme is contention free there will be no loss from collisions. However, nodes being disconnected from the network and path breaks may cause packet losses. The use of multipath routing creates redundancy in the network, however the dropping of out dated frames may cause data to be lost, the characteristics of which are also captured by this metric.

From (10), the probability of a link breaking in the time between a packet being transmitted and its reception at the sink, P_{break}, can be determined as:

$$P_{break} = 1 - \left(1 - \frac{\pi \cdot v_{max}}{4 \cdot r}\right)^{D_{av}} \qquad (33)$$

and therefore the expected number of broken links, L_{broken}, can be determined by multiplying P_{break} by the expected number of active links:

$$L_{broken} = P_{break} \cdot \binom{n}{2} \cdot \left(\frac{\pi \cdot r^2}{L^2}\right) \qquad (34)$$

where $\binom{n}{2}$ is the binomial coefficient of *"n choose 2"*.

Since the average number of hops is given as *h*, this also indicates the number of links that a packet must traverse before it reaches the sink. Assuming that a link break on the packets path will cause packet loss then the PLR is given by:

$$PLR = 1 - (1 - P_{break})^h \qquad (35)$$

PDR can now simply be calculated as:

$$PDR = 1 - PLR = \left(1 - \frac{1}{t_{av}}\right)^{D_{av} \cdot h} \quad (36)$$

This expression can be simplified as follows:

$$PDR = 1 - \left[\frac{D_{av} \cdot \pi \cdot v_{max} \cdot h}{4 \cdot r}\right] \quad (36)$$

From this expression the effect of various parameters can be seen; increasing the number of nodes will increase the end-to-end delay causing lower PDR. Also, the faster the maximum speed of the nodes and the smaller the transmission radius, the lower the PDR. Increasing the size of the network will increase the average hop-count and also diminish the PDR.

5.4.3.3 Throughput

In this thesis throughput, *TP*, is defined as the number of data bits successfully delivered to the sink, per second, over the entire simulation time, as described in (5). So the expression is given as

$$TP = \frac{L_{data} \cdot N_p \cdot PDR}{T_t} \quad (38)$$

where N_p is the number of packets produced and T_t is the total deployment time of the network. The expression describes how the number of lost packets decreases the throughput, as does the number of packets produced.

5.4.3.4 Overhead

Overhead, *OH*, is a major factor in designing routing protocols for mobile networks since too much can cause congestion, which will limit the throughput of data. There are generally two types of overhead; packet overhead and control overhead. Packet overhead is the ratio of non-data bits to data bits in a data packet. Control overhead is the ratio of bits in control packets to bits in data packets. Control packets are often used to negotiate channel access, discover routes or share topology information.

Equations (3) and (4) describe the components of overhead separately, but here a measure is used to evaluate both control and packet overhead simultaneously. The overall overhead is characterised by the total number of bits transmitted per successfully delivered data bit:

$$OH = \frac{B_{tx}}{L_{data} \cdot N_p \cdot PDR} \quad (39)$$

where B_{tx} is the total number of bits transmitted.

To determine B_{tx} the following expression is used,

$$B_{tx} = \left[\left(N_f - 1\right) \cdot N_p \cdot L_{F_2}\right] + \left[(n-1) \cdot \frac{T_t}{\Delta n} \cdot L_{F_1}\right] + \left[\frac{T_t}{\Delta n} \cdot L_{F_s}\right] \quad (40)$$

where N_f is the number of forwarding neighbours, L_{F2} is the size of the type 2 frame, L_{F1} is the length of a type 1 frame and L_{Fs} is the size of the sink frame. The equation and derivation of N_f is in *appendix B*. The sink frame is a reduced size type 1 frame. The first set of square brackets calculates the number of additional type 2 frames that contain data forwarded from other nodes. The second set of square brackets determines the number of type 1 frames transmitted, which is one per cycle for every sensor node. The last set of square brackets is used to evaluate the contribution of transmissions from the sink, which also transmits once every cycle.

5.4.3.5 Energy Consumption

When analysing the energy consumption of the protocol, only the power used to transmit and receive messages is considered. This is because the transceiver has the largest energy cost compared with that of the processor. The other factors such as the compiler used, which may make code more or less efficient, will affect the processors energy consumption, as well as other tasks that need to be run. There are also other energy costs attributed to things like the sensors and other peripherals, the mobility platform and the battery type, which are hardware specific and difficult to account for. There is also the time in which the node is listening for a message, which, in PHASeR, occurs for one symbol at the beginning of each timeslot. However, for comparison purposes this has been omitted, such that protocols which do not consider sleep cycles are not given an unfair disadvantage.

The energy consumption, *EC*, is characterised in terms of joules per second per node:

$$EC = \left(\frac{V_{batt}}{R_b}\right) \cdot \left(\frac{(I_{tx} \cdot B_{tx}) + (I_{rx} \cdot B_{rx})}{n \cdot T_t}\right) \quad (41)$$

where R_b is the bit rate of the transceiver, V_{batt} is the voltage of the batteries, I_{tx} and I_{rx} are the current consumptions of the transceiver when transmitting and receiving respectively and B_{rx} is the total number of bits received.

Since PHASeR broadcasts packets to all neighbours, using N_n from (7), B_{rx} is given as

$$B_{rx} = B_{tx} \cdot N_n \quad (42)$$

This expression requires knowledge of the hardware but, V_{batt}, I_{tx} and I_{rx} can be substituted for temporary values based on potential hardware for comparison purposes.

5.5 Location Aware Sensor Routing (LASeR)

LASeR (Location Aware Sensor Routing) is the second protocol presented in this thesis. It differs from PHASeR in two main respects; firstly in its technique of gradient maintenance and secondly in the fact that it doesn't use the packet encapsulation. LASeR uses location awareness to provide a gradient metric, which means that the problem of gradient maintenance is avoided. The protocol also doesn't assume that nodes are regularly generating data and as such the data encapsulation is not used and packet size is reduced to be able to contain only one piece of data at a time. This means that timeslots are shorter and so messages can be delivered faster. LASeR still utilises the advantages of blind forwarding and also uses the GTDMA MAC layer.

5.5.1 Protocol Description

LASeR is designed for applications that require the use of location awareness, which is quite common in MWSNs. This means that nodes will already be equipped with some method of determining its geographic position, and LASeR aims to take advantage of this.

The packet structure used is shown in figure 5.9, in which Q_L is the quantisation level in meters and the total packet length is L_p.

Field name	Node ID	Location	Data	Priority bit	Packet ID
Size (bits)	$\lceil \log_2 n \rceil$	$\left\lceil \log_2 \left(\frac{\sqrt{2} \cdot L}{Q_L} \right) \right\rceil$	L_{data}	1	$\lceil \log_2 n \rceil$
Total size (bits)	$L_P = 2 \cdot \lceil \log_2 n \rceil + \left\lceil \log_2 \left(\frac{\sqrt{2} \cdot L}{Q_L} \right) \right\rceil + L_{data} + 1$				

Fig. 5.9. LASeR packet structure showing the five fields and their respective sizes. The expression for the total packet size, L_P, is also given, where $\lceil \cdot \rceil$ indicates the ceiling function.

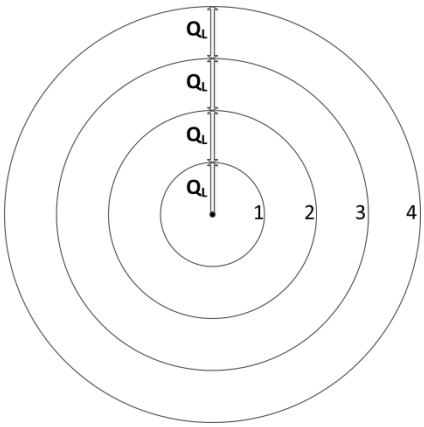


Fig. 5.10. Conceptual radial bands emanating from the sink, showing location index. The indexes are used by nodes as the gradient metric, which indicates the direction that packets should be sent.

The location information can be from any available geographic positioning technique, which may be application specific. Though it should be noted that some of these techniques require significant energy cost and their accuracy can be unreliable. For the purposes of this work the location information is assumed to be perfect. This is to isolate the routing protocol

such that its performance may be analysed without the added effects of an imperfect localisation technique.

Each nodes distance from the sink is quantised, such that an integer value can be used as a gradient. Conceptually, this creates radial bands emanating from the sink as shown in figure 5.10. A nodes location corresponds to the location index of the radial band that it is in. For example if the quantisation level, Q_L, is 5 meters and a nodes Euclidean distance from the sink is 12 meters, this would put it in the third radial band. Accordingly, the node would be assigned a location index of 3. These location indexes are the metrics used to create the gradient field.

In terms of the field size, using a 40 meter by 40 meter network as an example, the furthest possible distance a node may be from the sink is 56.57 meters. By splitting this length up into increments of 5 meters, 12 segments are created. In order to store these 12 location index values, 4 bits are needed in the packet, which is corroborated by the location field size equation in figure 5.9.

As with PHASeR, LASeR uses blind forwarding to transmit packets, however, as data is not expected to be generated at fixed time intervals, a more traditional FCFS queuing algorithm is used. LASeR still uses the priority and diversity classifications, as explained in section 5.3. As such, the protocol selects the oldest data with the highest priority to create the next packet to forward. The use of the priority bit increases the multipath route diversity of the protocol, which helps to alleviate the dead-end problem.

Whilst this protocol no longer relies on the GTDMA MAC layer for gradient maintenance, it is still used in the basic implementation of LASeR. This being said, in chapter 7 other options for the MAC layer are explored and tested.

In terms of methods of location determination a number of techniques are surveyed in [5.1]. Four main methods are identified, one of which is lateration, in which the distance between landmarks or other nodes is used to estimate a nodes position. Angulation is the technique of using the angular separation between other nodes of known positions, to determine a nodes location. A less accurate method is cellular proximity, whereby a node determines the region that it's in by its proximity to fixed nodes with known positions. When considering robot platforms, dead reckoning may be used, in which nodes keep track of their position by measuring the speed and time travelled in a certain direction. Common measurements from other nodes, in order to localise a node may include time of arrival, angle of arrival, time difference of arrival, received signal strength or Doppler shift. Each of these localisation methods and measurement techniques have their advantages and disadvantages, but may all be implemented in a MWSN. The use of some of these methods may also incur additional energy costs, which will need to be taken into account when considering deployment.

5.5.2 Operational Description

The basic operation of LASeR is shown in figure 5.11 and as with PHASeR it begins by check the timeslot number against its own ID. If it isn't this nodes timeslot then it will listen for a transmission and sleep if there isn't one. If a transmission is heard it will be a data packet and the incoming data should be extracted. If the data is not already in the queue it should be added, else the existing entry should be updated. If it is the nodes timeslot then the node should check the queue for data to send. The data should be selected based on the position in the queue and its priority. Once the data has been selected it can be compiled into a packet along with the source nodes ID and the transmitting nodes quantised distance from the sink. The packet is then broadcast to the nodes neighbours. If there's no data to transmit, the node should sleep until the next timeslot.

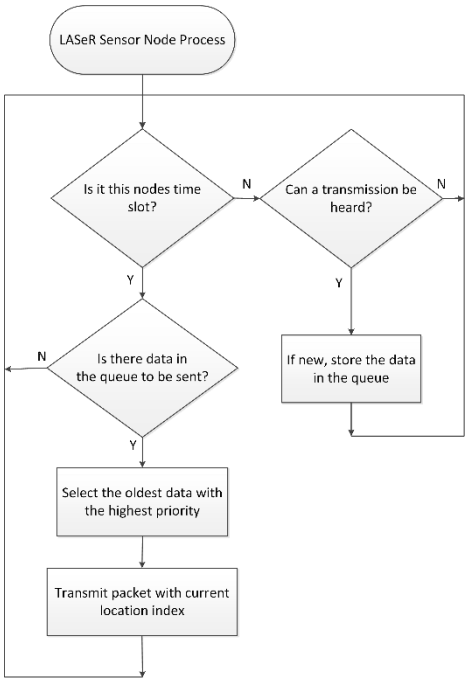


Fig. 5.11. Flow chart of LASeRs basic operation.

5.5.3 Mathematical Analysis

Again the aforementioned five metrics, average end-to-end delay, packet delivery ratio, throughput, overhead and energy consumption, will be analysed to give generalised mathematical expressions.

5.5.3.1 Average End-to-End Delay

The arrival rate of packets to a node, λ, is assumed to follow a Poisson distribution and each node is considered to be a single server. Since the global TDMA has a deterministic medium access time, the service time, T_s, is constant. For this reason each node is modelled as an M/D/1 FCFS queue. In each node packets are created at a rate of f_p. The total number of packets

forwarded by a node is equal to the packet creation rate multiplied by the number of nodes, whose data it forwards:

$$\lambda = \frac{f_p \cdot (n-1)}{N_n^2} \tag{43}$$

As each node gets the chance to transmit once in a TDMA cycle and the sink doesn't need a slot, the service time is simply

$$T_s = \Delta \cdot (n-1) \tag{44}$$

where Δ is the length of a single time slot, which is given by:

$$\Delta = \frac{L_p}{R_b} + \frac{r}{c} \tag{45}$$

where R_b is the bit rate and c is the propagation velocity of the signal.

Using the Pollaczek-Khinchin formula for the mean time in the system, T_q, as in (21), the delay can be described as:

$$D_{av} = \frac{h \cdot T_s \cdot (2 - \lambda T_s)}{2 \cdot (1 - \lambda T_s)} \tag{46}$$

This expression highlights the contribution of the number of hops and packet arrival rate to the delay time.

5.5.3.2 Packet Delivery Ratio

Given that the GTDMA MAC layer is still being used, as is the blind forwarding technique, the expression derived for PDR is the same as in PHASeR and is given in (37).

5.5.3.3 Throughput

Similar to PDR, the throughput expression is the same as PHASeR; namely equation (38).

5.5.3.4 Overhead

The overhead metric is calculated using (39), however B_{tx} is given as

$$B_{tx} = N_f \cdot N_p \cdot L_p \tag{47}$$

This expression is different because LASeR only has one size of packet that it transmits.

5.5.3.5 Energy Consumption

As with PDR and throughput, the equation derived for energy consumption is the same as used in PHASeR, where B_{rx} is calculated using equation (42) and EC is given by (41).

5.6 Robust Ad-hoc Sensor Routing (RASeR)

RASeR (Robust Ad-hoc Sensor Routing) is the third protocol presented in this thesis. It uses the same short packet size as LASeR, to decrease end-to-end delay, but it does not rely on location awareness for its gradient maintenance. Instead it uses the hop-count metric, which is maintained through the use of a GTDMA MAC layer, which negates the dead-end problem and bad routing decisions from inaccurate location information.

5.6.1 Protocol Description

RASeR is designed to be the most general purpose of all the protocols as it doesn't assume the sensors are regularly generating data or that there is some kind of location awareness available. As RASeR still uses the GTDMA method of gradient maintenance, in every slot nodes are required to transmit a data packet, as in figure 5.12, or a beacon packet if the node has no data to forward. A beacon packet is simply the first two fields of the data packet, namely the node ID and hop count. In this way, each node will have the chance to overhear the transmission of every node that's in range, before its own timeslot comes around and its gradient can be determined before it has to transmit.

Field name	Node ID	Hop Count	Data	Priority bit	Packet ID
Size (bits)	$\lceil \log_2 n \rceil$	$\lceil \log_2 n \rceil$	L_{data}	1	$\lceil \log_2 n \rceil$
Total size (bits)	$L_P = 3 \cdot \lceil \log_2 n \rceil + L_{data} + 1$				

Fig. 5.12. RASeR packet structure, where *n* is the number of nodes, L_{data} is the size of data and L_p is the total size of a data packet.

This means that in a single cycle, one node will hear all of its neighbours and transmit at least a beacon packet. As such, the hop count metric is kept up-to-date and the regular flooding of the network is mitigated. RASeR also uses the blind forwarding technique to pass data through the network, so each node simply broadcasts its data packet and then it is the responsibility of the receiving node to decide whether it should forward the packet or not. Again, the priority system is used to assign packets either priority or diversity status, such that packets coming from nodes that are further away get priority status and priority packets coming from nodes with an equivalent gradient are assigned diversity status. Overall, the GTDMA schedule ensures that the gradient metric is kept up-to-date and the blind forwarding technique ensures that data is always passed towards the sink. The inherent route diversity is controlled through the protocols priority system and FCFS queuing algorithm, which ensures that the oldest data with the highest priority is given preference. The combination of the low overhead forwarding and the ability to maintain a gradient field without flooding means that the protocol has very low overhead, which enables its use in highly mobile environments.

5.6.2 Operational Description

Figure 5.13 shows a flow chart of RASeRs basic operation, which illustrates the steps a node will take during a timeslot. The first thing the node should do it determine whether the current timeslot is its own dedicated slot. If it is the node should check the queue for data to be transmitted. If there is no data to send then the node can sleep until the next slot. If there is data to send, then the oldest piece of data with the highest priority should be selected. It can then be compiled into a packet with the source nodes ID and the transmitting nodes hop count. Once compiled, the packet can be broadcast to all of the nodes neighbours. If it isn't the nodes dedicated timeslot, then it should listen for a transmission. If the transmitting node is not in range then the listening nodes should sleep until the next timeslot. If a packet is heard then the relevant information should be extracted. Firstly, the hop count should be used to update the current hop count of the receiving nodes. Then if it is a beacon packet the node can sleep, otherwise the data should be taken. If the data is not already present in the queue it should be added. If the data is already in the queue then its status should be updated accordingly.

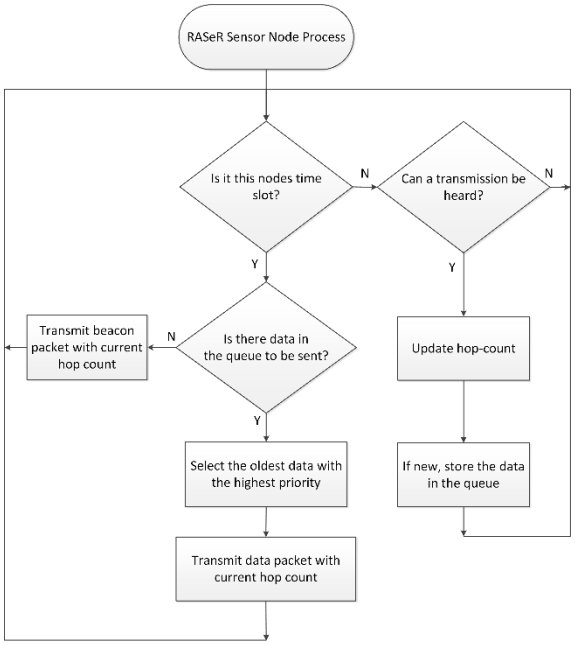


Fig. 5.13. Flow chart of RASeRs basic operation.

5.6.3 Mathematical Analysis

In this section generalised analytical expressions are derived for packet delivery ratio, average end-to-end delay, throughput, overhead and energy consumption, and are intended to characterise the performance of RASeR.

5.6.3.1 Average End-to-End Delay

In RASeR the packet arrival rate, λ, still follows a Poisson distribution and is modelled the same way as in LASeR. This means that the D_{av} expression will be the same as (46). However, RASeR requires that the sink is also allocated a timeslot, so the service time is given as

$$T_s = \Delta \cdot n \tag{48}$$

5.6.3.2 Packet Delivery Ratio

PDR is a key routing metric and as it uses the same blind forwarding technique as both PHASeR and LASeR, the analytical expression is the same; (37).

5.6.3.3 Throughput

As with PDR, throughput is modelled the same with all three protocols using equation (38).

5.6.3.4 Overhead

Again, like LASeR and PHASeR, the overhead is characterised by (39), however B_{tx} is given by

$$B_{tx} = \left(\left[\left(\frac{T_t}{\Delta n}\right) - (N_f \cdot f_p \cdot T_d)\right] \cdot L_{p_{beacon}} \cdot n\right) + \left(N_f \cdot N_p \cdot L_p\right) \tag{49}$$

where N_f is the number of forwarding neighbours from appendix B, T_d is the amount of time for which nodes are producing data and $L_{Pbeacon}$ is the length of a beacon packet. The last part of the equation in the last set of parentheses, simply calculates the contribution of bits from data packets. The rest of the expression determines the contribution of beacon packets. This is done by calculating the maximum possible number of beacon packet transmissions, or the total number of time slots.

5.6.3.5 Energy Consumption

Energy consumption is characterised in the same way as LASeR, using equations (41) and (42).

5.7 COMPARISON

This section gives a descriptive comparison of the three protocols presented; PHASeR, LASeR and RASeR. All three use the gradient routing technique of blind forwarding, which uses limited overhead to route data down multiple paths towards the sink. All of the protocols use only a simple sorting algorithm, which makes them suitable for even cheap processors and they have an algorithmic complexity of only $O(q)$, where q is the length of the queue.

The major difference between PHASeR and the other two is in the use of encapsulation; PHASeR encapsulates pieces of data from multiple nodes into one packet. This gives very good packet overhead to data ratio, but increases the slot length, which subsequently increases the delay time. LASeR and RASeR use a shorter packet length, which allows packets to be delivered faster and also doesn't rely on the nodes regular generation of data. In RASeR this means that short beacon packets must be used to keep the gradient up-to-date. Since these beacon packets are only used to broadcast local topology information, they only use a limited amount of bandwidth and still negate the need to flood the network.

The main contrast between LASeR and the other two protocols is the use of location awareness to maintain the gradient metric. Using location information that is already present on the nodes, alleviates the overhead of gradient maintenance from the routing protocol. However, location acquisition schemes are not always accurate and can be slow. If the location information isn’t updated regularly enough, or is incorrect, then the gradient metric will be corrupted and packets may be routed incorrectly, which can cause delays and packet loss. The use of location information can also lead to the dead-end problem, whereas using a hop-count metric will prevent this.

Additionally, PHASeR nodes will be transmitting data in every cycle, so the cycle time will directly correlate to the sensor sample time. This is good for energy consumption, but may not be enough to keep up with topology changes. Contrastingly, LASeR and RASeR will only be transmitting data packets when they have data to send. However, the gradient maintenance requires that RASeR nodes send at least a beacon packet every cycle, whereas LASeR relies on the existing location information. This will results in RASeR requiring more power than LASeR. Although from the perspective of the entire system, nodes running LASeR will have some additional energy usage from whatever method of location awareness is present, whereas, in PHASeR and RASeR nodes, this may or may not be present.

5.8 CONCLUSION

This chapter has described the key considerations from which the design choices were motivated and the key routing principles used in this work were also identified. Subsequently,

three novel MWSN routing protocols were presented along with generalised mathematical expressions. The three protocols each have their own strengths and weaknesses, which will be further explored in the next chapter, which will evaluate each of these three protocols and compare their performance in simulation.

CHAPTER 6

MODELLING, SIMULATION AND RESULTS

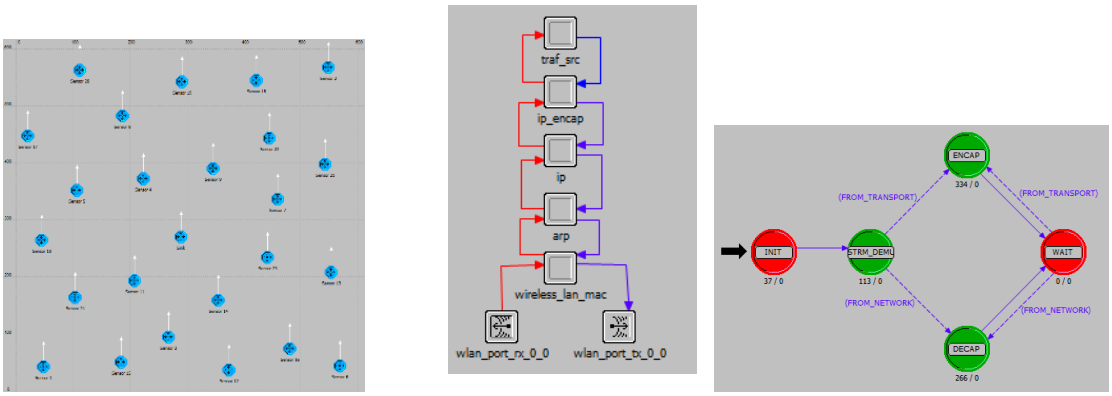
6.1 INTRODUCTION

Smulation is a useful tool to capture the complex interactions of large networks without expensive hardware or the generalisations often used in mathematical analysis. This chapter will firstly describe the framework of the OPNET [4.9] simulator and how the models of the proposed protocols were constructed. Next, the simulation parameters are described, followed by a thorough set of simulation results and discussion for the proposed protocols. Further to this, RASeR is also tested in a fading environment in order to evaluate the protocols robustness to imperfect channel conditions.

6.2 OPNET

OPNET models consist of four main levels; the network level, the node level, the state level and the code level. The highest is the network level as shown in figure 6.1(a), which is where the type, quantity and topology of nodes is set. Although, in the case of a mobile network, the topology will change as the simulation progresses. Mobile nodes are indicated by a white arrow. Inside each node is a node model, which is made up of a number of process blocks, as shown by figure 6.1(b). The process blocks can pass packets between themselves as well as statistics. Inside each process block is a process model, which is shown by figure 6.1(c), and consists of a number of different states. Contained inside these states is the code level, which is where the code is written that defines the behaviour of the nodes. Each state in the process model is split into two parts, the enter executives (EE) and exit executives (XE). As such, when a state is entered, the code in the EE is run and just before a state is exited, the code in the XE is run. There are two types of state available, forced and unforced. A forced (green) state, when entered, will run the EE, then the XE and then exit the state. Whereas, unforced (red) states will run the EE and then wait for an interrupt before the XE is run. Upon running the process model, the first state to be executed is indicated by a large black arrow, this is the “INIT” state in figure 6.1(c). With interrupt based systems, such as sensor nodes, it is common to have a wait state, to which the node defaults. Then as interrupts are received they are dealt with by running the code in the appropriate state.

OPNET provides process blocks at the node level for wireless communication, within which PHY parameters such as centre frequency and data rate can be set. The wireless communication is governed by the transceiver pipeline, which consists of thirteen stages. These stages determine whether two transceivers are able to hear each other, the time it takes to transmit, the delay between transmission and reception, the amount of background noise, the amount of noise from interference, the resulting SNR (Signal to Noise Ratio), BER (Bit Error Rate) and number of errors present in the packet, and subsequently whether the packet is passed on to the next layer. Some contributing factors to this outcome are the power of the transmitter, the gain of the receiver, the type of modulation being used and whether multiple packets are arriving at the receiver at the same time.



(a) Network Level (b) Node Level (c) State Level

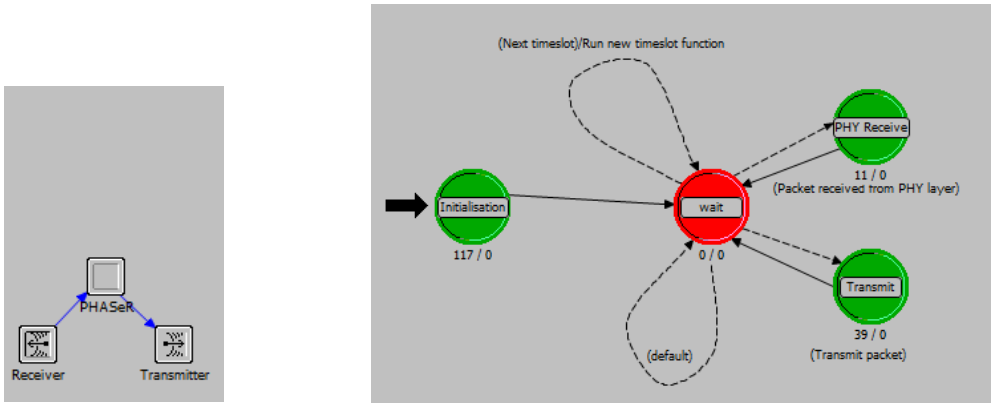
Fig. 6.1. OPNET Modelling.

6.3 MODELLING

This section will describe the interrupt orientated modelling of the protocols within OPNET.

6.3.1 PHASeR

PHASeR was the first protocol modelled and the nodes consisted of three process block, as shown in figure 6.2(a). Essentially, the routing, MAC and data generation are all in the PHASeR process block and the receiver and transmitter blocks act as basic radios, which simply broadcast whatever they are passed. Figure 6.2(b) shows the PHASeR process model, which has four states. The initialisation state is the first state to be run and it defines variables, sets initial values and starts the timeslot timer. The wait state allows the protocol to stay idle whilst it's waiting for an interrupt.



(a) Node Model (b) Process Model

Fig. 6.2. PHASeR OPNET model.

If an interrupt is received, indicating that a new timeslot should begin, then PHASeR runs the new timeslot function, the flow chart for which is shown in figure 6.3(a). This function first increments the slot number and check to see if it's this nodes time slot. If it is, then the process model transitions to the transmit state. Finally, the function always sets the timer for the next timeslot before returning to the wait state. The transmit state is shown in figure 6.3(c), when it's called the protocol first selects the highest priority frames from the queue and compiles them into a packet. Since PHASeR is time driven, it mimics the reading of a sensor and generates its own data. This data is also added to the packet before being transmitted. In the case of a packet being received from the receiver process block, the node uses the packet to update its hop count before extracting the frames and storing them in the queue, as shown by figure 6.3(b).

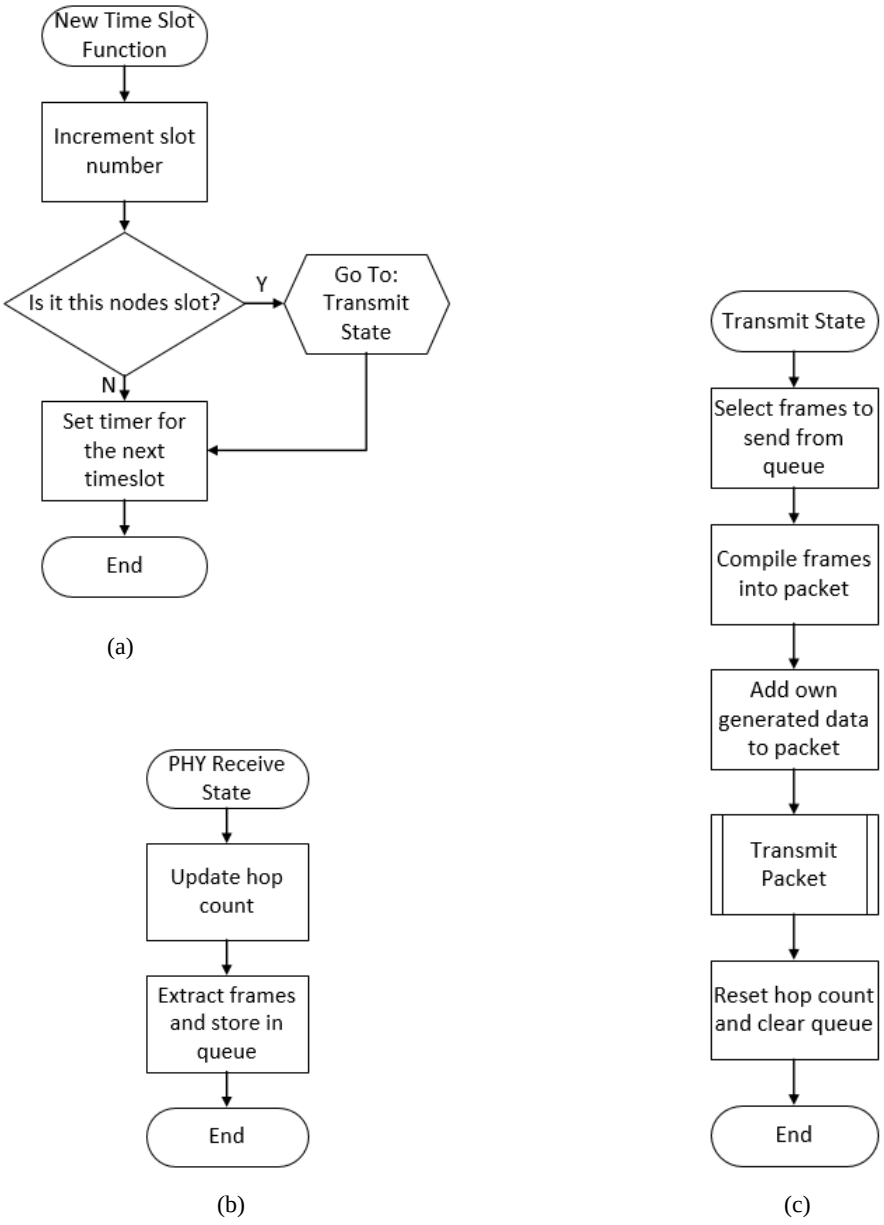
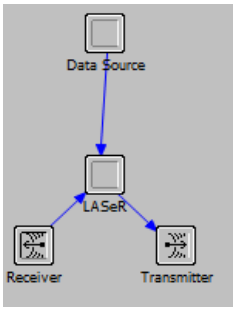


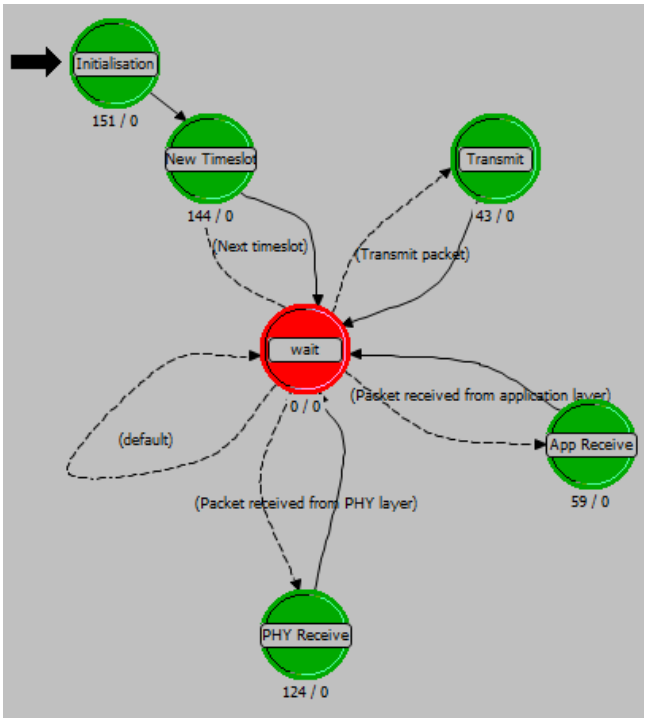
Fig. 6.3. PHASeR OPNET Model Flow Charts.

6.3.2 LASeR

Figure 6.4(a) shows the node model for LASeR, which differs from the PHASeR model as it now has a separate process block for generating data. Since LASeR is not necessarily time driven, this extra block allows data arrival rates and distributions to be defined. The LASeR process model in figure 6.4(b), again shows an initialisation state, which sets up required variables and then the default wait state.



(a) Node Model



(b) Process Model

Fig. 6.4. LASeR OPNET model.

Unlike PHASeR, the initialisation state doesn't set the timeslot timer, instead, after it has executed, the process model transitions straight into the new timeslot state. Figure 6.5(a) shows how this state functions in exactly the same way as in PHASeR, by first incrementing the slot number, checking whether it is this nodes timeslot and setting the timeslot timer. If it is this nodes timeslot the process model is instructed to transition to the transmit state, which is shown in figure 6.5(d). The first thing that this state does is determine the nodes current location before it checks the queue for data to send. If there is data to send then the oldest packet with the highest priority is selected and compiled into a packet along with the gradient metric. The packet is then broadcast. If there is no data to send, the process model simply returns back to the wait state. In the event that the data source block passes a packet down to the LASeR block, the application receive state will be entered into. In LASeR, this simply assigns the data a packet ID and stores the information in the queue, as shown in figure 6.5(b). Alternatively, if a packet is received from the PHY layer, the node will first check whether the received data is already in

the queue. If it is, then the information will be updated, otherwise a new entry will be created, this is outlined by figure 6.5(c).

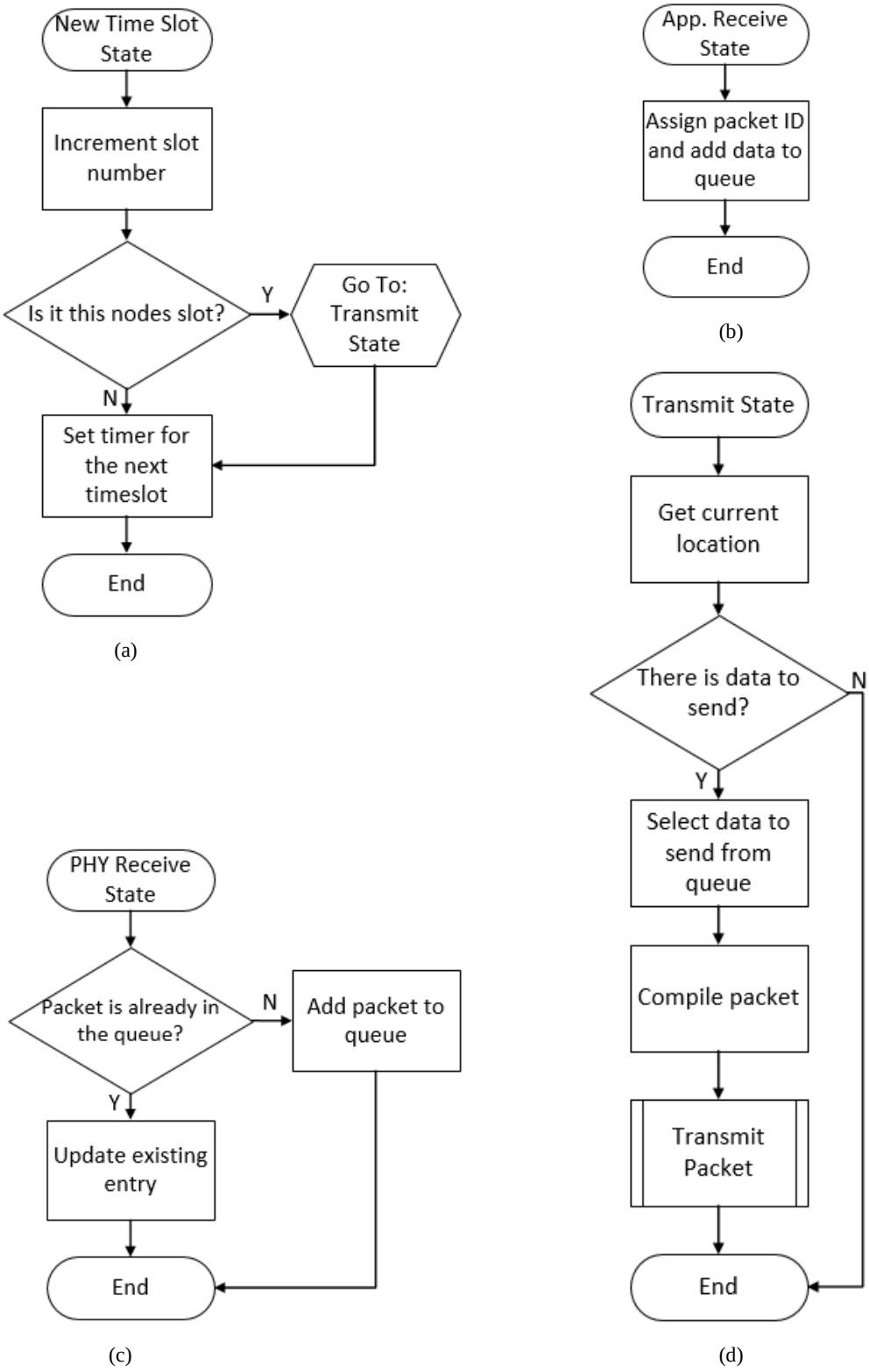
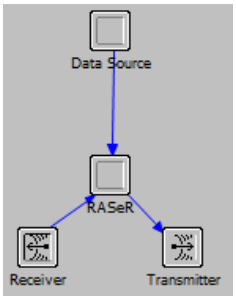


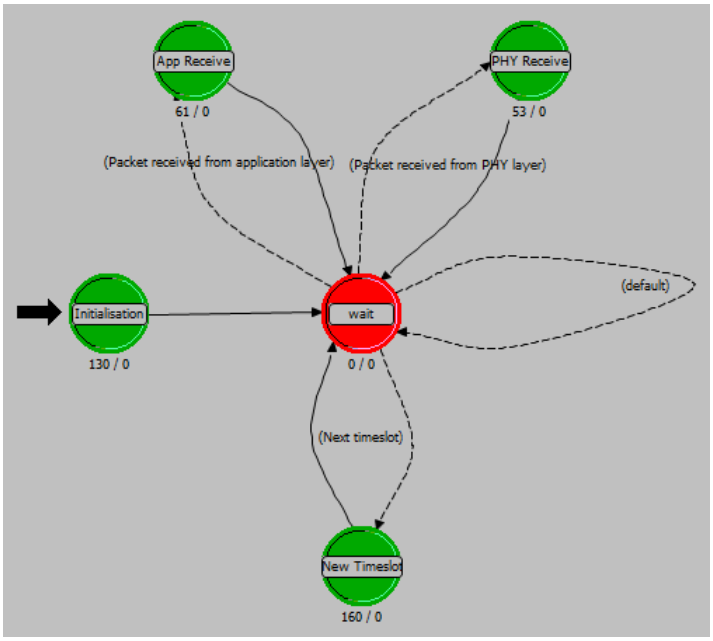
Fig. 6.5. LASeR OPNET Model Flow Charts.

6.3.3 RASeR

RASeR was modelled using the node model in figure 6.6(a), which has the same structure as LASeR, with the separate data source block. Figure 6.6(b) shows RASeR's process model, which begins with an initialisation state to set up variables and start the timeslot timer. The wait state is also present.



(a) Node Model



(b) Process Model

Fig. 6.6. RASeR OPNET model.

When the timer expires the new timeslot state is transitioned into, which, as shown by figure 6.7(a), first increments the slot number before checking whether it is this nodes dedicated slot. If it is, then the node checks for data to send, if there is no data to send then a beacon packet is compiled. If there is data to be transmitted, the oldest data with the highest priority is selected and compiled in to a packet. Once compiled, the node transmits the data or beacon packet before resetting its hop count. Then, regardless of whether there was data to send or not, the timer is set for the next timeslot. Figure 6.7(b) gives the flow chart for receiving a packet from the application layer, which simply tells the node to assign the data a packet ID and store it in the queue. If a packet is received from the PHY layer, figure 6.7(c) shows how the node first updates its hop count. If the received packet is a data packet, the queue is checked to see if the data is already present. If it is then the entry information is updated, otherwise the packet is added to the queue.

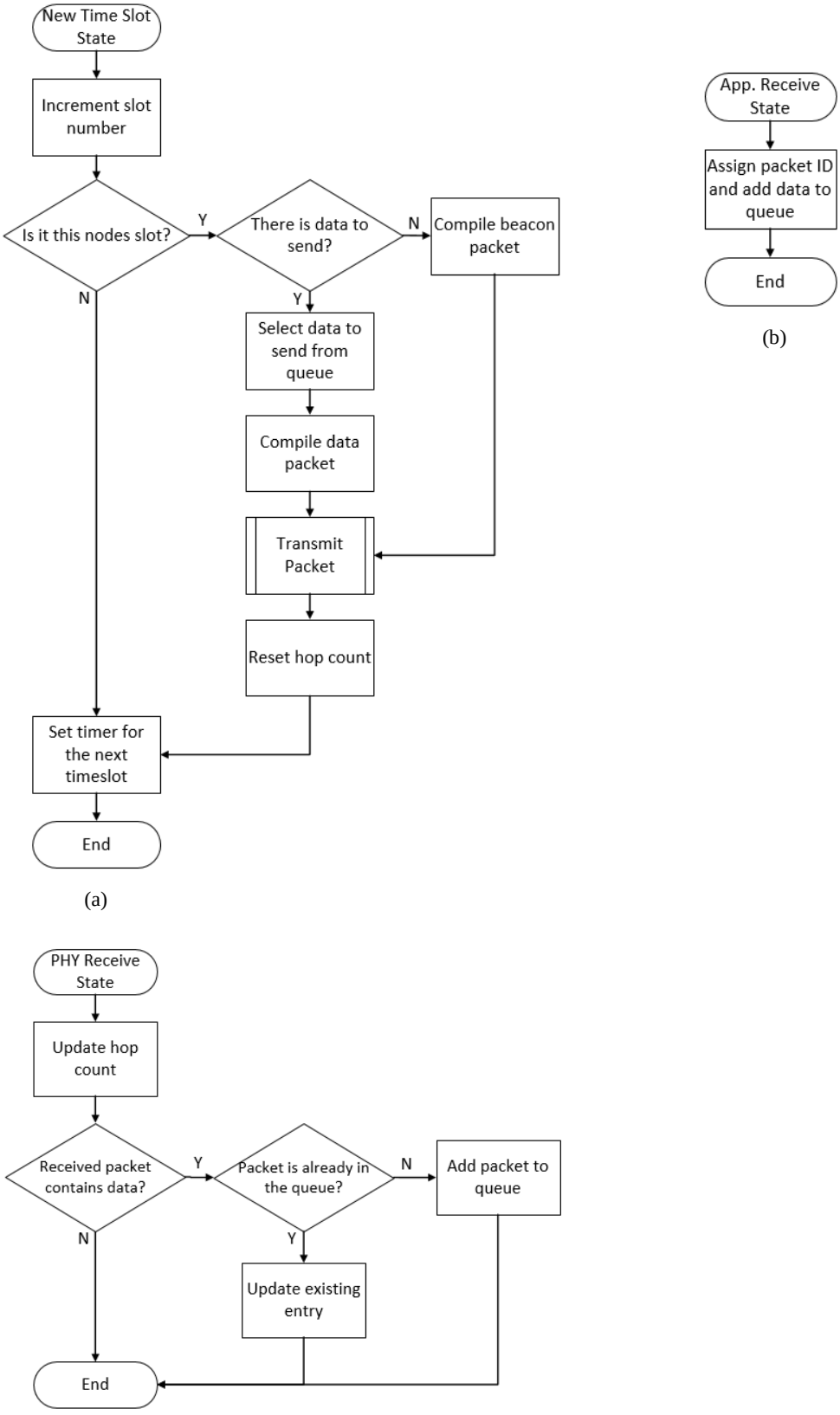


Fig. 6.7. RASeR OPNET Model Flow Charts.

6.4 Simulation and Analytical Results

The simulation parameters used were generally based on the Memsic IRIS motes [4.20] and the potential application of UAV aided search and rescue (SAR), which is discussed in more detail in chapter 8. The scenario described involves a swarm of autonomous drones reporting the location of identified target to a helicopter team. Given this scenario, base parameters are defined, which are kept constant over all simulations unless otherwise stated. The base parameters use a swarm of *25 nodes* deployed in a *600m* by *600m* network area as a medium sized search site. A random waypoint mobility model is used to vary the networks topology with a uniform distribution of speeds between *0m/s* and *25m/s*, and zero pause time. The maximum speed is close to both the cruising speed of a fixed wing UAV and the top speed of a rotary wing UAV. The random waypoint mobility model was used to imitate this varying topology of a cooperative search pattern, which may be executed by the UAVs. A packet generation rate of *1pk/s*, for every node, was set to give a reasonable traffic level for a swarm detecting multiple targets within the search area. This could be from something such as a plane crash at sea, which would give a high density of victims in the search site. The size of the generated data was considered to be *32bits*, which is enough to report the nodes relative position and some extra application specific data if necessary. Additionally, the queue length is limited to *n*, which will keep the memory requirements low.

In order to closely evaluate the routing protocols, the effects of the channel were removed. In this way the routing may be analysed in isolation, without interference from external causes of errors. As such, the transmission radius, *r*, was set to 250m, in accordance with the Memsic IRIS motes. Using a fixed transmission radius, nodes may only communicate whilst they are within this distance. Additionally, if a packet collides with another, it will be assumed to be corrupted and subsequently dropped. Another factor that is mitigated, are errors caused from incorrect location information. Furthermore, as no specific method of localisation is targeted or studied in this work, the potential additional energy that this would require, has not been taken into account.

Also, based on the IRIS motes, the parameters R_b, V_{batt}, I_{tx} and I_{rx} were set to *250kbps*, *3v*, *16.5mA* and *15.5mA* respectively.

The following sections give the results, in which PHASeR, LASeR and RASeR are compared with three other routing protocols; MACRO, AODV and OLSR. The popular ZigBee standard [6.1], is one of the most commonly used protocols in sensor networks and is based on AODV. For this reason AODV is used as a performance baseline of general sensor networks.

Since, AODV is reactive, the proactive protocol, OLSR, is included in our results as a comparison for completeness. MACRO is a recent high performance routing protocol, which is designed specifically for MWSNs. MACRO is used as a representation of the current developments in MWSN routing protocols. Simulation results are given for all protocols and analytical results are also given for PHASeR, LASeR and RASeR. The metrics used are PDR, average end-to-end delay, overhead, throughput and energy consumption, as are defined in section 4.2.

6.4.1 Mobility

Figures 6.8(a-e) show the results for varying mobility with all other parameters as described above. The mobility will be varied by adjusting the maximum speed in the random waypoint mobility model to *[0, 5, 15, 20, 25, 50, 75, 100]m/s*. The PDR for RASeR, LASeR and MACRO in figure 6.8(a) is near perfect for the whole range of mobility levels, with the minimum value being MACRO at 99.69%. This high result is predicted by the analytical results. PHASeR starts off well, but its PDR decreases as the maximum speed increases to a low of 94.41%, which is under estimated by the generalised analytical results. Both AODV and OLSR have significantly lower delivery ratios that never exceed 77%. In general, the same can be said for throughput in figure 6.8(d), in which RASeR and LASeR reach the upper bound and PHASeR gives slightly diminished performance. Again, AODV and OLSR are significantly lower than the rest. However, MACRO shows a lower throughput than RASeR and LASeR of more than 44bits/s, which is due to its increased delay. The lowest average end-to-end delay in figure 6.8(b) is given by RASeR and LASeR at 2.83ms, followed by MACRO at 49.45ms, and then PHASeR at 78.46ms. The worst delay comes from AODV at over a second. The delay of OLSR is shown to be very low, but this is due to the reduced number of packets that are being delivered. Essentially, OLSR is dropping around half of the packets being produced by the network, which means that the other packets are able to take advantage of the spare bandwidth and be delivered much faster. The analytical results for both RASeR and LASeR are quite accurate, however PHASeR's analytical results predict a higher delay than is simulated. The overhead results in figure 6.8(c) highlight the cost of RASeR's good performance, which is also reiterated in the energy results in figure 6.8(e), which gives a maximum of 7.77joules/s. LASeR and PHASeR have low overhead, which is also shown by their very low energy consumption of less than 1.14joules/s. MACRO requires slightly more overhead than PHASeR and LASeR, which is again illustrated in the energy results, giving MACRO a maximum of 2.03joules/s. Contrastingly, AODV and OLSR have significantly high overhead and power consumption.

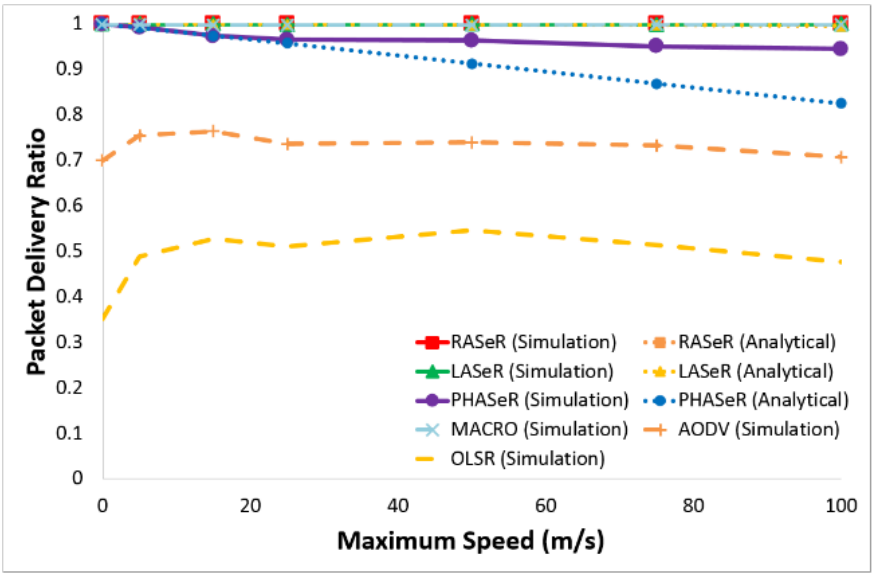


Fig. 6.8(a). Simulation and analytical PDR results for PHASeR, LASeR and RASeR, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*. With MACRO, AODV and OLSR simulations for comparison.

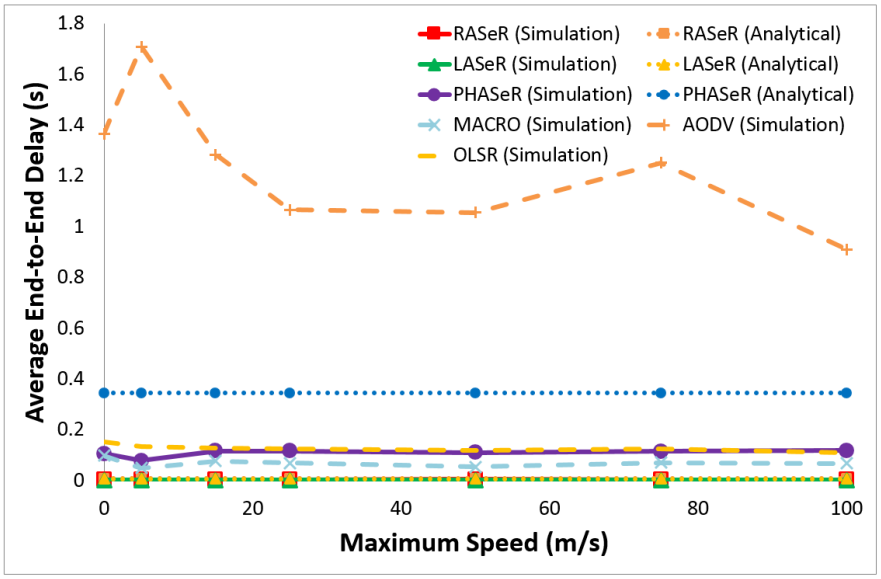


Fig. 6.8(b). Simulation and analytical end-to-end delay results for PHASeR, LASeR and RASeR, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*. With MACRO, AODV and OLSR simulations for comparison.

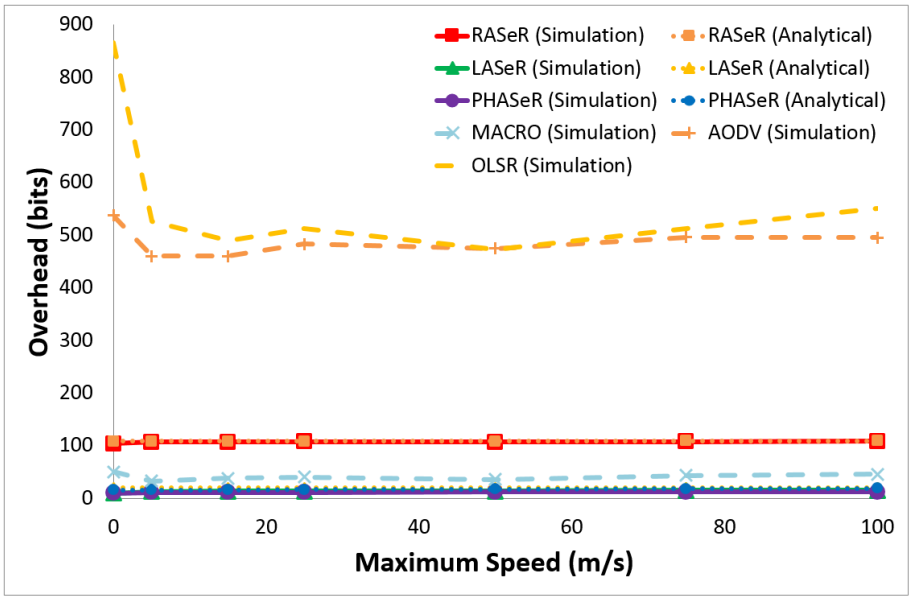


Fig. 6.8(c). Simulation and analytical overhead results for PHASeR, LASeR and RASeR, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*. With MACRO, AODV and OLSR simulations for comparison.

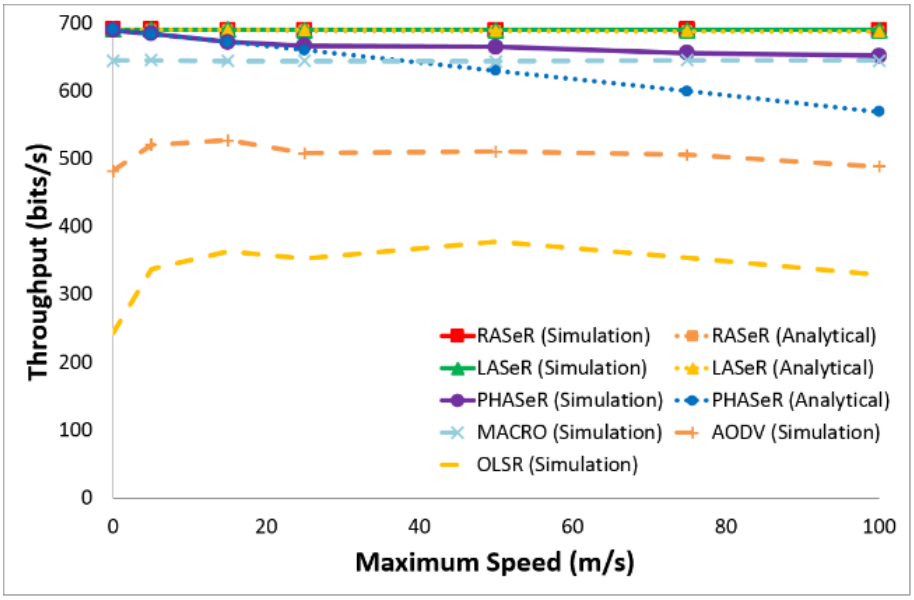


Fig. 6.8(d). Simulation and analytical throughput results for PHASeR, LASeR and RASeR, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*. With MACRO, AODV and OLSR simulations for comparison.

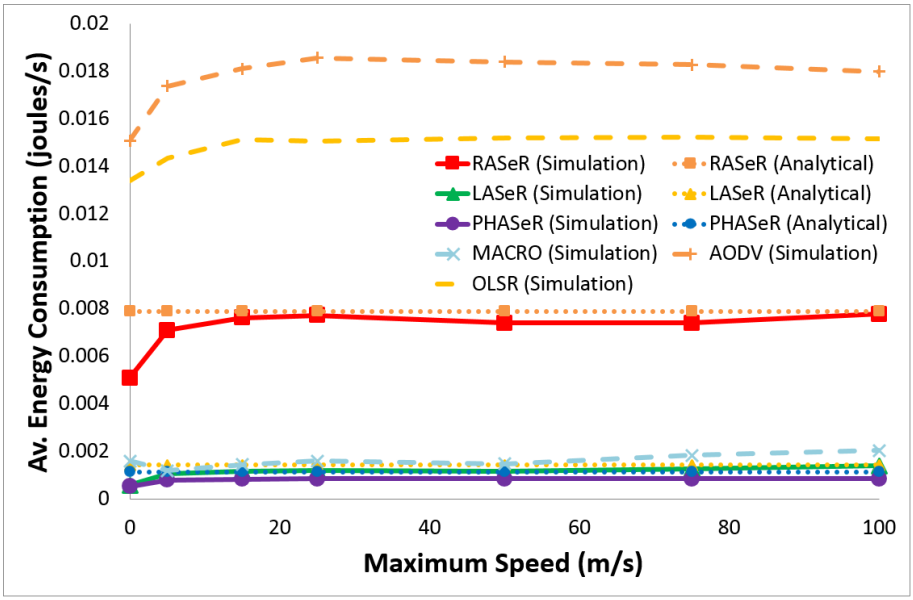


Fig. 6.8(e). Simulation and analytical energy consumption results for PHASeR, LASeR and RASeR, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*. With MACRO, AODV and OLSR simulations for comparison.

6.4.2 Scalability

Figures 6.9(a-e) give the simulation and analytical results for varying numbers of nodes. It uses the base parameters described above, with the number of nodes being varied between *[15, 25, 50, 75, 100]nodes*. In order to maintain a roughly similar node density across the simulations, the network area will also be varied by changing *L* to *[400, 600, 1000, 1200, 1500]m* respectively. Figure 6.9(a) shows that the PDR of LASeR is near perfect over the entire range and, save for a tiny dip to only 99.31% with 100 nodes, so is the PDR of RASeR, which dips to 97.59%. Also, the analytical results closely match the simulations. At the other end of the scale, AODV and OLSR give lower performance, which declines significantly as the number of nodes is increased, to the point of being unusable. This is due to the fact that neither AODV nor OLSR are not designed for this kind of high mobility scenario or for this type of network, in which all of the traffic is directed towards a single sink node. PHASeR also shows a steep decline in performance as the number of nodes is increased, which is due to the requirement for each packet to contain a larger number of frames. This necessitates an increase in the timeslot length and combined with the need for additional slot to accommodate the extra nodes, the length of a cycle becomes very large. This affects the time it takes for each node to determine its hop-count. As such, given the high level of mobility, the topology is changing at a rate that becomes increasingly more difficult to keep up with as more nodes are added. These results also highlight the limitations of MACRO, which performs well with low numbers of nodes. However, at 50 nodes and higher, both RASeR and LASeR outperform MACRO on all metrics. In the average end-to-end delay results in figure 6.9(b), AODV has the highest latency,

but, similarly to the mobility results, OLSR gives very low delay times. This occurs for the same reason as before, in that the high number of dropped packets free up bandwidth and reduce congestion for the successfully delivered packets. The delay of RASeR and LASeR are also low with a maximum of 199.56ms, with the analytical results reflecting this. Similarly, PHASeR's delay is low, but its analytical expression overestimates the effect of increasing the number of nodes. MACRO's delay begins low, but the added nodes cause the end-to-end delay to steadily climb and reach just over 1.5s, which is also reflected by its overhead results in figure 6.9(c). The overhead of AODV and OLSR also steadily increase as more nodes are added, with OLSR being the worse affected. LASeR and PHASeR retain a consistently low overhead of less than 56.22bits over the entire range. Contrastingly to the other protocols, RASeRs overhead declines as the number of nodes increases, which is due to the shift in balance of the timeslots being used for beacons and those being used for data packets. The increased number of nodes, creates an increase in traffic, which requires that more of RASeRs timeslots are used for transmitting data packets. This means that less beacon packets are sent, which reduces the overhead used relative to the amount of data being delivered. As such, when the number of nodes is increased, the relative overhead decreases. However, this is not the reason why there is a similar trend in the energy results in figure 6.9(e). In this graph, since the energy consumption is a node average, if each node only used a fixed amount of power, then the results would be static as the average level of power consumption would remain constant. However, in these simulations this is not the case, since every node is generating data, as more nodes are added the traffic level also increases. In RASeR, the increase in traffic from a single node will demand an increase in the power consumption over the whole network, but because this increase is not proportional to the initial level of consumption the average energy consumption will go down, which is predicted by the analytical results. LASeR's energy consumption shows an increase as more nodes are added, but still remains low with a maximum of 0.0021joules/s. The consumption of PHASeR remains consistently lower than 0.0013joules/s over the entire range, although this is due to the significant packet loss caused by increasing the number of nodes. MACRO gives reasonably low energy results, which increase considerably with the higher levels of nodes, up to 0.0084joules/s. AODV and OLSR also show a large increase to a maximum of 0.021joules/s. In terms of throughput, figure 6.9(d) shows RASeR and LASeR to have a steady increase, proportional to the increase in traffic, as would be expected, which is closely represented by the analytical results. MACRO also shows a steady increase, although with a lower gradient. Whereas PHASeR seems to increase well to begin with, before reaching its limit and flattening out at around 1041.6bits/s. AODV also seems to follow this trend. Contrastingly, OLSR shows no increase in throughput over the entire range and remains low.

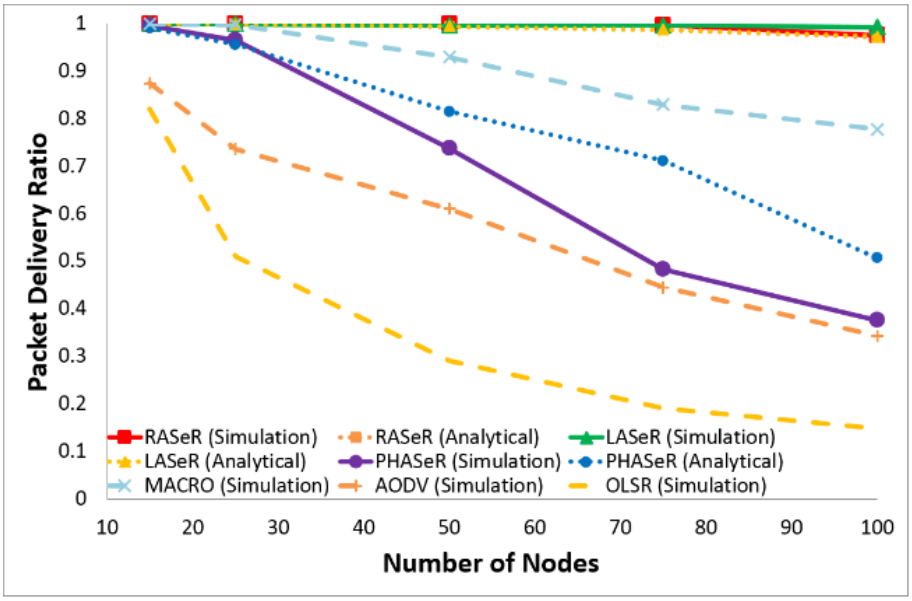


Fig. 6.9(a). Simulation and analytical PDR results for PHASeR, LASeR and RASeR, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*. With MACRO, AODV and OLSR simulations for comparison.

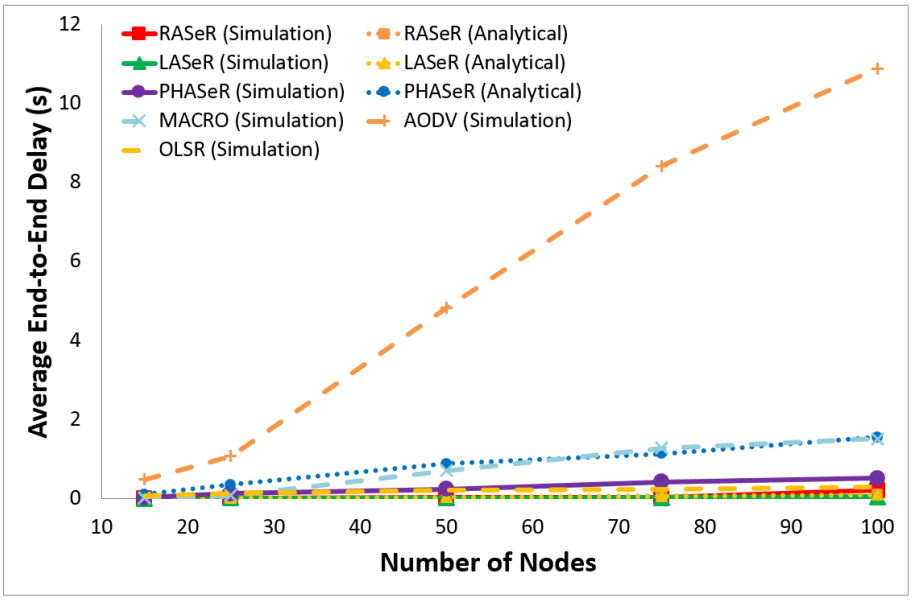


Fig. 6.9(b). Simulation and analytical end-to-end delay results for PHASeR, LASeR and RASeR, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*. With MACRO, AODV and OLSR simulations for comparison.

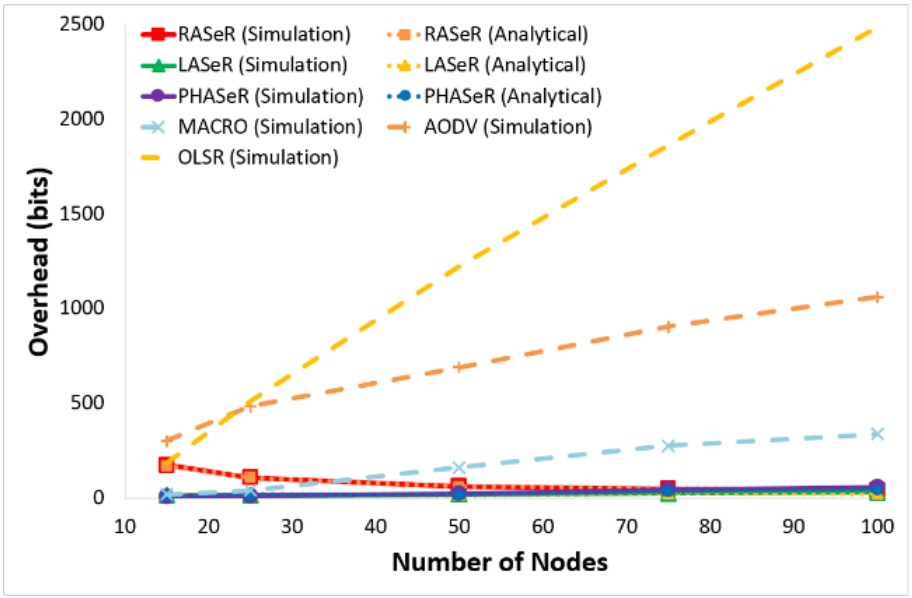


Fig. 6.9(c). Simulation and analytical overhead results for PHASeR, LASeR and RASeR, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*. With MACRO, AODV and OLSR simulations for comparison.

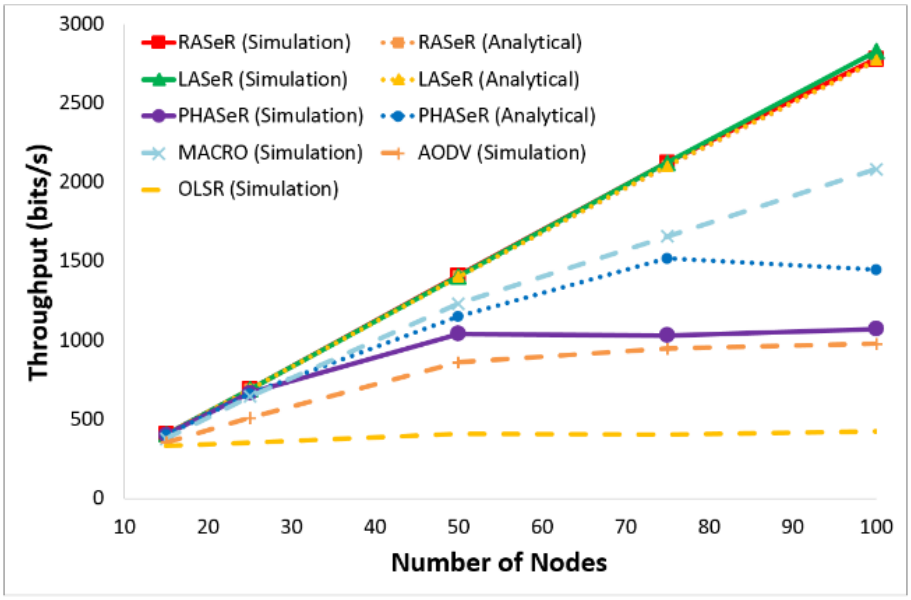


Fig. 6.9(d). Simulation and analytical throughput results for PHASeR, LASeR and RASeR, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*. With MACRO, AODV and OLSR simulations for comparison.

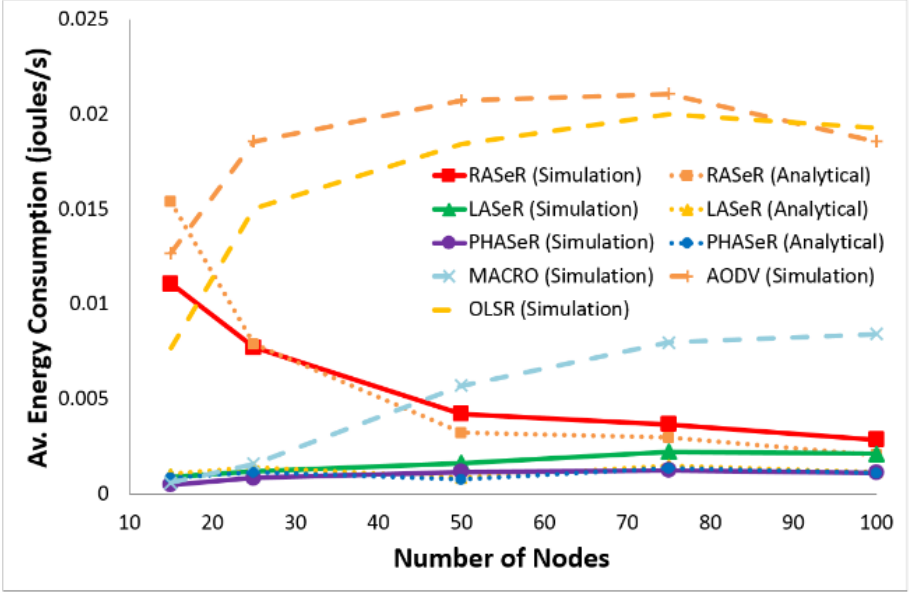


Fig. 6.9(e). Simulation and analytical energy consumption results for PHASeR, LASeR and RASeR, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*. With MACRO, AODV and OLSR simulations for comparison.

6.4.3 Traffic

Figures 6.10(a-e) present the traffic results, which vary the packet generation rate of each node as *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*. This corresponds to a network wide generations rates of *[2.4, 12, 24, 60, 120, 240]pk/s*. As with the previous results, RASeR and LASeR, in both simulation and analytical give near perfect PDR over the entire range of packet generation rates, with a minimum values of 98.80% and 99.74% respectively, as shown in figure 6.10(a). AODV and OLSR yield the worse performance, which declines significantly as the traffic level increases. MACRO gives near perfect PDR with a minimum of 98.97%, until *10pk/s*, where its limit seems to have been surpassed and its performance drops to only 51.91%. This is also corroborated by its maximum delay of 5.02s in figure 6.10(b) and its maximum throughput of 3337.02bits/s in figure 6.10(d). PHASeR's PDR actually increases initially with the added traffic. This is because PHASeR's GTDMA schedule is defined by the frequency at which packets are generated. So, as the generation rate increases, nodes may transmit more often, which improves end-to-end delay. This improvement in latency is illustrated by PHASeR's delay results, which show that the average end-to-end delay decreases as the packet generation rate. This increase in efficiency subsequently helps PHASeR deliver more packets, which results in a higher PDR. The delay results in figure 6.10(b) for RASeR and LASeR are consistently low for both simulation and analytical, with a maximum of 5.41ms. AODV has a high delay, which increases dramatically as the traffic level increases, whereas OLSR has a low initial delay due to its very low PDR, but it reaches its limit and begins to increase its latency at *10pk/s*. In figure 6.10(d), the throughput of PHASeR, LASeR and RASeR shows a proportional increase with the added traffic, which the analytical results also mimic. Contrastingly, AODV

and OLSR give significantly worse performance, which is also reiterated in the energy results. The other four protocols all show consistently increasing energy consumption, with the best results from PHASeR. The overhead in figure 6.10(c), illustrates how PHASeR, LASeR and MACRO all yield very low results across the entire range, with the maximum being from MACRO at 110.79bits. RASeR, although low, shows an initial decline in overhead, which is caused by the same reason as the declining overhead when the number of nodes is increased. Essentially, an increase in traffic causes more data to be created and delivered, but with very little additional overhead, which causes the ratio of data bits to overhead to decrease. Contrastingly, in figure 6.10(e), the energy results show RASeR's energy consumption to increase, due to the added energy required to deliver the high amount of generated data. This trend is true of the other protocols also, with AODV and OLSR giving the worst performance overall. PHASeR gave the best energy consumption, with a maximum of 0.0079joules/s.

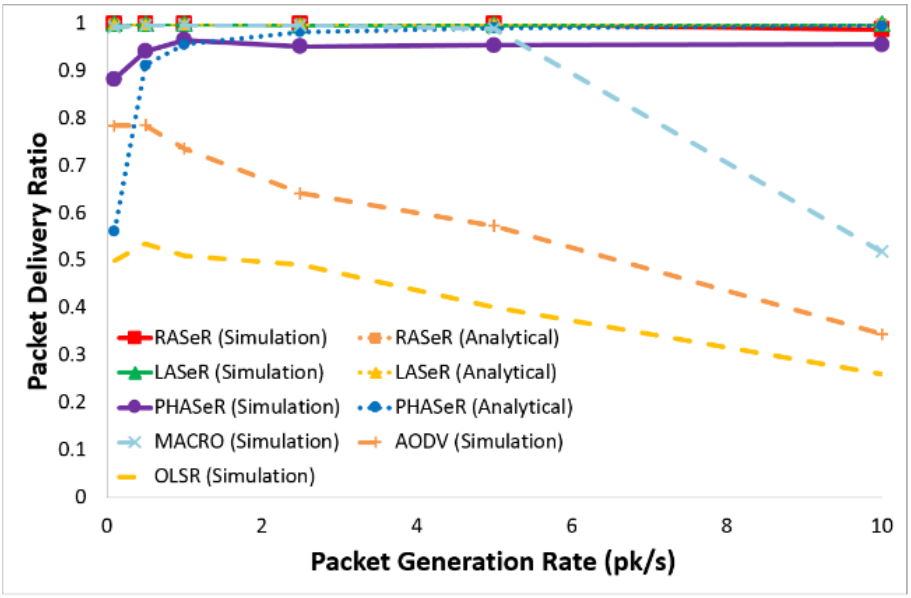


Fig. 6.10(a). Simulation and analytical PDR results for PHASeR, LASeR and RASeR, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*. With MACRO, AODV and OLSR simulations for comparison.

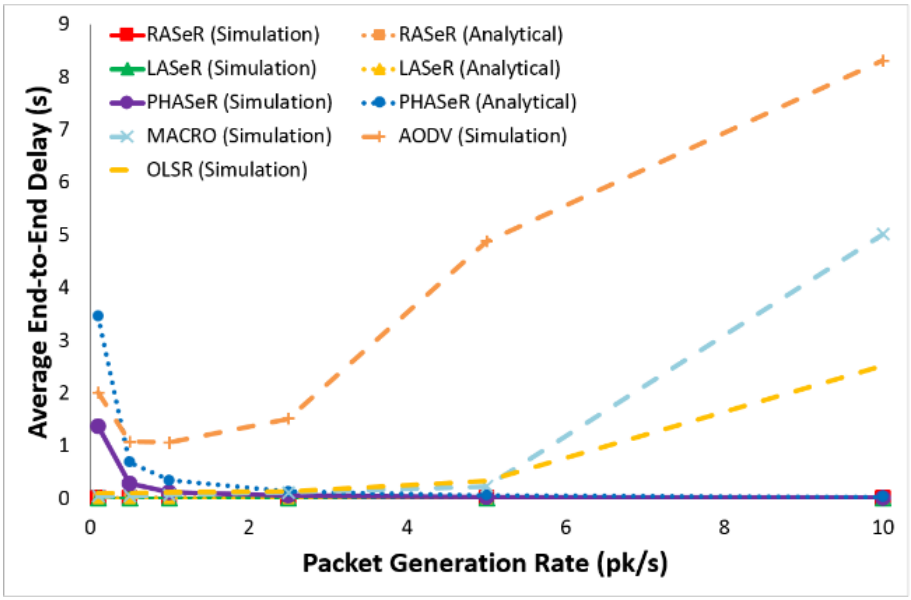


Fig. 6.10(b). Simulation and analytical end-to-end delay results for PHASeR, LASeR and RASeR, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*. With MACRO, AODV and OLSR simulations for comparison.

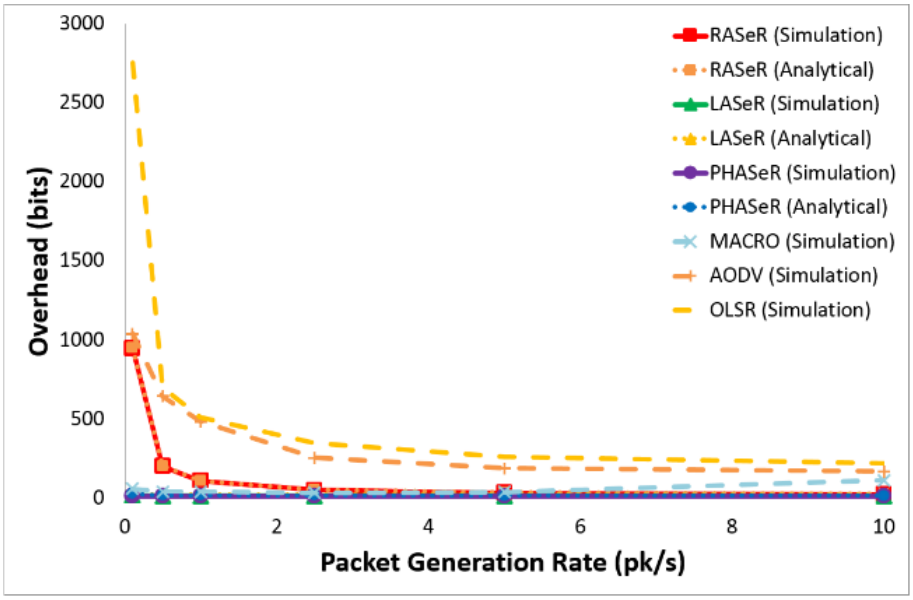


Fig. 6.10(c). Simulation and analytical overhead results for PHASeR, LASeR and RASeR, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*. With MACRO, AODV and OLSR simulations for comparison.

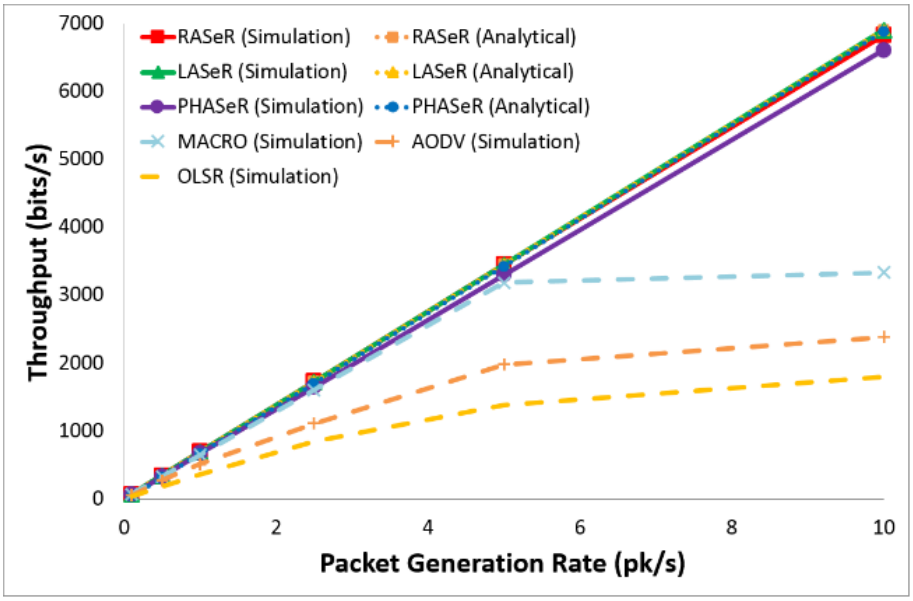


Fig. 6.10(d). Simulation and analytical throughput results for PHASeR, LASeR and RASeR, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*. With MACRO, AODV and OLSR simulations for comparison.

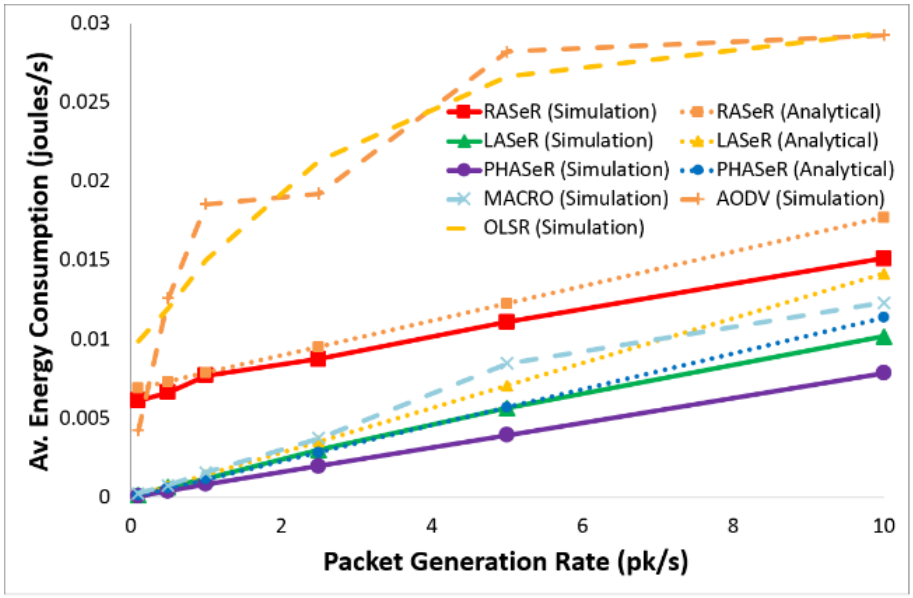


Fig. 6.10(e). Simulation and analytical energy consumption results for PHASeR, LASeR and RASeR, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*. With MACRO, AODV and OLSR simulations for comparison.

6.4.4 Conclusion

In general, both RASeR and LASeR show near perfect PDR and throughput over all scenarios at all ranges. LASeR's delay and overhead are kept low in all scenarios by utilising the existing location information, which results in a very low energy consumption. The need for RASeR to maintain its gradient metric requires additional overhead, which increases its energy consumption but allows it to retain very low end-to-end delay over all the presented scenarios. PHASeR tends to get worse with large numbers of nodes however its performance improves in

scenarios with higher traffic levels over all metrics. MACRO seems to perform well in scenarios with a low number of nodes and medium to low traffic levels. However, when the number of nodes is large or the traffic level is high, MACRO's performance drops significantly over all metrics.

In terms of comparison, table 6.1 gives the average values for the protocols over all the scenarios. Whilst this is not an accurate portrayal of a protocols performance in a given situation, it does serve as an easy way of comparing the protocols general ability. The table clearly shows LASeR to give the best overall performance, with the highest PDR and throughput, and the lowest end-to-end delay and overhead. LASeR's energy consumption is second only to PHASeR, which also has very low overhead, but slightly diminished PDR. RASeR is a close second to LASeR, achieving similar levels in PDR, delay and throughput. However, its slightly increased levels of overhead cause it to use slightly more energy than LASeR. MACRO shows slightly worse performance than both LASeR and RASeR, especially in terms of end-to-end delay and throughput. Whilst MACRO's energy consumption is low on average, RASeR's energy consumption is far superior in scenarios with 50 or more nodes. AODV and OLSR consistently performed the worse, with the exception of OLSR's delay, which is generally low due to its significant packet loss.

	PDR (%)	**Average End-to-End Delay (s)**	**Overhead (bits)**	**Throughput (bits/s)**	**Av. Energy Consumption (joules/s)**
PHASeR	87.46	0.2262	17.24	1204.87	0.0014
LASeR	99.86	0.0059	15.96	1460.02	0.0021
RASeR	99.74	0.0195	140.06	1453.21	0.0074
MACRO	94.02	0.5686	86.83	1124.01	0.0037
AODV	65.98	3.1717	545.67	762.75	0.0182
OLSR	44.31	0.2899	870.97	496.34	0.0166

Table 6.1. The average values for PHASeR, LASeR, RASeR, MACRO, AODV and OLSR for PDR, average end-to-end delay, overhead, throughput and average energy consumption, over all scenarios.

Overall, the proposed protocols have been shown to be highly reliable with low latency, which was one of the key aims of this work.

6.5 Fading Channel Evaluation

Propagating radio waves experience many different phenomena as they travel, such as reflection, diffraction and scattering. Subsequently, the signal received at the destination antenna is made up of multiple components from the many different paths that the signal can take. Combining this with the inherent noise that is introduced and the fact that a line-of-sight from the transmitter to the receiver may not exist, means that there is the possibility of the data becoming so error prone that it is indecipherable. Additionally, when considering mobile

devices, there is also the possibility of a Doppler shift causing the received carrier frequency to be altered.

The majority of results given previously in this thesis use a fixed radius, perfect physical layer, which mitigates the channel effects of path loss and fading. This was done to allow for the closer analysis of the protocol, without the addition of errors created by the medium. This also maintained the generality of the results, since the PHY layer parameters are very hardware dependant. However, in reality, the channel is an unavoidable obstacle and needs to be accounted for. A common choice for this is to model a Rayleigh fading channel, which does not assume that there is a dominant line-of-sight signal and accounts for the multipath fading effects of a realistic environment. As such, in this chapter the effects of a fading channel on the RASeR protocol will be evaluated. This will highlight the impact that channel errors have on this type of protocol.

6.5.1 Modelling

The simulations are modelled on the MEMSIC IRIS mote [4.20], which uses a low power AT86RF230 transceiver. The transceiver uses OQPSK to transmit at *250kbps*, with a bandwidth of *2.8MHz* and a power between *0dBm* and *6dBm*. For the purpose of these simulations, no error correction was used, so any error in the reception of a packet would result in it being dropped. The parameters used are summarised in table 6.2.

Physical Layer Parameters	
Fading Type	Rayleigh
Modulation	OQPSK
Pathloss Exponent	2.8
Data Rate	250kbps
Bandwidth	2.8MHz
Transmission Power	6dBm
Error Correction	None

Table 6.2. Summary of the physical layer parameters used in simulation.

By default, OPNET assumes a Gaussian channel with free-space path loss, which will significantly underestimate the effect of the medium. The simulator introduces channel errors by calculating the received power and the noise to get an SNR value. A BER curve is then used to estimate the number of errors present in a specific packet. The BER curve for OQPSK in an AWGN (Additive White Gaussian Noise) channel is given in figure 6.11 and the formula used to ascertain the SNR is given as

$$SNR = \frac{P_{tx}\left(\frac{f_{max} - f_{min}}{BW}\right)\left(\frac{\lambda}{4\pi d}\right)^2 G_{tx} \cdot G_{rx}}{(BW \cdot N_{Amb}) + (T_{BG} \cdot N_F \cdot BW \cdot k)} \qquad (53)$$

where P_{tx} is the transmission power, BW is the bandwidth, f_{max} and f_{min} are the maximum and minimum frequencies in which the transmitter and the receiver overlap, d is the distance between the transceivers, λ is the wavelength of the carrier frequency, G_{tx} and G_{rx} are the gains of the transmitting and receiving antennas respectively, N_{Amb} is the ambient noise level, T_{BG} is the background temperature, N_F is the receivers noise figure and k is the Boltzmann constant.

In order to improve the modelling of the channel, the path loss will be modelled as suggested in [6.2], which will give the following for SNR

$$SNR = \frac{P_{tx}\left(\frac{f_{max} - f_{min}}{BW}\right)\left(\frac{\lambda^2 d_0^{(\gamma-2)}}{4^2\pi^2 d^{\gamma}}\right)G_{tx} \cdot G_{rx}}{(BW \cdot N_{Amb}) + (T_{BG} \cdot N_F \cdot BW \cdot k)} \qquad (54)$$

where d_0 is a reference distance and γ is the path loss exponent.

In addition to changing the path loss, the AWGN BER curve will be replace with the Rayleigh fading curve, as shown in figure 6.11.

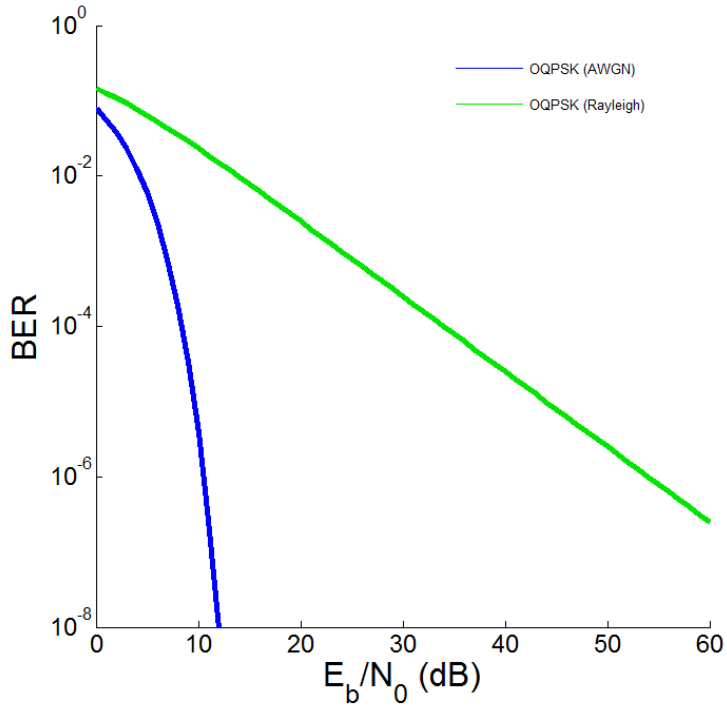


Fig. 6.11. BER Curves for OQPSK in an AWGN Channel and OQPSK in a Rayleigh fading Channel.

To evaluate effect of these changes packet success ratios were gathered over distances of up to 10km, maintaining the same power, bandwidth and data rate as described above. The results in figure 6.12 clearly show how the default AWGN channel with freespace path loss significantly overestimates the ability of the receiver, with perfect reception at a distance of 2km. In sharp contrast to this the Rayleigh fading channel, with the accurate path loss, shows a

steep decline with only 0.24% of packets being successfully delivered at 2km. In general, the fixed radius PHY at 250m, appears to be a reasonably close fit to the more accurately modelled Rayleigh fading channel, with an average error of only 3% in terms of the packet success rate.

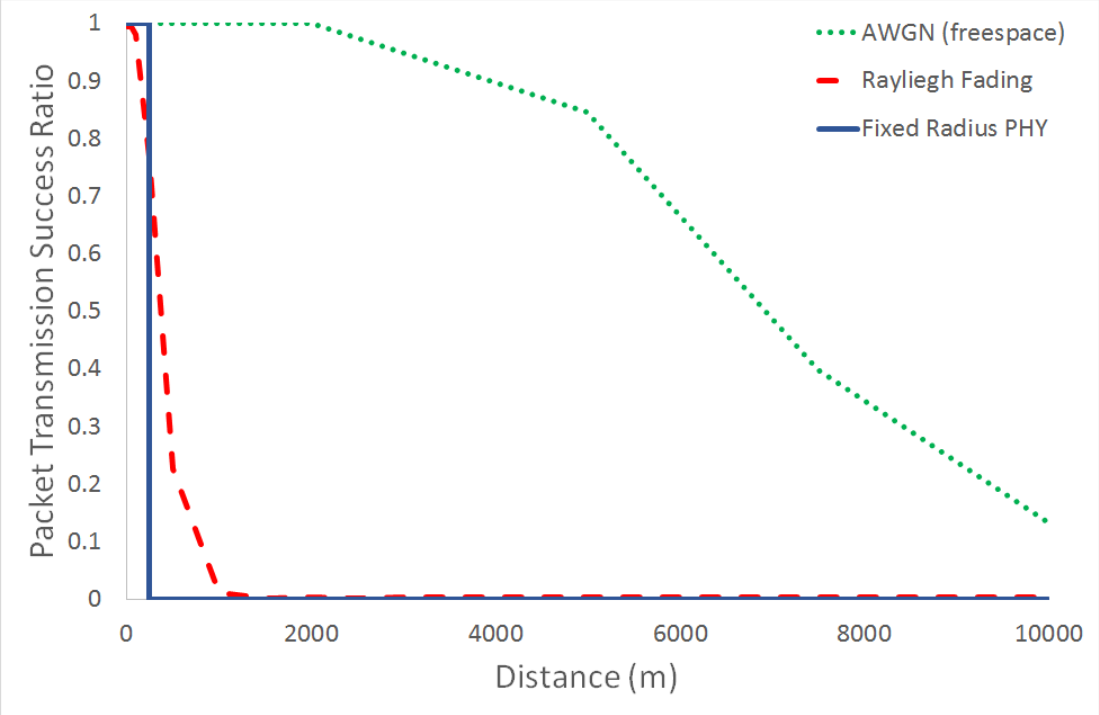


Fig. 6.12. Packet transmission success ratios for varying distances, using the default AWGN channel with free space path loss, a Rayleigh fading channel with accurate path loss and a fixed radius PHY.

6.5.2 Simulation and Results

Results were gathered based on the IRIS motes, using the parameters as above for the Rayleigh channel only to give an indication of how the protocol reacts given the worst case scenario. The AWGN channel wasn't used, since having perfect reception at distances of more than 2000m is unrealistic and would make a multiphop routing solution unnecessary. As stated in table 6.2, the path loss exponent was set to 2.8, which describes a usual outdoor environment [6.2]. The performance measures of PDR, average end-to-end delay, overhead, throughput and average energy consumption, as defined in chapter 4, were gathered in scenarios with varying mobility, scalability and traffic levels for RASeR, using both the fixed radius channel model and the Raleigh fading model.

Figures 6.13(a-e) give the mobility results for maximum speeds of *[0, 5, 15, 25, 50, 75, 100]m/s*. In general, figure 6.13(a) shows that the fading has caused a slight reduction in PDR, but the reliability of the protocol still remains above 95.69%. The delay in figure 6.13(b) shows very little change, with the fading results actually giving better latency at 100m/s. However, this is due to the slower packets being lost, which gives an improved average. The overhead in figure 6.13(c) and energy results in figure 6.13(e) highlight the increase in transmissions

required to maintain the high PDR with a maximum increase of 133.36bits and 0.0127joules/s respectively. Figure 6.13(d) shows that the throughput is high, with only a slight degradation in comparison to the fixed radius model.

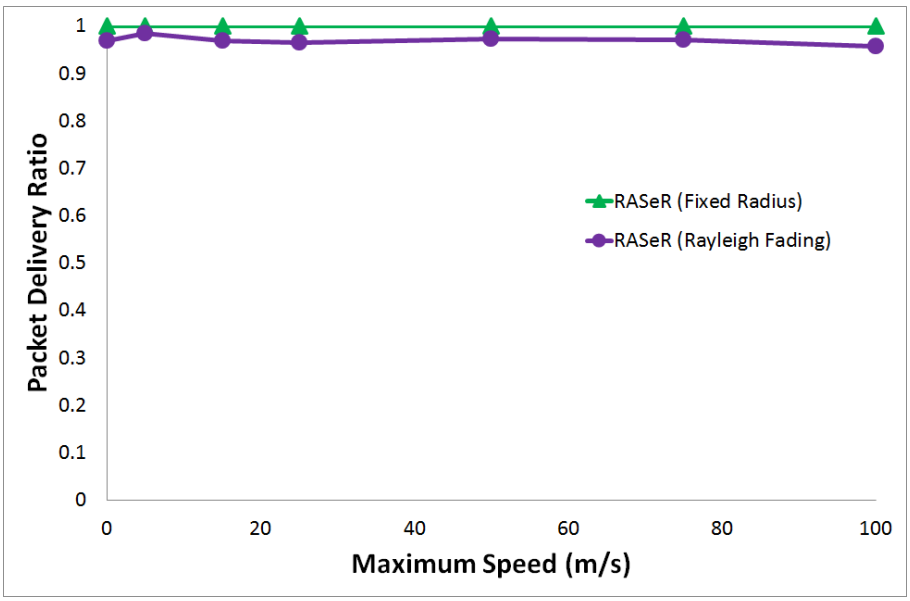


Fig. 6.13(a). Simulation PDR results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

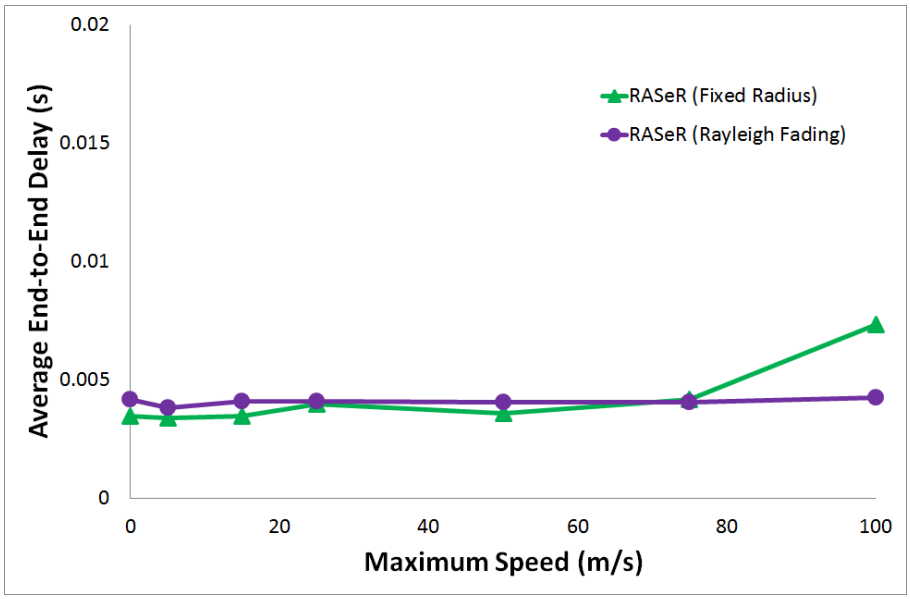


Fig. 6.13(b). Simulation end-to-end delay results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

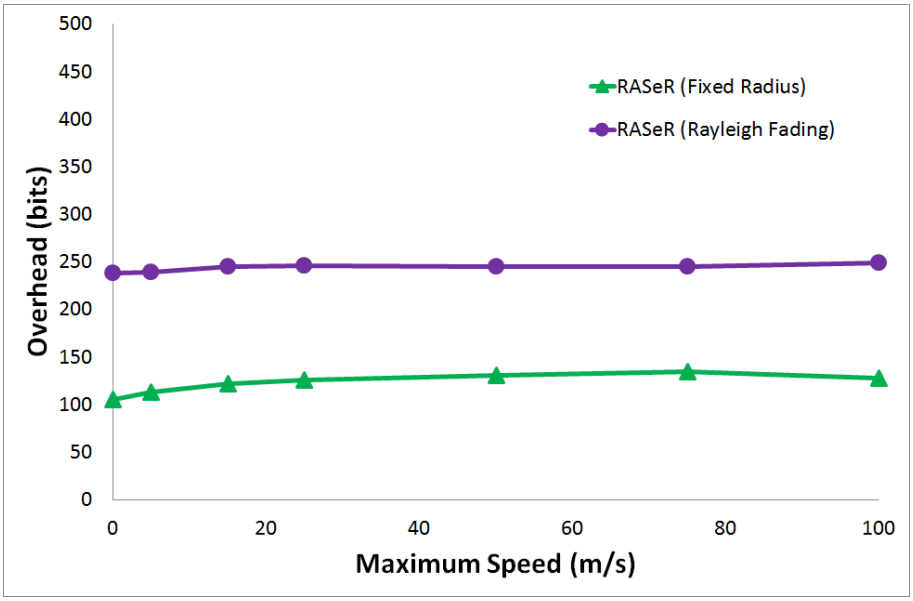


Fig. 6.13(c). Simulation overhead results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

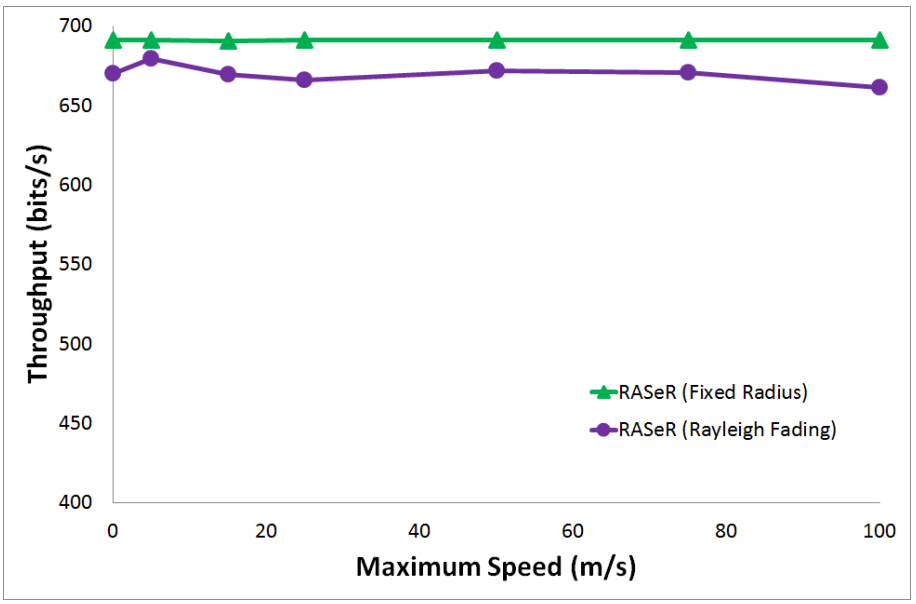


Fig. 6.13(d). Simulation throughput results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

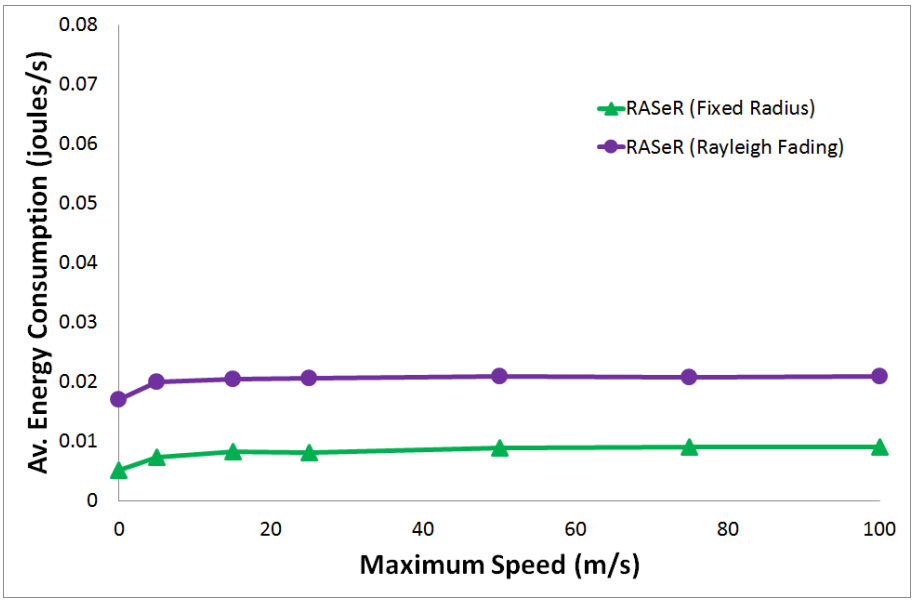


Fig. 6.13(e). Simulation energy consumption results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

The scalability results are given in figures 6.14(a-e) for *[15, 25, 50, 75, 100]nodes* with the square network lengths of *[400, 600, 1000, 1200, 1500]m* respectively. The PDR results in figure 6.14(a) show a definite decline in delivery ratio after 25 nodes. One reason for this is that the larger network areas mean that a packet will have to traverse more hops in order to reach the sink, which increases the chances of packet loss. However, the major contributing factor in this scenario is the increased pathloss from nodes being further apart. The larger network area means that the average distance between two nodes is increased, which causes significantly higher path loss, resulting in a high BER and subsequently, a large amount of packet loss. These two combining effects cause RASeR's performance to drop dramatically. This is also shown in the delay results in figure 6.14(b), which show a large increase in latency towards higher number of nodes, which is a result of the nodes behaving like they are disconnected and waiting to become reconnected before they attempt to transmit their packets. The overhead in figure 6.14(c) and energy in figure 6.14(e) shows a slight increase over the fixed radius channel from multiple attempts to deliver packets. Figure 6.14(d) highlights this performance degradation after 25 nodes in terms of the throughput, which seems to reach at peak of 1379.58bits/s at 75 nodes.

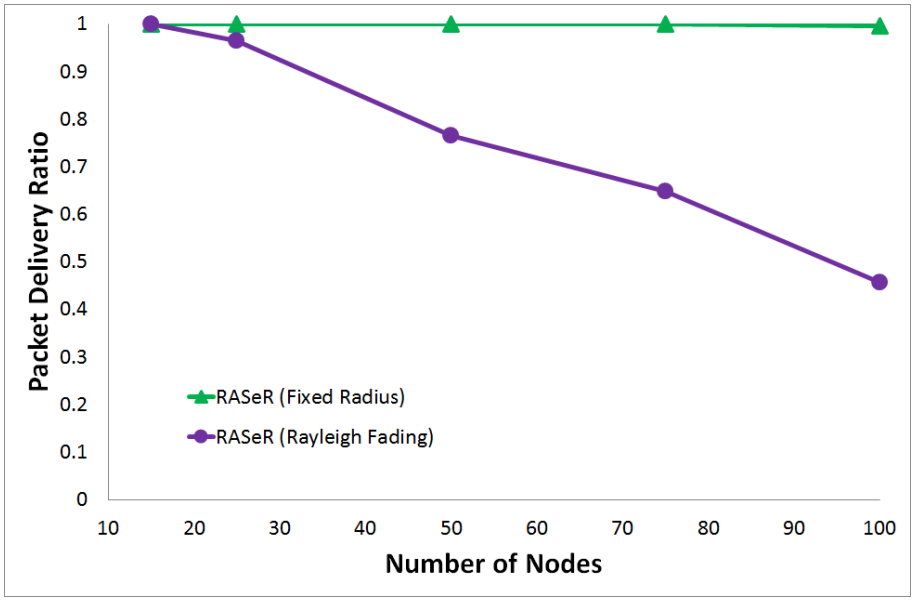


Fig. 6.14(a). Simulation PDR results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

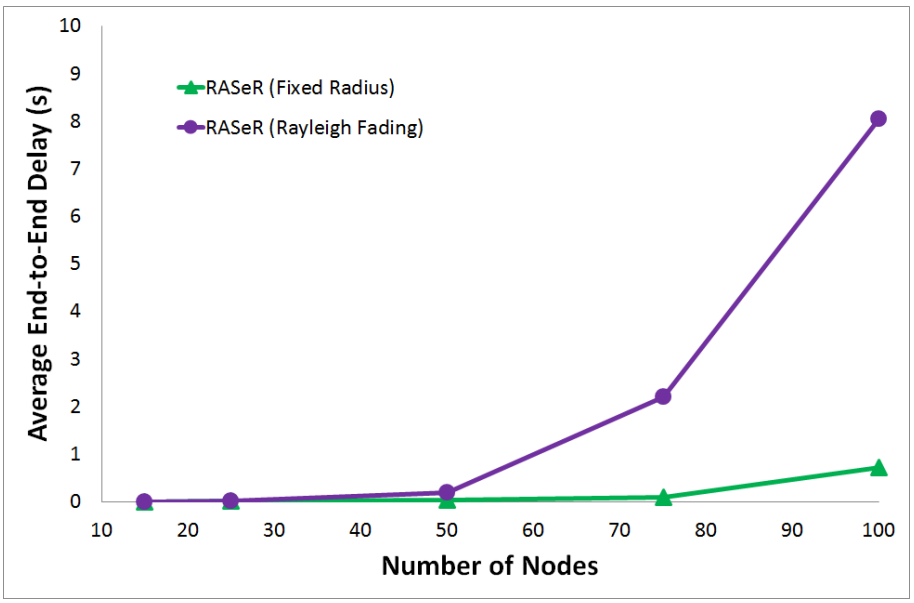


Fig. 6.14(b). Simulation end-to-end delay results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

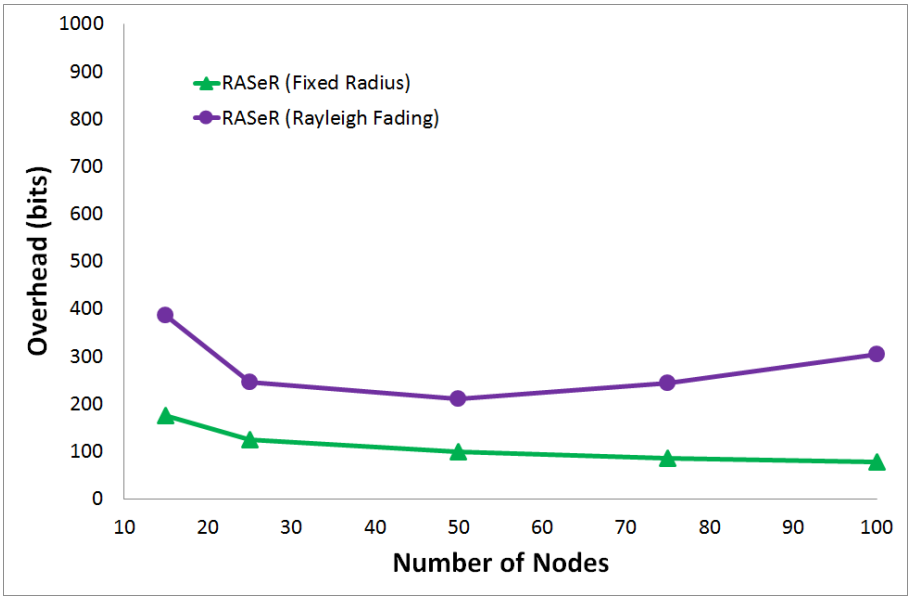


Fig. 6.14(c). Simulation overhead results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

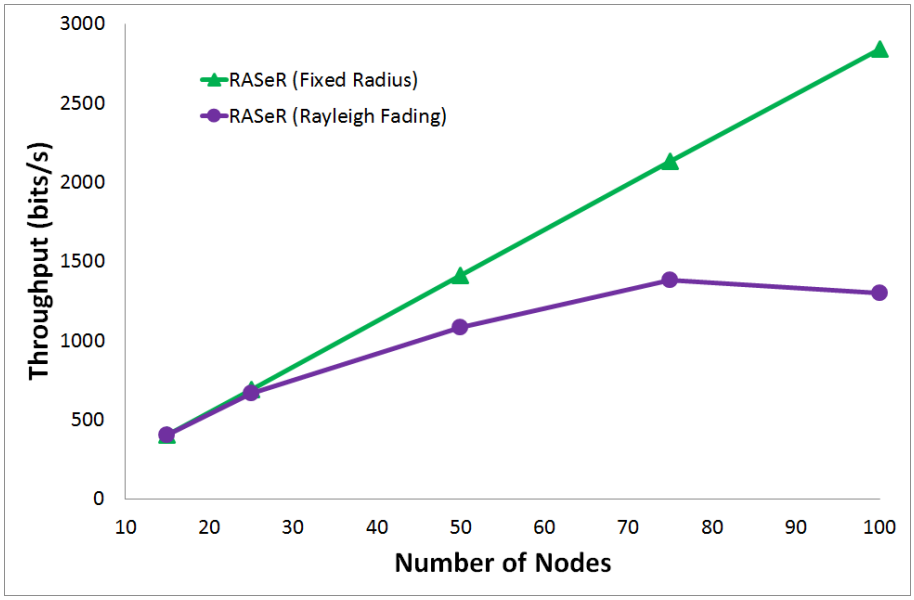


Fig. 6.14(d). Simulation throughput results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

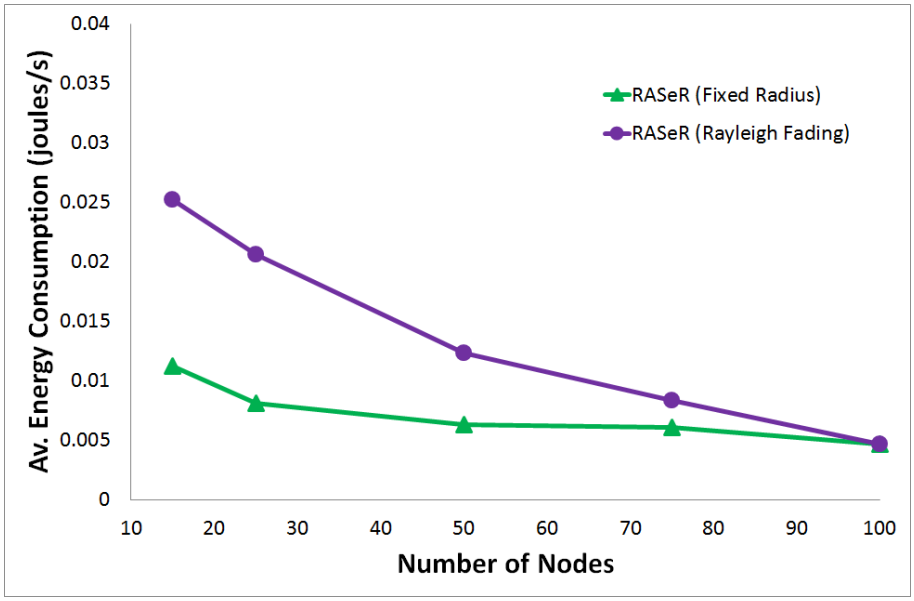


Fig. 6.14(e). Simulation energy consumption results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

The results in figures 6.15(a-e) show how the protocol behaves under traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pks/s*. The PDR levels in figure 6.15(a) show only a very minor decrease in the fading results, with less than a 3.6% decrease over the entire range. This minor decrease is also echoed by the throughput results in figure 6.15(d). The delay results in figure 6.15(b) show a low level of latency, but a small increase of 9.27ms at 10pk/s, which is due to the added load causing congestion. In terms of the overhead in figure 6.15(c) and energy levels in figure 6.15(e), the results from the fading channel show a slight increase from the protocol requiring additional transmissions in order to maintain the high level of reliability.

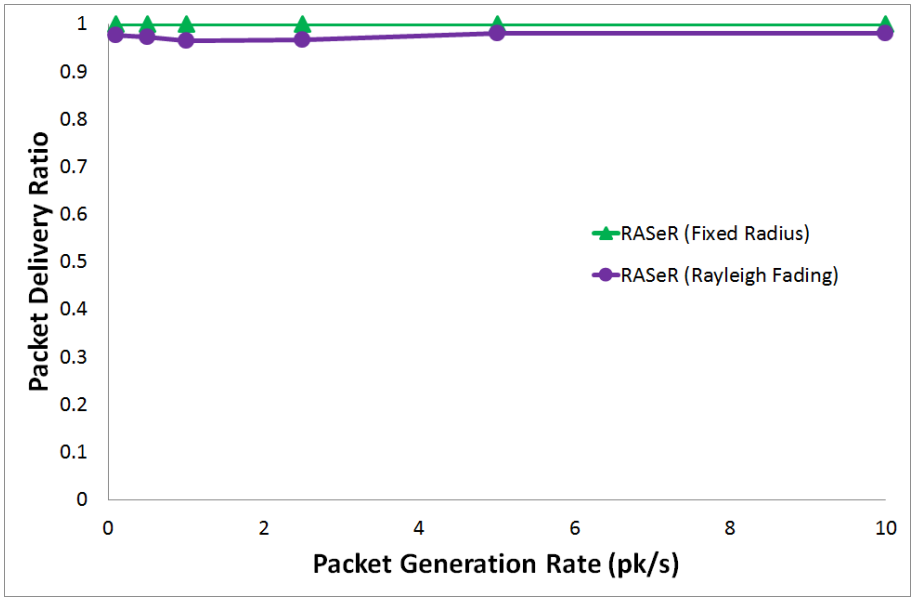


Fig. 6.15(a). Simulation PDR results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

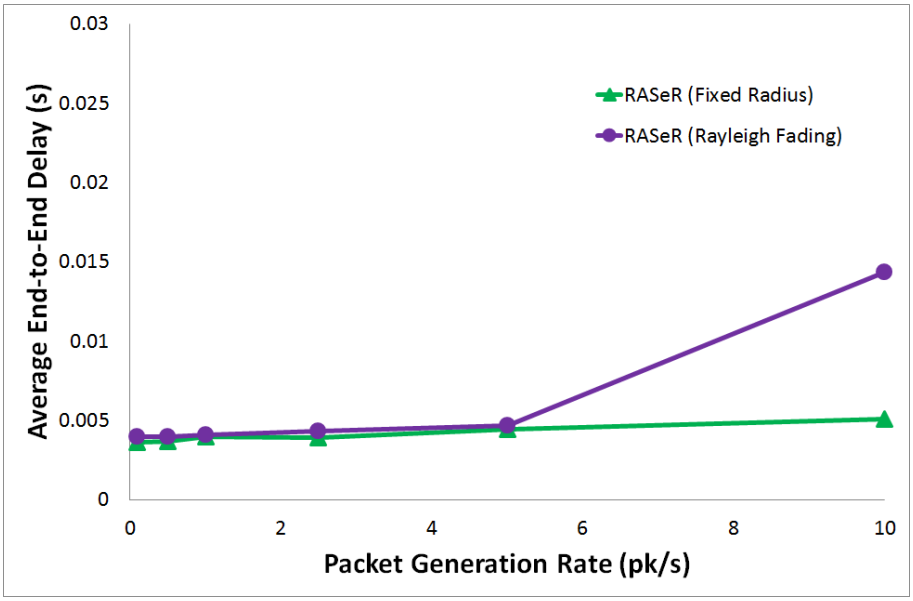


Fig. 6.15(b). Simulation end-to-end delay results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

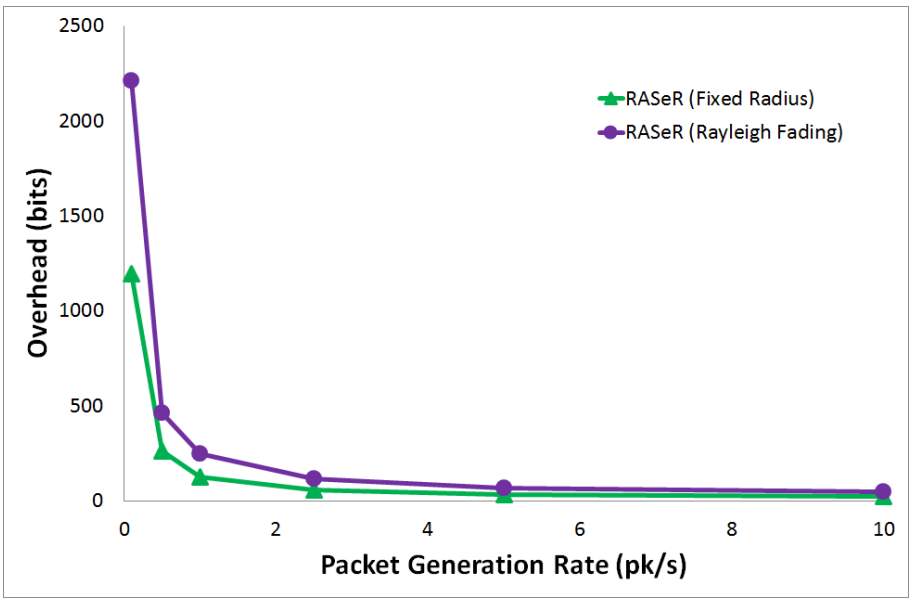


Fig. 6.15(c). Simulation overhead results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

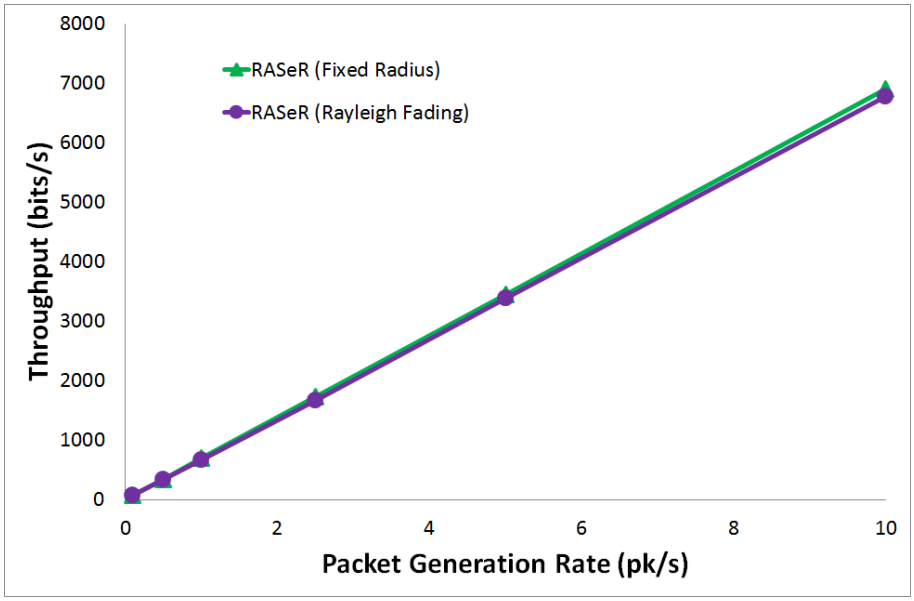


Fig. 6.15(d). Simulation throughput results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

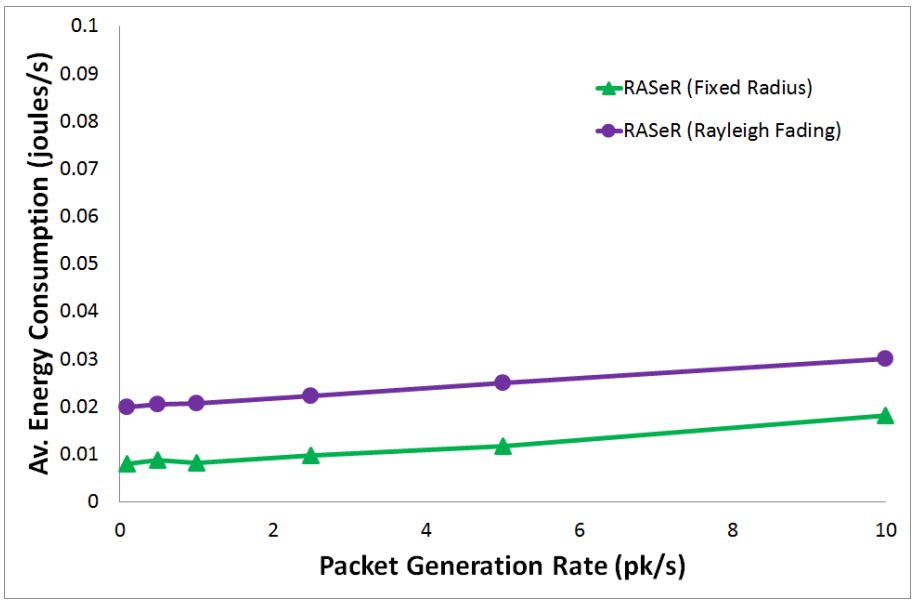


Fig. 6.15(e). Simulation energy consumption results for RASeR in both a fixed radius PHY and a Rayleigh fading channel, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

6.5.3 Conclusion

Expectedly, the fading has the most effect in scenarios in which there is a large average distance between nodes. Subsequently, this causes significant packet loss due to the larger number of errors occurring in a packet. Given that the nodes cannot hear each other due to the large number of errors, they begin to behave as though they are disconnected, which creates large delays. Essentially, this means that the main issue is in the network size and transmitter power and not with the protocol, since RASeR has been shown to be scalable to large numbers of nodes in the results presented earlier in this section.

However, in all environments with a lower average distance between nodes the protocol performs very well, even with the detrimental effects of a fading channel. RASeR gives high levels of PDR and low delay over varying levels of mobility and traffic, which is also reflected in the throughput results. There is a slight increase in overhead and energy consumption due to the requirement for additional transmissions. In general, the reliability of the protocol is maintained due to the protocols use of route diversity, despite the highly error prone channel conditions.

6.6 CONCLUSION

This chapter has described the simulation environment of OPNET and presented results of the three proposed protocols. Their performance was evaluated in terms of mobility, scalability and traffic levels over five metrics. All the protocols performed well, with RASeR and LASeR showing significant improvements over the comparisons. RASeR was also evaluated in a fading environment, which showed only a small degradation in the protocols performance, though it remained highly reliable. Based on these results and the demanding requirements of MWSN applications, the next chapter will explore potential alternative MAC protocols for LASeR.

CHAPTER 7

LASER MAC PROTOCOLS

7.1 INTRODUCTION

In LASeR, the use of location awareness negates the protocols reliance on the GTDMA MAC to maintain an up-to-date gradient metric. As such it is possible to use LASeR with other MAC protocols. This chapter investigates potential alternative MACs by first determining the requirements for the MAC, then analysing the literature to identify candidates for testing. As well as selecting protocols from the literature, novel protocols are also developed to suit LASeRs needs. Subsequently, both the selected and the proposed protocols will then be evaluated and analysed by simulation for their suitability.

7.2 MAC SELECTION AND DESIGN

The criteria for selecting MAC protocols for testing are based on their ability to be used in harmony with LASeR, no need for additional hardware and no need for additional communications overhead. MACs that require additional overhead have been rejected due to the delay incurred. Since LASeR is designed to have low end-to-end delay times, the added overhead will increase the latency of a packets path from sensor to sink. Also, in highly mobile networks, the added delay may reduce the protocols ability to adapt to a frequently changing topology in a timely manner. Once chosen, the suitable MACs will then be tested through simulation to evaluate their reliability, latency and energy consumption when used with LASeR.

The blind forwarding technique used in LASeR means that there's no prior communication between the transmitting node and the receiving node before the actual transmission of the data packet. This severely limits the number of suitable MAC protocols. Additionally, the MWSN MAC protocols reviewed in chapter 3, are designed for use in a hierarchical architecture and the majority are based on a low duty cycle, which would not be suitable for low latency protocols such as LASeR. Also, in high mobility systems, the added delay may inhibit the protocols ability to react to frequent topology changes. For these reasons, the low duty cycle MACs have been deemed unsuitable. LASeR is also a flat routing protocol in which all the sensor nodes are expected to be mobile, so the cluster based MACs and MACs with static nodes have also been rejected.

Additionally, it should be noted that CDMA and FDMA are unsuitable due to the requirement of expensive hardware. TDMA requires a central station to allocate time slots, which is useful for hierarchical routing protocols. However LASeR is flat and the overhead required to distribute scheduling information across the whole network from a central node, such as the sink, would be overly large. Although, LASeR does work with a GTDMA as described in chapter 5.

7.2.1 Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

One of the most popular MAC layers is the 802.11 DCF MAC, which uses the technique of carrier sense multiple access (CSMA) with collision avoidance (CA). The 802.11 DCF MAC dictates that if the channel is sensed to be busy then the node wishing to transmit should wait for a random amount of time before trying again. The primary reason that this will not work with LASeRs blind forwarding is that after a single transmission multiple nodes may decide to forward the received packet. Subsequently they will sense the medium, which will likely be free, causing multiple nodes to forward the packet simultaneously, resulting in many collisions. Also, the 802.11 DCF MAC also defines the use of acknowledgements (ACKs) and a request to send (RTS), clear to send (CTS) handshake. These mechanisms also fail when used with LASeR: The ACK is used by the intended next-hop receiving node to indicate to the transmitting node that the packet was received successfully. However in LASeR, there is no single intended node, so ACKs should be disabled. Otherwise, it is likely that more than one receiving node will reply with an ACK. These multiple ACKs are highly likely to collide, which means it will not be received by the transmitting node and could cause the packet to be retransmitted unnecessarily. Similarly, as there is no intended path in LASeR, a RTS/CTS handshake cannot take place. If a RTS is broadcast, multiple nodes may respond with a CTS, causing collisions. Additionally, the responding nodes may not even be neighbours who will then decide to forward the packet.

For the purposes of testing, this work will consider a modified CSMA/CA MAC, with no ACKs or RTS/CTS handshakes (hereafter referred to simply as CSMA/CA). The operation of this modified MAC is shown in figure 7.1. The flow chart begins by checking that there is data to send, essentially this means that the MAC should only run once it has been passed some data from another layer or if it has some data waiting to be transmitted after a previous attempt. There is then a backoff flag, which indicates whether the backoff timer has expired or not been set. If this is the case then the node may listen to the channel and transmit if it's clear. If the backoff timer has been set and hasn't expired then it is decremented and the process is repeated. Additionally, if the channel is sensed as busy, then the backoff timer should be set.

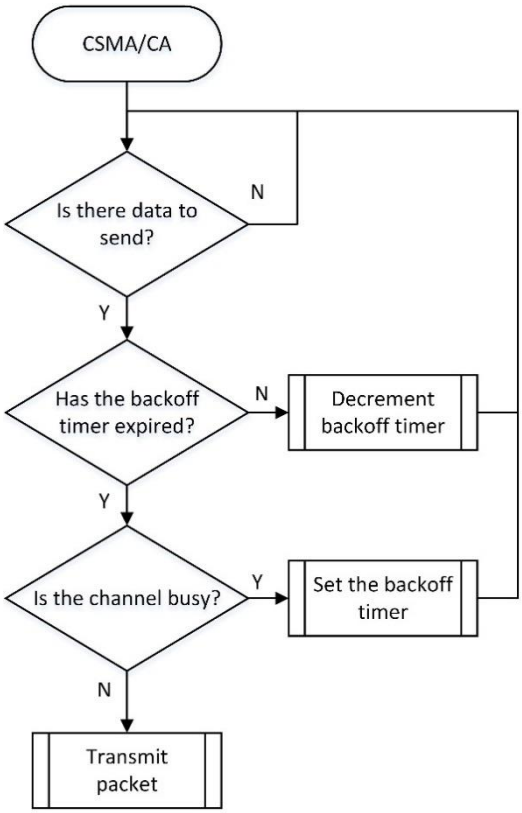


Fig. 7.1. CSMA/CA operational flow chart.

7.2.2 Carrier Sense Multiple Access with Dedicated Slots (CSMA/DS)

One alternative, novel technique proposed here is the use of dedicated channel sensing slots (CSS), in which the node will perform a clear channel assessment (CCA). This is a compromise between the CSMA/CA technique and a GTDMA, and is referred to as CSMA/DS (CSMA with Dedicated CCA Slots). In CSMA/DS, each node is allocated a small slot in which to sense the medium. If the medium is sensed to be free the node may then transmit. If a node begins to transmit then the medium will appear to be busy to the nodes with subsequent CCA slots. In this way other nodes will defer their transmissions and collisions can be avoided. The frame structure is shown in figure 7.2, which illustrates how the nodes all have sufficient time to sense the medium in their own slot and then transmit a packet. It can also be seen how a node with an earlier time slot may use the medium and essentially block transmissions by other nodes. This prevents any local collisions within a nodes neighbourhood from two nodes simultaneously sensing the medium to be free. As such it essentially replaces the RTS/CTS handshake, however the hidden and exposed node problems may still occur. It should also be noted that no ACKs or RTS/CTS handshakes are used in the CSMA/DS MAC. This MAC protocol takes advantage of the fixed number of nodes assumption made by LASeR, as such the number of CSSs is set to *n* before deployment and each node is allocated their slot.

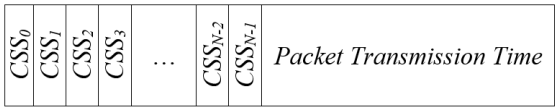


Fig. 7.2. CSMA/DS frame structure, where CSS_n is the channel sensing slot for node *n* in a network of *N* nodes.

The flow chart in figure 7.3 describes the operation of CSMA/DS. Firstly, there must be data to send, if there is data to send then the node will wait for its next dedicated CSS and then listen to the medium. If the channel is clear the packet will be transmitted, otherwise the transmission will be deferred to the next frame.

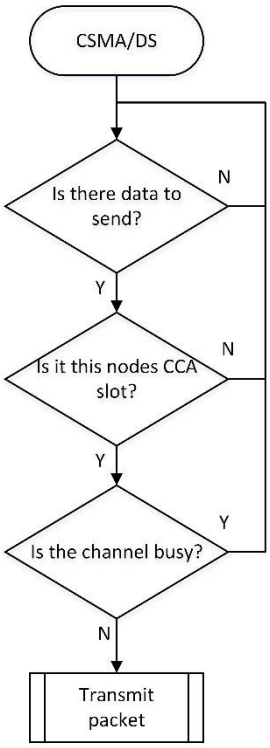


Fig. 7.3. CSMA/DS operational flow chart.

7.2.3 Network Division Multiple Access with Collision Avoidance (NDMA/CA) and Network Division Multiple Access with Dedicated Slots (NDMA/DS)

Since LASeR uses location awareness, this information could be utilised by the MAC layer a network division multiple access (NDMA) scheme is proposed. This technique divides the network area into cells such that spatial reuse can take place to increase efficiency. The NDMA method introduced here is similar to the location aware MAC protocol (LAMP) proposed in [7.1]. In the implementation presented here groups of non-interfering cells take it in turn to be active. If a node is in an active cell then it may contend for the medium. Figure 7.4 illustrates how the network may be split up in to cells, labelled *A* to *P*. The letters correspond to the order in which the cells are active, such that all of the cells with the same letter maybe active at the same time.

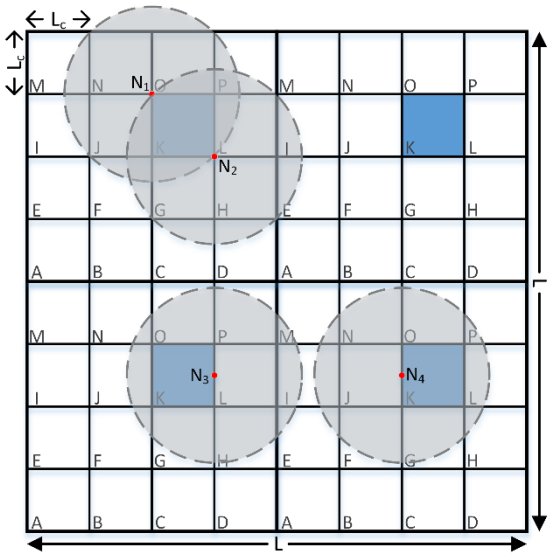


Fig. 7.4. NDMA network area split into cells, where L is the length of the area and L_c is the length of a single cell. Nodes are denoted as N_i where i is their node number.

In figure 7.4 the two constraints for the cells size are shown. In the figure the highlighted k cells are active. The first constraint is that every node in the cell should be able to hear every other node in the same cell. Such that, when one node is transmitting the other nodes in the cell can hear it and defer their own transmissions. This means that the diagonal length of the cells must be less than the transmission radius, r, so the height of a cell, L_c, must be less that $r/\sqrt{2}$, this is shown by nodes N_1 and N_2. The second constraint is that the transmissions from two nodes in different active cells should not collide. So active cells should be more than $2r$ apart, in other words, active cells must have at least $2\sqrt{2}$ cells between them, as illustrated by nodes N_3 and N_4.

Since nodes in the active cells will have to compete between each other for the medium, NDMA cannot be used alone. It is therefore suggested to use the collision avoidance mechanism from the CSMA/CA MAC layer. This protocol will be referred to as NDMA/CA. NDMA/CA allows nodes in active cells sense the medium and if it's free they may transmit. If the medium is sensed to be busy a backoff counter is started, such that if the timer expires and the cell is still active, the node may try again. Alternatively, NDMA may be used with dedicated CSS slots (NDMA/DS), in which nodes in active cells use their allocated slot to sense the medium. If it's free they may transmit, otherwise they will have to wait until the cell becomes active again. The use of NDMA eliminates the hidden node problem and local collisions are avoided with the dedicated CCA slots, NDMA/DS is collision free.

Figures 7.5 and 7.6 show the operation of NDMA/CA and NDMA/DS respectively. Both protocols begin with some data being passed to them from another layer and initially they both check to see if they are in an active cell. If they aren’t in an active cell, the transmission is deferred. With NDMA/CA, the next step is to listen to the medium and if it’s free the node may

transmit. Whereas, NDMA/DS must first wait for its dedicated CCA slot, at which point it may sense the medium. Then if the medium is free the node may transmit.

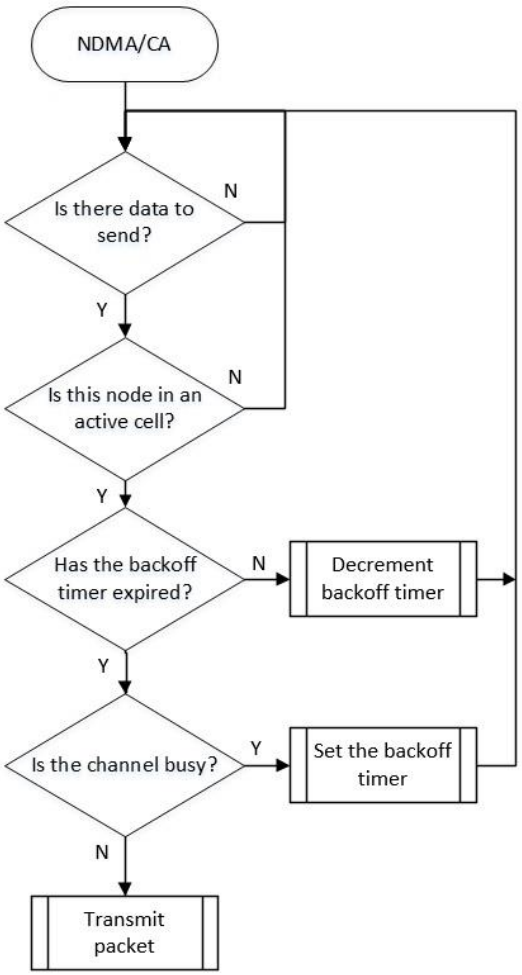


Fig. 7.5. NDMA/CA operational flow chart.

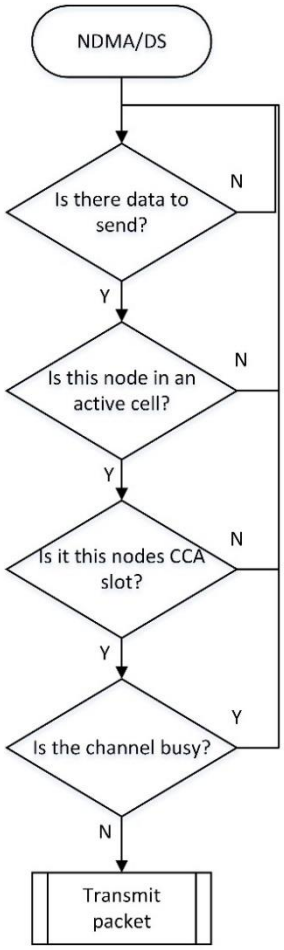


Fig. 7.6. NDMA/DS operational flow chart.

Based on the above literature and designed protocols, further results will be gathered for LASeR with the originally suggested GTDMA MAC, the modified CSMA/CA, CSMA/DS, NDMA/CA and NDMA/DS. Additionally, to give a baseline, a slotted ALOHA (SA) MAC will also be simulated, as will slotted ALOHA with CCA (SA/CA). The SA/CA MAC allows any node with data to sense the medium and then send the packet. If the medium is sensed as busy, then there is no backoff, the node simply tries to transmit in the next slot. For comparison, the flow charts of all the selected MACs are given in *appendix C.*

7.3 Simulation, Modelling and Results

The network will be evaluated in two ways; focusing on the performance of the MAC layers and the performance of the network from the point of view of the routing. In this way, the isolated MAC layers may be analysed as well as their impact on the routing protocol and the performance of the network.

The MAC layer metrics used will be MAC PDR, average queuing delay, collision ratio, throughput and total number of collisions. The routing level metrics used will be PDR, average end-to-end delay, average overhead, throughput and energy consumption.

The MAC PDR is defined as the ratio of successfully received packets to the total number of potential packet receptions. The average queuing delay is the average length of time a between a packet being received and it being forwarded. The collision ratio is the total number of collisions over the total number of possible packet receptions. MAC throughput is defined as the number of bits per second successfully received by a node. Lastly, the total number of collisions is the sum of all the collisions that occurred over the simulation time. The routing metrics are as described in chapter 4.

The results are gathered for two scenarios; the first is a dense scenario, in which *25* nodes are deployed in a *600m* by *600m* area. Each node can transmit at *250kbps* and has a transmission range of *250m*. In order to focus the results on the effect of the MAC and routing layers, the effects of the physical channel are omitted. The traffic levels are varied in this scenario by adjusting the per node packet generation rate to *[0.1, 1, 5, 10]packets per second* (pk/s). This represents network data generation rates of between *2.4pk/s* and *240pk/s*.

The second scenario also uses *25* nodes, but they are deployed in an area of *1500m* by *1500m*, to represent a sparse network. The transmission rate and range are kept constant at *250kbps* and *250m* respectively. The packet generation rate is set to *0.1pk/s*, but the packet data size is varied at *[32, 256, 4096, 65536]bits*.

7.3.1 MAC: Varying Traffic Levels

The results in figures 7.7(a-e) show the performance of the various MAC protocols in the first scenario as the traffic level is varied. Unsurprisingly, the collision free MACs, GTDMA and NDMA/DS achieve 100% PDR, with CSMA/DS also achieving a high level of success, with a minimum PDR of 94.38%, as shown in figure 7.7(a). CSMA/CA and NDMA/CA show a medium level of delivery success of between 71.38% and 83.52%, whereas the two ALOHA based MACs yield the lowest PDR. The inverse of this is shown as collision ratio in figure 7.7(c), with both GTDMA and NDMA/DS causing no collisions. The delay results in figure 7.7(b) for every MAC show a steep decline between *0.1pk/s* and *1pk/s*, this is due to the queue size; with a low packet generation rate, each node can store every packet that's received and it takes time to transmit them all. However, as the packet generation rate increases, each node will receive more packets and the length of the queue will be exceeded, with the older packets being dropped in favour of newer ones. This allows newer packets to be transmitted faster since they don't have to wait for older ones to be serviced first. In LASeR, this loss of older packets is not a problem as the multipath nature of the protocol makes it highly likely that the dropped packets

will be delivered by another node. This will be highlighted by the PDR levels shown in the routing results given later. Overall, NDMA/DS and NDMA/CA give the worst delay performance, since nodes have to wait for their cell to be active and then for the medium to be free. The ALOHA based MACs give the best delay, but this is due to their short service times and their low delivery rates, which means that many packets are dropped and allows the successful packets to be delivered faster. In comparison, GTDMA and CSMA/DS have both very low delay and very high PDR. The throughput results in figure 7.7(d) show GTDMA to consistently have the highest throughput, with CSMA/DS less than 130530.62bits/s behind it. CSMA/CA shows a reasonable level of throughput, however the other four MACs are significantly lower. In the case of SA and SA/CA this is due to low PDR, but in the case of NDMA/CA and NDMA/DS it is due to high delay. The total number of collisions in figure 7.7(e) show that both GTDMA and NDMA/DS are collision free. CSMA/CA has the highest number of collisions in total, at 4404474, however because it transmits a lot of packets its collision ratio remains relatively low. Contrastingly, SA and SA/CA show a lower number of collisions but a higher collision ratio, which implies that they transmitted less packets, but more of them were involved in collisions.

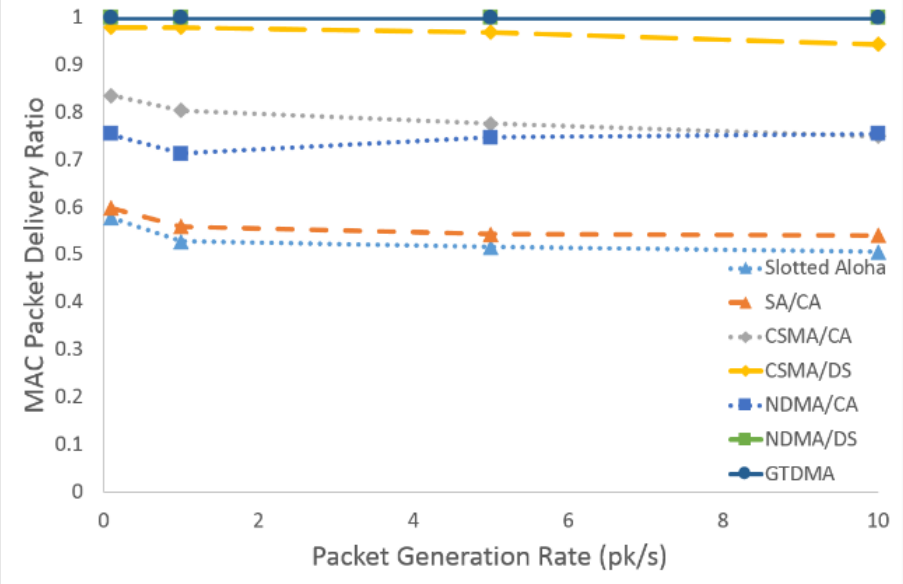


Fig. 7.7(a). MAC layer PDR simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

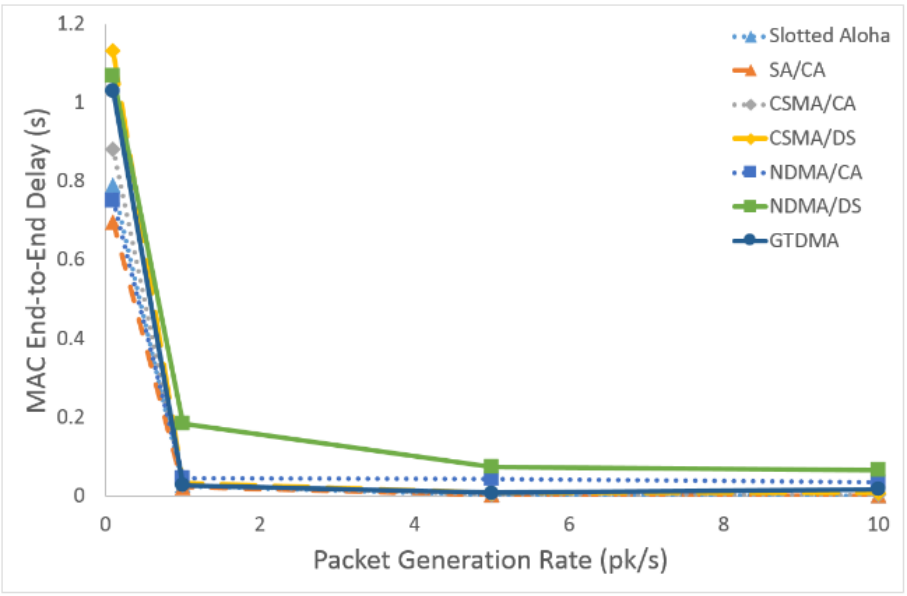


Fig. 7.7(b). MAC layer end-to-end delay simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

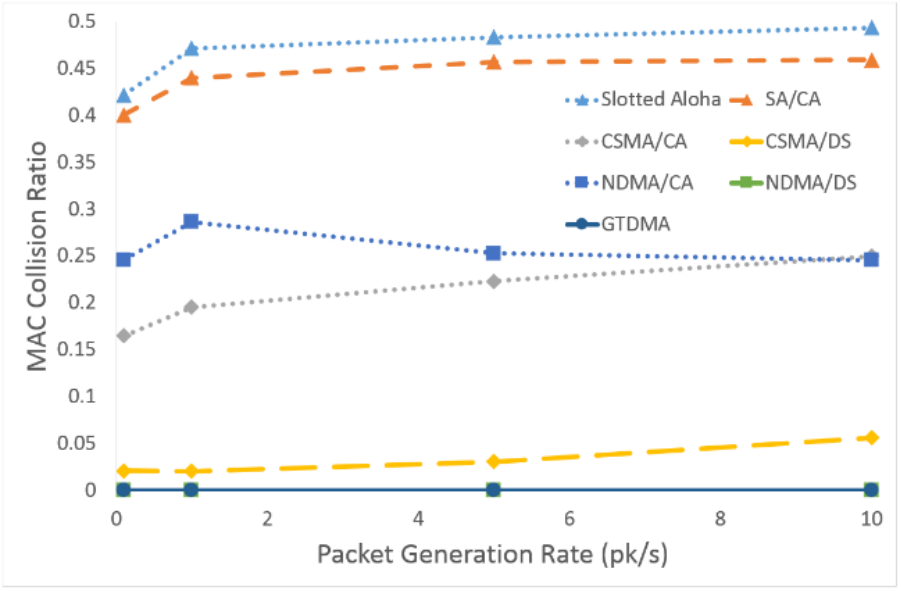


Fig. 7.7(c). MAC layer collision ratio simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

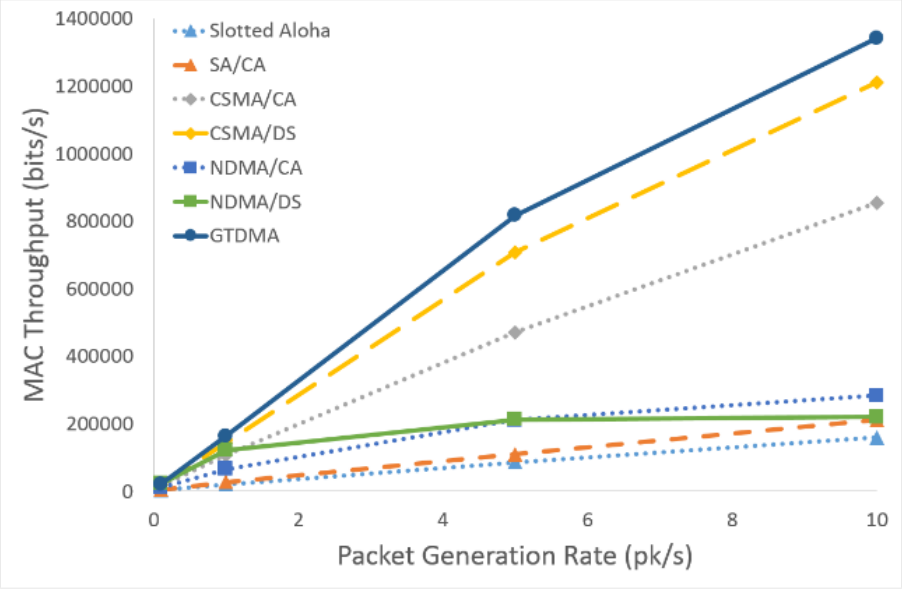


Fig. 7.7(d). MAC layer throughput simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

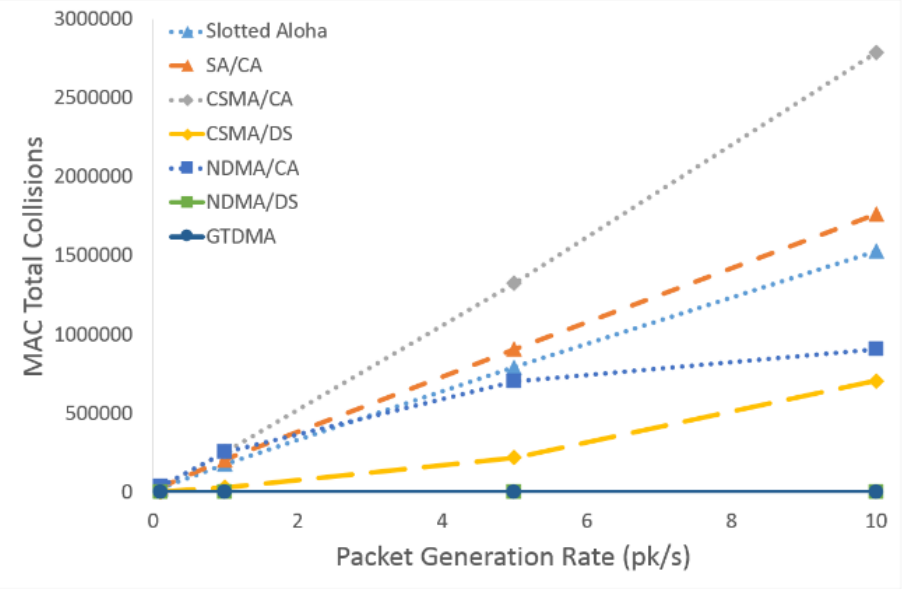


Fig. 7.7(e). MAC layer total collisions simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

7.3.2 Routing: Varying Traffic Levels

The routing results given in figures 7.8(a-e) were collected at the same time as the MAC results in figure 7.7. The routing PDR in figure 7.8(a) shows the GTDMA MAC to have the best performance at over 98.81%, with CSMA/DS also being very high, over 96.12%, which echoes the MAC PDR results. However, NDMA/DS gives worse results, which is mostly due to it having the worst routing delay characteristics causing packets to be dropped. This trend is also true of NDMA/CA. This occurs because the high MAC layer delay is only typical of a single hop and, since LASeR is a multihop routing protocol, this high delay is amplified as packets

experience more hops, which eventually causes packet loss. Contrastingly, GTDMA and CSMA/DS give very low routing delay, as shown in figure 7.8(b), with a maximum delay of 4.56ms. Similarly to the MAC results, SA and SA/CA also give low delay due to their low level of PDR. From figure 7.8(c), it can be seen that LASeRs overhead is generally low for all MAC protocols, with NDMA/DS having the lowest. This is due to its high MAC delay time, which means that it is taking a long time to transmit packets and therefore transmissions occur less often, so less redundancy is created. In terms of throughput, NDMA/CA is the worst, since it has low PDR and high delay, as shown in figure 7.8(d). GTDMA and CSMA/DS have the highest level of throughput, but at the cost of high energy consumption, which is shown in figure 7.8(e). The lower PDR of the other protocols are reflected in their energy consumption, since the dropped packets use up no additional energy. These results highlight the clear trade-off between power consumption and PDR.

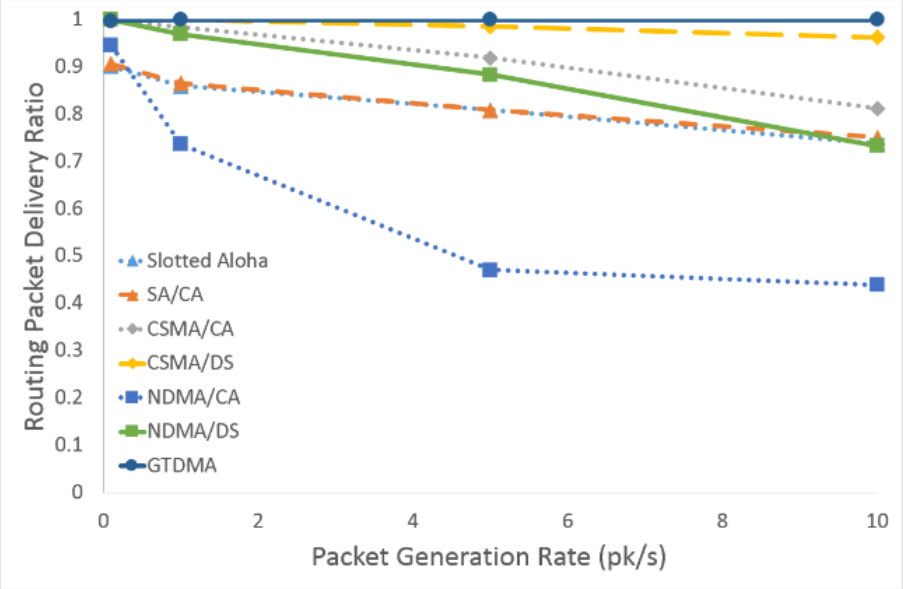


Fig. 7.8(a). Routing layer PDR simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

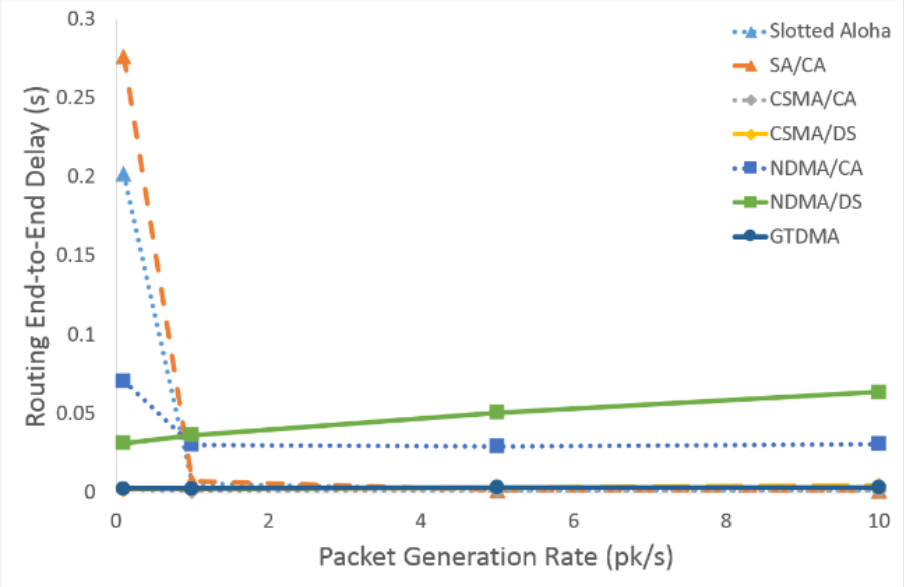


Fig. 7.8(b). Routing layer end-to-end delay simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

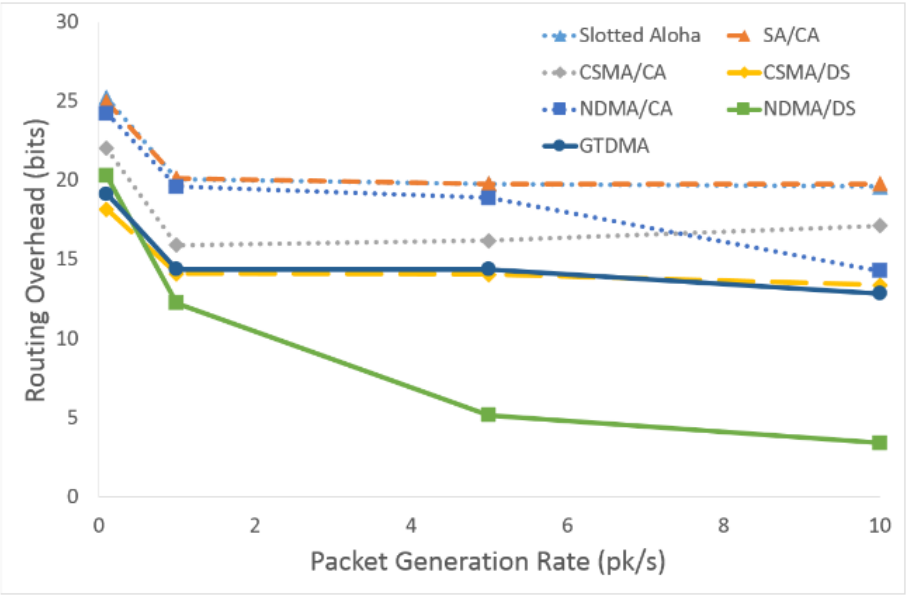


Fig. 7.8(c). Routing layer overhead simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

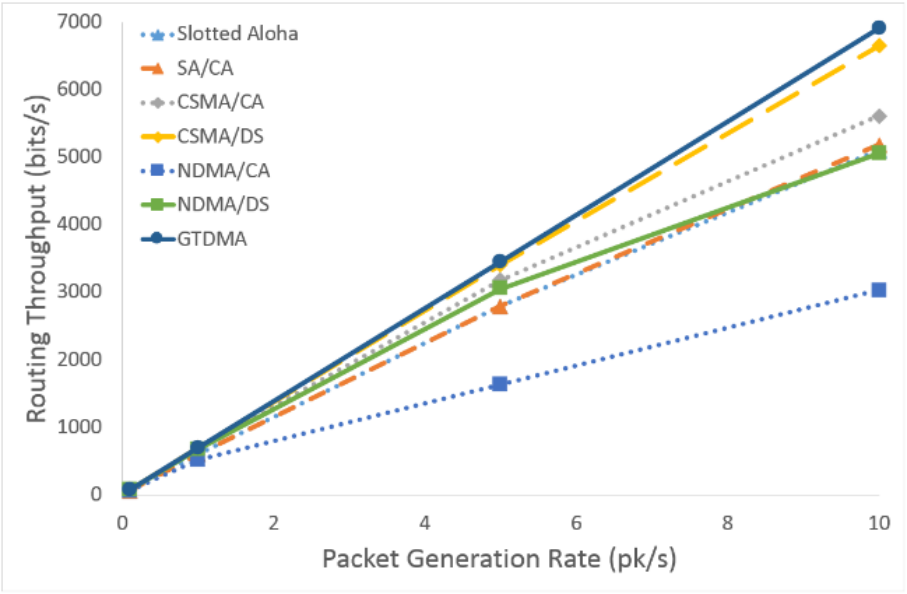


Fig. 7.8(d). Routing layer throughput simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

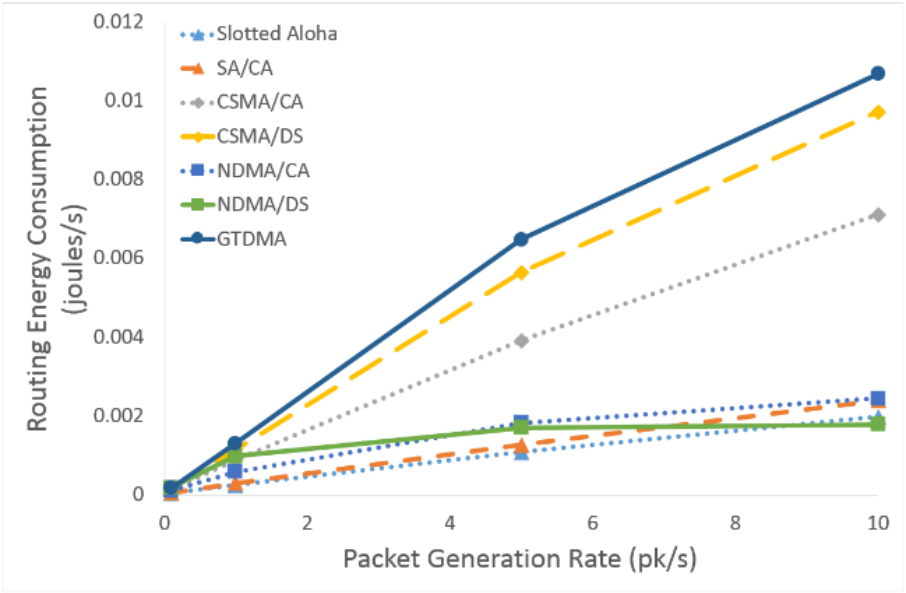


Fig. 7.8(e). Routing layer energy consumption simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying packet generation rates: *[0.1, 1, 5, 10]pk/s*.

7.3.3 MAC: Varying Packet Size

The MAC results for the second scenario are shown in figures 7.9(a-e), which highlights the sparse nature of the scenario meaning that there will be very low amounts of contention. This is backed up by figure 7.9(a), which shows all the MAC layers to have very high PDR. As in the first scenario, the collision free MACs give the best PDR, with CSMA/DS just behind. Again CSMA/CA and NDMA/CA are in the middle, with SA and SA/CA at the bottom, giving the worst performance. The MAC delays in figure 7.9(b) also show a significant increase in comparison to the first scenario, which is due to the additional time needed to transmit the

longer packets. As such, the GTDMA MAC is now one of the worst and the SA and SA/CA MACs still have the lowest delay due to their short service time and low PDR. As with the PDR results, the collision ratio results in figure 7.9(c), follow a similar pattern to those of the first scenario, with the ALOHA based schemes having the highest ratio, followed by CSMA/CA and NDMA/CA. Then the CSMA/DS MAC and finally GTDMA and NDMA/DS having no collisions. This hierarchy is also reflected in the results for the total number of collisions in figure 7.9(e). The unusual changes in direction in these two figures comes from the fact that the longer packets make collisions more likely, but the high number of collisions reduces the number of packets that are forwarded, which in turn reduces the number of collisions. The MAC throughput results in figure 7.9(d) show GTDMA to be the highest, with NDMA/DS close behind and CSMA/DS not far behind that.

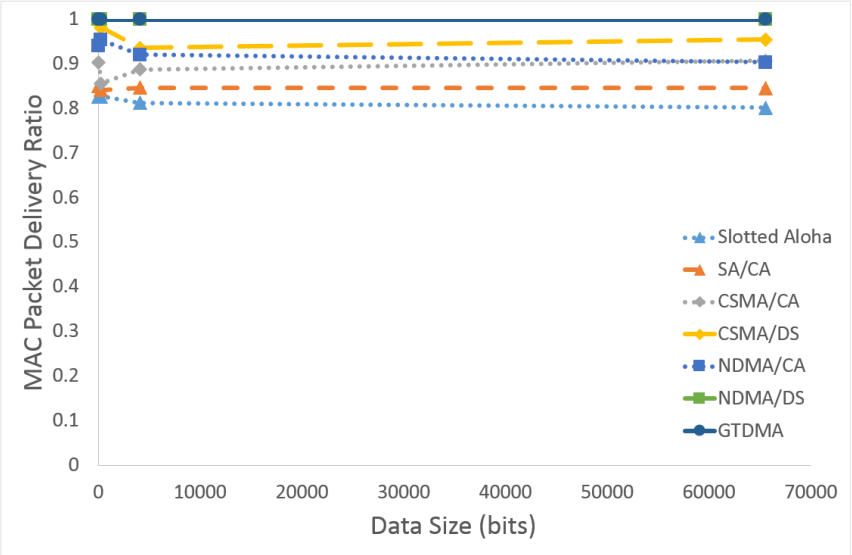


Fig. 7.9(a). MAC layer PDR simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

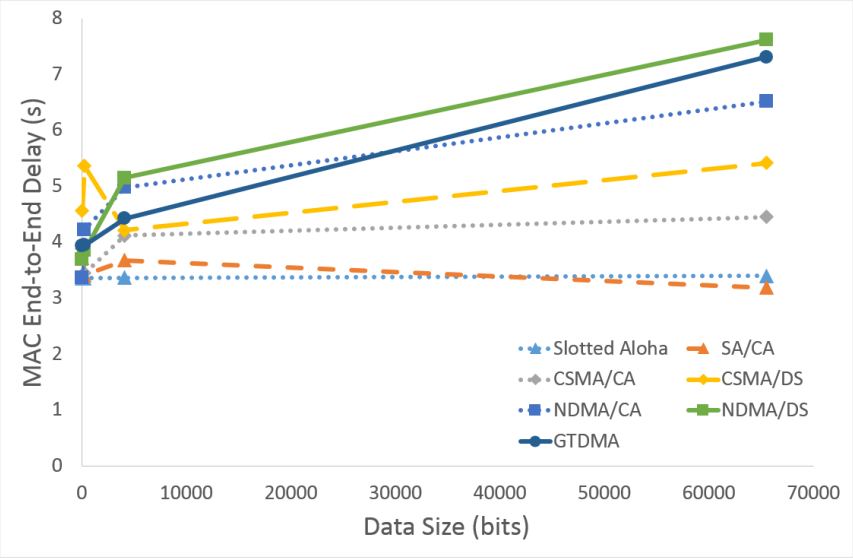


Fig. 7.9(b). MAC layer end-to-end delay simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

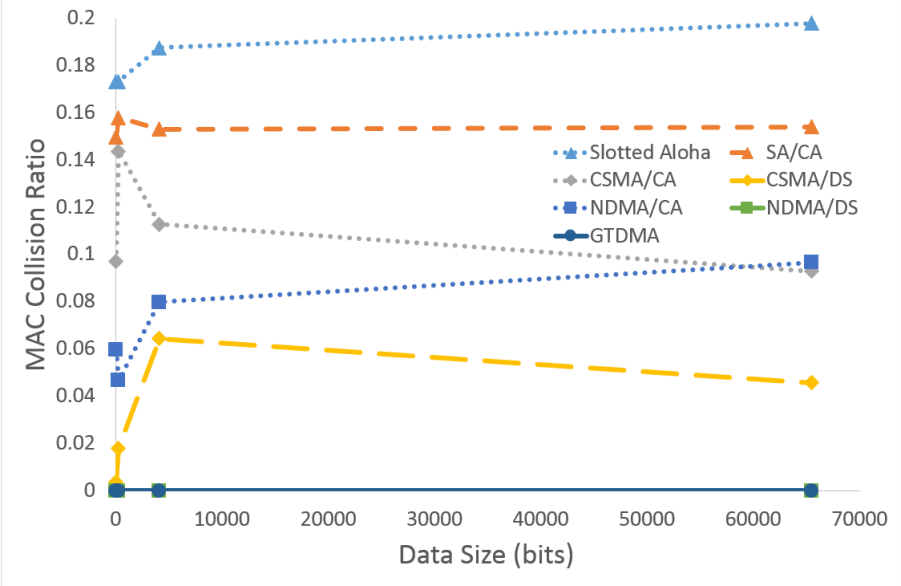


Fig. 7.9(c). MAC layer collision ratio simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

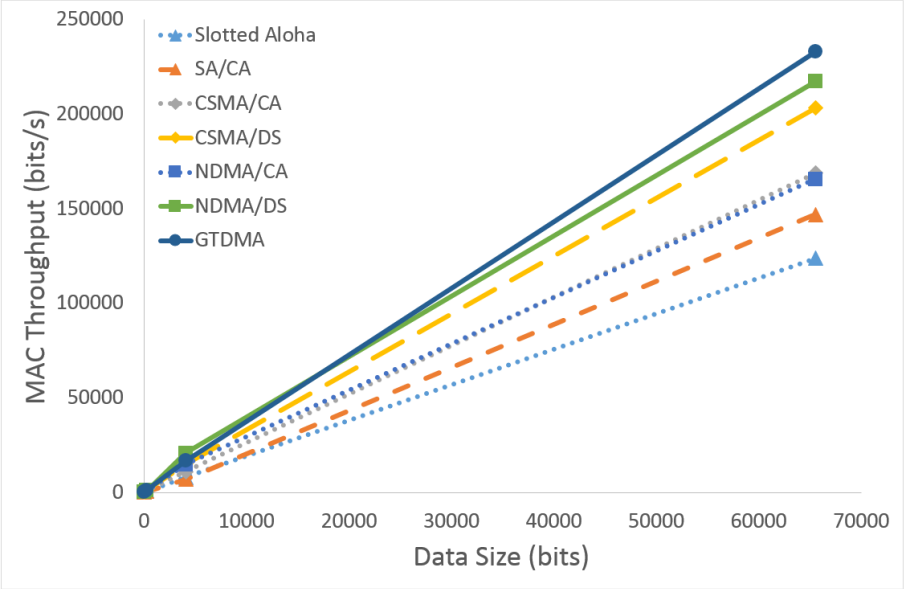


Fig. 7.9(d). MAC layer throughput simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

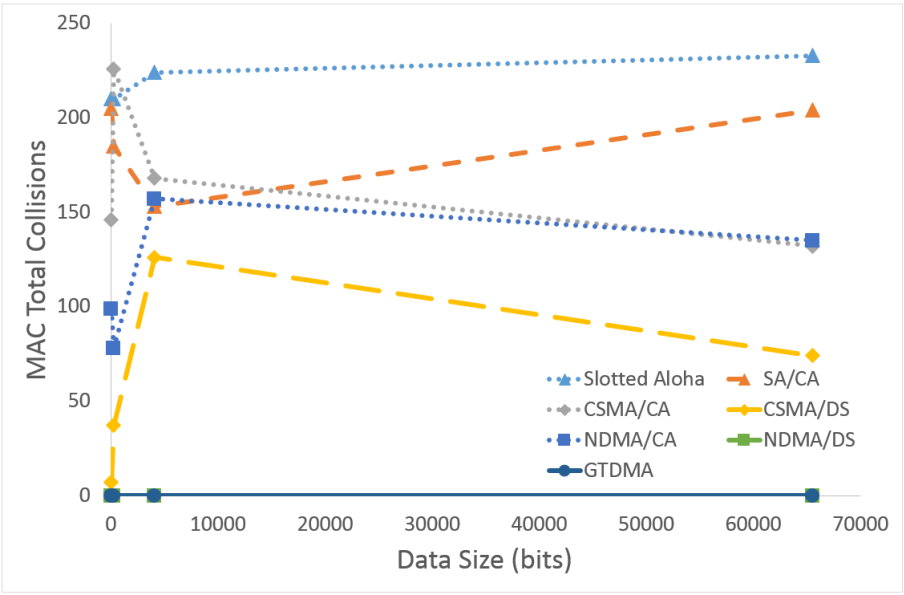


Fig. 7.9(e). MAC layer total collisions simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

7.3.4 Routing: Varying Packet Size

Figures 7.10(a-e) give the routing results for the second scenario, in which figure 7.10(a) show a very low routing PDR for all MACs, which is in direct contrast to the MAC PDR results in the same scenario. This is mostly due to the significant increase in routing delay, which is also a product of the sparsity of the scenario. The NDMA/DS MAC has the best PDR as the packet size increases, which is because the relative size of the CCA time to the packet transmission time is decreased: In very small packet scenarios, the CCA time may be comparable to the time taken to transmit a packet. In such cases it is beneficial to allocate time

slots, such that significant time isn't wasted checking the medium, which may return false or yield a false positive in the event of a hidden node and adding further delay. However, as the packet size increases, the CCA time becomes insignificant in comparison to the transmission time of a packet. In this case, the duration a node must wait in a time slotted scheme is far greater than that of a node which checks the medium and then transmits. Given this improved delay characteristic and high MAC PDR, NDMA/DS gives the best routing PDR result. However figure 7.10(b) shows that it has the second worst routing delay, which is due to the fact that the other MACs have lower PDR and therefore experience less congestion. The best MAC in terms of routing PDR at low packet sizes in this scenario is CSMA/DS, with a 52.29% PDR with 32bit data size. Second to both CSMA/DS and NDMA/DS in PDR is GTDMA, however it has the worst delay, reaching 10.49s, due to the large packet sizes dramatically increasing the length of a time slot. Contrastingly, CSMA/DS has the lowest delay of all the MACs, which is also due to the relative length of a packet transmission to the CCA time. The overhead results in figure 7.10(c) are very low for all the MACs, with NDMA/DS being the lowest. However, they show a sharp drop and then a general decrease in overhead as the packet size is increased. This is because of the ratio in a packet between the data and the control information; the fraction of the packet that is considered overhead decreases as the amount of data contained within the packet increases. The throughput results in figure 7.10(d) show NDMA/DS and CSMA/DS to be the highest, however this is at the expense of energy consumption as shown in figure 7.10(e).

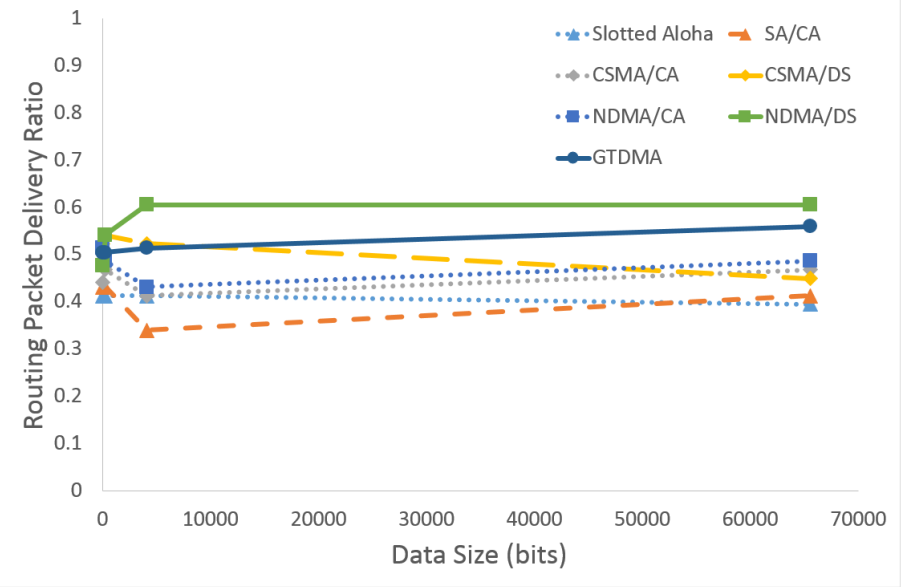


Fig. 7.10(a). Routing layer PDR simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

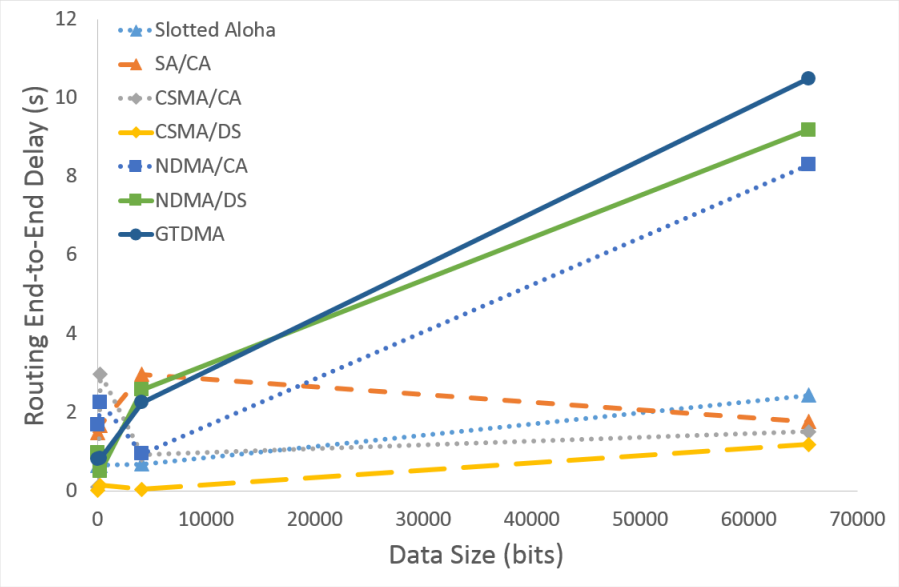


Fig. 7.10(b). Routing layer end-to-end delay simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

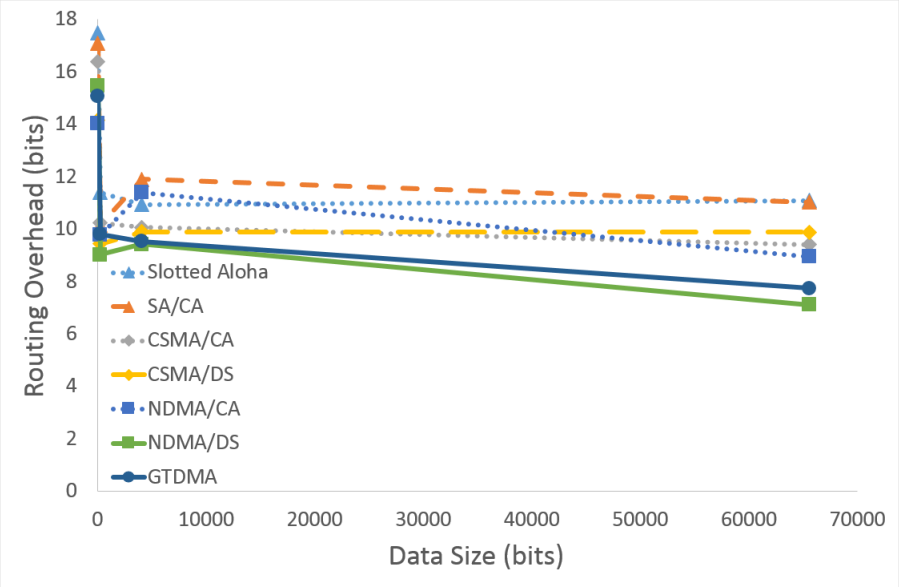


Fig. 7.10(c). Routing layer overhead simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

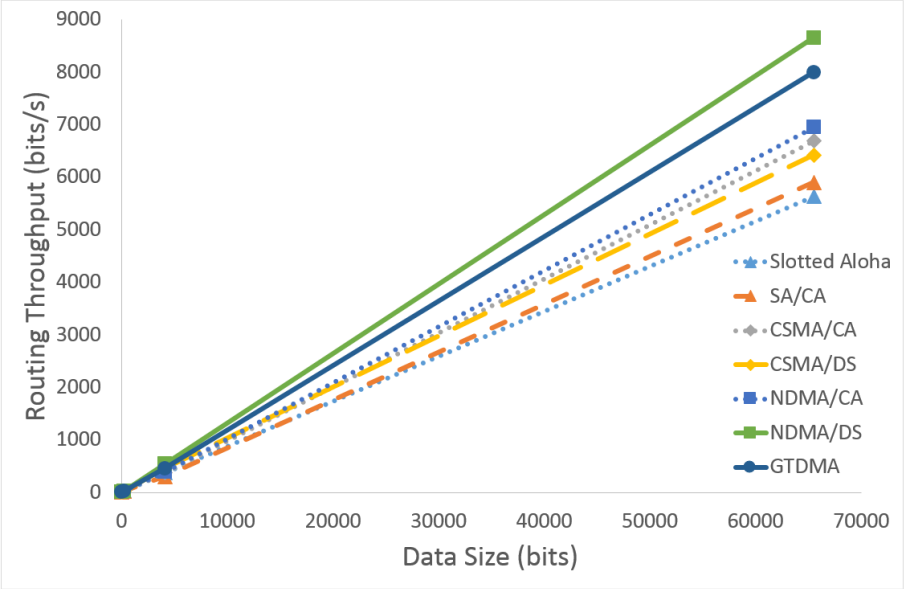


Fig. 7.10(d). Routing layer throughput simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

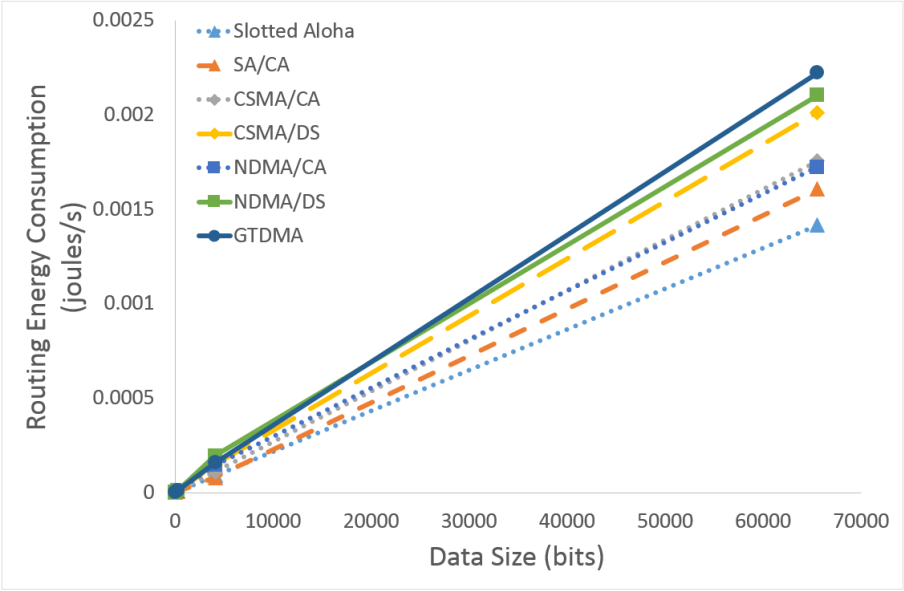


Fig. 7.10(e). Routing layer energy consumption simulation results for Slotted Aloha, SA/CA, CSMA/CA, CSMA/DS, NDMA/CA, NDMA/DS and GTDMA, over varying data sizes: *[32, 256, 4096, 65536]bits*.

7.3.5 Conclusion

Overall, these results show how that the GTDMA MAC gives the best performance in high density scenarios with low packet sizes over all traffic levels. However, as the packet size increases, the time slot length also increases and as such the delay incurred by a GTDMA MAC

grows dramatically, which subsequently affects the PDR. For this reason, in scenarios with high packet sizes, NDMA/DS gives better results, due to the ratio of packet transmission time to CCA time and the advantage of spatial reuse allowing multiple simulations to occur simultaneously. GTDMA and NDMA/DS are both collision free, which is beneficial in dense networks and scenarios with large packet sizes, where collisions are likely. However, in scenarios with low densities and low packet sizes, the CSMA/DS protocol gives the best performance. In these situations collisions are less likely, so transmissions are likely to be successful and allow the MAC to perform multiple simultaneous transmissions that are not delayed by long service times.

7.4 Conclusion

This chapter evaluated a large variety of MAC layer protocols through simulation, with the aim of determining their suitability for use with LASeR. The majority of MWSN MACs are designed to be low duty cycle or hierarchical. As LASeR is a flat routing protocol, the hierarchical MACs were considered unsuitable. Also, since a low-end-to-end delay is one of the main aims of LASeR, the low-duty cycle MACs were also considered to be unsuitable. Additionally, LASeRs use of blind forwarding makes many MAC layers unusable since ACKs and other handshakes are likely to cause a high number of collisions, so the 802.11 DCF MAC is also unusable without modification. MACs like CDMA and FDMA have been rejected because they require more resources than a traditional sensor mote is equipped with.

Additionally, three MACs were proposed as suitable alternatives. These three MACs were based on two ideas; dedicated CCA slots and spatial reuse. The dedicated CCA slots allowed the mitigation of local collisions and can be used in conjunction with CSMA to create CSMA/DS. The second idea uses the location information provided by LASeR to split the network into cells and allows nodes in non-interfering cells to transmit simultaneously. This can then be used with a normal collision avoidance mechanism or the dedicated CCA slots, to create NDMA/CA and NDMA/DS respectively.

Subsequently, seven MACs were selected for simulation, including the originally suggested GTDMA, as well as SA and SA/CA to provide a baseline measurement. A reduced CSMA/CA will also be tested along with the proposed CSMA/DS, NDMA/CA and NDMA/DS. Two scenarios were simulated with data gathered for ten metrics; MAC PDR, MAC delay, collision ratio, MAC throughput, total number of collisions, routing PDR, routing delay, overhead, routing throughput and energy consumption.

The results suggest that in high density scenarios with low packet sizes, the originally selected GTDMA MAC is preferable over all traffic levels. However, in the case of large packet

sizes, the preference shifts in favour of the collision free NDMA/DS MAC. Although, in low density environments with low packet sizes, collisions are less likely to occur and as such the CSMA/DS MAC becomes optimal.

Given these results a dynamic switching MAC may be preferable, which can be set predeployment and then the protocol used can be changed as the scenario parameters change. As such, each node should be programmed with the three MACs, GTDMA, NDMA/DS and CSMA/DS, and upon a network wide command issued from a central controller the network can switch to the desired MAC.

The next chapter will detail further additions to RASeR, which will enable it to be optimised for specific applications.

CHAPTER 8

RASER'S ADDITIONAL MODES OF OPERATION

8.1 INTRODUCTION

Of the three protocols presented, RASeR is the most adaptable, since it makes no assumptions about packet generation or the availability and accuracy of location information. However, many MWSN applications will require additional features. For this reason, four optional modes have been added to RASeR in order to fulfil the needs of the most advanced emerging mobile sensor network applications. These enhancements are described and simulated evaluation is performed and presented. Furthermore, this chapter concludes with a case study, evaluating the suitability of RASeR to the application of UAV aided search and rescue. This case study highlights the advantages of having these modes in a real world scenario, which is the primary motivation for this chapter.

8.2 SUPERSEDE MODE

In some applications it may be advantageous for the sink to receive the latest data rather than every acquired sample. This would be the situation if newly generated data would cause the previous samples to become redundant. For example in the application of UAV aided SAR, where the target is out at sea and being carried by the current. In this case it's more important to report the last known location of the target than its previous whereabouts since the old location is redundant. In this type of situation RASeR can be set to work in *supersede mode*, in which only the newest packets from each node are kept in the queue. In this way, packets that are considered out-of-date are dropped, which allows the newer packets to traverse the network faster. A node will consider a packet to be out-of-date if it receives another packet from the same source node that has a more recent packet ID, so the new packet will then replace the old one. This will occur when a newer packet catches up to an older one, which happens mainly when the network is congested. For example, if a packet reaches a congested node and is held in the queue for a long time, the same source node may generate another packet. If the second packet reaches the congested node, who has yet to send the first packet, then the first packet will be dropped in favour of the second. Another situation where this may occur is through the mobility of the nodes. In this case a node close to the sink may have received a packet from a

node that is further from the sink through multiple hops. However, before it can forward the packet the distant node moves closer, to within a single hop, at which point it generates another packet. In this case the forwarding node will drop the older packet from its queue, since it has now received a newer one.

In practical terms, this limits the amount of memory required since the queue only needs a length of *n*. The results given were performed using the same parameters as described in chapter 6 and the analytical results are from the derivations in chapter 5. Figures 8.1(a-e) give the results for varying mobility from static to a maximum speed of *25m/s*. Over this variation there is not much difference between the two modes, which is because RASeR copes very well in mobile scenarios regardless of the mode used.

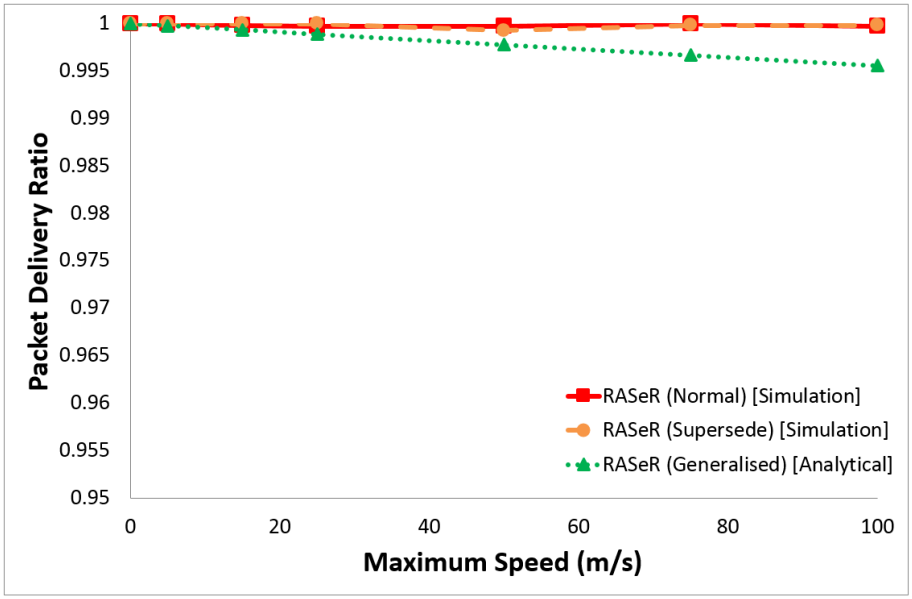


Fig. 8.1(a). Simulation and analytical PDR results for RASeR in both normal and supersede mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

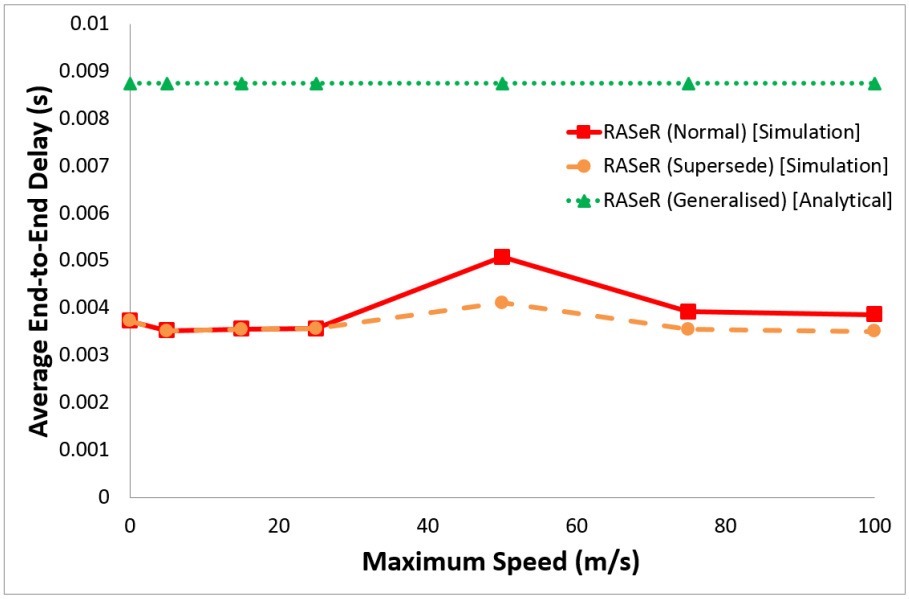


Fig. 8.1(b). Simulation and analytical end-to-end delay results for RASeR in both normal and supersede mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

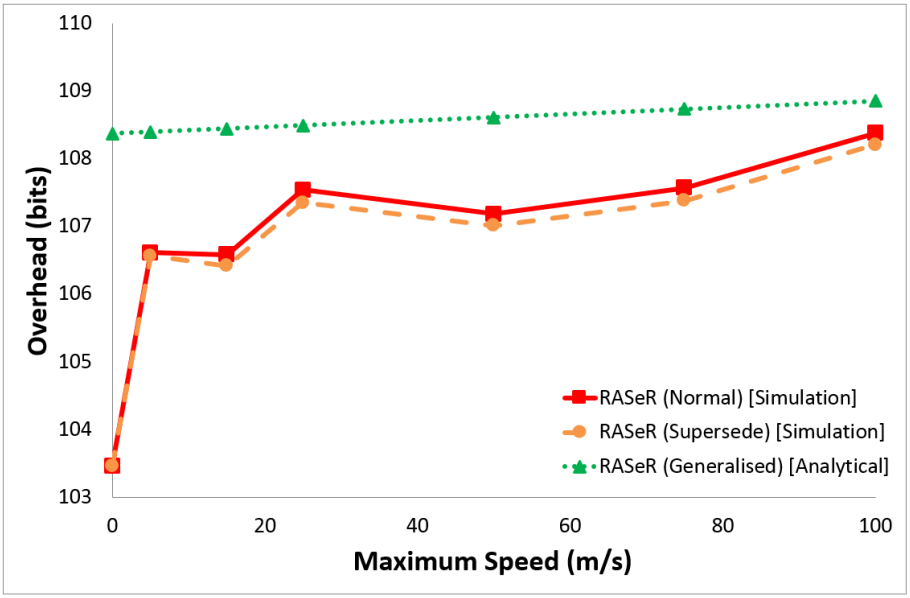


Fig. 8.1(c). Simulation and analytical overhead results for RASeR in both normal and supersede mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

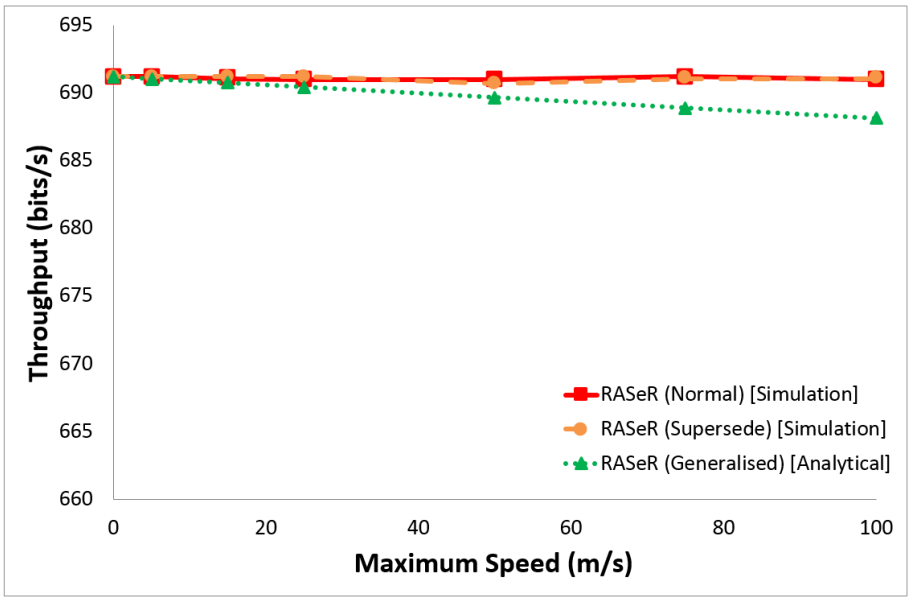


Fig. 8.1(d). Simulation and analytical throughput results for RASeR in both normal and supersede mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

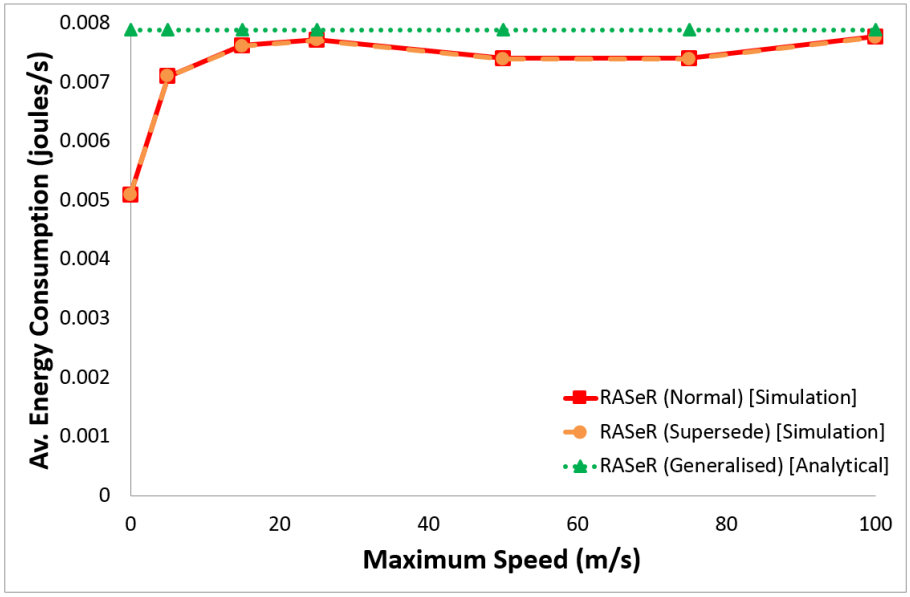


Fig. 8.1(e). Simulation and analytical energy consumption results for RASeR in both normal and supersede mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

The results for varying numbers of nodes are shown in figures 8.2(a-e), and in terms of overhead, throughput and energy consumption, the two modes are almost identical. However, the differences between the modes are clearly highlighted when looking at the PDR and end-to-end delay results. In situations with a high number of nodes normal mode tries to deliver every

packet and as such has a higher PDR. Whereas, supersede mode only attempts to deliver the newest packets, which keeps the delay low. Additionally, the dropped packets in supersede mode should only be those that have been deemed out-of-date; in that they have been superseded by a more recent packet.

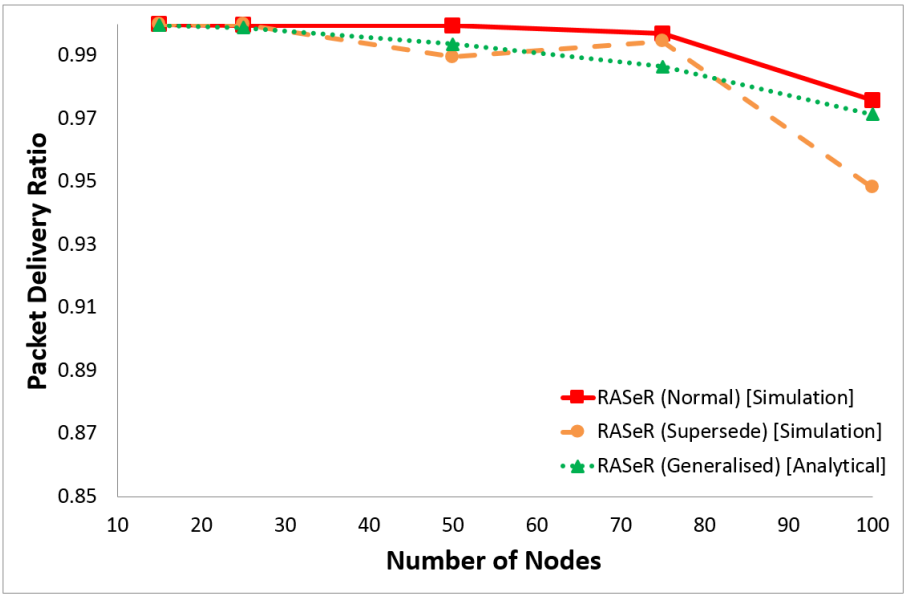


Fig. 8.2(a). Simulation and analytical PDR results for RASeR in both normal and supersede mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

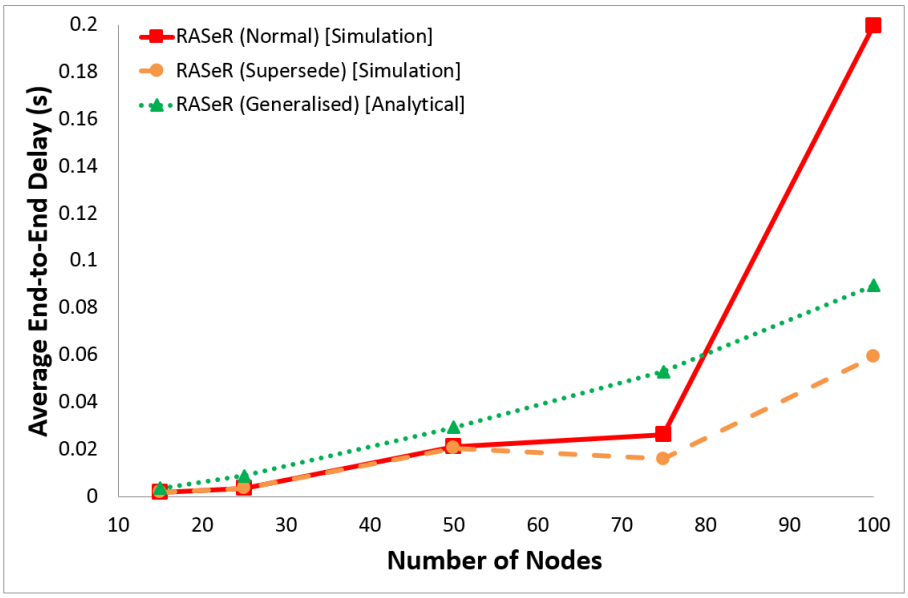


Fig. 8.2(b). Simulation and analytical end-to-end delay results for RASeR in both normal and supersede mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

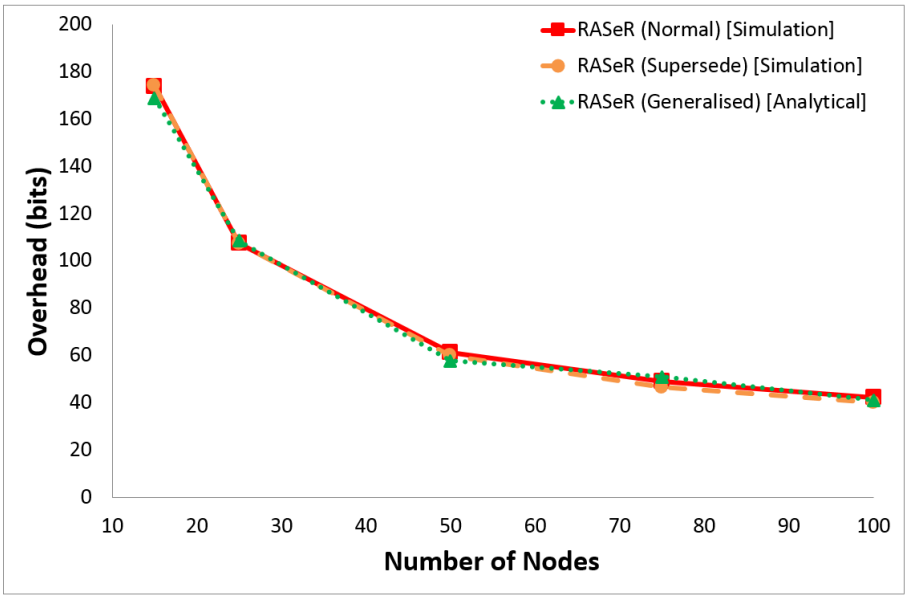


Fig. 8.2(c). Simulation and analytical overhead results for RASeR in both normal and supersede mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

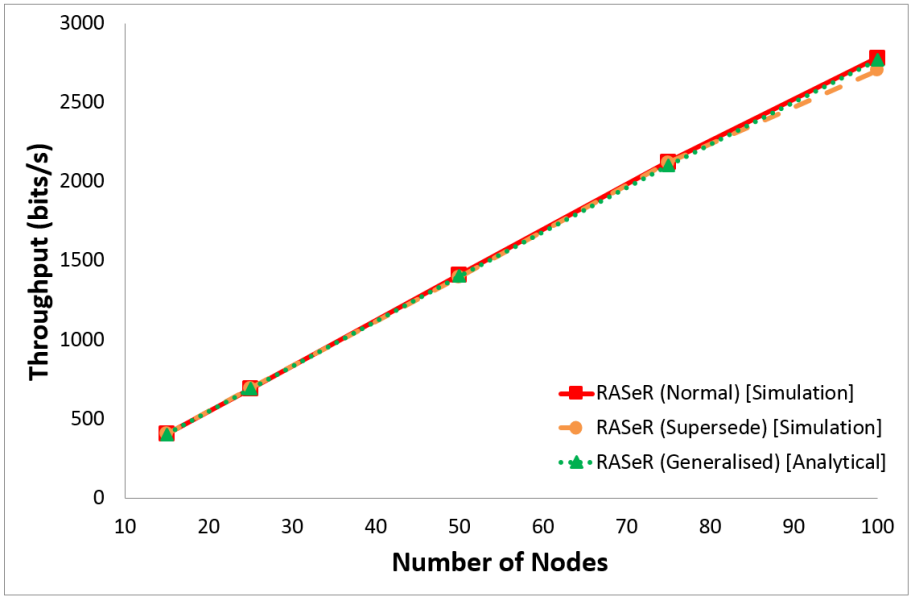


Fig. 8.2(d). Simulation and analytical throughput results for RASeR in both normal and supersede mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

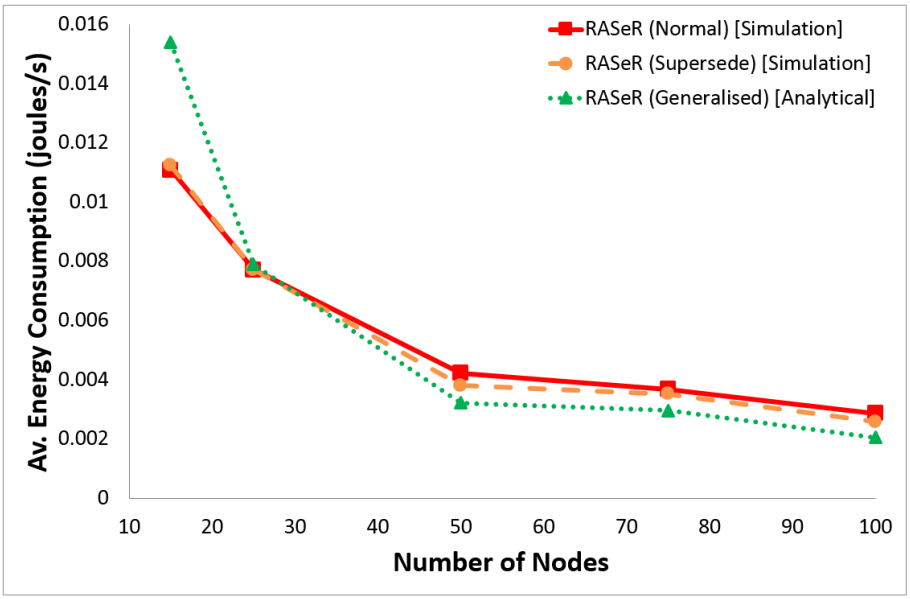


Fig. 8.2(e). Simulation and analytical energy consumption results for RASeR in both normal and supersede mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

There are similar characteristics in the results in figures 8.3(a-e), in which the traffic levels are varied. Again, the overhead in figure 8.3(c), throughput in figure 8.3(d) and energy results in figure 8.3(e) are identical, but the differences are shown in the PDR in figure 8.3(a) and delay in figure 8.3(b) results. At high traffic levels, supersede mode shows an obvious superiority over normal mode, in that it has both a higher PDR and a lower average delay. This occurs because of the volume of traffic used in these simulations. Since normal mode tries to deliver every packet, the queue fills up with every node trying to service each packet in a first-come-first-served manner. This means that the oldest packets are given priority and newer packets maybe lost to queue overflow. Contrastingly, in supersede mode, priority is given to the newest packets. This means that nodes are not having to wait for the old packets to be serviced before they can transmit the new data. This keeps the delay low, but also reduces the congestion that causes packets to be lost in normal mode, which allows the natural robustness of RASeRs design to maintain a high delivery ratio.

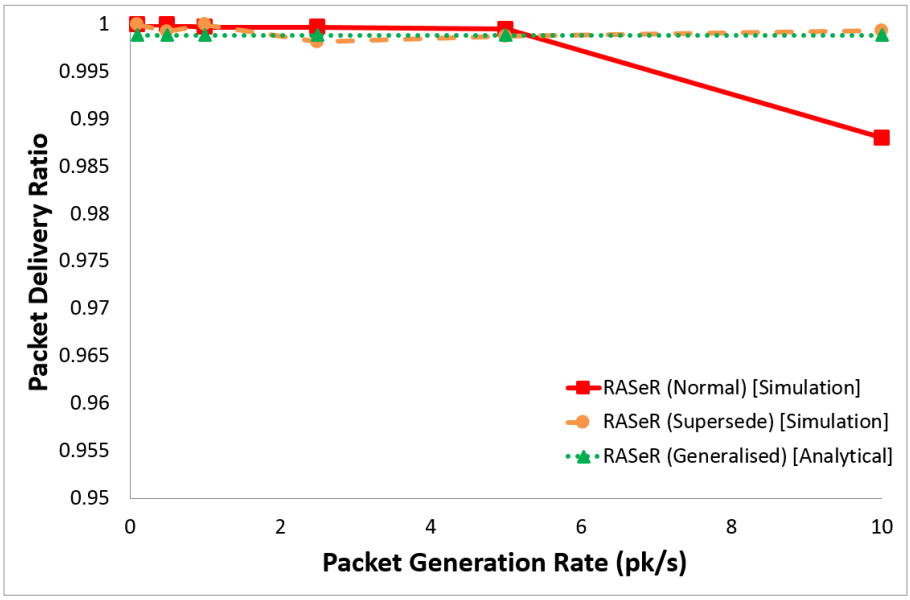


Fig. 8.3(a). Simulation and analytical PDR results for RASeR in both normal and supersede mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

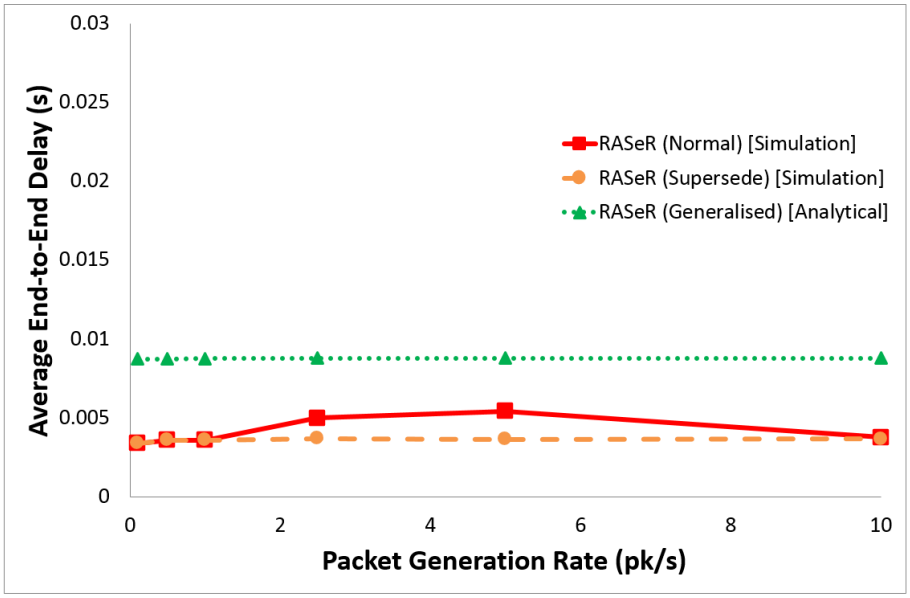


Fig. 8.3(b). Simulation and analytical end-to-end delay results for RASeR in both normal and supersede mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

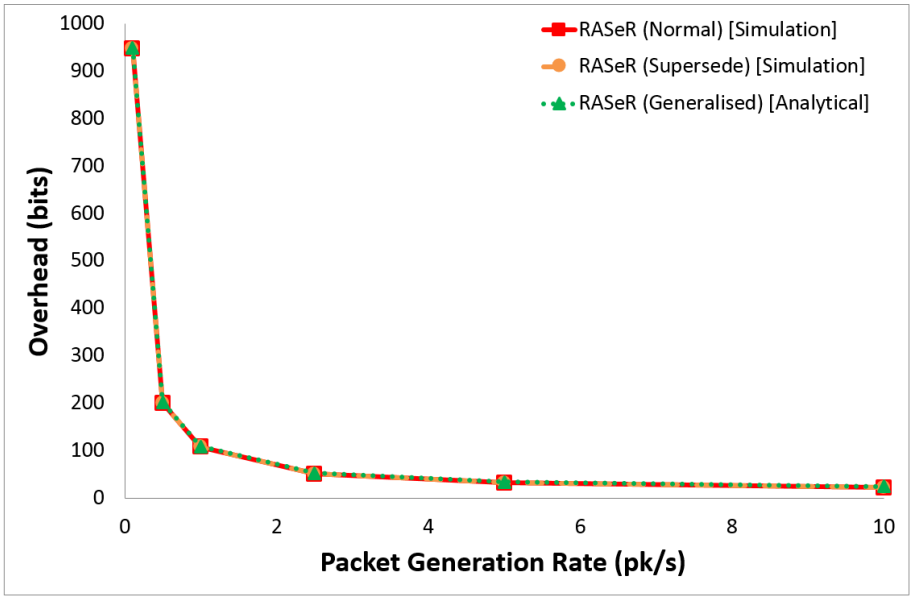


Fig. 8.3(c). Simulation and analytical overhead results for RASeR in both normal and supersede mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

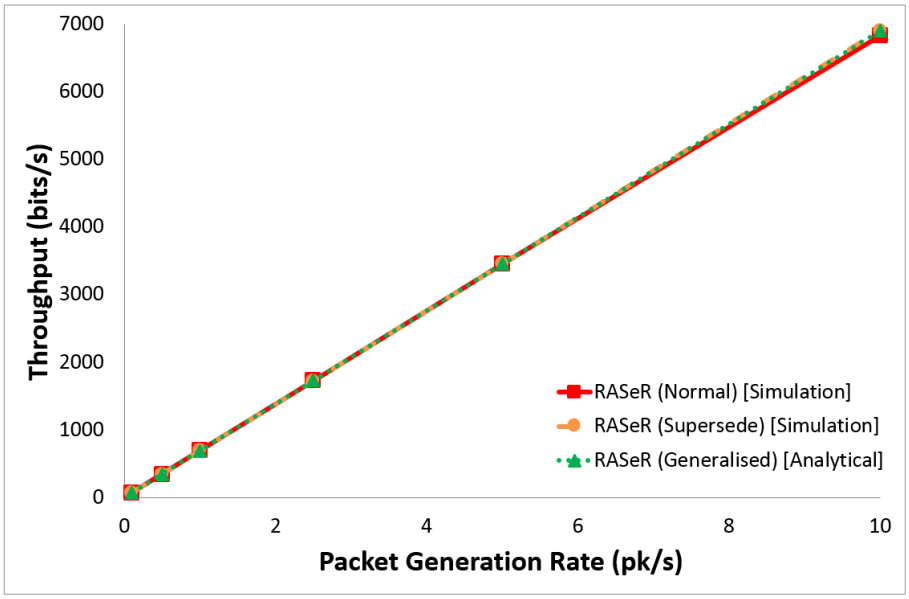


Fig. 8.3(d). Simulation and analytical throughput results for RASeR in both normal and supersede mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

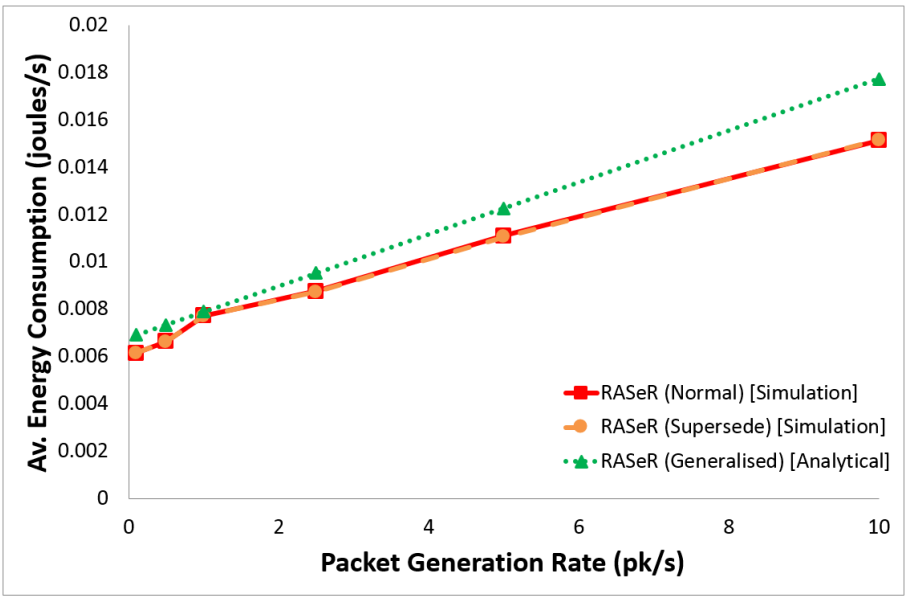


Fig. 8.3(e). Simulation and analytical energy consumption results for RASeR in both normal and supersede mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

In general, these results show that supersede mode is preferable in high traffic scenarios as well as applications in which low latency is critical or receiving the latest data rather than every sample is preferable. However, normal mode would still be a better choice in applications where reliability of data delivery is the main requirement.

8.3 Reverse Flooding

Most sensor network applications only require data to be taken from the sensor nodes to a sink, however in many applications, it is sometimes necessary to disseminate a message in the opposite direction. For this reason, RASeR has a built in reverse flooding mechanism, which can be used to transmit a message from the sink to every node in the network. The technique simply requires each node to retransmit the sinks message once. The packets from the sink are the same as shown in figure 5.9, but they always have a node ID of zero and a hop count of zero. In the implementation, these sink packets are prioritised above any sensor data in order to keep the delay time low, which may be important for applications that require time-critical mission adjustments. Another motivation for the implementation of a reverse flooding protocol is that it can take advantage of a hierarchical based synchronisation protocol such as TPSN (Timing-Sync Protocol for Sensor Networks) [3.26], which would be useful for the strict timing requirements of a GTDMA MAC layer.

In the mathematical analysis, the expression for packet arrival rate to each node, (43), will also need to account for sink packets:

$$\lambda = \frac{f_p \cdot (n-1)}{N_n^2} + f_{p_{sink}} \qquad (50)$$

where f_{psink} is the packet generation rate of the sink. The B_{tx} equation has also been adjusted, as described in the next section.

The results in figures 8.4(a-e) are based on the same parameters as described in chapter 6, however the sink is now also producing data. So, each of the 24 sensor nodes are still generating packets at a rate of *1pk/s*, but the sink node is also generating packets at *[0, 0.05, 0.1, 0.2, 0.5]pk/s*. All nodes are still mobile, with a maximum speed of *25m/s*. To evaluate the effects of the added sink packets, all the metrics will be gathered as before. So the sensor packets are those produced by the sensors, and are analysed in terms of their PDR, delay, and throughput. The overhead results for the sensor packets consider the sink packets to be additional overhead. The energy results are calculated in the same way as before, so they are given as the average per node, which includes the sink packets as well as the sensor packets. The sink packets PDR, is measured as the sum of the unique sink packets received by each node, over the total number of sink packets multiplied by the total number of sensor nodes. As such, perfect PDR would be achieved when every sensor node receives every sink packet.

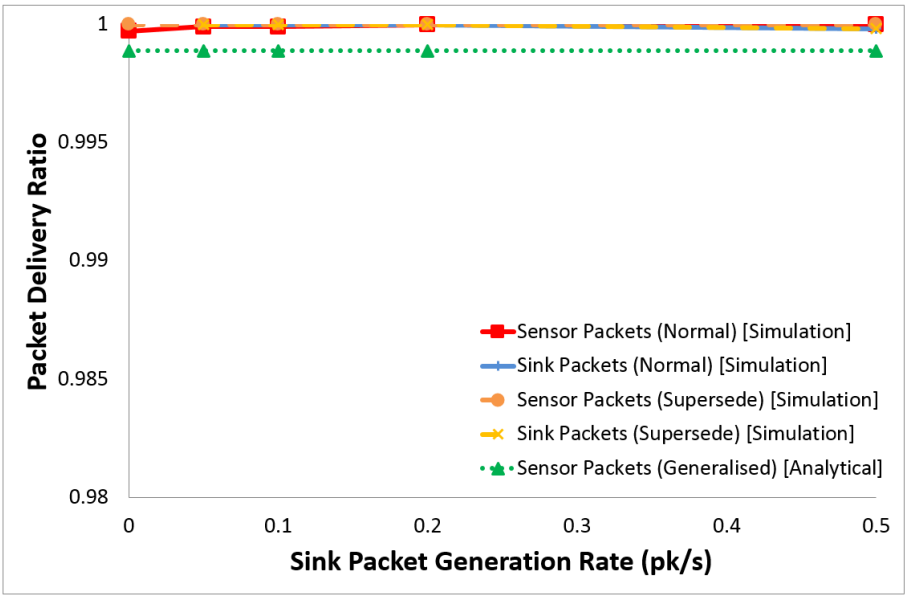


Fig. 8.4(a). Simulation and analytical PDR results for RASeR in both normal and supersede mode, over varying sink packet generation rates: *[0, 0.05, 0.1, 0.2, 0.5]pk/s*.

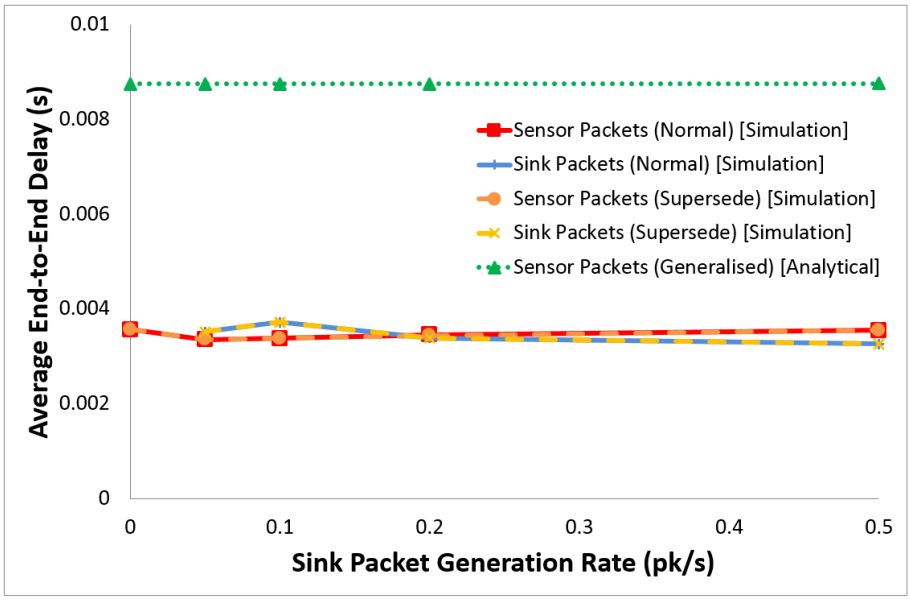


Fig. 8.4(b). Simulation and analytical end-to-end delay results for RASeR in both normal and supersede mode, over varying sink packet generation rates: *[0, 0.05, 0.1, 0.2, 0.5]pk/s*.

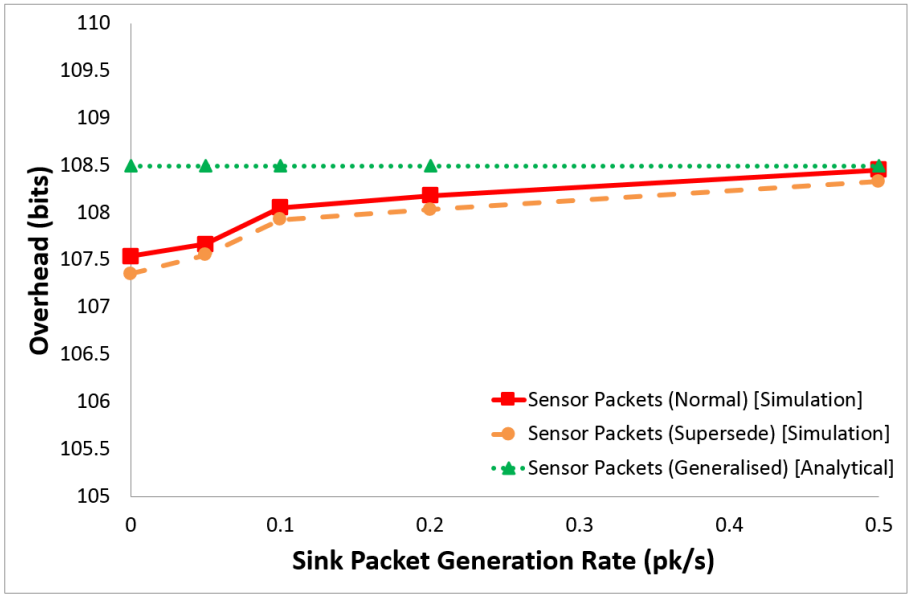


Fig. 8.4(c). Simulation and analytical overhead results for RASeR in both normal and supersede mode, over varying sink packet generation rates: *[0, 0.05, 0.1, 0.2, 0.5]pk/s*.

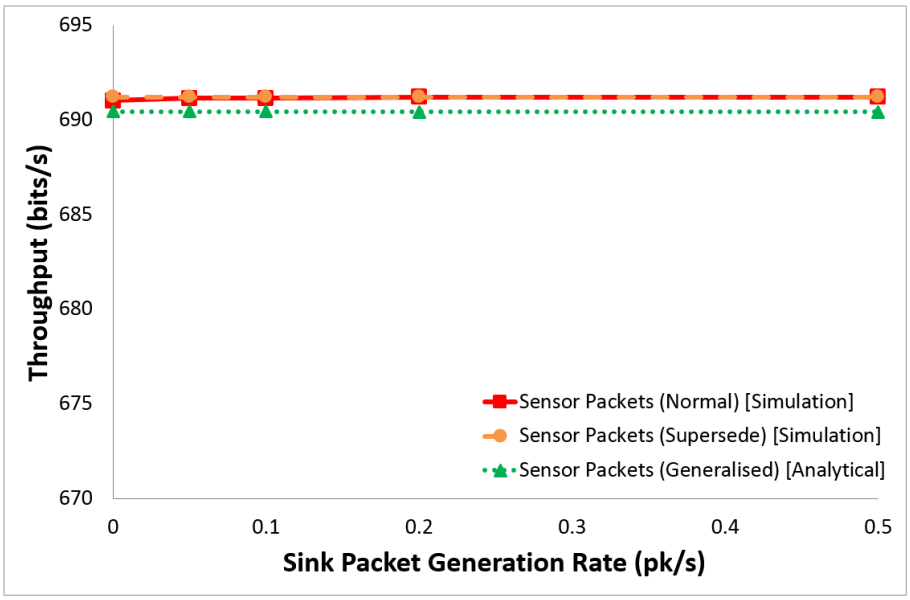


Fig. 8.4(d). Simulation and analytical throughput results for RASeR in both normal and supersede mode, over varying sink packet generation rates: *[0, 0.05, 0.1, 0.2, 0.5]pk/s*.

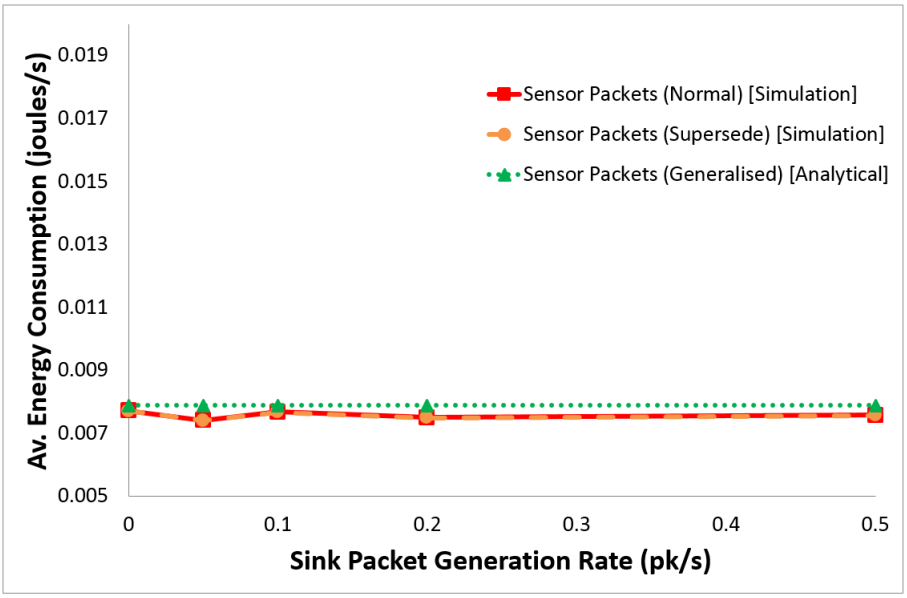


Fig. 8.4(e). Simulation and analytical energy consumption results for RASeR in both normal and supersede mode, over varying sink packet generation rates: *[0, 0.05, 0.1, 0.2, 0.5]pk/s*.

The PDR results in figure 8.4(a) show that both the sink packets and the sensor packets achieve near perfect packet delivery over the whole range, with the lowest value being 99.72%. The delay results in figure 8.4(b) indicate that the delay time for both the sensor packets and the sink packets are low and remain relatively constant over the range, with a maximum sink packet delivery time of 3.72ms and a maximum sensor packet delivery delay of 3.57ms. The overhead results in figure 8.4(c) show the expected increase, which comes from the fact that the sink packets are simply counted as additional overhead. So whilst the sensor packets require no additional transmission to be delivered, the extra sink packets add further load to the sensors,

which is clearly shown in this figure. Similarly to the PDR and delay results, the throughput and energy results remain fairly constant, as shown in figures 8.4(d) and 8.4(e) respectively.

These results indicate that the reverse flooding mechanism is not only fast and highly reliable, but it also has minimal effects on the routing of data from the sensors.

8.4 Energy Saving

One key challenge in most sensor networks is that of energy consumption, since nodes are generally battery powered, they have a finite amount of energy. Even considering the assumptions made previously about nodes having sufficient power to complete a mission, by saving energy the possible network lifetime can be extended. In RASeR it is possible to schedule sleep cycles, such that no nodes will transmit during this time. A *round* is made up of a number of active cycles followed by a number of sleep cycles, both of which can be defined individually, such that any fraction of sleep time to active time can be realised. The network should be programmed as a whole with the sleep slot requirements, prior to deployment. In general, much of RASeRs energy consumption comes from the broadcasting of beacon packets in order to cope with a highly dynamic topology. However, if the topology is not moving so fast, then the number of beacon packets needed to maintain the gradient field is reduced and sleep cycles can be used to save energy without any effect on the protocols performance. It should also be noted that energy can also be saved by extending the length of each timeslot; since only a single node will make a single transmission during one timeslot, the remainder of the slot duration can be spent sleeping. As such, longer timeslots can extend the amount of time a node may spend sleeping. Additionally, with the increasing transmission rates of modern radios, it's possible that implementations of RASeR will be able to keep up with a highly dynamic topology, even when sleep cycles or extended timeslots are used. Essentially, in implementation, a node will follow the predetermined schedule of sleep cycles and active cycles.

In some cases it may also be advantageous to shut down other parts of the node, such as the sensors or mobility, during these cycles as the applications will allow, in order to further reduce the energy consumption.

The increased delay caused by including sleep slots is characterised in the mathematical analysis by adjusting RASeRs service time (48) to be

$$T_s = \frac{\Delta \cdot n(S_A + S_S)}{S_A} \qquad (51)$$

where S_A is the number of active slots in the round, S_S is the number of sleep slots in the round and Δ is the length of a single time slot.

Also, the use of reverse flooding and energy saving means that the expression (49) for RASeRs B_{tx} should now be described by

$$B_{tx} = \left(\left[\left(\frac{S_A}{S_A + S_S}\right) \cdot \left(\frac{T_t}{\Delta n}\right) - (N_f \cdot f_p \cdot T_d)\right] \cdot L_{p_{beacon}} \cdot n\right) + \left(N_f \cdot N_p \cdot L_p\right) \quad (52)$$

This removes the contribution of the sink beacons that are now sink data packets. Also, the reduced number of transmissions from the inclusion of sleep slots has been accounted for.

The results in figures 8.5(a-e) vary the amount of active time by adjusting the ratio of active slots to sleep slots, which gives the range *[1, 5, 8.33, 10, 25, 50, 100]%*. This variation has the effect of reducing the protocols energy consumption. The simulations were performed with the same parameters as described in chapter 6, but with the varying amounts of active time. Additionally, results for LASeR using the GTDMA MAC and with no energy enhancement has also been plotted as a comparison. In terms of delay, overhead and energy, LASeR will serve as an upper bound of performance. Though it should be noted that RASeR will not be able to achieve this, since, unlike LASeR, it is required to maintain its own gradient metric, rather than rely on existing location information. The general trend of the results show the trade-off between energy consumption and the protocols performance. The exception to this is the overhead in figure 8.5(c), which increases steadily as the active time is lengthened, due to the increased number of timeslots per second, which causes more beacon packets to be sent regardless of the rate of topology change or data generation. In terms of the PDR in figure 8.5(a), delay in figure 8.5(b) and throughput in figure 8.5(d), there is little to no degradation in performance, even when the active time is down to 25%. However, below this point the PDR and throughput begin to drop and the latency increases, although even at 5% the general performance is still very good, at which point the average energy consumption is less than that of LASeR, as shown in figure 8.5(e). The increases in delay at low energy, is due to the additional time that a node has to wait before it can transmit a packet, since packets can only be sent whilst the node is active. This can also cause a build-up of packets in the nodes' queues, which in turn will cause packet loss. Additional errors and delay can also be caused by the slowing of the gradient maintenance refreshing. If the cycle time is too long in comparison with frequency of topology changes, then the network won't be able to keep the gradient metric up-to-date, which will then cause routing errors. Overall, the active time should be set in an application specific manner, such that the performance requirements are balanced with the energy consumption according to the specification of the application. Additionally, it may be preferable to increase the length of the time slots to save energy, which essentially means that in

the time between the end of a packet transmission and the beginning of the next slot, nodes are able to sleep. Whilst this is a valid method of saving power, it was not simulated as the power required to start up a node often makes it preferable to have longer sleep time less frequently rather than shorter, more frequent sleep times.

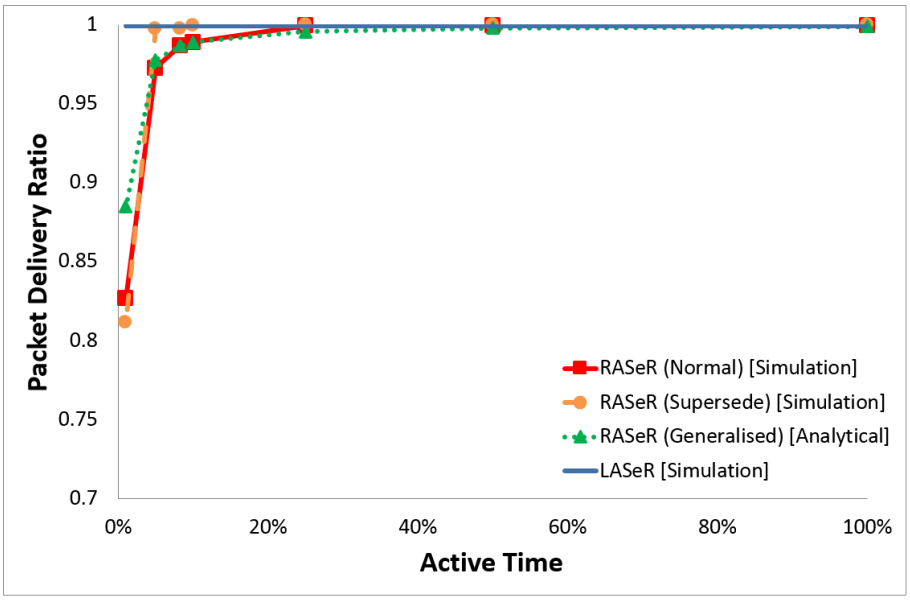


Fig. 8.5(a). Simulation and analytical PDR results for RASeR in both normal and supersede mode, over varying amounts of active time: *[1, 5, 8.33, 10, 25, 50, 100]%*. The simulated LASeR results are included as a comparison.

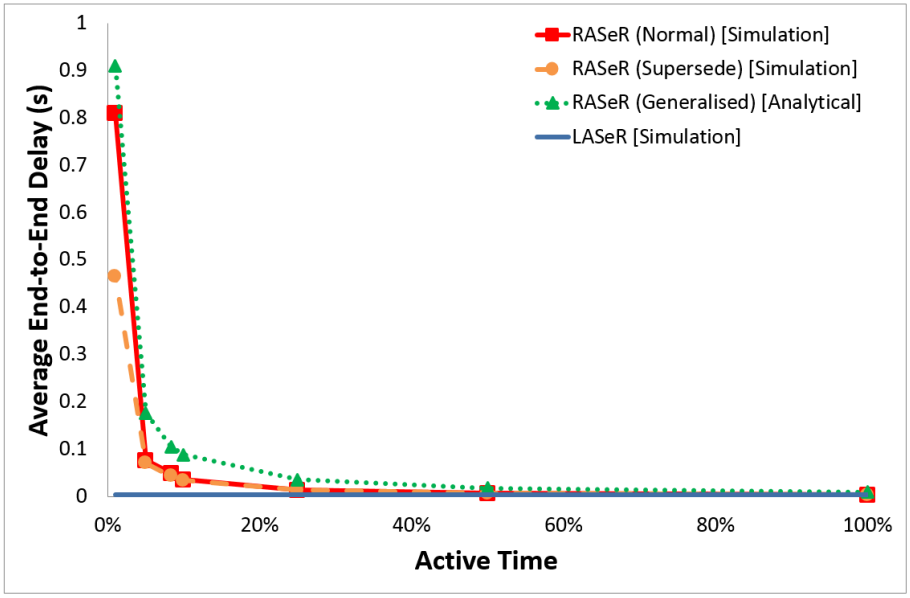


Fig. 8.5(b). Simulation and analytical end-to-end delay results for RASeR in both normal and supersede mode, over varying amounts of active time: *[1, 5, 8.33, 10, 25, 50, 100]%*. The simulated LASeR results are included as a comparison.

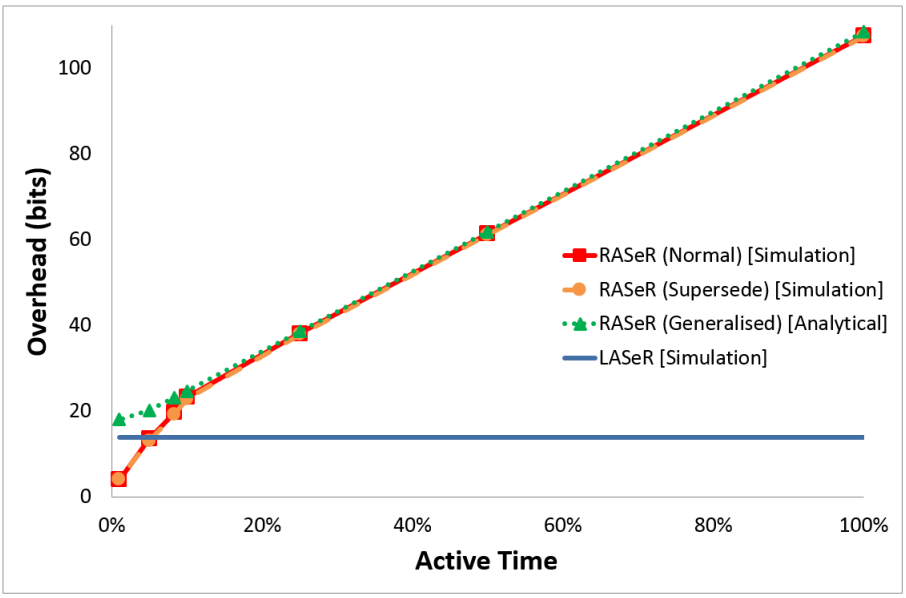


Fig. 8.5(c). Simulation and analytical overhead results for RASeR in both normal and supersede mode, over varying amounts of active time: *[1, 5, 8.33, 10, 25, 50, 100]%*. The simulated LASeR results are included as a comparison.

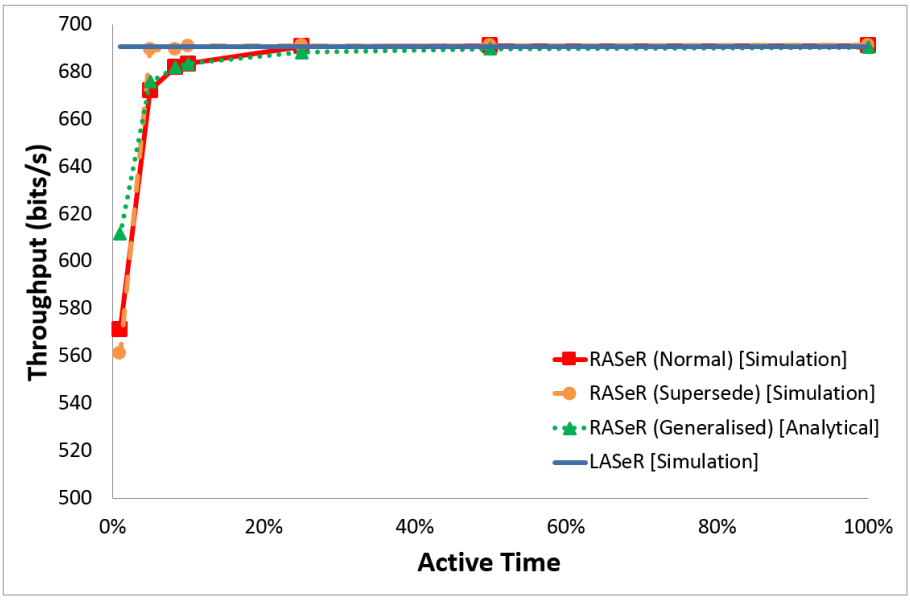


Fig. 8.5(d). Simulation and analytical throughput results for RASeR in both normal and supersede mode, over varying amounts of active time: *[1, 5, 8.33, 10, 25, 50, 100]%*. The simulated LASeR results are included as a comparison.

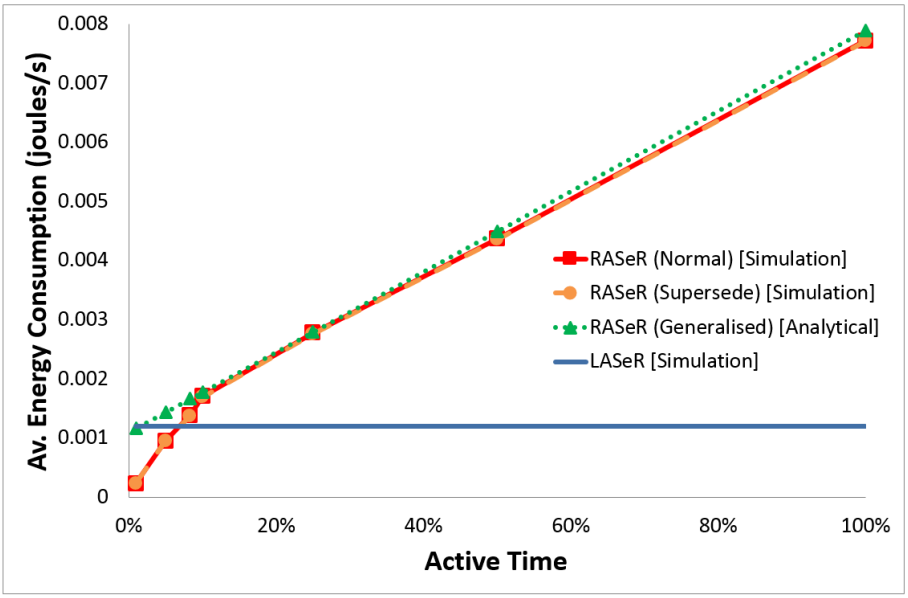


Fig. 8.5(e). Simulation and analytical energy consumption results for RASeR in both normal and supersede mode, over varying amounts of active time: *[1, 5, 8.33, 10, 25, 50, 100]%*. The simulated LASeR results are included as a comparison.

8.5 Pseudo Acknowledgements

Commonly, networking protocols define the use the use of acknowledgements (ACKs) to ensure the successful delivery of a packet. In this way a transmitting node will forward a packet and if the receiving node hears the packet correctly then it will respond with an ACK to let the transmitter know. Subsequently, if the transmitting node does not receive an ACK, it assumes the packet has been lost and may attempt to retransmit it.

RASeRs use of blind forwarding means that a transmitting node does not target one specific receiver, instead, any of the nodes that hear the transmission may decide to forward the packet. This makes the use of ACKs difficult, since it's likely that multiple nodes will receive the transmission and all attempt to respond with an ACK at the same time, causing collisions. If a transmitting node is waiting to receive an ACK, but the ACKs collide, then they won't be heard by the transmitting node and it may decide to try and resend the packet, which would be a waste of bandwidth and power.

To prevent this, without the addition of any more overhead, pseudo ACKs (pACKs) are introduced. Essentially, after a node has transmitted, the packet is not removed from the queue, it is instead marked as unacknowledged. Thereafter, if the node overhears another node broadcasting the same packet, it will know that the packet was received and may then remove it from the queue. Additionally, the priorities with which a node selects the next packet to send as slightly adjusted, such that the highest priority is given to the oldest packet in the queue with priority status. The second highest priority is given to the oldest packet with diversity status and

the third highest priority is given to the oldest unacknowledged packet. In this way, until the node confirms that the packet was successfully received by another node, it will keep retransmitting the packet intermittently.

To accommodate the added number of packets being stored in the queue, and to reduce the number of packets lost from queuing overflow, the size of the queue has also been increased. The combination of using pACKs and an increased queue length should reduce the number of lost packets from nodes transmitting when they aren't in range of any potential forwarding neighbours and also from packets lost from queuing overflow.

The results given use the parameters described in chapter 6, and compare RASeR in normal, supersede and pACK mode. Figures 8.6(a-e) present the modes under varying mobility levels. Figure 8.6(a) shows that the pACK has a slight improvement in PDR, with an average of 99.99%, however the already high performance of both normal and supersede mode means that there isn't much improvement that can be made in this scenario. The same is true of the throughput results in figure 8.6(d). Since packets are more likely to be retransmitted in pACK mode, the overhead in figure 8.6(c) and consequently the energy consumption in figure 8.6(e) show slightly increased levels, with a maximum increase of only 26.81 bits and 0.0017 joules/s. Additionally, the delay remains low as shown by figure 8.6(b), but there is a marginal increase at the highest speed, due to slightly more packets being delivered.

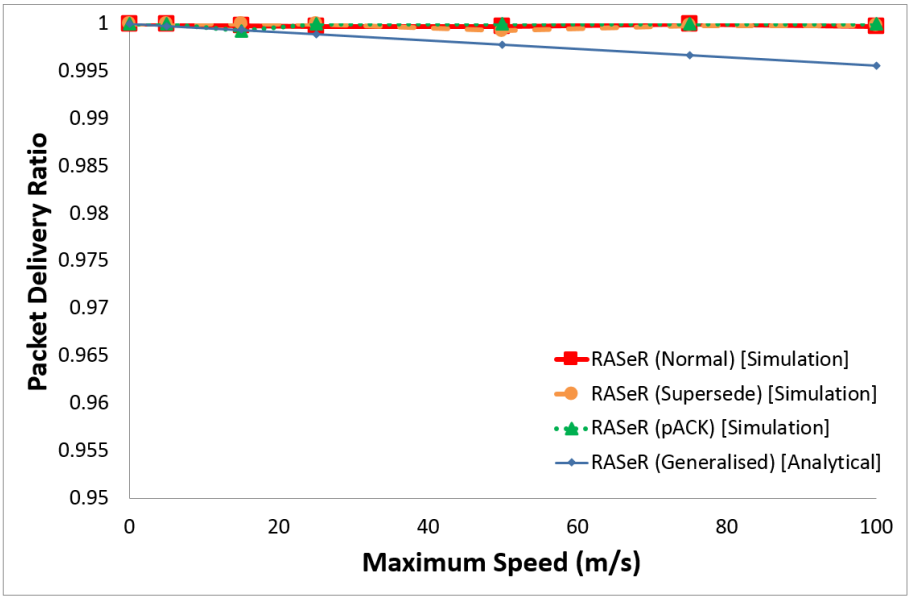


Fig. 8.6(a). Simulation and analytical PDR results for RASeR in, normal, supersede and pACK modes, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

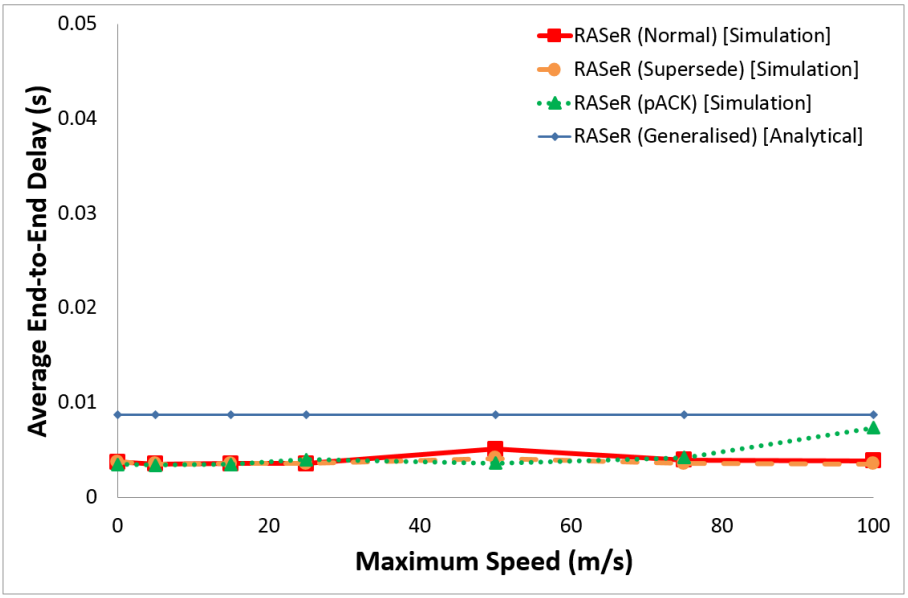


Fig. 8.6(b). Simulation and analytical end-to-end delay results for RASeR in, normal, supersede and pACK modes, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

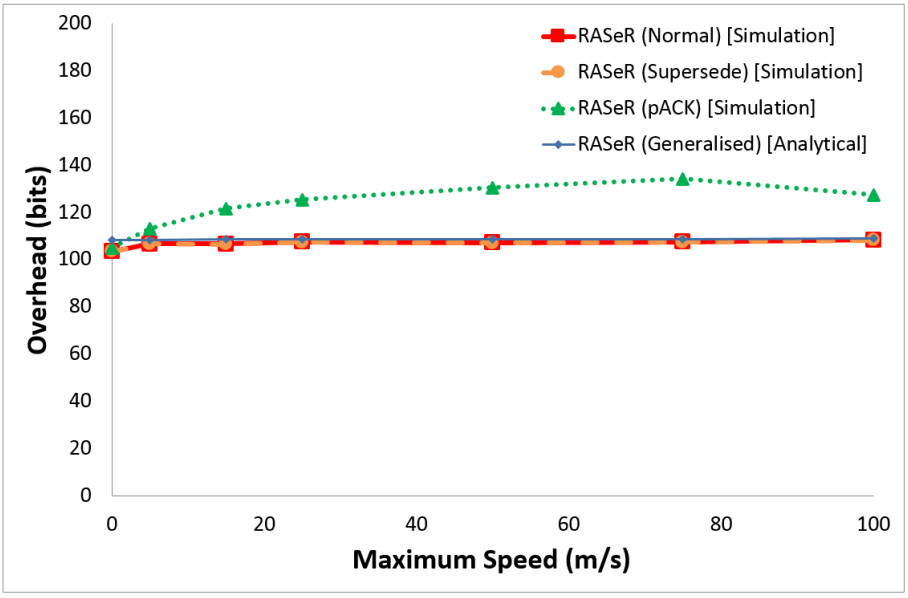


Fig. 8.6(c). Simulation and analytical overhead results for RASeR in, normal, supersede and pACK modes, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

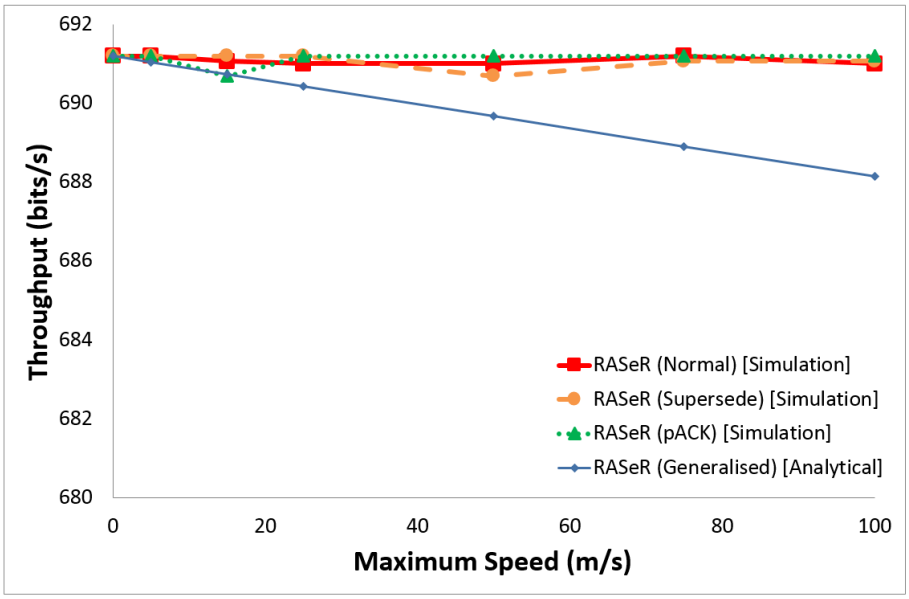


Fig. 8.6(d). Simulation and analytical throughput results for RASeR in, normal, supersede and pACK modes, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

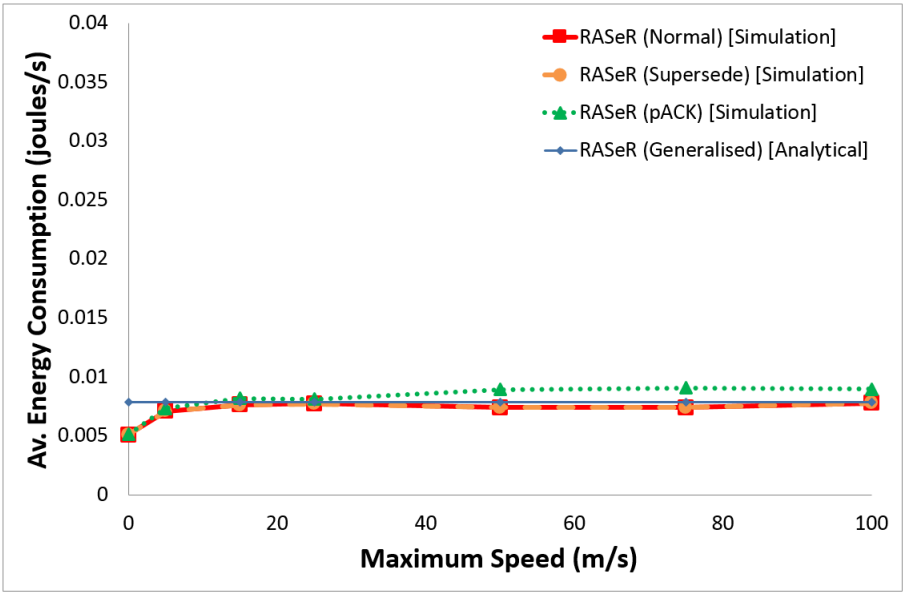


Fig. 8.6(e). Simulation and analytical energy consumption results for RASeR in, normal, supersede and pACK modes, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

The scalability results in figures 8.7(a-e) highlight the advantage of the pACK mode, especially in figure 8.7(a), which shows the sustained high level of PDR even with 100 nodes, where both normal and supersede mode begin to dip but pACK mode reaches a minimum of 99.66%. As before, the cost of this is in the small amount of added overhead, which causes a slight increase in energy consumption, shown in figures 8.7(c) and 8.7(e) respectively. Also, the latency is increased due to the higher packet delivery, as shown by figure 8.7(b), but the throughput shows an improvement of up to 138.05 bits, which can be seen in figure 8.7(d).

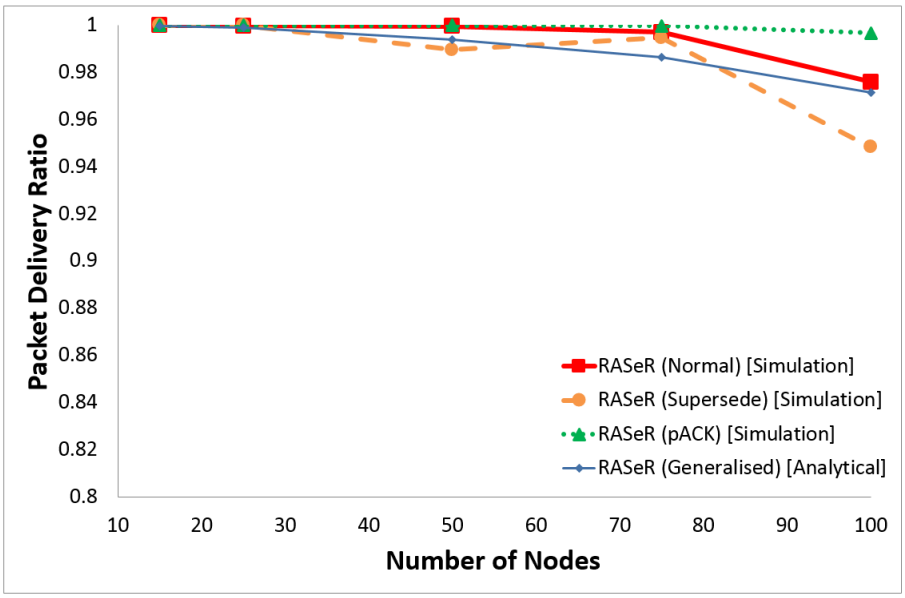


Fig. 8.7(a). Simulation and analytical PDR results for RASeR in, normal, supersede and pACK modes, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

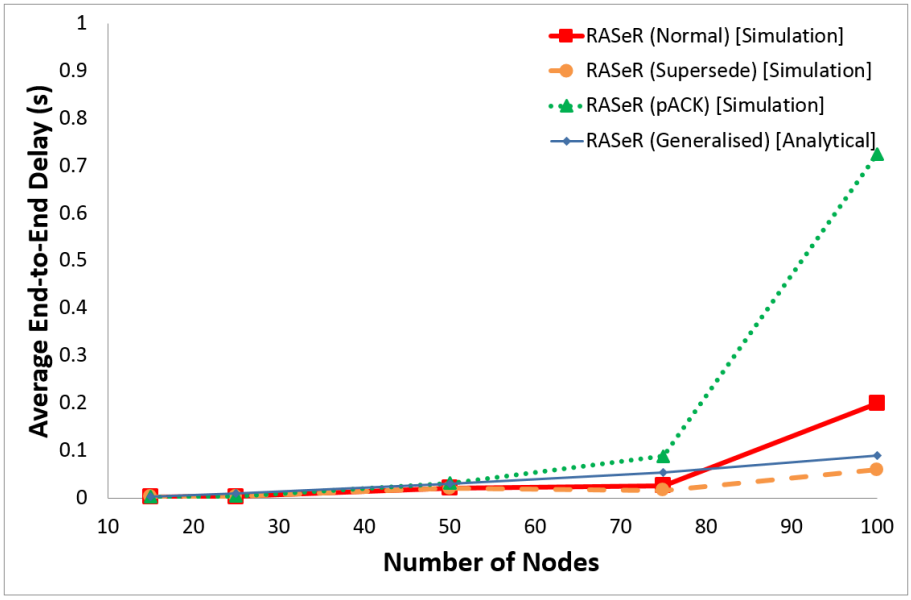


Fig. 8.7(b). Simulation and analytical end-to-end delay results for RASeR in, normal, supersede and pACK modes, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

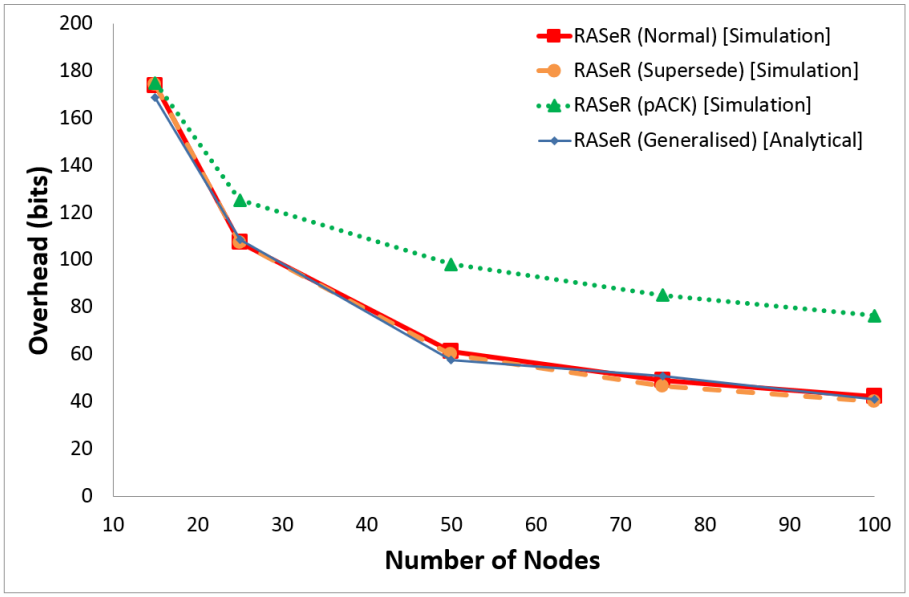


Fig. 8.7(c). Simulation and analytical overhead results for RASeR in, normal, supersede and pACK modes, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

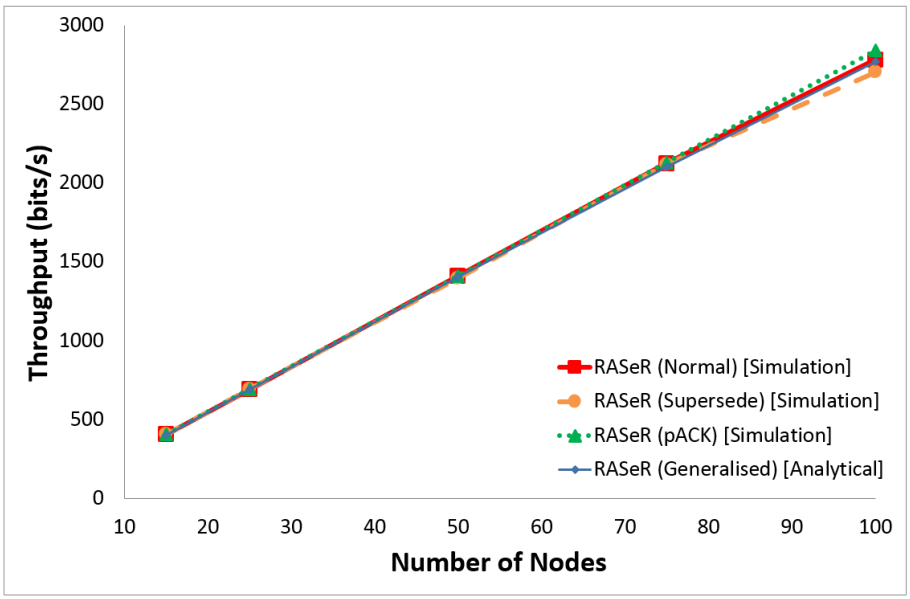


Fig. 8.7(d). Simulation and analytical throughput results for RASeR in, normal, supersede and pACK modes, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

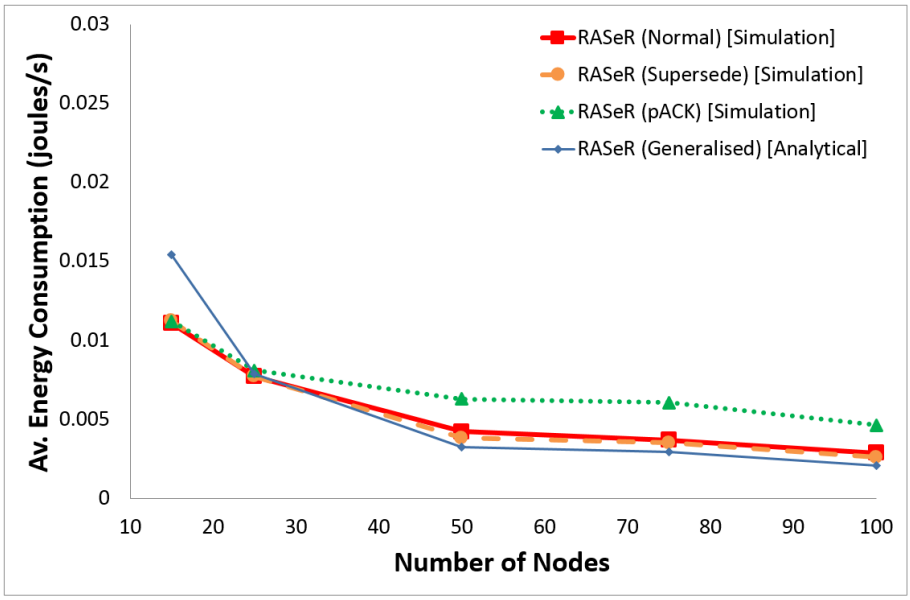


Fig. 8.7(e). Simulation and analytical energy consumption results for RASeR in, normal, supersede and pACK modes, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

Figures 8.8(a-e) give the results for varying traffic levels, which again shows that pACK mode has the best PDR of at least 99.95%, as can be seen in figure 8.8(a). As with the last two scenarios, the trade-off for this added reliability is in the small increase in energy consumption. This is highlighted by figure 8.8(e), which shows the increase in energy consumption has a maximum value of 0.0030 joules per second. However, the overhead in figure 8.8(c), the delay in figure 8.8(b) and the throughput in figure 8.8(d) show pACK mode to give comparable results to both normal and supersede modes. barely any increase in overhead.

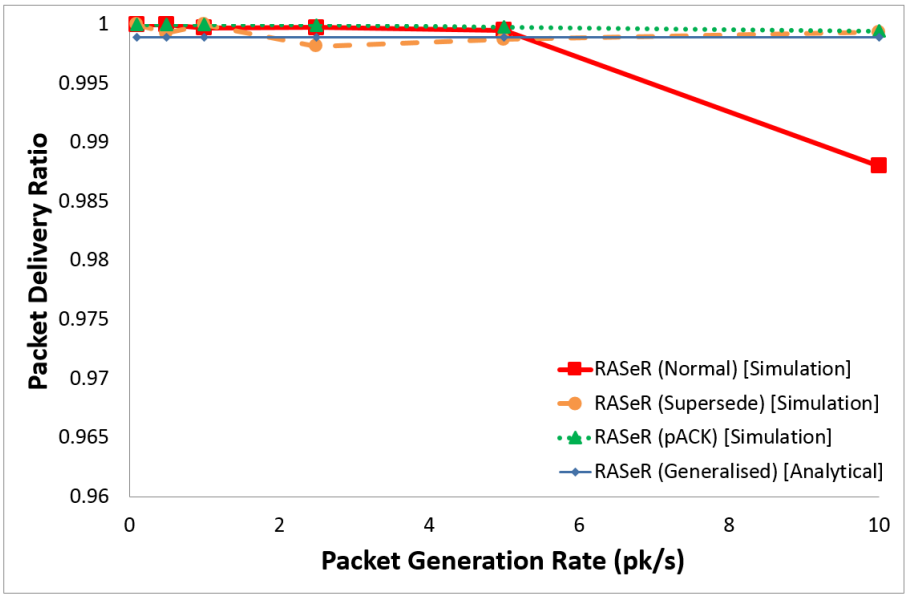


Fig. 8.8(a). Simulation and analytical PDR results for RASeR in, normal, supersede and pACK modes, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

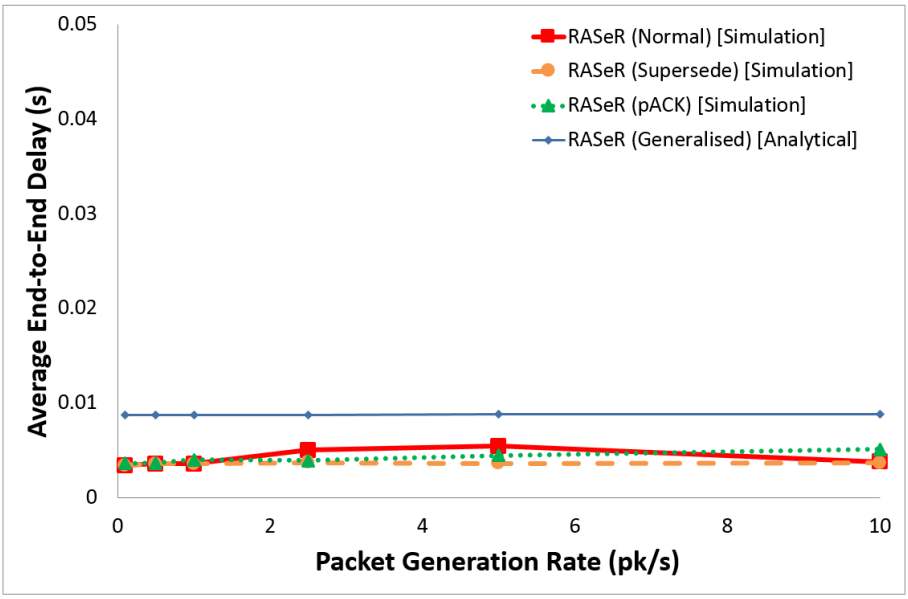


Fig. 8.8(b). Simulation and analytical end-to-end delay results for RASeR in, normal, supersede and pACK modes, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

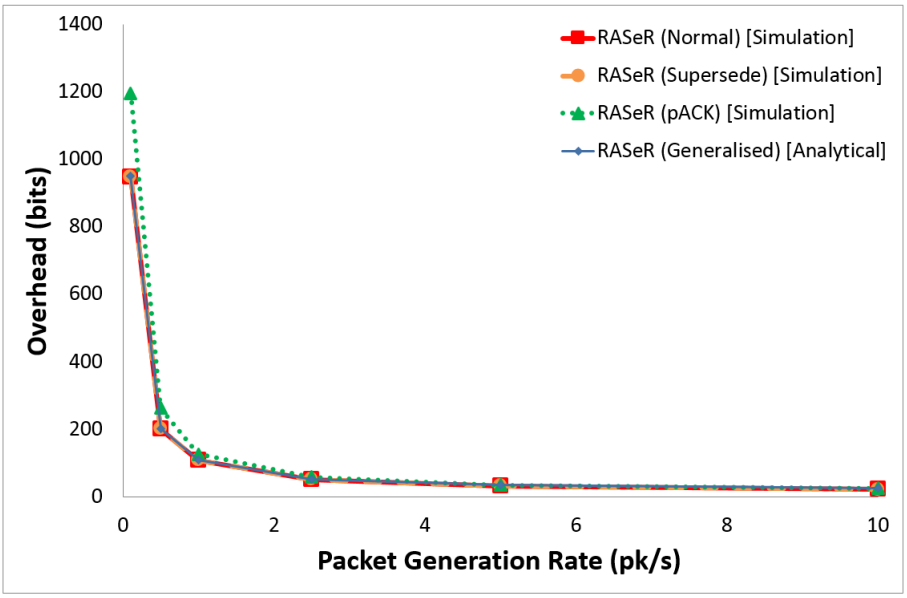


Fig. 8.8(c). Simulation and analytical overhead results for RASeR in, normal, supersede and pACK modes, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

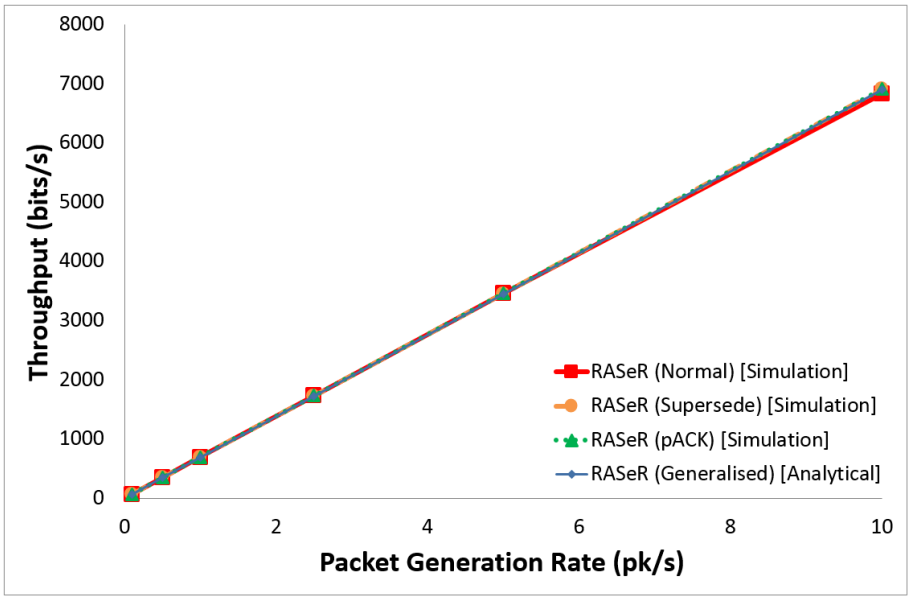


Fig. 8.8(d). Simulation and analytical throughput results for RASeR in, normal, supersede and pACK modes, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

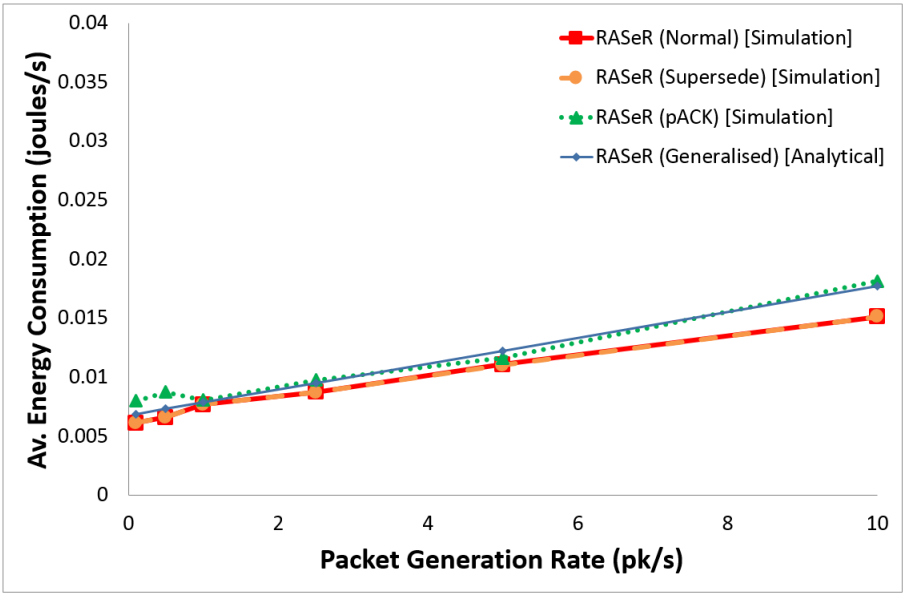


Fig. 8.8(e). Simulation and analytical energy consumption results for RASeR in, normal, supersede and pACK modes, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

In general, pACK mode is designed for scenarios in which receiving every packet is worth the trade-off in terms of energy consumption. Whereas, supersede mode is more suited to the timely delivery of the latest data, which comes at the cost of some packet loss. Comparatively, normal mode lies somewhere between the two, with no weighting in any direction.

8.6 Unmanned Aerial Vehicle Aided Search and Rescue

UAV aided search and rescue (SAR) is a relatively new research topic [2.8, 8.1, 8.2] but shows significant promise for advancing and improving critical SAR missions. The envisaged

scenario is one in which SAR teams are aided by the use of a number of intelligent UAVs. One advantage of using a group of UAVs is that they can quickly search a much larger area than a single helicopter. They also have a much quicker deployment time as they don't need to wait for a crew to assemble and prepare, which could be critical in saving lives. This section describes the application, its requirements and the implication of these requirements on the communications protocols used. The aim is to illustrate how RASeR is not only a high performance protocol, but is well suited to real world deployments and that additional modes of operation enhance this.

8.6.1 Scenario

Similar to [8.3], the imagined framework for this application is depicted in figure 2.3, which shows a small group of UAVs supporting the search effort of a SAR team in a helicopter. To speed up response times the UAVs will report directly to the team in the helicopter.

It is clear that as technology advances, drones will become smarter and subsequently require less human intervention [8.4]. For example, in this scenario a simple set of UAVs may require manual flight control and have to transmit images back to base in order for an operator to identify a target. This kind of basic technology puts high demands on communications, which are expected to provide high bandwidth, bidirectional links to each individual UAV. This is often done with direct, point-to-point transmissions, which are usually very power hungry. In contrast to this, an intelligent set of UAVs could cooperatively determine their own search paths based on a general geographic area. They may also be equipped with image processing technology, capable of identifying a target automatically [8.5]. In this case, when detected, only the target location would need to be reported to the operator, which requires significantly less bandwidth and power.

In the future it may also be the case that swarms are expected to assist with the rescue itself, by extracting targets from danger zones.

8.6.2 Application Requirements

As with many applications, UAV aided SAR has a relatively specific set of requirements. These include hardware restraints on weight, size, power and cost. Also, the reliability of the gathered sensory data is important; such that there is a low probability of a UAV reporting a false negative or a false positive on a targets location [2.8].

Another requirement of this application is in utilising efficient search strategies [2.8], as a more efficient method could save time and power. Commonly there are two general options; fixed formation and cooperative search. Fixed formation involves the UAVs flying in a predefined arrangement to cover a fixed area [8.3], which is a simple, thorough and reliable

method. Alternatively, cooperative search algorithms may be controlled by a single central controller [8.6] or require the UAVs to share information about where they have searched and where they are going to search [8.7]. This information is then used by individual UAVs to make logical decisions about where should be searched next.

A major requirement of these systems is collision avoidance [8.8], which includes mitigating collisions between UAVs as well as between the UAVs and other objects. Again, there are two options; sensor based object detection or position reporting. Sensor based object detection is based on the UAVs being able to utilise inputs from sensors, such as cameras or radar, to navigate obstacles such as trees, buildings or other vehicles. The other option necessitates that each UAV reports its position or destination, so that nearby UAVs can avoid it. Sensor based detection is generally a better option as it can detect any type of object, whereas position reporting can only prevent UAVs colliding with each other. However, due to the complexity involved in signal processing, the position reporting method is more common [8.8].

The fusion of data from multiple sources [2.8] at the sink is important for the interpretation of the results. Data is often presented in the form of a belief map, in which the search area is split into grids. Each of the squares on the map are then given a probability based on the reported data from the UAVs, which should indicate the most likely location of the target [8.2].

Control of the UAVs is also a concern in these systems [8.8] since the mission parameters may change during an operation. The new information will need to be distributed to the UAVs, which will have to be equipped to understand and translate the commands into actions. These actions may include changing formation, changing the search location or ending the mission.

8.6.3 Communication Implications

The main communication requirement for this application is the timely delivery of sensor data; if a node detects the target it is important that this is reported quickly and reliably to the sink. The faster the data is reported, the faster the SAR team can respond. However, it is also important that packet loss is kept to a minimum in order to reduce the chance of the sink failing to receive crucial information about the targets location.

The ideal solution to collision avoidance would be a sensor based object detection method, which would require no additional communications overhead. However, using a position reporting technique would require each UAV to periodically broadcast its location to its neighbours.

Additionally, a fixed formation search strategy, which is simple and can be optimal for a fixed target, would also negate the possibility of UAVs colliding with each other and require no extra bandwidth. On the other hand, with a cooperative search method, the UAVs would require

the ability to share certain information. This additional information may require a significant amount of overhead since a globally coordinated search effort would require all of the nodes to keep up-to-date with the current state of the search. In this way data from each node would need to be propagated to every other node.

With the collision avoidance and search strategy issues, there are preferable solutions that don't require additional communications. However, with regard to the control of the UAVs, predefined commands will need be transmitted to all nodes from the sink. These commands may include a *'come home'* message, which would indicate the end of a mission and instruct all UAVs to return to base. If a fixed formation search is used, UAVs could be programmed to perform the search pattern relative to a single centre location. This would enable the sink to transmit a single coordinate to all of the UAVs and the nodes would shift the search pattern around the new centre point. The UAVs may even have multiple predefined search patterns, which can then be switched between by sending a command from the sink. The significant additional communications overhead for this is unavoidable.

In terms of data display for the end user, a belief map could be constructed at by the sink to compile the gathered data and inform the SAR teams search. Additionally, the technique of data fusion may be introduced at a node level based on repeated sightings of a single target. For example, two nodes may report the location of the target to an intermediary node, which can then aggregate the data into a single packet before forwarding the data. This can save on bandwidth and potentially reduce traffic in the network, at the cost of the additional complexity and processing time at the node level.

8.6.4 RASeR Suitability

The results given in this chapter and in chapter 6, illustrate how RASeR protocol is well suited to the application of UAV aided SAR due to its high reliability in terms of PDR and very low end-to-end delay times. These two aspects are critical for the timely delivery of data to the sink. The results also show how RASeR can handle different levels of scalability, traffic and mobility, making it suitable for a variety of other applications as well.

How the design of RASeR is advantageous to the application of UAV aided SAR is also highlighted by the results presented in this chapter. In particular, as supersede mode disregards out-of-date data in favour of newer data, delay times can be reduced, such that the latest position of the target is given priority.

With regard to the control of the UAVs, predefined commands can be transmitted to all nodes with the use of the reverse flooding mechanism. The simulations have shown this to have no adverse effect on the protocol. These commands will allow the sink to control the swarm by

relocating the search mid-mission, changing the parameters of the mission or ending the mission.

It should also be noted that although a sensor based scheme for preventing the UAVs from colliding is preferable, the alternative would require the local sharing of location information. In RASeR it would be simple to include an extra field in the beacon packets for this purpose. This would mean that current location data is regularly broadcast to each nodes neighbours. In addition to avoiding collisions with UAVs and other objects, this would also enable a cooperative search strategy to be used. Since the additional overhead is added to the beacon packets, the slot length would not need to be adjusted, so it would have minimal effect on the performance of the protocol.

8.7 CONCLUSION

This chapter has presented four additional modes of operation for RASeR. The first was a supersede mode for giving priority the low latency delivery of the most up-to-date data. The second is a reverse flooding technique for disseminating network wide messages from the sink to all sensor nodes. Thirdly, an energy saving mechanism for introducing coordinated sleep cycles to conserve power was introduced. Lastly, a pACK mode was presented, which increases the reliability of the protocol. Additionally, the application of UAV aided SAR was evaluated to highlight how the additional modes enabled RASeR to satisfy the requirements of a particular application more thoroughly, which further justifies the need for these additions.

Overall, by equipping RASeR with these options, the protocol may be further customised and optimised to be suitable for a wider range of applications. The protocol may even be adjusted during deployment as the networks mission objectives change.

The next chapter will analyse RASeR with the use of a testbed experiment.

CHAPTER 9

RASER TESTBED

9.1 INTRODUCTION

Generally, the primary method for evaluating routing protocols is through simulation, however some behaviours cannot be captured through this technique. For this reason experimental testbeds are sometimes used to further investigate the performance of systems since they are often the closest analogue to a real deployment. In this chapter RASeR is evaluated using a small testbed in order to see the practical implementation issues as well as how it performs outside of a simulator. RASeR was selected for testing since it is the most widely applicable; it doesn't require any additional hardware or techniques for location awareness and it doesn't have any expectations in terms of the frequency of data generation.

9.2 EXISTING TESTBEDS

There are a number of testbeds designed for testing wireless multihop protocols, however, only a small selection of these have incorporated some method of node mobility. One example of such a testbed is the Uppsala University's APE (Ad-hoc Protocol Evaluation) testbed [9.1], which uses the choreographed movements of volunteers to enable the mobility of nodes. The volunteers carry laptops with the software installed and an 802.11 network card. The mobility is orchestrated by the volunteers performing movements based on instructions that appear on the laptops. Each node logs its own data and after the experiment these logs are uploaded to a mobile node for analysis.

Alternatively, the Mobile Emulab [9.2], uses autonomous ground robots with additional hardware running Linux as well as Wi-Fi cards and sensor motes. The robots paths are defined by waypoints and obstacles are avoided by using visual feedback. The visual feedback is provided by a network of cameras tracking the positions of nodes in the network. The test network is created using the sensor motes and the Wi-Fi cards are used to create a control network, which is connected to the internet so that users may control the testbed from any internet terminal.

MiNT-m (Miniaturized Network Testbed – Mobile) [9.3], also uses cameras to track the position of nodes. This enables the trajectories of nodes to be adjusted dynamically to avoid

collisions. The nodes themselves are modified autonomous ground vehicles, with four 802.11 cards each. Radio signal attenuators are used to limit the transmission range of the nodes, so that experiments can be performed in smaller spaces. The testbed also provides automatic recharging functionality, such that the testbed may be unmanned.

The SCORPION (Santa Cruz Mobile Radio Platform for Indoor and Outdoor Networks) testbed [9.4] is one of the most diverse systems in terms of mobility platforms. The testbed consists of autonomous ground and autonomous air vehicles, manned 'briefcase' nodes as well as some nodes attached to buses. All the nodes are equipped with various computational hardware as well as 802.11 radios, which are used to create the test network.

Three options of ground vehicle are provided by the Pharos testbed [9.5], all of which are controlled by a microcontroller and a processor running Linux on board each node. The communications is done through 802.11 radios and the robots navigation is done using a selection of sensors.

The Sensei-UU testbed [9.6] is made up of sensor motes combined with a smartphone and carried by a Lego robot. The sensor nodes are controlled by host nodes, which consist of Wi-Fi routers and sensor motes. The host nodes are, in turn, controlled by a laptop, which uses the 802.11 radios as a control channel.

The testbeds discussed here have been designed to enable the general testing of protocols inadaptable scenarios. For this reason, many have chosen highly robust mobility platforms and the high data rate radios provided by the 802.11 standard. In the next section the aims of our testbed will be detailed and the design decisions explained.

9.3 Testbed Design

The primary aim of this chapter is to evaluate the possibility of deploying RASeR on real hardware. To achieve this, a small testbed will be created to run the protocol on and a selection of metrics will be collected. The testbed should be able to run in an office sized area or smaller. Since RASeR is a multihop protocol designed to be used in mobile scenarios, each node in the testbed will require some form of transceiver and mobility platform. Whilst the requirements of our testbed could be fulfilled by constructing purpose built nodes, this would be time consuming and expensive. As such commercial-off-the-shelf (COTS) products are favoured. Since there are currently no COTS mobile sensor motes, each node will consist of a normal sensor mote and some kind of mobility platform.

There are many COTS sensor motes available, with varying profiles. For this project the popular MEMSIC IRIS motes [4.20] were chosen for their low cost and availability. The IRIS motes provide a complete sensor node, with a 250kbps, 2.4GHz transceiver and an 8-bit

microcontroller. The 128 Kbytes of memory will provide more than enough space for RASeR. Additionally, the IRIS motes come with dedicated software that allows over-the-air programming (OTAP), which is a major advantage. The nodes are commonly loaded with the TinyOS [4.23] firmware, which will allow for faster implementation of RASeR, as it can be done within the existing TinyOS framework.

TinyOS is C based open-source software designed for use in event driven environments such as sensor networks. TinyOS allows for interrupts to trigger event handlers, which then may add tasks to a queue, from which they may be selected run. Large tasks are split into smaller parts, such that no one task occupies the processor for too long.

The IRIS mote also has an expansion connector, allowing sensor boards to be added, which are also supported by the TinyOS firmware. Additionally, the low power consumption of the mote ensures that it will have a long lifetime and is suitable for many applications.

In terms of a mobility platform, there is a vast array of ground, air and even submersible options, of varying complexity. For its simplicity and cost, the DFRobotShop Rover V2 [9.7] was chosen. It is a tracked ground robot, which is controlled by an Arduino [9.8] built in to the chassis. The dual motor rover allows for multiple devices to be mounted onto a designated prototyping area. The simple Arduino software makes programming the device very easy.

A complete node can be seen in figure 9.1, with the assembled rover and an IRIS mote mounted on top.

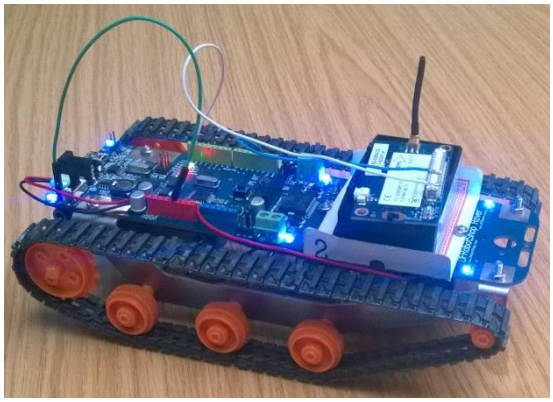


Fig. 9.1. One of the fully assembled nodes, consisting of a rover and an IRIS mote.

9.4 PROTOCOL IMPLEMENTATION

The majority of the implementation work was in the writing of RASeR in the nesC style of TinyOS, the main structure for which is shown in the flow charts in figure 9.2. There are four main event triggers in RASeR, power up, the GTDMA timer, a packet being received and a packet being generated. Flow chart (a) shows how, on power up the node simply runs an initialisation routine to initialise variables and allow other modules to run their own

initialisation routines. Flow chart (b) shows the nodes procedure when a new timeslot starts. It first checks whether the timeslot is its own based on the GTDMA schedule and if it is, then the queue is checked for data waiting to be transmitted. If there is data to be transmitted then the oldest packet with the highest priority is selected, then the packet is compiled and broadcast. If it's the nodes timeslot but there is no data to be sent, then the node transmits a beacon instead.

To mimic the act of a sensor taking a reading or being triggered by an event, the nodes are given a schedule, which indicates when the node should create a packet. In the event that a node should impersonate receiving some sensor data, flow chart (c) shows that the node will simply construct a new packet and add it to the queue. Upon the reception of a packet, flow chart (d) shows that the node should first update its hop count. Then, if the packet has new data in it, the data is stored in the queue. If the data is not new, the information in the queue is updated.

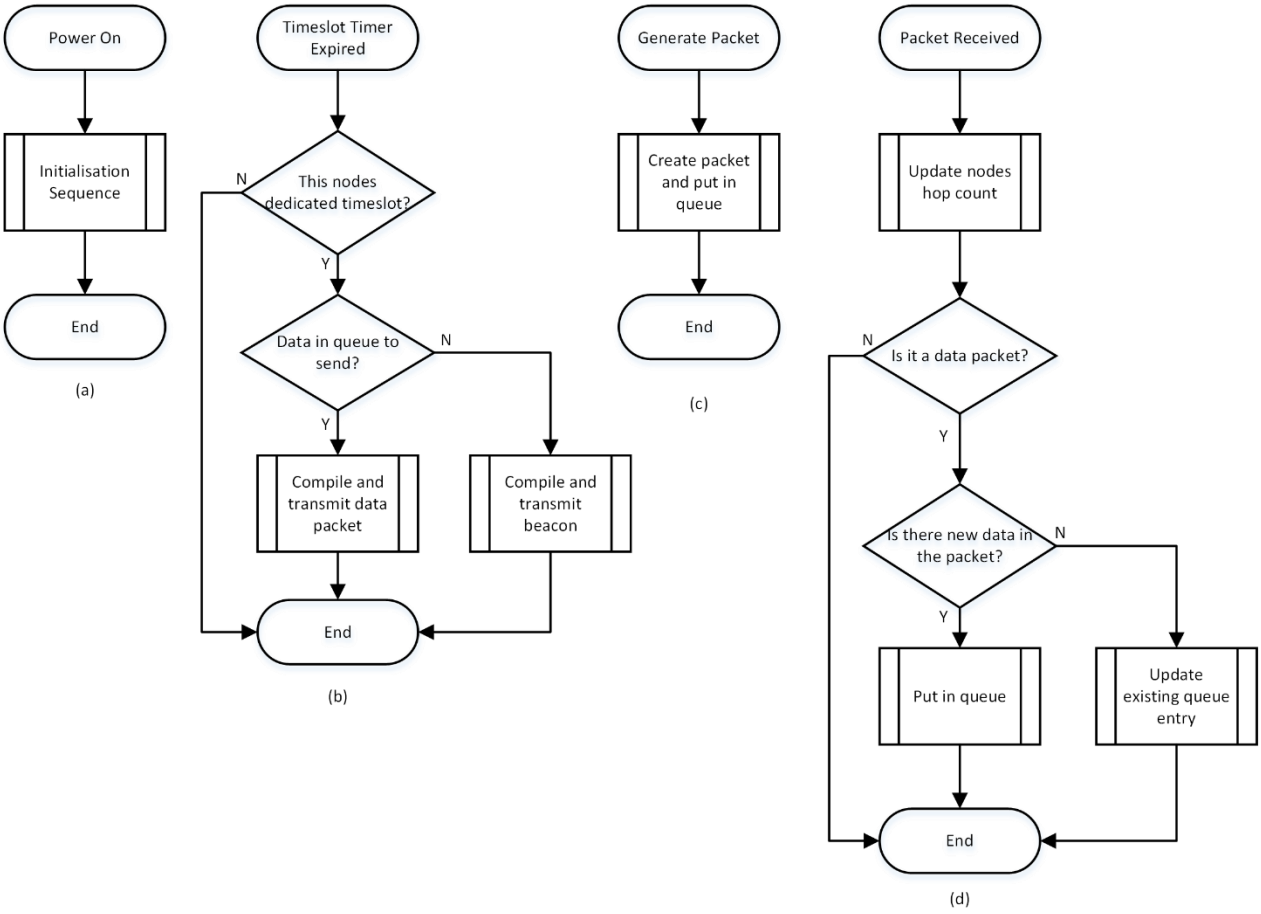


Fig. 9.2. Event driven structure of RASeR, shown by flow charts of the motes responses to different interrupts, namely (a) Power On, (b) Timeslot Timer Expired, (c) Generate Packet and (d) Packet Received.

Since the testbed is designed to be used in a small area, the relatively large communications range of the IRIS motes needed to be reduced. For this an RSSI (Received Signal Strength Indicator) threshold was chosen. In this way, nodes will reject packets that are received and weaker than the threshold. By shifting the threshold the effective communication range of the nodes can be adjusted. Using this method the sink can be used in a promiscuous way to listen to

the communications over the entire network, which can be used for debugging and protocol analysis. The choice of RSSI threshold level will be discussed in the next section.

There are two exceptions to the use of this RSSI threshold, the first is the initial beacon packet sent by the sink. This packet is received by every node in the network and triggers the timeslot timer. This means that the network is synchronised to begin with, such that every nodes timer is started at the same time and that everyone starts counting the slot numbers at the same time. The second exception is when the sink issues an OTAP command, which is simply a beacon packet with an additional flag. If this flag is set then the nodes that hear the beacon should run a command which causes the mote to reboot into the OTAP program. This then allows the nodes to be programmed wirelessly.

Furthermore, the transmitted packets were designed to be larger than specified by RASeR, in order to carry additional information such as the number of hops taken by that packet. It should also be noted that RASeR was built on top of the existing default method of communications designed to work with the MoteWorks suite. As such there is also added overhead from this and the packets are sent using a CSMA/CA MAC. Essentially, this means that RASeR and its GTDMA MAC sits on a CSMA/CA MAC. This was done because the 2.4GHz band is very popular and is therefore at high risk of external interference. However, RASeR is designed to be used in isolation from other networks, either through frequency selection, spread spectrum techniques or other methods. As such it will need some kind of way to avoid significant packet loss caused by collisions from external sources, which the CSMA/CA MAC can assist with. Essentially, the GTDMA slot time is increased to allow the CSMA/CA MAC time to ensure that packets are delivered as reliably as possible. So whilst the two MACs are stacked on top of each other, the existence of CSMA/CA is completely transparent to the GTDMA and upper layers. As such, the protocol runs exactly as it would if deployed on custom hardware, since the effects of external interference and the CSMA/CA MAC are absorbed by the extended timeslot length.

To test how well RASeR handles the movement of nodes, the rovers are programmed with a random mobility model, as described by the flow chart in figure 9.3. Initially the program sets a random seed by taking an analogue reading from an unconnected pin. Based on a pseudo random number generator the robot then chooses a direction, speed and duration and consequently moves in that direction at the specified speed for the chosen duration. This process is then repeated indefinitely.

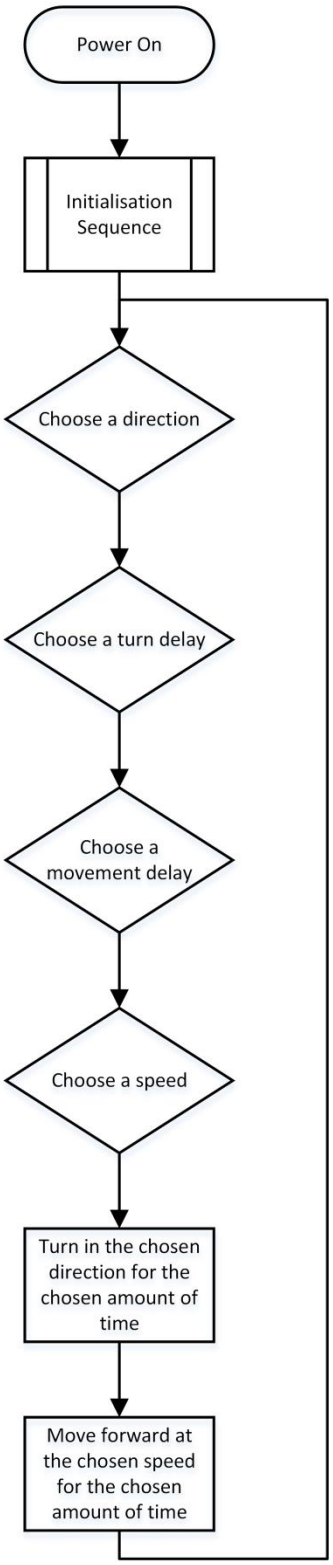


Fig. 9.3. Flow chart of the random mobility model implemented on the rover robots.

9.5 RSSI Range Limiting

In order to limit the range of the nodes an RSSI threshold was set, such that motes can filter out transmissions from other nodes that are deemed to be out of range. In this way the effective communications range of the nodes can be reduced and the testbed can be deployed in a much smaller area whilst still being able to gather meaningful results. The fact that RASeR uses a GTDMA MAC makes this possible; otherwise this would result in large numbers of collisions. For this reason, most testbeds in the literature use attenuators to reduce the strength of the transmission, in order to limit the range. The added advantage to using an RSSI threshold is that the transmissions can still be heard by any device in promiscuous mode from anywhere in the network. This can then be used for both debugging and collecting data whilst the network is running.

To determine the appropriate level for the threshold a test was run consisting of two like sensor nodes, *A* and *B*, programmed to regularly send messages to each other. The transmitted messages contained the RSSI level of the last received message so that the sink could be setup to eavesdrop on this communication and log the data. In this way the RSSI levels received by the two nodes were recorded at 10cm intervals, from 10cm to 300cm, the results from which are shown in figure 9.4.

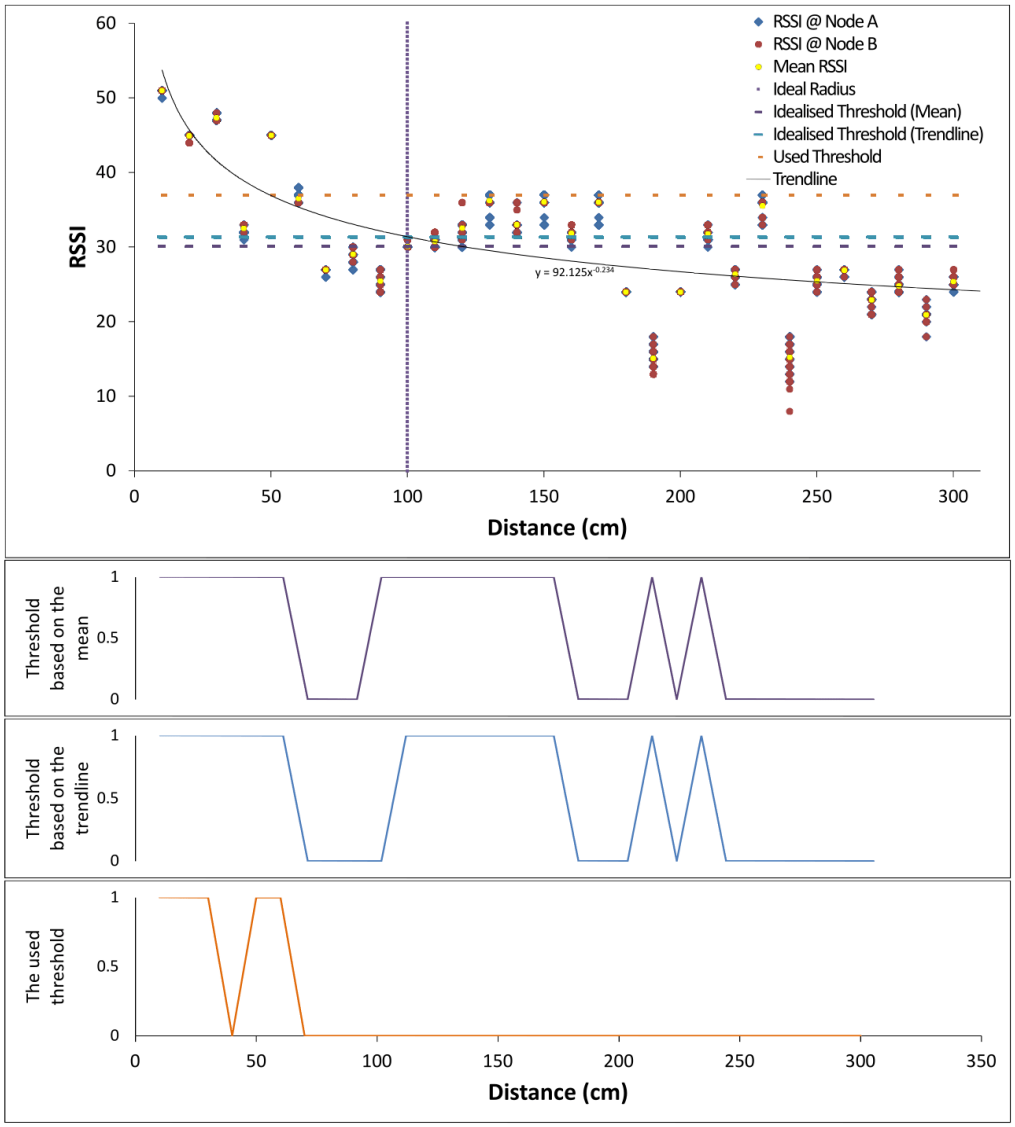


Fig. 9.4. The top graph shows RSSI measurements between nodes A and B over distances between 10 and 300cm, with 10cm intervals. The values from node A are plotted in blue, the values from node B are in red and the mean values are plotted in yellow. The ideal transmission radius of 1m has been displayed as a vertical line and a generated trend line has also been added. Three potential threshold choices are also plotted and the lower three graphs show the effect of the three threshold choices on the packets received.

Figure 9.4 plots the measurements from the two sensor nodes, which show an expected decrease in power as the distance increases. However, there is a high level of variance at each interval, which causes the mean values at each increment to fluctuate considerably. This high spread of values makes selecting an appropriate threshold quite difficult. The graph also gives a vertical ideal radius marker, which indicates the 1m distance at which the transmission boundary should be placed. A trend line has also been plotted, which highlights the general downward trend of the data based on the curve $y=92.125x^{-0.234}$.

In choosing the threshold to use, initially the mean value of 30, at the 1m point, was considered. However, this value was particularly low and would allow nodes to communicate at distances of more than 2m. The same is true for the second threshold option, which was taken from the intersection of the trend line with the ideal radius. Subsequently, the used threshold of

37 was chosen to eliminate any communication between nodes that are more than 1m apart. Although, it is noted that this will also impede some communications between devices that are within the desired transmission range and as such make the environment more challenging for the routing protocol under test. In addition to this, the varying channel conditions will add to the naturally dynamic topology of the mobile network. A comparison of the three described thresholds are shown in the lower section of figure 9.4. These binary graphs show the distances at which nodes would accept packets, given the selected threshold.

9.6 Experimental Setup

The experiment consists of five mobile sensor nodes and one fixed sink in a network area of *1.6m* by *1.6m* as shown in figure 9.5. The RSSI threshold was set to limit the communications range to approximately *1m* and the maximum speed of the rovers was measured at *0.14m/s*. Since the sink is fixed in the corner, only 31% of the total network area is within the *1m* transmission radius, meaning that 69% of packets are expected to take 2-hop or even 3-hop paths to the sink.

Fig. 9.5. The testbed area with the five mobile nodes and the fixed sink.

The timeslot length was set to *100ms* to keep the energy consumption down and allow for the increased overhead of the testbed. This includes the additional packet size and the added delay from the CSMA/CA MAC layer. The larger timeslot size will also help to alleviate any clock drift issues, though the crystal on the IRIS motes are rated at *20ppm* drift, which should be more than adequate for these experiments.

This testbed could represent a scaled down UAV aided search and rescue application as presented in chapter 8, in which five UAVs are deployed in an area to search for people. The random mobility could be representative of the movements of a collaborative search algorithm.

The packet generation rate is based on the network being deployed in an area with a high number of targets. Having a high number of targets will cause the nodes to generate traffic at a high frequency. In terms of targets reported per minute the network data generation rates are set to *[55, 70, 100, 165, 500]* packets per minute. This translates to each node having a packet generation rate of *[0.18, 0.23, 0.33, 0.55, 1.67]* packets per second. Whilst these data rates are high for the representative application, it is possible that the nodes are deployed in a target rich environment and the nodes are repeatedly reporting multiple targets, which will increase the traffic. These high data rates will also test the limits of RASeR, especially as the GTDMA MAC layer limits the rate at which a node may transmit. The fixed transmission rate means that each node may transmit at a maximum rate of $^{1}/_{\Delta n}$, where Δ is the length of a timeslot and n is the total number of nodes. Given the parameters of this experiment, the maximum transmission rate of each node is one packet every $^{5}/_{3}$ *seconds*, which is equal to the highest packet generation rate tested. This means that when the nodes are generating packets at a rate of *1.67pk/s*, they are at saturation. So they will be producing packets at the same rate as they can transmit them, which leaves no additional bandwidth for forwarding packets from other nodes, which will create high amounts of packet loss.

The packet generation schedules for each node are precomputed using a random number generator and then loaded on to each node with the firmware.

9.7 Testbed Results

To evaluate the performance of the protocol five metrics will be used, as described in chapter 4, namely PDR, average end-to-end delay, overhead, throughput and average energy consumption.

PDR is defined as the ratio of packet received at the sink to the total number of packets created. Since the packet generation schedules are precomputed, the number of packets generated over the run time is already known and the number of packets delivered is recorded at the sink.

Average end-to-end delay is defined as the average time taken between a packet being created and it subsequently being received at the sink. So to measure this, the sink records the time at which it receives the first instance of a packet and since the packet generation schedule is known, the delay can be calculated. Then a mean average of these measurements is taken.

Overhead generally consists of two major parts; control overhead and packet overhead. Control overhead is measured as the amount of non-data communications that take place in the network. Similarly, packet overhead is measured as the amount of a packet that isn't data. In this experiment both are measured simultaneously with an overall measure of overhead, which

is the ratio of bits transmitted to the number of data bits successfully delivered to the sink. Since the range limiting technique chosen was to apply a RSSI threshold, the sink can eavesdrop on the communications over the entire network. As such it records the total number of bits transmitted by all nodes. This value is then used with the number of received data bits at the sink, to give the overhead metric. It should also be noted that to focus on the analysis of the routing protocol only, the additional bits transmitted for analysis purposes are omitted from the metric.

In this work throughput is defined as the number of data bits successfully delivered to the sink per second. This is a simple time average of the number of data bits recorded as received by the sink.

To give a protocol specific measure of energy used, this metric focuses on the energy used by RASeR communication only. This will allow the energy consumption of the protocol to be considered aside from things such as the mobility platform, any sensor inputs or any localisation techniques, which are all changeable depending on the hardware or application in which the protocol is deployed. Also additional bits are included in packets for analysis purposes, which are omitted from this metric. The metric uses the characteristics of the transceiver to calculate the energy consumption from the number of bits transmitted and the number of bits received.

All of the metrics were calculated offline, by running a MATLAB program using the data log generated by the sink node.

The gathered results are shown in figures 9.6(a-e), alongside simulated and analytic results for the same scenario. The simulation results were performed in OPNET [4.9] and designed to mimic the environment of the experiment. The analytic results were calculated from the expressions given in chapter 5, however it should be noted that the analytical expressions were derived under the assumption that the nodes were deployed in a well-connected network. However, the low density and number of nodes in this experiment would not be considered well-connected, so it is not expected that the expressions yield accurate results, although they have been included for completeness.

The PDR results in figure 9.6(a), highlight the protocols reliability even up to the high packet generation rate of 0.33pk/s, which is a network-wide packet generation rate of 100 packets per minute. At 0.55pk/s there is a slight deterioration in performance, however the PDR remains higher than 91.52%. Even when the network reaches saturation the protocol still manages to deliver over 64.24% of the generated packets, which is made possible by the protocols low overhead. The simulated results follow the trend of the experimental results closely; however they overestimate the number of packets lost. As expected, the analytical

results are not very accurate and predict a more constant PDR over the entire range of packet generation rates, with a decrease from 91.92% to 91.37%.

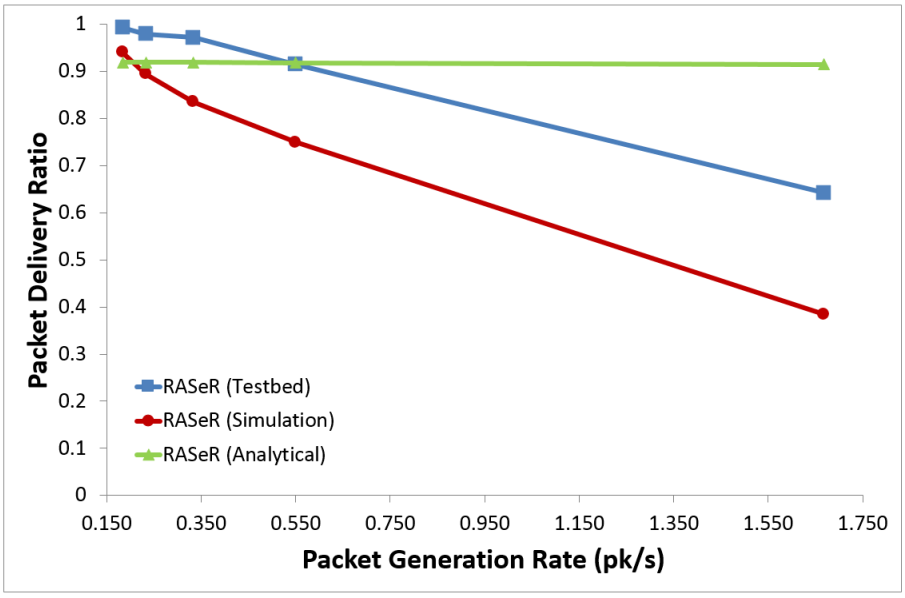


Fig. 9.6(a). Testbed, simulation and analytical PDR results for RASeR, over varying packet generation rates: *[0.18, 0.23, 0.33, 0.55, 1.67]pk/s*.

Figure 9.6(b) gives the end-to-end delay results, which are very low considering the long slot time of 0.1s. Initially the delay increases with the packet generation rate and peaks at 423.07ms. This increase is due to the increased traffic causing queuing delays. Then, when the PDR begins to drop further, the delay decreases since more packets are being lost and therefore the packets that are being delivered, can be delivered faster. Similarly to the PDR results, the simulation predicts worse delay times than are measured in the experiment. Also, the analytical results, again predict a more constant level of end-to-end delay.

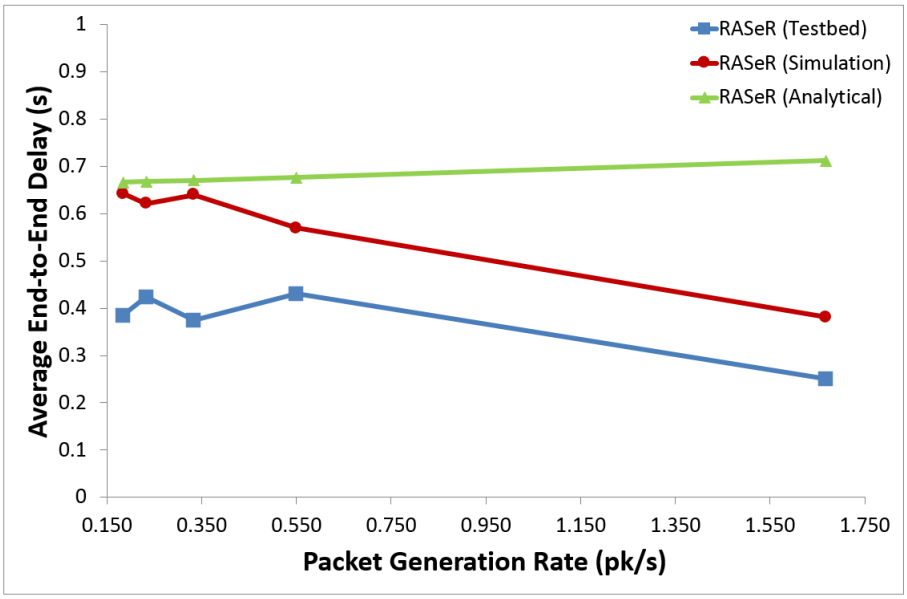


Fig. 9.6(b). Testbed, simulation and analytical end-to-end delay results for RASeR, over varying packet generation rates: *[0.18, 0.23, 0.33, 0.55, 1.67]pk/s*.

RASeR's very low overhead is illustrated in figure 9.6(c), which shows all three curves following a very similar trend. The simulated results still tend to overestimate the measured results, but the analytical expression gives a much closer representation. The overhead decreases as the packet generation rate increases, which is due to the decreasing number of redundant transmissions. The multipath nature of RASeR means that a packet may be delivered to the sink multiple times, which generally improves the protocols reliability, but also increases the overhead. However, in the saturated case, the nodes don't often have time to service multiple packets, so there is significantly less redundancy and as such the overhead is reduced.

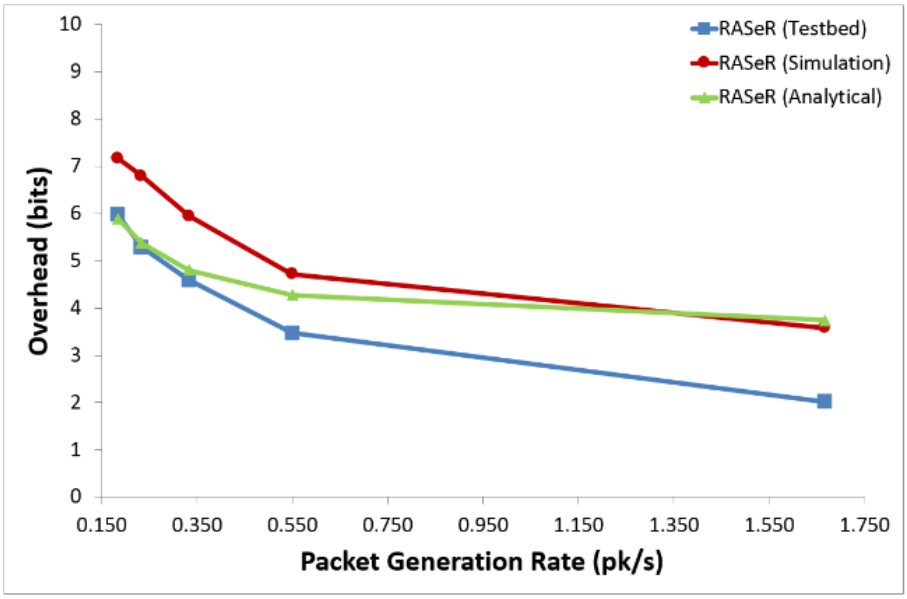


Fig. 9.6(c). Testbed, simulation and analytical overhead results for RASeR, over varying packet generation rates: *[0.18, 0.23, 0.33, 0.55, 1.67]pk/s*.

As one would expect, figure 9.6(d) shows how the throughput increases as the packet generation rate increases. This is because, at the highest network-wide rate of 8.33pk/s, even with a PDR of only 64.24%, a greater number of packets are delivered to the sink than at the lower rate 0.92pk/s. The analytical expression predicts a very linear increase, however the simulation and experimental results show a decreased gradient towards the higher packet generation rates, which is expected as the network reaches its saturation point.

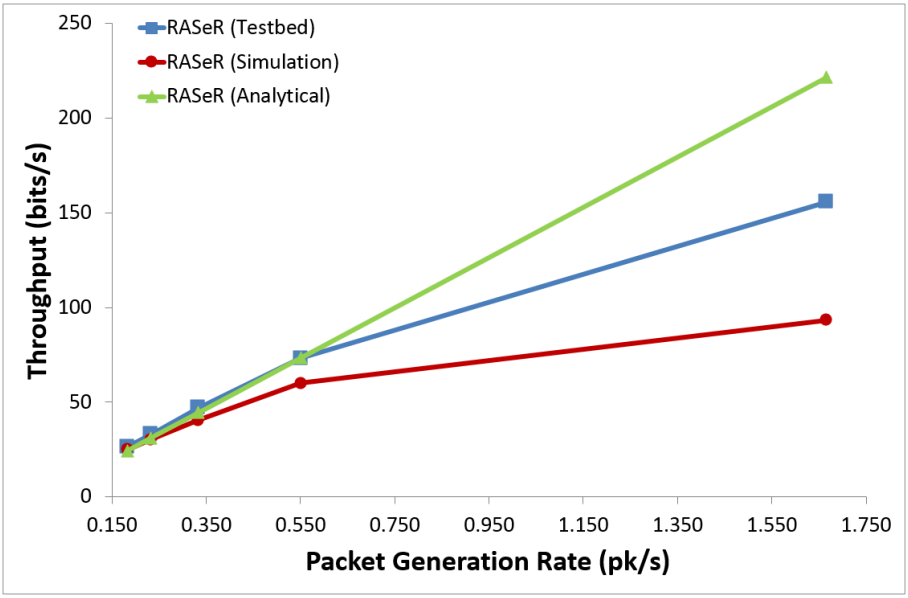


Fig. 9.6(d). Testbed, simulation and analytical throughput results for RASeR, over varying packet generation rates: *[0.18, 0.23, 0.33, 0.55, 1.67]pk/s*.

In a similar way to the throughput, the analytical results for energy consumption, in figure 9.6(e), show a linear increase in power usage. The experimental and simulated results show a low energy consumption, which remains fairly constant over the range of packet generation rates. There is a slight increase toward the higher traffic level where nodes are transmitting more data packets and so more energy is used.

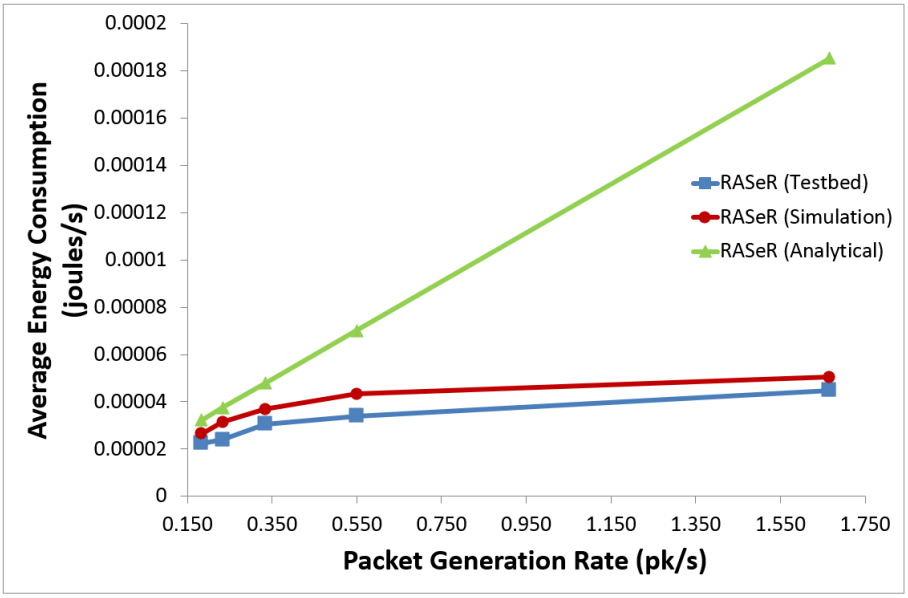


Fig. 9.6(e). Testbed, simulation and analytical energy consumption results for RASeR, over varying packet generation rates: *[0.18, 0.23, 0.33, 0.55, 1.67]pk/s*.

In general the experimental results give better results than those predicted by simulation. The analytical results are not particularly accurate, due to the assumption of a well-connected network, however in many cases they capture the general trend. Overall, the results suggest that RASeR has high reliability in high traffic conditions, though its performance drops at the network approaches saturation. The delay is consistently low and could be made lower still by decreasing the long slot time. The overhead and energy consumption are also low and the throughput shows a steady increase with the packet generation rate, as expected.

9.8 Conclusion

This chapter has demonstrated that it is possible to implement RASeR on commercially available hardware and also illustrates the high performance capabilities of the protocol by showing high PDR, low delay and overhead in high traffic scenarios. The protocol only begins to deteriorate as the traffic levels approach the saturation point of the network. Additionally, RASeR has been shown to perform better in a testbed scenario than was predicted by simulation, though the simulation results are still relatively accurate. The implications of these results are in the fact that RASeR has been proven to successfully function in a physical testbed and could therefore be deployed in a real-world scenario. One critical point of concern in the

implementation was the time synchronisation, however the protocol did not experience any issues with nodes becoming unsynchronised. The most optimal deployment of the protocol would be on dedicated hardware using a private frequency band, such that the slot time could be reduced and the results presented in this chapter can be seen as a preliminary study towards this goal. However, this work has shown that RASeR can be deployed on top of an existing CSMA/CA MAC layer and still be functional, which suggests that it could be implemented cheaply on COTS hardware, or even in existing systems.

CHAPTER 10

SECURE ROBUST AD-HOC SENSOR ROUTING (RASER-S)

10.1 INTRODUCTION

Security is a major concern in any system and wireless networks are particularly susceptible to attack due to the nature of the shared medium and the fact that devices are often expected to be able to join the network in an ad hoc way. Many applications have security requirements, these include all military and surveillance applications. Another area in which security is of high importance is in healthcare and the emergency services; if the network is compromised and packets are either delayed or destroyed, there can be life-critical repercussions. Additionally, the privacy of data in nearly all networks is considered important, especially where sensitive or personal information is being transported. For these reasons this chapter firstly identifies the requirements of a secure system before describing the proposed security framework. This framework is then evaluated against known attacks and simulations are run to determine the extent of the frameworks effects on the performance of the underlying routing protocol. RASeR is chosen as the underlying protocol due to its wide applicability and the addition of security would make it a well-rounded solution, which can be adapted to suit the needs of emerging applications.

10.2 REQUIREMENTS

Research into information security has highlighted three core requirements for secure communication [10.1];

- **Confidentiality:** This is the requirement that only the intended recipient of a message is able to read it. In a MWSN this is usually the sink, who should be the only node able to read the data being sent from the sensor nodes.
- **Integrity:** This is the requirement that the received message is the same as the transmitted message. This is especially important in a multihop scenario in which a message may be passed through several nodes before reaching its destination. So in a MWSN it should be ensured that changes to the message are reliably detectable.
- **Availability:** This is the requirement that the information is available when it is needed, so the system should be resilient to attacks. In MWSNs, attacks such as

denial of service (DoS) could potentially render the information in the network inaccessible.

Many additional requirements for security have also been identified and are applicable in certain applications [10.1];

- **Non-repudiation:** This is the requirement that once a transmission has been made, the sender is unable to deny that he was the one who sent it. In MWSNs, this means that malicious nodes can be identified and they will be unable to deny the sending of corrupt packets.
- **Authentication:** This is the requirement that nodes are able to verify the identity of the other nodes that they are communicating with. Since MWSNs use a shared medium, authentication will prevent injected packets from foreign nodes being accepted and acted upon by the network.
- **Authorisation:** This is the requirement that only legitimate nodes are permitted to access certain resources. In MWSNs, these resources may be the sensors themselves, in which case authorisation will prevent a malicious node demanding sensed data from the network and having its request fulfilled.
- **Freshness:** This is the requirement that the age of a packet can be determined. Since not all MWSN applications will require data to be time stamped, knowing the age of a packet will prevent old packets being re-injected into the network by an adversary at a later time.
- **Semantic security:** This requires that plaintext, which is encrypted more than once, will not give the same ciphertext. In MWSNs, it is quite common for nodes to generate packets that are the same as previously transmitted packets, especially in the case of control messages. So it is important not to reveal anything about the information being transmitted by ensuring semantic security.
- **Anonymity:** This is the requirement that the identity of the participants in the communication, remains secret. In MWSNs, some nodes are given different roles, which makes some nodes more valuable. These valuable nodes, such as the sink, may become targets for those wishing to disrupt the network and should therefore be kept hidden. Additionally, some applications have nodes that are associated with specific people or groups, in which case the fact that certain information is originating with them may need to be obfuscated.

Some of the requirements listed here are critical in any system and many apply directly to MWSNs. With this in mind, the next section will detail the specifics of RASeR-S.

10.3 Protocol Description

RASeR has some features that naturally assist with keeping the network secure, such as multipath routing. The fact that a packet may take multiple different paths to the sink simultaneously, means that if one copy of the message is intercepted by an adversary, it is likely that another copy will still make it to the sink. Additionally, the use of GTDMA makes it very hard for an adversary to simply inject packets into the network as they wish. Since every timeslot is reserved for a specific node, they would have to listen to the nodes around them, find a gap in the schedule caused by a node being out of range. Then, on the next cycle, they may transmit in this timeslot with less risk of causing a collision. The assumption that a fixed number of nodes is used, means that there is no mechanism by which a malicious node may join the network.

RASeR-S (Secure Robust Ad-hoc Sensor Routing) is based on RASeR, however some major changes are implemented. One of which is the packet structure; the packet structure for RASeR-S is shown in figure 10.1. The majority of the packet fields are the same as RASeR and are used in the same way. RASeR-S firstly introduces a beacon bit, which is used to indicate whether a packet is a beacon or a data packet. Cyclic redundancy checks (CRCs) are also added to both the data and the whole packet, which will ensure that the packet is received correctly and hasn't been changed in any way.

Field name	Data	CRC
Size (bits)	L_{data}	8

Encrypted with:	Individual Key
	Network Key

Field name	Beacon Bit	Node ID	Hop Count	Encrypted Data	Priority Bit	Packet ID	Packet CRC
Size (bits)	1	$\lceil \log_2 n \rceil$	$\lceil \log_2 n \rceil$	L_{data}+8	1	$\lceil \log_2 n \rceil$	8

Total size (bits)	$L_P = 3 \cdot \lceil \log_2 n \rceil + L_{data} + 18$

Fig. 10.1. RASeR-S packet structure, total length and the corresponding encryption keys for each section.

Without the beacon bit, data packets are identified by the length of the transmission. As such the addition of the beacon bit will prevent an adversary from transmitting data directly after a beacon packet, in order to make receiving nodes think it is a corrupted data packet. An adversary could also decide to perform a traffic analysis, which will look at when data packets are sent and where from. In some applications, this information may be sensitive, such as in military reconnaissance, where an adversary will be listening for an original broadcast of a data packet. If one is heard, then the adversary will not only know that a target has been found, but also where the target is located. To avoid this, decoy data packets may be randomly generated.

These will be the same length as data packets, however the beacon bit will be set, indicating that it is in fact a beacon packet.

10.3.1 Dual Keying

In general the limited computation capabilities of sensor nodes restrict them to symmetric key ciphers. RASeR-S uses the block cipher RC5 [10.2], mainly because it has a variable block size, the smallest of which is 32 bits. This small block size ensures that a minimal amount of space is wasted when a packet doesn't divide perfectly into the chosen block size. Additionally, RC5 has no known vulnerabilities and has been implemented successfully in the literature [3.69].

Dual keying is used in RASeR-S, such that each node is programmed with two keys, as shown in figure 10.1. The first key is the nodes individual key, $Key_{Individual}$, and is used for encrypting the data. The second is the network key, $Key_{Network}$, which is used for encrypting the entire packet. The sink has knowledge of all the nodes individual keys as well as the network key. In this way, the data generated by each node is encrypted such that it cannot be decrypted by intermediate nodes, essentially creating a secure link between each node and the sink. Each packet is also encrypted by the network key, which is known by all of the authorised nodes in the network. This allows the routing information to remain secure from outsiders, as well as preventing malicious nodes inside the network from accessing data from other nodes, whilst still maintaining the functionality of the routing protocol. The advantage of this dual keying, is that if an adversary discovers the network key, they will still not be able to decrypt the data contained within the packets. Additionally, if an adversary compromises a node and discovers the network key and that nodes individual key, they will still be unable to decrypt the data from every other node in the network.

Since a packet may be longer that the block size of the cipher, a block cipher mode will be required. There are six commonly used modes; electronic codebook (ECB), cipher block chaining (CBC), propagating cipher block chaining (PCBC), cipher feedback (CFB), output feedback (OFB) and counter (CTR). Aside from the simplicity of some modes, one distinct advantage to CFB, OFB and CTR, is the fact that they only require a one way cipher implementation; in other words the cipher's encryption algorithm can be used for both the encoding of the plaintext into ciphertext and the decoding of the ciphertext back into the original plaintext. Given the program memory limitations on sensor nodes, it would be advantageous to select a mode which possesses this quality.

Another consideration is the use of the CRC to verify the contents of a packet. CRCs are often a preferred choice because of their simplicity. However, as they are based on a repeated XOR operation and some modes output ciphertext by simply XORing the plaintext with a

generated string, it is possible that the message could be altered in such a way as to create a new valid message. Since the order of the packet fields remains unchanged, some bits in the message section could be flipped and then the appropriate bits in the CRC section could also be flipped to create a packet that would be accepted by any valid node in the network. This is a serious security risk as the changed packet could cause the network to behave unpredictably and potentially allow for information to be leaked.

In order to counter act this, the CRC should be put in the first block of a packet and a mode should be chosen in which the decryption of one block is dependent on the previous block. In this way, if the CRC section of the ciphertext is altered in any way, then the subsequent block will be decrypted incorrectly, causing the CRC to be incorrect also. This property is present in the CBC, PCBC and CFB modes. As an example, in the CFB mode shown in figure 10.2, the first ciphertext block is both used for decoding the first block of plaintext, as well as being fed into the cipher to enable the decryption of the second block of ciphertext. Therefore, if an adversary adjusts the CRC section in the first block of ciphertext, it will affect the decryption of the second block. Due to the diffusion property of the cipher used, these changes are somewhat unpredictable and will subsequently cause the packets CRC to become invalid.

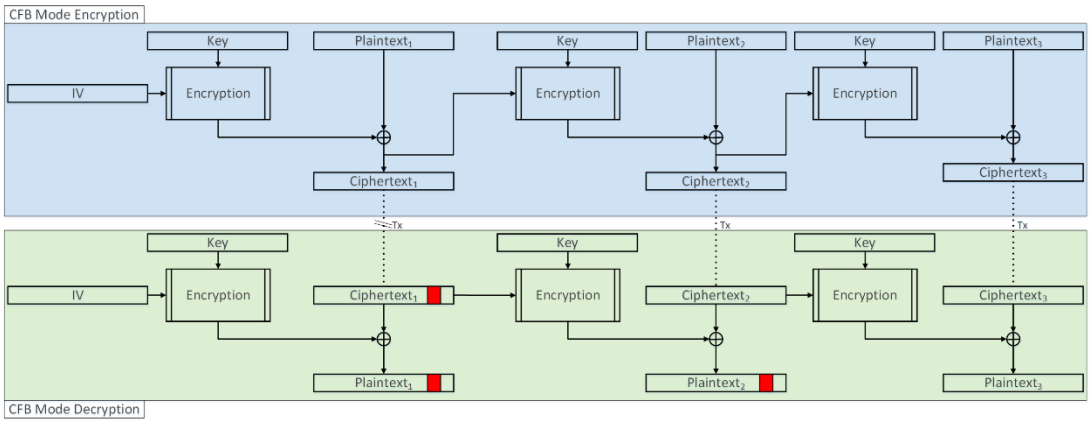


Fig. 10.2. This figure shows the flow chart of the CFB mode for both encryption and decryption. The IV is the initialisation vector. The figure also shows how an error in the transmission of the first block of ciphertext causes errors in both the first and second block of plaintext.

As, CFB is the only mode in which errors in one ciphertext block affect the decoding of the next block and the ciphers encryption algorithm can be used for both the encoding and decoding process, this mode will be used in RASeR-S.

CFB also requires an initialisation vector (IV) to begin the encryption and decryption process. In the proposed protocol the IV is made up of a N_{once} concatenated with a counter. The N_{once} is a random string that will be chosen and distributed along with the encryption keys. When encrypting data with a nodes individual key, the counter used will be the packet ID, which increments every time a node generates a packet with new data. When encrypting the whole packet with the network key, the counter will be the global timeslot number. The global

timeslot number begins at zero when the network is first deployed and then increments at every timeslot. This requires that the nodes are synchronised, such that they know the current timeslot number in order to correctly decrypt packets received in that slot. Using these counters will provide weak freshness and semantic security, as well as preventing malicious nodes from overhearing transmissions and replaying them at a later time.

Figure 10.3 gives a full flow chart of the encryption and decryption process in the source, sink and intermediate nodes using the CFB mode and RC5 cipher. The initial stage shows the source node that has some data to transmit. Firstly, a CRC is added to the data, which is encrypted with the source nodes individual key, using the N_{once} and the packet ID as the IV. Since this will be the next packet to be sent by this node, the packet ID can be generated and the encryption can be done as soon as the data is ready. Then, just before the node has access to the medium it can construct the rest of the packet, by assembling all the fields in figure 10.1, including the nodes ID, hop count, packet ID and a generated CRC. After the packet is compiled with the encoded data, it is encrypted with the network key and uses the N_{once} and the current global slot number as the IV. This is then broadcast to the nodes neighbours, who each decide whether they should forward the data or not.

The second part of the diagram shows an intermediate node decrypting the routing overhead using the network key and the current global slot number. This allows the node to decide to store the information in its queue, along with the encrypted data. Then the third part shows how the intermediate node prepares a packet for transmission. It first populates the packet fields with the source node ID, the transmitting nodes hop count, the packet ID and encrypted data from the queue, and finally a generated CRC. This is then encrypted using the network key and the current global slot number before the node transmits the packet.

Once the packet has been received at the sink, it can firstly be decoded to extract the routing information using the network key and the current global slot number. Then the decrypted packet ID can be used with the origin nodes individual key, in order to decrypt the data.

The diagram assumes that the data and CRC take two blocks to encode and the routing overhead is only a single block, to mimic the parameters used in the simulations. However, this idea can be applied to any number of blocks, so it will work with any size of data, CRC, routing overhead and any block size.

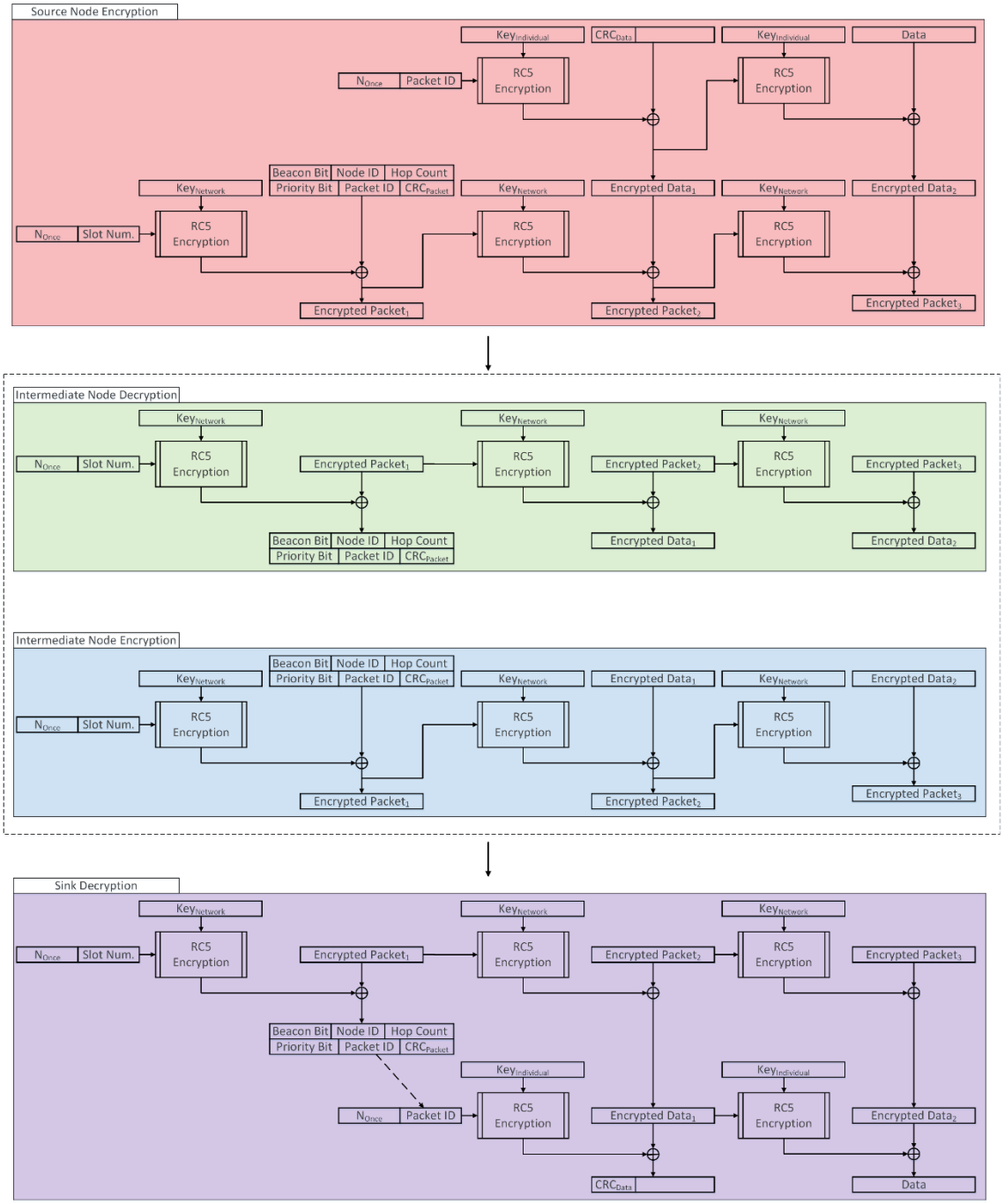


Fig. 10.3. Dual keying and cipher structure flow chart.

To give the nodes enough time to perform encryption and decryption of packets, the time slot length will have to be increased; figure 10.4 shows the structure of a timeslot. Since the encryption and decryption of data doesn't have to be done during a timeslot, this will not affect the slot length. Sensor nodes may encrypt their sensed data when there is available processor time, for example in a timeslot in which they are not receiving any data. The sink is expected to be more powerful than the sensor nodes, so it will not have any problem decrypting the data whilst performing other tasks.

	Slot Structure																			
T_x	Wakeup	→	Compile Packet	→	Encrypt Packet	→	Transmit	→	Sleep	→	→	→	→							
R_x	Sleep	→	→	→	Wakeup	→	Receive	→	Decrypt Packet	→	Store Packet	→	Sleep							

Fig. 10.4. Slot structure for a transmitting node (T_x) and a receiving node (R_x).

10.3.2 R4 Framework

As has been highlighted in chapter 5, one of the key differences between a static WSN and a MWSN is the addition of a mobility platform. In applications such as UAV aided search and rescue, military reconnaissance or environment mapping, the mobility platform may be an unmanned autonomous vehicle, be it terrestrial, aquatic or aerial. These nodes will be considerably more expensive that the traditional disposable sensor nodes used in static sensor networks. Due to the increased price of these precision manufactured, state-of-the-art sensor platforms, they will be expected to be reusable for multiple missions. It is therefore anticipated that fewer nodes will be used and their power capacity will be sufficient to complete the mission and return to the base station.

In cryptography, one of the main issues is key distribution, which is the problem of getting known private keys to both parties in a secure manner. RASeR-S takes advantage of the fact that these MWSN nodes are rechargeable and will need to regularly return to base when the mission is complete, or in order to recharge. It is assumed that the recharge station is a secure facility in which the nodes will visit to recharge during a mission, be stored between missions and be reprogrammed and redeployed for their next mission. RASeR-S utilises this by introducing the Recharge, Reprogram, Rekey and Redeploy (R4) framework. The R4 framework may be used in any MWSN that provides a secure base station facility. The framework solves the key distribution problem by taking advantage of the fact that all nodes will have to return the base station periodically at the end of a mission or to recharge. As such, when nodes are programmed before a mission, they are also provided with a new individual key and a new network key.

Additionally, every time a node returns to the secure base station it is synchronised and assessed for tampering. Due to the fact that the sensor nodes may go for long periods of being unattended, it is possible that they may be captured and without tamperproof hardware they could be hacked. Since tamperproof hardware is expensive, an alternative is to erase and reprogram the nodes every time they return to the base station to eliminate any malicious software that may have been downloaded on to the device. Additionally, it is possible that someone has attached a device to the node, which may do something as simple as read values from a pin and transmit them to an adversary. It therefore may be good practise to weigh the nodes each time they return, such that any anomalous changes in the nodes mass can be detected and further investigated. These precautions can potentially limit the damage caused by a compromised node. Further protection may be provided if the nodes are all expected to return mid-mission to recharge, they can be rekeyed at this point also, which may be advantageous in long missions or in highly sensitive applications. Another advantage of rekeying the nodes regularly is that, if a key is discovered, it will only be useful for the length of that mission, or until the nodes are rekeyed again. The regular rekeying of the network also means that adversaries have a very limited time in which to attempt a brute force attack.

In conclusion RASeR-S uses the RC5 cipher in CFB mode to maintain the confidentiality of the information. The integrity of the information is verified through the use of a CRC and the availability of the network is aided by the inherent multipath routing of the protocol. Since the protocol doesn't allow a node to join a deployed network, the knowledge of the network key is used as authorisation to allow a node to participate in the network. Weak freshness and semantic security are provided through the use of the global slot number and the packet ID in the encryption process. Additionally, some anonymity as to the sources of data may be given through the transmission of decoy data packets. Also, the use of the R4 framework solves the key distribution problem, which negates the need for complex, time consuming key agreement schemes.

10.4 Threat Analysis

In any kind of network, security threats can come from an innumerable number of sources, which makes it impossible to account for every different possible attack. However, this section will analyse the protocol based on some well-known and some less well-known attacks [10.3]. These attacks will be categorised as passive or active and internal or external. Passive attacks are those in which the adversary is not participating in the network. This means that they are not transmitting any kind of data and, in a wireless network, as such they can remain completely undetected. Active attacks require an adversary to transmit something, usually to try and engage with the network in some way. These attacks are often more powerful, however they do have

the added risk of the adversaries activities being discovered by the network. External attacks are those that originate from sources outside of the network, whereas internal attacks come from nodes that have become associated with the network.

10.4.1 Passive External Attacks

Passive external attacks are often based around eavesdropping, which in a wireless medium is very easy. However, the basic addition of a cipher means that the transmissions that are overheard, cannot be understood. Without knowledge of the semantic content of the packets, an adversary may still ascertain information such as the existence of the node and its location. Another major attack is traffic analysis, where an adversary will take note of the length and frequency of transmissions as well as from which node they are originating. From this they may discern things such as the location of a sensed event or which nodes are most active in the network. Then, depending on the networks application, they may be able to determine the identity of certain nodes or which nodes would be the best targets to destroy in order to disrupt the network. By allowing nodes to randomly transmit decoy data packets, RASeR-S add noise to the traffic analysis results, making it very difficult to extract any information. Additionally, the use of decoy data packets could be used as misdirection. In this way, a less important node could create a large quantity of decoy packets, which would then draw an adversary's attention away from more important activity occurring in other nodes.

10.4.2 Active External Attacks

There are two main active external attacks; the replay attack and DoS through jamming. The replay attack involves an adversary listening to a legitimate transmission and simply repeating what is has heard at a later time. RASeR-S prevents this by using the global time slot number as part of the IV, which means that if the packet is replayed in a later timeslot it will be decrypted incorrectly and discarded. A variation on this is the wormhole attack, in which a high-speed communications link is used to take in a packet from one area of the network and simultaneously output it somewhere else in the network. In this instance the robustness of the underlying routing protocol means that the receiving nodes will simply accept this packet. So, if the receiver closer to the sink than the original transmitting node, then they will forward the packet, otherwise they will drop it. However, if the wormhole takes sink packets and replays them in the same timeslot elsewhere in the network a sinkhole is created. The nodes that hear the replayed message will assume that they are one hop away from the sink and this information will propagate out causing some nodes to forward data towards the end of the wormhole. This attack is unlikely since it requires an adversary to be able to listen to a packet transport is to another location and then replay it, within the space of a single timeslot. This will require very expensive hardware and there is also high energy cost associated with this process. Also, the

mobility of the nodes makes it unlikely that the same nodes will always be affected and makes it difficult for the adversary to maintain the attack.

Another important active external attack is DoS from jamming. This is where an adversary will broadcast high power transmissions that overlap with the radio frequency used in the target network. This has the effect of causing the legitimate transmissions from the network to be drowned out by the noise of the jammer, which will prevent the nodes from communicating. There is relatively little that can be done in the networking and medium access layers to avoid this, however on advantage of RASeR-S is the use of multipath routing. The fact that multiple copies of a packet may simultaneously take different routes to the sink, means that if one packet is disrupted by jamming, it is likely that another copy of the packet will still make it to the sink. Additionally, as RASeR was originally designed for use in highly dynamic environments, RASeR-S is able to quickly adapt to localised jamming, by routing data around these dead-zones.

10.4.3 Gaining Network Access

Internal attacks occur from inside the network and so to perform any of these attacks an adversary must first gain access to the network. There are two primary ways that this may be done; the first is through the discovery of the network key. This could be done by brute force, which could take a very long time if the key is large enough, by which point RASeR-S may well have already changed all the keys in the network. Alternatively, it is conceivable that the key could be obtained by capturing a node, forcing a memory dump and then locating the key in the output before releasing the node again. However, this can be prevented by choosing secure hardware and encrypting the way in which the keys are stored in memory. An adversary may also attach some hardware to the node, which can read a pin on the node and transmit it wirelessly. This device would be designed to capture the key exchange between the secure base station and the node. The leakage of information in this way can be prevented by weighing the nodes upon arrival to the base station and alerting someone to any changes in their mass, which could then be followed by a manual inspection. Additionally, node compromise may not just aim to reveal the network key, but to adjust or rewrite the nodes firmware. This could lead to a malicious node acting inside the network and could leak sensitive data, information about the network and its structure or launch a DoS attack by flooding the network with data packets at a high rate. Again, this could be prevented by the use of secure tamperproof hardware. However, it is important to note that with all internal attacks, the damage caused is strictly limited to a single mission because of the R4 framework. The regular refreshing of keys means that the old keys, that may have been compromised, will become obsolete and the uploading of malicious firmware will be erased by the reprogramming of the nodes.

10.4.4 Passive Internal Attacks

Passive internal attacks are simply based around the idea of eavesdropping, whereby an adversary with the network key may listen to the routing information and packet transfer in the network and may perform traffic analysis in the same way as described previously. However, due to the dual keying, the adversary will not have access to the data being transported by the nodes without significant further work. To ascertain a nodes individual key, the adversary must either use the time consuming brute force method, or compromise that specific node. If a node has been compromised then the adversary will still only have access to the data generated by that specific node.

10.4.5 Active Internal Attacks

There are many active internal attacks that could be thought up in order to achieve various outcomes. Since RASeR assumes a fixed number of nodes, there is no joining process, which means that many classic sensor network attacks would not be effective. These include the Sybil attack, in which an adversary pretends to be lots of different nodes trying to join the network in order to overload it. Additionally, a HELLO flood, where an adversary rapidly broadcasts HELLO messages, and acknowledgement spoofing, in which an adversary transmits false acknowledgement packets, will not work as RASeR doesn't use the appropriate packet type. Another classic attack is selective forwarding, whereby the malicious node will simply drop packets that the network is expecting it to forward. This will have no effect on RASeR because the protocol has no expectation on nodes to forward particular packets and the multipath routing will increase the chance of the packet being delivered anyway.

The simplicity of the routing protocol also means that there are very limited things you can do by altering a packet, however one potentially effective attack is the sinkhole attack. This is similar to the replay attack described above, however, instead of having to listen to the sinks transmission and then replay it, the adversary can simply impersonate the sink. This will have the effect of splitting the network, such that some nodes will forward data towards the real sink and some will forward data to the adversary. This attack could also be done with a high power transmitter, which will mean that more nodes will hear the adversary's broadcast and subsequently more nodes will forward their data towards the fake sink. This may be used in order to gather encrypted data at the fake sink, which will then be recorded and attacked with brute force offline. However, the brute forcing of keys to gain access to the encrypted data will take significant time, by which point the data will be out of date and no longer useful in most MWSN applications. Additionally, by drawing data away from the real sink for long periods of time, would appear to the base station as though some nodes had been lost, which is highly likely to arouse suspicion and potentially cause the mission to be aborted and the nodes recalled.

Overall, RASeR-S is highly secure to external attacks and the encryption combined with the R4 framework significantly reduces the chances of an adversary gaining access to the network. The R4 framework also ensures that even if an adversary does gain access to the network, the amount of damage they may cause is severely limited.

10.5 Simulation, Modelling and Results

To evaluate the effect of including encryption on the routing protocols performance, simulation results were gathered using the same parameters and routing metrics as described in chapter 4. This means that the data size generated by each node is set to *32bits* and since in RASeR-S, the RC5 block size is 32bits as well, the data and an *8bit* CRC check will have to be encrypted in two blocks. The remaining packet overhead will occupy only a single block, such that the total packet size for RASeR-S is *96bits*. Whereas, RASeR only requires a packet size of *54bits*, in all modes. As shown in figure 10.4, the slot time for RASeR-S had to be extended and was calculated based on the RC5 encryption latency results given in [3.69]. Results are given for RASeR-S and RASeR in pACK mode for comparison.

10.5.1 Mobility

In the first scenario, the maximum speed of the nodes was varied between *[0, 5, 15, 25, 50, 75, 100]m/s* and the results are given in figures 10.5(a-e). The PDR results in figure 10.5(a) show how both RASeR and RASeR-S achieve near perfect packet delivery across the entire range, with a minimum value of 99.93%. However, as expected, the added time in RASeR-S from the encryption and the longer packet size, yields a noticeable increase in end-to-end delay of 13.94ms at most, as shown in figure 10.5(b). Figure 10.5(c) shows the overhead results, and whilst the overhead requirement for encryption is greater than in RASeR, the overhead results show a lower level. This is due to the extended slot length in RASeR-S, which reduces the rate at which packets can be transmitted, resulting in a lower level of overhead when compared to RASeR. The same is true for the energy consumption results in figure 10.5(e). The throughput results in figure 10.5(d) show RASeR-S to achieve the same high performance of RASeR. The analytical results for RASeR-S under estimate the PDR and throughput, especially at high speeds. In terms of delay, the analytical consistently overestimates the end-to-end latency in all cases. Whereas, the analytical energy results initially overestimate the consumption, but become more accurate at higher speeds.

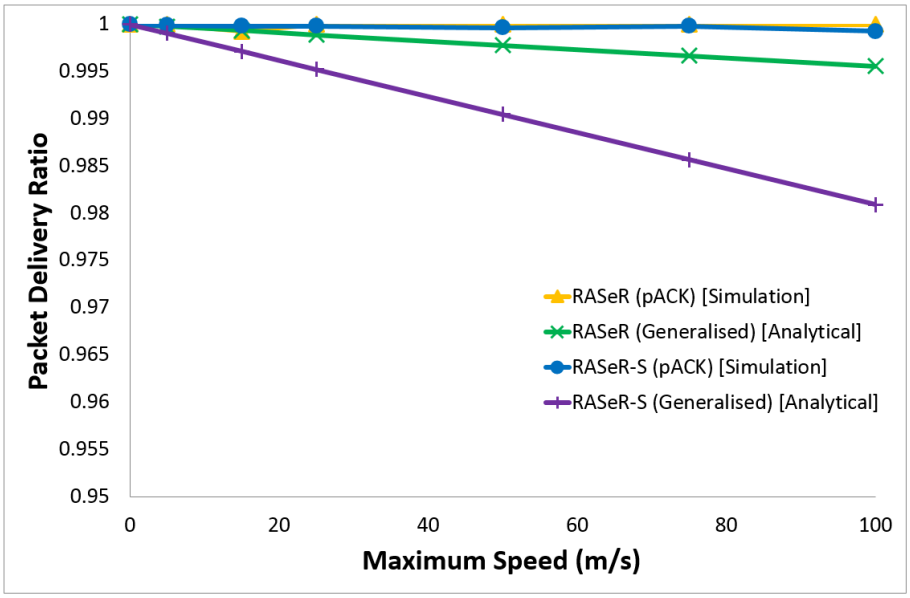


Fig. 10.5(a). Simulation and analytical PDR results for RASeR and RASeR-S in pACK mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

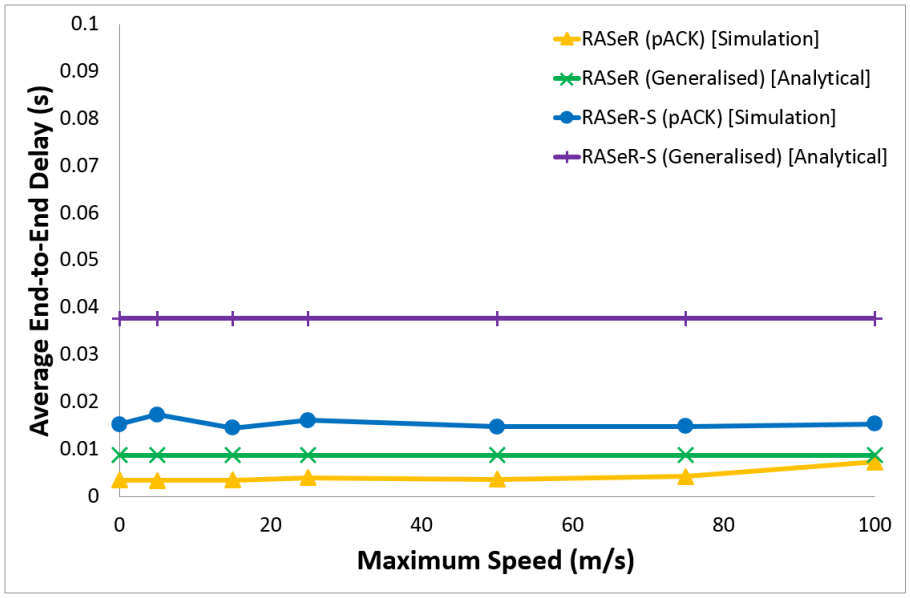


Fig. 10.5(b). Simulation and analytical end-to-end delay results for RASeR and RASeR-S in pACK mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

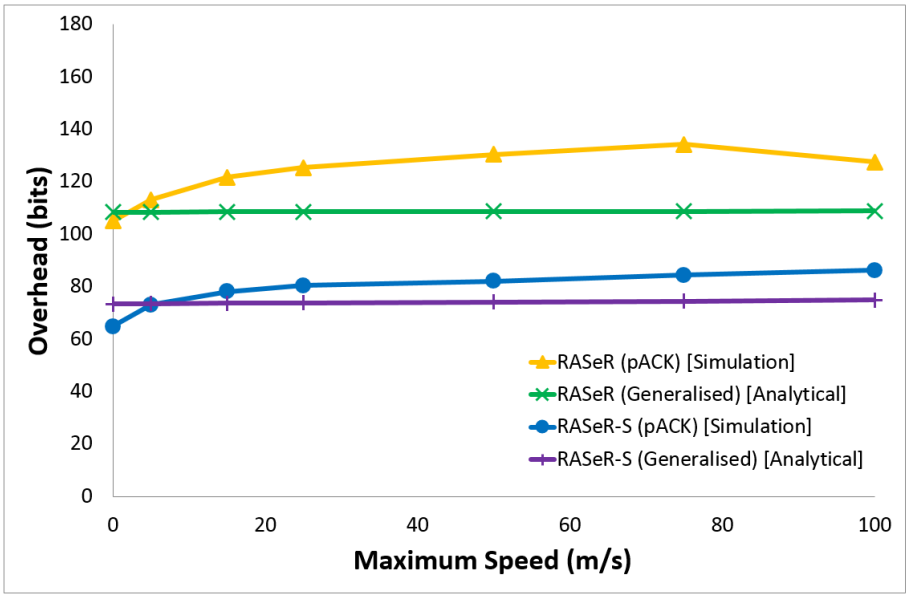


Fig. 10.5(c). Simulation and analytical overhead results for RASeR and RASeR-S in pACK mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

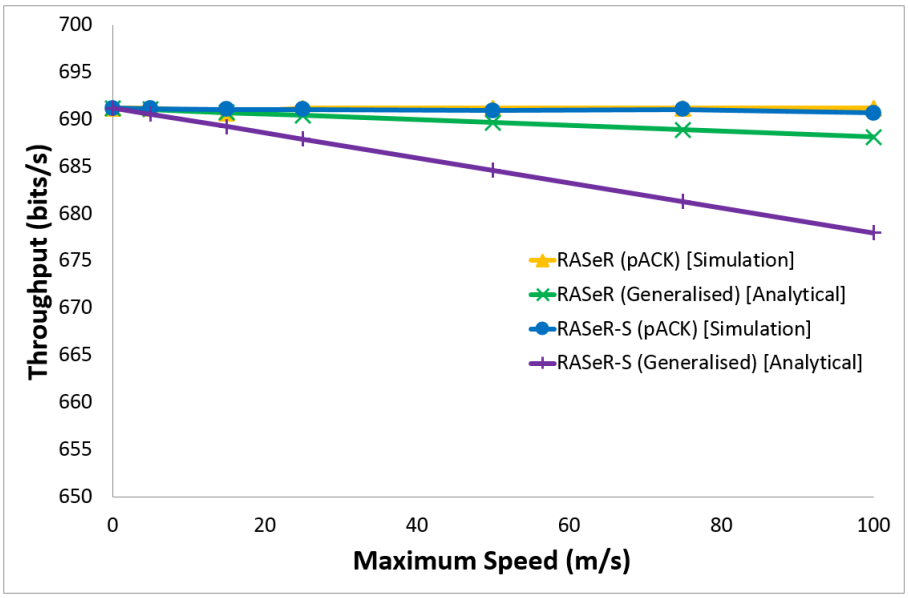


Fig. 10.5(d). Simulation and analytical throughput results for RASeR and RASeR-S in pACK mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

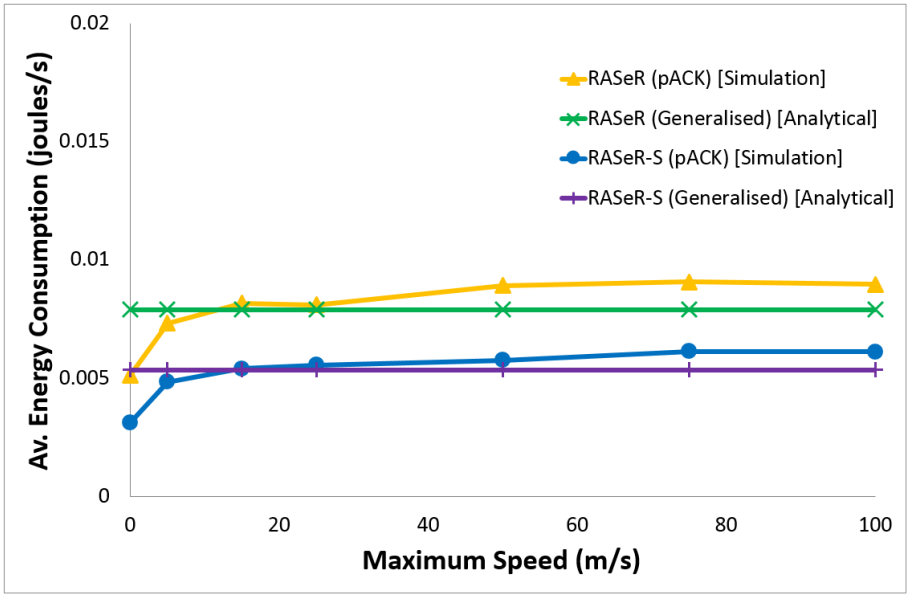


Fig. 10.5(e). Simulation and analytical energy consumption results for RASeR and RASeR-S in pACK mode, over varying maximum speeds of *[0, 5, 15, 20, 25, 50, 75, 100]m/s*.

10.5.2 Scalability

The simulation results in figures 10.6(a-e) used a variable number of nodes; *[15, 25, 50, 75, 100]nodes*. Figure 10.6(a) shows RASeR-S to have very high PDR until the number of nodes increases to 100, at which point it drops to 84.75%. This is due to the increased end-to-end delay, as shown by the delay results in figure 10.6(b). The cause of this increased delay is in the large number of nodes each requiring a timeslot, and due to the long length of each timeslot, a significant amount of delay is created between each opportunity a node gets to transmit. Essentially, congestion is created from the long cycle time and the increased level of traffic from the additional nodes. This is also reflected in the throughput results figure 10.6(d), which start well but show a decline at 100 nodes, and is closely matched by the analytical results. The overhead and energy, in figures 10.6(c) and 10.6(e) respectively, tend to decrease as the number of nodes increases. In the overhead results, this is because there is more traffic and more packets are being delivered for relatively little additional overhead, which causes the average overhead to decrease. Whereas, with the energy results, more nodes are added but the network wide energy consumption has only increased by a relatively small amount, which causes the average energy consumption to decrease. The analytical results for overhead and average energy consumption show a reasonable estimation of the simulated results.

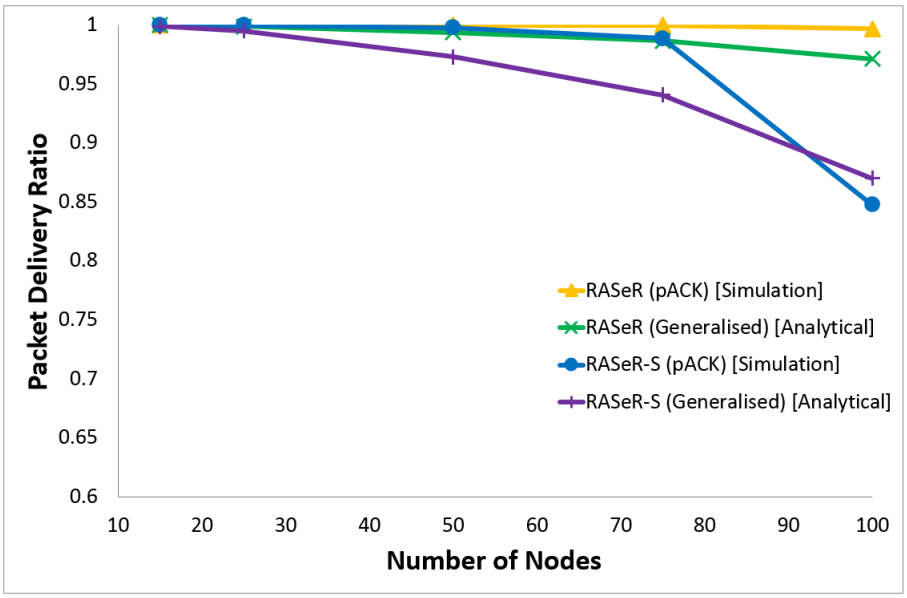


Fig. 10.6(a). Simulation and analytical PDR results for RASeR and RASeR-S in pACK mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

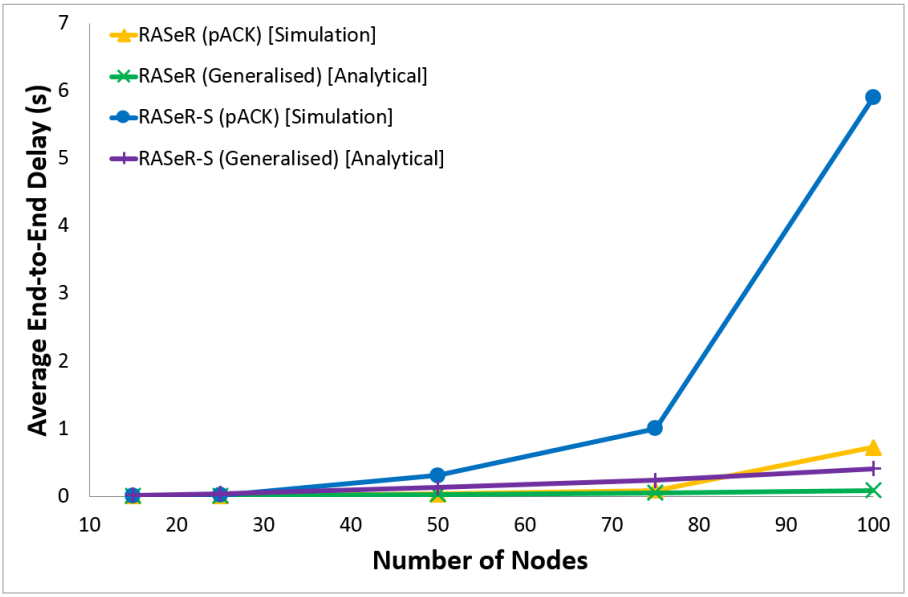


Fig. 10.6(b). Simulation and analytical end-to-end delay results for RASeR and RASeR-S in pACK mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

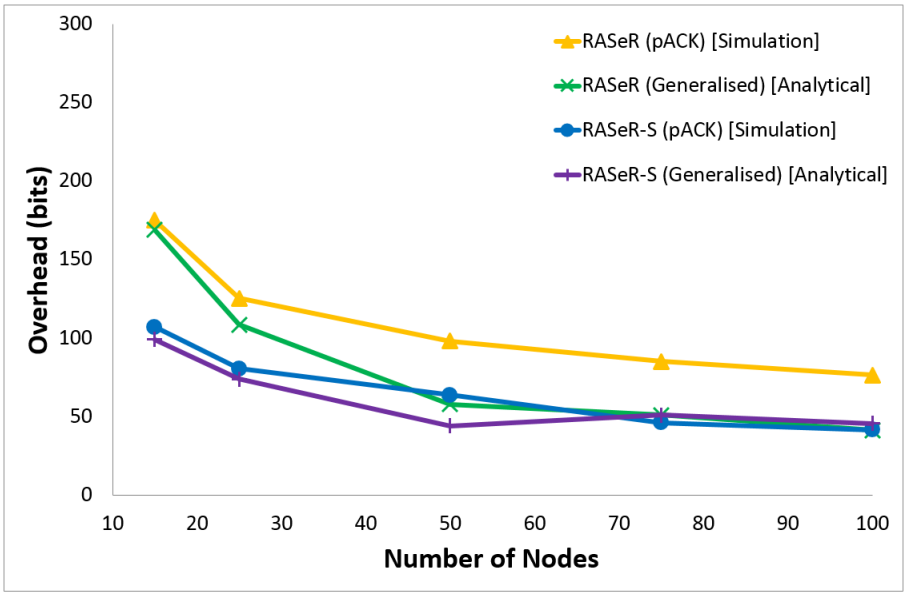


Fig. 10.6(c). Simulation and analytical overhead results for RASeR and RASeR-S in pACK mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

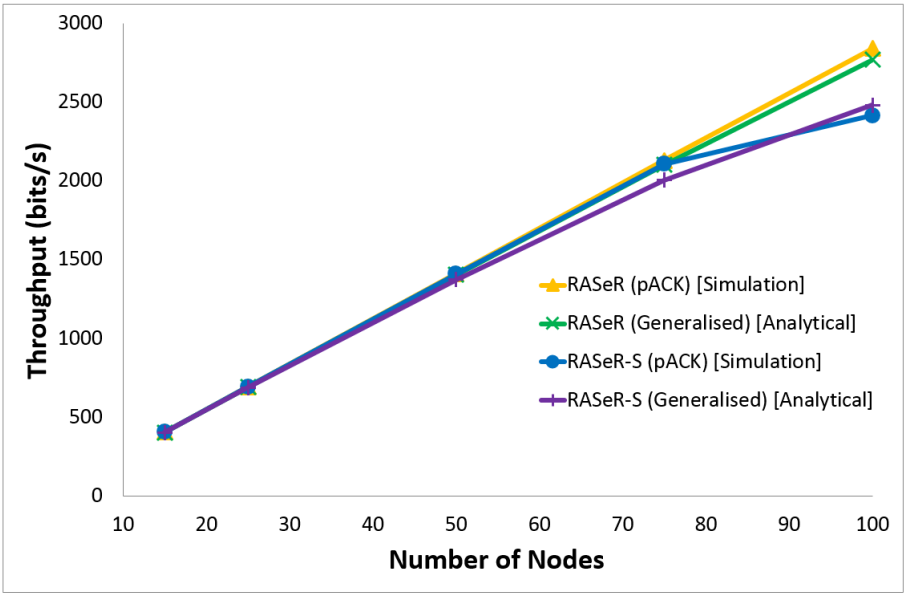


Fig. 10.6(d). Simulation and analytical throughput results for RASeR and RASeR-S in pACK mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

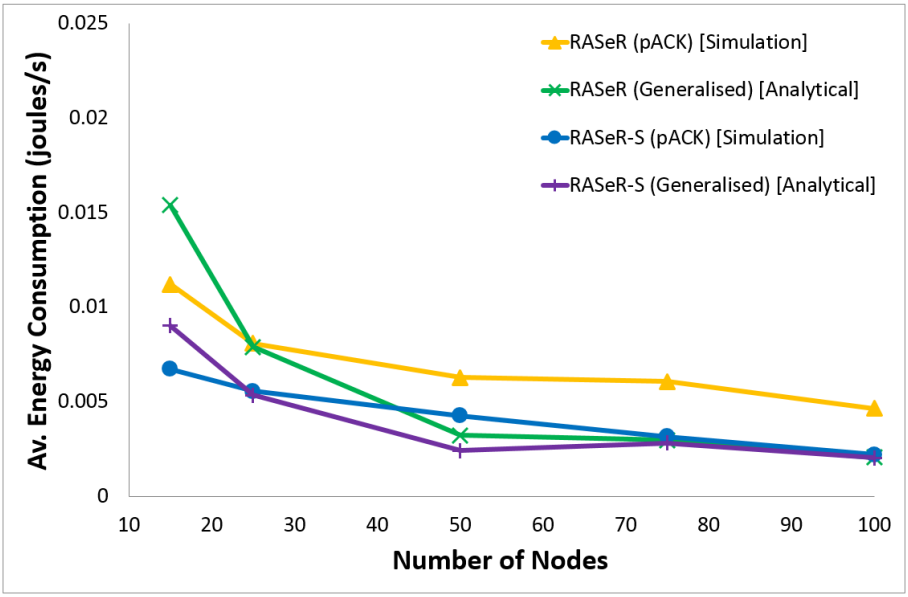


Fig. 10.6(e). Simulation and analytical energy consumption results for RASeR and RASeR-S in pACK mode, over varying numbers of nodes: *[15, 25, 50, 75, 100]nodes*.

10.5.3 Traffic

Figures 10.7(a-e) show the results for varying traffic levels, which vary between *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*. As with the scalability results, the PDR of RASeR-S is near perfect until the last scenario, where it drops to 97.25%, as can be seen in figure 10.7(a). The increased cycle time restricts the number of packets that can be serviced, so when the amount of traffic is increased high levels of congestion are created. The congestion also causes latency, which is clearly shown in the end-to-end delay results in figure 10.7(b), which increases to 82.25ms at 10pk/s. However, figure 10.7(d) shows that the throughput of RASeR-S remains high but the energy consumption in figure 10.7(e) seems to reach a limit at the 2.5pk/s mark. This is due to dedicated timeslots all being filled, which prevents anymore energy from being used. As in the scalability results, the overhead in figure 10.7(c) shows a decline as the amount of traffic is increased. The analytical results for overhead and throughput seem to be particularly accurate, whereas the PDR, delay and energy results tend to anticipate worse performance than what is shown by simulation.

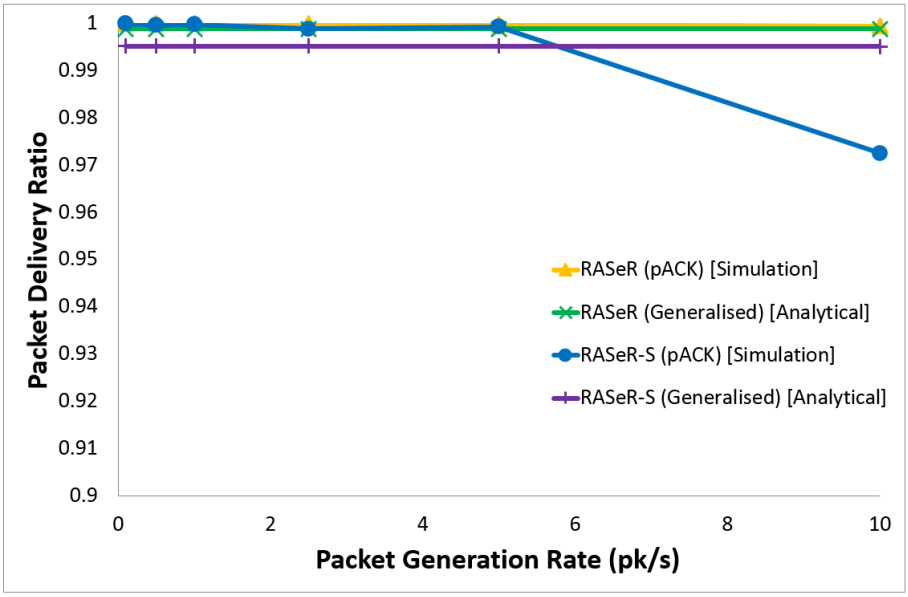


Fig. 10.7(a). Simulation and analytical PDR results for RASeR and RASeR-S in pACK mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

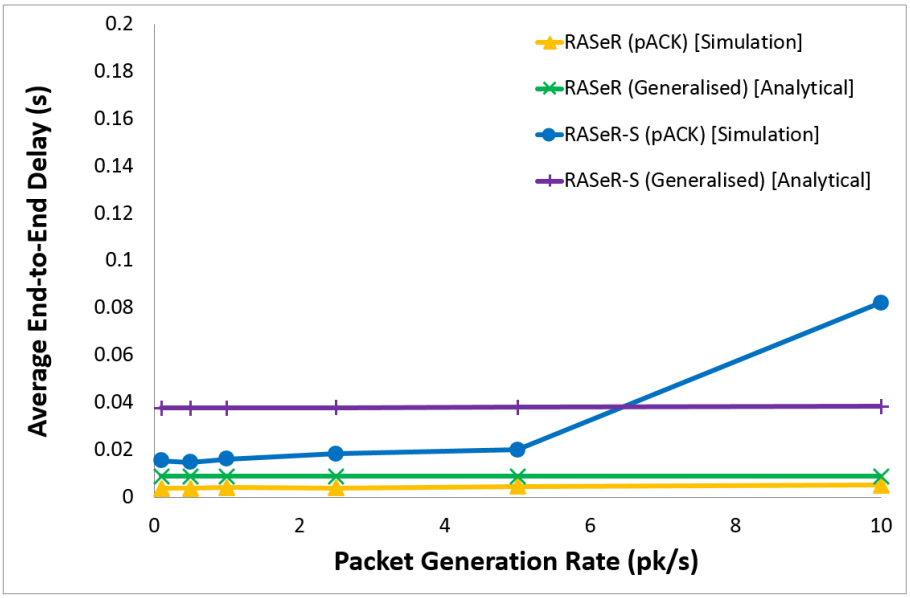


Fig. 10.7(b). Simulation and analytical end-to-end delay results for RASeR and RASeR-S in pACK mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

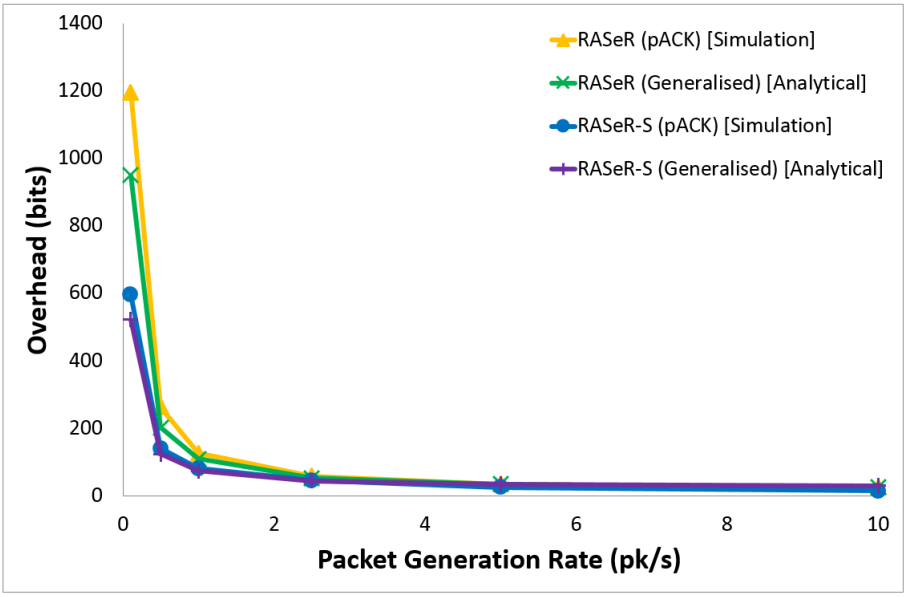


Fig. 10.7(c). Simulation and analytical overhead results for RASeR and RASeR-S in pACK mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

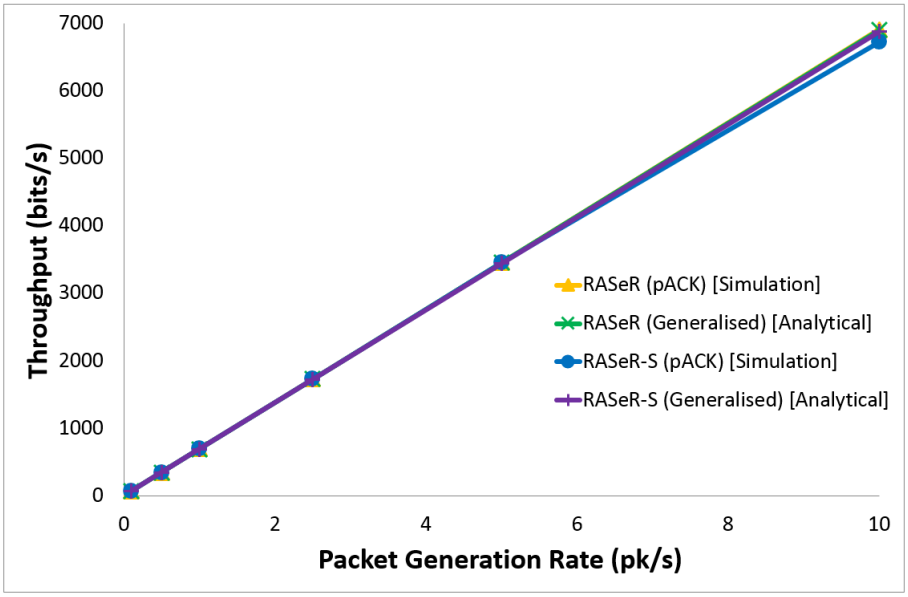


Fig. 10.7(d). Simulation and analytical throughput results for RASeR and RASeR-S in pACK mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

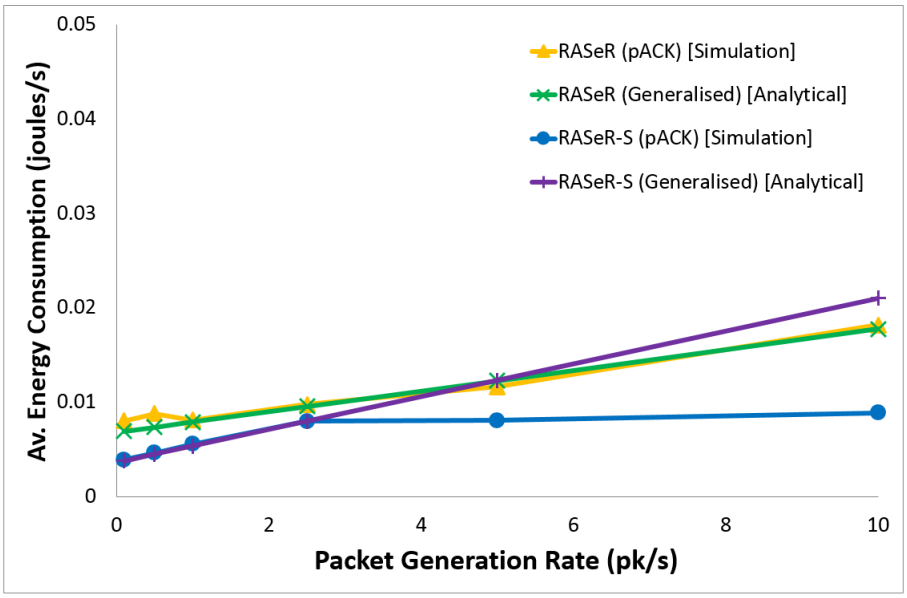


Fig. 10.7(e). Simulation and analytical energy consumption results for RASeR and RASeR-S in pACK mode, over the varying traffic levels of *[0.1, 0.5, 1, 2.5, 5, 10]pk/s*.

10.5.4 Conclusion

Generally, RASeR-S gives excellent performance across all metrics when the extended slot time is taken into account. This causes an increase in delay, which can affect the PDR in extreme scenarios. The longer timeslots also reduces the speed at which nodes are able to send packets, which reduces both overhead and energy consumption.

10.6 Conclusion

This chapter describes RASeR-S, which is a secure version of RASeR, which uses the RC5 cipher to firstly encrypt data and then packets using two different keys. The R4 framework is also introduced, which suggests that the nodes are rekeyed and reprogrammed during their recharge cycles. The thorough performance investigation of RASeR-S shows that it doesn't perform as well as RASeR, which is expected due to the large increase in overhead. However, the results still show a high PDR over and relatively low delay over all scenarios, with a slight degradation in very high traffic levels or large numbers of nodes.

In terms of security, an in-depth analysis of attacks and countermeasures was given, which illustrates the high level of information security that the protocol provides. Common attacks, such as eavesdropping, the replay attack and traffic analysis have been prevented. Additionally, without the use of very expensive and highly specialised equipment for a real-time wormhole attack, an attacker will be forced to try and compromise a node. However, even if this was successful, there are still very few attacks that the adversary could perform, data from the rest of the network would still be unreadable and the compromise would be limited to that mission at the most.

It should also be noted that the R4 framework could be applied to many MWSN applications as it gives a reliable and secure method of solving the key distribution problem, which can avoid the high overhead of key agreement schemes. It also increases security by encouraging the regular rekeying of the network, which then severely limits the amount of time an adversary has to compromise the network by brute force or other means.

The next chapter will conclude the thesis and suggest further work to be done in the future.

CHAPTER 11

CONCLUSIONS AND FUTURE DIRECTIONS

11.1 CONCLUSIONS

This thesis has focused predominantly on the topic of routing MWSNs. Initially, potential MWSN applications were categorised and conclusions drawn about their characteristics and requirements in order to motivate the work. Additionally, the techniques used for evaluating routing protocols were also discussed, which included performance metrics, a mathematical framework for network analysis, considerations for both simulation modelling and simulators, as well as testbed options.

In addition to this background information, an extensive literature review was presented, covering both routing and MAC protocols designed for MWSNs, as well as influential routing protocols designed for MANETs and WSNs. The review highlighted the need for reliable protocols with low overhead to enable them to handle the highly mobile topologies.

As a solution to the problem of routing in highly mobile WSNs, based on the identified requirements and in keeping with the envisaged scenario, the concept of using a gradient metric with a blind forwarding technique was suggested. This would satisfy the needs of a MWSN in that it's highly adaptable to topology change and also requires very little overhead. However, this creates the problem of maintaining an up-to-date gradient metric in highly mobile scenarios, a satisfactory solution for which has not been published. As such, the novel use of a GTDMA MAC layer was proposed to enable the constant refreshing of the gradient with very little additional overhead.

Based on this, two novel protocols were designed using a hop count metric and a third was proposed that makes use of the potential availability of location information. Designs were influenced by the real needs of existing and future applications. Thorough descriptions of PHASeR, LASeR and RASeR were given and their designs were compared. Additionally, mathematical expressions were derived to characterise the protocols.

In order to evaluate the performance of the protocols, models of all three were developed for use in a network simulator. Extensive simulations were then performed, which compared the three proposed with another state-of-the-art MWSN routing protocol and two MANET protocols. The results show that the fastest and most reliable protocols were RASeR and

LASeR, which attained an overall average PDR of over 99.74% and an overall average end-to-end delay of less than 19.5ms. PHASeR proved to be the most energy efficient solution with an overall average energy consumption of 0.0014joules/s. In addition to this, the effects of a fading environment were also evaluated using RASeR, which highlight the robustness of the protocol with the results showing only the expected loss from channel errors.

Furthermore, since LASeR is not reliant on the GTDMA MAC layer, potential alternate MAC protocols were identified, including GTDMA, CSMA/CA, SA and SA/CA. The idea of dedicated sensing slots was also introduced and three novel MACs were suggested, namely, NDMA/CA, NDMA/DS and CSMA/DS. Simulations of all the potential MAC alternatives were performed and the results show a preference for GTDMA, but in some cases NDMA/DS or CSMA/DS become the best choices.

Given the impressive results of RASeR and the fact that it doesn't require any location awareness, it is the most applicable in that it can be deployed in any environment and still perform well. For this reason RASeR was chosen to be improved further, firstly by the additional suggested modes of operation, which allowed the protocol to be customised to suit the needs of the application. These additions include a supersede mode, which prioritises the low latency delivery of the latest data, for applications in which receiving the most up-to-date information is important. A pACK mode for applications that are concerned with the reliability of data delivery, which is done by giving the nodes some awareness of the success of a transmission. The added reverse flooding mechanism allows the sink to broadcast network wide messages to the entire network, which may be used for mission oriented commands. Lastly, an energy saving mechanism, which introduces sleep cycles was also evaluated, such that RASeR can be used in power critical applications. Further to this, an analysis of the communication requirements was also made for the application of UAV aided search and rescue and then, with the additional modes in mind, the suitability of RASeR is evaluated.

A key concern in any communications network is that of security, as such, a secure routing framework has been proposed. RASeR-S is based on RASeR and uses an RC5 cipher in CFB mode with two layers of encryption. This is done by first encrypting the data with one key and the entire packet with another. The R4 framework was also suggested, which uses the fact that the mobile platforms in a MWSN will need to regularly be recharged. As such, given a secure recharging location, the nodes may be rekeyed and reprogrammed. The combination of dual keying with the R4 framework will significantly limit the number of attack vectors available to any adversary and severly limit the impact of any successful brute force attack as well as any tampering with the nodes firmware.

A testbed implementation of RASeR was developed and evaluated in order to assess the validity of RASeR in a real world deployment. The results showed that this style of protocol is not only a plausible solution but a reliable one. It also proves that the protocol can be implemented on commercially available hardware.

The contributions of this work are in the novel solution to the problem of gradient maintenance and the extensive investigation into using this, along with blind forwarding, in the context of a MWSN. This includes the proposal of three novel routing protocols, three novel MAC protocols and one security framework. The proposed ideas have been thoroughly evaluated through a combination of simulation, mathematical analysis and testbed deployment. The high impact of this work comes from the wide applicability of these techniques to current and future applications of MWSNs and cater for their demanding requirements. This is especially true for RASeR, whose design can be customised to suit the needs of a high number of scenarios. The protocols simulation results have shown it to give high performance in various scenarios and be robust to channel effects. In addition to this its implementation in a testbed displayed the protocols potential for real-world deployment and proves that it can be implemented on COTS hardware. Furthermore, the addition of security makes RASeR a very complete solution to the problem of routing in a MWSN.

11.2 Future Work

One key area of future work for these protocols should involve the case study of a realistic deployment, in order to evaluate the proposed solutions in a full scale deployment. One ideal scenario for such a test would be the implementation of RASeR in a UAV aided search and rescue system. In this way deeper analysis can be done, which considers how multiple systems work together and share resources. Furthermore, provisions for these additional systems should be made in subsequent simulation models. In more detailed models considerations should be made for power and processing limitations from the requirements of the sensor, any signal processing and the mobility platform. Traffic models can be further refined based on sampling time, resolution of samples and data aggregation. Mobility models can also be enhanced by considering a more realistic scenario in terms of the platform and terrain.

An additional power consumption limitation option could also be imposed on RASeR, in which the rate of beacon packets is limited. In this way the energy used could be reduced without delaying the progress of data packets. The rate at which beacons are transmitted should be based on the frequency of topology change, such that the nodes are still able to maintain an accurate hop count. Comparatively, the current energy saving mode completely shuts down the node, causing unnecessary delays in the data, but it does allow for other parts of the node to sleep and save energy as well.

Another enhancement to the protocols would be to allow for a variable number of nodes. In RASeR this could be done by implementing a timeout at the sink, for each node in the network. In this way, if the sink doesn't receive any data from a certain node for a length of time, it is assumed that the node has left the network. This time threshold would have to be selected based on the sampling frequency of the sensor. In this scenario, the sink should then send a reduction message using the reverse flooding mechanism, which will tell all nodes to reduce the number of timeslots in the round and also adjust their schedules and ID numbers. For example if the third of five sensor nodes was to leave the network, upon receiving the sinks reduction message, the first and second nodes would simply reduce the number of timeslots in the round. The fourth node would adjust the number of timeslots accordingly, begin to transmit in the third timeslot and consequently change its ID number to three. Similarly, the fifth node would adjust the number of timeslots, change its ID to four and transmit in the fourth timeslot after the sink. In the event that a node wants to join the network there will be an additional, unallocated timeslot at the end of the round for joining nodes. In beacon packets the current number of nodes in the network and the timeslot length should be included. Then, by overhearing one beacon packet, the joining node can determine when this unallocated timeslot is, which will be *n* minus the ID of the currently transmitting node, multiplied by the length of a timeslot. The joining node may then transmit a data packet in this slot with a randomly selected temporary ID value that is greater than *n*. The random temporary ID value is used in case more than one node is trying to join the network. When the sink receives this new packet it should send out an addition message with the temporary ID. When the nodes receive the addition message they should just increase the number of timeslots by one. When the joining node receives the addition message it may then begin using the unallocated timeslot with the appropriate ID. If two nodes attempt to join the network at the same time, the sink will select one. The unsuccessful node will realise that it has been unsuccessful because its temporary ID will not be in the addition message and it should therefore retry. In the case of a node being timed out before it has left the network, it may re-join as a new node. Alternatively, to prevent this, nodes may opt to send keep alive messages to the sink in the form of null data packets.

It may also be useful to apply the additional modes of RASeR to the other two protocols, especially in the case of LASeR, since it yields the best performance.

Further investigation of the security framework would also be appropriate and some research into its potential application to other protocols and network types should be done. However, in the case of RASeR-S, whilst it is difficult to simulate the success of certain attacks, simulations could be run to evaluate the effects of jamming and malicious nodes on the network. A jamming node would simply cause interference, effectively limiting the communications range of the sensor nodes. With this attack, using more jamming nodes means that more of the

network could potentially be disabled. A malicious node, or nodes, would firstly need to successfully penetrate the networks security, but assuming this was done, there are two main attacks that would be useful to simulate. The first involves the malicious device transmitting as much data as often as possible, in the hope of causing congestion. Whilst this would add some delay to the routing, the GTDMA MAC and the FCFS style queuing would limit the effectiveness of this attack. A more effective attack would be if the malicious node was to impersonate the sink. Whilst this wouldn't affect every node, it would certainly draw a significant amount of data away from the true sink. However, depending on the message set implemented, the fake sink could send a reverse flooding message and completely shut down the network.

APPENDIX A: α DERIVATION

In PHASeR each packet has a fixed maximum number of frames, which it is able to forward. The number of frames held by a packet is called the frame capacity. If the maximum frame capacity is too small a node will not be able to forward all the data required and so some frames will be dropped. If the frame capacity is too big there could be a large amount bandwidth wasted. In this appendix, we solely consider losses from the dropping of frames.

The ideal required frame capacity, *F*, for a fixed topology is directly related to the number nodes in the network, *n*, and the number of existing connections to the sink, *c*.

So the maximum required frame capacity for a given *n* and *c* is given as:

$$F_{max} = n - c \tag{A1}$$

Conversely, the minimum required frame capacity is given by:

$$F_{min} = \left\lceil \frac{n-1}{c} \right\rceil \tag{A2}$$

The total number of possible topologies for a network with *n* nodes is:

$$T = 2^{\frac{n(n-1)}{2}} \tag{A3}$$

Further to this, the number of possible topologies for a given *n* and *c* is:

$$T(n,c) = \binom{n-1}{c} \cdot 2^{\frac{(n-1)\cdot(n-2)}{2}} \tag{A4}$$

If a network had a maximum number of connections for which no losses could occur, *x*, then no losses would occur as long as $c \geq x$. So the total number of topologies that are possible without loss, for a given *x*, is:

$$T(n,x) = 2^{\frac{(n-1)\cdot(n-2)}{2}} \cdot \sum_{c=x}^{n-1} \binom{n-1}{c} \tag{A5}$$

Equation (A1) can be rearranged to give *x* for a fixed *F*:

$$x = n - F \tag{A6}$$

Substituting (A6) in to (A5) will give the total number of topologies that are possible without loss for a fixed *F*:

$$T(n,F) = 2^{\frac{(n-1)\cdot(n-2)}{2}} \cdot \sum_{c=n-F}^{n-1} \binom{n-1}{c} \tag{A7}$$

This can subsequently be described as the fraction of possible topologies without loss, *a*, by simply dividing (A7) by (A3), which gives:

$$\alpha = 2^{1-n} \cdot \sum_{c=n-F}^{n-1} \binom{n-1}{c} \qquad \text{(A8)}$$

Now various values for *F* may be evaluated using (A8), in order to achieve a satisfactory proportion of topologies that can occur without loss. This will help to minimise *F*, which will limit wasted bandwidth and reduce delay by making packet sizes smaller. Also, *a* can be maximised, which will decrease the chances of losses occurring from insufficient frame capacity.

APPENDIX B: N_F DERIVATION

N_f is the approximate number of forwarding neighbours for a packet, based on a blind forwarding method of routing, assuming that each node has correct knowledge of its location and the sinks location.

Taking the average distance between two nodes, d_{av}, is given as

$$d_{av} = L\frac{\sqrt{2}}{3} \tag{B1}$$

This can then be divided by the transmission radius, to give

$$d = \left\lceil \frac{L\sqrt{2}}{3r} \right\rceil \tag{B2}$$

Drawing these divisions as radial bands, as in figure B.1, highlights the areas within which a node is both within range and closer to the sink than the transmitting node.

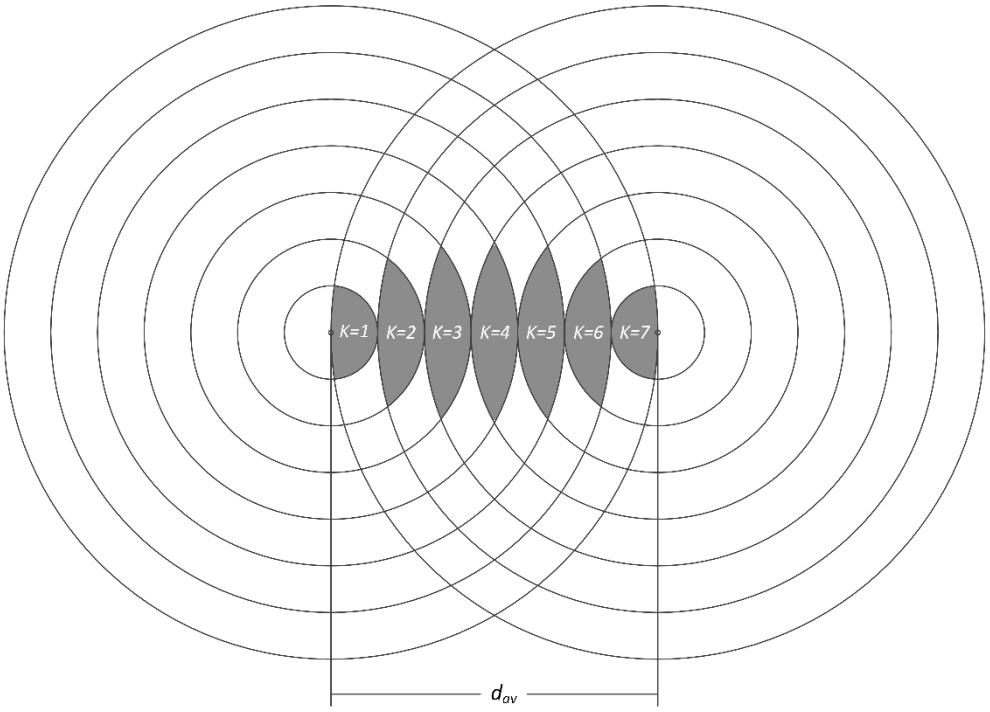


Fig. B.1. This figure highlights the area in which nodes will receive a packet and opt to forward the data towards the sink.

The sum of the lens shaped areas are indexed as k, and their total area, T_A, is given as

$$T_A = \sum_{k=1}^{k=d} [(d-k+1)r]^2 \cos^{-1}\left(\frac{[dr]^2 + [(d-k+1)r]^2 - [kr]^2}{2dr^2(d-k+1)}\right) \ldots + [kr]^2 \cos^{-1}\left(\frac{[dr]^2 + [kr]^2 - [(d-k+1)r]^2}{2dr^2k}\right) \ldots -0.5\sqrt{(-dr+(d-k+1)r+kr)(dr+(d-k+1)r-kr)(dr-(d-k+1)r+kr)(dr+(d-k+1)r+kr)} \quad \text{(B3)}$$

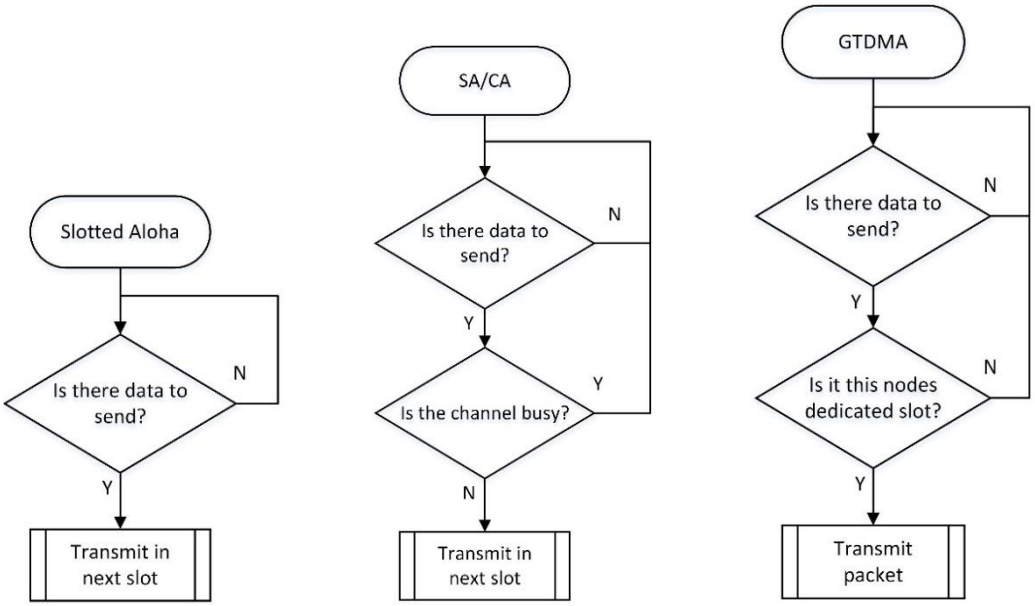
All of the nodes within T_A are expected to forward the data, so we calculate the number of forwarding nodes as the fraction of nodes expected to reside within this area. As such, an even distribution of nodes is assumed and the ratio of T_A to the total network area is multiplied by the number of nodes, excluding the source and destination. This is given as

$$N_f = \left\lceil \frac{T_A}{L^2}(n-2) \right\rceil + 1 \quad \text{(B4)}$$

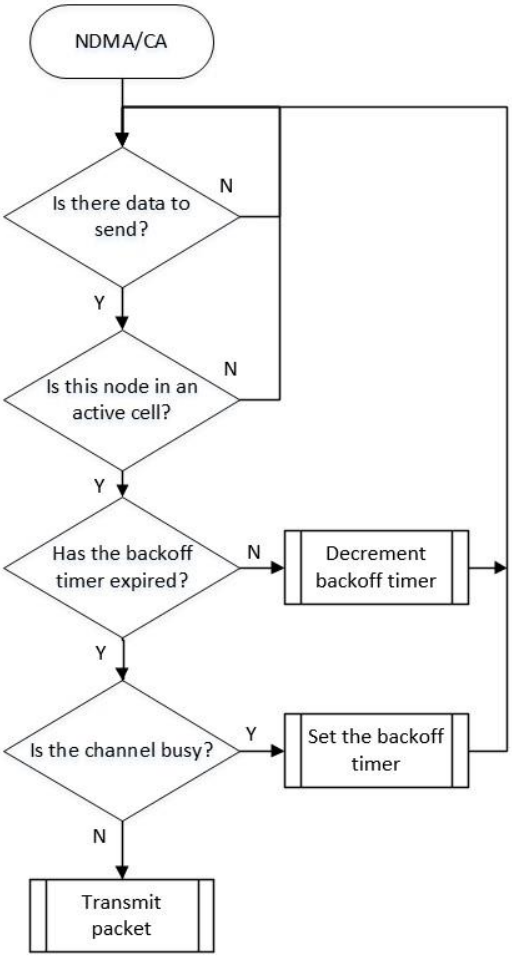
The addition of a single node at the end of the expression is to account for the source node being the first to transmit the packet.

APPENDIX C: MAC PROTOCOL FLOW CHARTS

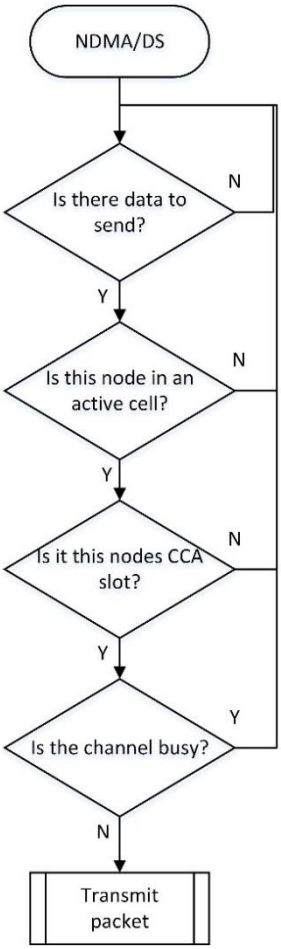
Flow charts for the selected MAC protocols; Slotted ALOHA, SA/CA, GTDMA, NDMA/CA, NDMA/DS, CSMA/CA and CSMA/DS. Flow chart (a) shows the basic slotted Aloha protocol, which simply transmits in the next slot after receiving some data from another layer. The SA/CA protocol in flow chart (b), is based on the slotted ALOHA protocol, but instead of sending in the next slot, it uses the CCA mechanism to transmit in the next slot that is sensed to be free. Flow chart (c) shows that when the GTDMA MAC has data, it simply waits for the nodes dedicated timeslot before transmitting. NDMA/CA, NDMA/DS, CSMA/CA and CSMA/DS, in flow charts (d), (e), (f) and (g) respectively, are as described in section 7.1 and have been reproduced here for comparison.



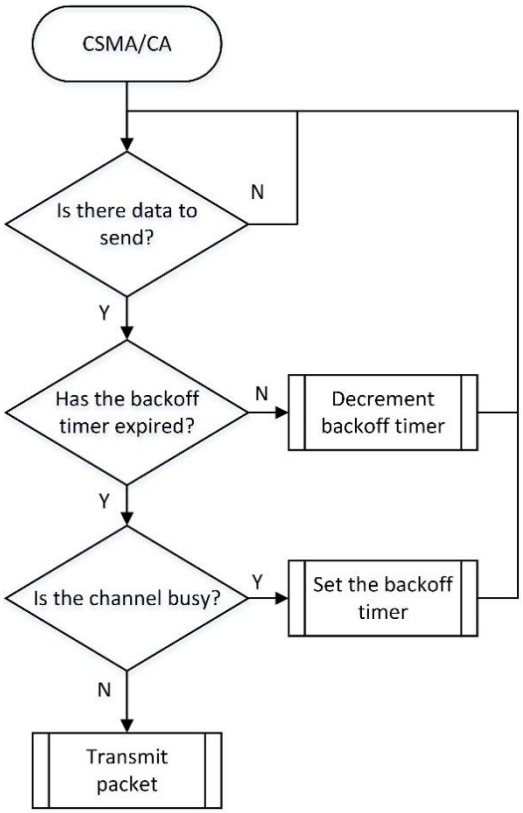
(a) (b) (c)


NDMA/CA
Is there data to send?
N
Y
Is this node in an active cell?
N
Y
Has the backoff timer expired?
N
Decrement backoff timer
Y
Is the channel busy?
Y
Set the backoff timer
N
Transmit packet

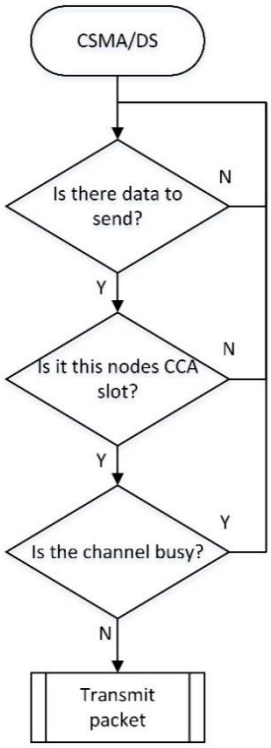
(d)


NDMA/DS
Is there data to send?
N
Y
Is this node in an active cell?
N
Y
Is it this nodes CCA slot?
N
Y
Is the channel busy?
Y
N
Transmit packet

(e)


CSMA/CA
Is there data to send?
N
Y
Has the backoff timer expired?
N
Decrement backoff timer
Y
Is the channel busy?
Y
Set the backoff timer
N
Transmit packet

(f)


CSMA/DS
Is there data to send?
N
Y
Is it this nodes CCA slot?
N
Y
Is the channel busy?
Y
N
Transmit packet

(g)

REFERENCES

Chapter 1.

[1.1] I.F. Akyildiz, S. Weilian, Y. Sankarasubramaniam and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, Aug. 2002.

[1.2] E. Ekici, Y. Gu and D. Bozdag, "Mobility-Based Communication in Wireless Sensor Networks," *IEEE Communications Magazine*, vol. 44, no. 7, pp. 56-62, July 2006.

Chapter 2.

[2.1] S. Basagni et al, "Mobile Ad Hoc Networking: Cutting Edge Directions," 2nd Ed., John Wiley and Sons, 2013.

[2.2] W. Dargie and C. Poellabauer, "Fundamentals of Wireless Sensor Networks: Theory and Practice," John Wiley and Sons, 2010.

[2.3] F. Viani et al, "Wireless Architectures for Heterogeneous Sensing in Smart Home Applications: Concepts and Real Implementation," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2381-2396, Nov. 2013.

[2.4] S. Ehsan et al., "Design and Analysis of Delay-Tolerant Sensor Networks for Monitoring and Tracking Free-Roaming Animals," *IEEE Transactions on Wireless Communications*, vol. 11, no. 3, pp. 1220-1227, 2012.

[2.5] X. Li et al., "Performance Evaluation of Vehicle-Based Mobile Sensor Networks for Traffic Monitoring," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, pp. 1647-1653, 2009.

[2.6] T. Templeton et al, "Autonomous Vision-based Landing and Terrain Mapping Using an MPC-controlled Unmanned Rotorcraft," *IEEE Int'l Conf. Robotics and Automation*, pp. 1349-1356, Apr. 2007.

[2.7] R. Pollanen et al. "Radiation surveillance using an unmanned aerial vehicle," *Elsevier Journal of Applied Radiation and Isotopes*, vol. 67, no. 2, pp. 340-344, 2009.

[2.8] S. Waharte and N. Trigoni, "Supporting Search and Rescue Operations with UAVs," *Proc. Int'l Conf. Emerging Security Technologies (EST '10)*, pp. 142-147, Sept. 2010.

[2.9] S. Zhang, "Object Tracking in Unmanned Aerial Vehicle (UAV) Videos Using a Combined Approach," *IEEE Int'l Conf. Acoustics, Speech and Signal Processing (ICASSP '05)*, pp. 681-684, Mar. 2005.

[2.10] H. Yan, H. Huo, Y. Xu and M. Gidlund, "Wireless Sensor Network Based E-Health System – Implementation and Experimental Results," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, pp. 2288-2295, 2010.

[2.11] B. Grocholsky, J. Keller, V. Kumar and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16-25, 2006.

[2.12] B. White et al., "Contaminant Cloud Boundary Monitoring Using Network of UAV Sensors," *IEEE Sensors Journal*, vol. 8, no. 10, pp. 1681-1692, 2008.

[2.13] K. Sha, W. Shi and O. Watkins, "Using Wireless Sensor Networks for Fire Rescue Applications: Requirements and Challenges," *Proc. IEEE Int'l Conf. Electro/Information Technology*, pp. 239-244, May 2006.

Chapter 3.

[3.1] K. Wong, "Physical Layer Considerations for Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Networking, Sensing and Control*, pp. 1201-1206, 2004.

[3.2] J. Lee, "Performance Evaluation of IEEE 802.15.4 for Low-Rate Wireless Personal Area Networks," *IEEE Trans. Consumer Electronics*, vol. 52, no. 3, pp. 742-749, Aug. 2006.

[3.3] J. Salehi, “Code Division Multiple-Access Techniques in Optical Fibre Networks. I. Fundamental Principles,” IEEE Trans. Communications, vol. 37, no. 8, pp. 824-833, 1989.

[3.4] R. Gagliardi, “Frequency-Division Multiple Access,” Springer Satellite Communications, 1991.

[3.5] P. Huang et al, “The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey,” IEEE Communications, Surveys and Tutorials, vol. 15, no. 1, pp. 101-120, 2013.

[3.6] H. Xu, X. Wu, H.R. Sadjadpour and J.J. Garcia-Luna-Aceves, “A Unified Analysis of Routing Protocols in MANETs,” *IEEE Transactions on Communications*, vol. 58, no. 3, pp. 911-922, Mar. 2010.

[3.7] IEEE 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Standard, IEEE, Aug. 1999.

[3.8] L. B. Jiang and S. C. Liew, “Improving Throughput and Fairness by Reducing Exposed and Hidden Nodes in 802.11 Networks,” *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 34-49, Jan. 2008.

[3.9] H. Pham and S. Jha, “Addressing Mobility in Wireless Sensor Media Access protocol,” *Proc. Intelligent Sensors, Sensor Networks and Information Processing Conf. (ISSNIP 2004)*, pp. 113-118, Dec. 2004.

[3.10] W. Ye, J. Heidemann and D. Estrin, “Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493-506, June 2004.

[3.11] P. Raviaj et al., “MOBMAC – An Energy Efficient and Low LatencyMAC for Mobile Wireless Sensor Networks,” *Proc. Systems Communications*, pp. 370-375, Aug. 2005.

[3.12] F. Peng, “A Novel Adaptive Mobility-Aware MAC Protocol in Wireless Sensor Networks,” *Springer Wireless Personal Communications Journal*, vol. 81, no. 2, pp. 489-50, March 2015.

[3.13] A. Gonga, O. Landsiedel and M. Johansson, “MobiSense: Power-Efficient Micro-Mobility in Wireless Sensor Networks,” *Proc. Int'l Conf. Distributed Computing in Sensor Systems and Workshops*, pp. 1-8, June 2011.

[3.14] M. Ali, T. Suleman and Z.A. Uzmi, “MMAC: A Mobility-Adaptive, Collision-Free MAC Protocol for Wireless Sensor Networks,” *Proc. IEEE Int’l Conf. Performance, Computing and Communications (IPCCC 2005)*, pp. 401-407, Apr. 2005.

[3.15] V. Rajendran, K. Obraczka and J.J. Garcia-Luna-Aceves, “Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks,” *ACM Int’l Conf. Embedded Networked Sensor Systems (SenSys ‘03)*, pp. 181-192, Nov. 2003.

[3.16] A. Jhumka and S. Kulkarni, “On the Design of Mobility-Tolerant TDMA-Based Media Access Control (MAC) Protocol for Mobile Sensor Networks,” *4th Int'l Conf. Distributed Computing and Internet Technology*, pp. 42-53, 2007.

[3.17] W. Dargie, “A Medium Access Control Protocol that Supports a Seamless Handover in Wireless Sensor Networks,” *Elsevier Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 778-786, 2012.

[3.18] M. Buettner et al., “X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks,” *Proc. 4th Int'l Conf. Embedded Networked Sensor Systems*, pp. 307-320, 2006.

[3.19] B. Khan and F. Ali, “Mobile Medium Access Control Protocols for Wireless Sensor Networks” *Wireless Sensor Networks: Current Status and Future Trends*, CRC Press, pp. 107-126, 2012.

[3.20] IEEE 802.15.4, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks*, Standard, ANSI/IEEE, Sep. 2006.

[3.21] B. Khan and F. Ali, “Collision Free Mobility Adaptive (CFMA) MAC for Wireless Sensor Networks,” *Springer Telecommunications Systems Journal*, vol. 52, no. 4, pp. 2459-2474, 2013.

[3.22] P. Szczytowski, A. Khelil and N. Suri, "LEHP: Localized Energy Hole Profiling in Wireless Sensor Networks," *Proc. IEEE Symp. Computers and Communications (ISCC)*, pp. 100-106, 2010.

[3.23] National Coordination Office for Space-Based Positioning, Navigation and Timing, "GPS," 2014; www.gps.gov/.

[3.24] G. Mao, B. Fidan and B. Anderson, "Wireless Sensor Network Localization Techniques," *ACM International Journal of Computer and Telecommunications Networking*, vol. 51, no. 10, pp. 2529-2553, July 2007.

[3.25] J. Elson, L. Girod, D. Estrin, "Fine-Grained Time Synchronization using Reference Broadcasts," *Proc. 5th Symp. Operating Systems Design and implementation (OSDI 2002)*, pp. 147-163, Dec. 2002.

[3.26] Y. Jo et al, "Energy Effective Time Synchronization in Wireless Sensor Network," *Proc. Int'l Conf. Computational Science and its Applications (ICCSA '07)*, pp. 547-553, Aug. 2007.

[3.27] S. Ganeriwal et al., "Timing-sync Protocol for Sensor Networks," *Proc. 1st Int'l Conf. Embedded Networked Sensor Systems (SenSys '03)*, pp. 138-149, 2003.

[3.28] W. Zhu, "TDMA frame synchronization of mobile stations using radio clock signal for short range communications," *Proc. 44th IEEE Vehicular Technology Conference*, pp. 1878-1882, June 1994.

[3.29] F. Sivrikaya and B. Yener, "Time Synchronization in Sensor Networks: A Survey," *IEEE Network*, vol. 18, no. 4, pp. 45-50, 2004.

[3.30] R.Mangharam, A.Rowe and R.Rajkumar, "FireFly: a cross-layer platform for real-time embedded wireless networks," *Real-Time Systems*, vol. 37, no. 3, pp. 183-231, 2007.

[3.31] D. Kim and Y. Chung, "Self-Organization Routing Protocol Supporting Mobile Nodes for Wireless Sensor Network", *Proc. 1st Int'l Multi-Symposiums on Computer and Computational Sciences (IMSCCS '06)*, pp. 622-626, June 2006.

[3.32] G.S. Kumar, M.V. Vinu, P.G. Athithan and K.P. Jacob, "Routing Protocol Enhancement for Handling Node Mobility in Wireless Sensor Networks," *Proc. IEEE Region 10 Conf. (TENCON)*, pp. 1-6, 2008.

[3.33] W.R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy Efficient Communication Protocol for Wireless Micro Sensor Networks," *Proc. 33rd Hawaii Int'l Conf. System Sciences (HICSS '00)*, p. 8020, 2000.

[3.34] S.A.B. Awwad, C.K. Ng, N.K. Noordin and M.F.A. Rasid, "Cluster Based Routing Protocol for Mobile Nodes in Wireless Sensor Network," *Proc. Int'l Symp. Collaborative Technologies and Systems (CTS '09)*, pp. 233-241, 2009.

[3.35] R.U. Anitha and P. Kamalakkannan, "Enhanced cluster based routing protocol for mobile nodes in wireless sensor network," *Proc. Int'l Conf. Pattern Recognition, Information and Medical Engineering (PRIME)*, pp. 187-193, 2013.

[3.36] S. Deng, J. Li and L. Shen, "Mobility-Based Clustering Protocol for Wireless Sensor Networks with Mobile Nodes," *IET Wireless Sensor Systems*, vol. 1, no. 1, pp. 39-47, Mar. 2011.

[3.37] J. Kulik, W. Heinzelman and H. Balakrishnan, "Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 169-185, Mar. 2002.

[3.38] C. Intanagonwiwat et al., "Directed Diffusion for Wireless Sensor Networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2-16, Feb. 2003.

[3.39] A. Aronsky and A. Segall, "A multipath routing algorithm for mobile Wireless Sensor Networks," *Proc. 3rd Joint IFIP Wireless and Mobile Networking Conf.*, pp. 1-6, Oct. 2010.

[3.40] X. Huang, H. Zhai and Y. Fang, "Robust cooperative routing protocol in mobile wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 5278-5285, Dec. 2008.

[3.41] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, T. Imielinski and H. Korth, Kluwer Academic Publishers, pp. 153-181, 1996.

[3.42] C.E. Perkins and E.M. Royer, "Ad-hoc On-demand Distance Vector Routing," *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pp. 90-100, 1999.

[3.43] M.K. Marina and S.R. Das, "On-Demand Multipath Distance Vector Routing in Ad Hoc Networks," *Proc. 9th Int'l Conf. Network Protocols*, pp. 14-23, Nov. 2001.

[3.44] S. Ren et al. "An Improved Wireless Sensor Networks Routing Protocol Based on AODV," *Proc. IEEE 12th Int'l Conf. Computer and Information Technology (CIT '12)*, pp. 742-746, Oct. 2012.

[3.45] H. Soliman and M. AlOtaibi, "An Efficient Routing Approach over Mobile Wireless Ad-Hoc Sensor Networks," *Proc. 6th IEEE Consumer Communications and Networking Conf. (CCNC '09)*, pp. 1-5, Jan. 2009.

[3.46] S. Cakici et al., "A Novel Cross-Layer Routing Protocol for Increasing Packet Transfer Reliability in Mobile Sensor Networks," *Springer Wireless Personal Communications Journal*, vol. 77, no. 3, pp. 2235-2254, Aug. 2014.

[3.47] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol," *IETF RFC 3626*, Oct. 2003; http://www.ietf.org/rfc/rfc3626.txt.

[3.48] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 234-244, Oct. 1994.

[3.49] S. Biswas and R. Morris, "ExOR: Opportunistic Multi-Hop Routing for Wireless Networks," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 133-144, Oct. 2005.

[3.50] G. Huo and X. Wang, "An Opportunistic Routing for Mobile Wireless Sensor Networks Based on RSSI," *Proc. 4th Int'l Conf. Wireless Communications, Networking and Mobile Computing (WiCOM '08)*, pp. 1-4, Oct. 2008.

[3.51] Y. Han and Z. Lin, "A geographically opportunistic routing protocol used in mobile wireless sensor networks," *Proc. 9th IEEE Int'l Conf. Networking, Sensing and Control (ICNSC)*, pp. 216-221, Apr. 2012.

[3.52] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. 6th Annual Int'l Conf. Mobile Computing and Networking (MobiCom '00)*, pp. 243-254, 2000.

[3.53] R. Kouah, M. Aissani and S. Moussaoui, "Mobility-based Greedy Forwarding Mechanism for Wireless Sensor Networks," *Proc. 9th Int'l Conf. Networking and Services (ICNS '13)*, pp. 140-145, Mar. 2013.

[3.54] L. Zou, M. Lu and Z. Xiong, "A Distributed Algorithm for the Dead End Problem of Location Based Routing in Sensor Networks," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 4, pp. 1509-1522, July 2005.

[3.55] L. Zou, M. Lu and Z. Xiong, "PAGER-M: A Novel Location Based Routing Protocol for Mobile Sensor Networks," *Proc. Broadwise*, Oct. 2004.

[3.56] S. Kwangcheol, K. Kim and S. Kim, "ADSR: Angle-Based Multi-hop Routing Strategy for Mobile Wireless Sensor Networks," *Proc. IEEE Asia-Pacific Services Computing Conference (APSCC)*, pp. 373-376, Dec. 2011.

[3.57] E. Lee, S. Park, H. Park, J. Lee and S. Kim, "Geographic routing based on on-demand neighbor position information in large-scale mobile sensor networks," *Proc. Int'l Symp. Autonomous Decentralized Systems (ISADS '09)*, pp. 1-7, Mar. 2009.

[3.58] U. Ahmed and F.B. Hussain, "Energy efficient routing protocol for zone based mobile sensor networks," *Proc. 7th Int'l Wireless Communications and Mobile Computing Conf. (IWCMC)*, pp. 1081-1086, 2011.

[3.59] L. Karim and N. Nasser, "Reliable location-aware routing protocol for mobile wireless sensor network," *IET Communications*, vol. 6, no. 14, pp. 2149-2158, Sept. 2012.

[3.60] Y. Wang and H. Wu, "Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN): A New Paradigm for Pervasive Information Gathering," *IEEE Transactions on Mobile Computing, vol. 6, no. 9*, pp. 1021-1034, Sept. 2007.

[3.61] H. Kanai et al. "Gradient-Based Routing in Delay Tolerant Mobile Sensor Networks Incorporating Node Mobility," *Proc. 9th IEEE Consumer Communications and Networking Conf. (CCNC)*, pp. 235-239, Jan. 2012.

[3.62] C. Shurgers and M.B. Srivastava, "Energy Efficient Routing in Wireless Sensor Networks," *Proc. IEEE Military Communications Conference (MILCOM '01)*, pp. 357-361, 2001.

[3.63] Y. Zhao, Y. Chen, B. Li and Q. Zhang, "Hop ID: A Virtual Coordinate-Based Routing for Sparse Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1075-1089, 2007.

[3.64] R. Jurdak et al., "Directed Broadcast with Overhearing for Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 1, pp. 3:1-3:35, 2009.

[3.65] L. Zhao, G. Liu, J. Chen and Z. Zhang, "Flooding and Directed Diffusion Routing Algorithm in Wireless Sensor Networks," *Proc. Ninth IEEE Int'l Conf. Hybrid Intelligent Systems*, pp. 235-239, 2009.

[3.66] R. Farivar, M. Fazeli and S.G. Miremadi, "Directed Flooding: A Fault-Tolerant Routing Protocol for Wireless Sensor Networks," *Proc. IEEE Conf. Systems Communications*, pp. 395-399, Aug. 2005.

[3.67] M. Maroti, "Directed Flood-Routing Framework for Wireless Sensor Networks," *Proc. 5th ACM/IFIP/USENIX Int'l Conf. Middleware*, pp. 99-114, 2004.

[3.68] D. Hubner, J. Kaltwasser, J. Kassubek and F. Reichert, "A Multihop Protocol for Contacting a Stationary Infrastructure," *Proc. 41st IEEE Conf. Vehicular Technology*, pp. 414-419, May 1991.

[3.69] H. Liu et al, "Opportunistic Routing for Wireless Ad Hoc and Sensor Networks: Present and Future Directions," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 103-109, Dec. 2009.

[3.70] A. Perrig et al., "SPINS: Security Protocols for Sensor Networks," *Kluwer Wireless Networks Journal*, vol. 8, no. 5, pp. 521-534, 2002.

[3.71] A. Perrig et al., "Efficient Authentication and Signing of Multicast Streams Over Lossy Channels" *IEEE Symp. Security and Privacy*, pp. 56-73, May 2000.

[3.72] J. Deng, R. Han and S. Mishra, "A Performance Evaluation of Intrusion-Tolerant Routing in Wireless Sensor Networks" *Proc. 2nd Int'l Conf. Information Processing in Sensor Networks*, pp. 349-364, 2003.

[3.73] C. Karlof, N. Sastry and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks" *Proc. 2nd Int'l Conf. Embedded Networked Sensor Systems*, pp. 162-175, 2004.

[3.74] TinyOS, "TinyOS," 2015; http://www.tinyos.net.

[3.75] A. Wood et al., "SIGF: A Family of Configurable, Secure Routing Protocols for Wireless Sensor Networks," *Proc. 4th ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 35-48, 2006.

[3.76] B. Blum et al., "IGF: A State-Free Robust Communication Protocol for Wireless Sensor Networks," *Technical Report CS-2003-11*, Univ. Virginia, 2003.

[3.77] L. Zhou, J. Ni and C. Ravishankar, "Supporting Secure Communication and Data Collection in Mobile Sensor Networks," *Proc. 25th IEEE Int'l Conf. Computer Communications*, pp. 1-12, Apr. 2006.

[3.78] A. Rasheed and R. Mahapatra, "The Three-Tier Security Scheme in Wireless Sensor Networks with Mobile Sinks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 958-965, 2012.

[3.79] C. Blundo et al., "Perfectly-Secure Key Distribution for Dynamic Conferences," *Proc. 12th Ann. Int'l Cryptology Conf. Advances in Cryptology*, pp. 471-486, 1993.

[3.80] A. Sarma, B. Kar and C. Mall, "Secure Routing Protocol for Mobile Wireless Sensor Network," *Proc. Sensors Applications Symp.*, pp. 93-99, Feb. 2011.

[3.81] A. Sarma et al., "Energy Efficient Communication Protocol for Mobile Wireless Sensor Network System," *Int'l Journal of Computer Science and Network Security*, vol. 9, no. 2, pp. 386-394, 2009.

[3.82] J. Clark et al., "Threat Modelling for Mobile Ad Hoc and Sensor Networks," *Ann. Conf. of ITA*, 2007.

[3.83] A. Das, "A Key Establishment Scheme for Mobile Wireless Sensor Networks Using Post-Deployment Knowledge," *Int'l Journal of Computer Networks & Communications*, vol. 3, no. 4, pp. 57-70, 2011.

[3.84] S. Schmidt et al., "A Security Architecture for Mobile Wireless Sensor Networks," *Proc. 1st European Workshop on Security in Ad-hoc and Sensor Networks*, pp. 166-177, Aug. 2004.

[3.85] R. Anderson, E. Biham and L. Knudsen, "A Proposal for the Advanced Encryption Standard," http://ftp.cl.cam.ac.uk/ftp/users/rja14/serpent.pdf, 1998.

Chapter 4.

[4.1] P.A. Contla and M. Stojmenovic, "Estimating Hop Counts In Position Based Routing Schemes For Ad Hoc Networks," *Telecommunications Systems*, vol. 22, no. 1-4, pp. 109-118, 2003.

[4.2] R. Hekmat and P. van Mieghem, "Degree Distribution and Hop Count in Wireless Ad-hoc Networks," *Proc. IEEE Int'l Conf. Networks (ICON '03)*, pp. 603-609, Oct. 2003.

[4.3] P. Samar and S.B. Wicker, "On the Behaviour of Communication Links of a Node in a Multi-Hop Mobile Environment," *Proc. 5th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc '04)*, pp. 145-156, 2004.

[4.4] G. Bolch et al., "Queueing Networks and Markov Chains: Modelling and Performance Evaluation with Computer Science Applications," 2nd Ed., John Wiley and Sons, 2006.

[4.5] W. Chan, T. Lu and R. Chen, "Pollaczek-Khinchin Formula for the M/G/1 Queue in Discrete Time with Vacations," *Proc. IET Conf. Computers and Digital Techniques*, pp. 222-226, July 1997.

[4.6] G. S. Fishman, "Principles of Discrete Event Simulation," John Wiley & Sons, New York, USA, 1978.

[4.7] NSNAM, "ns-2," 2011; http://nsnam.isi.edu/nsnam/index.php/Main_Page/.

[4.8] NSNAM, "ns-3," 2014; http://www.nsnam.org/.

[4.9] Riverbed, "OPNET," 2014; http://www.riverbed.com/products/performance-management-control/opnet.html?redirect=opnet/.

[4.10] Scalable Network Technologies, "QualNet," 2014; http://web.scalable-networks.com/content/qualnet/.

[4.11] X. Zeng, R. Bagrodia and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," *Proc. 12th Workshop on Parallel and Distributed Simulation (PADS '98)*, pp. 154-161, May 1998.

[4.12] J-Sim, "J-Sim Official," 2005; https://sites.google.com/site/jsimofficial/.

[4.13] R. Barr, Z. Haas and R. van Renesse, “JiST: An Efficient Approach to Simulation Using Virtual Machines,” *Wiley Journal of Software: Practice and Experience*, pp. 539-576, May 2005.

[4.14] Cornell, “JiST/SWANS,” 2014; http://jist.ece.cornell.edu/.

[4.15] E. Weingartener, H. vom Lehn and K. Wehrle, “A Performance Comparison of Recent Network Simulators,” *Proc. IEEE Int’l Conf. Communications (ICC ‘09)*, pp. 1-5, June 2009.

[4.16] Texas Instruments, “eZ430-RF2500 Development Tool: User’s Guide,” 2009; http://www.ti.com/lit/ug/slau227e/slau227e.pdf/.

[4.17] H. Lunndgren et al. “A Large-Scale Testbed for Reproducible Ad Hoc Protocol Evaluations,” *Proc. IEEE Wireless Communications and Networking Conf. (WCNC ‘02)*, pp. 412-418, Mar. 2002.

[4.18] Memsic, “TELOSB,” 2014; http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf/.

[4.19] Memsic, “MICAz,” 2014; http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf/.

[4.20] Memsic, “IRIS,” 2014; http://www.memsic.com/userfiles/files/Datasheets/WSN/IRIS_Datasheet.pdf/.

[4.21] Shimmer Sensing, “Shimmer Wireless Sensor Unit,” 2013; http://www.shimmersensing.com/images/uploads/docs/Shimmer_Wireless_Sensor_Platform_Spec_Sheet.pdf/.

[4.22] ArduPilot Mega, “APM,” 2014; http://www. ardupilot.com/.

[4.23] TinyOS, “TinyOS,” 2014; http://www.tinyos.net/.

[4.24] Contiki, “Contiki: The Open Source OS for the Internet of Things,” 2014; http://www.contiki-os.org/.

[4.25] LiteOS, “LiteOS: A Unix-like Operating System for Embedded Controllers and Sensor Networks,” 2014; http://www.liteos.net/.

[4.26] S. Bhatti et al, “MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms,” *ACM Mobile Networks and Applications*, vol. 10, no. 4, pp. 563-579, Aug. 2005.

Chapter 5.

[5.1] I. Amundson and X. Koutsoukos, “A Survey on Localization for Mobile Wireless Sensor Networks,” *Proc. 2nd Int’l Workshop on Mobile Entity Localization and Tracking in GPS-less Environments*, pp. 235-254, Sept. 2009.

Chapter 6.

[6.1] J. Li et al., “Study on ZigBee Network Architecture and Routing Algorithm,” *Proc. 2nd Int’l Conf. Signal Processing Systems (ICSPS)*, pp. 389-393, July 2010.

[6.2] Z. Ren et al., “Modelling and Simulation of Rayleigh Fading, Path Loss and Shadowing Fading for Wireless Mobile Networks,” *Elsevier Simulation Modelling Practice and Theory Journal*, vol. 19, no. 2, pp. 626-637, 2011.

Chapter 7.

[7.1] D. Corbett and D. Everitt, “A Partitioned Power and Location Aware MAC Protocol for Mobile Ad Hoc Networks,” *Proc. 11th European Wireless Conf. Next Generation Wireless and Mobile Communications and Services*, pp. 1-7, Apr. 2005.

Chapter 8.

[8.1] G. Remy et al. "SAR.Drones: Drones for Advanced Search and Rescue Missions," *Proc. Journees Nationales des Communications dans les Transports (JNCT '13)*, May 2013.

[8.2] S. Waharte, N. Trigoni and S. Julier, "Coordinated Search with a Swarm of UAVs," *Proc. 6th Annual IEEE Communications Society Conf. Sensor, Mesh and Ad Hoc Communications and Networks Workshops (SECON Workshops '09)*, pp. 1-3, June 2009.

[8.3] A. Ryan and J.K. Hedrick, "A Mode-Switching Path Planner for UAV-Assisted Search and Rescue," *Proc. IEEE Conf. Decision and Control and European Control Conf. (CDC-ECC '05)*, pp. 1471-1476, Dec. 2005.

[8.4] P. Fabiani et al., "Autonomous flight and navigation of VTOL UAVs: from autonomy demonstrations to out-of-sight flights," *Elsevier Aerospace Science and Technology Journal*, vol. 11, no. 2-3, pp. 183-193, Mar.-Apr. 2007.

[8.5] P. Rudol and P. Doherty, "Human Body Detection and Geolocalization for UAV Search and Rescue Missions Using Color and Thermal Imagery," *Proc. IEEE Aerospace Conf.*, pp. 1-8, March 2008.

[8.6] J. Riehl, G. Collins and J. Hespanha, "Cooperative Search by UAV Teams: A Model Predictive Approach using Dynamic Graphs," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 4, pp. 2637-2656, Oct. 2011.

[8.7] M. Polycarpou, Y. Yang and K. Passino, "A cooperative search framework for distributed agents," *Proc. IEEE Int'l Symp. Intelligent Control (ISIC '01)*, pp. 1-6, Sept. 2001.

[8.8] A. Ryan et al. "An Overview of Emerging Results in Cooperative UAV Control," *Proc. 43rd IEEE Conf. Decision and Control (CDC '04)*, pp. 602-607, Dec. 2004.

Chapter 9.

[9.1] H. Lundgren et al., "A Large-scale Testbed for Reproducible Ad hoc Protocol Evaluations," *Proc. IEEE Wireless Communications and Networking Conference (WCNC2002)*, pp. 412-418, Mar. 2002.

[9.2] D. Johnson et al., "Mobile Emulab: A Robotic Wireless and Sensor Network Testbed," *Proc. IEEE Conf. Computer Communications (INFOCOM)*, pp. 1-12, Apr. 2006.

[9.3] P. De et al., "MiNT-m: An Autonomous Mobile Wireless Experimentation Platform," *Proc. Int'l Conf. Mobile Systems, applications and services (MobiSys '06)*, pp. 124-137, June 2006.

[9.4] S. Bromage et al., "SCORPION: A Heterogeneous Wireless Networking Testbed," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 13, no. 1, pp. 65-68, Jan. 2009.

[9.5] C. Fok et al., "Pharos: A Testbed for Mobile Cyber-Physical Systems," *Technical Report TR-ARiSE-2011-001*, Univ. Texas at Austin, 2011.

[9.6] O. Rensfelt et al., "Sensei-UU: A Relocatable Sensor Network Testbed," *Proc. ACM Int'l Workshop on Wireless Network testbeds, experimental evaluation and characterization*, pp. 63-70, 2010.

[9.7] RobotShop Inc., "DFRobotShop Rover,"2015; http://www.robotshop.com/media/files/pdf/dfrobotshop-rover-user-guide.pdf.

[9.8] Arduino, "Arduino," 2015; http://www.arduino.cc.

Chapter 10.

[10.1] Y. Ren et al., "Security in Mobile Wireless Sensor Networks – A Survey," *Academy Publisher Journal of Communications*, vol. 6, no. 2, pp. 128-142, 2011.

[10.2] R. Rivest, "The RC5 Encryption Algorithm," *Workshop on Fast Software Encryption*, pp. 86-96, 1995.

[10.3] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Elsevier Ad-Hoc Networks Journal Special Issue on Sensor Network Applications and Protocols*, vol. 1, no. 2-3, pp. 293-315, 2003.