Flexible-Schedule-Based TDMA Protocols for Supporting Fault-Tolerance, On-Demand TDMA Slot Transfer, and Peer-to-Peer Communication in Wireless Sensor Networks

THIS THESIS IS

PRESENTED TO THE

SCHOOL OF COMPUTER SCIENCE & SOFTWARE ENGINEERING

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

OF

THE UNIVERSITY OF WESTERN AUSTRALIA

By

Winnie Louis Lee

January 2008

Abstract

This thesis develops a scheduled protocol (time division multiple access, TDMA) called *flexible-schedule-based TDMA Protocol* (FlexiTP), to address the problem of providing *end-to-end guarantees on data delivery*, whilst also respecting severe *resource constraints* of wireless sensor networks. FlexiTP achieves this balance through a distributed, synchronised, and loose slot structure in which sensor nodes can build, modify, or extend their schedules based on their local information. In FlexiTP, it is not necessary to predetermine the number of slots required for a network. FlexiTP's local repair scheme allows nodes to adjust their schedules dynamically and autonomously to recover from node and communication faults. Hence, it maintains a reliable and self-organising multihop network.

Most sensor network protocols designed for data gathering applications implicitly assume a periodic rate of data collection from all nodes in the network to the base station. However, nodes may want to report their data more rapidly or slowly depending on the significance and importance of their data to the end-user. The problem is that traditional TDMA-based protocols are not flexible to changes in traffic patterns because of their rigid slot structure schemes. This thesis aims to solve this problem by developing an *on-demand TDMA slot transfer* method that leverages the flexible-slot structure algorithm of FlexiTP to transfer time slots from one part of the network to another part. Hence, it allows wireless sensor networks to be adaptive to dynamic traffic patterns in the channel.

Many wireless sensor networks are designed with a specific communication pattern such as broadcast from the base station to all sensor nodes in a network, or convergecast from all nodes in a network towards the base station. While these communication patterns are sufficient for monitoring applications, individual sensor nodes may need to send their data to multiple destination nodes across the network in order to execute a distributed cooperative-function based on their local environment. This *peer-to-peer communication pattern* makes sensor networks more reactive to triggers from the environment. This thesis attempts to solve the problem of lack of peer-to-peer communication in the design of a TDMA-driven protocol by extending the idea of on-demand TDMA slot transfer method to allow each sensor node in the network to claim extra time slots to communicate with any other nodes (peers) in the network, without going through the base station.

Nodes in the network may have different priorities of data because of event-triggering sensor readings or various types of sensor readings (e.g., light, temperature, and humidity) they provide. When nodes with high priority packets increase the frequency of their data collections, the network bandwidth may be dominated by these nodes. It is desirable to allow nodes with low priority packets to aggregate their packets and so enabling these nodes to send their data to the base station under the current available network bandwidth. This thesis proposes an *on-demand data aggregation* algorithm that enables sensor nodes to perform an in-network-aggregation based on their current sensing requirements and network capacity constraints.

In summary, this thesis describes the design, implementation, and evaluation of protocols for wireless sensor networks that focus on achieving energy-efficiency, provisioning performance assurances, and supporting reactivity and adaptability in constantly changing environment.

Preface

Recent trends in the pervasive computing community move towards implementing wireless networks of resource-constrained sensor nodes in a whole new range of applications, including telemonitoring of human physiological data, habitat monitoring of wildlife, automated and smart homes video surveillance, traffic monitoring, vehicle tracking and detection for battlefield surveillance, tracking and monitoring doctors and patients inside a hospital, environmental detection of fire and flood, microclimate monitoring for precision agriculture, and vibration-based structural condition monitoring. Sensor nodes gather information by observing the physical world at fine granularity and transmitting information to more powerful hardware that can process it.

This thesis focuses on designing energy-efficient, fault-tolerant, reactive, and adaptive TDMA protocols for wireless sensor networks. The proposed protocols are implemented, using C++ and OTcl, into the network simulator (NS-2). All of the software and protocols implementation developed in this thesis are available online at:

`http://www.csse.uwa.edu.au/~winnie/programs`

The thesis work has been carried out from March 2004 to July 2007 at the School of Computer Science and Software Engineering. This thesis consists of eight chapters. Four chapters contain papers that are accepted by or intended for international journals or proceedings. These chapters cover the report of our proposed scheduling and routing protocol, traffic-adaptive protocol, peer-to-peer-driven protocol, and network management protocol, for wireless sensor networks.

Acknowledgements

I found joy not only in finishing my doctoral work but also in doing it. This thesis is the result of three and half years of work whereby I have been accompanied and supported by many people. It is a pleasure to express my gratitude to all of them.

It is difficult to overstate my gratitude to my PhD supervisors. Associate Professor Amitava Datta has greatly inspired me, encouraged me to develop research skills, taught me how to be an independent researcher, and given me a lot of research directions. I would have been lost without him. I am deeply indebted to Associate Professor Rachel Cardell-Oliver for her encouragement, steadfast support, valuable advice, and all the interesting discussions throughout the course of my PhD programme.

I am deeply grateful to anonymous referees whose comments helped me improve my research papers that in turns improve my thesis chapters.

I wish to thank student colleagues: Babak Pazand (The University of Western Australia), Valance Phua (The University of Western Australia), and Ajit Warrier (North Carolina State University) for fruitful discussions on NS-2, it was great to collaborate with you all.

I am grateful to the researchers, computer systems administrators, and secretaries in the school of computer science and software engineering, for assisting me in many different ways. Dr David Glance, Dr Chris McDonald, Jing Bo Sun, Ashley Chew, Laurie McKeaig, and Sam Sein Muan Tie, deserve special mention.

My research has been partially supported and funded by Australian Postgraduate Award, a scholarship from the School of Computer Science & Software engineering at the University of Western Australia, ARC Linkage Grant, the contributions from both my supervisors, the substantial travel grant from the Council of Convocation (the University Graduates Association), and the travel grant from the Postgraduate Student Association.

I wish to thank my sister, Berlina Louis Lee, and also my dear friends for providing a fun, supportive, and loving environment for me.

I wish to thank my fiance, Hans William Sendjaja, for helping me get through the difficult times and for his continuous emotional support.

Lastly, and most importantly, I wish to thank my parents, Benny Louis Lee and Deasy Louis Lee. They raised me, supported me, and loved me. To them I dedicate this thesis.

Publications

International Journal Publication (*Fully Refereed*)

[1] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver, “FlexiTP: a flexible-schedule-based TDMA protocol for fault-tolerant and energy-efficient wireless sensor networks”, accepted to appear in *IEEE Transactions on Parallel and Distributed Systems*.

The paper forms Chapter 4 of this thesis.

[2] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver, “On-demand TDMA slot transfer for supporting adaptive sensing in wireless sensor networks”, submitted to *ACM Transactions on Sensor Networks*.

The paper forms Chapter 5 of this thesis.

[3] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver, “Peer-to-peer multi-casting overlay over TDMA schedule for reactive wireless sensor networks”, submitted to *Elsevier Ad Hoc Networks*.

The paper forms Chapter 6 of this thesis.

Book Chapter Publication (*Fully Refereed*)

[4] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver, ”Network management in wireless sensor networks”, accepted to appear in *Handbook of Mobile Ad Hoc and*

Pervasive Communications, edited by Mieso K. Denko and Laurence. T. Yang, published by American Scientific Publishers.

The comprehensive literature review presented in this paper contributed towards Chapter 7 of this thesis.

International Conference Publications (*Fully Refereed*)

[5] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver, “FlexiMAC: a flexible TDMA-based MAC protocol for fault-tolerant and energy-efficient wireless sensor networks”, in *Proceedings of the 14th IEEE International Conference on Networks (ICON’06)*, volume 2, pages 1-6, September 2006.

The preliminary ideas of this paper were refined and extended to contribute towards [1].

[6] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver, “A novel systematic resource transfer method for wireless sensor networks”, in *Proceedings of the 49th IEEE Global Telecommunications Conference (GLOBECOM’06)*, pages 1-6, December 2006.

The preliminary ideas and results of this paper were refined and extended to contribute towards [2].

International Conference Publications (*Refereed on the Basis of Abstract*)

[7] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver, “Peer-to-peer multicasting overlay over TDMA schedule for reactive wireless sensor networks”, submitted to the fifth ACM Conference on Embedded Networked Sensor Systems (SenSys’07).

The preliminary ideas and results of this poster abstract were refined and extended to contribute towards [3].

Contribution of Candidate to Published Papers

My contribution in all the papers was 85%. I developed and implemented the protocols, performed the simulations and wrote the papers. My supervisors, Associate Professor Amitava Datta and Associate Professor Rachel Cardell-Oliver, reviewed the papers and provided useful feedback for improvement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A journey of a thousand miles begins with a single step.

Lao-tzu

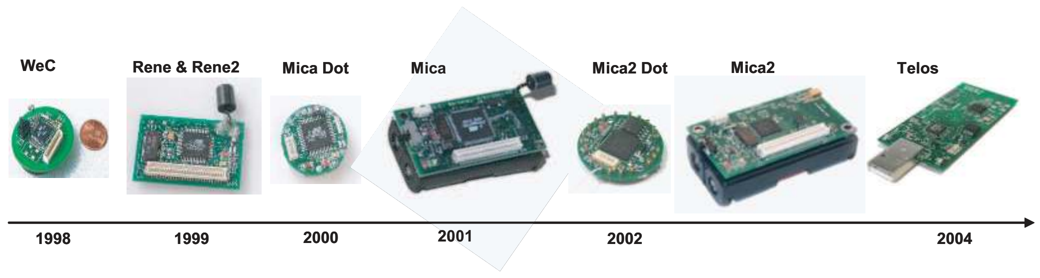

Figure 1: The UC Berkeley mote evolution from 1998 to 2004

In the last decade, we have witnessed the rapid growth of wireless sensor network, that is the network consisting of many small, low-power, and intelligent sensor nodes and one or more base stations. Figure 1 shows the evolution of the wireless sensor network hardware platforms, called motes, developed at UC Berkeley. A sensor node (or mote) is a battery-powered device that is equipped with sensors such as seismic, light, temperature, and acoustic; a processor for mapping a physical quantity taken from the environment to a quantitative measurement as well as performing network protocol functions; and a radio transceiver for communication. A sensor node works under severe resource

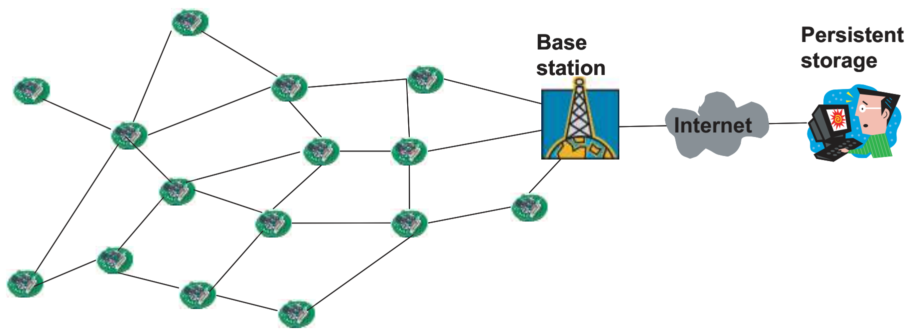

Figure 2: A wireless sensor network consisting of motes and a single base station. Autonomous motes cover a limited physical area of environment, form a multihop network, and send data to the base station, through which an end user retrieves the data.

constraints such as limited battery power, computing power, memory, wireless bandwidth, and communication capability, while the base station has more computational, energy and communication resources.

Figure 2 illustrates a wireless sensor network. Sensor nodes are placed in a physical area of interest, for example, natural ecosystems, battlefields, and man-made environments [3, 41, 79, 121] to gather high-level description of events such as temperature, sound, motion, vibration, or pressure from the environment and send this information to the base station. The base station is a laptop, PDA, or a wireless hand-held device. It stores the information in a database and an end-user can retrieve this information through the Internet. The base station acts as a gateway between sensor nodes and the end user.

Wireless sensor network applications can be categorised according to their operational paradigm: data gathering and event-driven. Data gathering applications require sensor nodes to periodically report their data to the base station [13]. Sensor nodes serve as both gatherers and routers for the data. Examples of data gathering applications are temperature and humidity monitoring in the canopy of potato plants for precision agriculture [30], habitat monitoring at Great Duck Island [79], and microclimate monitoring in redwood trees [110]. In the event-driven paradigm, sensor nodes only send data when an event of interest occurs. For example, sensor nodes are used to navigate users through the monitored environment in the presence of hazardous events (e.g., a chemical spill [18], a traffic accident [72]), to track a fire as it spreads and form a perimeter for the fire area [42], to follow the movement of a source to be tracked (e.g., object tracking [12], trespasser detection [34]), to detect intrusion in the sand [11], and to monitor vibration (e.g., wind and earthquakes) that could damage the structure of a building [121].

In wireless sensor networks, nodes have limited resources, particularly energy and wireless bandwidth, that are significantly different from that in traditional wireless networks such as wireless local area networks, cellular networks, and ad hoc networks. The following are characteristics that are unique to wireless sensor networks.

First, sensor nodes are typically battery-powered and deployed in a remote location. It is expensive in terms of cost and effort and often impractical to replace their batteries. Sensor nodes must operate efficiently in order to prolong the network lifetime and so make sensor network applications economically feasible [65].

Second, wireless sensor networks adopt multihop infrastructure where every node communicates through other nodes in a sensor network to produce information-rich results (e.g., temperature and soil-moisture in a certain region of the network). Furthermore, intermediate nodes can perform data aggregation and caching to reduce communication overheads [48, 56]. Single hop infrastructure in large network areas would drain much energy from sensor nodes because the transmission power required to send data increases proportionally with distance.

Lastly, wireless sensor networks view a network as a distributed system consisting of many autonomously cooperating sensor nodes [60], any of which may have a role in data gathering, data processing, or routing. Sensor nodes report data that are gathered periodically or triggered by an external event to the base station for processing and accordingly generating an appropriate response [65].

1.1 Motivation and Objectives

1.1.1 Protocol Design Criteria for Sensor Networks

A protocol designed for wireless sensor networks must take into account the properties that are important to sensor network applications. The following are design goals of sensor network protocols:

- *Lightweight operation.* An important challenge in the design of sensor network protocol is to conserve energy. Every activity in wireless sensor networks must be run efficiently without consuming too many resources or interfering with the operation of the sensor nodes. Lightweight operation prolongs network lifetime.
- *Self-organisation and autonomous operation.* In most sensor network applications, sensor nodes are randomly deployed in remote and harsh conditions and so they need to be able to self-organise without prior knowledge of the network topology and ideally to function autonomously without (much) human intervention.
- *Self-configuration and fault tolerance.* Wireless sensor networks are prone to network failures such as dropped packets due to external interferences in the channel, nodes dying when running out of energy, becoming disconnected, powering on or off, and new nodes joining the network. A protocol designed for sensor networks should be resilient to these network dynamics and able to reconfigure the network as required in order to maintain efficient operations.
- *Minimal storage requirement and memory efficiency.* A sensor network protocol should maintain information needed to support sensing application, but must also respect the memory constraints of sensor nodes.
- *Scalability.* It is envisioned that sensor network applications will involve large number of nodes. A sensor network protocol should be able to operate efficiently in any network size.

1.1.2 Medium Access Control

Recall that a sensor node has three main hardware resources: sensors, the processor, and the radio transceiver. Most current research on sensor network protocols has focused on medium access techniques for radio transmission and reception since the radio transceiver consumes more energy than any other hardware [54]. The medium access control (MAC) layer sits on the top of the physical layer and controls the radio. Thus, MAC protocol has the most direct control over the radio utilisation. Sensor nodes conserve energy by turning off unneeded hardware (sleeping mode) because most hardware, even when not active, consumes a non-negligible amount of energy. Several techniques, such as contention-based MAC protocols, schedule-based MAC protocols, or both, exist and will be discussed in Chapter 2.

MAC protocol designers must reduce transceiver energy consumption by preventing or limiting collisions, overhearing, idle listening, overhead, and over-emitting. A *collision* is caused by the hidden terminal problem where nodes within the interference range of each other send a packet at the same time. Frequent collisions result in high energy waste and hence reduce the network lifetime.

Overhearing is a situation where nodes overhear the packet that is destined to other nodes and they waste energy for receiving and processing the packet. *Idle listening* happens when sensor nodes' transceivers are in receive mode when there is no activity in the radio channel (i.e., no sensor node sends a packet). Although the transceiver in receive mode consumes less energy than in transmit mode, it results in higher energy waste than in sleep mode.

MAC protocols designed for sensor networks often use RTS/CTS packets to avoid hidden terminals and ACK packet to provide reliability. The transmission of these control packets requires a significant amount of energy *overhead* and hence should be avoided. Finally, *over-emitting* is a case where an intended receiver of a packet is not ready while a sender has already begin transmitting the packet to it. MAC protocol designers should consider time synchronisation issues and radio state switching times in order to prevent violating timing of sensor node activity. For example, a sensor node that sleeps for a period shorter than the radio switching time may miss a transmission intended for it.

Since sensor nodes have limited memory, MAC protocol designers must also ensure that *memory is used efficiently*. MAC protocols should maintain minimal network state information such as link state quality and neighbourhood traffic generation history. Moreover, MAC protocols designers must consider packet flow control in the network in order to prevent the loss of packets due to buffer overflow. That is MAC protocols should ensure that intended receivers have sufficient memory to store packets that are destined for them.

Scalability poses a further problem for protocol designers. MAC protocols must provide scalability both in network size and network density to support sensor networks of hundreds to thousands of sensor nodes. In wireless sensor networks, distributed or decentralised protocols are preferable to centralised protocols. The centralised scheme relies on a central node to perform sensor network operations and processing based on the global information of the network, while the decentralised scheme enables nodes to autonomously perform sensor network tasks based on their local information. Centralised protocols incur high bandwidth and energy for data polling, and these overheads limit their scalability. Furthermore, the central node is a single point of data traffic concentration and potential failure. If a network is partitioned, nodes that are unable to reach the central node become useless. On the other hand, distributed protocols have lower communication costs than centralised protocols and so provide better reliability and energy-efficiency. Distributed protocols are complex and may be computationally too expensive for resource-constrained sensor network nodes. For example, sensor nodes may need to regularly share information with their neighbours to perform a distributed task based on their local neighbourhood states. This neighbourhood information exchange is energy expensive since sensor nodes perform multihop communication to reach their neighbours. The protocol designer should balance the advantage of information sharing among nodes in a neighbourhood with the cost of exchanging that information.

1.2 Challenges

Existing research in wireless sensor networks has focused largely on the energy efficiency aspect of sensor networks in general and MAC in particular; provisioning classical performance parameters such as throughput, latency, and fairness in sensor networks. Developing traffic-adaptive and fault-tolerant protocols remain open research problems.

First, the challenge is to design an *energy-efficient* protocol that maintains *good network performance*. The most common approach to conserve energy is to alternate sensor node duty cycles between high power active state and low power sleep state. Several researchers have already shown extensive solutions for achieving energy-efficient MAC protocols based on CSMA [15, 37, 86, 114, 123, 124], TDMA [10, 21, 40, 64, 71, 84, 115, 116], or both [47, 52, 88, 89, 91, 94, 128]. These MAC protocols typically trade off performance for the reduction of energy consumption to prolong a sensor node's lifetime [65]. However, as new applications of sensor networks emerge, classical network performance parameters such as latency, throughput, and reliable data delivery, gain importance. For example, latency assurance is important in wireless sensor network applications such as factory monitoring and vehicle tracking and detection. Throughput guarantee of data gathered by sensor nodes is crucial in smart homes video surveillance in order to achieve optimum benefits of such application. Furthermore, vibration-based structural monitoring and telemonitoring of human physiological data require reliable delivery of sensor readings. The challenge for protocol designers is to find an optimal solution that provides a balance between energy efficiency and network performance.

Second, the challenge is to design a traffic adaptive protocol that can learn the current channel conditions and *adapt* themselves accordingly in order to reduce transceiver energy consumption. In most sensor network applications, traffic loads in different parts of the networks can vary significantly over time. For example, a sensor network may start with light traffic where nodes only respond to a query disseminated to the network; and when nodes in certain part of the network detect an event of interest, that part of the network begins rapid data gathering tasks. A protocol that works well under low-contention traffic, but is unable to adapt to changing traffic patterns, may consume more energy than necessary and so shorten network lifetime. Following the previous example, sensor nodes in this type of protocol often return to the idle listening phase after they have finished reporting an event. A traffic-adaptive protocol should be able to operate on varying levels of traffic conditions in order to conserve more energy, for example, letting nodes sleep instead of idle listening, when there is no activity in the channel. However, adaptation often includes complexity and protocol designers should consider how that complexity affects the processing and memory resources of sensor nodes.

The last challenge is to allow protocols to adapt to changes in network topologies in order to conserve energy, maintain network performance, and maintain the fine-granularity of information gathered from the networks. Although in most sensor network applications, nodes are designed to be stationary, these nodes may suffer temporary or permanent failures due to external factors or energy depletion, and new nodes may be added to the network at any time. A protocol designed for sensor networks should be able to detect these network dynamics and function properly without degrading the network performance.

These challenges and observations lead to the design of a *flexible-schedule-based TDMA Protocol* (FlexiTP). FlexiTP is distinguished from existing energy-efficient protocols by providing end-to-end guarantees on data delivery within energy and memory constraints of wireless sensor networks. FlexiTP also adapts to traffic fluctuations and topology changes.

1.3 Solutions

1.3.1 FlexiTP

As mentioned previously, it is desirable to prolong the lifetime of a wireless sensor network by minimising energy consumption in sensor network operations. However, the users of wireless sensor networks also need end-to-end guarantees on data delivery [24, 59]: predictable throughput for gathered data, fair access to the network for all data gathering nodes, and robust self-healing of the network when nodes join into or leave from the network, and when communication conditions change [23, 27, 58]. In practice, it has proved difficult to achieve these end-to-end guarantees on data delivery, because of the severe resource constraints (battery power and data memory) of sensor network nodes, and the hostile environments in which they must operate [19].

We propose FlexiTP that provides end-to-end guarantees on data delivery, whilst also respecting the severe energy and memory constraints of wireless sensor networks. FlexiTP achieves this balance through the following schemes:

- *FlexiTP maximises network lifetime* by putting nodes into sleep mode without disrupting the ongoing traffic and fairly distributing energy intensive operations across the network.

- *FlexiTP is scalable for large numbers of network nodes* because 1) local repair is used to recover from node and communication failures, 2) nodes minimise buffered information (i.e., information received from other nodes in the network), and 3) communication slots are reused by nodes outside each others' radio interference range.

- *FlexiTP provides reliable data delivery* because 1) scheduled communication is used to minimise communication errors from radio interference and to guarantee a communication path for each node, and 2) local repair is used to recover from node and communication faults.

FlexiTP is a self healing protocol designed for data gathering applications. FlexiTP uses a TDMA scheduling approach in which nodes only transmit and receive packets in their own time slot(s) and sleep until their slots turn up again. This approach conserves energy as nodes in sleep mode consume much less energy than in idle mode [7]. FlexiTP nodes require minimal local buffering since the network schedule is chosen so that each node forwards messages immediately. FlexiTP is designed for a static, ad hoc network, in which nodes may suffer temporary or permanent failure, and new nodes may be added to the network at any time but nodes otherwise remain in (nearly) the same location. FlexiTP adjusts to different node topologies and densities, to create an efficient, self-healing data delivery tree. Furthermore, buffering and retransmission are used to recover from a small number of lost packets.

The central contribution of the FlexiTP protocol is its synchronised and loose slot structure. Nodes can claim or remove a slot based on the current information in their lookup table (i.e., schedule) without exchanging information with any other nodes in the network prior to modifying their schedules. The node lookup table provides information for scheduling, routing, and time synchronisation purposes. In FlexiTP, it is not necessary to specify the number of slots required for the network in advance and nodes can join or leave the network at any time. This flexible slot structure makes FlexiTP strongly fault-tolerant and also highly energy-efficient. In FlexiTP, initially, all nodes in a network build their data gathering schedules. After this initial global one-off setup phase, all repair operations are local. As network density increases, the initial network setup cost and packet latency increase. Our simulations based on Mica2 Mote hardware (presented in Chapter 4) show that FlexiTP is scalable for large number of nodes (up to 400 nodes) because it sustains low initial network setup cost and low packet latency as well as maintaining good throughput for different network densities.

1.3.2 On-Demand TDMA Slot Transfer Method

The use of wireless sensor networks for gathering environmental and safety-critical data in real time is becoming prevalent in recent years. However, most protocols for data gathering and routing in sensor networks implicitly assume a regular rate of data gathering by individual nodes. While this is sufficient for sensing parameters that change slowly over time, individual nodes in a small part of a network may need to increase the rate of data gathering considerably for reporting important data in real-time. For example, in structural monitoring applications, sensor nodes are deployed to monitor vibration (e.g., wind and earthquakes) that could damage the structure of a building [121]. It is critical for sensor nodes to send their data more often to the base station when they detect event triggers such as sensor readings changing rapidly or exceeding user-specified thresholds.

Recall that FlexiTP is originally designed for periodic data gathering applications. The main challenge for this TDMA-based protocol is to allow nodes to adjust their schedule according to their current sensing requirement and traffic conditions. The distributed and flexible slot structure algorithm of FlexiTP enables nodes to modify their lookup tables during network execution. By leveraging this feature, we propose an *on-demand TDMA slot transfer* (OST) method that uses FlexiTP as a base protocol for transferring time slots from one part of the network to another part and dynamically changing the number of slots in the network (i.e., enabling the protocol to shorten or lengthen data gathering cycles). The proposed method allows FlexiTP to be traffic adaptive by reconfiguring nodes to report their data more rapidly or slowly depending on the significance and importance of their data to the end-user. This extends FlexiTP's applicability for event-sensing applications.

We propose an *on-demand aggregation* (ODA) algorithm, a TDMA-slot blockage mechanism that allows nodes to reserve some of their slots according to a desired data aggregation compression ratio so that they can still send their aggregated data to the base station if their schedules are transferred to other nodes with higher priority. The problems of on-demand TDMA slot transfer and on-demand aggregation are important properties for most sensor network applications. To the best of our knowledge, these problems have never been investigated before.

We utilise FlexiTP and the on-demand TDMA slot transfer method to support on-demand peer-to-peer communication and to perform management functions. We propose *wireless sensor network peer-to-peer* (WiSeP2P) algorithm to enable nodes to claim extra time slots from other nodes if they want to communicate with any other nodes in the network, thus further supporting adaptive and reactive sensing in different parts of a network.

Finally, we propose an adaptive policy-based management system for sensor networks, called *wireless sensor network management system* (WinMS). In WinMS, the end user predefines management parameter thresholds on sensor nodes that are used as event triggers, and specifies management tasks to be executed when the events occur. WinMS adapts to changing network conditions by allowing the network to reconfigure itself according to current events (local management) as well as predicting future events (central management).

1.4 Contributions of this Thesis

The following is a summary of the major contributions in this thesis:

- A novel *flexible-schedule-based TDMA protocol* that provides guarantees on end-to-end data delivery, while also achieving energy-efficiency. FlexiTP's distributed and loose slot structure is utilised to seamlessly support data-gathering, fault tolerance, on-demand TDMA slot transfer, and on-demand peer-to-peer communication.

- A novel *on-demand TDMA slot transfer* algorithm that allows time slots from one part of the network to be transferred to another part. Hence, it supports non-uniform and adaptive sensing in different parts of a network. The proposed algorithm also alleviates the inherent problem of over-provisioning of time slots in TDMA-based protocols. Generally, TDMA-based protocols predetermine the number of slots required for a network. This fixed frame approach leads to the under-utilisation of the channel bandwidth at low traffic loads due to unused idle slots.

- A novel *wireless sensor network peer-to-peer* algorithm that allows any node to dynamically communicate with any other node in the network efficiently, without going through the base station.

- A novel *on-demand aggregation* algorithm, the TDMA-slot blockage scheme that enables nodes in the network to perform on-demand aggregation based on current network conditions and capacity constraints.

- A *wireless sensor network-management system*, an adaptive policy-based network management system that continually observes network states and accordingly performs management functions in order to maintain the performance of the network and achieve effective networked node operations.
- FlexiTP utilises cross layer interactions between the MAC layer, the routing layer, and the application layer, to increase network performance and to limit resources needed for operation. Information sharing between layers enables nodes to route their data to intended sinks based on their schedules, and to adjust their schedules dynamically and efficiently according to network sensing requirements, network communication constraints, and changes in network topology.

1.5 Thesis Overview

This thesis is organised as follows. Chapter 2 reviews related work in the fields of wireless sensor network protocol design: medium access control protocols and traffic-adaptive protocols. This chapter also includes background on cross-layer design optimisations in wireless sensor networks.

Chapter 3 summarises the assumptions we made about wireless sensor networks. Furthermore, it describes our simulation methodology and tool.

In Chapter 4, we describe the design, implementation, and evaluation of FlexiTP. We also provide performance comparisons between FlexiTP and other protocols.

In Chapter 5, we describe how our proposed on-demand TDMA slot transfer algorithm can be used to enable a schedule-based protocol to adapt to dynamic traffic patterns of wireless sensor networks. We also show that the proposed protocol performs much better than existing traffic-adaptive protocols both in terms of energy-efficiency and performance in supporting non-uniform and adaptive sensing.

In Chapter 6, we present a novel peer-to-peer algorithm for wireless sensor networks that utilises FlexiTP to enable nodes to communicate with any subset of nodes in the network in order to support reactive sensing in different parts of a network.

In Chapter 7, we present our wireless sensor network management framework. We discuss the role of FlexiTP and its functionalities for monitoring and managing wireless sensor networks.

Finally, in Chapter 8, we identify challenges and limitations of our family of protocols, present open problems, and discuss future directions in wireless sensor networks.

Chapter 2

Literature Review

He who learns but does not think, is lost. He who thinks but does not learn, is in great danger.

Confucius

In this chapter, we review the relevant work on MAC protocols for wireless sensor networks that serves as a background to our research.

2.1 MAC Protocols for Sensor Networks

MAC protocols designed for wireless sensor networks can be categorised into two groups: contention-based and scheduled-based protocols. The contention-based MAC protocols use carrier sense multiple access (CSMA) for node transmission in which a node that intends to transmit, listens to the medium and is allowed to transmit if no other node is transmitting. In contrast, scheduled-based MAC protocols attempt to coordinate node communication by scheduling when each node should listen, receive, or transmit. Our proposed protocol, FlexiTP, falls into this category. The most common schedule-based approach is TDMA. TDMA separates nodes in time, which means that nodes own a time slot for transmitting and receiving a message without contending for medium access. This chapter briefly reviews several prominent contention-based MAC protocols for wireless sensor networks. The chapter focuses on scheduled-based MAC protocol for sensor networks.

2.1.1 Contention-based MAC Protocols

The main advantage of contention-based MAC protocols is that they can allocate resources on-demand, and so they can adapt to traffic fluctuations and changes in node density easily. For example, new sensor nodes can join the network quickly because they do not need to build schedules to begin their activities. Contention-based MAC protocols are typically simple to implement because sensor nodes do not need to exchange information with their neighbours prior to sending their messages. Therefore, in low contention networks, contention-based MAC protocols can ensure low latency and high throughput [35]. Furthermore, since sensor nodes do not share and maintain neighbourhood state, contention-based protocols minimise memory usage and storage requirement.

The main challenge for contention-based MAC protocols is to reduce the energy consumption caused by collision, overhearing, and idle-listening [37], that happens due to a lack of communication coordination. Another challenge is to attain fairness in which nodes have equal priorities or rights to access the channel.

Contention-based MAC protocols can either use a random access or slotted access [65] approach. The random access approach characteristically requires nodes to listen to the channel for activity on regular basis and they need to contend for the channel if they wish to send a packet. Examples are Berkeley-MAC (B-MAC) [86] and Energy-Aware Data-centric (EAD) protocol [15]. The slotted access approach improves the previous CSMA-style access by introducing the RTS/CTS handshake scheme to reduce overhearing, and implementing a fixed duty cycle scheme where nodes alternate their active and sleep cycles. Examples are Sensor-MAC (S-MAC) [123], Timeout-MAC (T-MAC) [114], Scheduled Channel Polling-MAC (SCP-MAC) [124].

B-MAC

The B-MAC protocol [86] uses CSMA for communication where sensor nodes send their messages if they do interfere the current ongoing packet transmissions. To reduce energy consumption, sensor nodes schedule themselves to sleep based on the duty cycle of the network. This goal is achieved by forcing every sensor node to listen to its neighbours' control messages to check whether it is the intended receiver of the packet before putting itself to the sleep mode. This scheme increases idle listening energy waste.

EAD

The EAD protocol [15] uses CSMA with RTS/CTS handshake mechanism to reduce collisions and hence conserve energy. EAD provides further energy efficiency by rotating nodes in the network to act as backbone sensors based on their residual energy. These backbone sensors are responsible for in-network data processing and packet routing, while the rest of nodes in the network sleep to conserve energy. EAD provides high energy saving for event-driven applications where nodes only report events upon their detection.

EAD uses a forwarding-to-parent routing scheme where a tree is constructed at the initial network setup and so every node in the network knows where to route its packet during sensor network execution. This simple routing scheme is suitable for static sensor networks without (much) node or communication failures since it sustains high throughput and good energy efficiency [15].

S-MAC and T-MAC

Slotted protocols such S-MAC [123] and T-MAC [114] periodically put nodes into sleeping mode after overhearing other nodes' communication, as illustrated in Figure 3 and Figure 4 respectively. This approach introduces high latency because routers on the path to the base station that are more than one or two hops away from a source node are not

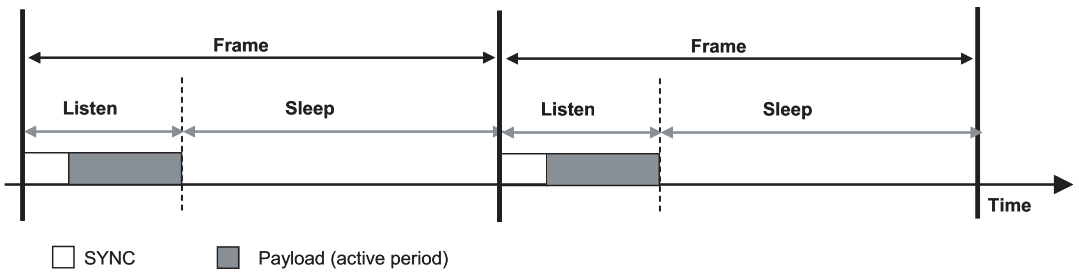

Figure 3: S-MAC frame format and slot structure. Each node maintains a fixed duty cycle of sleep and active.

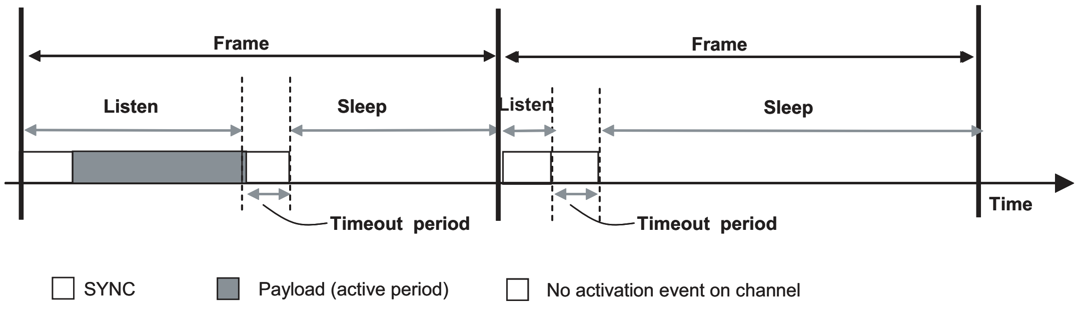

Figure 4: T-MAC frame format and slot structure. Each node maintains a variable length of sleep and active cycle.

notified of the ongoing traffic, and therefore the source node must wait for the next active period before it can transfer data. In addition to increased latency, this approach sacrifices per hop fairness. T-MAC attempts to alleviate this problem by allowing sensor nodes to dynamically adjust their sleep and active cycles based on communication of neighbouring nodes.

S-MAC and T-MAC employ the RTS/CTS handshake scheme to reduce collisions. However, the overhead of RTS/CTS is quite high for typically small-sized data packets in wireless sensor networks [94]. In S-MAC, a node that has more data to send can monopolise the radio channel. This is unfair for other nodes which have short packets to send but need to wait for the completion of transmission of the long packet.

SCP-MAC

The SCP-MAC protocol [124] improves the contention scheme of S-MAC and T-MAC to conserve more energy by reducing the total energy consumption. SCP-MAC divides a frame into slots and each slot includes a contention window. If a node wants to send a message, it randomly backoffs within this contention window and then listens to the activity in the channel. If the channel is clear, the sender will start sending a preamble. Other potential senders that overhear this preamble will postpone their own transmissions. When the contention window expires, the winning sender, if any, begins transmitting its message while the rest of nodes in the network wake up and perform a carrier sense to determine if they are the intended receiver of the ongoing message transmission. If the channel is clear from message transmissions, then these nodes switch to sleep mode. In this way, SCP-MAC conserves more energy than S-MAC and T-MAC.

2.1.2 Scheduled-based MAC Protocols

TDMA differs from contention-based protocols in that each node in the network owns a time slot to send its data and nodes do not interfere with each others transmissions. Thus, TDMA-driven protocols guarantee collision-free communication and reduce idle listening. This results in significant energy savings. By scheduling node communication, TDMA-driven protocols conserve energy since nodes not participating in message transmission may sleep until their next communication activity: transmit, forward, or receive a message. Since TDMA-driven protocols coordinate which nodes should utilise the channel at any time, fairness for all nodes in the network can be guaranteed.

The main challenges of TDMA approach are determining collision free slots to be assigned to nodes in multiple hop networks, synchronising nodes in the network, and adapting to changes in traffic loads and network topologies. Collision-free time slot assignment is a complex process as sensor nodes need to know and maintain the state of the nodes in their two-hop neighbourhood and this neighbourhood information exchange incurs energy and memory overhead [104]. The high initial overheads to set up and distribute a schedule through the network, however, can be amortised over the network lifetime and eventually balanced by improved throughput and energy efficiency [94]. TDMA-driven protocols should also ensure accurate time synchronisation so that nodes' time slots do not overlap [38, 66, 105].

The nature of predetermined frame length (i.e., the number of slots in a frame) of TDMA-approach cause it to under-utilise the channel at low traffic loads (light traffic generation). This inflexibility leads to over-provisioning of time slots and decreased throughput due to idle slots. Furthermore, TDMA-driven protocol should also minimises the effect on increased latency and limited throughput in high traffic loads, which is normally acceptable for delay-tolerant networks such as environmental monitoring applications. A good TDMA-driven protocol should be able to adapt to changing traffic fluctuations, from light traffic to heavy traffic loads and vice versa.

When the network topology changes (e.g., a new node added to the network, broken links due to radio interference), a good TDMA-driven protocol should be able to stabilise the network to the correct schedule. In the case where a sensor node dies, a TDMA-driven protocol should be able to remove idle slots in the network and hence preventing unnecessary delays.

Scheduled-based MAC protocols can be classified into TDMA-access or hybrid access. The most common TDMA-access approach uses a frame-based communication scheme in which sensor nodes are assigned collision-free slots based on their two-hop neighbourhood information. A hybrid access-style uses both CSMA and TDMA scheme for communication. Examples of TDMA-driven protocols are, EYES MAC (EMACS) [116], Lightweight MAC (LMAC) [115], Adaptive Information-centric Lightweight MAC (AI-LMAC) [21], Self-Stabilising TDMA (SS-TDMA) [64], Bit Map Assisted (BMA) [71], Power Aware Clustered TDMA (PACT) [84], and the protocol by Arisha et al. [10]. Examples of hybrid protocols are Traffic-Adaptive MAC (TRAMA) [89], Flow-Aware Medium Access (FLAMA) [88], Zebra-MAC (Z-MAC) [94], Pattern-MAC (PMAC) [128], and Crankshaft [47].

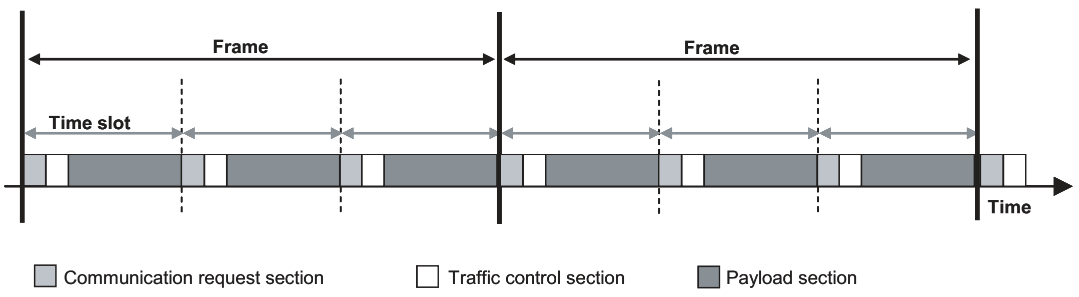

Figure 5: EMACS frame format and slot structure. Each frame consists of a number of slots and each slot has three sections: communication request, traffic control, and payload.

EMACS

The EMACS protocol [116] divides time into frames and each frame consists of a number of time slots. The base station initiates the time slot assignment process by broadcasting a control message. Each sensor node in the network then selects a slot randomly and begin sending during that slot. Neighbours of a node may select the same slot and this situation results in collisions. Upon detecting collisions, the node re-selects another slot. This slot selection scheme prevents nodes within the interference range of each other from selecting the same slot.

Figure 5 shows that EMACS time slot has three sections: communication request, traffic control, and payload. Each node in the network always listens to the communication and traffic section of all packets sent by its neighbours. If a slot non-owner wants to send data, it can request for the slot during the communication request section of that slot. The slot owner can give up its ownership to the requestor through the traffic control section of the slot. Data is transmitted during the payload section of the slot.

In EMACS, sensor nodes can be active, passive, or dormant throughout their lifetime in the network. A node is in an active state when it fully utilises its own slot by transmitting data in that slot. On the other hand, a passive node is a slot non-owner that requests the slot from the slot owner. A node switches to the dormant state when it has no activity to perform and so it goes to sleep to conserve energy. The main advantages of EMACS are that it allows sensor nodes to conserve energy when they are not needed and provides adaptability to changing traffic conditions, although, at the cost of idle listening.

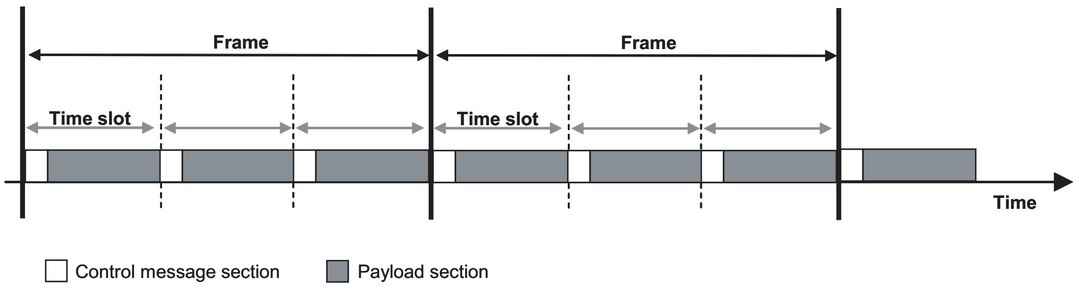

Figure 6: LMAC frame format and slot structure. Each frame is divided into time slots and each time slot consists of control message and payload section.

LMAC

The LMAC protocol [115] divides a frame into time slots and each time slot has two sections: control message and data payload, as illustrated in Figure 6. Time slot assignment is similar to EMACS where sensor nodes select a slot that is not owned by their neighbours. LMAC differs from EMACS in two ways. First, each sensor node in the network is a slot owner and the node can only transmit data at its own slot. Second, each sensor node must operate in the active state at all times. The main drawback of LMAC inflexible slot structure scheme is that it cannot adapt to changing traffic loads. Furthermore, LMAC increases idle-listening overhead since nodes must always listen to the control sections of all slots in a frame in order to allow listening nodes to receive data intended for them and to accommodate new nodes to join the network anytime.

Both LMAC and EMACS adopts a simple time slot assignment, but it comes at the cost of high initial network setup overhead and high network configuration delay. These overheads are mainly due to frequent collisions among neighbouring nodes before these nodes manage to attain collision-free slots during the time slot assignment phase.

AI-LMAC

The AI-LMAC protocol [21] is a variant of LMAC with the same slot structure as LMAC. AI-LMAC attempts to address several weaknesses of LMAC. AI-LMAC addresses the problem of adaptability to changing traffic conditions by allowing sensor nodes to vary the number of slots they own based on current traffic loads. Each sensor node in the network observes the traffic conditions by recording statistics of its neighbourhood activity in a Data Distribution Table (DDT). AI-LMAC organises sensor nodes into a tree-hierarchy with the parent-child relationship. Based on DDT, a parent node decides when to give more time slots to its children and when to take time slots from its children. The advantages of this scheme are that it ensures fairness among children in a sub-tree and prevents buffer overflow (bottleneck) at the parent node. Instead of constant idle listening approach like LMAC, AI-LMAC nodes only transmit their control message at the first slot that they own. Hence, AI-LMAC reduces idle-listening overhead.

The main drawback of AI-LMAC is its DDT scheme. Creating and maintaining DDT is computationally and energy expensive for resource-constrained sensor nodes. Furthermore, the memory cost of DDT may grow over time, which is undesirable for memory-constrained sensor nodes.

SS-TDMA

The SS-TDMA protocol [64] is designed to operate on a regular grid topology such as rectangular, hexagonal, and triangular grids. SS-TDMA assumes a two-band model for communication. The communication range is the distance to which a node can communicate with other nodes with a high probability. The interference range is the distance to which a node can communicate with other nodes with a low probability. The collision group of a node is the set of nodes that are in the interference range of the node. SS-TDMA assigns a time slot to each node in the network based on the collision-group information, for example a node labeled *A* and a node labeled *B* can use the same time slot as long as *A* is not within the collision-group of *B*. The TDMA algorithm proposed in SS-TDMA is self-stabilising. It tolerates faults such as nodes that are improperly initialised, slots assigned to corrupted nodes, and nodes with clock drifts. SS-TDMA uses a diffusing computation to re-validate the slots assigned to a node. It freezes a node from transmitting messages if the revalidation message is not received. When the revalidation message is received, the node can continue to transmit messages. The main drawback of SS-TDMA is its constraint on the location of the nodes, which is impractical for many sensor network applications. Furthermore, SS-TDMA assumes that communication range, interference range, and collision range can be reliably estimated, which is not possible in most sensor network applications.

Clustering-based TDMA Protocols

The protocol by Arisha et al. [10], the BMA protocol [71], and the PACT protocol [84] use a clustering-based TDMA protocol where nodes are grouped into clusters and use TDMA approach for communication. Nodes within a cluster can share information with each other to perform a distributed task. This clustering approach provides an advantage of scalability because each cluster in a network operates based on its local states.

PACT [84] uses passive clustering to create a communication network of cluster heads and base stations. The role of cluster heads and base stations is rotated based on the residual energy at nodes. To prolong the network lifetime, only a subset of nodes (cluster heads and base stations) are active. At the start of a TDMA frame, these active nodes get preference in claiming a sequence of communication slots. On the other hand, BMA [71] protocol uses the LEACH approach [49] to manage cluster formation and rotation. At the start of each TDMA frame, nodes send one bit of information to their cluster head indicating whether or not they have data to send. Based on this information the cluster head determines the slot assignment for all nodes within its transmission range (i.e., cluster). PACT and BMA rotate the role of cluster heads to amortise the overheads of the TDMA scheduler. However, this clustering-based TDMA method requires the cluster heads to rebuild a schedule for all nodes within their clusters and this coordination activity increases energy consumption. Furthermore, when the network topology changes, cluster formation must be refreshed and reconfigured, and hence BMA and PACT suffer the energy overhead from schedule rebuilding.

The protocol by Arisha et al. [10] arranges each cluster head in the network to send traffic and battery-level information from the nodes in its cluster to the base station. Based on this information the base station builds a schedule (transmission and sleep) for nodes

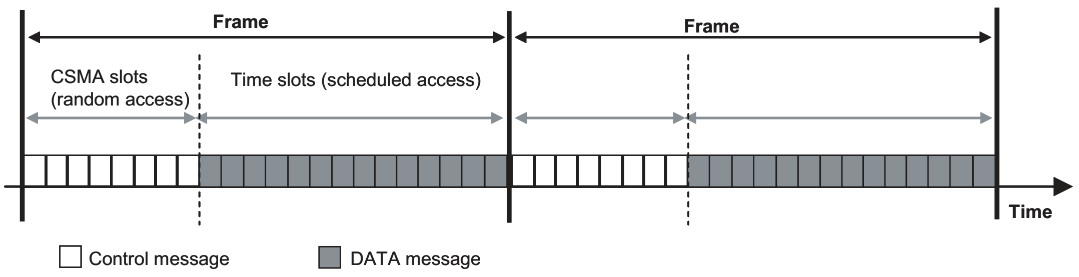

Figure 7: TRAMA frame format. A frame consists of CSMA and TDMA time slots.

within each cluster. This approach is not scalable since cluster heads need to directly communicate with the base station. Furthermore, the protocol uses a fixed length TDMA frame and so the maximum number of nodes must be known before deployment.

TRAMA

The TRAMA protocol [89] organises time into frames and each frame consists of a number of slots dedicated for CSMA and TDMA communication, as shown in Figure 7. At the beginning of a frame, sensor nodes perform a CSMA-style communication for sending periodic control messages, then sensor nodes use TDMA slots for transmitting, forwarding, or receiving data. To minimise clock drifts among nodes in the network, each sensor node always listens to the last data message of all nodes within its one-hop neighbourhood in order to synchronise its schedule with neighbours' schedules.

TRAMA uses a distributed election scheme based on traffic information at each node to determine which node can transmit at a particular slot. To achieve this goal, TRAMA proposed three sub-protocols: the Neighbour Protocol (NP), the Schedule Exchange Protocol (SEP), and the Adaptive Election Algorithm (AEA). The role of NP is to allow sensor nodes to share information about network topology. SEP allows nodes to exchange information about traffic conditions in their two-hop neighbourhood. AEA uses a distributed hash function to determine a collision-free slot for a data transmission based on information provided by NP and SEP.

There are three main advantages of TRAMA design. First, sensor nodes can adjust their schedules according to traffic conditions, through regular neighbourhood information exchange. If a node has few packets to send, it may release its slot for the remainder of the frame to other nodes that have many packets to send. This scheme results in a high bandwidth utilisation. Second, the scheduled data transmission feature of TRAMA prevents collisions and hence saves energy. Lastly, TRAMA provides a high percentage of sleep duration by grouping slots that a sensor node gives up at the end of a frame.

TRAMA suffers several drawbacks that are inherent to typical scheduled-based protocols. First, when network topology changes, sensor nodes waste energy due to incorrect schedules and this state can degrade the performance of the network. Second, TRAMA synchronisation scheme where all sensor nodes continually listen to their one-hop neighbours' last data transmission, increases energy consumption. Third, since sensor nodes are always in active mode during the CSMA slots of each frame, TRAMA wastes energy for idle listening. Fourth, TRAMA does not ensure active slots to be contiguous and this approach may increase energy consumption for radio switching. Finally, TRAMA's distributed slot selection scheme based on activities of two-hop neighbours results in high memory usage, high latency, and high algorithmic complexity for creating and maintaining neighbourhood table.

FLAMA

The FLAMA protocol [88] extends TRAMA in an attempt to reduce idle listening overhead (for neighbourhood traffic information exchange) of TRAMA. FLAMA does this by leveraging traffic pattern characteristics or traffic prediction models of sensor network applications. For example, in periodic data gathering applications, sensor nodes forward data towards the sink. FLAMA uses the data gathering tree information to determine the next-hop router of a node's transmission. FLAMA views this one-hop traffic information as *flow* and uses the *flow* for setting up transmit, receive, and sleep schedules of sensor nodes in the network. FLAMA adopts a pull-based mechanism instead of TRAMA push-based mechanism. TRAMA requires sensor nodes to broadcast traffic information regularly in order to preserve their schedules. In contrast, FLAMA only requires sensor

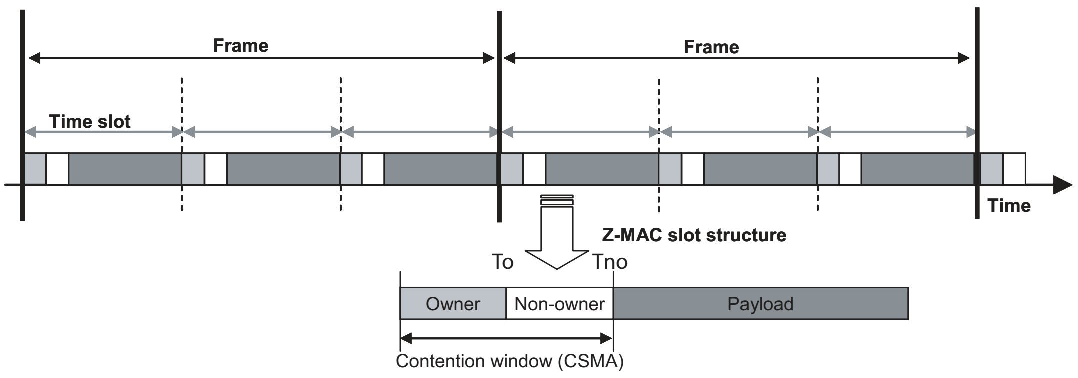

Figure 8: Z-MAC frame format and slot structure with built-in preference for slot owners.

nodes to exchange traffic information during the random access period of a frame. Furthermore, instead of calculating the priorities of nodes in two-hop neighbourhood like in TRAMA, FLAMA provides adaptability to traffic fluctuations by utilising local traffic information at a node to determine the number of slots required by that node. For example, nodes that forward more data, get more time slots.

Z-MAC

The Z-MAC protocol [94] is a hybrid MAC protocol that starts off as CSMA and switches to TDMA if the traffic load in the network increases. In Z-MAC, each node in the network executes a distributed slot selection algorithm to get a collision-free TDMA slot based on its two-hop neighbourhood schedule information.

Figure 8 depicts Z-MAC slot structure and each slot consists of a small contention window, followed by payload/data transmission. Sensor nodes in the network are in the listen mode during the contention window of all slots in a frame to determine whether or not their neighbours send a message in a particular slot. The slot owner has a higher priority to send message, if any, than non-owners. When the slot owner has no data to send, it allows other nodes to use its slot. For example, if a node has data to send (i.e., sender), it checks the current slot in the network. If the sender is the owner of the slot, it back offs within *To* period, else it back offs between *To* and *Tno*. Afterwards, the sender performs CSMA (using B-MAC's random access approach) and if the channel is clear, it proceeds to send its data.

When multiple nodes within a sub-network attempt to send messages at the same time, collisions often occur due to hidden terminals. In such high contention networks, Z-MAC uses explicit congestion notification (ECN) messages to limit the occurrence of hidden terminals. When a sender detects heavy traffic load, it broadcasts the ECN message to its direct neighbours. These direct neighbours further propagate the ECN message to their neighbours, hence, nodes in a two-hop neighbourhood of the sender get notified of the ongoing traffic. These neighbours then can only attempt to transmit a message during the scheduled slot of itself and its direct neighbours. These neighbours resume their previous states when they receive no more ECN messages.

The main advantage of Z-MAC is that it allows sensor networks to rapidly adapt to changing traffic conditions. Z-MAC operates toward CSMA communication in light traffic generation and toward TDMA communication in high traffic generation. This flexibility of varying node state can conserve energy in sensor networks.

There are several drawbacks of Z-MAC. First, Z-MAC suffers inherent problems of TDMA protocols, that are, high energy cost and high delay in setting up node schedule. Furthermore, network topology changes such as node redeployment and node death further complicate the schedule maintenance. If the network topology changes dramatically, Z-MAC needs to re-run the costly initial network setup. Second, each Z-MAC time slot includes a small contention window in every slot and this approach leads to added network latency and increased energy consumption under high contention networks. Third, the ECN scheme can limit hidden terminals in a local contention, however, at the cost of increasing energy consumption for propagating ECN messages and extending delay for moving toward TDMA operation, on an already heavily-loaded networks. Lastly, Z-MAC inherits limitations of B-MAC since it uses the underlying random access scheme of B-MAC.

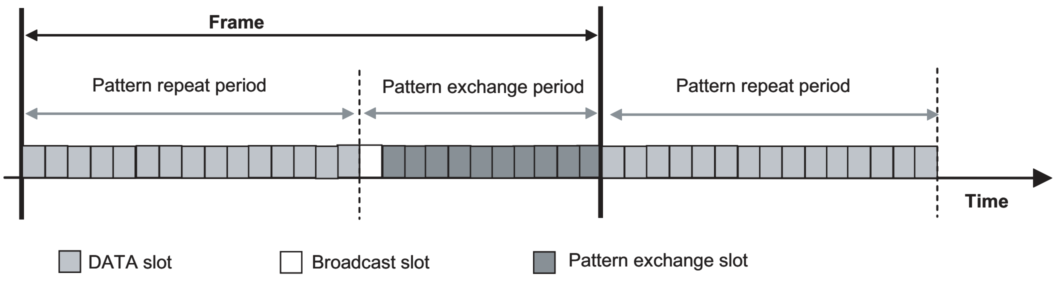

Figure 9: PMAC frame format: pattern repeat section (data slots) and pattern exchange section (broadcast slot and pattern exchange slots).

PMAC

Similar to TRAMA and Z-MAC, the PMAC protocol [128] allows sensor nodes to dynamically adjust their sleep and wakeup cycles based on current traffic loads. Sensor nodes that have more messages can claim more slots than nodes with no message to send. PMAC differs from TRAMA and Z-MAC in the way sensor nodes adapt to traffic changes, that is, based on information given by sensor nodes about the expected traffic events. Sensor nodes in the network propose their duty cycles for the next frame through exchanging patterns with their local neighbourhood.

Figure 9 illustrates PMAC frame format, which consists of a number of slots for data transmission and a number of reserved slots for pattern exchange task. A sensor node's pattern is represented as a bitmap of time slots during which it plans to sleep (bit cleared) or wake up (bit set) during the next frame. During the pattern repeat period of a frame, sensor nodes follow the sleep/wake schedule they have announced in the previous frame. Nodes wake up during their scheduled transmission slots and also wake up during scheduled transmission slots of their neighbours. PMAC uses RTS/CTS/DATA/ACK scheme for data transmission to avoid collisions and provide reliability. To conserve energy, a sensor node may wakeup and listen for a short period and switch to sleep mode if there is no activity in the channel. At the end of the pattern repeat period, sensor nodes start announcing their chosen patterns. During this pattern exchange period, all sensor nodes with traffic proposal contend for the pattern exchange slots to send their pattern information. The pattern exchange period must be long enough to allows all nodes to successfully

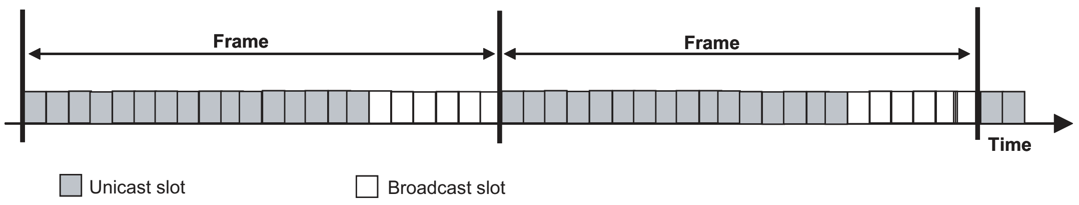

Figure 10: Crankshaft frame format: unicast slots and broadcast slots.

send their pattern information and so the total number of pattern exchange slots must be at least as many as the maximum number of neighbours.

The main advantage of PMAC is that it provides good adaptability to changing traffic loads. Furthermore, PMAC schedule assignment through traffic announcement within a local neighbourhood is simple to implement. However, the schedule assignment scheme is prone to errors, for example, some sensor nodes may receive an incorrect pattern due to interference signals while other nodes successfully receive the correct pattern. This problem results in schedule inconsistencies among sensor nodes in the network and leads to increased energy waste due to collisions, idle listening, and wasted transmissions. PMAC requires sensor nodes to perform a pattern exchange whenever their traffic in the next frame is expected to change. If the pattern exchange happens frequently, it significantly increases energy consumption because of the processing power and active power involved for updating the pattern.

Crankshaft

The Crankshaft protocol [47] shares the same scheduling approach as PMAC in which sensor nodes in the network schedule their receive slots and hence reduces energy waste due to overhearing. Furthermore, Crankshaft uses the efficient contention scheme of SCP-MAC for node communication.

Figure 10 shows the frame format of Crankshaft. Crankshaft divides time into frames and each frame consists of a number of slots for unicast communication and for broadcast communication pattern. A frame starts with all the unicast slots, followed by the broadcast slots. Crankshaft allocates one unicast slot in a frame for every node in the network. The unicast slot owned by a node corresponds to the node's MAC address.

In the unicast period of a frame, sensor nodes that are scheduled to receive messages wake up at their unicast slot, while the rest of nodes in the network remain sleeping. During the unicast slot of a node, any senders that want to send a message to that node contend for the channel. Since a node's MAC address represents the node's unicast slot ID, these senders can determine the start of intended receiver's unicast slot in a frame. The sender that wins the contention then can start transmitting its message. During the broadcast period of a frame, all sensor nodes wake up and listen for incoming messages at all broadcast slots. Any node with a message to send contends for the channel and transmits the message if the channel is clear.

The Crankshaft's unicast slot structure is similar to that of Z-MAC in the way it incorporates a contention window in every unicast slot before the actual data transmission within that slot. Crankshaft, however, reduces idle-listening and overhearing overhead as potential senders only awake during the contention period of a unicast slot. Note that Crankshaft requires sink nodes in the network to listen to all unicast slots of every frame in order to reduce the bottleneck around the sinks. To accomplish this goal, the end-users must ensure that these sinks have more energy power.

Crankshaft is best suited for environmental monitoring applications in which network lifetime is the primary goal. This is because Crankshaft can lengthen the network lifetime of dense sensor networks by reducing the overhead of overhearing neighbours' activities and by utilising the efficient channel polling and contention scheme of SCP-MAC. On the other hand, the energy-efficient slot structure of Crankshaft comes at the cost of inflexibility, which limits the applicability of the protocol for other types of sensor network applications.

2.1.3 Two-Radio-Based MAC Protocols

Miller and Vaidya [83] proposed a two-radio based MAC protocol that addresses end-to-end guarantees on data delivery in terms of memory and energy efficiency. The protocol uses a second low power radio to allow senders to wake receivers if a specified number of packets are buffered and so a buffer overflow can be avoided. The protocol also determines an optimal period of the wakeup to minimise energy consumption. On the other hand, the protocol developed Yuan et al. [125] utilises the second radio to adjust node modulation level. The proposed protocol uses the base station to adjust the modulation level (bits per symbol) of each cluster head in a network based on the cluster head's feedback in order to make good trade-off between energy consumption and transmission quality. The major drawbacks of this protocol are the queuing process at cluster heads and the centralised decision making. The number of packets arriving at cluster heads is not controlled and so the packets will be dropped if the buffer overflows, resulting in latency and packet loss.

2.1.4 Discussion

In this section, we highlight key strengths and weaknesses of protocols reviewed in this chapter. We also describe main differences in approaches taken by reviewed protocols and compare related protocols with FlexiTP.

CSMA Versus TDMA

There is no single best MAC protocol that fit all sensor network applications. Most protocols are designed to be application specific. The simplest approach in rationalising the decision between the contention-based or scheduled-based MAC protocol is to characterise the traffic pattern and behaviours of sensor network application. Data gathering applications constantly generate traffic and so scheduled-based MAC protocols are the more appropriate choice because nodes always have data to send and so dedicated time slots are not wasted. A light traffic generation is identical to event-detection applications where sensor nodes typically only send few messages and stay dormant most of the time until they detect an event. When an event of interest occurs, a part of the sensor network may enter high-contention state as relevant sensor nodes generate a large number of messages. Contention-based MAC protocols for this type of application provide the best benefits as they offer flexibility to handle different traffic loads and they promote energy conservation by letting nodes sleep when there are no significant activities in the network.

The main limitation of existing contention-based protocols such as S-MAC, T-MAC, and SCP-MAC is that they provide no implicit mechanism to ensure fairness in the channel utilisation. In contrast, TDMA-based protocols guarantee fair access since no node can monopolise the medium and nodes have their own slots to transmit data. Furthermore, nodes do not need to contend for the medium to send data, so that the time it takes for the data to reach from a source node to the base station is reduced and predictable.

Assumed Network Topology

SS-TDMA is the only scheduled-based MAC protocol that is designed to be grid-application specific, while the rest of protocols can cater to any network topology.

TDMA Slot Structure

All reviewed scheduled-based MAC protocols except PACT and BMA divide their frames into slots. EMACS, LMAC, and AI-LMAC share the idea of scheduling transmission slots of nodes in the network and having receivers to stand by at beginning of every transmission slot. They also have similar time slot assignment approach where sensor nodes randomly selecting a slot not owned by their neighbours. EMACS and LMAC require sensor nodes to listen to control messages of their neighbours and this scheme incurs large idle listening overhead. Furthermore, in EMACS and LMAC, sensor nodes maintain a fixed schedule throughout their lifetime. In contrast, AI-LMAC allows sensor nodes to have more of fewer slots in a frame and hence supporting adaptability to traffic fluctuations.

Crankshaft and PMAC take an opposite approach to EMACS, LMAC, and AI-LMAC, they schedule receive slots of nodes in the network with senders contending for the channel. The benefit of this scheme is that sensor nodes only wake up during their own scheduled slots to check if there is any incoming message for them. This approach significantly reduces overhearing energy-waste compared to sender-based schemes like EMACS and LMAC, where nodes perform listening on every slot in a frame. Crankshaft conserves more energy than PMAC because it utilises the energy-efficient contention scheme of SCP-MAC, while PMAC uses the energy-intensive CSMA/CD scheme.

Traffic-Adaptive Protocols

PMAC outperforms Crankshaft from adaptability point of view. In PMAC, sensor nodes observe current traffic conditions and dynamically adjust their schedules. Hence, PMAC is a traffic-adaptive protocol. Other traffic-adaptive protocols such as TRAMA, FLAMA, and Z-MAC also allow nodes to adapt to changing traffic loads, from light to heavy traffic generation and vice versa. These protocols allow sensor nodes to utilise slots not assigned within the two-hop neighbourhood. TRAMA and FLAMA require sensor nodes to calculate the priority of their one-hop and two-hop neighbours in order to determine the neighbourhood traffic load in each frame. This constant neighbourhood traffic exchange drains a lot of energy for processing and transmissions, which can decrease the network lifetime. In contrast, Z-MAC only sends traffic notification messages (through ECN exchange) when the level of contention at a sensor node increases. Additionally, TRAMA builds a schedule when a node has data to send and this random scheduling scheme increases queuing delays. Z-MAC also introduces added delay due to a small contention period in each time slot in the network.

FlexiTP Versus Existing MAC Protocols

Based on our literature review of MAC protocols for sensor networks, most protocols are designed to be non-adaptive. Limited works have been done on implementing a traffic-adaptive and fault-tolerant protocol for data gathering application that can reconfigure the network when the traffic load in different parts of the network changes or when the topology of the network changes (node density or movement), in the energy-efficient manner. Although contention-based protocols can easily adapt to changes in traffic patterns in theory, they incur high energy overhead due to message collisions and retransmissions. On the other hand, TDMA-based protocols naturally provide collision-free communication, however, at the cost of reduced flexibility.

TDMA-based protocols with a rigid slot structure suffer from utilisation problem during periods of light traffic generation and network recovery problem when the network topology changes. The main challenge of this scheme is to allow nodes to adjust their schedule according to their current sensing requirement. Several hybrid protocols such as TRAMA, FLAMA, PMAC, and Z-MAC have been proposed to alleviate this problem, however, they do this at the expense of high energy consumption for sharing traffic information and idle listening overhead for maintaining high-duty cycle. Our proposed protocol, FlexiTP, attempt to address these disadvantages.

FlexiTP has a flexible slot structure in which nodes modify their schedules (e.g., add or remove slots) without exchanging information with any other nodes in the network. Previously reviewed scheduled-based protocols use a fixed length frame and so the maximum number of slots must be known before deployment. In contrast, FlexiTP slot structure can be changed dynamically during the runtime of the application to accommodate changes in traffic and network topology. In Chapter 4, we describe FlexiTP approach in details.

In contrast to EAD, FlexiTP is designed for periodic data gathering applications where each node has data to send at regular intervals. EAD uses CSMA/CA for data transmission, while FlexiTP eliminates RTS/CTS handshake during data gathering and this results in low packet overhead. Similar to EAD, FlexiTP uses forwarding-to-parent routing scheme, a simple routing scheme that is proven to maintain good network performance in static sensor networks [15].

FlexiTP provides flexibility for periodic gathering networks to move towards event-sensing applications and to return to periodic gathering tasks anytime. This goal is achieved 1) by requiring resource-constrained sensor nodes to analyse their local traffic requirements and 2) by using the more powerful node (i.e., the base station) to analyse the global traffic requirements. FlexiTP transfers slots among nodes systematically using scheduled (TDMA) time slots of nodes in the network. In FlexiTP, nodes with high-priority data (i.e., large volume of important messages) can claim more slots from nodes with low-priority data, hence, they can transmit their messages at a higher rate than their normal reserved slots. Chapter 5 describes this on-demand TDMA slot transfer method in details. FlexiTP with the on-demand TDMA slot transfer method allow a sensor network to conserve energy while providing responsive performance as it can operate over a range of conditions.

Similar to Z-MAC, FlexiTP performs a time slot assignment once at the initial network setup. This high initial overhead can be amortised over the network lifetime and eventually balanced by improved throughput and energy efficiency [94]. Z-MAC needs to re-run the time slot assignment when a network experiences significant topology changes. Unlike Z-MAC, to maintain correct state after node and communication faults, FlexiTP allows sensor nodes to reconfigure their schedules during the application runtime. Additionally, FlexiTP can remove unused slots, say due to a node death, and so no slots wasted on idle listening.

Recall that the time slot assignment of EMACS, LMAC, and AI-LMAC incurs high delay and high energy consumption because sensor networks stabilises slowly due to a large volume of data retransmissions. FlexiTP addresses this problem by using a token-passing mechanism with token acknowledgement for reliability, instead of relying on pure CSMA approach. Furthermore, FlexiTP provides superior energy efficiency to LMAC because nodes are only active in their own time slots and sleep when they are unneeded.

Recall that in TRAMA, nodes build their schedule when they have data to send and this scheme increases queuing delays. In contrast, FlexiTP reduces queuing delays as schedules are assigned to nodes at the time of initial network setup and nodes maintain this schedule throughout their lifetime until the network topology changes. In this way FlexiTP also avoids the need for a centralised scheduler. Furthermore, TRAMA does not attempt to make a sensor node's receive and transmit slots contiguous and this may increase energy consumption for radio-switching. FlexiTP uses the depth-first-search scheduling to minimise the frequency of radio switching. In addition, FlexiTP has a simpler algorithm to establish a schedule compared to TRAMA.

The two protocols proposed in [83] and [125], described in Section 2.1.3, that did address end-to-end guarantees on data delivery issues in wireless sensor networks, were different from our approach as they used a two-radio architecture. Even though a direct comparison between the performance of FlexiTP and these protocols has been performed, the different objectives between FlexiTP and these protocols makes this direct comparison difficult.

Time synchronisation is an important issue that need to addressed by any TDMA-driven protocol and sensor nodes typically consume bandwidth and energy resource to maintain synchronisation. FlexiTP performs time synchronisation locally with each parent synchronising its children in order to minimise the cost of time synchronisation while also maintaining an accurate time synchronisation.

2.2 Cross Layer Design Optimisations in Sensor Networks

Recently there has been increased interest in protocols for wireless sensor networks that rely on significant interactions between different layers. The knowledge about the wireless medium are shared among physical layer, medium access control layer, and routing layer in order to provide efficient methods of allocating network resources and hence improves the performance of wireless sensor networks. In this thesis, we focus on reviewing solutions that integrate MAC protocol with routing protocol.

Safwat et al. [98] proposed two routing algorithms that utilise the information about successful/unsuccessful clear-to-send (CTS) and acknowledgement (ACK) reception. Cui et al. [29] proposed a variable time slot length solution in which the slot duration of each sensor node is determined based on several criteria such as the traffic pattern of the node and the distance between the node and each of its neighbours in the network. The proposed solution takes these values as inputs for calculating the optimum number of slots assigned for each pair of nodes in the network and uses this information for providing relevant routing information. The protocol is ideal for applications with predictable traffic patterns and so all necessary data are available for computation. However, in most sensor network applications, information about node location is difficult to obtain, let alone approximating the distance among nodes in the network. Furthermore, the problem of solving an optimal number of slots is computational-intensive and can only be run on a powerful node.

Multihop Infrastructure Network Architecture (MINA) [36] is another cross-layer design solution that integrates MAC and routing protocols. MINA proposed a multiple layers network architecture that group sensor nodes with the same hop-count to the base station to be in the same layer (cluster). MINA integrates a TDMA-driven protocol with CDMA or FDMA for node communication. Each frame is divided into a control section, a beacon section, and a data transmission section. The beacon and data section are time slotted. During beacon slots, sensor nodes (members) within a cluster send their transmission requests to the cluster head. The cluster head then advertises the transmission schedule (data slots) of members in its cluster. MINA uses a simple routing scheme where each node has a router at one nearer layer to the base station. MINA rotates the role of router based on the energy levels of nodes in a cluster.

FlexiTP shares many similarities with MINA. First, at the initial network setup, a path from each source node in the network to the base station is defined. In terms of network architecture, FlexiTP differs from MINA as it uses hierarchical tree instead of clustering. Second, both FlexiTP and MINA adopt a simple forwarding-to-parent scheme. Unlike MINA, in FlexiTP, each node remain connected to its parent unless the parent becomes unreachable due to battery depletion or external environmental factors.

The main drawback of MINA is that the frame length of nodes are fixed at the network startup and this makes the system intolerant to changes in network topology. In contrast, FlexiTP can dynamically modify the length of the network frame anytime.

Chapter 3

Research Methodology

What we observe is not nature itself, but nature exposed to our method of questioning.

Werner Heisenberg

In this chapter, we describe the network model that we assume for all of our work, describe the simulation framework for our protocol implementation, and define performance metrics for evaluating the performance of our protocol against existing sensor network protocols.

3.1 Wireless Sensor Network Model

Protocol designers should understand the parameters and objectives that are important to a sensor network application. In this thesis, we use the following concepts and assumptions:

- *Single-sink and multiple-sinks sensor network architecture.* FlexiTP is originally designed for data gathering applications where all sensor nodes forward their data to the base station. Our proposed peer-to-peer communication algorithm extends FlexiTP functionality to support multiple-sinks architecture where any node can send its data to any other node in the network.

- *Homogenous sensor network.* In this thesis, we focus on sensor network applications with nodes having the same hardware specifications and capabilities. For example, equal battery capacities are given to each of the sensor nodes in the network.

- *Sensor network without mobility.* FlexiTP is developed for static networks, hence, all nodes remain in the same position throughout their lifetime in the network.

- *Single radio sensor node.* FlexiTP is designed to save energy on a single radio architecture sensor node.

3.2 Simulation Framework

For analysis we have implemented FlexiTP in the network simulator (NS-2) version 2.30 [2]. NS-2 is a standard network simulator widely used by wireless sensor network researchers. The wireless sensor network components in the current NS-2 distribution code contain substantial contributions from the UC Berkeley Daedelus and CMU Monarch projects and Sun Microsystems.

Figure 11 illustrates the FlexiTP architecture of modules for sensor nodes in NS-2. It depicts the relationship between FlexiTP and NS-2 components, which will be described in the next sections.

3.2.1 WirelessPhy Module

Each sensor node has a WirelessPhy module to represent the properties of its radio and physical layer protocol. The WirelessPhy modules of all sensor nodes operating in the same channel are connected to a common Wireless Channel object. All WirelessPhy modules are also linked to a common Propagation Model object. A MAC802.11 module sits directly on the top of WirelessPhy. The *Send* (packet) function and *Recv* (packet)

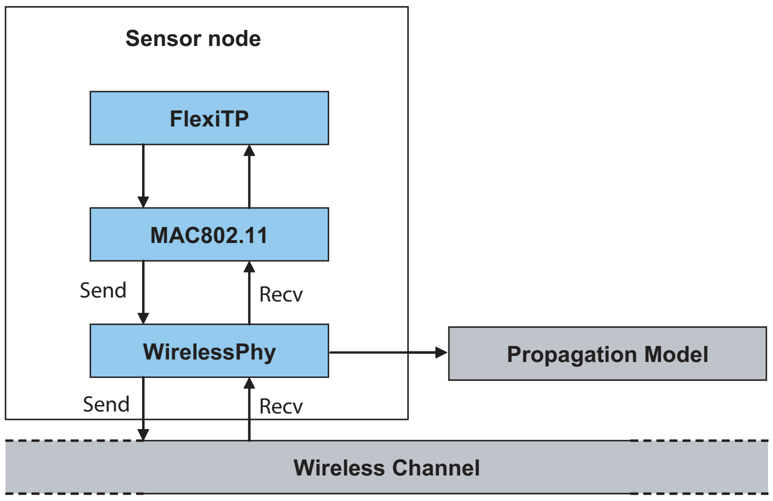

Figure 11: FlexiTP simulation framework in NS-2.

function provide the interface between MAC and WirelessPhy as well as between WirelessPhy and Wireless Channel.

The physical layer modeling is implemented as follows. To transmit a packet, the sender stores some context information, such as transmission power and its location, inside a common header section of a packet. WirelessPhy then triggers a transmission event, which is passed to all other WirelessPhy modules through Wireless Channel. Upon notified of a transmission event, WirelessPhy passes necessary information such as transmission power and sender and receiver positions, to Propagation Model. The propagation model calculates the receive power value of the packet and passes this value to WirelessPhy. WirelessPhy filters out an incoming frame if the received power is lower than a predefined carrier sense threshold. Otherwise, the packet is delivered to MAC802.11 for further processing.

3.2.2 MAC802.11 Module

This section briefly describes the IEEE 802.11 implementation in NS-2 and major modifications that we made to it for FlexiTP implementation. MAC802.11 is responsible for traditional physical layer functions such as frame body reception, collision detection, and channel state maintenance. MAC802.11 uses a channel access mechanism similar to

CSMA/CA (using RTS/CTS handshake) and use acknowledgments for reliability.

MAC802.11 performs virtual carrier sensing by tracking channel states. The receive state can be set to values of *receiving*, *collision* or *idle*. If the receive state is *idle* when a packet arrives from WirelessPhy, MAC802.11 goes into a reception mode and the value of the receive state is set to *receiving* for the duration of packet receiving. If there is any new packet arriving before the end of the reception mode, the receive state is set to *collision* until the end of this new packet's transmission. A node receives a packet successfully if all of the following conditions are met:

- the packet comes when the node's receive state is *idle*,
- the node's received power is above a predetermined receive threshold, and
- there is no collision through its reception.

FlexiTP utilises the collision-free nature of TDMA approach to eliminate the need of RTS/CTS handshake mechanism. In FlexiTP, sensor nodes broadcast their packets and so the RTS/CTS/DATA/ACK mechanism of MAC802.11 is disabled. Since at any particular time slot, only the sender and intended receivers are awake (in active mode), unicast and multicast communication patterns are supported implicitly.

3.2.3 Propagation Module

NS-2 provides implementations of three radio propagation models: two-ray ground, free space, and shadowing. Two-ray ground is the most commonly used radio propagation model for calculating received power of a packet. Rappaport [92] shows that the two-ray ground model gives more accurate prediction at a long distance than the free space model. The two-ray ground model considers both the direct path and a ground reflection path. This radio propagation model is deterministic: given a particular transmission power, there is only one possible receive power value at any particular distance. The received power at a distance is calculated based on the following equation:

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4} \tag{1}$$

where

$P_r(d)$ is the receive power given a transmitter-receiver distance,

P_t is the transmission power,

G_t is the antenna gain of the transmitter,

G_r is the antenna gain of the receiver,

h_t is the height of the transmitting antenna above ground, and

h_r is the height of the receiving antenna above ground,

d is the distance between the transmitter and the receiver.

In this thesis, all simulated sensor nodes have uniform transmission power and all simulations are based on the two-ray ground propagation model. In all simulations, we used an omnidirectional antenna with the following parameters: G_r = G_r = 1 and h_t = h_r = 1.5 m, 914MHz Lucent WaveLAN DSSS radio interface, and no system loss (L = 1).

3.2.4 Energy Module

Our simulations use the energy model module for wireless sensor networks implemented by SMAC developers for NS-2 [122]. Table 1 presents MICA2 platform transmission ranges and current drawn measured with a 3V power supply, which were obtained by Anastasi et al. in [8]. We used the Lagrange Interpolating Polynomial [118] based on the known four points to find current consumptions for other transmission ranges. Furthermore, our simulation parameters are based on Mica2 Mote hardware specifications used in [1], [8], [83], [87], and [120], as shown in Table 2.

Table 1: Mica2 Mote energy consumption models.

Tx Power (dBm)	Transmission Range (m)	Current Consumption (mA)
-20	5	8.6
-10	18	10.1
0	50	16.8
5	68	25.4

Table 2: Mica2 Mote property.

Mica-2 parameters	Description
CPU	ATmega128 8-bit, 8 MHz
Memory	4KB RAM 128KB Flash
Radio	CC1000
Channel bandwidth	19.2 Kbps (effective communication)
Transmission range	60 m
Transmit power for 60 m transmission range	63 mW
Receive power	30 mW
Idle power	30 mW
Sleep power	0.003 mW
Transition power	30 mW
Transition time	2.45 ms
Energy	54,000 J
Packet size	56 bytes (30 bytes payload and 20 bytes packet header)

3.2.5 Network Topology Model

We slightly modified the I-LENSE topology generator [76] to generate a sensor network topology with random node placement and every node within communication range of at least one other node in the network. The transmission range in all network topologies is 60 meters (m). Our simulation results are based on the mean value of 20 different network topologies involving up to 400 nodes located randomly in a network area of 300 x 300 m. The location of the base station was fixed at the top-center of the network map.

3.3 Performance Metrics

Two types of performance metrics that we consider important for evaluating the performance of our proposed protocols are energy efficiency metrics and end-to-end data delivery guarantee metrics. The performance metrics capture the notions of residual energy, energy conservation, quality-of-service assurances, and the connection between the application and the network.

The following are energy efficiency metrics:

- *Initial network setup cost per node* measures the total energy required for building a data gathering tree and TDMA schedule, averaged over every node.
- *Energy consumption per node* measures the total energy consumed for listening, transmitting, receiving, switching from sleep to idle mode and vice versa, averaged over every node.
- *Energy consumption per packet* measures the ratio of total energy consumed to the total number of packets received at the sink. This metric assesses the quality of active mode operations. A protocol with low energy per packet typically encounters few hidden terminals and packet retransmissions.
- *Network lifetime* measures the time until the first node in the network dies. The usefulness of the sensor network depends on the availability of sufficiently large

number of sensors to cover the area of interest, therefore, prolonging the lifetime of a sensor node is important.

- *Percentage of sleep time* measures the ratio of total sleeping time of nodes in the network, to total node active times for listening, receiving, and transmitting in a data gathering cycle, averaged over every node. A protocol with low idle listening period would provide high ratio of energy conservation.
- *Percentage of energy wastage* measures the ratio of total energy overhead due to collisions, overhearing, idle listening, and over-emitting, to total communication energy, averaged over every node. A protocol with low overhead, few collisions, and low idle period would result in small percentage of energy wastage.

The following metrics are used for evaluating end-to-end guarantees on data delivery:

- *Packet latency* measures the total time required for a packet to reach the sink node since it is sent by a source node.
- *Error rate* measures the number of transmission errors (collisions due to radio interference) over the total number of transmissions.
- *Network throughput* measures the total number of data packets received at the sink node to the total number of packets sent. A high network throughput indicates a small error rate for packet transmission and a low level of contention for medium access.

In this thesis, we included the 95% confidence interval for the simulation results into graphs, provided it shows significant variations within the results and it does not affect readability of the graphs.

Chapter 4

Flexible-Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks

If they lacked flexibility they wouldn't be able to adapt to different situations and come out winning. When you're flexible, you're willing to consider the best approach for each particular situation.

Wendy Hearn

In this chapter, we present FlexiTP, a novel flexible-schedule-based TDMA protocol that provides end-to-end guarantees on data delivery: predictable latency, predictable throughput, fair access, and robust self-healing, whilst also respecting the severe energy, memory, and bandwidth constraints of wireless sensor networks.

4.1 Introduction

FlexiTP is a novel TDMA-based protocol that offers a synchronised and loose slot structure. Sensor nodes in the network can build, modify, or extend their scheduled number of slots during execution, based on their local information. Nodes then wake up at their scheduled slots, otherwise switch into power-saving sleep mode. This flexible schedule allows FlexiTP to be strongly fault tolerant and highly energy efficient. FlexiTP is scalable for large number of nodes because its depth-first-search schedule minimises buffering, furthermore it allows communication slots to be reused by nodes outside each others' interference range. Hence, the overall scheme of FlexiTP provides end-to-end guarantees on data delivery: predictable throughput for gathered data, fair access to the network for all data gathering nodes, and robust self-healing of the network when nodes join or leave the network, and when communication conditions change; whilst also minimising energy and memory resources of wireless sensor networks.

FlexiTP utilises the cross layer interaction between the MAC and routing layers to provide efficient methods for allocating energy, bandwidth, and memory resources in wireless sensor networks. This goal is achieved by using knowledge about the sensor node schedule in the MAC protocol to enable the routing protocol to collect and disseminate data to and from nodes in wireless sensor networks. For data gathering tasks, FlexiTP uses forwarding-to-parent routing over a tree where nodes route packets to their parents until they lose connection with the parents. Boukerche et al. [15] shows that this simple routing scheme sustains high throughput and good energy efficiency. After a tree is constructed, every node in the network knows where to relay its packet since nodes remain static throughout their lifetime.

The rest of the chapter is organised as follows. Section 4.2 describes our FlexiTP approach. Section 4.3 presents simulations in NS-2. It shows that FlexiTP ensures energy efficiency and is robust to network failures under various network configurations. Furthermore, under high contention, FlexiTP outperforms Z-MAC in terms of the average packet delay, energy saving, and throughput. Section 4.4 concludes the chapter.

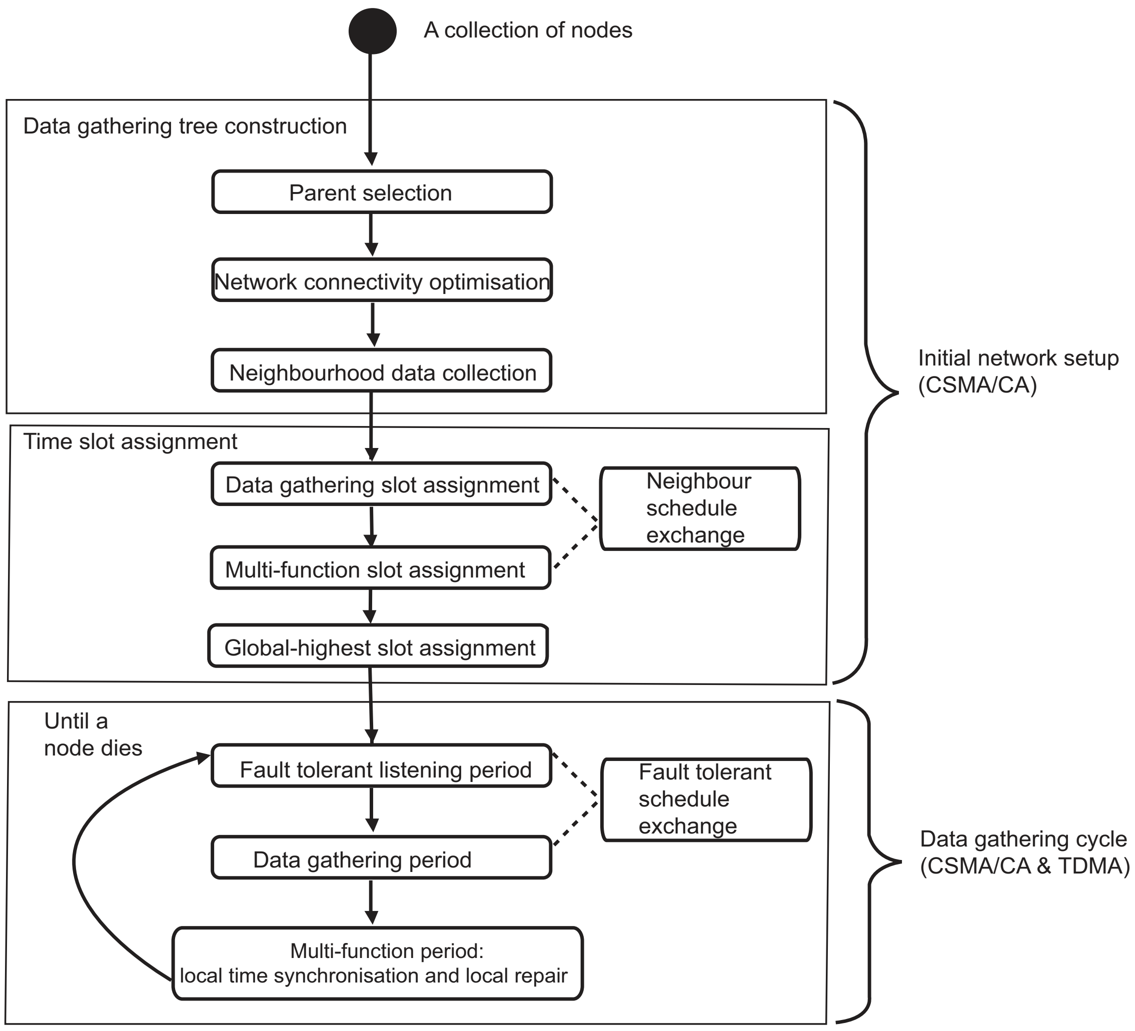
A collection of nodes
Data gathering tree construction
Parent selection
Network connectivity optimisation
Neighbourhood data collection
Initial network setup (CSMA/CA)
Time slot assignment
Data gathering slot assignment
Neighbour schedule exchange
Multi-function slot assignment
Global-highest slot assignment
Until a node dies
Fault tolerant listening period
Fault tolerant schedule exchange
Data gathering period
Data gathering cycle (CSMA/CA & TDMA)
Multi-function period: local time synchronisation and local repair

Figure 12: The execution of FlexiTP consists of one-off initial network setup phase and continuous data gathering phase.

4.2 FlexiTP Description

4.2.1 Protocol Overview

FlexiTP has two main phases: an initial network setup and data gathering cycles. Figure 12 shows the order of steps executed in a network. Users can implement any data gathering tree construction algorithm, the time slot assignment is the important feature of FlexiTP. Some steps within phases are carried out in parallel by the nodes, which are the neighbour schedule exchange and the fault tolerant schedule exchange (depicted by dashed lines).

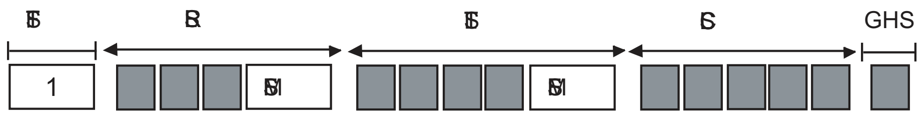

Figure 13: FlexiTP node schedule structure. The slots are not contiguous in time.

Initially, FlexiTP builds a data gathering tree and assigns nodes' schedules. Nodes then maintain their schedules throughout their lifetime in the network. Thus, this initial network setup is a one-off phase. During the initial network setup, FlexiTP uses CSMA/CA for packet transmission and so nodes' receivers are always on (i.e., all nodes in the network are in listen mode). FlexiTP uses a token passing scheme in which nodes perform procedures only if they hold a token, hence avoiding collisions. After the initial network setup finishes, nodes perform regular data gathering tasks using their TDMA schedules. They can also modify their schedules when the network topology changes. FlexiTP uses a forwarding-to-parent routing scheme [15] over a data gathering tree where nodes forward packets to their parents until these parents become unreachable due to energy depletion or external environmental factors.

Figure 13 shows a node's schedule structure. The schedule represents the list of slots when a node should be active. Slots need not be contiguous. In FlexiTP, slot number 1 is dedicated to *fault-tolerance slot (FTS)*, a short CSMA period where all nodes in the network are in listen mode. This feature allows nodes to adjust themselves according to their local neighbourhood states. Nodes switch to the receive mode for their scheduled *receive slot list (RSL)* slots, transmit mode for scheduled *transmit slot list (TSL)* slots, or else they switch to the sleep mode. During the data gathering cycle, a transmit slot of a node overlaps with the receive slot of the node's parent. This cross-layer interaction between MAC and routing protocol optimises the efficient usage of energy, memory, and bandwidth resources as only sensor nodes involved in a communication awake at a particular slot.

A *multi-function slot (MFS)* is used for local time synchronisation and is utilised for local repairs. A MFS slot in a node's TSL means that the node becomes the sender of a synchronisation packet in that slot, whilst the receivers of the packet record the MFS slot in their RSL. The *conflict slot list (CSL)* records slots that are used by a node's first-level and second-level neighbours. Nodes also maintain the *global-highest slot (GHS)* field that contains the network's highest slot number. A data gathering cycle is the range from the slot number one to GHS. Since nodes know the start and the end of a data gathering cycle, they can determine their positions in time.

In summary, main functions of FlexiTP are to perform:

- the establishment of routes (data gathering tree construction),
- the establishment of node schedule (time slot assignment),
- time synchronisation to minimise clock drifts, and
- local repair for a network in the event of faults (fault tolerance).

Throughout this thesis, we will use the following terminology (illustrated in Figure 14):

- *Ancestor* refers to an upstream node that is more than one-hop away from a source node, which the source node's packet is forwarded to and which is closer to the base station (in terms of hop-count). For example, BS is the ancestor of B and D.
- *Child* refers to an immediate node from which a node receives a packet and which is further from the base station (in terms of hop-count) compared to the node. For example, A, C, and E are children of BS.
- *Descendant* refers to a node that is more than one-hop away from a node and which is further from the base station (in terms of hop-count) compared to the node. For example, B and D are descendants of BS.
- *Parent* refers to an immediate node that forwards a source node's packet and that is closer to the base station (in terms of hop-count) compared to the node. For example, A is the parent of B and so B is the *child* of A.

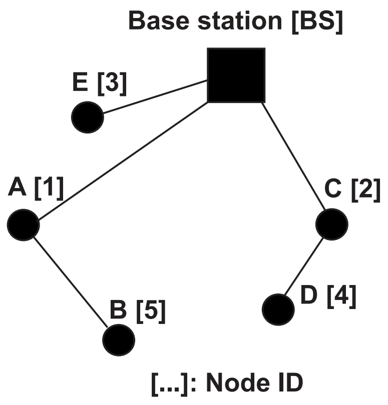

Figure 14: A tree built for gathering data in the network of five nodes.

- *First-level neighbour* refers to any node (not necessarily along the tree) within the communication range of a node. Figure 14 illustrates that B's first-level (direct) neighbours are A and D.
- Second-level neighbour refers to any node within the communication range of a node's first-level neighbour(s). Recall that B's first-level neighbours are A and D. Accordingly, B's second-level neighbours are BS, C, and E.
- *Sub-tree* refers to a branch of a tree that consists of a parent and one or many children. For example, BS, A, C, E make a sub-tree; C and D also make a sub-tree.
- *Network density* refers to the total number of nodes in a network.

4.2.2 Data Gathering Tree Construction

FlexiTP's data gathering tree construction phase is responsible for establishing connectivity among nodes and creating a tree-path for routing data from source nodes in a network to the base station. After a data gathering tree is constructed, each node knows its parent, children, descendants, and first-level neighbours. A data gathering tree is constructed in three phases: parent selection, network connectivity optimisation, and neighbourhood data collection. The nodes use CSMA/CA during these phases.

Parent Selection

The base station acts as the root of a data gathering tree. The base station generates a connectivity token and initiates the data gathering tree construction using a simple flooding scheme. The connectivity token is distributed using a top-down sub-tree by sub-tree approach. A sub-tree consists of a parent and its children. The rule is that the connectivity token is passed from a node to the child with the smallest node ID in the current sub-tree.

When a node obtains the connectivity token, it broadcasts a tree construction signal in order to find its prospective children. Other nodes in the network that are able to receive the signal become the children of the broadcaster node. If the broadcaster node receives no reply from any node within a period of time, it will retry to broadcast the tree construction signal for a certain number of times until it receives a reply. Thus, a node can release the connectivity token if either of the following conditions is met:

- the node has found its children, or
- the node is unable to find its children after executing the maximum number of tree construction broadcast attempts.

This parent selection phase ends when the token comes back to the base station finally, i.e., after all the children of the base station and their descendants have selected their parents.

FlexiTP allows the human manager to have control over the breadth of a data gathering tree by specifying 1) the maximum number of children a node can have, and 2) the maximum number of broadcast retries for tree construction.

Network Connectivity Optimisation

In the network connectivity optimisation phase, a node that is unable to get a tree construction signal after a certain period of time (user-defined parameter) will send a distress signal to find a parent. Recall that nodes use CSMA/CA during this phase. A node backs off for a random period of time if it overhears another distress signal. If not, it broadcasts a distress signal. Neighbouring nodes that receive this distress signal send a reply and confirm their availability to accept the distressed node as their child. The distressed node then selects the neighbour node with the lowest hop-count to the base station to become its parent.

Neighbourhood Data Collection

This phase collects information about a node's direct (first-level) neighbours. During the parent selection and network connectivity optimisation phase, a node may send signalling packets and receive multiple replies from other nodes within its communication range. The node then records these nodes as its direct neighbours in its neighbourhood table. This neighbourhood data is propagated to the base station and hence the base station builds a network connectivity map.

4.2.3 Time Slot Assignment

FlexiTP uses a depth-first-search schedule to reduce data buffering. This scheduling scheme allows data sent by a source node to be forwarded by routers to the base station in an interleaving manner. This scheme enhances reliability by preventing the loss of packets due to buffer overflow. After a data gathering tree is constructed, the local topology is known to nodes in the network in that each node knows its parent and children (if any). The base station then generates a time slot assignment token and initiates the time slot assignment phase. The token passing is done using the depth-first-search technique. The token is first passed to the child that has the lowest node ID.

FlexiTP time slot assignment function consists of four main phases: data gathering slot assignment, multi-function slot assignment, neighbour schedule exchange, and global-highest slot assignment. In the *data gathering slot assignment*, each node in the network builds its data gathering schedule by selecting a slot to transmit its own data and then routers on the path to the base station assign *receive* and *forward* slots for that node's data. Once a node claims a slot, it executes the *neighbour schedule exchange* to propagate the claimed slot to its first-level and second-level neighbours. After all nodes in the network build their data gathering schedules, they execute the *multi-function slot assignment*. Finally, the base station informs the network highest slot to all nodes in the network through the *global-highest slot assignment*.

In FlexiTP, the slot selection always starts from slot number 2 because slot number 1 is dedicated to fault-tolerance slot. A node can claim a slot if the slot is not listed in its receive slot list, transmit slot list, and conflict slot list. Therefore, in selecting a slot, there is no need to know the true value of global-highest slot. This slot selection scheme allows a slot to be reused by many nodes as long as nodes' transmissions are not within interference range of each other. This scheme maximises slot reuse.

Data Gathering Slot Assignment

When a node obtains the token, it allocates a transmit slot to send its own data by choosing the lowest slot number that is not listed in its receive slot list, transmit slot list, and conflict slot list fields. The node (current requestor) informs its claimed slot to its parent (one of its first-level neighbours) by unicasting a *forward_slot_request* as well as piggybacking an *inform_slot_request*. In this way, slots to forward the node's data to the base station are allocated. Furthermore, the *inform_slot_request* on the broadcast packet triggers the neighbour schedule exchange phase that will be described later in Section Neighbour Schedule Exchange.

The *forward_slot_request* contains the claimed slot information: the slot number, the receiver's ID (parent ID), and the neighbour level counter. Upon the reception of a *forward_slot_request* packet, first-level neighbours of the requestor check if they are intended receivers of that packet. If a neighbour is the intended receiver of the packet, it knows that it is the parent of the current requestor and so lists the received slot number in its receive slot list, else it will execute the neighbour schedule exchange. The parent then assigns the corresponding transmit slot, i.e. a slot to forward the requestor's data. The corresponding transmit slot must be bigger than the received slot number and is not listed in the parent's receive slot list, transmit slot list, and conflict slot list. Afterwards, the parent executes the neighbour schedule exchange and sends the *forward_slot_request* to the requestor's ancestor (i.e., routers). The data gathering slot assignment as explained before is repeated until the base station receives the *forward_slot_request*. The data gathering slot assignment ends when the token is returned to the base station.

Multi-Function and Global-Highest Slot Assignment

FlexiTP adopts a top-down multi-function slot structure in which the base station initiates the multi-function packet propagation in a data gathering cycle. In multi-function period, each parent node in the network synchronises its children. FlexiTP performs the multi-function slot assignment right after the data gathering slot assignment ends.

Initially, the base station generates an assignment token. The base station becomes the current parent to claim a multi-function slot. The multi-function slot selection scheme is based on the following criteria:

1. a node claims a multi-function slot that is bigger than any slots in its receive slot list and transmit slot list because the multi-function slot is claimed after all nodes build their data gathering schedule, and

2. a node claims a multi-function slot that is not listed in its conflict slot list.

The base station adds the claimed multi-function slot in its transmit slot list. The base station then informs the multi-function slot to all of its children by multicasting an *inform_slot_request* packet consisting of the slot number, the receivers' node ID (children node IDs), and the neighbour level counter. The children then assign the multi-function slot as the reserved receive slot in their receive slot list. The neighbour schedule exchange is then also performed. Afterwards, the base station performs the depth-first-search token passing. When a node in the network receives the token but has no child, the node skips claiming a multi-function slot and continues passing the token accordingly. When the token is returned to the base station, the multi-function slot assignment ends.

After the data gathering and multi-function slot assignment phase, the base station knows the highest slot claimed by any node in the network. The base station then again initiates depth-first-search token-passing mechanism to inform all nodes in the network of the global-highest slot number allocated and thus the current size of the frame (data gathering cycle).

Neighbour Schedule Exchange

The neighbour schedule exchange phase is performed through CSMA/CA communication in which a node broadcasts its claimed slot to its direct neighbours (first-level neighbours) and then each of these direct neighbours propagate the slot information again to their direct neighbours (second-level neighbours). In this way, a slot claimed by a node is informed to all neighbours within twice the node's communication range. FlexiTP can guarantee collision-free traffic because it prevents nodes that are potentially within the interference range of each other from claiming the same slot.

If during the data gathering slot assignment or the multi-function slot assignment phase a node receives an *inform_slot_request* in a packet, it means that the neighbour schedule exchange phase needs to be executed. The neighbour slot exchange is controlled by the neighbour level counter information in the packet, which indicates the status of a neighbour, whether it is a first-level neighbour (the neighbour level counter is equal to one) or a second-level neighbour (the neighbour level counter is equal to two). If a node receives a packet with the level counter less than or equal to two, it will include the slot in its conflict slot list. Furthermore, if the current level counter is less than two, the node increments the level counter value and broadcasts the packet to its direct neighbours; if not it stops the neighbour schedule exchange.

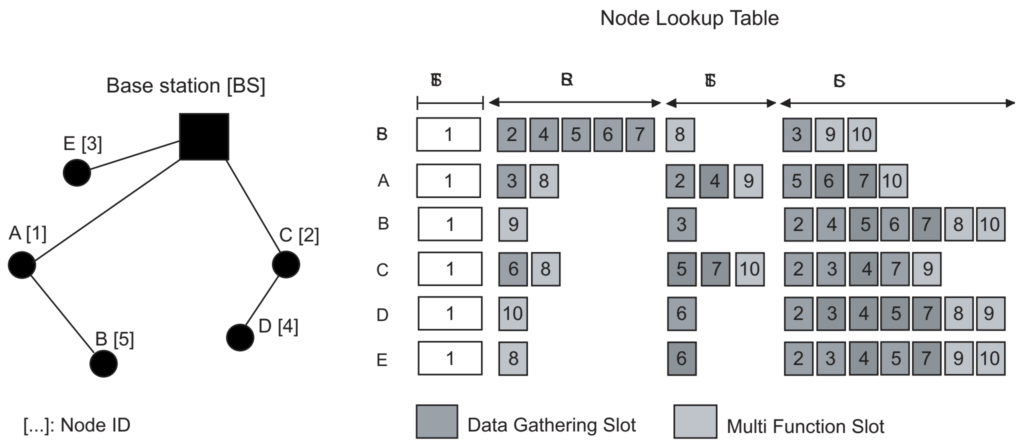

Figure 15: The schedule lookup table of FlexiTP nodes.

Example

We now illustrate the time slot assignment algorithms described previously by using the data gathering tree shown in Figure 15. The base station generates a token and initiates the time slot assignment function. The token passing is done using the depth-first-search technique. The token is passed to the child that has the lowest node ID. Thus, the base station will pass the token to node *A* first and *A* becomes the token holder. At this stage, no node has claimed slot number 2, so *A* claims slot number 2 and records this slot in its transmit slot list (TSL). Node *A* then propagates the slot information to its parent (*BS*), and its first and second-level neighbours. Node *A*'s first-level neighbours are nodes *BS*, *B*, and *E*. Since *BS* is the intended receiver of *A*'s transmission, BS lists the slot in its receive slot list (RSL), while *B* and *E* list the slot in their conflict slot list (CSL). Again, *BS*, *B*, and *E* broadcast the slot information to their direct neighbours and hence *C* and *D* also list slot number 2 in their CSL. Afterwards, *A* passes the token to its child, node *B*.

When node *B* becomes the token holder, it claims the lowest available slot that is not listed in its RSL, TSL, and CSL, which is slot number 3. Again, *B* propagates this slot information to its parent and neighbours. Node *B*'s parent, *A*, then selects the next available slot to forward *B*'s data to the base station, which is slot number 4. Node *A* propagates the claimed forward slot to its parent and neighbours. This token passing mechanism is replicated until all nodes in the network receive the token. Note that node

E can reuse slot number 6 because this slot does not appear in *E*'s conflict slot list. Thus, in slot number 6, both node *D* and *E* send their packets to their parents, while *C* and *BS* expect to receive a packet from their children in that slot. When the token is returned to the base station permanently, the data gathering slot assignment phase finishes. Afterwards, the base station initiates the multi-function slot assignment.

The base station generates a token and claims the multi-function slot number 8. The base station propagates this slot to its children, and its first and second-level neighbours. The base station adds the multi-function slot number 8 in its TSL, while its children (node *C* and *E*) add the multi-function slot in their RSL, and its neighbours (node *B* and *D*) add the multi-function slot in their CSL. The rest of this phase is performed similarly to the depth-first-search token passing used in the data gathering slot assignment.

Since the base station knows the network topology and nodes' schedules, it can use the token-passing mechanism again to inform nodes the global-highest slot in the network and the start of the data gathering cycle. After FlexiTP's initial network setup finishes, nodes perform regular data gathering tasks using their TDMA schedules.

4.2.4 Time Synchronisation Scheme

Time synchronisation is critical in TDMA-based MAC protocols because nodes that are involved in a scheduled communication must wake up at the same time to exchange information. In FlexiTP, a hierarchical structure is constructed in the data gathering tree construction phase. Each node in the tree knows its parent and its children. FlexiTP performs time synchronisation locally, with each parent synchronising its children. This multi-hop parent-children broadcast synchronisation approach is similar to root-neighbours synchronisation approach in FTSP [81].

In the multi-function slot of a node, the node broadcasts its clock and the current global-highest slot number known to that node to the children. The purposes of propagating the global-highest slot are to allow nodes in the network to keep the period of data gathering cycle up-to-date and to allow nodes to perform a local repair. FlexiTP time synchronisation scheme is desirable because children only need to have the same clocks as their parent to ensure that a parent is in the receive mode when a child sends data to it and vice versa. This local synchronisation scheme is simple but effective because clock drift is minimised by synchronising nodes during each data gathering cycle. Furthermore, it incurs low overheads since the synchronisation message is piggybacked to the multi-function packet.

In addition to the local time synchronisation scheme, FlexiTP adjusts its slot length to include radio switching times, which are the times required for a sensor node radio to transit from the sleep state to idle and from the idle state to sleep. Thus, when the sender wants to transmit it is guaranteed that the intended receiver(s) are awake and listening. In the worst case, a node falls out of synchronisation before its multi-function slot cycle. The node then tries to re-synchronise itself to the network by listening to its neighbourhood activities. If the node overhears any of its neighbour's data transmission, it retrieves the current slot number of the transmission and uses the slot number to adjust its virtual clock. It then uses this virtual clock as the reference for reading its look up table (i.e., schedule).

4.2.5 Fault Tolerance

Wireless sensor networks are prone to network failures such as dropped packets, nodes dying, being disconnected, powering on or off, and new nodes joining the network. In such environments, the networks need to be able to self-configure without knowing anything of the network topology in advance [9, 15, 44, 45, 61].

In FlexiTP, nodes listen to environment activities during the fault-tolerance slot to allow these nodes to self-configure in the event of failures without prior knowledge of the network topology. If any node sends a distress signal during the fault-tolerance slot, nodes in the network that receive this signal will reply to it and perform a local self repair.

In the event that a new node is added to the network or when an existing node in the network wants to find a new parent, FlexiTP allows nodes in the network to adopt these faulty nodes as their children, assign schedules for these faulty nodes, and rebuild their own schedules. FlexiTP utilises the existing TDMA schedule to perform a local repair and so a collision-free and cost-free schedule rebuilding and schedule exchange is guaranteed.

Packet Loss

FlexiTP handles packet loss during the initial network setup as well as data gathering. In the initial network setup, FlexiTP utilises RTS/CTS/DATA/ACK scheme of IEEE 802.11 to handle packet loss during the depth-first-search token passing. For example, when a parent does not receive a token ACK from its child (receiver), the parent keeps retransmitting the token until it receives an ACK, for a specified period of time (user-parameter). If after the time expires, the parent still receives no ACK, it proceeds to pass the token to the next child. On the other hand, if a child does not receive a token ACK from its parent after a period of time, it will find a new parent. Since in the initial network setup phase, nodes work in CSMA/CA and they are all in listening mode, the child can broadcast a distress signal to find a new parent. The child's neighbours that receive the signal send a reply. The child then selects the best parent based on the lowest hop-count to the base station. The selected parent updates its list of children and propagates this topology change to the base station and so the token-passing flow in the network reflects the current network connectivity.

In real world implementations, there would be environmental factors such as obstacles and humidity that could affect node communications and transmissions. Moreover, field trials of sensor network for environmental monitoring of soil moisture in [19] showed that nodes may lose connectivity for extended periods. In FlexiTP, during data gathering cycles, nodes detect packet loss based on their scheduled receive slots. For example, when a node does not receive a packet when it expects data from its child, it will request a data retransmission from that child during the multi-function slot. The child appends the requested data onto the data gathering packet that will be sent in the next data gathering cycle. Furthermore, if a node receives the data retransmission request repeatedly, it decides to find a new parent because the current link connectivity is too noisy.

New Nodes

In the fault-tolerance slot, a new node sends a distress signal. Nodes that are within the communication range of the new node become neighbours and prospective parents of the new node. The new node then selects one of them as its parent, based on the lowest hop-count to the base station. The selected parent allocates two slots to the new node: one data gathering slot (that is the parent's global-highest slot incremented by one) and one multi-function slot (the parent's multi-function slot). If the parent does not have a multi-function slot because it did not have a child before, it will assign a multi-function slot to itself first. The multi-function slot is equal to the current global-highest slot incremented by two. The parent allocates this multi-function slot to the new node. Afterwards, the new node's routers build the new node's data gathering schedule by utilising existing data gathering slots and multi-function slots. The selected parent (let it be a requestor), first, proposes a forward slot (i.e., forward slot request) for the new node to its parent. The requestor piggybacks the proposed slot on the data packet sent in the immediate data gathering slot. The requestor's parent then checks the proposed slot against its lookup table. If the proposed slot exists in its lookup table, a next available slot will be chosen to replace it. The approved slot will be listed in the requestor parent's receive slot list and also will be sent to the requestor in the multi-function slot. The requestor then claims the approved slot and lists the slot in its transmit slot list. Afterwards, the requestor's parent becomes the current requestor and proposes a forward slot to its parent. This scheme is executed by traversing the path to the base station. Note that every time a node claims a new slot, it triggers the fault tolerance schedule exchange (executed within a data gathering cycle).

Dead Nodes

The term 'dead node' refers to one of the following conditions: a node whose battery is weak or dead, or a node that is unreachable or unable to communicate due to environmental factors such as fog or obstacles [14]. If a node does not receive the expected data during all scheduled receive slots of a child or descendant after two data gathering cycles, it assumes that the child or descendant is dead. It removes the child or descendant from its children list or descendant list respectively. The node performs a slot garbage collection in which all receive, transmit, and multi-function slots that are associated with that child or descendant are deleted from its lookup table so that these slots are not wasted on idle listening. For example, when node *D* in Figure 14 fails to send data to its parent, node *B*, *B* does not forward *D*'s data to *BS*. If *BS* and *B* do not receive *D*'s data after two data gathering cycles, they can assume that *D* is dead.

A node becomes an orphan if it does not receive a synchronisation message in its multi-function slot from its parent after two data gathering cycles. The orphan node then broadcasts a distress signal in the fault-tolerance slot to find a new parent. The orphan node first tries to find prospective parents that are neither its children nor descendants. The orphan node selects a prospective parent that has the lowest hop-count to the base station as its new parent. If other nodes in the data gathering tree are outside the orphan's communication range, a child or descendant of the orphan becomes eligible to be a new parent if the child or descendant can reach a node that has a path to the base station. The data gathering schedule assignment for the orphan node is the same as the new node's assignment, except that there may be many proposed slots since the orphan node may have children and descendants. The orphan node needs to propose its data gathering slots (including the forward slots for its children and descendants) in its transmit slot list to the new parent. This method allows the orphan node to transfer its transmit slot list to the new parent while maintaining its receive slot list. That is, children and descendants of the orphan node do not need to rebuild their schedules.

Fault Tolerant Schedule Exchange

The purpose of the fault tolerant schedule exchange is to inform all nodes within interference range of a node's claimed slot. The fault tolerant schedule exchange phase uses the fault-tolerance schedule and existing TDMA schedule: data gathering slots and multi-function slots.

Every time a node claims one or more transmit slots, the fault tolerant neighbour schedule exchange phase is triggered. The list of claimed slots is called *inform_slots* and the slot information (slot IDs and the neighbour level counter) is piggybacked on the node's data packet sent in the immediate data gathering slot and multi-function slot. A receiver of the packet retrieves the neighbour level counter. If the neighbour level counter is less or equal to three, the receiver lists slots in its conflict slot list, increments the neighbour level counter, and piggybacks the updated *inform_slots* on data packet sent in the immediate data gathering slot and multi-function slot; else the receiver drops the *inform_slots*. In the fault tolerance slot, the node also broadcasts its claimed slots to its first-level and second-level neighbours. This scheme ensures that first-level, second-level, and some of third-level neighbours of a node are informed of a new claimed transmit slot.

4.3 Performance Evaluation

We implemented FlexiTP in NS-2 [2] and the code is downloadable at [67]. Table 2 in Chapter 3 presents our simulation parameters. We set the length of FlexiTP data-gathering slot and multi-function slot to be 27 ms in order to be long enough to cover 23.3 ms for a packet transfer [8], 2.45 ms [83] for a node radio to change from the sleep state to idle, and 0.25 ms [83] for node radio to change from the idle state to sleep. Although the switching energy cost is non-negligible [120], using the FlexiTP TDMA approach is still better than not putting nodes into the sleep state if they are inactive. For example, suppose a node is inactive for three slots. By putting the node into the sleep state during these three slots, the amount of energy spent is:

$$(T_{\text{tran\_on}} * P_{\text{tran\_on}}) + (T_{\text{tran\_off}} * P_{\text{tran\_off}}) + (3 * T_{\text{slot}} * P_{\text{sleep}}) = 0.08\ mJ \qquad (2)$$

This value is much smaller than keeping the node idle that consumes:

$$(3 * T_{\text{slot}} * P_{\text{idle}}) = 2.34\ mJ. \qquad (3)$$

We set the length of FlexiTP fault-tolerance period to be 100 ms ($\approx$ four slots). Note that the length of fault-tolerance slot can be adjusted according to the application requirements. For example, users can either set a very short fault-tolerance period for a sensor network application that is deployed in a stable environment or an application that can tolerate a slow network stabilisation. A long fault-tolerance period is used for a sensor network application that is prone to network errors.

We included the upper and lower bounds of a 95% confidence interval for the simulation results into graphs, provided it shows significant variation within the results and it does not affect readability of the graphs.

4.3.1 Energy Efficiency

Initial Network Setup

The initial network setup cost is the energy cost for constructing a data gathering tree, collecting neighbourhood information, and building nodes' schedules. Recall that this setup cost is a one-off cost as nodes maintain their schedules throughout their lifetime in the network. Figure 16 shows the average node energy overhead in the initial network setup and how much of it spent on the time slot assignment phase. The initial network setup cost per node is less than 0.6% of the node's total initial energy of 54000 J, across various network scenarios. Thus, our simulations based on Mica2 Mote show that FlexiTP is scalable for up to 400 nodes since it maintains low initial network setup cost. Figure 17 shows the average time required for all nodes in a network to perform the initial network setup and are in the time slot assignment phase. As the network density

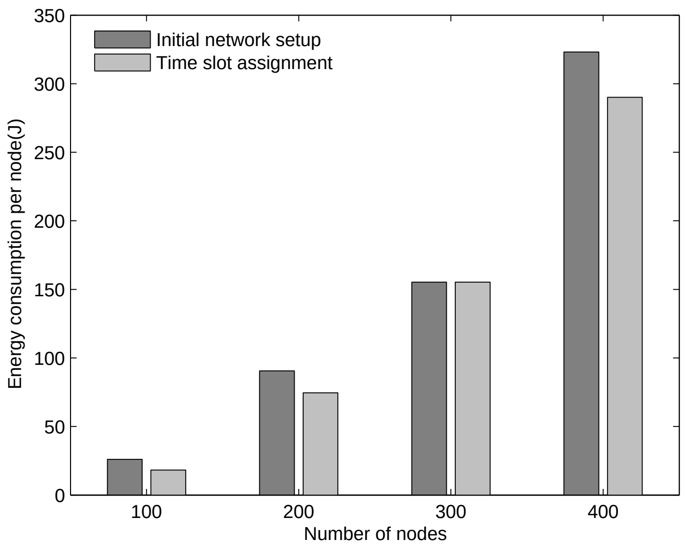

Figure 16: FlexiTP initial network setup cost per node versus network density. The initial network setup measures the energy cost for tree construction, neighbourhood data collection, and time slot assignment.

increases, the initial network setup cost increases because of the increase in idle, receive, and transmit activities in configuring the network.

Data Gathering Cycle

We measured node energy consumption per cycle by calculating the average energy consumed by a node for listening, switching, transmitting, receiving, and sleeping during a data gathering cycle. When the slot reuse feature of FlexiTP is switched on, the amount of energy spent on radio switching is reduced, which results in energy savings, as confirmed by our simulation results in Figure 18.

We also calculated energy consumption per packet, that is, the average energy consumed in transmitting and receiving a packet from a source node to the base station. For low network density, the network topology is sparse and there are more nodes with short links, leading to more routers. Two routers, for example, consume twice the receive energy cost of one router only. This can increase the average energy cost per packet, as confirmed by our simulation results in Figure 19.

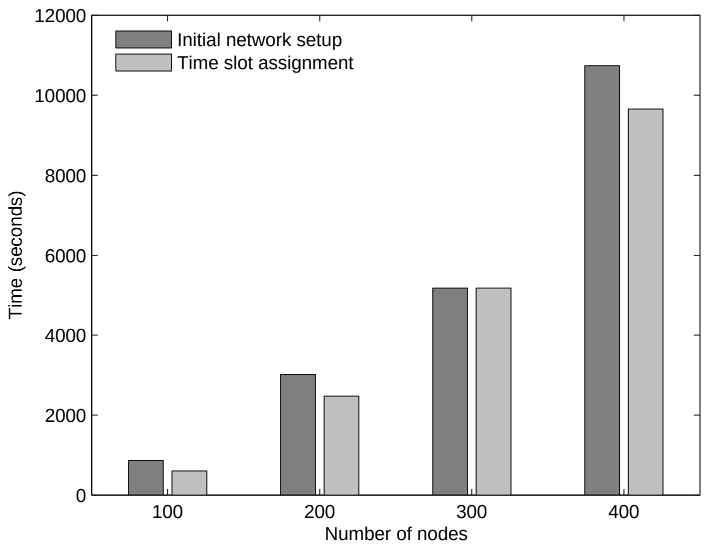

Figure 17: FlexiTP initial network setup duration versus network density.

An important goal in developing algorithms for wireless sensor networks is to minimise the energy consumption of sensor nodes' operations in order to increase the lifetime of the network. In FlexiTP, nodes switch to the low power sleep mode if they are not scheduled to receive or transmit data and therefore conserve energy and increase the node lifetime. Figure 20 shows the average network lifetime before the first node in the network dies.

In FlexiTP, nodes switch to low power sleep mode if they are not scheduled to receive or transmit data. Thus, nodes sleep most of the time in each data gathering cycle. Consequently, FlexiTP provides a high percentage of sleep time in each data gathering cycle over various network densities, as presented in Figure 21. As the network density increases, the average percentage of sleep time increases.

Lastly, since only nodes that are involved in a communication awake at every slot in a data gathering cycle and each source node's packet is delivered to the base station in an interleaving manner, FlexiTP guarantees reliable delivery of packets and *eliminates energy wastage* due to collisions, overhearing, idle listening, and over-emitting. Hence, FlexiTP provides an efficient data gathering scheme for wireless sensor networks.

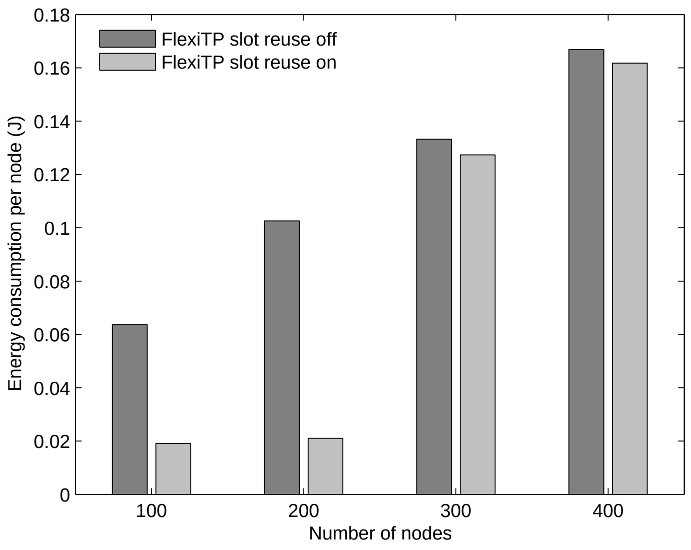

Figure 18: FlexiTP energy consumption per node versus network density.

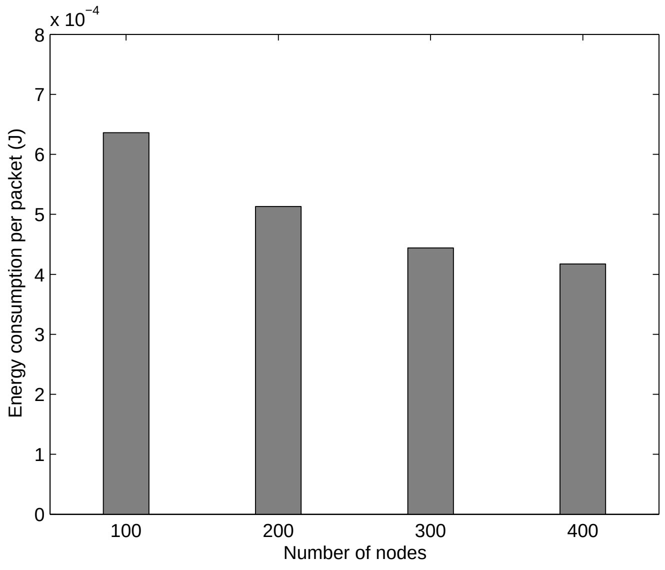

Figure 19: FlexiTP energy consumption per packet versus network density.

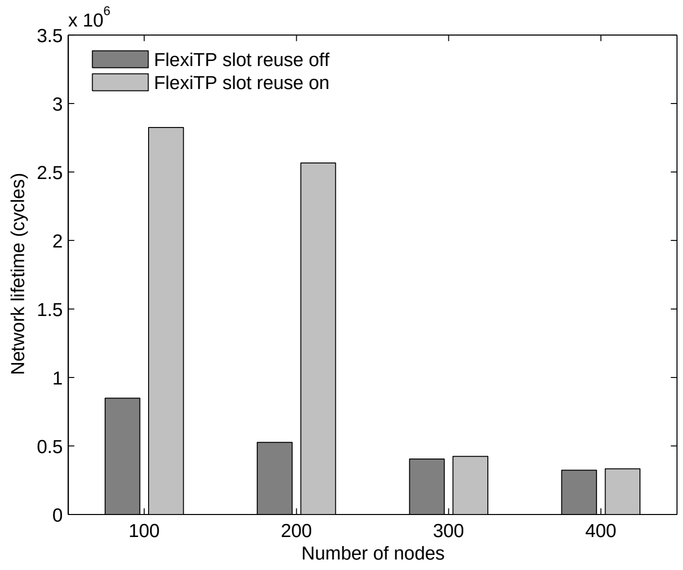

Figure 20: FlexiTP network lifetime versus network density.

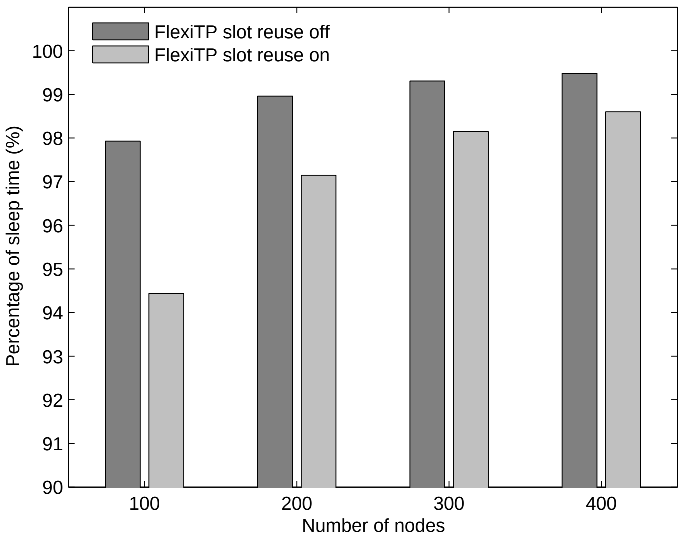

Figure 21: FlexiTP percentage of sleep time per data gathering cycle versus network density.

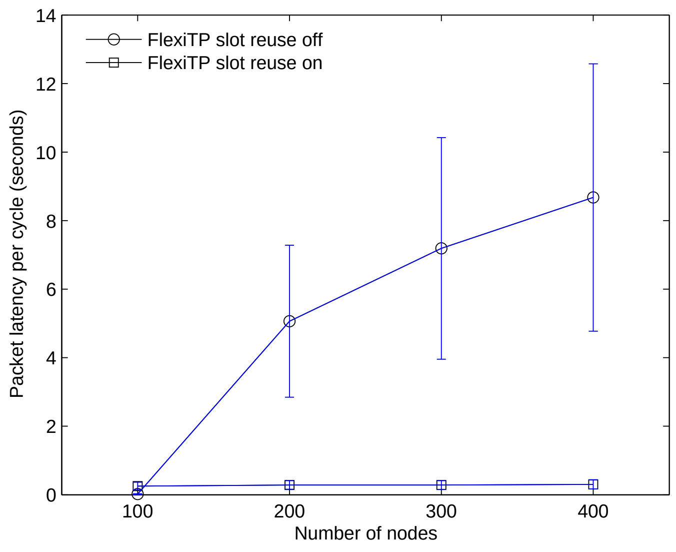

Figure 22: FlexiTP packet latency per gathering cycle versus network density.

4.3.2 End-to-End Guarantees on Data Delivery

When FlexiTP's slot reuse scheme feature switched on, all nodes always claim the lowest available slot and so slot reuse can be maximised. This method will also minimise the number of slots required in a network. We measured the percentage of slot reuse, that is, the rate of total number of slots that are used at least twice, to the total number of transmission slots in a network, averaged over all nodes in the network. Simulation results shows that FlexiTP provides slot reuse of at least 62% of the total transmission slots used in the network across different network scenarios and network densities.

We first evaluated packet latency, that is, the time required for a packet to reach the base station since it is sent by a source node. Figure 22 depicts the average latency per packet in a data gathering cycle as well as the upper and lower bounds of a 95% confidence interval for the average latency per packet. As expected, simulation results show that FlexiTP with slot reuse results in shorter latency than FlexiTP without the slot reuse scheme.

We now presented a simple analytical model to study the average latency per packet per data gathering cycle in FlexiTP without the slot reuse scheme. Let L represents the packet latency, `L(n)` represents the average latency per packet in a network of `n` nodes, `h` be the total number of hops required by a node to reach the base station, and T_slot be the duration of a time slot. In FlexiTP, data sent by a source node is forwarded by the node's routers to the base station in an interleaving manner. The packet latency is equal to the total time required by a source node and its routers to relay its data to the base station, which can be formulated as follows:

$$L = h * T\_\text{slot} \tag{4}$$

We measured the average latency per packet by calculating the average time it takes for all nodes in the network to send their packets to the base station in each data gathering cycle. We studied two most extreme network topologies: single hop network and line topology network. In a single hop network, each node in the network is one hop away from the base station. Therefore, nodes can send their packets to the base station directly. Based on Equation 4, the time it takes for each node in the network to send its packet to the base station is equal to $(1 * \texttt{T\_slot})$. Thus, the value of average latency per packet in the network of `n` nodes can be formulated as follows:

$$L(n) = \frac{n * T\_\text{slot}}{n} = T\_\text{slot} \tag{5}$$

In a line topology network, each node in the network has a single child. In this case, a source node that is `h` hops away from the base station, requires `h` routers to relay its data to the base station. For example, the average latency per packet in a network of three nodes is equal to $(1 * \texttt{T\_slot}) + (2 * \texttt{T\_slot}) + (3 * \texttt{T\_slot})$. Thus, the value of average latency per packet in the network of `n` nodes can be formulated as follows:

$$L(n) = \frac{\sum_{i=1}^{n} i * T\_\text{slot}}{n} \tag{6}$$

Thus, the lower and upper bound of FlexiTP's average latency per packet is as follows:

$$T_\text{slot} \leq L(n) \leq \frac{\sum_{i=1}^{n} i * T_\text{slot}}{n} \tag{7}$$

Sensor nodes have highly constrained computing power and memory. Multi-hop routing requires routers to buffer multiple packets prior to forwarding them to the next hop.

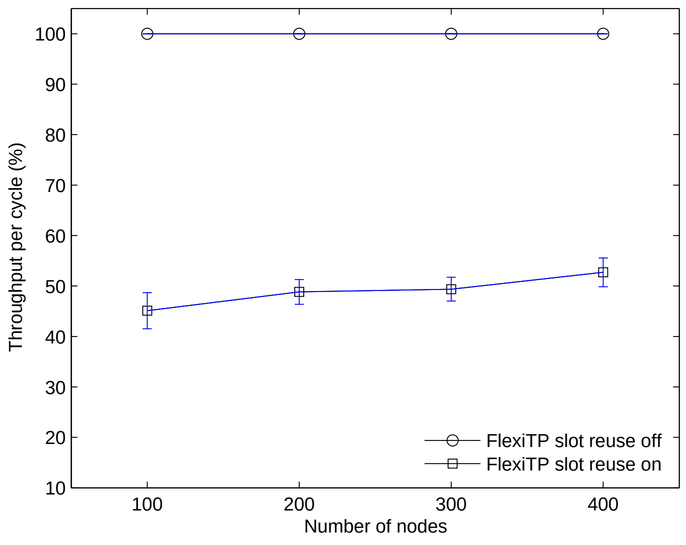

Figure 23: FlexiTP network throughput per data gathering cycle versus network density.

Packets may be dropped because of nodes' memory buffer overflow. This results in a low throughput. FlexiTP addresses this issue so that nodes only buffer one packet at a time to be sent to the next hop. This ensures regular data delivery and so provides predictable throughput.

Figure 23 shows FlexiTP throughput with and without the slot reuse scheme as well as the upper and lower bounds of a 95% confidence interval for the throughput. Theoretically, in multi-hop wireless sensors, a node's slot is safe to be re-used by other nodes that are two or more hops away [32, 95]. The FlexiTP time slot assignment scheme ensures that nodes and their first-level and second-level neighbours cannot use the same transmit slot to guarantee collision-free traffic. However, our simulation results show that this assumption does not guarantee a collision free slot reuse as FlexiTP with slot reuse scheme on produced a lower throughput than when the slot reuse scheme is off. We think this phenomenon is caused by interference and irregularity in radio channels. In order to validate this argument, we observed the collision at both senders and receivers in each slot in the network.

We measured the packet error rate, that is, the ratio of total packet collisions to the total packet transmissions. Figure 24 shows the packet error rate as well as the upper and lower

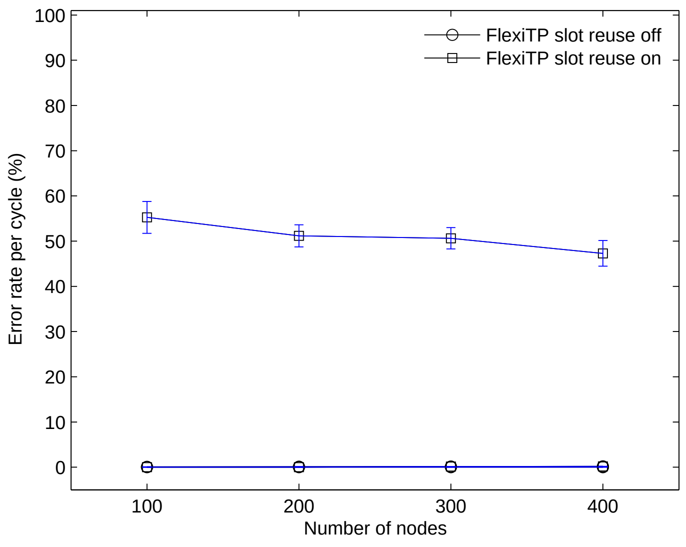

Figure 24: FlexiTP error rate per data gathering cycle versus network density.

bounds of a 95% confidence interval for the packet error rate. Simulation results show that FlexiTP without slot reuse allows all packets sent by source nodes in the network reached the base station because there were no collisions, while FlexiTP with slot reuse scheme introduces high error rates and hence provides less reliable data delivery.

4.3.3 Protocol Comparison

We compared the performance of FlexiTP with Z-MAC in terms of energy efficiency: energy consumption per node, energy consumption per packet, and network lifetime; and network performance: packet delay, throughput, fairness, and network utilisation. We followed the Z-MAC NS-2 installation manual detailed in [93] and configured Z-MAC according to default settings shown in Table 1 in [94], which are partly reproduced in Table 3 of this thesis. However, simulation of Z-MAC for networks beyond 100 nodes did not work. Thus, we can only showed the comparison of FlexiTP versus Z-MAC for network topologies of 100 nodes.

We simulated a periodic data gathering network for 300 seconds with inter-packet arrival of five seconds. In all simulations, each sensor node in the network, except the base

Table 3: Z-MAC simulation parameters in NS-2.

Simulation parameters	Description
Owner contention window size (To)	8
Non-owner contention window size (Tno)	32
Z-MAC TDMA slot size	50 ms
Communication range (NS-2)	200 ft
Interference range (NS-2)	300 ft
Communication bandwidth (NS-2)	19.2 Kbps
Routing protocol	DSDV

station, is a source node that always has data to send every five seconds (i.e., fixed data rate) for 300 seconds. In this high contention network case, FlexiTP performs better than Z-MAC in terms of energy efficiency and network performance, as confirmed by simulation results shown in Table 4.

The packet delay metric measures the time between a packet generation and the packet reception at the base station, during 300 seconds of simulation. The energy consumption per node metric measures total energy consumed for listening, transmitting, receiving, switching from sleep to idle mode and vice versa, averaged over all nodes in the network. FlexiTP outperforms Z-MAC in the average packet delay because each of a node's routers has already been pre-assigned a slot to relay the node's data to the base station, so that the time it takes for the data to reach from a source node to the base station is reduced and predictable. Furthermore, FlexiTP nodes send their data to their parent at the start of their slot, while Z-MAC nodes need to contend for the medium before sending their data. Z-MAC's small contention window in every slot results in a large slot size that increases data delay and energy consumption. In addition, Z-MAC nodes send explicit congestion notification messages when they experience high contention and this scheme increases the energy consumption per node for propagating explicit congestion notification messages to two-hop neighbourhood and also extends delay for moving toward TDMA operation, on an already heavily-loaded networks.

We also calculated energy consumption per packet, that is, the ratio of total energy consumed by all nodes in the network to the total number of packets received at the sink, during 300 seconds of simulation. Table 4 shows that FlexiTP outperforms Z-MAC

Performance metric	FlexiTP slot reuse off	FlexiTP slot reuse on	Z-MAC
Energy efficiency metrics			
Average energy consumption per node (J)	0.06	0.11	9.45
Average energy consumption per packet (J)	0.0000496	0.0000861	0.0205010
Average network lifetime (days)	3410	1663	20
Network performance metrics			
Average packet delay (seconds)	125.95	79.63	144.26
Average throughput (packets/second)	3.70	4.37	1.54
Average network utilisation (%)	70%	82%	29%

Table 4: FlexiTP versus Z-MAC: performance evaluation

in terms of the average energy consumption per packet and this result consistent with that of the average energy consumption per node. FlexiTP has a very low energy per packet because it guarantees reliable delivery of packets and eliminates overheads (energy wastage) due to collisions, overhearing, idle listening, and over-emitting. On the other hand, Z-MAC suffers contention-related collisions. Based on the average energy consumption per node and node initial energy of 54,000 J, Table 4 shows that FlexiTP prolongs the network lifetime by 83 times (with slot reuse) to 170 times (without slot reuse) compared to Z-MAC.

Table 4 also shows the throughput using FlexiTP versus using Z-MAC, that is, the total number of packets received at the base station from all nodes in the network over 300 seconds of simulation. FlexiTP produces a higher throughput than Z-MAC because every node in the network is guaranteed a path to the base station. Fairness reflects the ability of nodes in the network to share the channel equally. In FlexiTP, each node gets an equally sized time-slot and all nodes in the network can access the channel. Hence, FlexiTP ensures fairness. In high contention networks, the explicit congestion notification scheme of Z-MAC can promote fairness for nodes within one-hop neighbourhood of each other and can limit hidden terminals in a local contention, however, at the cost of overloading the network with the explicit congestion notification packets. This approach results in added delay that leads to a decrease in throughput. Recall that FlexiTP with slot reuse scheme on results in collisions that reduce the network throughput (as shown in Figure 23). However, when the slot reuse scheme is on, a network's data gathering cycle is shorter than a network without slot reuse. Table 4 shows that FlexiTP with slot reuse scheme on results in a higher throughput than FlexiTP without slot reuse because it allows nodes to send more packets over short data gathering cycles in 300 seconds.

We measured and compared the effective channel utilisation of FlexiTP and Z-MAC. The channel utilisation percentage metric is the percentage of total packets received at the base station in bits, per time unit in seconds per channel bandwidth in bytes, which can be formulated as follow (refer to Table 2 and Table 4 for parameter values):

$$Utilisation\_percentage = \frac{total\_packet\_received \times packet\_size \times 8}{simulation\_time \times (channel\_bandwidth \div 8)} \times 100 \quad (8)$$

Table 4 shows that FlexiTP achieves a better bandwidth utilisation than Z-MAC under a large number of senders. This is because FlexiTP allows senders to transmit their packets without contending for the medium at all, while Z-MAC requires senders to compete for their own slots and their one-hop neighbours' slots.

Note that in [94], Rhee et al. show that Z-MAC can sustain a good performance under high contention. Their simulation results were based on low density networks: one-hop benchmark sensor network of 21 nodes and two-hop benchmark sensor network of 18 nodes. In contrast, we simulated high density networks, hence, there are a higher number of routers required for relaying a source node's data to the base station. This multi-hop benchmark intensely increases contentions in local neighbourhoods and in the network overall. In high density networks, Z-MAC performance under high traffic loads degrades significantly.

4.3.4 Fault Tolerance

We tested FlexiTP robustness by focusing on the most problematic scenarios such as adding or removing a substantial number of nodes to or from the network. The fault-tolerance slot (FTS) is set to 500 ms. In the first simulation setup, we switched off a substantial number of nodes in the network gradually over time and so there were increasing numbers of orphan nodes. We evaluated FlexiTP local repair performance in terms of network re-connectivity ratio, energy expenditure, and local repair latency.

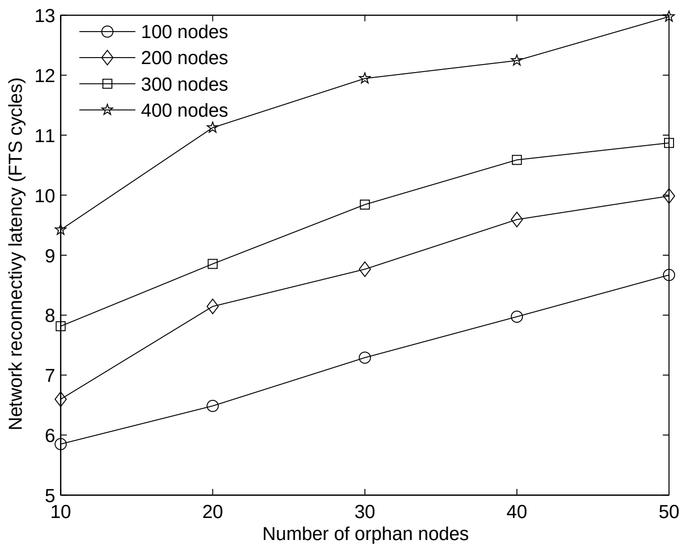

Figure 25: FlexiTP network connectivity re-establishment after a certain number of node failures.

The re-establishment of network connectivity after node failures is depicted in Figure 25. We found that the network connectivity re-establishment after FlexiTP local repair improves as the network density increases. Furthermore, the network gets partitioned as the number of node failures increases. The network connectivity represents the total number of nodes that are actually connected to the tree over the total number of nodes that are expected to be connected to the tree after a number of nodes in the network are switched off. These simulation results confirm that FlexiTP is adaptive to node failures since the remaining nodes still formed a connected network.

The energy expenditure for rebuilding nodes' schedules is free since FlexiTP allows nodes' routers to allocate slots to the nodes utilising existing data gathering and multi-function packets. We measured the average node energy expenditure to re-establish a network connectivity by computing the following energy costs: the cost for an orphan node to send a distress signal to find a parent, the cost for prospective parents to send a reply to the orphan node, the cost for the orphan node to select a parent by sending a confirmation signal, and the cost for the selected parent to send a schedule to the orphan node. Figure 26 shows that the average energy expenditure of a node involved in the local repair is less than 1.2 J that is only around 0.000022% of the node's initial energy, across

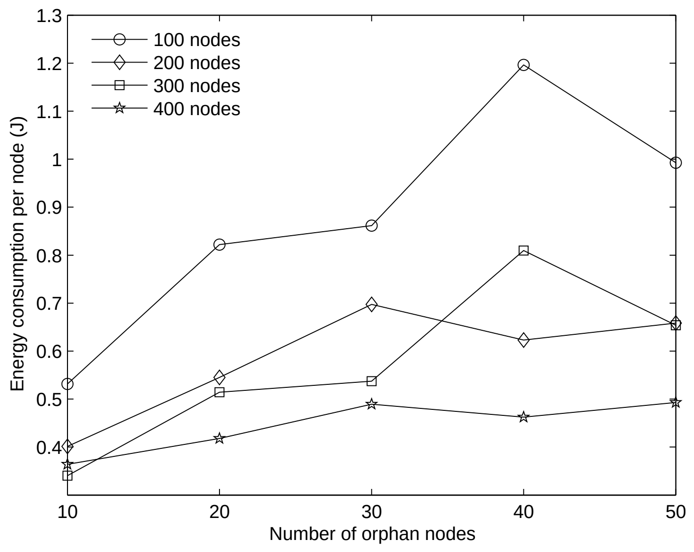

Figure 26: FlexiTP node energy expenditure for re-establishing a network connectivity after a certain number of node failures.

all scenarios.

Figure 27 shows the local repair latency in terms of the total number of FTS cycles required for orphan nodes to get a new parent and re-build their schedules, that is, in the range of 6 to 13 cycles. The local repair latency can be improved by increasing the FTS period. The human manager can customise this feature according to the application's dynamics and needs.

In the second simulation setup, we added a number of new nodes to the networks gradually over time and evaluated how fast FlexTP can incorporate the new nodes to the existing networks. Figure 28 shows that as the number of new nodes introduced to the network increases, the average number of FTS cycles required for these nodes to select a parent increases. Based on the simulation results, it is apparent that FlexTP is robust to node additions. However, FlexTP local repair performance starts to deteriorate when the number of nodes added is too large. This is because new nodes have to wait for a few more FTS cycles before they manage to select a parent. We also observed that the increase in the network density increases the latency of new nodes in selecting a parent. This is as expected because in high network densities, the networks are dense and so

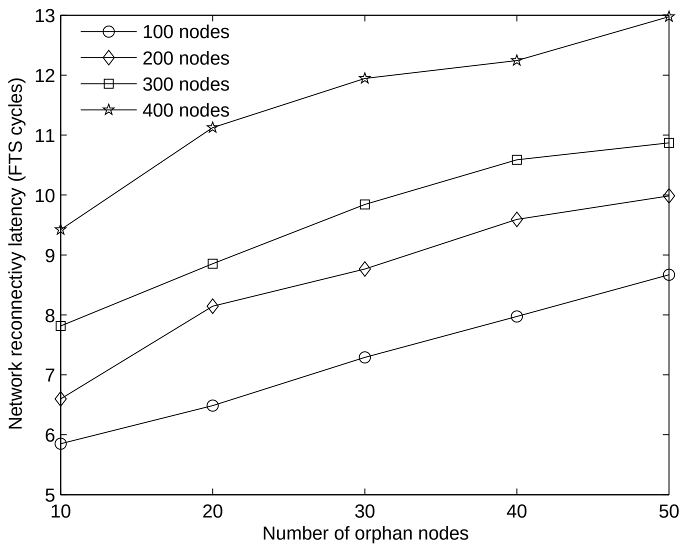

Figure 27: FlexiTP local repair latency in terms of the number of FTS cycles after a certain number of node failures.

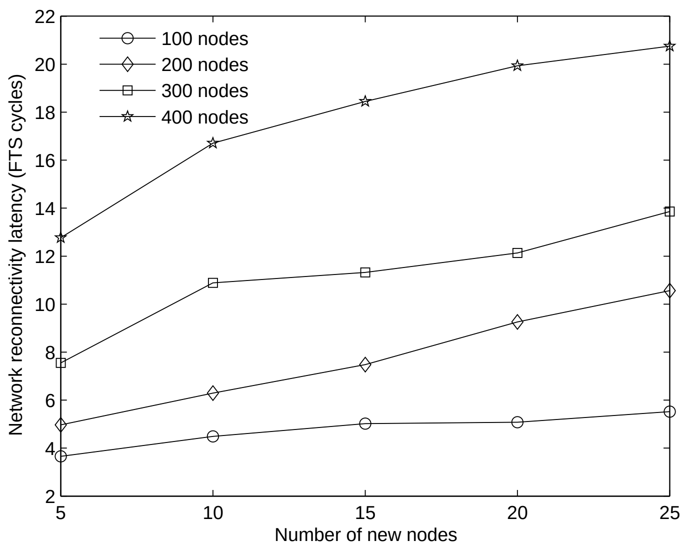

Figure 28: Local repair latency in terms of the number of FTS cycles after a certain number of new nodes join the network.

there are more nodes in the network that can reply to a new node's distress signal, possibly causing nodes to overhear each other's transmission. When that happens, other new nodes that lose in a contention wait for the next FTS and hence increases the latency for selecting a parent. The latency for selecting a parent can be improved by increasing the FTS period and so nodes in a network can take more new children during the FTS period in each data gathering cycle, i.e., more new nodes can join the network in each FTS.

4.4 Conclusion

FlexiTP is a novel TDMA-based protocol for scheduling, routing, and application layer time synchronisation. It provides end-to-end guarantees on data delivery such as predictable throughput for gathered data, fair access to the network for all sensor nodes, and robust self healing, whilst also respecting the severe operating constraints of wireless sensor networks. To the best of our knowledge, FlexiTP is the first integrated protocol for wireless sensor networks that proposes a scheduling and routing scheme that provide a balance among end-to-end guarantees on data delivery, energy efficiency, and memory efficiency.

FlexiTP offers a synchronised and flexible slot structure in which nodes in the network simply build, modify, or extend their schedules based on the local information available to them (receive slot list, transmit slot list, and conflict slot list). FlexiTP utilises TDMA scheduling for efficient data gathering in wireless sensor networks. Nodes are only active during their scheduled slots or else they sleep. This fixed scheduling approach offers a high energy saving by reducing idle listening, avoiding collisions, and avoiding overhearing. FlexiTP uses forwarding-to-parent routing scheme where nodes route packets to their parents until they lose connection with the parents. This routing scheme is suitable for periodic gathering sensor networks because nodes remain static throughout their lifetime.

FlexiTP implements local synchronisation to minimise clock drifts among nodes: a parent synchronises its children during a designated multi-function slot. As the network density increases, the length of data gathering cycle increases. One can increase the frequency of time re-synchronisation within a data gathering cycle by assigning additional multi-function slots in order to provide a tighter time synchronisation.

In FlexiTP, nodes can continue to function accurately in the event of failure of individual nodes (e.g. erroneous network link, dying node, and topology changes) by performing a local repair in the fault-tolerance slot. The length of the fault-tolerance slot can be adjusted according to the application requirements. For example, users can set a very short fault-tolerance slot (e.g., 50 ms) for a sensor network application that is deployed in a stable environment or an application that can tolerate a slow network stabilisation.

Through comprehensive analysis and simulation results we substantiate our argument that FlexiTP is fault-tolerant and energy-efficient across different network configurations. Our simulation results confirm that, in high contention networks, FlexiTP outperforms Z-MAC in terms of energy efficiency, packet delay, and throughput.

Chapter 5

On-Demand TDMA Slot Transfer for Supporting Adaptive Sensing in Wireless Sensor Networks

Change is inevitable . . . adapting to change is unavoidable, it's how you do it that sets you together or apart.

William Ngwako Maphoto

In this chapter, we present a novel on-demand TDMA slot transfer method, our algorithm for self-configuration that enables a network to adapt to changing traffic patterns. The proposed algorithm utilises a TDMA protocol for transferring time slots from one part of the network to another part to support non-uniform and adaptive sensing in different parts of a network. We discuss the design of this protocol and show that it outperforms a CSMA protocol and a hybrid protocol both in terms of energy-efficiency and effectiveness in supporting adaptive sensing.

5.1 Introduction

The use of wireless sensor networks for gathering environmental and safety-critical data in real time is increasing at a rapid rate. Some of the main criteria in designing sensor network architectures are energy-efficiency, self-management, and self-healing. However, most protocols for data gathering and routing in sensor networks implicitly assume a regular rate of data gathering by individual nodes. While this is sufficient for sensing parameters that change slowly over time, individual nodes in a small part of a network may need to increase their rate of data gathering significantly for reporting important data in real-time. For example, in structural monitoring applications, sensor nodes are deployed to monitor vibration (e.g., wind and earthquakes) that could damage the structure of a building [121]. It is critical for sensor nodes to send their data more often to the base station when they detect event triggers such as sensor readings changing rapidly or exceeding user-specified thresholds. CSMA protocols suffer from increased contention for the wireless medium in a small part of the network, which leads to increased message collisions and retransmissions.

TDMA protocols can be used to achieve good energy efficiency by guaranteeing collision-free communication [66]. Data gathering trees are constructed for regular collection of data and nodes are only active (transmit or receive) during their scheduled data gathering slots. TDMA protocols are especially useful for sensor networks where all sensor nodes are transmitting data at the same data rate and can tolerate bounded message latency. These requirements are found in many sensor network applications. Several TDMA protocols have been designed to meet these requirements. However, the rigid slot structure of most traditional TDMA protocols causes under-utilisation of network bandwidth during light traffic generation periods. Typical TDMA implementations assign a time slot for each node in the network. When a node has no data to send, its reserved slot goes unused and it may waste energy on idle-listening. It is crucial for the network to be able to reclaim this lost bandwidth. Thus, an important challenge for TDMA schemes is to allow nodes to adjust their schedules according to their current sensing requirements in order to optimise the utilisation of the network bandwidth and to conserve energy.

In Chapter 2 of this thesis, we describe existing works on modified TDMA protocols that enable nodes to adjust their slot schedules according to traffic conditions through either regular neighbourhood negotiation or node contention for unused slots. Recall that TRAMA [89] periodically recalculates the priorities of nodes within two-hop neighbourhood and rebuilds slot schedules for these nodes. If a node has few packets to send, it may release its slot for the remainder of the frame to other nodes that have many packets to send. This scheme results in a high bandwidth utilisation, however TRAMA does so at the expense of high latency and high algorithmic complexity. Furthermore, TRAMA relies on forecasting of traffic that may be inaccurate. Z-MAC [94] focuses on recapturing wasted slots by allowing nodes to contend for all slots with preference for slot owners. The owner uses its slot only if it has data to send, otherwise it allows other nodes to use its slot. Thus, unlike TRAMA, Z-MAC allows nodes to recapture unused slots without having to renegotiate the slot schedule. However, this approach introduces the hidden terminal problem. In high contention networks, Z-MAC uses explicit congestion notification messages to limit the occurrence of hidden terminals in the local neighbourhood.

These observations lead to the following problem statement: "*How to implement a traffic-adaptive protocol that allows any node to adjust their schedules dynamically and efficiently according to their current sensing states?*". We propose an *on-demand TDMA slot transfer* (OST) algorithm that allows time slots from one part of the network to be transferred to another part. Hence, OST supports non-uniform and adaptive sensing in different parts of a network. OST utilises the flexible TDMA slot structure of FlexiTP to support data-gathering and communication rescheduling seamlessly.

The remainder of the chapter is organised as follows. Section 5.2 describes the proposed on-demand TDMA slot transfer algorithm. Section 5.3 conducts extensive simulations to evaluate the performance of the on-demand TDMA slot transfer algorithm against existing traffic-adaptive protocols in terms of energy efficiency and capability to support non-uniform sensing and traffic pattern fluctuations. Section 5.4 concludes the chapter.

5.2 OST Description

The OST algorithm can be added to any TDMA protocol that allows nodes to build, modify, or extend their scheduled number of slots during execution. In this thesis, we used FlexiTP as a base protocol for OST. The main feature of the FlexiTP protocol is its synchronised and loose slot structure. Nodes can claim or remove a slot based on the current information in their lookup table without exchanging information with any other nodes in the network prior to modifying their schedules. In addition, FlexiTP uses a depth-first-search schedule to reduce data buffering and hence preventing the loss of packets due to buffer overflow.

In OST, FlexiTP slot number 1 is dedicated as a *fault tolerance slot*, a short CSMA period where all nodes in the network are in listen mode. This feature allows nodes to adjust their schedules according to their local neighbourhood states. In addition to performing local repair when network topology changes, the fault-tolerance slot can be utilised to allow nodes to request extra slots from their neighbouring nodes and so they can transfer their data at higher rates. Nodes switch to receive mode for their scheduled *receive slot list*, transmit mode for scheduled *transmit slot list*, or else they switch to the sleep mode. A *multi-function slot* of FlexiTP is utilised for transferring list of slots. OST adopts a top-down multi-function slot structure in which the base station initiates the multi-function packet propagation in a data gathering cycle.

The OST algorithm can be executed either locally or globally. In the local (decentralised) scheme, sensor nodes autonomously adjust their slot schedules based on their local neighbourhood states. Whilst, the central scheme enables the central manager (base station) to reconfigure slot schedules of nodes in the network based on global states of the network. The local OST and central OST will be described in Section 5.2.1 and Section 5.2.2 respectively.

There are four main contributions of our OST algorithm. First, the OST algorithm allows a protocol to reconfigure the traffic patterns of sub-networks in the network based on analysis of sensor data. It enables a node to get extra slots from the other nodes in the network. The requesting node can specify the number of extra slots required in a data gathering cycle and specify or change how long these extra slots will be required.

Second, both local and central OST algorithms provide many functionalities. The main advantage of the centralised OST scheme is that the central manager can distribute loads among sensor nodes to prolong the lifetime of the network by determining which sub-networks need more slots than other sub-networks. For example, if data reported by nodes in certain parts of the network is stable over a period of time, the central manager can reconfigure the traffic pattern of the network to conserve node energy. Depending on the application management policy, the central manager may decide to shut down that sub-network completely and hence shorten the length of data gathering cycle (the TDMA frame length). Nodes in that sub-network are relieved of sensing task and do not send their data to the base station for a period of time. Alternatively, the central manager may allow that sub-network to sense and send data to the base station less frequently, perform data aggregation, or transfer slots from that sub-network to other parts of the network. Thus, central OST provides the potential for better monitoring, control, and management of sensor networks.

Third, when the OST algorithm is used with FlexiTP, FlexiTP's flexible-scheduling and routing scheme ensures slots are systematically transferred among nodes with a low energy overhead and fast convergence in which traffic patterns in the network can be reconfigured and stabilised in one data gathering cycle.

Finally, nodes that borrow or lend slots can resume their normal data gathering TDMA schedules without extra communication cost by simply removing extra slots from their schedules or adding extra slots to their schedules once the on-demand TDMA slot transfer timer expires.

Throughout this chapter, we will use the following terms:

- *Slot-requestor node* refers to a sensor node that requests for extra time slots.
- *Slot-supplier node* refers to a sensor node that allows the temporary use of its time slots by a slot-requestor node.

5.2.1 Local OST Algorithm

In the decentralised scheme, a sensor node becomes a slot-requestor node upon the occurrence of externally triggered events. It then initiates information exchange to try to get extra slots from nodes in its neighbourhood through executing the local OST algorithm. A neighbouring node is a prospective slot-supplier node if it is not asking for extra slots and it has not lent its slots to a slot-requestor node.

A slot-supplier node only lends its data gathering slots not its multi-function slot. The rationale behind this is to keep nodes in the network synchronised at all times, and to allow the human manager to access (query) the nodes even if their data gathering schedule is shut down or borrowed by other nodes. Note that a slot-requestor node uses a slot-supplier node's schedule only for a specified length of time. They then simply resume their previous schedule once this time expires.

Figure 29 (a) shows that nodes B, F, G, and H require extra slots since they sense a sensor reading that exceeds the user-defined threshold. These nodes then become slot-requestor nodes that request for one extra slot in a data gathering cycle from nodes in their neighbourhood, in order to report their readings to the base station more frequently. The main task of the local OST algorithm is to find and select a slot-supplier node. We use four techniques for finding a prospective slot-supplier node to get extra slots from:

1. a slot-requestor node's sub-network,

2. a slot-requestor node's parent's sub-network,

3. a slot-requestor node's ancestor sub-network, or

4. a slot-requestor node's neighbours' sub-network.

We will describe these techniques in the subsequent sections.

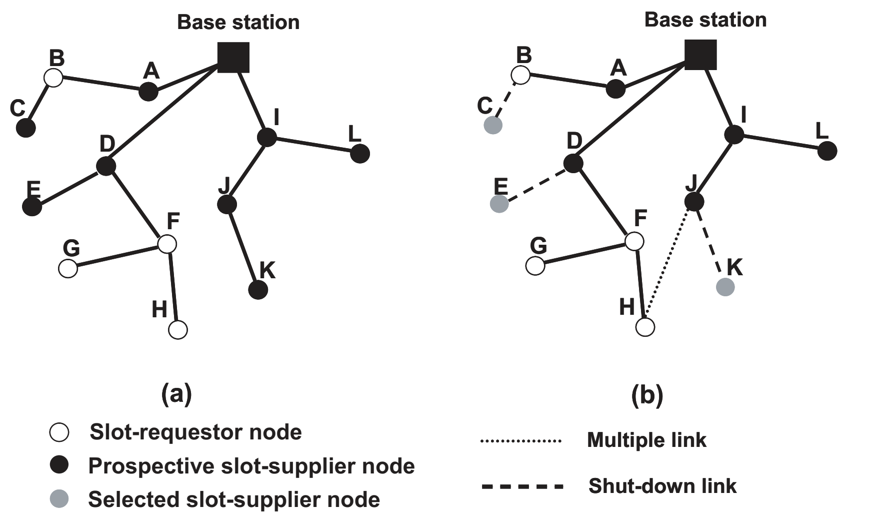

Figure 29: (a) Four slot-requestor nodes in the network: *B*, *F*, *G*, and *H* need to find a slot-supplier node. (b) After they execute the local OST algorithm, *B*, *F*, and *H* manage to get extra slots from the slot-supplier node *C*, *E*, and *K* respectively. Thus, *B* and *F* use their existing path, and *H* uses multiple paths, to send their data to the base station more frequently. Whilst, *C*, *E* and *K* shut down their schedule temporarily.

Slot-requestor Sub-network

Upon detecting event-triggered data, a slot-requestor node first checks whether it can get extra slots from its sub-network. It checks if any of its children and descendants can be prospective slot-supplier nodes. For example, in Figure 29 (a), the slot-requestor node *B* waits for its children and descendants to send their packets to it during their scheduled data gathering slots. The children and descendants are prospective slot-supplier nodes if and only if:

1. they do not piggyback an extra slot request onto packets sent during the data gathering cycle, i.e., they are not slot-requestor nodes, and

2. they do not piggyback a notification that they have lent their data gathering schedule to other slot-requestor nodes, onto packets sent during the data gathering cycle, i.e., they are not slot-supplier nodes.

The slot-requestor node then claims the slot-supplier node's data gathering slots. In the example, node *C*, the only child of the slot-requestor node *B*, can lend its slot schedule to

B. In the multi-function slot, the slot-requestor node *B* then commands the selected slot-supplier node *C* to shut down its slot schedule temporarily. Since the slot-requestor node is one of the slot-supplier's routers, in the next data gathering cycle, the slot-requestor node just claims the slot-supplier's data gathering slot that is already listed in its transmit slot list table, i.e., the transmit slot that is originally used by node *B* to forward node *C*'s data is now used for sending its own data. This approach does not require extra communication slots to perform the local OST algorithm and so adds no extra energy consumption.

Parent Sub-network

If none of the slot-requestor node's children or descendants can become a slot-supplier node, the slot-requestor node tries to get extra slots from its parent's sub-network. It piggybacks its extra slot request onto the data packet sent to its parent using one of its descendant's data gathering slots. The parent then checks if any of its children or descendants other than the slot-requestor node can lend slots. Again, the parent selects the immediate available slot-supplier node for the slot-requestor node. In the multi-function slot, the parent informs the slot-requestor node that the extra slot is available, and also informs the selected slot-supplier node to shut down its data gathering schedule temporarily. For example, Figure 29 (a) shows that node *F* is unable to use its children's data gathering slots (node *G* and *H*) because they are also slot-requestor nodes. Node *F* requests extra slots from its parent, node *D*. Since node *E* is a prospective slot-supplier node, node *D* allocates *E*'s transmit slot to *F*. In the next data gathering cycle, the slot-requestor node *F* uses the slot-supplier node *E*'s data gathering schedule in addition to its own schedule.

Ancestor Sub-network

If none of children or descendants of the slot-requestor's parent can lend their data gathering slots, the parent will propagate the request to the slot-requestor's ancestor. The request will keep propagating to the slot-requestor's ancestors, up to the ancestor that is one-hop away from the base station, until the requested slots are allocated. The slot-supplier selection, slot allocation, and slot transfer mechanisms are similar to that of getting requested slots from the parent's sub network.

The advantage of getting requested slots from nodes on the same branch of the tree is that the slot-requestor shares some similar paths with the slot-supplier and so some of the slot-requestor's routers may not need to rebuild their schedules.

Neighbour Sub-network

If the slot-requestor node does not receive its requested slots within the data gathering cycle in which it sends its request, the slot-requestor waits for the global listening slot, that is the fault-tolerance slot, to send an extra slot request. This technique allows the slot-requestor node to try to get extra slots from its neighbouring sub-networks. In the fault-tolerance slot, a slot-requestor node broadcasts an *OST_signal*, indicating that it requires extra slots. A neighbouring node replies to the *OST_signal* only if it has any child or descendant that is eligible to become a slot-supplier node and so the slot-requestor node can use that node's child or descendant's data gathering slots to send data to the node.

Following the previous example shown in Figure 29 (a), node *G* and *H* are unable to get a slot-supplier node during the data gathering period. In the fault-tolerance slot, node *H* receives a reply from node *J* that it can use node *K*'s data gathering slots (node *J*'s child) and so *H* can use *K*'s slot to send its data to *J*. After the fault-tolerance slot finishes, *J* informs *K* and *K*'s routers to update their lookup table with the new schedule, by piggybacking the schedule update onto data gathering and multi-function packets in the current data gathering cycle. In the next data gathering cycle, *H* uses multiple paths to send its data more frequently to the base station. That is *H* gets extra slots for data gathering by using its slot-supplier node's path in addition to its own path. These multiple paths are:

1. *H - F - D - Base station*
2. *H - J - I - Base station*

The slot-requestor node *G* is unable to get any slot-supplier node within its communication range and so its extra slot request is propagated up to the base station in the next data gathering cycle. This is where the central OST algorithm comes into play.

5.2.2 Central OST Algorithm

The central manager (the base station) has unlimited energy and so it is able to execute a wide-range of management functions by regularly analysing sensor readings reported by all nodes in the network. For example, Figure 30 shows sensor readings in the shaded sub-network change rapidly compared to the rest of the network. This phenomenon cannot be detected locally since individual nodes have no global knowledge of the network. The central OST algorithm is then executed to allocate extra slots for the needy sub-network and so nodes in this sub-network can send their data to the base station more often. In the centralised scheme, the base station performs sensor data analysis, processing, and TDMA slot allocation. Thus, the central OST algorithm is cost-free since it removes the computation burden from energy-constrained sensor nodes. The central OST algorithm consists of four main steps: slot demand, slot-supplier selection, slot assignment, and slot delivery.

Slot Demand

The base station decides the extra slots (let this number be α) required by a slot-requestor node to send its data in a data gathering cycle based on management policy of a particular event. When a node asks for one extra slot, the base station should also allocate one extra slot for each of the slot-requestor node's routers. Thus, the total number of data gathering slots required to meet the slot-requestor node's request is equal to α multiplied by the slot-requestor node's tree level (the number of hops required by the node to reach the base

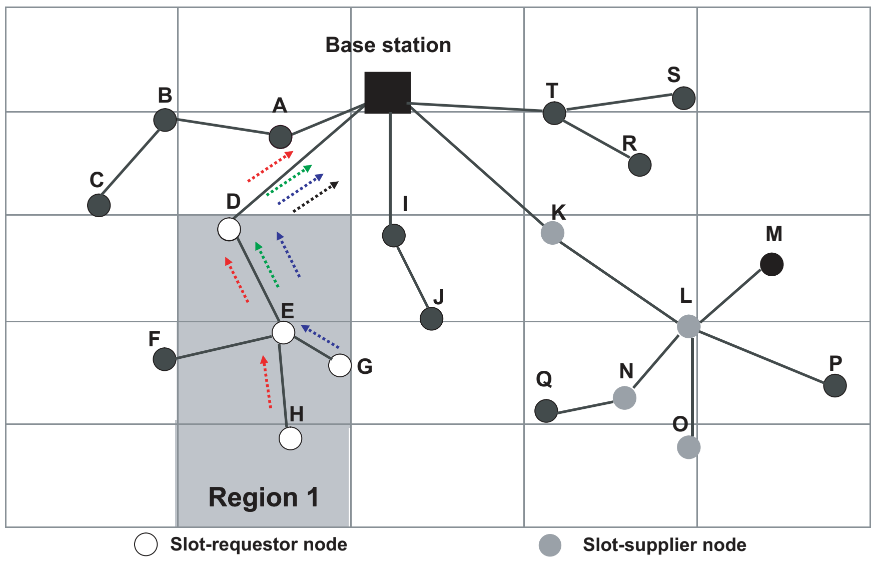

Figure 30: Sensor readings in the shaded Region 1 change rapidly in a period of time. The supply of extra slots from the entire network is higher than the demand of extra slots by slot-requestor nodes in the Region 1.

station). Since the base station knows the network connectivity, it can determine how many hops away the slot-requestor node is from it. For simplicity, each slot-requestor node in Figure 30 requires one extra slot in a data gathering cycle. But in general, each slot-requestor node may ask for more extra slots. Thus, the number of extra slots required by slot-requestor nodes in Figure 30 is equal to nine slots: one slot for D, two slots for E, three slots for G, and three slots for H.

Slot-supplier Selection

The base station determines which nodes are eligible to supply extra slots requested by a slot-requestor node, and determines how many slot-supplier nodes are required to meet the slot demand. A node's data gathering schedule consists of slots that are used by the node and its routers to forward its packets to the base station in data gathering cycles. For example, Figure 30 shows that node M has three data gathering slots: one for itself, one for its router, L, to forward the data sent by M to K, and one slot for router K to forward the data to the base station. Given a demand of n slots, the base station determines how many slot-suppliers are required to meet the demand.

The slot-supplier's ordering according to selection criteria are as follows:

1. *Unreserved slots*. The base station claims unreserved slots from existing slot-suppliers if the number of supplied slots is sufficient to meet the slot-requestor's demand. For example, if node *M* has already given up two out of its three slots to a slot-requestor, there is one unreserved slot that can be claimed by any other slot-requestor.

2. *Equilibrium*. The number of data gathering slots supplied by a slot-supplier is equal to the slot requestor's demand.

3. *Surplus*. The number of data gathering slots supplied by a slot-supplier is higher than the slot requestor's demand.

4. *Deficit*. The number of data gathering slots supplied by a slot-supplier is lower than the slot requestor's demand. Thus, more slot-suppliers are required to meet the slot requestor's demand (includes unreserved slots whenever possible).

In the case where the total demand of all slot-requestor nodes is less than supply, any eligible slot-supplier node can supply their slots to each individual slot-requestor node according to the above criteria. For example, Figure 30 shows that any nodes other than slot-requestor nodes in the shaded Region 1 can lend their slots, e.g., node *K*, *L*, *N*, and *O* can become the slot-suppliers for the slot-requestor *D*, *E*, *G*, and *H* respectively.

Figure 31 shows an example where extra slots in the network are limited and demand is higher than supply. We propose three techniques to address this issue:

1. Data aggregation.

2. Round-robin cycle strategy.

3. Combination of data aggregation and round-robin cycle strategy.

The data aggregation technique can address the demand-supply problem if slot-requestor nodes within a sub-network report similar data values over a period of time. The nodes'

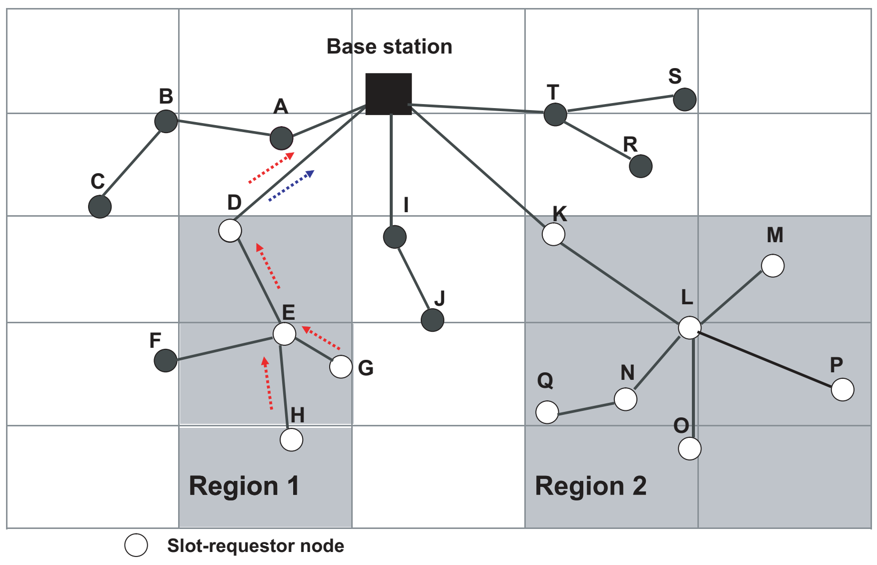

Figure 31: Sensor readings in both shaded Region 1 and 2 change rapidly. The supply of extra slots is less than demand of extra slots by slot-requestor nodes in the Region 1 and 2.

data can be aggregated instead of forwarding the nodes' individual data, hence reducing the number of extra slots required. For example, the slot-requestor node *E*, *G*, and *H* in Figure 31 reported similar data values in the past three data gathering cycles because their locations are within proximity of each other. The base station then allocates fewer extra slots to these nodes by allowing node *D* to aggregate these nodes' data prior to sending them. Instead of allocating nine extra slots, only five extra slots are assigned to these nodes: one slot each for *H* and *G* to send data to *E*, one slot for *E* to send the aggregated data of its children and its own data to *D*, one slot for *D* to forward the aggregated data received from *E* to the base station, and one slot for *D* to send its own data to the base station.

In the round-robin strategy, slot-requestor nodes may share the same extra slots and use them alternately in alternate data gathering cycles. For example, slot-requestor nodes in the shaded Region 1 can use extra slots in the immediate data gathering cycle after receiving them in the current cycle, then wait for one cycle before using the extra slots again. Consequently, when slot-requestor nodes in the shaded Region 2 receive the extra slots in the current cycle, they need to wait for one more data gathering cycle prior to using the extra slots.

Slot Assignment

The base station is responsible for ensuring that the supplied slots do not conflict with the schedule of slot-requestor nodes and their routers (parents and ancestors). This approach avoids schedule duplication and collisions by simply checking whether the supplied extra slots exist in the tables (receive slot list, transmit slot list, and conflict slot list) of a slot-requestor node and its routers. The base station performs this function by coupling node schedule information with node neighbourhood information and so the base station can construct the receive slot list, transmit slot list, and conflict slot list table of nodes in the network.

Figure 30 illustrates that the slot-supplier *C* can lend its schedule to the slot-requestor *E*. Say, node *C*'s data gathering schedule is slot numbers 5, 6, and 7. Since the slot-requestor *E* requires two extra slots, only two slots of *C* are given to *E*: slots 5 and 6. Therefore, the surplus slot number 7 becomes an unreserved slot that can be claimed by any other slot-requestor. The base station checks whether slot number 5 is in *E*'s tables. Since *E* is one of *C*'s neighbours, *C*'s transmit slot list table is recorded in *E*'s conflict slot list table. However, now *E* can use slot number 5 because *C* has given up its schedule, in that, *C* will no longer be transmitting data during that slot for a specified period. The same slot checking is performed at each of the slot-requestor node's routers. Thus, the next slot checking is performed on node *D* (*E*'s parent) and *B* (*C*'s parent). If a slot-supplier node shares the same path as the slot-requestor node, a slot checking is only performed on uncommon paths. In Figure 30, the slot-supplier *F* and the slot-requestor *H* share the same path at node *D* and *E*. Node *H* can simply use *F*'s existing data gathering schedule that is already listed at their common routers. Furthermore, in the case where a number of slot-supplier nodes is required for a single reporting node, the base station first filters and sorts extra slots provided by the slot-supplier nodes in increasing order, and then the similar slot checking process is executed.

Slot delivery

In the multi-function slot cycle of FlexiTP, the base station propagates the extra slot numbers allocated to slot-requestor nodes and information on how long these requesting nodes can use those extra slots. The slot allocation information is piggybacked on the multi-function packet. The slot allocation information consists of:

1. a list of extra slots,
2. a list of slot-requestor nodes that should update their schedules with the allocated extra slots,
3. a list of slot-supplier nodes that should release their slots, and
4. the timer for the extra slots and temporarily released slots (in terms of the number of data gathering cycles).

A sensor node in the network that receives the multi-function packet checks whether its ID or any of its descendant's ID is on the list of slot-requestor nodes or slot-supplier nodes. If a node or a node's descendant is on the list of slot-requestor nodes, the node updates its table with the allocated extra slot and sets the timer for the extra slot. If a node or a node's descendant is on the list of slot-supplier nodes, the node shuts down the corresponding released slot (i.e., the data gathering slot that belongs to the node or the node's descendant), and also sets a timer for the temporarily released slots. The node then updates the multi-function packet by removing the retrieved slot allocation information from the packet prior to propagating this packet to its children. If a node and a node's descendants are not on the list, the node will drop the piggybacked slot allocation information and only propagate the multi-function packet. In this way, the slot allocation information is only propagated to intended nodes, not all nodes in the whole network. Thus, the central OST algorithm allows the network to reconfigure its traffic pattern within one data gathering cycle in an energy-efficient manner, i.e., in a data gathering cycle, the slot-requestor nodes can get extra slots and the slot-supplier nodes are informed to shut down their data gathering schedule.

After the timer for borrowing or releasing extra slots expires, the slot-requestor nodes and their routers remove extra slots from their tables, while the slot-supplier nodes and their routers resume their previous schedule. Note that the base station can extend the timer before the timer expires by simply propagating timer extension information in the multi-function slot cycle.

5.3 Performance Evaluation

This section provides detailed simulation evaluations of OST against existing traffic-adaptive protocols, addressing their energy efficiency and effectiveness in satisfying a variety of demands in different network scenarios and configurations. Our main objective is to provide performance comparison of channel allocation technique using TDMA, CSMA, or a combination of TDMA and CSMA.

We implemented the OST algorithm on the top of FlexiTP's flexible TDMA schedule. The code is downloadable at [69]. For simplicity, in all simulations, a slot-requestor node is restricted to ask for one extra slot in a data gathering cycle. Though, OST allows a slot-requestor node to ask for more extra slots. In addition, we utilised FlexiTP without its slot reuse mechanism in order to guarantee collision-free traffics.

We simulated behaviours of a CSMA-based MAC protocol by utilising the RTS/CTS and DATA/ACK mechanisms of MAC802.11 implementation in NS-2 version 2.30 (refer to Chapter 3 for details). The CSMA-based MAC protocol is a contention protocol where sensor nodes wishing to transmit have to first listen to the channel and send their packets if the channel is clear. This channel access technique can easily allocate TDMA slots on-demand and adapt to changing traffic conditions because sensor nodes do not need to exchange information with their neighbours prior to sending their messages.

We simulated NS-2 implementation of Z-MAC, which can be found in [93]. Z-MAC is a hybrid MAC protocol that starts off as CSMA and switches to TDMA if the network load increases. If a node experiences high traffic loads, it will propagate an explicit

Table 5: Simulation parameters for evaluating the performance of traffic-adaptive protocols: OST, Z-MAC, and the CSMA-based protocol.

Simulation parameters	Description
FlexiTP-OST	
`Data gathering slot size`	27 ms
`Multi-function slot size`	27 ms
`Fault-tolerance slot size`	0 ms
`Routing protocol`	Forwarding-to-parent
Z-MAC	
`TDMA slot size`	50 ms
`Owner contention window size (To)`	8
`Non-owner contention window size (Tno)`	32
`Routing protocol`	AODV
CSMA	
`Routing protocol`	AODV

congestion notification message to limit congestions and hidden terminals in its two-hop neighbourhood.

5.3.1 Simulation Setup

Table 2 in Chapter 3 presents our NS-2 simulation parameters. Table 5 presents simulation parameters of OST, Z-MAC, and CSMA. All of the simulation results are based on the mean value of 20 different network topologies involving 100 to 400 nodes with a transmission radius of 60 meters (m), located randomly in a network area of 300 m x 300 m.

We simulated data gathering sensor networks with non-uniform inter-packet arrival rate for 200 seconds. A certain number of nodes in the network perform data gathering tasks with inter-packet arrival of two seconds (low data rate), while other nodes increase their offered loads with inter-packet arrival of one-second (high data rate). The latter nodes are referred to as slot-requestors.

Recall that in FlexiTP, the fault-tolerance slot is basically a CSMA period in each data gathering cycle used for fault tolerance and utilised for slot allocation from the local neighbourhood. Since the simulated networks do not experience topology changes, this fault-tolerance slot will be wasted if nodes have no slot requests. We set the value of the

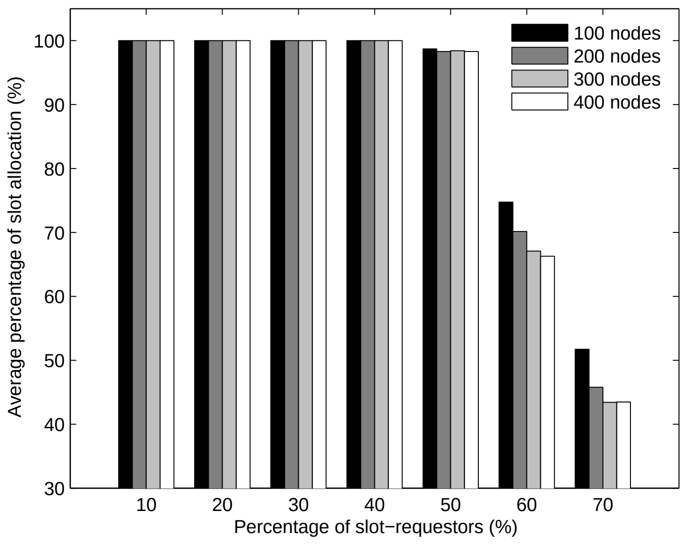

Figure 32: OST percentage of demand allocation versus total percentage of nodes in the network with demand, across various network densities. For example, in the network of 400 nodes, if 70% of nodes become slot-requestors, it means that 280 nodes in the network ask for one extra slot in a data gathering cycle.

fault-tolerance slot to be 0 ms. Although this setup will disable nodes from getting slot allocations from their neighbouring nodes as described on page 89, it allows FlexiTP-OST to be comparable with Z-MAC and CSMA simulation setup. Furthermore, we switched off the data aggregation and round-robin strategy scheme of OST because both Z-MAC and CSMA assume no data aggregation.

5.3.2 OST Slot Allocation

Figure 32 shows the percentage of demand (slot) allocation using OST versus diverse demand, across various network densities. In low to medium demand cases, FlexiTP-OST can satisfy more than 98% of the slot demand. As the demand increases, the supply of extra slots in the network decreases and so results in a reduced slot allocation. Thus, in high demand cases, FlexiTP-OST slot allocation drops around 23% to 54%. Under reasonable network conditions, where the total slot-requestors are up to 50% of total nodes in a network, simulation results show that FlexiTP-OST sustains a high percentage of slot allocation because when a slot-requestor node fails to get extra slots locally, it

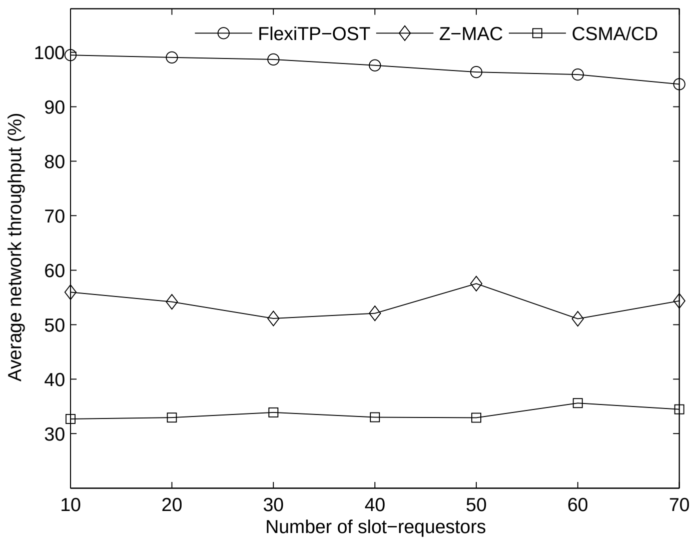

Figure 33: Percentage of network throughput versus total nodes with demand.

may get the requested slots centrally through the central OST. Furthermore, the data aggregation and the proposed round-robin cycle technique would enable OST to take even more demand.

5.3.3 Network Performance

In the next section, 100 sensor nodes form a network over a 300 m x 300 m terrain. The inter-packet arrival rate of nodes in the network is similar to the previous setup, where some nodes have low data rate (one packet generated every two seconds) and some nodes have high data rate (one packet generated per second), over 200 seconds simulation. We compared and evaluated the performance of FlexiTP-OST, Z-MAC, and CSMA in terms of energy efficiency and capability to support non-uniform sensing and traffic pattern fluctuations.

Throughput

Figure 33 shows the average network throughput using either FlexiTP-OST, the CSMA-based protocol, or Z-MAC protocol. The network throughput metric is the ratio of the

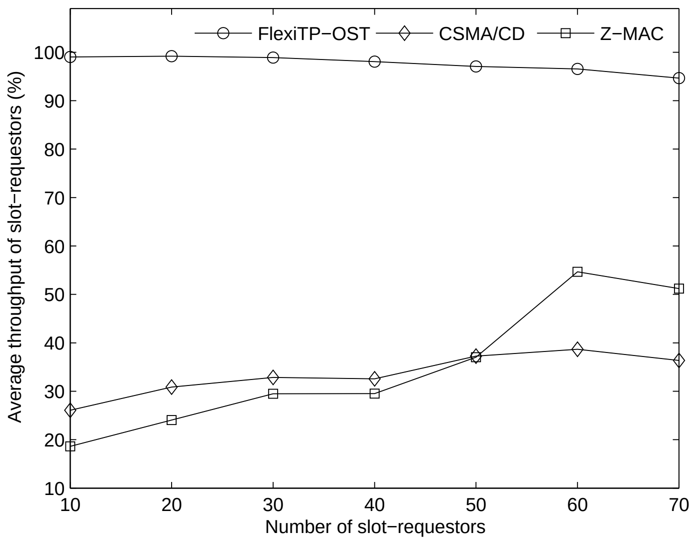

Figure 34: Percentage of slot requestor throughput versus total nodes with demand.

overall packets received at the base station to the total packets sent by nodes in the network, during 200 seconds of simulation. The graph shows that FlexiTP-OST sustains a good network throughput independent of the number of slot-requestors. Z-MAC improves the network throughput by around 20% as compared to CSMA.

We also measured the throughput for slot-requestor nodes, that is, the total slot requestors' packets received at the base station over the total number of packets sent by the slot requestors during 200 seconds simulation. Figure 34 shows that FlexiTP-OST enables more than 95% of slot-requestors (nodes with high data rate) to send their data to the base station successfully, across various demands. As the number of slot-requestors increases, the performance of Z-MAC improves, particularly Z-MAC outperforms CSMA in high demand cases. This is because Z-MAC gradually moves toward TDMA operation when the traffic load in a local neighbourhood increases. Z-MAC performs worse than OST because of the overhead of explicit congestion notification packet propagation on already heavily-loaded networks. As shown in Figure 35, Z-MAC requires more than 55 explicit congestion notification (ECN) packets to allow nodes to activate their full TDMA scheme under high contention.

In all simulation scenarios, FlexiTP-OST outperforms both Z-MAC and CSMA in terms

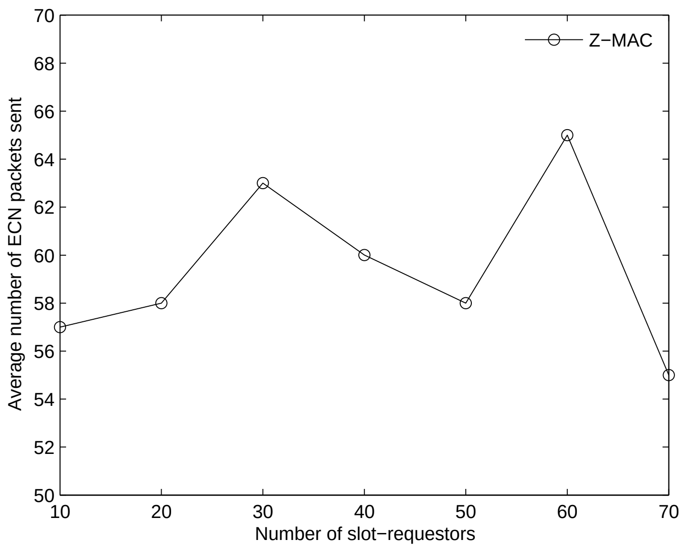

Figure 35: Total explicit congestion notification packets versus total nodes with demand.

of the average throughput. This shows that OST on the top of FlexiTP's TDMA scheme performs better than a CSMA scheme and a combination of CSMA and TDMA scheme, in supporting non-uniform traffic patterns. In FlexiTP-OST, nodes including the slot-requestors have distinct transmission slots and so collision-free traffics can be guaranteed. This reliable delivery ensures high throughput as no packets will be dropped due to collisions. On the other hand, Z-MAC can only limit the occurrence of hidden terminals within two-hop neighbourhood. CSMA does not have deterministic transmission characteristics and so reliable packet delivery and a predictable throughput rate under a variety of conditions, cannot be ensured.

Packet Latency

We measured the average latency of packets sent by all networked nodes and the average latency of packets sent by slot-requestors, as shown in Figure 36 and Figure 37 respectively. The packet latency metric calculates the time between a packet generation and the packet reception at the base station.

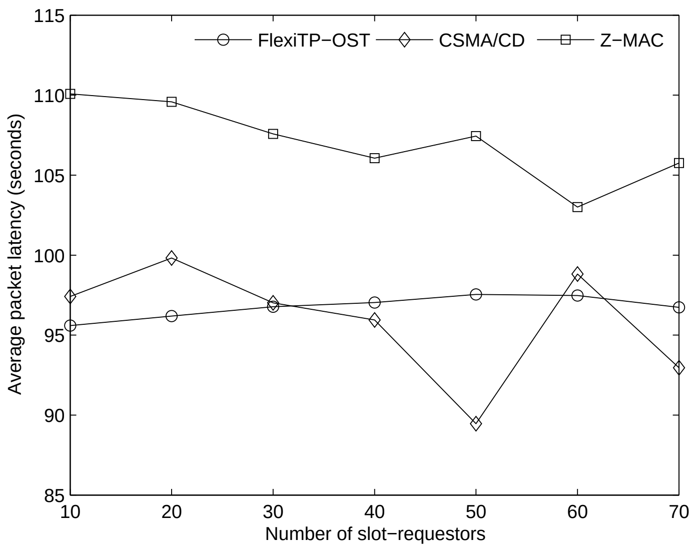

Figure 36: Network packet latency versus total nodes with demand.

Figure 36 shows that FlexiTP-OST provides a lower packet latency than Z-MAC because FlexiTP-OST allows source nodes to send their packets without contending for the medium at all, while Z-MAC requires source nodes to compete for their own slots and the slots of their one-hop neighbours under high contention. Furthermore, in FlexiTP-OST, each of a node's routers has already been pre-assigned a slot to relay the node's data to the base station, so that the time it takes for the data to reach from a source node to the base station is reduced, while Z-MAC routers need to contend for the medium before forwarding the source node's data. On the other hand, the network packet latency of CSMA is unpredictable due to its non-deterministic channel access scheme.

Figure 37 shows that as the number of slot-requestors increases, the average slot requestor packet latency of either Z-MAC or CSMA increases, whilst FlexiTP-OST decreases. This is because CSMA and Z-MAC suffer increased contention on local neighbourhoods in high demand cases. Recall that in Z-MAC, when nodes detect high contentions, these nodes start to propagate explicit congestion notification packets and this scheme results in added latency. In contrast, FlexiTP-OST provides a lower packet latency than Z-MAC because it utilises the depth-first-search-forwarding scheme. Furthermore, as the number of slot requestors increases, there are more and more slot-requestors close to the base

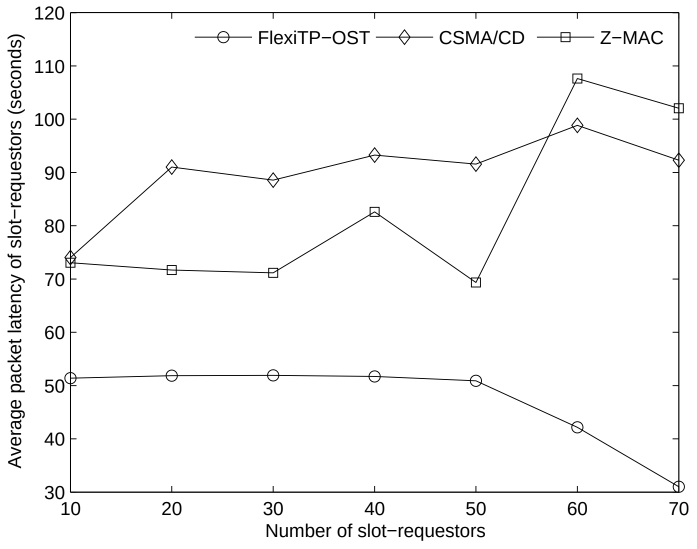

Figure 37: Slot-requestor packet latency versus total nodes with demand.

station and this reduces the overall packet latency of slot requestors.

5.3.4 Energy Efficiency

We observed the energy consumption per packet, that is, the ratio of overall energy consumed by nodes in the network to the total number of packets received at the sink. Figure 38 shows that FlexiTP-OST outperforms Z-MAC and CSMA in terms of average of energy consumption per packet. In FlexiTP-OST, when slot-requestors use extra slots to send their packets, slot-suppliers of these slot-requestors shut down their schedules and so they do not send their packets. Hence, FlexiTP-OST energy consumption per packet is constant across various demands.

FlexiTP-OST provides a very low energy expenditure per packet because it guarantees reliable delivery of packets and eliminates overheads due to collisions, overhearing, idle listening, and over-emitting. FlexiTP-OST allows nodes to sleep when they are not scheduled to transmit or receive and so achieves significant energy savings. In contrast, Z-MAC and CSMA suffer contention-related collisions. Nodes in Z-MAC and CSMA are awake at all times to stand by for receiving packets destined for them and so this

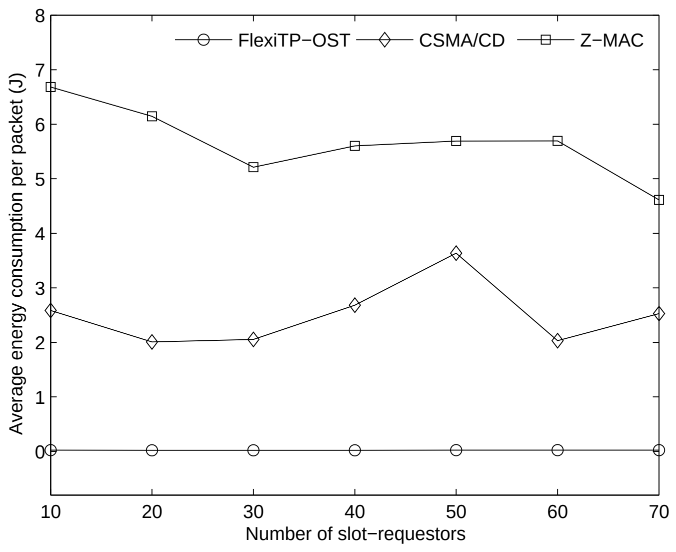

Figure 38: Energy consumption per packet versus total nodes with demand.

increases idle-listening. Even though CSMA can reduce collisions through RTS/CTS handshake mechanism, increased contention in the network results in high collisions and retransmissions. Z-MAC energy consumption per packet is higher than CSMA. This is because, under high traffic loads, Z-MAC performs a two-hop explicit congestion notification propagation scheme to limit the occurrence of hidden terminals.

5.4 Conclusion

OST is a novel on-demand TDMA slot transfer algorithm for transferring time slots between nodes to support non-uniform and adaptive sensing in different parts of a network. The proposed algorithm can be executed locally or centrally. Autonomous sensor nodes perform the local on-demand TDMA slot transfer algorithm based on their local neighbourhood states. Upon the occurrence of externally triggered events, a sensor node becomes a slot-requestor node and initiates information exchange to try to get extra slots from nodes in its neighbourhood. The base station regularly analyses sensor data reported by all sensor nodes and performs the central on-demand TDMA slot transfer algorithm when it detects event triggers or receives slot requests from nodes in the network. In this way, OST provides the potential for better monitoring, control, and management of sensor networks.

Our simulation results confirm that the proposed OST algorithm on FlexiTP performs better than the CSMA-based protocol and Z-MAC (hybrid MAC protocol) in terms of energy efficiency and performance in supporting dynamic traffic patterns. OST is energy-efficient because FlexiTP's flexible-scheduling and routing scheme ensures slots are transferred among nodes with a low energy overhead. In the simulations, a slot-requestor node only asked for one extra slot in a data gathering cycle, i.e., it wishes to send one extra packet per time unit. CSMA and Z-MAC will perform much worse if a slot-requestor node asks for more than one extra slot.

Chapter 6

Peer-to-Peer Multicasting Overlay over TDMA Schedule for Reactive Wireless Sensor Networks

We will surely get to our destination if we join hands.

Aung San Suu Kyi

In this chapter, we present a novel TDMA-driven protocol that can seamlessly support data gathering and on-demand peer-to-peer communication through a flexible TDMA slot structure. We discuss the design of this protocol and show that it outperforms CSMA protocols in terms of latency and performance in supporting reactive sensing.

6.1 Introduction

Recent research explores a whole new range of sensor network applications that can support sophisticated real-time analysis. In such applications, sensor nodes are not only used for surveillance or monitoring, but also to detect events of interest and respond to those events accordingly. In these scenarios, a wireless sensor network becomes a

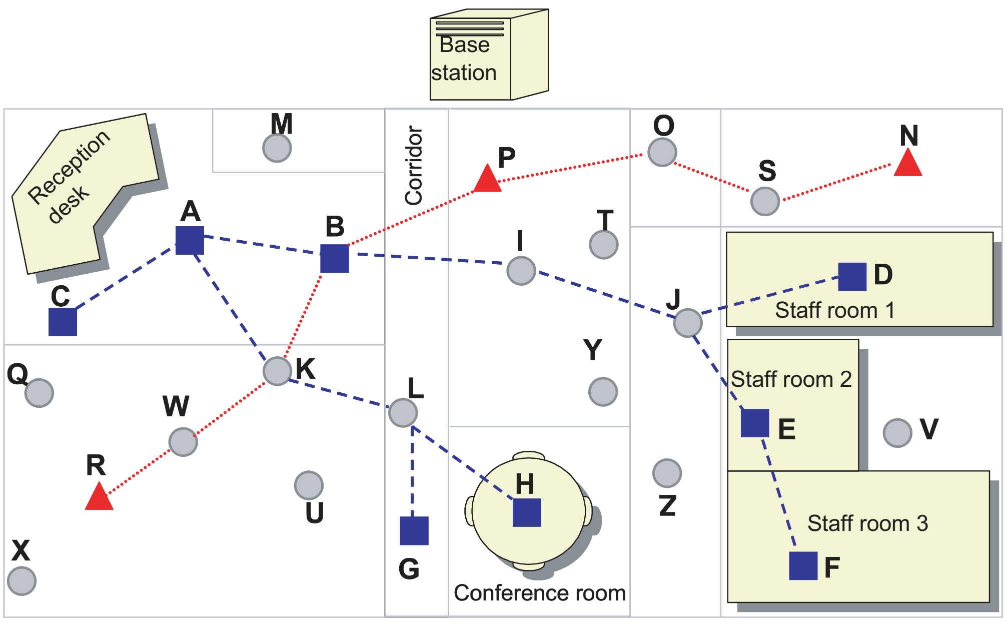

Figure 39: A smart-office application with two P2P-groups: 1) three peer-nodes marked with a triangle form a logical P2P-group that are responsible for controlling temperature in different office rooms and 2) eight peer-nodes marked with a square are physical peers that are responsible for controlling temperature around the office corridor and the conference room based on the information disseminated by nodes located in the reception and staff rooms.

reactive system. However, most wireless sensor networks are organised hierarchically, with the base station serving as the single sink for data sent by sensor nodes. While this communication pattern is sufficient for monitoring applications, individual sensor nodes may need to send their data to multiple destination nodes across the network in order to execute a distributed cooperative-function based on their local environment. This peer-to-peer (P2P) structured communication pattern in wireless sensor networks is attractive not only because it allows information collected by a node to be disseminated to a subset of nodes in the network without going through the base station, but also because it makes sensor networks more reactive to triggers from the environment.

A *P2P-group* is a subset of nodes that need to exchange information. Nodes in a P2P-group are *peer-nodes* that share a common interest and are able to communicate and collaborate with each other to perform a distributed task. The human manager may want to organise nodes in the network into various logical or physical P2P-groups and assign specific tasks to each group. To motivate our research, consider the smart-office application in Figure 39. Nodes *R*, *P*, and *N* with their temperature sensors form a logical P2P-group. These logical peer-nodes are responsible for controlling air conditioners in different locations in the office. On the other hand, nodes *A*, *B*, *C*, *D*, *E*, *F*, *G*, and *H* are physical peer-nodes. Nodes *A*, *B*, and *C* are responsible for monitoring object movements in the reception area, nodes *D*, *E*, and *F* are responsible for monitoring object movements in staff rooms, while nodes *G* and *H* monitor the temperature in an office corridor and a conference room respectively. Suppose the human manager wants to use *G* and *H* to control the air conditioner around the conference room area based on data fed by either nodes in the reception area or staff rooms. Furthermore, if the reception area gets too busy, staff may want to be aware of the number of visitors in the reception area for administrative purposes. To achieve these adaptive behaviors, these peer-nodes should be able to share their data with each other if any of them detects events of interest.

The support of P2P communication is also useful for safety-critical data gathering applications that require sensor nodes to detect event triggers and autonomously adapt themselves in the modified state of the physical environment. For example, sensor nodes are used to navigate users through a monitored environment in the presence of hazardous events (e.g., a chemical spill, a traffic accident [18] [72]), track a fire as it spreads and form a perimeter for the fire area [42], and follow the movement of a source to be tracked (e.g., object tracking [12], trespasser detection [34], traffic control [102]).

Traditional distributed algorithms to support P2P communication in wireless sensor networks rely upon a CSMA-based flooding scheme [50, 51, 57, 74, 80]. One of the notable flooding-based routing protocols for wireless sensor networks, namely directed diffusion [57], allows any node to become a source node, or sink for any other node's data in the network. The main difference is that directed diffusion discovers source-sink trees by flooding and also can only support one-to-many, not many-to-many communication pattern. In directed diffusion, when a node poses a query it becomes a sink and the query is disseminated throughout the network. Nodes with data matching the interest then propagate their data towards the originator of interest or sink. Whilst, the P2P communication problem focuses on allowing any nodes in a network to communicate with any specific subset of nodes across the network. The idea of CSMA-based-flooding of directed diffusion can be extended to support P2P communication by allowing nodes to keep propagating their data in the network until the data are received by the intended sinks. Although this scheme avoids the overheads of routing and path discovery, it uses excessive network bandwidth as each node query results in a large number of messages flooding the network. Furthermore, increased contention for the wireless medium in a small part of the network leads to increased message collisions and retransmissions. Thus, CSMA-based approach such as flooding does not scale well due to high communication cost and high latency.

TDMA-based protocols can be used to achieve good energy efficiency by separating nodes in time, which means that nodes are only active during their scheduled slots. However, they lack P2P support due to their rigid slot structures. The TDMA scheme is especially beneficial over CSMA scheme for sensor networks that require certain performance quality assurances such as high energy savings and bounded latency. In TDMA-based sensor networks, data gathering trees are constructed for periodic collection of data, however, P2P communication is an irregular requirement for supporting reactive sensing. The challenge is to design a protocol for sensor networks that is able to support infrequent peer-to-peer communication between nodes in a data gathering network along with the network's standard routing tree for periodic data gathering.

These observations lead to the following problem statement: "*How to deploy a distributed, adaptive, and reactive data gathering sensor network that allows any node to communicate dynamically with any other node in the network efficiently, without going through the base station?*". We propose a *wireless sensor network peer-to-peer* (WiseP2P) algorithm that allows each node in the network to communicate with any subset of nodes (peers) in the network, directly or through other nodes if its peers are outside the communication range of the node, using scheduled time slots. The WiseP2P algorithm allows nodes to claim extra time slots from other nodes if they want to communicate with their peers. This P2P-communication framework facilitates any implementation of distributed-function that requires sensor nodes to work cooperatively based on their local environment. WiseP2P leverages the distributed and flexible slot structure of FlexiTP and extends the on-demand TDMA slot transfer algorithm, to support data-gathering and on-demand P2P communication efficiently and seamlessly. P2P allows reactive sensing in different parts of a sensor network. This research, to our knowledge, is the first attempt to study P2P-communication between nodes in wireless sensor networks.

There are four main contributions of our WiseP2P algorithm. First, it enables a node to reserve a number of communication slots from the network in order to disseminate its query or data to its peers. Thus, WiseP2P supports the implementation of any type of distributed-cooperative function by facilitating the sharing of time slots and sensor data between participating nodes. The requesting node can specify the number of extra slots required in a data gathering cycle and specify or change how long these extra slots will be required.

Second, WiseP2P slot allocation can be performed locally or centrally. In the local (decentralised) scheme, sensor nodes can aggressively claim extra slots based on their local neighbourhood states. The central scheme uses the base station to which a requestor's query is directed, to allocate extra slots for the requestor, but P2P communication still takes place between peers. The main advantage of the central scheme is that it removes the computation burden of finding extra slots from energy-constrained sensor nodes.

Third, WiseP2P utilises TDMA-scheduling mechanism that allows the co-existence of P2P traffic and data gathering without interference. Furthermore, when WiSeP2P is used with FlexiTP, FlexiTP's flexible-scheduling and routing scheme ensures slots are transferred among nodes with a low energy overhead.

Lastly, the design of WiseP2P algorithm takes into account the unique requirements of wireless sensor networks: lightweight-operation, robustness, and scalability. WiseP2P provides efficient mechanisms for communication among nodes in the network by guaranteeing collision free communication, avoiding of routing all traffic through the base station, and pre-selecting P2P paths. WiseP2P is robust and scalable because it uses a distributed hash tables (DHT)-based scheme to avoid flooding when initiating communication among nodes in the network. Unlike DHT Internet protocols [99], where nodes maintain routing information of all pairs of nodes in the system, WiseP2P's DHT-based lookup table is used by requiring each node to *only* maintain a list of immediate relay nodes' IDs that are used for reaching any other node in the network. Basically, a node selects a P2P-relay node based on its local routing table and lookups are executed by following references from node to node until the destination peer is found. WiseP2P adapts to network dynamics by allowing nodes to adjust their internal tables dynamically to repair P2P communication failures and to reflect newly joined nodes as well as node failures. This preserves a node's ability to reach its peers in the network.

This chapter is organised as follows. We first describe related P2P algorithms in wireless sensor networks in Section 6.2. We then describe our WiseP2P approach in Section 6.3. In Section 6.4, we evaluate our approach through simulation and show how WiseP2P allows adaptive scheduling, achieves energy savings, and ensures good network latency. We also show that the proposed TDMA-based P2P communication scheme is proved beneficial over CSMA schemes in terms of energy efficiency and effectiveness in satisfying various P2P communication demands, especially in heavily-loaded networks. Section 6.5 concludes the chapter.

6.2 Related Work

A prominent research avenue of P2P communication in wireless sensor networks is local query-based protocols. Several researchers have developed query-processing algorithms that allow users to query data from nodes in sensor networks (e.g., through PDA) without requiring the base station to disseminate the query [6, 34, 43]. Demirbas and Ferhatosmanoglu [34] proposed a hierarchical peer-to-peer cluster tree based on R-tree algorithm [46] to perform nearest neighbour queries, whereby nodes can query other nodes at lower cluster level. In [102], each node informs which type of sensor data (let it be *d*) it provides to *d*'s root. The proposed algorithm uses the Tapestry-location-aware based algorithm to look up for service *d* by informing users the information about the location of source nodes providing service *d*. Whilst, Ali and Uzmi [6] proposed a Chord-based [108] lookup table that allows users to query a service from a subset of nodes in the network. In principle, nodes check if they provide the requested service in their local lookup tables, otherwise nodes keep propagating the query to nodes closest to meeting the requested service. Ganesh [43] proposed a PUBLISH/SUBSCRIBE model where a subset of sensor nodes in the network act as sinks for the subscription and publication. The PUBLISH/SUBSCRIBE algorithm allows users to pre-program subscriptions (condition that needs to be meet) at nodes. When nodes detect such events, they will publish their data to local sinks and users then retrieve data from those sinks.

The strategies adopted by previously reviewed query-processing algorithms require nodes to form clusters and cluster heads are elected to act as local base stations. Nodes are restricted to communicate with the cluster head within a cluster. Thus, P2P communication between nodes in the network is not directly supported.

Ali and Langendoen [4] took P2P research in wireless sensor networks to another spectrum. They proposed Tiered Chord (TChord), an underlying architecture for supporting the integration of wireless sensor networks with IP networks (Internet). TChord uses a distributed hash table-based P2P overlay over traditional sensor networks in order to eliminate the need of proxy servers (like in the IP networks) and to provide flexible access to sensor node data. The research scope of TChord is different from ours. TChord focuses on enabling sensor networks to interact with IP networks, while we focus on interactions among nodes within the sensor network.

P2P communication between nodes in the network can occur either 1) through the base station or 2) through direct communication. In the first approach, by utilising FlexiTP, a node with data to send, can route its data to the base station by piggybacking the data onto data gathering packets. The base station then routes the data to the node's peers by piggybacking it onto multi-function packets. The advantage of this scheme is that a node can communicate with its peers without modifying the TDMA slot structure. However, when a node has more data packets to send (say, sensor readings changing rapidly), the base station still routes these packets one at a time in the multi-function slot, because the multi-function slot only can carry one data packet. Thus, the main drawback of this scheme is that for n P2P-packets to be sent, n data gathering cycles are required for sending those packets to the destination nodes. We argue that this is a limitation in any TDMA protocol for data gathering as the flow of sensor data is usually from the nodes to the base station. If a TDMA protocol wants to implement a significant amount of data flow from the base station to sensor nodes, it has to allocate a large number of slots during the building of the TDMA schedule. However, these slots will not be used in most situations as P2P communication is required infrequently. Furthermore, the base station is a single point of data traffic concentration and failure. In contrast, WiseP2P allows nodes to send more packets to their peers directly or through relay nodes, by claiming extra communication slots from the network.

6.3 WiseP2P Description

The WiseP2P algorithm consists of four phases: P2P group assignment, P2P multicast tree assignment, P2P virtual path activation, and P2P virtual path repair. Initially, in the *P2P group assignment* phase, the base station groups nodes that wish to communicate with each other. Afterwards, the base station performs *P2P multicast tree assignment* by building a spanning tree that connects all nodes in a group and assigns virtual connectivity links that can be used by a node to communicate with nodes in its group. The base station disseminates this information using the multi-function slot cycle before the start of a data gathering cycle. Nodes maintain this information throughout their lifetime in the network. This initial network setup is a one-off phase.

During data gathering cycles, a node can execute *P2P virtual path activation* to get extra time slots from the network, whenever it wishes to communicate with nodes in its group. There are two types of virtual path activations: local and central. In the local scheme, nodes can autonomously claim extra slots based on their local neighbourhood states. The central scheme enables the base station to allocate extra communication slots to requesting nodes. Nodes execute the *P2P virtual path repair* phase when they detect failures due to communication errors or topology changes.

Throughout this chapter, we will describe the WiseP2P algorithm using the following terms:

- *P2P-group* refers to a collection of sensor nodes that can communicate with each other. A P2P-group can be based on physical region or logical roles (e.g., all nodes with certain sensors).
- *Peer* refers to nodes that belong to the same P2P-group.
- *P2P-requestor* refers to an initiator node that requests for extra slots to communicate with its peers.
- *P2P-relay* refers to a participating node that acts as a relay for nodes in a P2P-group if any of these nodes is unable to reach its peers. P2P-relays provide connectivity among peers in a P2P-group or between P2P-groups.
- *P2P-router* refers to a node that forwards a P2P-requestor's packet towards the P2P-requestor's peers.

6.3.1 P2P Group Assignment

The human manager may wish to allow nodes with the same sensor type to form a P2P-group or allow nodes with different sensor types to collaborate together as a P2P group. For example, at the time of deployment, the human manager may define thresholds on sensor nodes that are used as event triggers and specify tasks to be executed when the events occur. In WiseP2P, initially, when a node enters the network, it advertises the types of sensor readings it provides (e.g., light, humidity, soil moisture, temperature) to the base station. WiseP2P allows nodes to piggyback this information during the FlexiTP time slot assignment phase. The base station then forms P2P-groups according to the application requirements. For example, as described in Section 6.1, the human

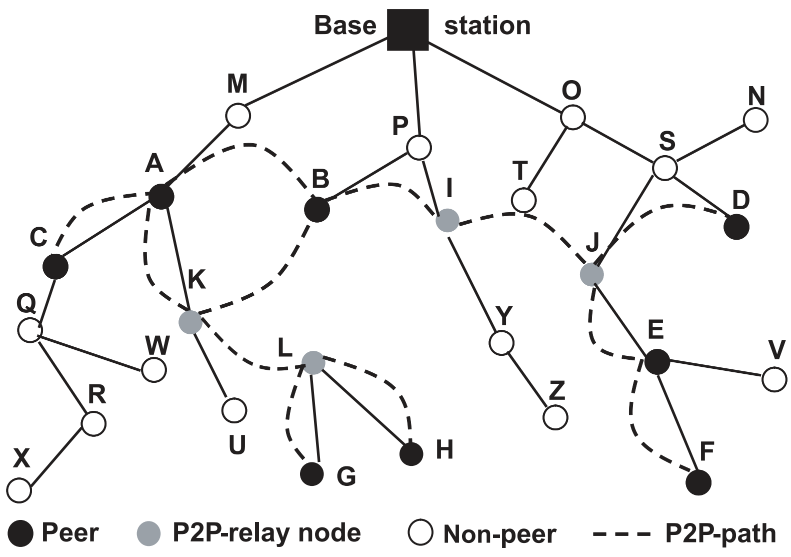

Figure 40: A network with one P2P-group consisting of eight peers highlighted in black.

manager organises eight nodes in the network into a physical P2P-group for smart-office applications, which are node *A*, *B*, *C*, *D*, *E*, *F*, and *G*. Figure 40 maps this sensor network into a data gathering tree structure.

6.3.2 Multicast Tree Assignment

The WiseP2P multicast tree assignment phase is responsible for creating a multicast tree for each P2P-group in the network. A multicast tree is used to enable bi-directional communication for all nodes in a P2P-group, i.e., every node in a P2P-group can communicate with all of its peers. The WiSeP2P algorithm assumes that the sensor network has no significant holes in the coverage by sensor nodes. Since sensor nodes serve as both gatherers and routers for the data, they can act as relays for others, and therefore offer multiple different paths. It is necessary to develop mechanisms for easily and rapidly identifying relay nodes that yield good alternative paths. WiseP2P includes an algorithm that pre-selects a P2P-relay, when initiating communication between a pair of peers in the network.

The WiseP2P multicast tree assignment phase consists of two steps. First, WiseP2P ensures that any peer can reach any other peer in a P2P-group. After the FlexiTP initial network setup, the base station knows the neighbour list of every node in the network. A neighbour of a sensor node is any node within its communication range, not only neighbours connected by the data gathering tree. The base station can determine whether or not nodes can reach their peers based on the list. If a node is unable to reach any of its peers, the base station executes Dijkstra's algorithm [25] to select non-peer nodes to act as P2P-relay that relays the node's data to the node's peers.

Second, WiseP2P executes Prim's spanning tree algorithm [26] in order to construct a P2P-group's multicast tree with minimum exposure path, based on the connectivity among P2P-relay nodes and peer-nodes in a P2P-group. The base station then stores the multicast tree's node-connectivity information (P2P-virtual-path) in the *global P2P-adjacency table*. After the base station determines the P2P virtual-path of all nodes in the network, it distributes this information to the nodes using FlexiTP's multi-function slot during the initial network setup. Each node in the network only records its reachable peers and P2P-relays in its *local P2P-adjacency table*. This approach is scalable because the size of a P2P-adjacency lookup table depends on a few neighbours, so it is feasible to support even very large networks.

Figure 40 illustrates that some of peers in the network are unable to reach some of their peers directly because of their constrained communication range. To solve this problem, communications between peers can take place through one or more P2P-relay nodes selected among non-peers. Figure 40 shows that nodes *I*, *J*, *K*, and *L* are chosen as P2P-relay nodes of the P2P group.

6.3.3 P2P Virtual Path Activation

Though the multicast tree assignment phase determines the next hop for each node in a P2P group, the group cannot be activated until each node knows its sending and receiving slot numbers. The task of virtual path activation phase is to inform each node in which slots it should send and receive packets for facilitating the P2P communication. In WiseP2P, when a sensor node wants to communicate with any of its peers, it becomes a

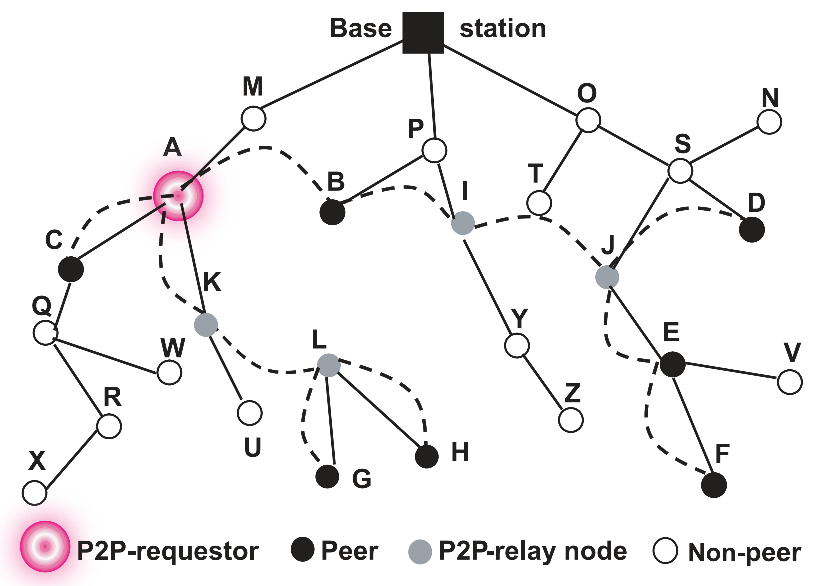

Figure 41: A P2P-multicast tree rooted at the P2P-requestor *A*.

P2P-requestor that requests extra slots to send its data or query to the destination peer(s). A P2P-requestor is the root of its P2P-group's multicast tree.

WiseP2P virtual path activation can be executed locally or centrally. In the local scheme, a P2P-requestor activates its virtual-paths by assigning slot(s) to its next-hop node(s) for P2P-communication, based on its local P2P-adjacency lookup table. The central scheme uses the base station to activate a P2P-requestor's virtual paths by allocating slots to the P2P-requestor, the P2P-requestor's peers, and the P2P-requestor's relay nodes, based on the global P2P-adjacency lookup table. Both schemes activate the same virtual-paths.

Figure 41 shows the WiseP2P multicast tree rooted at the P2P-requestor *A*. Recall that *A*'s peers are *B*, *C*, *D*, *E*, *F*, *G*, *H* and its P2P-relays are *I*, *J*, *K*, and *L*. Based on either local or global lookup table, the P2P-requestor *A* has six P2P-routers that forwards its packet to its peers and P2P relays. These P2P-routers are nodes *B*, *I*, *J*, *E*, *K*, and *L*. In the next sections, we will describe the slot allocation technique of the local and central virtual path activation schemes in details.

6.3.4 Local P2P Virtual Path Activation

In the local WiseP2P scheme, a P2P-requestor activates its P2P-virtual paths by initiating a message-based P2P-connection during FlexiTP's fault-tolerance slot, that is, the common CSMA-contention period. The P2P-requestor can specify the number of extra slots required in a data gathering cycle and specify how long these extra slots will be required. The main advantage of local WiseP2P function is that nodes can aggressively claim extra slots from the network based on their local slot schedules.

Slot Assignment

The local WiseP2P slot selection scheme allows a node to claim a slot if and only if the following criteria are satisfied:

1. the claimed slot is not listed in its receive slot list, transmit slot list, and conflict slot list, and
2. the claimed slot number is less than or equal to the global-highest slot.

If both criteria cannot be met, the local WiseP2P employs a wrap-around slot selection scheme, whereby nodes can select lower slots from the next data gathering cycle. Thus, nodes are not restricted to choose forward slot numbers that are lower than their receive slots if the data gathering cycle bound is reached. The overall slot selection scheme allows a slot to be reused by many nodes as long as nodes' transmissions are not within interference range of each other and hence the spatial slot re-use can be maximised. Furthermore, the scheme ensures that a node gets a conflict-free slot without extending the network's data gathering cycle.

In the FlexiTP's fault-tolerance slot, WiseP2P also executes the FlexiTP's neighbour schedule exchange. Every time a node claims a slot, it broadcasts its claimed slot to its direct neighbours and then each of these direct neighbours propagate the slot information again to their direct neighbours. In this way, a slot claimed by a node is informed to neighbours that are within twice the node's communication range. This approach prevents nodes that are potentially within the interference range of each other for claiming the same slot.

Example

Let node *A* in Figure 41 be the P2P-requestor. For simplicity, assume the P2P-requestor *A* requires one extra slot in a data gathering cycle. It first tries to claim one extra slot by checking its slot schedule. Suppose it can claim slot number *5* since slot *5* does not appear in its receive slot list, transmit slot list, or conflict slot list. It then lists slot *5* in its transmit slot list table. The P2P-requestor *A* retrieves the information about the intended receivers (either peers or P2P-relay nodes) of its P2P packet from its P2P-adjacency lookup table, which are node *B*, *C*, and *K*. Afterwards, *A* tries to broadcast slot *5* to the receivers through CSMA/CA contention in the fault-tolerance slot. Consequently, *B*, *C*, and *K* list slot *5* as a receive slot in their receive slot list tables. Since, *B* and *K* are *A*'s P2P-routers, they try to claim an extra slot for forwarding *A*'s P2P-packet by selecting a forward slot number that is bigger than slot number *5*. For example, suppose both *B* and *K* claim slot number *7* and list this slot in their transmit slot list tables. However, *B* manages to broadcast its claimed slot before *K*, to the next P2P-packet's receiver(s), that is, node *I*. Nodes within the communication range of *B*, including *K*, overhear this broadcast information and they list slot *7* in their conflict slot list tables. Furthermore, the P2P-router *K* removes slot *7* from its transmit slot list table and re-tries to claim another conflict-free slot.

In the case where a node is unable to claim a forward slot number that is higher than the receive slot and lower than the global-highest slot, it can try to claim the next available slot. For example, suppose node *L*'s receive slot is slot number *20* while the value of the global-highest slot is also *20*. Node *L* then tries to claim a lower conflict-free slot number, say slot number *3*. This wrap around slot-selection scheme is beneficial to improve nodes' aggressive attempt to get extra slots they require, without changing the length of data gathering cycles. A lower numbered slot is basically a slot from the next data gathering cycle. P2P-routers that are unable to broadcast their claimed slot in the current fault-tolerance slot, wait for the next fault-tolerance contention period. Note that any slot (say *s*) claimed by a node (say *i*) maybe used by some other node (say *j*) which is outside the transmission range of *i*, as *s* does not appear in the receive slot list, transmit slot list, or conflict slot list of *i*. Hence, there is no need to inform *j* that *i* has taken its slot.

If a P2P-requestor requests for *n* packets in a data gathering cycle, it can try to claim *n* slots at a time before broadcasting these slots to its P2P-routers. Note that nodes claim extra slots for a specified length of time, that is, in terms of the number of data gathering cycles. They then simply resume their previous schedule once this time expires. Recall that FlexiTP performs a time synchronisation in every data gathering cycle and so it ensures that clock drifts among nodes are minimised.

In the case where the FlexiTP's fault-tolerance slot is set to 0 ms, a P2P-requestor is unable to activate its P2P path locally. The request is then propagated up to the base station using data gathering slots in the next data gathering cycle. This is where the central WiseP2P virtual path activation comes into place.

6.3.5 Central P2P Virtual Path Activation

In contrast to local WiseP2P where nodes aggressively claim extra conflict-free slots from the network without knowing the owners of the slots, central WiseP2P utilises the base station to exclusively get extra slots from one or more nodes in the network. The base station analyses the priorities of all P2P-requestors' requests it receives and tries to allocate extra slots for requests with a higher priority first. The central WiseP2P function extends the on-demand TDMA slot transfer algorithm described in Chapter 5. It consists of four main steps: slot demand, slot-supplier selection, slot assignment, and slot delivery. The steps will be illustrated through the example network shown in Figure 41.

Slot Demand

The base station decides the extra slots (let this number be α) required by a P2P-requestor to communicate with its peers in a data gathering cycle based on priority status of a particular event reported by the P2P-requestor. When a P2P-requestor node asks for one extra slot in a data gathering cycle, the base station should also allocate one extra slot for each of the P2P-requestor's routers. Thus, the total number of slots required to meet the P2P-requestor's request is equal to α multiplied by the number of P2P-requestor's virtual paths. When a P2P-requestor requests for more than one extra slot in a data gathering cycle, the base station allows the P2P-requestor's routers to pipeline the P2P-requestor's packets by reusing the supplied (claimed) extra slots if the slots are not listed in their receive slot list, transmit slot list, or conflict slot list. This approach improves the network bandwidth utilisation.

For example, the P2P-requestor *A* in Figure 41 requires two extra slots in a data gathering cycle. The total number of extra slots required by *A* is equal to 14 slots: two slots each for *A* and its P2P-routers: *B*, *I*, *J*, *E*, *K*, and *L*. Recall that a transmit slot for one router coincides with the receive slot for the next router. The base station then needs to allocate seven slots for *A*'s first P2P-packet propagation and another seven slots for *A*'s second P2P-packet propagation.

Slot-supplier Selection

The base station determines which nodes will allow the temporary use of their data gathering schedule by P2P-requestor(s), called slot-suppliers. A node's data gathering schedule consists of slots that are used by the node and its routers to forward its packets to the base station in data gathering cycles. For example, Figure 41 shows that node *I* has two data gathering slots: one for itself and one for its router, *P*, to forward the data sent by *I*. Given a demand of *n* slots, the base station determines how many slot-suppliers are required to meet the demand.

The slot-supplier's ordering according to selection criteria are as follows:

1. *Unreserved slots*. The base station claims unreserved slots from existing slot-suppliers if the number of supplied slots is sufficient to meet the P2P-requestor's demand. For example, if node *X* has already given up three out of its five slots to a P2P-requestor, there are two unreserved slots that can be claimed by any P2P-requestor.

2. *Equilibrium*. The number of data gathering slots supplied by a slot-supplier is equal to the P2P requestor's demand.

3. *Surplus*. The number of data gathering slots supplied by a slot-supplier is higher than the P2P requestor's demand.

4. *Deficit*. The number of data gathering slots supplied by a slot-supplier is lower than the P2P requestor's demand. Thus, more slot-suppliers are required to meet the P2P requestor's demand (includes unreserved slots whenever possible).

Figure 41 shows that there is no single slot-supplier that can supply seven slots for *A*'s first P2P-packet propagation. In this *deficit* case, the base station selects a collection of slot-suppliers to meet the demand. For example, assume nodes *X* and *Y* are two of the nodes that can lend their data gathering schedule and hence they become *A*'s slot-suppliers.

Slot Assignment

The base station is responsible for ensuring that the supplied extra slots do not conflict with the schedule of P2P-requestors and their P2P-routers. This approach avoids schedule duplication and collisions by simply checking whether the supplied extra slots exist in the schedule-table (receive slot list, transmit slot list, and conflict slot list) of P2P-requestors and their P2P-routers. The base station performs this function by coupling node schedule information with node neighbourhood information and so the base station can construct the receive slot list, transmit slot list, and conflict slot list table of nodes in the network.

A node's data gathering schedule consists of slot numbers that are used by the node's routers (parent and ancestors) to route the node's packet to the base station. For example, in Figure 41, say the slot-supplier *X*'s data gathering schedule is slot numbers 2, 3, 4, 5, 6, and 7, whilst *Y*'s is 8, 9, and 10. The base station first filters and sorts extra slots provided by the slot-suppliers in an increasing order, and then executes slot checking to ensure conflict-free slot allocation. The base station checks whether slot number 2 is in *A*'s table. If not, the base station checks if slot number 3 is in *B*'s table. The same slot checking is performed at each of the P2P-requestor's routers. If the consecutive slot numbers 2 to 8 are conflict free slots, the base station then assigns these slots to *A* and its P2P-routers. Thus node *A*, *B*, *I*, *J*, *E*, *K*, and *L* receive slot number 2, 3, 4, 5, 6, 7, and 8 respectively. Now, slot 9 and 10 are recorded as unreserved slots.

Now, the base station needs to allocate extra slots for propagating P2P-requestor *A*'s second packet in a data gathering cycle. The base station tries to pipeline the second packet by reusing some available slots from the previously claimed extra slots for *A*. The base station then selects to reuse the extra slot number assigned to *J* because *J* is outside the interference range of *A*, that is, slot number 4. Consequently, the base station assigns consecutive slot numbers 4, 5, 6, 7, and 8 to *A*, *B*, *I*, *J*, and *E*, while *K* and *L* are assigned unreserved slot numbers 9 and 10 respectively. In this scenario, the total number of slot reused are five slots: 4, 5, 6, 7, and 8. Therefore, the percentage of slot reuse is 36% of the total slot demand. In addition, the total number of distinct slots claimed is nine: 2, 3, 4, 5, 6, 7, 8, 9, and 10.

Slot delivery

In the FlexiTP's multi-function slot, the base station propagates the extra slot numbers allocated to P2P-requestors and the information on how long these requesting nodes can use those extra slots, by piggybacking the slot allocation information on the multi-function packet. The P2P-allocation packet contains the following information:

- A list of P2P-slots and the timer for these slots (in terms of the number of data gathering cycles).

- A list of P2P-nodes that should update their schedules with the allocated P2P-slots.
- A list of slot-suppliers that should shut down their data gathering schedules and the timer for the temporarily released schedules.

A node that receives the P2P-allocation packet checks whether its ID is in the list of P2P-nodes or slot-suppliers. If a node is in the list of P2P-nodes, the node updates its table with the allocated extra slot and sets the timer for the extra slot. Whilst, if a node is on the list of slot-suppliers, the node shuts down the corresponding released slot (i.e., the data gathering slot that belongs to the node), and also sets a timer for the temporarily released slots.

After the timer for borrowing or releasing extra slots expires, the P2P-requestors and their routers remove extra slots from their tables, while the slot-suppliers resume their previous schedule. Similar to the central OST algorithm described in Section 5.2.2, the base station can propagate timer extension information during the multi-function slot cycle in order to extend the timer before it expires.

There are two main advantages of the central WiseP2P function. First, it allows nodes to reconfigure their traffic patterns to support P2P communication next to their regular data gathering tasks within one data gathering cycle. Second, it allows nodes to adjust their schedules fairly easily and in an energy-efficient manner.

6.3.6 P2P Virtual Path Repair

In dynamic energy-constrained sensor networks, sensor nodes and communication links are prone to errors, and nodes can join or leave the network at any time. The main challenge in implementing P2P operations is to preserve the ability of a node to communicate with its peers in the network. WiseP2P leverages and extends FlexiTP's fault tolerance mechanisms to provide P2P slot allocation failure avoidance, to repair P2P communication failures, to allow new nodes to be incorporated into appropriate existing P2P-multicast trees, and to handle gracefully the departure of nodes in the trees.

Dynamic Changes in Link Quality

The error-prone nature of wireless links may cause packet loss due to congestion or interference, which can degrade the performance of sensor network applications. Unlike CSMA's probabilistic channel access method, WiseP2P leverages the scheduled-communication approach to guarantee transmission reliability.

Although the TDMA-based P2P communication scheme of WiseP2P prevents collisions, packets may be dropped as a result of external environmental factors. For example, a packet that carries P2P slot allocation information may be corrupted or dropped due to lossy communication links. WiseP2P improves the reliability of packet delivery through P2P packet redundancy. WiseP2P sends multiple P2P slot allocation packets over a number of data gathering cycles (can be adjusted according to the application's characteristics) to accommodate potential P2P communication failures.

Topology Changes

In WiseP2P, each node in a network maintains a P2P-adjacency redundancy table that stores a list of alternative P2P routing paths. When a node n fails, other nodes whose P2P-adjacency tables include n as their P2P-router must find n's successor or replacement. In addition, the failure of n must not be allowed to disrupt the current progress of P2P virtual activation as the system is re-stabilising. The key step in WiseP2P failure recovery is maintaining correct successor pointers, since in the worst case a predecessor can make progress using only successors. To help achieve this, each node maintains a *successor-list*.

A node queries its P2P-routers using a RTS/CTS handshake mechanism in the fault-tolerance slot at regular intervals. If the node notices that any of its P2P-routers has

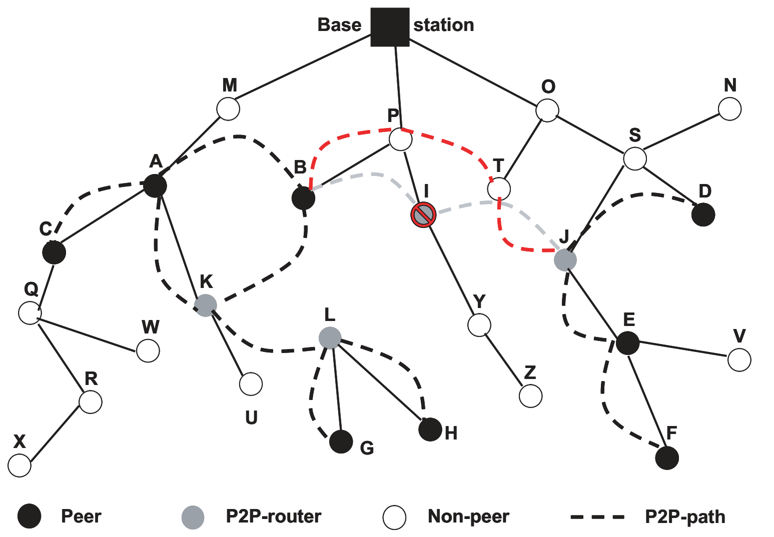

Figure 42: A P2P virtual path repair example. When the P2P-relay *I* is dead, P2P-relays *P*, *T*, and *J* are selected to preserve P2P virtual path between peer *A* and peer *E*.

failed, that is when the node receives no reply from its P2P-router, it replaces the P2P-router with the first live entry in its *successor-list* that can be found in its P2P-adjacency redundancy table. Figure 42 shows that the original P2P virtual path from node *A* to *E* is: *A* - *B* - *I* - *J* - *E*. When P2P-relay *I* fails, node *B* selects node *P* as *I*'s successor. The selected P2P-relay *P* accordingly uses node *T* and *T* uses node *J* to reach *E*. Thus, node *A*'s virtual P2P path to peer *E* is updated to: *A* - *B* - *P* - *T* - *J* - *E*.

WiseP2P extends the local repair scheme of FlexiTP to accommodate topology changes due to node addition. When a new node is connected to the data-gathering tree of the network, the base station will re-execute three WiseP2P phases: P2P group assignment, P2P multicast tree assignment, and P2P virtual path assignment. Afterwards, the base station sends the information about virtual-path changes to relevant nodes in the network during the multi-function slot cycle and these nodes update their local P2P-adjacency tables.

Table 6: Simulation parameters for evaluating the performance of protocols that support P2P communication: WiseP2P and the CSMA-based protocol.

Simulation parameters	Description
FlexiTP-WiseP2P	
`Data gathering slot size`	0.002882 ms
`Multi-function slot size`	0.002882 ms
`Fault-tolerance slot size`	0 ms
`Routing protocol`	Forwarding-to-parent
CSMA	
`Routing protocol`	AODV

6.4 Performance Evaluation

In this section, we simulate WiseP2P using the NS-2 simulator. The code can be found at [70]. We also provide evaluations of WiseP2P against a CSMA-based protocol, addressing their performance and effectiveness in satisfying various P2P communication demands across different network scenarios and configurations.

6.4.1 Simulation Setup

Table 6 presents simulation parameters of WiseP2P and CSMA. The simulated CSMA-based protocol uses a RTS/CTS handshake mechanism and ACK for reliability. Recall that the FlexiTP's fault-tolerance slot is basically a CSMA period in each data gathering cycle. It is used for local repair and is utilised for allowing P2P-requestors to claim extra slots aggressively based on their local lookup tables: slot schedule and P2P-adjacency table. Since the simulated networks do not experience topology changes, this fault-tolerance slot will be wasted if nodes have no slot requests. We set the value of the fault-tolerance slot to be 0 ms. Although this setup will disable nodes from utilising the local P2P-path-activation scheme (described in Section 6.3.4), it allows FlexiTP-WiseP2P to be comparable with CSMA simulation setup.

All reported results are averaged over 15 different network topologies. The transmission range in these simulations is 60 m. Our simulations contain 100 to 400 sensor nodes placed randomly in a network area of 300 m x 300 m. All simulations run for 200 seconds. Each node in a network has data to send to the base station every two seconds. The packet size is 56 bytes (30 bytes for payload and 20 bytes for packet header). In all simulations, the channel bandwidth is 2Mbps.

In all simulations, all networks of 100, 200, 300, and 400 nodes have five P2P-groups with each P2P-group of 20, 40, 60, and 80 nodes respectively. A P2P-group consists of nodes randomly selected from the network. In our simulations, we tested the most extreme scenario in which every P2P-requestor wishes to communicate with all peers in its P2P-group. Suppose a node in a network density of 400 nodes wishes to communicate with its peers, this means the node requires extra slots to communicate with its 79 peers.

We conducted two types of experiments to explore the behaviours of WiseP2P and CSMA under worst case scenarios. In all experiments, upon the occurrence of externally triggered events, nodes become P2P-requestors. Note that WiseP2P allows any type of distributed-cooperative function implementation and so it is not restricted only to event detection applications.

In the *first experiment*, we gradually generated sensor data with values exceeding the user threshold in some nodes in a network of 100 nodes. We started by introducing event-triggered sensor data to one node and then gradually introduced it to up to five nodes within the same P2P-group. For this experimental setup, we simulated data gathering sensor networks with uniform and non-uniform inter-packet arrival rate for 200 seconds. The uniform traffic setup requires all nodes, including P2P-requestors, to perform data gathering with inter-packet arrival of two seconds. This means that each P2P-requestor in the WiseP2P network sends one packet to its peers in a data gathering cycle. This uniform traffic setup is denoted by *WiseP2P-1* and *CSMA-1*. In the non-uniform traffic setup, in addition to a periodic data collection every two seconds, P2P-requestors increase their offered loads with inter-packet arrival of one-second. In this way, WiseP2P's P2P-requestors ask for two extra slots in a data gathering cycle, whilst CSMA's P2P-requestors have data to send every one second. The non-uniform traffic setup is denoted by *WiseP2P-2* and *CSMA-2*.

In the *second experiment*, we introduced the event-triggered sensor data to a node in a

P2P-group and then gradually introduced it to up to five nodes from different P2P-groups, in a network of 100 to 400 nodes. In this experimental setup, all nodes, including P2P-requestors, have an uniform inter-packet-arrival rate of two seconds.

6.4.2 Network Performance

P2P Slot Allocation

We evaluated the capability of WiseP2P in supporting P2P communication by measuring the percentage of demand satisfied using WiseP2P for the first and the second experimental setups. Under the given traffic loads and P2P slot demand in both experiments, simulation results show that WiseP2P can fully satisfy the P2P slot demand in different network scenarios and configurations. In high demand cases where P2P-requestors send their packets to their peers twice in a data gathering cycle, WiseP2P sustains 100% demand allocations because it utilises the pipelined slot allocation technique whenever the slot demand is higher than supply. The simulation results also reflect that WiseP2P ensures fairness for P2P-requestors, in that all P2P-requestors can share the channel equally under reasonable network scenarios.

Throughput

We measured network throughput, that is, the ratio of the total number of packets successfully transferred from a sender to a receiver, to the total number of packets transmitted by the senders in the network, in 200 seconds. This metric calculates the ratio of the sum of total of packets received at the base station and at intended peers, to the total packet transmissions.

Figure 43 presents the average network throughput using WiseP2P versus using CSMA, where P2P-requestors are from the same P2P-group. Simulation results show that WiseP2P outperforms CSMA across diverse demands. The network throughput of both protocols remains almost the same independent of the number of P2P-requestors.

Figure 44 shows the average network throughput of networks with five P2P-requestors where each of these P2P-requestors is from a distinct P2P-group. As the network density increases, WiseP2P's throughput slightly decreases, while CSMA's throughput decreases significantly. WiseP2P still achieves more than 99% network throughput due to the efficiency of FlexiTP and WiseP2P's TDMA collision avoidance scheme.

The reliability of WiseP2P and CSMA in supporting P2P communication can be analysed by observing the actual throughput at a P2P-requestor's peers. The P2P-requestor throughput is the ratio of the number of packets successfully transferred from a P2P-requestor to its peers, to the number of P2P packet transmissions, in 200 seconds.

Figure 45 shows the average P2P-throughput given each P2P-requestor in the network belongs to the same P2P-group. It shows that WiseP2P outperforms CSMA in the average of P2P-requestor throughput. WiseP2P achieves at least 98% throughput, while CSMA's throughput is lower than 36% across various scenarios.

Figure 46 shows the throughput of P2P-requestors when each of them is from distinct P2P-group. As the network density increases, WiseP2P sustains 100% throughput, while CSMA only manages to successfully deliver around 15% to 20% of packet transmissions.

In all simulation scenarios, WiseP2P provides much higher throughput than CSMA because it utilises the scheduled P2P-communication scheme that guarantees collision-free P2P traffic. On the other hand, CSMA floods and uses contention-based medium access for node communication that is susceptible to contention for bandwidth and hidden terminals. Simulation results prove that a TDMA-based protocol better for P2P communication than a CSMA scheme, especially in heavily-loaded networks.

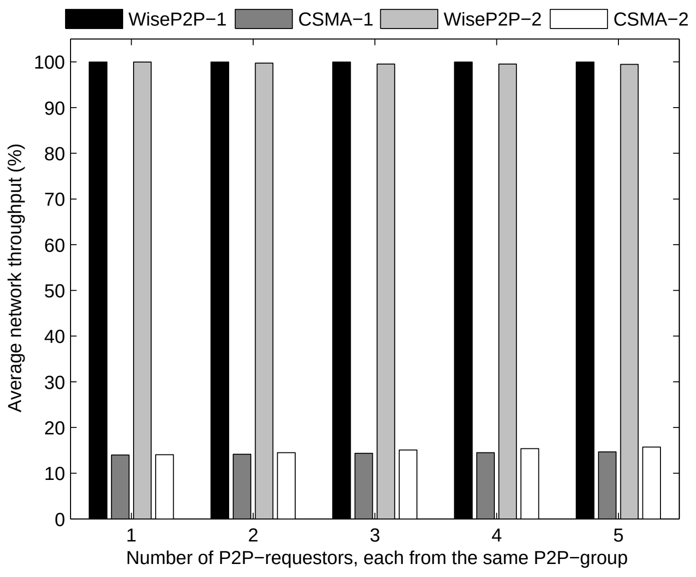

Figure 43: Percentage of network throughput versus total P2P-requestors in the network of 100 nodes. Each P2P-requestor is from the same P2P-group.

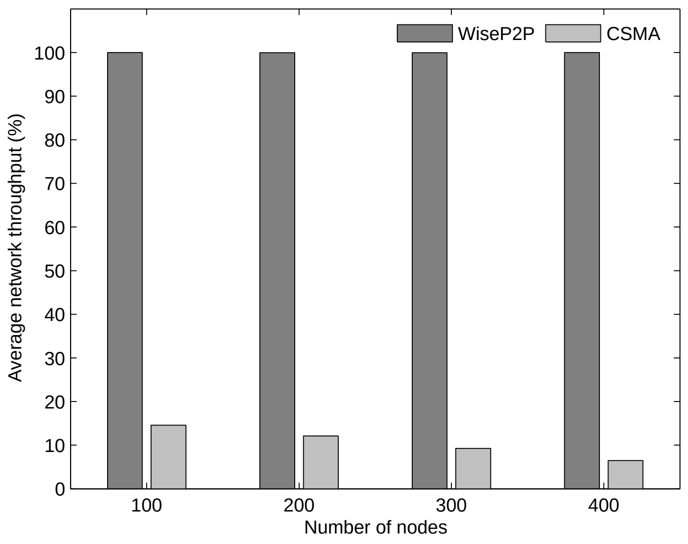

Figure 44: Percentage of network throughput versus network density, with P2P-requestors from distinct P2P-groups.

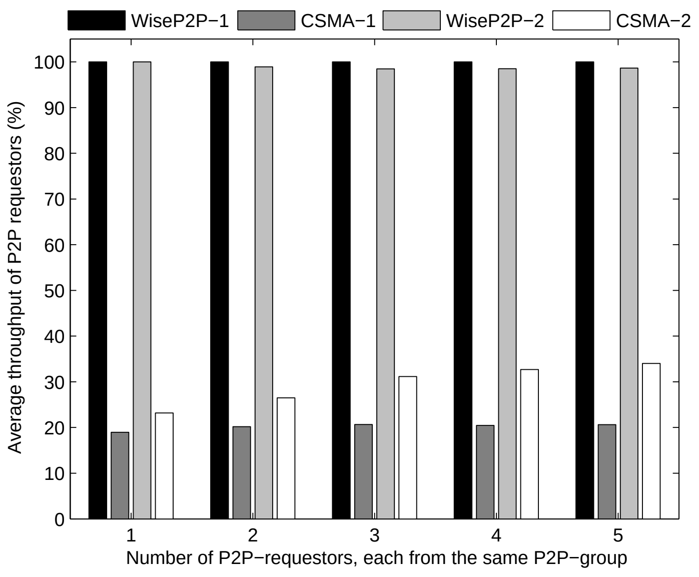

Figure 45: Percentage of P2P-requestor throughput versus total P2P-requestors in the network of 100 nodes. Each P2P-requestor is from the same P2P-group.

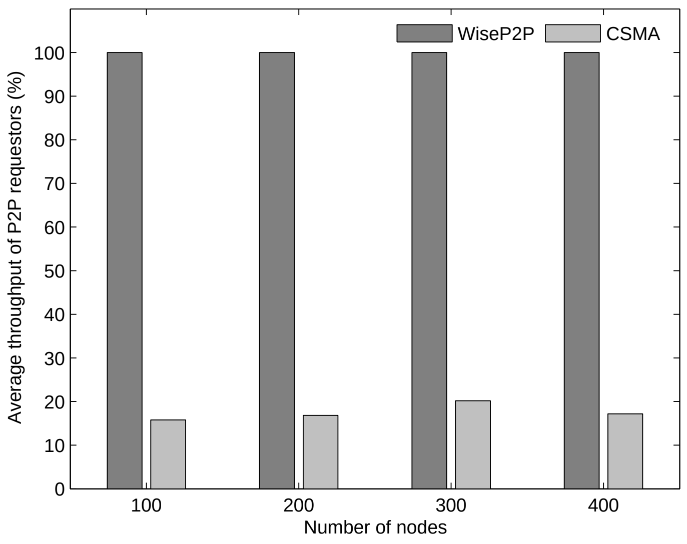

Figure 46: Percentage of P2P-requestor throughput versus network density, with P2P-requestors from distinct P2P-groups.

Latency

We evaluated packet latency by calculating the delay from when a P2P-requestor has a packet to send until the packet is successfully received by the intended peer. In Figure 47, the packet latency is plotted versus the number of P2P-requestors, for WiseP2P and CSMA. P2P-requestors are from the same P2P-group. The reported results show that WiseP2P enables P2P-requestors to transfer their packets successfully to their peers about three times faster than CSMA when these P2P-requestors only have one P2P packet to send per interval. As the number of P2P-requestor's packet transmissions increases, the packet latency of both protocols increases, however, WiseP2P's packet latency is still much lower than CSMA.

Figure 48 shows that as the network density increases, the average packet latency of both WiseP2P and CSMA increases. P2P-requestors are from distinct P2P-groups. In WiseP2P, the increase in packet latency is due to the increase in the length of the network's data gathering cycle. However, the packet latency is still bounded and lower than CSMA. CSMA experiences high latency because of its RTS/CTS mechanism and its non-deterministic channel access scheme. In contrast, WiseP2P allows nodes to perform a P2P-communication by executing their scheduled tables. This TDMA approach results in low packet latency and it reduces collisions.

In wireless sensor networks, the significance of packet latency depends on the application's requirements. In critical sensor applications such as seismic monitoring on a bridge, it is crucial to allow relevant nodes to be able to communicate with each other rapidly upon detection of an event of interest and so they can perform a distributed task accordingly. In such a scenario, the TDMA-based P2P communication of WiseP2P is proved beneficial over the CSMA scheme as WiseP2P provides assurance of predictable latency.

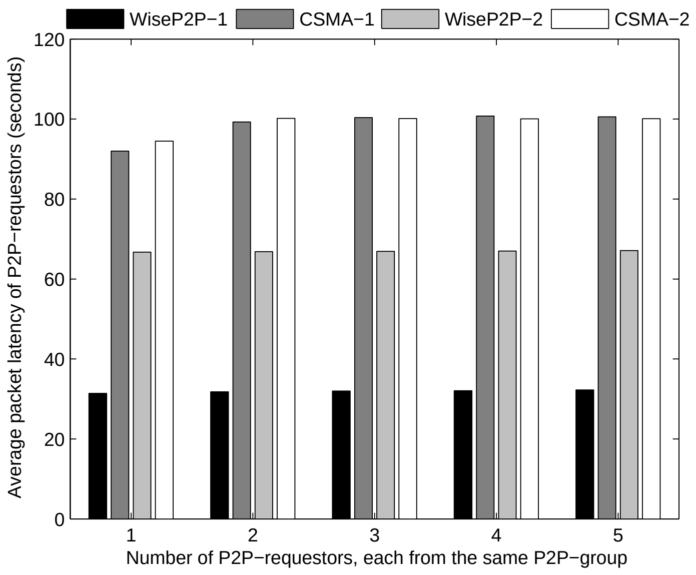

Figure 47: P2P-requestor packet latency versus total P2P-requestors in the network of 100 nodes. Each P2P-requestor is from the same P2P-group.

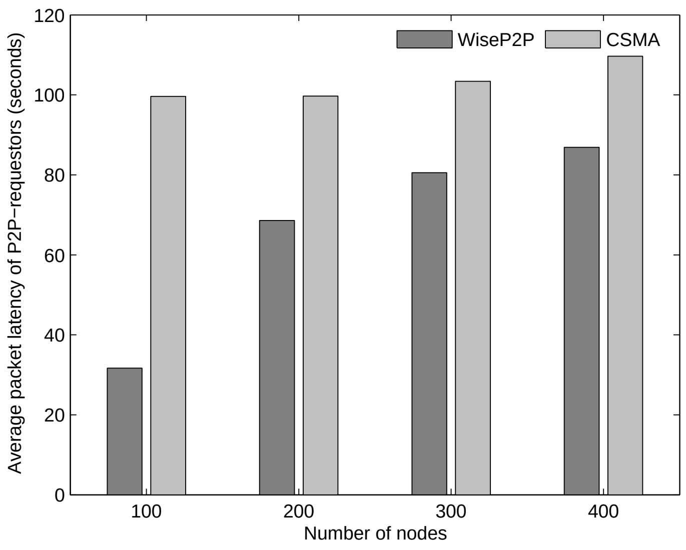

Figure 48: P2P-requestor packet latency versus network density, with P2P-requestors from distinct P2P-groups.

6.4.3 Energy Efficiency

Energy efficiency is one of the most important parameters for sensor-networks. Energy savings significantly affect the overall node lifetime. We evaluated the energy efficiency of WiseP2P and CSMA by observing the energy consumption per node and the energy consumption per packet over 200 seconds simulations.

The energy consumption per node metric measures the overall energy consumed by nodes, averaged over all nodes in the entire network. Figure 49 and Figure 50 show that WiseP2P requires much lower overhead for both data gathering tasks and P2P communication compared to CSMA. As shown in Figure 49, WiseP2P's energy consumption per node is around 24 times more efficient than CSMA under varying demands. Furthermore, Figure 50 shows that as the network density increases, WiseP2P still maintains a very low energy overhead, while CSMA's energy overhead per node increases proportionally. FlexiTP-WiseP2P achieves energy savings by controlling the radio to avoid energy waste from idle-listening and allowing nodes to switch to the low-power sleep mode when they are not scheduled to send or receive. In CSMA, nodes stay active regardless of their participation status in an activity. In such cases, idle listening is a dominant factor of radio energy consumption.

The energy consumption per packet metric measures the ratio of the total energy dissipated by nodes in the network to the total number of packets successfully received at the intended receivers (sinks). Compared to CSMA, Figure 51 and Table 7 show that WiseP2P maintains very low energy per packet under varying number of P2P-requestors and network densities. This is because WiseP2P uses TDMA time slots for both data gathering tasks and P2P communications and so it eliminates energy wastage due to collisions, overhearing, idle listening, and over-emitting. In contrast, CSMA performs much worse because it suffers contention-related collisions, especially in heavily-loaded sensor networks. Recall that in our simulations, several nodes in the network have data to send to their peers in addition to their normal (periodic) data gathering tasks.

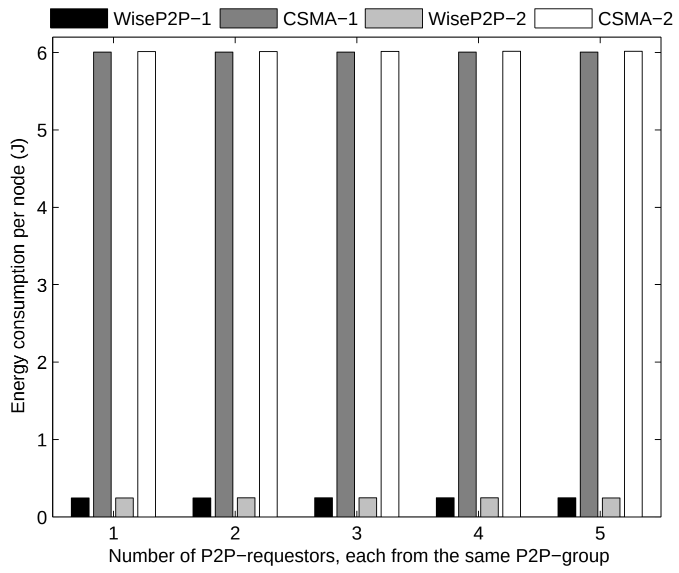

Figure 49: Energy consumption per node versus total P2P-requestors in the network of 100 nodes. Each P2P-requestor is from the same P2P-group.

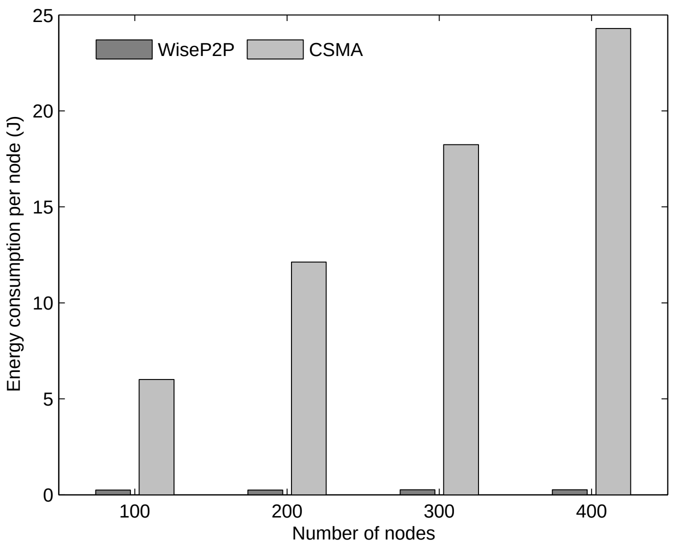

Figure 50: Energy consumption per node versus network density, with P2P-requestors from distinct P2P-groups.

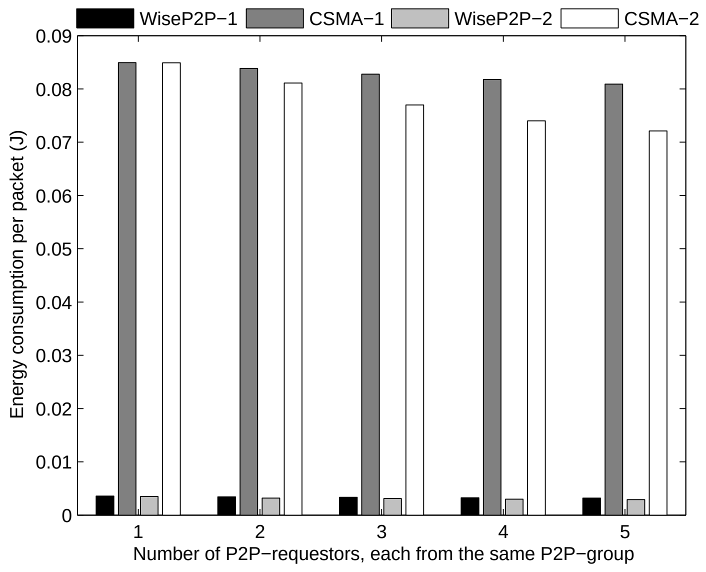

Figure 51: Energy consumption per packet versus total P2P-requestors in the network of 100 nodes. Each P2P-requestor is from the same P2P-group.

Table 7: Energy consumption per packet versus network density, with P2P-requestors from distinct P2P-groups.

Energy consumption per packet (J)	100 nodes	200 nodes	300 nodes	400 nodes
`FlexiTP-WiseP2P`	0.0028	0.0024	0.0022	0.0022
`CSMA`	0.0815	0.1151	0.2182	0.6504

Simulation results prove that WiseP2P's scheduled-P2P-communication approach provides the best utilisation of limited energy resource of sensor nodes, leading to a longer operational lifetime.

6.5 Conclusion

WiseP2P is a novel flexible-schedule-based TDMA protocol that supports peer-to-peer communication between nodes in a data gathering network along with the network's standard routing tree for periodic data gathering. WiseP2P allows sensor nodes to claim extra time slots to communicate with any other node in the network without going through the base station. It provides adaptive and reactive sensing in different parts of a network. This P2P-communication framework facilitates any implementation of distributed-function that requires sensor nodes to work cooperatively based on their local environment. For example, event-detection applications that require sensor nodes to detect events of interest and to adapt themselves autonomously in the modified state of the physical environment.

Time synchronisation is critical for the whole scheme of WiseP2P to work because nodes that are involved in a scheduled communication must wake up at the same time to exchange information. WiseP2P uses FlexiTP to perform time synchronisation locally, with each parent synchronising its children during a designated multi-function slot.

The reported simulation results show that WiseP2P's scheduled peer-to-peer communication scheme performs better than the CSMA-based scheme in terms of throughput, latency, and energy efficiency, especially in heavily-loaded networks. WiseP2P achieves energy efficiency and guarantees reliable data delivery through collision-free traffic, reduced hash table sizes, and low-overhead slot allocation schemes.

Chapter 7

Network Management in Wireless Sensor Networks

Management by objective works - if you know the objectives. Ninety percent of the time you don't . . .

Peter F. Drucker

In this chapter, we present a policy-based, TDMA-driven protocol for managing wireless sensor networks in order to optimise the operational and functional properties of sensor networks. We also present a TDMA-slot-blocking scheme that enables nodes in the network to perform an on-demand aggregation based on network requirements and network capacity constraints in addition to human managers' queries.

7.1 Introduction

Network management is the process of managing, monitoring, and controlling the behaviour of a network. Wireless sensor networks have different management objectives from traditional networks. In traditional networks, the primary goals are minimising response time and providing comprehensive information, but in sensor networks the primary goal is minimising energy use and the main means for doing this is by reducing the amount of communication between nodes [119]. Optimising the operational and functional properties of wireless sensor networks may require a unique solution for each application problem [16]. Wireless sensor networks are highly dynamic and prone to faults, mainly because of energy shortages, connectivity interruptions, and environmental obstacles. Network failures are common events rather than exceptional ones [32]. Thus, in wireless sensor networks, we are mainly concerned with monitoring and controlling node communication in order to optimise the efficiency of the network, ensure the network operates properly, maintain the performance of the network, and control large numbers of nodes without human intervention.

A network management system designed for wireless sensor networks should provide a set of management functions that integrate configuration, operation, administration, security, and maintenance of all elements and services of a sensor network. We focus on applications that provide management schemes in terms of monitoring and controlling sensor networks. Security management is beyond the scope of this thesis.

The main task of sensor network monitoring is to collect information about the following parameters: nodes status (e.g., battery level and communication power), network topology, wireless bandwidth, link state, and the coverage and exposure bounds of the sensor network. A sensor network management system can perform a variety of management control tasks based on the collected information. It can control sampling frequency, switch nodes on/off (power management), control wireless bandwidth usage (traffic management), and perform network reconfiguration in order to recover from node and communication faults (fault management).

Monitoring individual nodes in a large sensor network may be impractical. It is sufficient to control the network by ensuring specific network coverage. Managing wireless sensor networks presents new challenges due to the need for a distributed management mechanism that can efficiently adapt to the dynamic nature of these networks. Despite the importance of sensor network management, there is no existing generalised solution for sensor network management [96].

Developing an energy-efficient and reliable communication strategy for wireless sensor networks requires effective cross-layer collaboration of all layers. This cross-layer interaction approach poses challenges in the areas of data management, network management, and power management. We propose a *wireless sensor network management system* (WinMS), a policy-based-management approach that provides flexibility and adaptability in the face of the unique operational characteristics of wireless sensor networks, in an energy-efficient manner. WinMS does this by utilising FlexiTP to continually observe network states. It utilises the on-demand TDMA slot transfer algorithm to provide necessary management functions to maintain the performance of the network and achieve effective networked node operations. The WinMS framework consists of a local and a central management scheme that allow a network to adapt to changing network conditions such as topology changes, fluctuations in network behavior, and event detections, by reconfiguring itself based on current events as well as predicting future events.

The limited energy supply in sensor nodes is a major issue for all protocols designed for wireless sensor networks. In-network aggregation is a mechanism to minimise energy consumption by allowing nodes to aggregate data instead of propagating raw sensor readings. It reduces the number of transmissions in the network and reduces wireless bandwidth usage [41, 48, 56, 63, 82]. In this chapter, we also propose a TDMA-slot blockage mechanism through an *on-demand aggregation* (ODA) algorithm to allow sensor nodes to reserve some of their slots according to a desired data aggregation compression ratio. This approach facilitates sensor nodes to perform in-network aggregation and so to conserve energy based on requirements and capacity constraints of a network.

The remainder of this chapter is organised as follows. In Section 7.2, we survey network management protocols and systems designed for wireless sensor networks. Section 7.3 describes architecture, design principles, and functional operations of WinMS. It also elaborates local and central network management solutions of WinMS. Section 7.4 presents our proposed on-demand aggregation algorithm. In Section 7.5, we analyse the performance of WinMS and show how WinMS's local and central management solutions enable a network to adapt to changing network conditions and to balance sensing requirements, network communication constraints, and energy efficiency. Finally, we conclude the chapter in Section 7.6.

7.2 Related Work

The function of network management systems is to monitor and control a network. These activities are wide ranging from traditional network monitoring to application specific management goals.

Systems for sensor networks that are based on *traditional network management systems* include BOSS [107] and MANNA [96]. BOSS serves as a mediator between UpnP networks and sensor nodes. MANNA provides a general framework for policy-based management of sensor networks. sNMP [31] provides network topology extraction algorithms for retrieving network state.

Other researchers have designed novel *routing protocols* for network management. For example, TopDisc [32] and STREAM [33] are used in sNMP for extracting network topology, RRP [75] uses a zone-flooding protocol, Sympathy [90] uses Tiny-Diffusion protocol (a routing algorithm based on directed diffusion), and SNMS [109] introduces the Drip protocol. Our proposed system, WinMS is based on the FlexiTP protocol. In SNMS, Sympathy, and sNMP, the network state update is performed infrequently to conserve sensor node resources. Furthermore, Sympathy's flooding approach means that the central manager may have imprecise knowledge of global network states and so may make inaccurate analyses. In contrast, WinMS efficiently performs a regular network state update by allowing sensor nodes to piggyback management information on their scheduled data packets. Since sensor nodes have scheduled communications, they all have fair access to the network and so they can deliver their data to the central manager without contending for the medium. This guarantees collision-free traffic and predictable latency.

Fault detection is an important focus of the systems TP [53], Sympathy [90], MANNA [97], and WinMS. In TP, each node monitors its own health and its neighbours' health, so providing local fault detection. Sympathy goes one step further by providing a debugging technique to detect and localise faults that may occur from interactions between multiple nodes. MANNA performs centralised fault detection based on analysis of gathered wireless sensor network data. In WinMS, there is a scheduled period where nodes listen to their environment activities and can self-configure themselves in the event of failure without prior knowledge of the full network topology. In addition, WinMS provides a centralised fault management scheme that analyses network states to detect and to predict potential failures and takes corrective and preventive actions accordingly. TP, Sympathy, and MANNA focus solely on fault detection and debugging, they provide no automatic network reconfiguration to allow the network to recover from faults and failures. WinMS differs from the rest as it adaptively adjusts the network by providing local and central recovery mechanisms.

TinyDB [77] and MOTE-VIEW [112] are *visualisation tools* that provide graphical representations of network states to the end user. TinyDB is a query-based interface that allows the end user to retrieve information from sensor nodes in a network. MOTE-VIEW also allows the end user to control sensor node settings such as transmission power, radio frequency, and sampling frequency. In these systems the central server analyses data collected from the network. The main disadvantage of such passive monitoring schemes is that they are not adaptive to current network conditions, and provide no self-configuration in the event of faults. Furthermore, the end user must manually manage the network and interpret the graphical representation of collected data. In WinMS, monitoring sensor nodes are autonomous and do need attention from the end user. WinMS's central management scheme uses the base station, with its global view of the network, to continually analyse network states and to execute corrective and preventive management actions according to management policies predefined by the end user. This scheme allows self-configuration without (or limited) human intervention.

Energy is one of the most important resources to be managed in a sensor network because sensor nodes are mostly battery powered and in many cases it is impractical to recharge these batteries [20]. Several network management systems that focus on *managing power resources* in sensor networks in order to achieve energy-efficiency include Agent-Based Power Management [113], SenOS [60], AppSleep [91], and Node-Energy Level Management [17]. Agent-Based Power Management utilises intelligent mobile agents to manage sub-networks and perform local power management processing. It can reduce the sampling rate of nodes with critical battery and reduce node transmission power. Other systems such as SenOS, AppSleep, and Node-Energy Level Management [17] use common sensor nodes to perform power management. SenOS and AppSleep put nodes to sleep when they are not needed. Node-Energy Level Management [17] allows nodes to reject a management task based on the importance of that task. There are also other protocols that incorporate methods for reducing energy consumption. SPAN [22] and STEM [100] allow nodes to sleep if they are not involved in a forwarding task. LEACH [50] uses an in-network data aggregation method to filter redundant data in a dense sensor network and so reduce the number of transmissions. WinMS utilises the distributed TDMA-scheme of FlexiTP to provide energy-efficient management, data transport, and local repair. WinMS also further conserves energy by distributing loads among sensor nodes based on network conditions (e.g., traffic and node residual energy). For example, if data reported by nodes in certain parts of the network is stable over a period of time, WinMS may decide to shut down that sub-network completely to conserve energy. Alternatively, these nodes may perform data-aggregation for their own sub-networks and send the aggregated data to the base station.

Traffic management functions are provided in Siphon [117], DSN RM [126] and WinMS. Siphon [117] uses multi-radio nodes to redirect traffic from common-nodes in a network in order to prevent congestion at the central server and in the primary radio network. In contrast, DSN RM [126] uses single-radio common-nodes to evaluate each of their incoming and outgoing links and apply delay schemes to these links when necessary in order to reduce the amount of traffic in the network. Congestion can also be avoided by modifying sensor nodes' reporting rate based on the reliability level of sensor node data [85]. By reducing a node's reporting rate, the number of packets transmitted in the network is reduced, so avoiding potential congestion. WinMS provides superior traffic management functions because it utilises the on-demand TDMA slot transfer algorithm to reconfigure nodes to report their data more rapidly or slowly depending on the significance and importance of their data to the end-user.

Several network management protocols are *application specific* rather than general purpose schemes. RRP [75] is tailored for real-time data gathering applications with bursty and bulky traffic. AppSleep [91] is designed for latency-tolerant applications. TP [53] is a fault detection system best suited for monitoring and surveillance applications. SenOS [60] power management is designed for SenOS operating system-based sensor networks. Network management systems such as RRP [75] and Siphon [117] require special hardware: RRP uses GPS nodes for implementing the proposed zone flooding protocol, while Siphon requires multi-radio nodes to act as virtual sinks. WinMS is designed for periodic data gathering sensor networks. However, it also allows a network to adapt itself dynamically to changing traffic patterns during the application run time.

Existing sensor network management protocols such as sNMP [31], Sectoral Sweeper [39], RRP [75], AppSleep [91], SNMS [109], and MOTE-VIEW [112], mainly provide *passive monitoring* schemes that maintain network states. WinMS differs from these protocols as it proactively maintains the performance of the network by adjusting the network adaptively in changing traffic patterns, detecting and correcting current network faults (recovery mechanism), and preventing future network failures.

The sensor network management systems and protocols reviewed in this chapter have been designed from many different management perspectives: network-health monitoring, fault-detection, traffic management, congestion avoidance, power management, and resource management. None of the reviewed systems provides a fully integrated view of all sensor network management functions. We attempt to address some of these issues through WinMS.

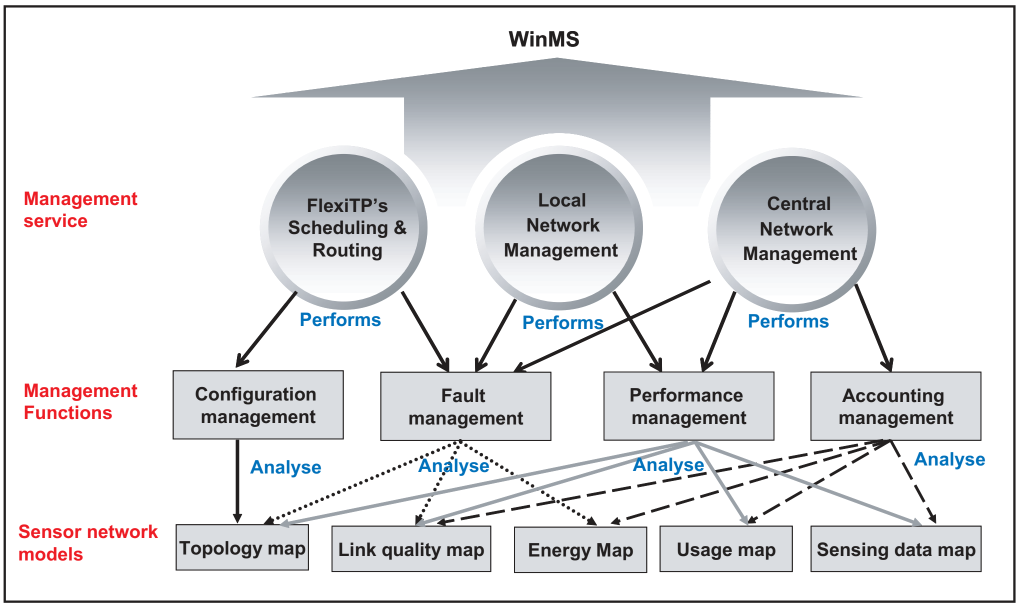

Figure 52: WinMS architecture: the relationship among management services, management functionalities, and wireless sensor network models.

7.3 WinMS Description

WinMS is an adaptive policy-based management system that allows human managers to predefine management parameter thresholds on sensor nodes that are used as event triggers, and to specify management tasks to be executed when any of these events occur. Throughout this chapter, a sensor node that meets a network policy is referred as a *self-reporting node*.

Figure 52 shows the WinMS architecture. FlexiTP is the underlying MAC and routing protocol that schedules node communication and collects and disseminates data continuously and efficiently, to and from sensor nodes in a data gathering tree. A local network management scheme provides autonomy to individual sensor nodes to perform management functions according to their neighbourhood network states, such as topology changes and event detection. The central network management scheme uses the central manager (base station), with global knowledge of the network, to analyse sensor network models and to execute corrective and preventive management maintenance accordingly.

7.3.1 Sensor Network Models

Managing sensor networks involves various network administration activities to depict the network state. The central manager maintains the following sensor network models:

- *Sensing data map* maintains sensor readings reported by nodes in the network.
- *Topology map* depicts the connectivity among nodes in the network.
- *Link quality map* gives information about the number of packets retransmitted, dropped, corrupted, and lost.
- *Usage map* gives information about network throughput based on the number of expected packets received by the base station, i.e., the ratio of the number of packets received over the total number of packet transmissions.
- *Energy map* gives the battery levels of nodes in the network.

The central manager analyses the correlation among these sensor network models to detect interesting events, such as areas of weak network health, possible network partition, noisy areas, and areas of rapid data changes.

7.3.2 Management Functionalities

WinMS provides a set of management functions that integrate configuration, performance, accounting, and fault management of all management entities (the base station and sensor nodes) of a network, while taking into account unique characteristics of wireless sensor networks. WinMS utilises FlexiTP to run on sensor nodes without consuming too much energy or interfering with the operation of the sensor nodes and hence resulting in energy savings and greater network lifetime. Furthermore, the distributed and flexible slot structure of FlexiTP and FlexiTP's depth-first-search forwarding scheme allows WinMS to operate efficiently in any network size.

The synergy among management functions enables WinMS to ensure that a sensor network remains highly available and continues to provide accurate information in the face of network failures.

Configuration Management

The main tasks of WinMS's configuration management are to organise nodes into multihop network (self-forming) and to monitor and control the network topology and node status. Nodes in sensor networks can die, move, or switch into sleep mode to save energy. Configuration management is used to identify the sensor nodes, including failed ones, and also to keep track of their energy levels. The configuration reports are the network topology map and energy map. WinMS achieves configuration management goals by using FlexiTP to build a data-gathering tree for a network and to support dynamic configuration of the network for routing based on the schedule table and collision-slot information of individual nodes, without prior knowledge of the network topology.

Fault Management

Wireless sensor networks are prone to network failures such as dropped packets, nodes dying, being disconnected, powering on or off, and new nodes joining the network and so the networks need to be able to self-configure themselves without knowing anything about network topology in advance. WinMS's fault management strategy is to utilise FlexiTP's local repair to identify whether a fault has occurred, to allow sensor nodes to discover their neighbours dynamically, and to reconfigure the network to function autonomously, so providing self-configuration and self-stabilisation. FlexiTP is fault-tolerant and robust as it allows networks to survive despite occurrences of faults in individual nodes in a network. By using FlexiTP, WinMS is also capable of updating the network topology map without too much overhead.

Performance Management

The main objectives of WinMS's performance management are to monitor the quality of information acquisition by logging the number of packets received at the base station, gathering network statistics, analysing sensor data, predicting future events, and establishing rules for data aggregation. Data aggregation is a method that packs management results into a single message buffer. This approach creates less preamble and message header overhead than returning values individually. Reducing the amount of data transmitted, reduces energy consumption, and so minimises communication overhead. WinMS's data aggregation technique will be described later in Section 7.4. By analysing the sensing data map, WinMS executes the on-demand TDMA slot transfer algorithm, if needed, to allow a network to adapt to current network conditions both locally and globally.

Accounting Management

The main task of WinMS's accounting management is to ensure that wireless bandwidth and energy resources are distributed and used equitably. WinMS establishes metrics and thresholds that are used to trace the network behavior and to limit the usage of resources. For example, WinMS forbids a sensor node with a low battery level to execute an energy consuming task. For example, recall from Chapter 6 that P2P-routers play important roles in preserving connectivity between peers in a P2P-group. WinMS can avoid P2P-routing failures by regularly examining the energy levels of these P2P-routers and informing sensor nodes to select an alternative P2P-router from their local P2P-adjacency redundancy table whenever the P2P-router's energy drops to a certain threshold.

7.3.3 Management Services

In this section, we describe WinMS's local and central network management.

Local Network Management

WinMS's local network management functions are network state update and event detection. The network state update function is responsible for collecting and transporting network states from sensor nodes to the base station through which human managers retrieve the data. The event detection function identifies events of interest and reconfigures the network accordingly.

WinMS's local management scheme consists of two core components: data dissemination and data collection. The *data dissemination* component is responsible for distributing management tasks to a set of managed nodes in a sensor network. In WinMS, the base station piggybacks management tasks onto data packets sent during the FlexiTP's multi-function slot. This approach is superior to the flooding approach in terms of energy consumption and latency. The flooding approach disseminates management tasks by propagating them to the whole network using CSMA. This contention-based scheme introduces overhead due to collisions, and increases idle-listening overhead since nodes must always listen to the medium at all times in order to receive management tasks.

The *data collection* component is responsible for collecting data from managed nodes back to the base station using FlexiTP's data gathering schedule (transmit slot list and receive slot list), multi-function slot, and fault-tolerance slot. WinMS utilises these schedules for regular network state update. For example, human managers may query sensor nodes to report their battery level periodically, say every ten data gathering cycles. Sensor nodes then simply piggyback their replies (battery level) on data packets sent during a data gathering cycle. WinMS also uses the combination of the data gathering schedule and multi-function slot for event-triggered management in which sensor nodes respond to events initiated by external physical phenomena in addition to responding to management queries. Autonomy in this regard is desirable because it helps to conserve energy. For example, when a node's battery level is too low, the node can decide to forward only its own data to conserve energy. During a data gathering cycle, this self-reporting node propagates its management decision to the central manager (base station), while in the multi-function slot cycle, the self-reporting node commands its children to find a new parent using the fault tolerance scheme described in Section 4.2.5.

Central Network Management

The base station can act as the central network manager with unrestricted access to power, storage capacity, and computing capability. The base station is able to execute a wide-range of management and maintenance tasks based on post-mortem analysis of global states of the network. Recall that WinMS's local network management scheme allows the central manager to update sensor network models based on data collected from the network. WinMS's central network management maintenance is categorised into reactive and proactive network management maintenance.

Reactive network management maintenance refers to corrective actions that are taken in response to current events, as follows:

- *Data processing method.* The central manager decides whether or not to use in-network processing based on network conditions and management policies.
- *Self-reporting node query.* A self-reporting node can request extra communication slots from the central manager if it is unable to get the requested slots locally. In this case, the central manager executes the central on-demand TDMA slot transfer algorithm described in Section 5.2.2.
- *Fault detection.* The central manager detects and localises faults by analysing anomalies in sensor network models. The central manager can be programmed to analyse the topology map and the energy map to monitor link qualities and detect faults. For example, if the topology in a certain part of the network changes frequently while all nodes in that sub-network are still alive, this event may indicate varying link qualities. Another example is when the central manager has not heard anything from a particular node after few data gathering cycles and the potential faulty node has recently reported a low battery level, the central manager identifies

this event as a failure due to energy depletion. If energy levels of the faulty node and its routers are sufficient, and nodes that share the same paths as the faulty node have managed to send their data to the central manager, then the central manager identifies this event as a broken link between the faulty node and its parent.

Proactive network maintenance refers to preventive actions that are taken to prevent faults, by predicting future events, as follows:

- *Duty cycle assignment.* This central manager monitors nodes' energy level and avoids assigning management roles to nodes that have spent too much of their energy. The central manager can also distribute loads among sensor nodes to prolong the lifetime of the network by determining which sub-networks need more slots than other sub-networks. For example, if data reported by nodes in certain parts of the network are stable over a period of time, the central manager can reconfigure the traffic pattern of the network to conserve node energy. Depending on the application management policy, the central manager may decide to relieve sensing tasks temporarily in that sub-network and so nodes in that part of the network do not send their data to the base station for a period of time. Alternatively, the central manager may allow these nodes to perform on-demand aggregation to reduce the number of packet transmissions. Our on-demand aggregation algorithm will described in Section 7.4.

- *Traffic pattern configuration.* The central manager reconfigures the network according to the sensing data map, using the central on-demand TDMA slot transfer algorithm. For example, if the central manager identifies rapid sensor readings changes in a sub-network, it can allocate extra time slots to nodes in that needy sub-network and so allow these nodes to report their data more frequently in a data gathering cycle.

- *Fault prevention.* The central manager predicts future events and failures and takes preventive action accordingly. The central manager can detect areas of weak network health based on the energy and topology maps. A weak network area may cause network partitions. The central manager can take preventive action by instructing nodes in that sub-network to sense and send data to the base station less frequently and transferring unused slots from those nodes to other parts of the network. If the nodes are critical for covering the area, the central manager informs human managers to replace the nodes' batteries or to place redundant nodes in that area.

7.4 On-Demand Aggregation Algorithm

Traditional algorithms to support in-network aggregation in wireless sensor networks rely upon an imposed data aggregation scheme based on queries [55, 78, 101, 106], predefined threshold settings [80, 103], or data redundancy [62, 127]. Neither of these algorithms support on-demand aggregation, that is, in-network aggregation based on network capacity constraints such as non-uniform frequency of data collections and different priorities of data. Nodes in the network may have different priorities of data because of event-triggering sensor readings or various types of sensor readings they provide. Thresholds are important for detecting events of interest and so support reactive sensing in sensor networks. For example, abnormal or fast changing data are more important than slow changing data. When nodes with high priority packets increase the frequency of their data collections, the network bandwidth may be dominated by these nodes. It is desirable to allow nodes with low priority packets to aggregate their packets and so enable these nodes to send their data to the base station under the currently available network bandwidth.

Our proposed *on-demand aggregation* (ODA) algorithm facilitates a TDMA-slot blockage mechanism by allowing nodes to perform in-network aggregation based on network capacity constraints. The main idea is to allow a node to reserve few slots in its schedule for itself, according to a desired data aggregation compression ratio, and so other nodes can only claim the unreserved slots. When a node's unreserved slots are claimed by other nodes, the node then performs an on-demand aggregation using its reserved slots. The data aggregation ratio impacts the level of granularity of information sent to the base station. An aggregation packet may contain a result of any aggregation functions such as SUM, AVG, MAX, MIN, or MEDIAN.

The ODA algorithm can also be used for data aggregation schemes based on queries (pull aggregation) or some network policies (push aggregation). For example, the central manager can query some nodes in the network to send their data to the base station using their TDMA data gathering schedules while other non-queried nodes send their data to the base station through aggregation.

There are two key benefits of the ODA algorithm. First, it provides a means for data management in terms of data complexity and quality, by allowing human managers to specify the level of information granularity received from the network through a data aggregation compression ratio. Second, it can be used for power management by allowing nodes to perform an on-demand aggregation to achieve further energy conservation. In the next sections, we briefly describe existing data aggregation algorithms designed for wireless sensor networks and we then elaborate our on-demand aggregation algorithm.

7.4.1 Related Work

In wireless sensor networks, data processing and computation consume much less energy than data communication. The most common data aggregation model [55, 78, 101, 106, 111] builds a spanning tree upon a node's request through CSMA and the tree is rooted at the requesting node. Typically, the base station broadcasts a query throughout the entire network and a data aggregation tree is the inverse of a broadcast tree. The main drawback of this broadcast-based *pull* aggregation scheme is that it has to build a spanning tree upon a query. Furthermore, in a multihop environment, CSMA is subject to hidden terminal interference, which leads to high energy consumption. In contrast, the ODA algorithm utilises FlexiTP's scheduled-communication scheme to eliminate collisions. FlexiTP, initially, builds a data gathering tree and nodes' schedules. Nodes then maintain their schedules throughout their lifetime in the network. A query is disseminated to nodes in the network using nodes' TDMA schedules and hence collision-free traffic is guaranteed. Nodes in the network then adaptively adjust their schedules based on the query.

One of the notable data-centric routing protocols for wireless sensor networks, namely directed diffusion [55], uses CSMA-based protocol that allows any node to perform data aggregation if the node is sink of various links. However, nodes in the shortest path from sources to the sinks do most of data aggregation. Furthermore, data is forwarded upon reception. In contrast, TAG [78], Convergecasting [101], and cascading scheduling [106] use mechanisms that allow nodes to delay transmitting a sensor reading in order to aggregate readings from other nodes. These protocols employ an EPOCH division mechanism in which each query is given an EPOCH that represents the time to wait for getting the response for the query. They divide the EPOCH into short time slots. They then assign aggregation time slots to nodes with data matching the query. Although the EPOCH mechanism avoids collisions, some packets may be dropped because of a short EPOCH. In contrast, the ODA algorithm allows nodes to try getting any number of extra slots they require to send their data to a destination.

In APTEEN [80] and TiNA [103], nodes report and perform data aggregation when nodes' sensor readings meet the predefined thresholds. The ODA algorithm also supports this *push* aggregation scheme by allowing nodes to aggregate their children or descendants' data, or to switch to the sleeping mode during their scheduled slots if a child or descendant has no data to send.

All of the discussed aggregation protocols only implement *imposed-based data aggregation* schemes based on queries, predefined threshold settings, or data redundancy, whilst the ODA algorithm supports an on-demand aggregation based on the current network capacity constraints.

7.4.2 Algorithm Description

The ODA algorithm is used to facilitate a TDMA-slot blockage mechanism in which nodes reserve some slots in their schedules. It serves two purposes:

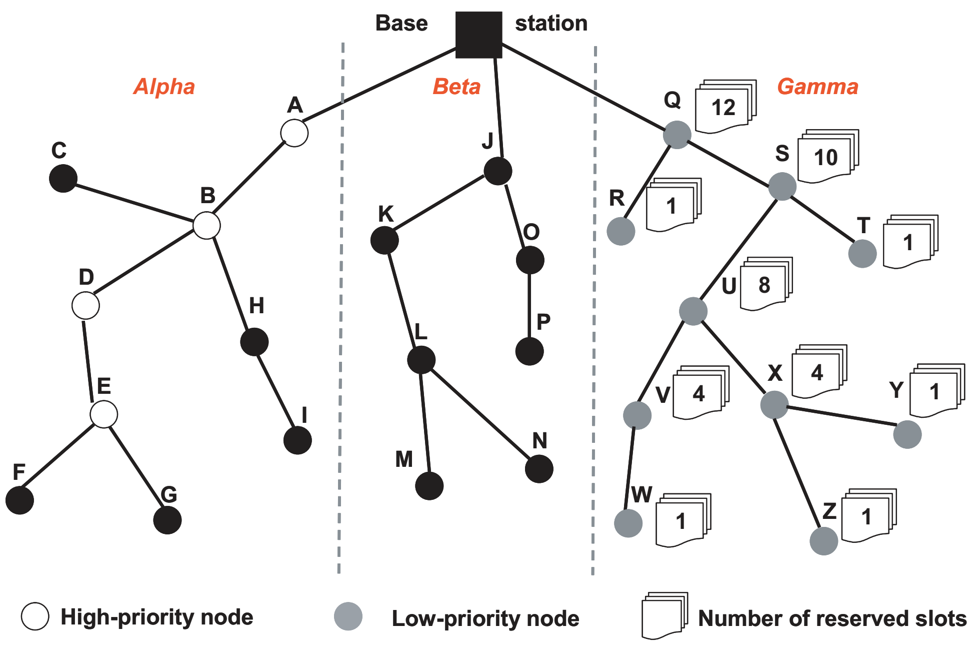

Figure 53: The network of sensor nodes for monitoring three different local regions: Alpha, Beta, and Gamma.

1. to allow nodes to adjust themselves according to current network constraints and

2. to allow human managers to impose queries and network policies.

To motivate our research, consider the environmental monitoring application in Figure 53. Sub-tree *A* that consists of nodes labeled *A* to *I* is responsible for monitoring the Alpha region, sub-tree *J* consisting of nodes labeled *J* to *P* is responsible for monitoring the Beta region, while sub-tree *Q* consisting of nodes labeled *Q* to *Z* monitors the Gamma region. Suppose the human manager wants to know environment conditions in all regions at all times, however, they prefer nodes that record rapidly changing sensor readings (high-priority nodes) to send data to the base station more frequently, while other nodes (low priority nodes) can lend some of their slots to the high-priority nodes but also reserve few slots to send aggregated data of their sub-trees.

The ODA algorithm allows human managers to specify the data aggregation compression ratio (let it be `r`) of every node in the network. If $r = 1$, it means that each non-leaf node (parent node or root of a sub-tree) must reserve:

- one reception slot for receiving data from its children in its sub-tree,
- one transmission slot for sending an aggregation packet for its own sub-tree,
- one reception slot each for receiving an aggregated packet from each of its descendants' sub-trees, and
- one transmission slot each for forwarding an aggregation packet for each of its descendants' sub-trees.

Thus, when `r = n`, a non-leaf node must reserve `n` reception slots and `n` transmission slots for its sub-tree and for each of its descendants' sub-trees. Each reserved transmission slot is used for sending different aggregation results. Note that a leaf node does not need to reserve any data gathering slots since it has no children.

In addition to the reserved data gathering slot(s) according to `r` value, a node also always reserves its multi-function slot in order to keep all slots in the network synchronised at all times. Therefore, every node in the network must reserve:

- one multi-function reception slot for receiving a synchronisation packet from its parent, and
- one multi-function transmission slot for multicasting a synchronisation packet to its children if any.

The novel approach of our on-demand aggregation scheme lies on the ability of a non-leaf node to select which of its children can send their data to it in a data gathering cycle based on their sensing requirements. For example, in Figure 53 parent *X* reserves four slots:

- one reception slot for collecting data from either child *Y* or child *Z*,
- one transmission slot for sending an aggregation packet of its sub-tree,
- one multi-function reception slot for receiving a SYNC packet from its parent *U*, and

- one multi-function transmission slot for multicasting a SYNC packet to *Y* and *Z*.

If child *Y* reports more important data than child *Z* in past data gathering cycles, parent *X* decides to dedicate the reserved reception slot for child *Y*. Otherwise, parent *X* can alternate the reserved reception slot between child *Y* and child *Z*. This data-oriented rostering approach can significantly reduce energy consumption, while meeting network bandwidth constraints.

7.4.3 Algorithm Formulation

Given a node `x`, let `d` be descendants of `x`. `S(x)` is an indicator whether a node is a non-leaf node or leaf node. `S(x) = 1` if a node is a non-leaf node, else `S(x) = 0`. `C(x)` denotes the number of compulsory reserved multi-function slots at `x`. By default, `C(x) = 2` if `x` is a non-leaf node, else `C(x) = 1` if `x` is a leaf node. The final value of `C(x)` is:

$$C(x) = 1, \quad if \quad S(x) = 0 \quad and \quad r > 0 \tag{9}$$

$$C(x) = 2, \quad if \quad S(x) = 1 \quad and \quad r > 0 \tag{10}$$

`T(x,r)` denotes the total number of reserved transmission slots at `x` with data aggregation compression ratio of `r`, which is calculated as follows:

$$T(x,r) = \sum_{i=1}^{d}(S(d) \cdot r) \tag{11}$$

`R(x,r)` denotes the total number of reserved reception slots at `x` with data aggregation compression ratio of `r`, which is calculated as follows:

$$T(x,r) = \sum_{i=1}^{d}(S(d) \cdot r) \tag{12}$$

Thus, the total number of reserved slots at `x` (denoted as `ODA(x,r)`) is formulated as follows:

$$ODA(x,r) = T(x,r) + R(x,r) + C(x) \tag{13}$$

FlexiTP's network can be modeled as `N(n)`, that is, the total number of slots per data gathering cycle in a network of `n` nodes without the slot reuse scheme. The value of `N(n)` includes multi-function slots in the network. Clearly, in a single hop network, no packets can be aggregated since nodes send their data directly to the base station. In this case, `N(n) = n + 1` because each node in the network only requires one transmission slot to send its data to the base station and the base station requires one transmission slot to multicast a multi-function packet to all nodes in the network. In contrast, deeper trees increase the number of data aggregation packets in the network as nodes' packets within a sub-tree can be merged into one or more aggregation packets, depending on the application requirements. However, the deepest tree scenario in which each node in the network only has a single child (often referred as the line topology network) also results in no aggregation. In the line topology network, `N(n)` $= \frac{n(n+1)}{2} +$ `n`. Thus, the lower and upper bound of the total number of reserved slots in the network with any value of `r` is as follows:

$$0 \leq \sum_{i=1}^{n} ODA(n,r) \leq \frac{n(n+1)}{2} + n \tag{14}$$

7.4.4 TDMA-Slot Blockage Aggregation Scheme

We first illustrate an on-demand aggregation case study based on network capacity constraints. Let us say nodes *A*, *B*, *D*, and *E* in the Alpha region in Figure 53 are self-reporting nodes that request one extra slot in a data gathering cycle, and all nodes in the Gamma region are willing to lend some of their schedules. According to slot demand and supply calculations described in Section 5.2.2, the total number of extra slots required by self-reporting nodes is *10* slots and the total supply of transmission slots provided by slot-supplier nodes in the gamma region is *44* slots: *39* slots for data gathering and *5* slots for synchronisation.

For simplicity, we assume the data aggregation compression rate is one for all nodes in the network. Thus, each of slot-suppliers *R*, *T*, *W*, *Y*, *Z* reserve one slot for receiving

SYNCH data from their parents in their multi-function slot. Both slot-suppliers *X* and *V* reserve four slots: one slot for receiving data from any of its children, one slot for sending aggregated data of its sub-tree, and one slot each for receiving and sending SYNCH packet. Furthermore, the slot-supplier *U* reserves eight slots: one slot for receiving a packet from its sub-tree, one slot for sending its sub-tree's aggregation packet, one slot each for receiving aggregation packet from descendant *V* and descendant *X*, one slot each for forwarding descendant *X* and *V*'s aggregation packet, and finally two slots for receiving and transmitting SYNCH packet. Recall that in FlexiTP, a transmission slot of a node overlaps with the reception slot of the node's parent. Thus, in total, slot-supplier nodes in the gamma region reserve 19 distinct transmission slots for data gathering and five transmission slots for synchronisation. The total number of unreserved slots is *20* slots (39 - 19). The self-reporting nodes in the Alpha region can use these unreserved slots as their extra communication slots. In this way, the human manager get more information from nodes in the Alpha region in a data gathering cycle, while still getting information from nodes in the gamma region at a lower granularity.

7.4.5 Imposed Aggregation Schemes

Now, we illustrate imposed aggregation schemes where nodes perform in-network aggregation in response to management queries (pull aggregation) or event triggers (push aggregation). The *ODA pull aggregation* scheme enables human managers to pose a query to a network and nodes with data matching the query send their data to the base station using their data gathering schedules, while the rest of the nodes in the network send their data to the base station through aggregation. In contrast, the *ODA push aggregation* scheme allows human managers to pre-program thresholds on sensor nodes in advance before node deployment and nodes use these thresholds to detect event triggers and perform a distributed task according to a predetermined management policy.

The human manager can organise nodes in the network into various logical or geographical subsets and assign specific tasks to each subset. In ODA, initially, when a node enters the network, it advertises the types of sensor readings it provides to the base station during the FlexiTP's time slot assignment phase. In this phase, each node in the network builds a *local identifier table* that stores sensor data types of its children and descendants. For example, the human manager may wish to select a set of dominating nodes to represent sensor data coverage of a network. In Figure 53, the human manager may select nodes *A*, *D*, *F*, and *I* in the Alpha region; nodes *J*, *L*, and *N* in the Beta region; and nodes *Q*, *T*, and *X* the Gamma region to be the network-coverage subset. Furthermore, the human manager may wish to allow nodes with the same sensor data type, different sensor types, or combination of both to form a subset. Thus, a subset of nodes can be formed according to the application's requirements. During data gathering cycles, the human manager may query a specific subset for a certain management purpose.

ODA Pull Aggregation

When the human manager decides to query a certain subset, nodes in the network dynamically locally adjust their TDMA schedules based on their local *identifier tables*. This pull aggregation scheme can be utilised for energy management. For example, the human manager can query a certain subset of nodes to send data to the base station using their data gathering schedules, while other nodes send their data through aggregation. This approach can promote energy conservation as only nodes matching a query fully use their data gathering schedules, while non-queried nodes can shut down some of their idle slots.

The ODA pull algorithm works as follows: a query is disseminated to nodes in the network through the multi-function packet propagation. Upon receiving a query, a node checks if its identifier matches the query. The node also checks if any of its children and descendants' identifiers matches the query. If a node's identifier matches the query, the node's data will be propagated by its routers to the base station, otherwise the node's data is only propagated up to its parent in order to be aggregated with other nodes' data. This means that a parent node aggregates all of its non-queried children and descendants' data into a specified number of aggregation packets (data aggregation compression ratio) and propagating these packets up to the base station using reserved slots. In contrast to the ODA's on-demand aggregation based on network capacity constraints, the total number of reserved slots at a node executing the ODA pull algorithm is independent of the number of descendants' sub-trees. Basically, a node only activates normal data gathering schedule for queried children and descendants, and uses an on-demand aggregation scheme for propagating non-queried children and descendants.

ODA Push Aggregation

Nodes in the network can be pre-programmed with network polices such as data tolerance level. The human manager may want a network to send an aggregated value of nodes with sensor readings within a tolerance level, over a certain number of data gathering cycles. This approach can reduce the number of transmissions in the network and hence save energy. To achieve this goal, we incorporated a bit notification scheme described in [111] with our FlexiT protocol. Basically, during a node's scheduled transmission slot, the node transmits a `bit value of 0` to its parent if it meets a network policy, otherwise it fully transmits its data to the parent. A parent node switches to low-power sleep mode for the rest of scheduled reception slot for the child's transmission if it receives a one-bit notification (`bit value of 1`), otherwise the parent keeps awake for receiving the child's data. We extended this idea by allowing nodes to aggregate their children or descendants' data in the next data gathering cycle whenever they receive one-bit notifications from their children or descendants in the current data gathering cycle.

The ODA push algorithm works as follows: each node that meets a network policy (self-reporting node) sends a one-bit notification to its parent. A self-reporting node's parent and ancestors keep propagating this one-bit notification up to the base station during scheduled data gathering slots of the self-reporting node. After one data gathering cycle, every node in the network adjusts its TDMA schedule based on one-bit notifications received during the previous data gathering cycle. A node merges packets received from its children and descendants that report one-bit notifications. Note that the number of aggregation packets depends on the application's data aggregation compression ratio.

The combination of ODA pull and push aggregation scheme can be used for on-the-fly query. This feature allow users to specify queries other than predetermined subset identifiers or predetermined thresholds. For example, the human manager may wish to only allow nodes with sensing readings within certain values to send their data to the base station normally, while others send through aggregation. The query is disseminated to the network through multi-function slots. Nodes with data matching the query propagate a one-bit notification to the base station. This allows nodes in the network to adjust their schedules adaptively based on this information. The scheme enables the network to respond and adapt itself to a query in just one data gathering cycle. Note that similar to the OST algorithm, there is a timer associated with all ODA schemes and so nodes resume their normal schedules once the timer expires.

7.5 Performance Evaluation

In this section, we evaluate the performance of both on-demand TDMA slot transfer algorithm and ODA TDMA-slot blockage algorithm (OST-ODA), under varying network constraints and energy conservation queries. The ODA algorithm is implemented as an extension to the FlexiTP protocol and the code is downloadable at [68]. The simulation environment used in this study consisted of a 300 x 300 square meter flat area, involving up to 400 nodes. All simulation results are averaged over 20 different network topologies. In all simulations, the FlexiTP's data gathering slot size and fault-tolerance slot size is set to 23 ms and 100 ms respectively. The channel radio operates at 19.2 Kbps. We switched off FlexiTP's slot reuse algorithm for all simulations presented in this section. Note that a node's multi-function slot is always reserved and so all slots in the network are still synchronised in the presence of ODA events.

7.5.1 TDMA-Slot Blockage Aggregation Evaluation

We used the similar non-uniform sensing simulation setup as in Section 5.3 in order to evaluate the performance of OST-ODA under varying network constraints. In simulated data gathering networks, a certain number of nodes in the network perform data gathering tasks with inter-packet arrival of two seconds (low-priority nodes) and a certain number of nodes increase their offered loads with inter-packet arrival of one-second (high-priority nodes). In this simulation setup, nodes with high-priority data request extra communication slots to send their data more rapidly in a data gathering cycle, while nodes with low-priority data perform on-demand aggregation in order to send their data to the base station using their reserved data gathering slots.

For simplicity, we set the data aggregation compression ratio to be equal to one in all simulations. The data aggregation compression ratio determines the supply of extra slots in the network. A data aggregation compression ratio of one means that every parent node in the network must reserve one transmission slot and one receive slot for its own sub-tree and for each of its children and descendants' sub-trees.

Figure 54 presents the average percentage of met demand using the OST-ODA algorithm across various network densities. In low to medium demand cases, where the percentage of slot-requestors is less than 60% of total nodes in the network, the OST-ODA algorithm can satisfy 85% to 100% of the slot demand.

Recall from Chapter 5, Figure 32 shows that the percentage of demand allocation using the on-demand TDMA slot transfer algorithm in the absence of ODA algorithm, is around 43% to 75% in high demand cases. Figure 54 shows that the performance of OST-ODA in supporting non-uniform sensing degrades by 3% to 19% in high demand cases, where the percentage of slot-requestors is more than 50% of total nodes in the network. The simulation results has shown that OST-ODA enables nodes to adapt their schedules locally to changing demand and performs well under considerable network constraints.

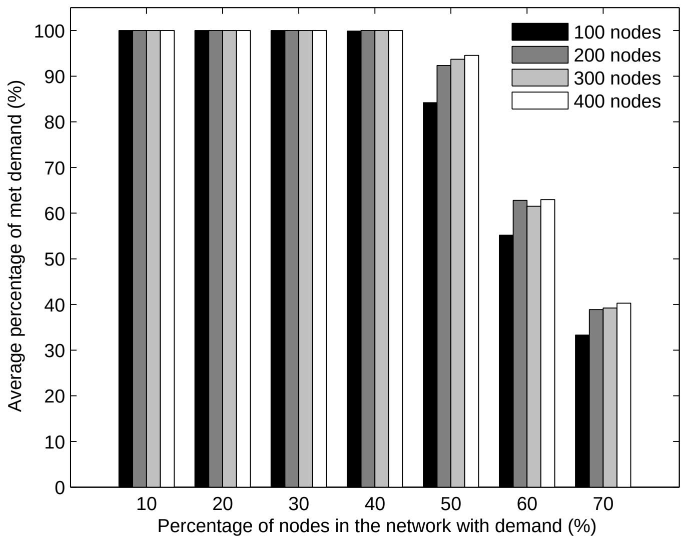

Figure 54: OST-ODA percentage of demand allocation versus total percentage of nodes in the network with demand, across various network densities. For example, in the network of 400 nodes, if 70% of nodes become slot-requestors, it means that 280 nodes in the network ask for one extra slot in a data gathering cycle.

7.5.2 Imposed Aggregation Evaluation

We conducted an experiment to evaluate the amount of energy saved if ODA imposed aggregation technique is employed. In this experiment, all nodes in a network are queried to perform data gathering using the data aggregation compression ratio of one. The inter-arrival packet time for all nodes in a network is set to two seconds.

Figure 55 shows the percentage of energy saving per node, that is, the ratio of total energy saving when nodes perform data gathering using their ODA schedules to the total energy consumed when nodes perform data gathering using their normal TDMA schedules. Our ODA imposed aggregation scheme resulted in an 18% to 24% improvement in energy efficiency across various network densities.

The simulation illustrates that the ODA imposed aggregation scheme provides a mean for reducing the energy consumption at all levels of the network. Furthermore, this power scheduling approach can be utilised to significantly reduce idle slots by locally managing

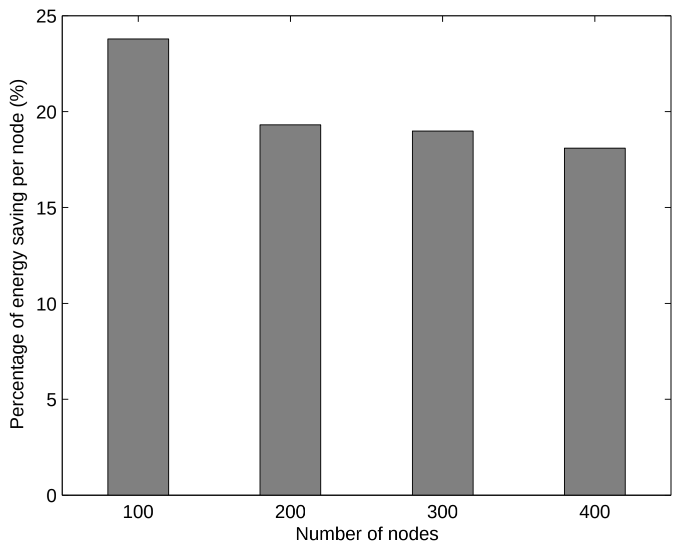

Figure 55: ODA percentage of energy saving per node versus network density.

supply and demand such that nodes can use their ODA-schedules whenever their reported sensor data are stable for a period of time. Nodes can even ask for more slots if they detect interesting data as shown in Section 7.5.1.

7.6 Conclusion

This chapter presented WinMS, an adaptive policy-based management system for wireless sensor networks. WinMS adapts to changing network conditions by allowing the network to reconfigure itself according to current and forecasted events, in order to maintain the performance of the network and achieve effective networked node operations.

WinMS local network management scheme allows individual sensor nodes to autonomously perform management functions according to their neighbourhood network states, such as topology changes and event detections. WinMS's central network management scheme uses the central manager with a global knowledge of the network to execute corrective and preventive management. The central manager stores and maintains sensor network models that depict global views of network states. The central manager analyses the correlation among sensor network models to detect interesting events such as areas of weak network health, possible network partition, noisy areas, and areas of rapid data changes. Thus, WinMS provides the potential for better monitoring, control, and management of sensor networks.

WinMS uses FlexiTP, a lightweight TDMA protocol, to provide energy-efficient management, data transport, and local repair. WinMS utilises the on-demand TDMA slot transfer algorithm to allow nodes to claim extra communication slots based on network conditions without human intervention. This algorithm provides automatic self-configuration and self-stabilisation both locally and globally. WinMS uses FlexiTP to support time slot transfer among nodes in the network and to ensure slots are systematically transferred among nodes, while maintaining low energy overhead and fast convergence. Utilising FlexiTP, the network can be reconfigured and stabilised within a single data gathering cycle.

WinMS also provides a TDMA-slot blockage mechanism that allows nodes in the network to reserve some of their slots according to a desired data aggregation compression ratio. In this way, nodes can preserve forwarding tasks for their own data, data from children, or descendants' data, if their schedules are claimed by other nodes. Nodes perform an on-demand aggregation based on network capacity constraints in addition to imposed aggregation schemes such as users' queries or some network policies. This scheme provides means for facilitating data management and energy management.

The policy-based network management approach is a promising one for the management of wireless sensor networks, but it requires the flexibility to adapt to a constantly changing environment. Through simulation studies presented in this thesis, we determine that using our proposed management solution approach, we can achieve our objective of a self-organising, robust, and efficient adaptive system for managing wireless sensor networks. Furthermore, our simulation results in this chapter show that the proposed ODA algorithm can sustain good performance under varying TDMA-slot blockage conditions and achieve significant energy savings when used for energy conservation queries.

Chapter 8

Conclusion and Future Work

The important thing is not to stop questioning.

Albert Einstein

This chapter highlights problems that we attempted to solve. We also revisit some of the assumptions and limitations of the approaches proposed in this thesis, discussing opportunities for future work.

8.1 Summary of Contributions

In this thesis, we present the FlexiTP architecture for scheduling, routing, supporting time slot-transfer, supporting on-demand peer-to-peer communication, and managing wireless sensor networks. To our knowledge, our research is the first attempt to apply the concept of flexible-schedule-based approach to achieve energy-efficiency, to provide end-to-end guarantees on data delivery, to support adaptability to changing network topology and traffic patterns, and to support reactivity to different communication requirements, in wireless sensor networks.

The main components and contributions of this thesis are summarised below:

1. *FlexiTP, a flexible-schedule-based TDMA protocol for wireless sensor networks.* In wireless sensor networks, the primary design principles are energy efficiency, effective collision avoidance, scalability, and adaptivity to changing network conditions and network densities. Classical performance attributes such as fairness, latency, throughput, bandwidth utilisation, and reliable data delivery, are normally secondary. However, as new applications of sensor networks emerge, providing these performance assurances gain importance. FlexiTP provides end-to-end guarantees on data delivery, whilst also respecting the severe energy and memory constraints of wireless sensor networks. FlexiTP achieves this balance through a novel distributed, synchronised, and loose slot structure in which sensor nodes can build, modify, or extend their schedules with minimal control overhead. FlexiTP further improves the reliability of data delivery by allowing a network to perform a local repair when it detects node and communication faults. Since nodes can dynamically adjust their schedules based on their local information during the runtime of the application, it is not necessary to specify the number of slots required for the network in advance and nodes can join or leave the network at any time. This flexible slot structure makes FlexiTP strongly fault-tolerant, adaptive, and also highly energy-efficient.

2. *OST, a novel on-demand TDMA slot transfer method for wireless sensor networks.* Most protocols designed for data gathering applications implicitly assume a periodic rate of data collection from all nodes in the network to the base station. While this is sufficient for sensing parameters that change slowly over time, upon detecting certain interesting events, individual nodes in a network may need to increase the rate of data gathering for reporting important data in real-time. Though CSMA-based protocols can easily adapt to traffic fluctuations in theory, they incur a high energy penalty due to message collisions and retransmissions at high traffic loads. In general, TDMA-based protocols can provide good energy efficiency, but they are not flexible to changes in traffic patterns because of their rigid slot

structure schemes. We addressed this problem by proposing the OST algorithm that allows time slots from one part of the network to be transferred to another part. Since nodes can adjust their schedules based on their current sensing requirements, the OST algorithm can optimise wireless bandwidth and conserve energy. When combined with FlexiTP, the OST algorithm provides flexibility for periodic data gathering networks to move towards event sensing applications and to return to periodic data gathering tasks anytime. Hence, OST allows FlexiTP to be an energy-efficient, reactive, and traffic-adaptive protocol.

3. *WiseP2P, a wireless sensor network peer-to-peer algorithm.* Most applications of wireless sensor networks adopt specific traffic patterns: broadcast from the base station to all sensor nodes in a network (query distribution) or convergecast from all nodes in a network towards the base station (data collection). Our WiseP2P algorithm addresses the problem of lack of peer-to-peer communication in the design of a TDMA-driven protocol by providing flexibility in TDMA source-to-sink communications. WiseP2P enables a node to communicate with any other node in the network without going through the base station. This approach allows an on-demand peer-to-peer communication to co-exist with periodic data collection. Improved energy efficiency is achieved by guaranteed collision free communication, reduced hash table sizes, simple slot allocation schemes, and low-overhead slot transfer schemes.

4. *WinMS, a wireless sensor network management system.* WinMS is a policy-based network management architecture that provides flexibility and adaptability in the face of changing network conditions. WinMS allows human managers to pre-program the entire network by setting event-trigger thresholds and setting certain policies at nodes. Nodes in a network autonomously manage and adapt to current levels of energy available, network failures, communication losses, changing network topologies, and changing data rates. WinMS achieves these management goals by utilising FlexiTP to retrieve local and global states of a network. WinMS

utilises the on-demand TDMA slot transfer algorithm to provide necessary management functions to maintain the performance of the network and achieve effective networked node operations. Furthermore, WinMS provides an on-demand aggregation algorithm, a TDMA-slot blockage mechanism that enables nodes in the network to reserve some slots in their schedules, therefore, other nodes can only claim unreserved slots. When a node's unreserved time slots are claimed by other nodes, the node then dynamically adjusts its TDMA schedule and performs on-demand aggregation using its reserved slots. This approach provides means for facilitating data management and energy management since nodes can autonomously perform data aggregation based on their current sensing requirements and capacity constraints of the network, such as non-uniform data collections and non-uniform priorities of sensor readings in different parts of the network.

5. *Cross layer design.* FlexiTP combines the operation of the MAC layer with the routing layer, and the application layer, to improve network performance and to limit resources needed for operation. In FlexiTP, the scheduling information of the MAC layer is utilised for routing. FlexiTP uses forwarding-to-parent routing over a tree where nodes route packets to their parents until they disconnect from their parents. One may argue that this simple cross-layer-based routing scheme may reduce the overall performance of the system. However, we have demonstrated that this simple routing scheme sustains high throughput and good energy efficiency because once a tree is constructed, every node in the network knows where to relay its packet since nodes remain static throughout their lifetime [15]. Furthermore, FlexiTP allows information sharing between the MAC layer, the routing layer, and the application layer, to allow nodes to dynamically and efficiently adjust their schedules according to network sensing requirements, network communication constraints, and changes in network topology.

8.2 Limitations

The proposed protocols presented in this thesis have several limitations. First, they are designed for a static sensor network in which nodes may suffer temporary or permanent failure, and new nodes may be added to the network at any time but nodes otherwise remain in the same location. Therefore, the proposed protocols are not unsuitable for a sensor network with mobility. Recall that in Chapter 4, we simulated extreme conditions where we added and removed a substantial number of nodes to or from the network. We can view actions of new nodes and dead nodes in the network as node-mobility since these nodes may reflect nodes that move from a part of network to another part of the network. Our simulation results show that as the total number of new nodes and dead nodes increases, FlexiTP local repair performance decreases.

Second, the proposed protocols do not support real-time recovery since all communications are slotted. FlexiTP requires at least a data gathering cycle (i.e., frame) to allow the network to stabilise itself against topology changes and different communication (traffic) requirements. The time required for a network to fully stabilise itself to the correct state depends on the length of FlexiTP's fault-tolerance slot. In the best case, participating nodes are able to adjust their schedules during FlexiTP's fault-tolerance slot in a frame. Users can either set a very short fault-tolerance period for a sensor network application that is deployed in a stable environment or an application that can tolerate a slow network stabilisation. A long fault-tolerance period is used for a sensor network application that is prone to network errors.

Our proposed protocols utilise the FlexiTP's multi-function slot to piggyback slot reassignment (recovery) information. Ideally, this slot should carry one packet. This approach limits the amount of recovery information that can be disseminated to nodes in the network in a frame. As the total nodes with demands (e.g., local repair, extra time slots, and peer-to-peer communications) increases, the network stabilisation time increases. Users can set a longer multi-function slot to allow multiple packets transmissions. However, they do this at the expense of high energy consumption for idle listening overhead.

8.3 Future Work

There are several avenues of future work. The next critical step is implementing FlexiTP in sensor network operating systems and applications. Drawing lessons from existing real-world experiments such as habitat monitoring of wildlife at Great Duck Island [79], the deployment of the wireless sensor network executing FlexiTP should consider the suitability of variety of hardware technologies and unique requirements of sensor network applications. The human manager can adjust different FlexiTP parameters to best support the goal of a sensor network application. For example, if a wireless sensor network is to be deployed for environmental monitoring in a remote area and the network is expected to run for a long time, the human manager can pre-program FlexiTP such that it conserves the highest energy possible while providing maximum application benefit to the end-user. The human manager may configure FlexiTP to allow nodes to perform data collection only every certain number of data gathering cycles, and to allow these nodes to perform regular data collections in each data gathering cycle when they detect interesting events. We have shown, through many simulations in NS-2, that radio interference range is far from a circular-disc model. When FlexiTP is implemented with its slot-reuse scheme, it is important to understand the relationship among node placement, effective communication range, realistic-interference range, and radio transmit power at each wireless sensor network deployment, in order to ensure collision-free traffic. Thus, real-world sensor network deployments may need a complex combination of autonomous and manual configuration.

The proposed protocols in this research are optimal for periodic data gathering networks. During network execution, these protocols enable periodic data gathering networks to move towards event-detection activities. In the future, we plan to investigate the optimal strategy for mediating access in multi-hop sensor networks with varying traffic characteristics. One solution is to allow FlexiTP to adjust nodes' slot-schedules dynamically based on historical behaviours of nodes in the network. For example, after assigning initial slot-schedules to nodes in the network, FlexiTP observes data collected by nodes in the network for a period of time. Based on this observation, FlexiTP can re-assign slots among nodes in the network in order to reconfigure nodes according to their specific traffic requirements and allow these nodes to adopt their renewed slot-schedules as their default slot-schedules throughout their lifetimes in the network.

Another scope to advance our research in the future is to enable the proposed protocols to adapt to changes in mobility patterns. Recent work has enabled deployments of wireless sensor networks consisting of both mobile and static nodes in applications ranging from environmental monitoring to medical monitoring. For example, sensor nodes attached to human body for tracking and monitoring doctors and patients inside a hospital. Precision agriculture applications could also take advantage of a mobile robot's capabilities. A robot can act as a gateway to the sensor network to periodically collect data along the network. A robot can also recalibrate sensor nodes with inaccurate sensor measurements. Therefore, protocol architectures for wireless sensor networks need to be developed to address the effects of mobile nodes [5].

WinMS incorporates management functions within application protocols. The development of general purpose network management protocols for wireless sensor networks remains a challenging problem that requires further investigation. Another significant open problem is the development of management policies and expressive languages or metadata for representing management policies and for representing the information exchanged between sensor nodes, managers, and end users.

Another future outlook is to optimise the technology of new generation of sensor network hardware in the protocol design. The advanced radio hardware technology enables the dynamic power control of radio. It can be utilised to empower the MAC layer to control low-level communications and to adapt to changes in the physical environment [28, 73]. For example, a sensor node can dynamically adjust its radio transceiver power for transmission based on its communication requirements. If a sensor node wishes to transmit data to its neighbouring nodes, it can minimise the energy consumption by decreasing the transmission power required for reaching those neighbours. This adaptive transmission power scheme also gives the advantages of obstacle avoidance and fault tolerance [14].

A sensor node can increase its transmission power when it is repelled by an obstacle. In sparse networks, a sensor node can also increase its transmission power if its current transmission range cannot reach any nodes in the network.

8.4 Concluding Remarks

In conclusion, the research described in this thesis contributes to our understanding about the best combination of techniques using a flexible-TDMA-slot-structure approach to achieve energy-efficiency, to guarantee predictable performance, to provide adaptability to topology changes and different communication (traffic) requirements.

FlexiTP is the base protocol for all algorithms proposed in our research. FlexiTP is suitable for energy-constrained sensor nodes because the initial network setup is performed once and this initial cost can be amortised over the network lifetime and eventually balanced by improved throughput and energy efficiency. In this thesis, we have demonstrated how FlexiTP's flexible-slot-structure supports a dynamic slot assignment. This approach enables nodes to adjust their slot schedules through the on-demand TDMA slot transfer algorithm, the wireless sensor network peer-to-peer algorithm, and the on-demand aggregation algorithm in order to adapt to constantly changing environments. Our proposed algorithms can support a variety of wireless sensor network applications, ranging from environmental monitoring, to vibration-based structural monitoring and detection, and to smart office surveillance.

Comprehensive simulation in NS-2 allowed us to evaluate the performance of our protocols for different network scenarios and network densities. The scalability of our protocols was tested with networks of up to 400 nodes. The reported simulation results confirmed that our protocols provide energy efficiency and sustain good network performance in terms of data latency, fairness, and throughput. Our protocols also demonstrated improvement over existing protocols. This will enable wireless sensor networks to grow to hundreds of autonomous sensor nodes, supporting monitoring and tracking at unprecedented granularity in both time and space.

Bibliography

[1] MICA2 Mote Datasheet. Available from `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf` [1 July 2007].

[2] The Network Simulator - NS-2. Available from `http://www.isi.edu/nsnam/ns` [1 July 2007].

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, Volume 38, Number 4, pages 393–422, 2002.

[4] M. Ali and K. Langendoen. A Case for Peer-to-Peer Network Overlays in Sensor Networks. In *Proceedings of the ACM International Workshop on Wireless Sensor Network (WWSNA'07)*, April 2007.

[5] M. Ali, T. Suleman and Z. A. Uzmi. MMAC: a Mobility-Adaptive, Collision-Free MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 24th IEEE International Performance Computing and Communications Conference (IPCCC'05)*, April 2005.

[6] M. Ali and Z. A. Uzmi. CSN: A Network Protocol for Serving Dynamic Queries in Large-Scale Wireless Sensor Networks. In *Proceedings of the second IEEE Annual Conference on Communication Networks and Services Research (CNSR'04)*, May 2004.

[7] G. Anastasi, M. Conti, M. D. Francesco and A. Passarella. An Adaptive and Low-Latency Power Management Protocol for Wireless Sensor Networks. In *Proceedings of the ACM International Workshop on Mobility Management and Wireless Access (MobiWac'06)*, October 2006.

[8] G. Anastasi, A. Falchi, A. Passarella, M. Conti and E. Gregori. Performance Measurements of Motes Sensor Networks. In *Proceedings of the seventh ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04)*, October 2004.

[9] F. Araujo and L. Rodrigues. On the Monitoring Period for Fault-Tolerant Sensor Networks. Technical report, Universidade de Lisboa, 2005. Available from `http://www.di.fc.ul.pt/~ler/reports/ladc05.pdf` [1 July 2007].

[10] K. Arisha, M. Youssef and M. Younis. Energy-Aware TDMA-Based MAC for Sensor Networks. In *Proceedings of IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT'02)*, May 2002.

[11] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora and M. Miyashita. A Line in the Sand: a Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks*, Volume 46, pages 605–634, 2004.

[12] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko and D. Rus. Tracking a Moving Object with a Binary Sensor Network. In *Proceedings of the first ACM International Conference on Embedded Networked Sensor Systems (Sensys'03)*, November 2003.

[13] M. Bhardwaj, T. Garnett and A. P. Chandrakasan. Upper Bounds on the Lifetime of Sensor Networks. In *Proceedings of the third IEEE International Conference on Communications (ICC'01)*, June 2001.

[14] A. Boukerche, I. Chatzigiannakis and S. Nikoletseas. Power-Efficient Data Propagation Protocols for Wireless Sensor Networks. *SCS Journal of Simulation: Transactions of the Society for Modeling and Simulation International, Special Issue: On Modeling and Simulation of Emerging Wireless and Sensor Network Technologies and Applications*, Volume 81, Number 6, pages 399–411, 2005.

[15] A. Boukerche, X. Cheng and J. Linus. A Performance Evaluation of a Novel Energy-Aware Data-Centric Routing Algorithm in Wireless Sensor Networks. *Wireless Networks*, Volume 11, Number 5, pages 619–635, 2005.

[16] A. Boulis, C. Han and M. B. Srivastava. Design and Implementation of a Framework for Efficient and Programmable Sensor Networks. In *Proceedings of the first ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'03)*, May 2003.

[17] A. Boulis and M. B. Srivastava. Node-Level Energy Management for Sensor Networks in the Presence of Multiple Applications. In *Proceedings of the first IEEE Annual Conference on Pervasive Computing and Communications (PerCom'03)*, March 2003.

[18] C. Buragohain, D. Agrawal and S. Suri. Distributed Navigation Algorithms for Sensor Networks. In *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM'06)*, April 2006.

[19] R. Cardell-Oliver, K. Smettem, M. Kranz and K. Mayer. A Reactive Soil Moisture Sensor Network: Design and Field Evaluation. *International Journal of Distributed Sensor Networks*, Volume 1, Number 2, pages 149–162, 2005.

[20] U. Cetintemel, A. Flinders and Y. Sun. Power-Efficient Data Dissemination in Wireless Sensor Networks. In *Proceedings of the third ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'03)*, September 2003.

[21] S. Chatterjea and P. Havinga L. van Hoesel. AI-LMAC: an Adaptive, Information-Centric and Lightweight MAC Protocol for Wireless Sensor Networks. In *Proceedings of the IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP'04)*, December 2004.

[22] B. Chen, K. Jamieson, H. Balakrishnan and R. Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. In *Proceedings of the sixth Annual ACM International Conference on Mobile Computing and Networking (MobiCom'00)*, August 2000.

[23] D. Chen and P. K. Varshney. QoS Support in Wireless Sensor Networks: A Survey. In *Proceedings of the International Conference on Wireless Networks (ICWN'04)*, June 2004.

[24] F. Chen, F. Dressler and A. Heindl. End-to-End Performance Characteristics in Energy-Aware Wireless Sensor Networks. In *Proceedings of the third ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN'06)*, October 2006.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*, Chapter Section 24.3: Dijkstra's Algorithm, pages 595–601. MIT Press and McGraw-Hill, 2001.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*, Chapter Section 23.2: The Algorithms of Kruskal and Prim, pages 567–574. MIT Press and McGraw-Hill, 2001.

[27] L. H. A. Correia, D. F. Macedo, A. L. dos Santos and J. M. S. Nogueira. Issues on QoS Schemes in Wireless Sensor Networks. Technical Report RT.DCC.004/2005, DCC/UFMG, April 2005.

[28] L. H. A. Correia, D. F. Macedo, D. A. C. Silva, A. L. dos Santos, A. A. F. Loureiro and J. M. S. Nogueira. Transmission Power Control in MAC Protocols for Wireless Sensor Networks. In *Proceedings of the eighth ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'05)*, October 2005.

[29] S. Cui, A. J. Goldsmith and A. Bahai. Energy-Constrained Modulation Optimization. *IEEE Transactions on Wireless Communications*, Volume 4, Number 5, pages 2349–2360, 2005.

[30] J. Thelen D. Goense and K. Langendoen. Wireless Sensor Networks for Precise Phytophthora Decision Support. In *Proceedings of the fifth European Conference on Precision Agriculture (ECPA'05)*, June 2005.

[31] B. Deb, S. Bhatnagar and B. Nath. Wireless Sensor Networks Management. Available from `http://www.research.rutgers.edu/~bdeb/sensor_networks.html` [1 July 2007].

[32] B. Deb, S. Bhatnagar and B. Nath. A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management. Technical Report DCS-TR-441, Rutgers University, May 2001.

[33] B. Deb, S. Bhatnagar and B. Nath. STREAM: Sensor Topology Retrieval at Multiple Resolutions. *Kluwer Journal of Telecommunications Systems*, Volume 26, Number 2, pages 285–320, 2004.

[34] M. Demirbas and H. Ferhatosmanoglu. Peer-to-Peer Spatial Queries in Sensor Networks. In *Proceedings of the third International Conference on Peer-to-Peer Computing (P2P'03)*, September 2003.

[35] I. Demirkol, C. Ersoy and F. Alagoz. MAC Protocols for Wireless Sensor Networks: a Survey. *IEEE Communications Magazine*, Volume 44, Number 4, pages 115–121, April 2006.

[36] J. Ding, K. Sivalingam, R. Kashyapa and L. J. Chuan. A Multi-Layered Architecture and Protocols for Large-Scale Wireless Sensor Networks. In *Proceedings of the 58th IEEE Vehicular Technology Conference (VTC'03)*, Volume 3, pages 1443–1447, October 2003.

[37] A. El-Hoiydi and J. D. Decotignie. WiseMAC: An Ultra Low Power MAC Protocol for Multi-Hop Wireless Sensor Networks. In *Proceedings of the first Springer-Verlag International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS'04)*, July 2004.

[38] J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks. In *Proceedings of the 15th IEEE International Parallel and Distributed Processing Symposium (IPDPS'01)*, April 2001.

[39] A. Erdogan, E. Cayirci and V. Coskun. Sectoral Sweepers for Sensor Node Management and Location Estimation in Adhoc Sensor Networks. In *Proceedings of the IEEE Military Communications Conference (MILCOM'03)*, October 2003.

[40] S. C. Ergen and P. Varaiya. PEDAMACS: Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks. *IEEE Transactions on Mobile Computing*, Volume 5, Number 7, pages 920–930, 2006.

[41] D. Estrin, L. Girod, G. Pottie and M. Srivastava. Instrumenting the World with Wireless Sensor Networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'01)*, May 2001.

[42] C. Fok, G. Roman and C. Lu. Mobile Agent Middleware for Sensor Networks: An Application Case Study. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, June 2005.

[43] A. Ganesh. Publish/Subscribe Model for Querying, Sensing and Actuation in Peer-to-Peer Wireless Sensor Network. In *Proceedings of the second IEEE International Conference on Intelligent Sensing and Information Processing (ICISIP'05)*, January 2005.

[44] A. Gonzalez, I. Marshall, L. Sacks, I. Henning and T. Khan. A Self-Synchronised Scheme for Automated Communication in Wireless Sensor Networks. In *Proceedings of the IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP'04)*, December 2004.

[45] G. Gupta and M. Younis. Fault-Tolerant Clustering of Wireless Sensor Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'03)*, March 2003.

[46] A. Guttman. R-trees: a Dynamic Index Structure for Spatial Searching. In *Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD'84)*, pages 47–57, June 1984.

[47] G. Halkes and K. Langendoen. Crankshaft: An Energy-Efficient MAC-Protocol for Dense Wireless Sensor Networks. In *Proceedings of the fourth European Conference on Wireless Sensor Networks (EWSN'07)*, January 2007.

[48] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan. Building Efficient Wireless Sensor Networks with Low-Level Naming. In *Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP'01)*, October 2001.

[49] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd ACM Hawaii International Conference on System Sciences (HICSS'00)*, January 2000.

[50] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan. An Application-Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Transactions on Wireless Communications*, Volume 1, Number 4, pages 660–670, October 2002.

[51] W. R. Heinzelman, J. Kulik and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proceedings of the fifth Annual ACM International Conference on Mobile Computing and Networking (MobiCom'99)*, August 1999.

[52] B. Hohlt, L. Doherty and E. Brewer. Flexible Power Scheduling for Sensor Networks. In *Proceedings of the third International Symposium on Information Processing in Sensor Networks (IPSN'04)*, April 2004.

[53] C. Hsin and M. Liu. A Two-Phase Self-Monitoring Mechanism for Wireless Sensor Networks. *Journal of Computer Communications special issue on Sensor Networks*, Volume 29, Number 4, pages 462–476, 2006.

[54] K. Kredo II and P. Mohapatra. Medium Access Control in Wireless Sensor Networks. *Computer Networks*, Volume 51, Number 4, pages 961–994, March 2007.

[55] C. Intanagonwiwat, D. Estrin and R. Gonvindan. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. In *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems (ICDCS'01)*, April 2001.

[56] C. Intanagonwiwat, D. Estrin, R. Govindan and J. Heidemann. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. Technical Report 01-750, University of Southern California, November 2001.

[57] C. Intanagonwiwat, R. Govindan and D. Estrin. Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks. *IEEE/ACM Transactions on Networking*, Volume 11, Number 1, pages 2–16, 2003.

[58] R. Iyer and L. Kleinrock. QoS Control for Sensor Networks. In *Proceedings of the fifth IEEE International Conference on Communications (ICC'03)*, May 2003.

[59] J. Kim, K. Park, J. Shin and D. Park. Look-Ahead Scheduling for Energy-Efficiency and Low-Latency in Wireless Sensor Networks. In *Proceedings of the third ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN'06)*, October 2006.

[60] T. H. Kim and S. Hong. Sensor Network Management Protocol for State-Driven Execution Environment. In *Proceedings of the International Conference on Ubiquitous Computing (ICUC'03)*, October 2003.

[61] F. Koushanfar, M. Potkonjak and A. Sangiovanni-Vincentell. Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks. In *Proceedings of the first IEEE international conference on sensors*, June 2002.

[62] B. Krishnamachari, D. Estrin and S. B. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks. In *Proceeding of the 22nd International Conference on Communications and Mobile Computing (ICDCSW'02)*, July 2002.

[63] B. Krishnamachari, D. Estrin and S. B. Wicker. Modeling Data Centric Routing in Wireless Sensor Networks. In *Proceedings of the 21st IEEE Conference on Computer Communications (INFOCOM'02)*, June 2002.

[64] S. S. Kulkarni and M. U. Arumugam. TDMA Service for Sensor Networks. In *Proceeding of the 24th International Conference on Communications and Mobile Computing (ICDCSW'04)*, March 2004.

[65] K. Langendoen. Medium Access Control in Wireless Sensor Networks. In H. Wu and Y. Pan (editors), *Medium Access Control in Wireless Networks, Volume II: Practice and Standards*. Nova Science Publishers, Inc., 2007.

[66] K. Langendoen and G. Halkes. *Embedded Systems Handbook*, Chapter Energy-Efficient Medium Access Control. CRC Press, 2005.

[67] W. L. Lee. FlexiTP Implementation in NS-2. Available from `http://www.csse.uwa.edu.au/∼winnie/programs/flexitp` [1 July 2007].

[68] W. L. Lee. ODA Implementation in NS-2. Available from `http://www.csse.uwa.edu.au/∼winnie/programs/oda` [1 July 2007].

[69] W. L. Lee. OST Implementation in NS-2. Available from `http://www.csse.uwa.edu.au/∼winnie/programs/ost` [1 July 2007].

[70] W. L. Lee. WiseP2P Implementation in NS-2. Available from `http://www.csse.uwa.edu.au/∼winnie/programs/wisep2p` [1 July 2007].

[71] J. Li and G. Lazarou. A Bit-Map-Assisted Energy-Efficient MAC Scheme for Wireless Sensor Networks. In *Proceedings of the third International Symposium on Information Processing in Sensor Networks (IPSN'04)*, April 2004.

[72] Q. Li, M. DeRosa and D. Rus. Distributed Algorithms for Guiding Navigation Across a Sensor Network. In *Proceedings of the second International Workshop on Information Processing in Sensor Networks (IPSN'03)*, April 2003.

[73] S. Lin, J. Zhang, G. Zhou, L. Gu, T. He and J. A. Stankovic. ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks. In *Proceedings of the fourth ACM International Conference on Embedded Networked Sensor Systems (Sensys'06)*, November 2006.

[74] S. Lindsey and C. S. Raghavendra. PEGASIS: Power-Efficient Gathering in Sensor Information Systems. In *Proceedings of the third IEEE International Conference on Communications (ICC'01)*, June 2001.

[75] W. Liu, Y. Zhang, W. Lou and Y. Fang. Managing Wireless Sensor Network with Supply Chain Strategy. In *Proceedings of the IEEE International Conference*

on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE'04), October 2004.

[76] S. Madden, J. Hellerstein and W. Hong. I-LENSE Topology Generator (topo_gen). Available from `http://www.isi.edu/ilense/software/topo_gen/topo_gen.html` [1 July 2007].

[77] S. Madden, J. Hellerstein and W. Hong. TinyDB: In-Network Query Processing in TinyOS. Available from `http://telegraph.cs.berkeley.edu/tinydb/tinydb.pdf` [1 July 2007].

[78] S. Madden, R. Szewczyk, M. J. Franklin and D. Culler. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. In *Proceedings of the first ACM international Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, June 2002.

[79] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proceedings of the first ACM international Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, September 2002.

[80] A. Manjeshwar and D. P. Agrawal. TEEN: a Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks. In *Proceedings of the 15th IEEE International Parallel and Distributed Processing Symposium (IPDPS'01)*, April 2001.

[81] M. Maroti, B. Kusy, G. Simon and A. Ledeczi. Robust Multi-Hop Time Synchronization in Sensor Networks. In *Proceedings of the International Conference on Wireless Networks (ICWN'04)*, June 2004.

[82] P. J. Marrón, A. Lachenmann, D. Minder, M. Gauger, O. Saukh and K. Rothermel. Management and Configuration Issues for Sensor Networks. *International Journal of Network Management*, Volume 15, Number 4, pages 235–253, 2005.

[83] M. J. Miller and N. H. Vaidya. A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio. *IEEE Transactions on Mobile Computing*, Volume 4, Number 3, pages 228–242, 2005.

[84] G. Pei and C. Chien. Low Power TDMA in Large Wireless Sensor Networks. In *Proceedings of the IEEE Military Communications Conference (MILCOM'01)*, October 2001.

[85] M. Perillo and W.B. Heinzelman. Providing Application QoS through Intelligent Sensor Management. In *Proceedings of the first IEEE Workshop on Sensor Network Protocols and Applications (SNPA'03)*, May 2003.

[86] J. Polastre, J. Hill and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the second ACM International Conference on Embedded Networked Sensor Systems (Sensys'04)*, November 2004.

[87] I. Raicu, L. Schwiebert, S. Fowler and S.K.S. Gupta. Local Load Balancing for Globally Efficient Routing in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, Volume 1, Number 2, pages 163–185, 2005.

[88] V. Rajendran, J. Garcia-Luna-Aceves and K. Obraczka. Energy-Efficient, Application-Aware Medium Access for Sensor Networks. In *Proceedings of the second IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'05)*, November 2005.

[89] V. Rajendran, K. Obraczka and J. J. Garcia-Luna-Aceves. Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks. In *Proceedings of the first ACM International Conference on Embedded Networked Sensor Systems (Sensys'03)*, March 2003.

[90] N. Ramanathan, E. Kohler and D. Estrin. Towards a Debugging System for Sensor Networks. *International Journal of Network Management*, Volume 15, Number 4, pages 223–234, 2005.

[91] N. Ramanathan, M. Yarvis, J. Chhabra, N. Kushalnagar, L. Krishnamurthy and D. Estrin. A Stream-Oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications. In *Proceedings of the second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, May 2005.

[92] T. S. Rappaport. *Wireless Communications: Principles and Practice*. IEEE Press, Piscataway, NJ, USA, 1996.

[93] I. Rhee. Z-MAC: Hybrid MAC for Wireless Sensor Networks. Available from `http://www.csc.ncsu.edu/faculty/rhee/export/zmac/software/zmac/zmac.htm` [1 July 2007].

[94] I. Rhee, A. Warrier, M. Aia and J. Min. Z-MAC: a Hybrid MAC for Wireless Sensor Networks. In *Proceedings of the third ACM International Conference on Embedded Networked Sensor Systems (Sensys'03)*, November 2005.

[95] I. Rhee, A. Warrier, J. Min and L. Xu. DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad-Hoc Networks. In *Proceedings of the seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'06)*, May 2006.

[96] L. B. Ruiz, J. M. Nogueira and A. A. F. Loureiro. MANNA: A Management Architecture for Wireless Sensor Networks. *IEEE Communications Magazine*, Volume 41, Number 2, pages 116–125, 2003.

[97] L. B. Ruiz, I. G. Siqueira, L. B. e Oliveira, H. C. Wong, J. M. S. Nogueira and A. A. F. Loureiro. Fault Management in Event-Driven Wireless Sensor Networks. In *Proceedings of the seventh ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04)*, October 2004.

[98] A. Safwat, H. Hassanein and H. Mouftah. ECPS and E2LA: New Paradigms for Energy Efficiency in Wireless Ad Hoc and Sensor Networks. In *Proceedings of the 46th Annual IEEE Global Telecommunications Conference (GLOBECOM'03)*, December 2003.

[99] S.Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, Volume 36, Number 4, pages 335–371, December 2004.

[100] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava. Optimizing Sensor Networks in the Energy-Latency-Density Design Space. *IEEE Transactions on Mobile Computing*, Volume 1, Number 1, pages 70–80, January 2002.

[101] L. Schwiebert, V. Annamalai and S. K. S. Gupta. On Tree-Based Convergecasting in Wireless Sensor Networks. In *Proceedings of the IEEE Wireless Communictions and Networking Conference (WCNC'03)*, March 2003.

[102] K. Sethom and H. Afifi. A New Service Discovery Architecture for Sensor Networks. In *Proceedings of the fourth IEEE Annual Wireless Telecommunication Symposium (WTS'05)*, April 2005.

[103] M. A. Sharaf, J. Beaver, A. Labrinidis and P. K. Chrysanthis. TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation. In *Proceedings of the third International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'03)*, September 2003.

[104] M. L. Sichitiu. Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks. In *Proceedings of the 23rd IEEE Conference on Computer Communications (INFOCOM'04)*, March 2004.

[105] M. L. Sichitiu and C. Veerarittiphan. Simple, Accurate Time Synchronization for Wireless Sensor Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'03)*, March 2003.

[106] I. Solis and K. Obraczka. In-Network Aggregation Trade-offs for Data Collection in Wireless Sensor Networks. *International Journal of Sensor Networks*, Volume 1, Number 2, pages 70–80, 2006.

[107] H. Song, D. Kim, K. Lee and J. Sung. Upnp-Based Sensor Network Management Architecture. In *Proceedings of the second International Conference on Mobile Computing and Ubiquitous Networking (ICMU'05)*, April 2005.

[108] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan. Chord: a Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM'01)*, August 2001.

[109] G. Tolle and D. Culler. Design of an Application-Cooperative Management System for Wireless Sensor Networks. In *Proceedings of the second European Conference on Wireless Sensor Networks (EWSN'05)*, February 2005.

[110] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay and W. Hong. A Macroscope in the Redwoods. In *Proceedings of the third ACM International Conference on Embedded Networked Sensor Systems (Sensys'05)*, November 2005.

[111] Z. Tu and W. Liang. Energy-Efficient Aggregate Query Evaluation in Sensor Networks. In *Proceedings of the International Conference on Mobile Ad-hoc and Sensor Networks (MSN'05)*, December 2005.

[112] M. Turon. MOTE-VIEW: A Sensor Network Monitoring and Management Tool. In *Proceedings of the second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, May 2005.

[113] R. Tynan, D. Marsh, D. O'Kane and G. M. P. O'Hare. Agents for Wireless Sensor Network Power Management. In *Proceedings of the IEEE International Conference on Parallel Processing Workshops (ICPPW'05)*, June 2005.

[114] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the first ACM International Conference on Embedded Networked Sensor Systems (Sensys'03)*, March 2003.

[115] L. F. W. van Hoesel and P. J. M. Havinga. A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks. In *Proceedings of the first International Conference on Networked Sensing Systems (INSS'04)*, June 2004.

[116] L. F. W. van Hoesel, T. Nieberg, H. J. Kip and P. J. M. Havinga. Advantages of a TDMA based, Energy-Efficient, Self-Organizing MAC Protocol for WSNs. In *Proceedings of the 59th IEEE Vehicular Technology Conference (VTC'04)*, May 2004.

[117] C-Y. Wan, S. B. Eisenman, A. T. Campbell and J. Crowcrof. Siphon: Overload Traffic Management using Multi-radio Virtual Sinks in Sensor Networks. In *Proceedings of the third ACM International Conference on Embedded Networked Sensor Systems (Sensys'05)*, November 2005.

[118] E. W. Weisstein. Lagrange Interpolating Polynomial. Available from `http://mathworld.wolfram.com/LagrangeInterpolating Polynomial.html` [1 July 2007].

[119] M. Welsh and G. Mainland. Programming Sensor Networks Using Abstract Regions. In *Proceedings of the first USENIX Symposium on Networked Systems Design and Implementation (NSDI'04)*, March 2004.

[120] G. Xing, C. Lu, Y. Zhang, Q. Huang and R. Pless. Minimum Power Configuration in Wireless Sensor Networks. In *Proceedings of the sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, May 2005.

[121] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan and D. Estrin. A Wireless Sensor Network for Structural Monitoring. In *Proceedings of the second ACM International Conference on Embedded Networked Sensor Systems (Sensys'04)*, November 2004.

[122] W. Ye. Energy Model Update in NS-2. Available from `http://www.isi.edu/ilense/software/smac/ns2_energy.html` [1 July 2007].

[123] W. Ye, J. Heidemann and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 21st IEEE Conference on Computer Communications (INFOCOM'02)*, June 2002.

[124] W. Ye, F. Silva and J. Heidemann. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling. In *Proceedings of the fourth ACM International Conference on Embedded Networked Sensor Systems (Sensys'06)*, November 2006.

[125] Y. Yuan, Z. Yang, Z. He and J. He. An Integrated Energy Aware Wireless Transmission System for QoS Provisioning in Wireless Sensor Network. *Elsevier Computer Communications (Article in Press)*, May 2005.

[126] J. Zhang, E. C. Kulasekere, K. Premaratne and P. H. Bauer. Resource Management of Task Oriented Distributed Sensor Networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'01)*, May 2001.

[127] J. Zhao, R. Govindan and D. Estrin. Computing Aggregates for Monitoring Wireless Sensor Networks. In *Proceedings of the first IEEE Workshop on Sensor Network Protocols and Applications (SNPA'03)*, May 2003.

[128] T. Zheng, S. Radhakrishnan and V. Sarangan. PMAC: an Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, April 2005.