Stack Architectures and Protocols for Emerging Wireless Networks

by

Chen-Hsiang Feng

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Wendi B. Heinzelman

Department of Electrical and Computer Engineering

Arts, Sciences and Engineering

Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester

Rochester, New York

2013

Biographical Sketch

The author received his Bachelor of Science degree in Electrical Engineering from Feng Chia University in 2004. He attended the Computer Science Department at Chung Hsiang University, Taichung, Taiwan from 2004 to 2006 and graduated with a Master of Science degree in Computer Science in 2006. He began graduate studies in the Department of Electrical and Computer Engineering at the University of Rochester in 2007 and received the Master of Science degree from the University of Rochester in 2011. He has worked with Intel Corporation since 2011.

The following publications were a result of work conducted during doctoral study:

Chen-Hsiang Feng, Ilker Demirkol, and Wendi B. Heinzelman, "*Enabling Heterogeneous Devices using Virtual Interfaces*," In Submission, 2013

Chen-Hsiang Feng, Yuqun Zhang, Ilker Demirkol, and Wendi B. Heinzelman, "*Stateless Multicast Protocol for Ad-Hoc Networks*," IEEE Transactions on Mobile Computing, 2012.

Chen-Hsiang Feng, Ilker Demirkol, and Wendi B. Heinzelman, "*UPS: Universal Protocol Stack for emerging wireless networks*," Elsevier Ad-Hoc Networks, 2010.

Yuqun Zhang, **Chen-Hsiang Feng**, Ilker Demirkol, and Wendi B. Heinzelman "*Energy-Efficient Duty Cycle Assignment for Receiver-Based Convergecast in Wireless Sensor Networks*," Proceedings of IEEE GLOBECOM 2010.

Chen-Hsiang Feng, Ilker Demirkol, Wendi B. Heinzelman, "*UPS: Unified Protocol Stack for wireless sensor networks*," Proceedings of Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, 2009.

Chen-Hsiang Feng and Wendi B. Heinzelman, "*RBMulticast: Receiver Based Multicast for Wireless Sensor Networks*," IEEE WCNC 2009.

Christophe J. Merlin, **Chen-Hsiang Feng**, and Wendi B. Heinzelman, "*Information-Sharing Architectures for Sensor Networks: the State of the Art*," SIGMOBILE Mobile Computing and Communication Review, 2009.

Acknowledgements

I would like to thank my advisor, Professor Wendi Heinzelman, for the opportunity to work with her over the past six years. I thank her for all the energy and time she has spent for me. Her incredible patience and support have motivated me to believe in myself and progress towards my doctorate degree.

I would also like to express my thanks to Professors Mark Bocko and Jefferey Bigham for acting as members of my thesis committee.

I would also like to thank to all of my colleagues at the University of Rochester in the Wireless Communications and Networking Group. Specifically, I would like to thank Yuqun Zhang. It was a pleasure to collaborate with him, and he has been a great friend to me over the years. I would also like to express my thanks to my colleagues from Intel Labs, Xiaocheng Zhou and Shoumeng Yan. They are terrific career role models, and their encouragement has kept me going though the difficult times.

I owe my thanks to my parents, for their strong support of my education and my move to America. Most of all, thanks to my wife, Shio-Chen, for her love and for giving me all the happiness as I finished my degree.

Abstract

Recent devices developed for emerging wireless networks, such as 4G cellular networks, wireless mesh networks, and mobile ad hoc networks, support multiple communication substrates and require execution of multiple protocols within a layer, which cannot be supported efficiently by traditional layered protocol stack approaches. Our goal in this thesis is to discover the minimal set of requirements for simultaneously supporting the use of multiple protocols in the same stack layer without requiring modifications of the protocols and retaining that the modularity of the stack architecture so that future protocols can easily be incorporated.

To achieve this goal, we propose Universal Protocol Stack (UPS), which provides support for the execution of multiple protocols within a layer simultaneously in a modular way through packet-switching, information-sharing, and memory management. The implementation and simulations of UPS show that the overhead incurred to implement UPS is very low, and little or no modifications are required to adapt existing protocols to the UPS framework, yet there is benefit to the application in terms of reduced traffic or reduced delay/energy. As an example, we develop an approach to support multiple radio interfaces by abstracting all the available interfaces using a single virtual interface within the UPS framework. The selection of the specific physical interface to use per packet is done by the virtual interface, thus ensuring that no modifications of the upper layer protocols are required. This provides the opportunity for algorithms at the virtual interface to optimize the selection of the physical interface to improve the network performance. Results from simulations show that the use of a virtual interface is feasible and can improve the network performance.

While new protocol stack architectures are important to support multiple protocols and communication interfaces, efficient protocols are equally important to support emerging networks. We propose a stateless receiver-based multicast protocol, called RBMulticast (Receiver-Based Multicast), which removes the need for costly multicast tree and neighbor table maintenance, yet provides high success rates and low delay. This makes RBMulticast an excellent choice for multicasting in dynamic networks, where state maintenance is costly. Additionally, using the idea of receiver-based routing for convergecast transmissions, we find the duty cycle of a node as a function of its distance to the sink to minimize the expected energy dissipation.

Contributors and Funding Sources

This work was supervised by a dissertation committee consisting of Professors Wendi Heinzelman (advisor) and Mark Bocko of the Department of Electrical and Computer Engineering and Professor Jefferey Bigham of the Department of Computer Sciences. The X-Lisa architecture described in Chapter 3 was developed by Dr. Christophe J. Merlin. The simulations described in Chapters 6 and 7 were conducted in part by Yuqun Zhang and were published in 2010 and 2012. All other work conducted for the dissertation was completed by the student independently. Graduate study was supported in part by the National Science Foundation under grant #CNS-0448046.

Contents

List of Tables **x**

List of Figures **xi**

1 Introduction **1**

1.1 Motivation for a New Protocol Stack Architecture 2

1.2 Existing Protocol Stack Architectures 3

1.3 Motivation for Receiver-Based Protocols 5

1.4 Research Contributions 5

1.5 Thesis Organization 7

2 Related Work **8**

2.1 Protocol Stacks for Multiple Protocols 8

2.2 Receiver-Based Routing 10

2.3 Multicast Routing 11

2.4 Duty Cycle Control 12

3 Evaluation of the X-Lisa Cross-Layer Protocol Architecture **14**

3.1 X-Lisa: Cross-Layer Information-Sharing Architecture 15

3.2 XLM: A Fused-Layer Protocol for Wireless Sensor Networks 16

3.3 XLM/X-Lisa: Example of Layer Fusion Decomposition 18

3.3.1 Layer Implementation 18

3.3.2 Information Exchange with X-Lisa 20

3.3.3 Input/Output Packet Flow 21

3.4 XLM/LPL: Example of Protocol Swapping 23

3.4.1 Protocol Swapping Implementation 23
3.5 Conclusions 25

4 UPS: Universal Protocol Stack for Emerging Wireless Networks 26
4.1 Introduction 27
4.2 UPS Framework 31
4.2.1 UPS Protocol Interface (*UPS-PI*) 34
4.2.2 UPS Message Pool and Message Pool Interface (*UPS-MPI*) . . 36
4.2.3 UPS Information-Sharing Interface (*UPS-ISI*) 38
4.2.4 Power Manager 42
4.3 UPS Architecture Case Studies 43
4.3.1 TinyOS Experiments for UPS with Wireless Sensor Networks . 43
4.3.2 Simulation Results for UPS with MANETs 50
4.3.3 Notes on Implementing UPS for Off-the-Shelf Protocols 53
4.4 Conclusions 53

5 Enabling Heterogeneous Devices using Virtual Interfaces 58
5.1 Introduction 58
5.2 The Virtual Interface 61
5.2.1 Internet Stack 61
5.2.2 Sending Packets 61
5.2.3 Receiving Packets 62
5.3 One vs. Multiple Interfaces 63
5.4 Weighted Random Interface Selection 64
5.5 Adaptive Random Interface Selection 66
5.6 Interface Selection while Using TCP 69
5.7 Limitations 71
5.8 Machine Learning Interface Selection 72
5.9 Conclusions 73
5.10 Future Work 73

6 Stateless Multicast Protocol for Dynamic Networks 74
6.1 Introduction 75
6.2 RBMulticast Protocol Description 77

6.2.1 RBMulticast Overview 78
6.2.2 Multicast Regions 80
6.2.3 Packet Splitting 80
6.2.4 Virtual Node 82
6.2.5 RBMulticast Header 84
6.2.6 Group Management 85
6.2.7 Summary 85
6.3 Analytical Bounds on Average Hop Count Performances 86
6.3.1 RBMulticast Performance 86
6.3.2 Analysis for Using Multiple Unicast as Multicast 91
6.3.3 Validation of Hop Distance Analysis and Performance Comparisons 94
6.4 RBMulticast Performance Evaluation 95
6.4.1 Outline of Implemented MAC Unicast Protocol 95
6.4.2 Split Packet Contention Problem in RBMulticast and the Proposed Solution 96
6.4.3 MAC Level Improvements 97
6.4.4 Simulation Results 98
6.5 Tmote Sky Implementation 108
6.6 Conclusions 110

7 Energy-Efficient Duty Cycle Assignment for Receiver-Based Convergecast 112
7.1 Introduction 113
7.2 Distance-based Duty Cycle Assignment Methods 115
7.2.1 Traffic Rate Analysis 115
7.2.2 Duty Cycle for a Given Expected Traffic Rate 117
7.2.3 Duty Cycle Assignment Methods Proposed 120
7.3 Performance Evaluation of Duty Cycle Assignment 121
7.4 Conclusions 124

8 Conclusions and Future Work 125
8.1 Conclusions 125
8.2 Future Work 126

8.2.1 Routing Layer Protocol Selection 126
8.2.2 Cross-Layer Protocol Selection 127

9 Bibliography 128

List of Tables

1.1 Comparison of emerging networks. 2

3.1 Simulation results of XLM/LPL/X-Lisa and XLM/X-Lisa. 24

4.1 Selected information-sharing protocols with their shared information and optimization goals. For more generalized classification, see [1, Table 1]. 39

4.2 The overhead of context switches in TinyOS (and hence UPS) and of sending a 36 byte packet using the Tmote Sky motes. This table is a modified version from P. Levis [2, Fig. 10]. Packet time for sending a 36 B packet is measured from our experiments. 49

4.3 Comparison of execution code size. 50

6.1 Definition of symbols in the multicast analysis. 88

6.2 Packet Delivery Ratio in terms of SPTI. 96

List of Figures

3.1 X-Lisa architecture. 15

3.2 A neighbor table is kept at every node i with information about itself and each of its neighbors j. 16

3.3 The original XLM (a), was broken into a layered scheme (XLM / X-Lisa) (b), and its MAC layer was replaced (c). Arrows show packet exchanges between layers, and squares show information exchanges. . 19

3.4 The flow graph for sending a packet. The arrows of β, λ, and d indicate the packet buffer usage, packet rate, and distance information exchanged through the X-Lisa interfaces. 21

3.5 The packet flow graph of a node acting as the receiver node. The arrows of β, λ, and d indicate the packet buffer usage, packet rate, and distance information exchanged through the X-Lisa interfaces. 22

3.6 The source node is at (0,0) and the sink node is at (60,60). Radio range is 30m. 24

4.1 An example of the high level system block diagram of the UPS framework. 29

4.2 Packet header format of XLM/Net and RBMulticast packets. 33

4.3 Packet flow diagram example for the UPS framework. 35

4.4 The internal packet structure of *UPS Message Pool*. The first memory block stores the packet header, and the packet data payload starts from the second block. This is a snapshot of an output packet inside the XLM/Net module. XLM/Net will be illustrated in Section 4.3.1. 37

4.5 Block diagram of the UPS framework and protocol modules developed for the WSN environment. 44

4.6 a) The experimental network in this chapter. The dotted node is the unicast destination, and the shaded nodes are multicast receivers. b) MC: Multicast only, c) MU: Mixed Multicast-Unicast, and d) UC: Unicast only. 45
4.7 Statistics of the experimental network in different scenarios. 47
4.8 a) Packet success rate. b) Total number of packets sent by all nodes. This number counts all MAC layer packets (e.g., RTS/CTS/DATA/ACK). Packet delay is not considered here because it is difficult to calculate time difference without synchronization among sensor nodes. 55
4.9 The simulation scenario of a mixed static and mobile network. 56
4.10 The high level system block diagram of the UPS stack consisting of the TCP/IP and AODV ad hoc routing protocols. 56
4.11 The performance comparison of the three protocol stacks for a) average packet success rate, and b) average traffic load of the network. 57

5.1 Virtual interface block diagram. 60
5.2 Success rate using three interfaces compared with one interface. WiFi is configured with OFDM rate 6 Mbps and CSMA is configured with channel bandwidth 5 Mbps and delay 2 ms. 63
5.3 Delay using three interfaces compared with one interface. 64
5.4 Success rate and delay using three interfaces and the weighted random selection method compared with the random selection method. 65
5.5 Additional traffic to saturate the WiFi channel. 68
5.6 Success rate and delay using the adaptive random, weighted random and uniform random client selection methods. 68
5.7 Success rate of the different interface selection algorithms using TCP. . 70
5.8 Delay of the different interface selection algorithms using TCP. 70
5.9 Situation showing the need for all physical interfaces to have the same MAC address. 72

6.1 Example showing how RBMulticast delivers multicast packets. The source node is the square node. Multicast members are shaded circles, and virtual nodes are dotted circles. Because every destination node will become a virtual node at the end, they are all shown with dotted circles. The number on the side of the lines indicate the destination of that packet. 82
6.2 Two possible ways to divide a space into multicast regions: a) dividing the space into four quadrants, and b) dividing the space into three 120° regions. c) demonstrates how to choose a next hop node. The solid node is the source node, and the gray nodes are the multicast members. The solid line is the route when choosing a target node near the geographic mean of the multicast members, and the dotted line is the route when choosing a target node close to the nearest multicast member. We can see that the longest distance is two hops distance in the first case, and it is three hops distance in the second case. 83
6.3 Packet header of the RBMulticast protocol. 84
6.4 The region considered in the multiple unicast analysis. 91
6.5 Performance comparisons for RBMulticast with perfect location information and no packet collisions. 93
6.6 Matlab simulation results of total number of hops with different number of sink nodes M for the original version of RBMulticast. 95
6.7 Adjustment of $timeout_{CTS}$ for MAC improvement. 97
6.8 Multicast member (sink) locations in the simulation. The additional members align along the boundaries for different simulations. Note that this figure shows a blow-up of the north-east quadrant of the simulation area. 99
6.9 Performance comparisons for RBMulticast: static scenario, 5 sinks. . . 101
6.10 Performance comparisons for RBMulticast: mobile scenario, 5 sinks. . . 102
6.11 Performance comparisons for RBMulticast: mobile scenario, varying number of sinks. 104
6.12 Packet delivery ratio vs. data generate rate. (mobile scenario with speed 10 m/s, 5 sinks). 105

6.13 Performance comparisons for RBMulticast: mobile scenario, uniformly distributed sinks. 107
6.14 Packet delivery ratio with location estimation error of $\mathcal{N}(0, \sigma^2)$: 5 uniformly distributed sinks. 108
6.15 The experimental network for comparing results in the Tmote Sky implementation and the TOSSIM simulations. Node 0 is the source node and the shaded nodes are multicast destination nodes. 109
6.16 Comparison of Tmote Sky implementation to TOSSIM simulations. . . 110

7.1 Sample network topology. 116
7.2 Representation of packet exchange durations. 119
7.3 Simulation results in terms of the number of sources. 122
7.4 Simulation results in terms of source packet generation rate λ_g. 123

Chapter 1

Introduction

Wireless network applications are becoming increasingly popular with the advancement of semiconductor technology leading to affordable mobile platforms. Entirely new networks and application areas, such as vehicular networks [3], wireless sensor networks [4], mesh networks [5], body area networks [6], and smart grid networks [7], have been launched because of the advances in speed, reliability, and low power operation of wireless devices. This has led to pervasive wireless deployments, providing users with Internet access or phone services virtually anywhere in developed areas. However, the current protocol architectures, which include the traditional layered approach (e.g., the OSI or TCP/IP layered protocol stacks), cross-layer information-sharing, and layer fusion architectures, do not fully utilize the advancement of wireless technology and cannot provide adequate support to these emerging networks and applications.

Additionally, advances are needed in protocols that support the network dynamics often encountered in emerging networks, due to node mobility, node duty-cycling for energy savings, and time-varying wireless channels. This thesis addresses these challenges, by proposing a protocol stack architecture and efficient protocols to best support these emerging networks.

Table 1.1: Comparison of emerging networks.

	4G	Wireless Mesh Networks (WMNs)	Multiple MAC
# of radios	Multiple (heterogeneous)	Multiple (homogeneous)	Single
Switching	Network	Radio/Channel	MAC
Switching frequency	Low (high penalty)	Rare (static network)	High (packet based)
Switching objective	Always best connected	Increase bandwidth	Always best connected

1.1 Motivation for a New Protocol Stack Architecture

The evolution of communication technology has gone from wired communication to wireless communication and from infrastructure-based communication to infrastructureless communication. Emerging wireless networks bring new challenges due to the distinct capabilities of the devices and the specific requirements of the corresponding applications.

For example, 4G network studies and standardization works target the integration of different radio access techniques into a common network. The concept of Always Best Connected (ABC) [8] aims to provide the best connectivity to applications anywhere and at anytime. Users and mobile devices can choose the best available access networks from 2G, 3G, WLAN, WMAN, etc., in a way that best suits their needs. To achieve this goal, the protocol stack architecture has to support mechanisms to recognize all possible incoming and outgoing traffic.

Several wireless mesh network (WMN) [9] studies assume routers have the ability to switch among multiple radios. Routers can be inter-connected wirelessly to form a network to replace the wired infrastructure network in rural areas. Similarly, multi-MAC networks [10] allow multiple MAC protocols sharing a single radio to increase the one-hop bandwidth to a neighbor node. The idea behind such systems is that the MAC protocol can be selected out of a set of MAC protocols at run time according to real time network conditions. To achieve this goal, the protocol stack architecture has to support fast interface switching on multiple radio devices on a per-packet basis.

In Table 1.1, we list the properties of these emerging networks that lead to the need for a new protocol stack architecture that supports multiple protocols and information-sharing. This table shows that these emerging networks either require switching between multiple radios, or switching between multiple communication protocols, or both. Thus, there is a need to investigate a new cross-layering scheme that has unified support for emerging wireless networks with multiple communication substrates.

1.2 Existing Protocol Stack Architectures

A communication protocol stack is placed within both the software and the hardware of a system. Typically, the physical layer and the MAC layer are within the hardware. The remaining protocol layers are implemented within the software, including the routing and transport protocols, along with intermediate middleware and platform specific modules, which are programmed in the kernel of operating systems such as Linux or TinyOS, the operating system designed for wireless sensor networks [11]. In order to support the communication protocols, system services are often required, such as services for domain name resolution (DNS), or services monitoring the residual energy and link quality of the node's neighbors, which may be necessary for making routing decisions.

Recent wireless devices, such as cellular phones, tablet computers, and laptop computers, are designed to support multiple radio communication substrates, including GSM, CDMA, WiFi, Bluetooth, or ZigBee. A common method to enhance the connectivity of current multi-radio devices is to provide an ordered list of networks to access. For example, when a user is within WiFi coverage, the multi-radio device uses WiFi for connectivity, and only if WiFi coverage is not available, then the device switches to a cellular network [12]. Such methods view the network and channel conditions in a very simplistic way, and they do not consider the QoS requirements of different applications, such as delay or throughput. Furthermore, such methods and supporting architectures are network and framework specific and thus are limited to predefined networks and devices.

The traditional layered protocol stack approach does not offer support for multiple protocols in a single layer; therefore the protocol designer has to make some appropriate modifications of the stack (e.g., middleware, or off-flow of the regular packet path) in order to support multiple MAC/PHY protocols or multiple routing protocols. In order to support different networks, researchers today mostly focus on supporting handover between specific networks, and only Layer 1 criteria are taken into consideration for the handover decision [13].

At the other extreme of the layered protocol stack are cross-layer protocols. These protocols are often designed for a specific network to improve the lifetime or end-to-end quality of service (QoS) provided to the application. There are two types of cross-layer protocols: ones that share information among several non-adjacent layers (information-sharing), and ones that combine two or more layers into a single, integrated layer (layer fusion). Both of these cross-layer designs let protocols cooperate and share information to best support the application and the specific network deployment, but they require custom designs and cannot easily support multiple protocols simultaneously. The lack of modularity also makes these cross-layer designs inflexible.

What is needed, instead, is an architecture that enables stack-level support for heterogeneous networks, as well as the protocols and algorithms that allow nodes to make smart decisions regarding the use of multiple heterogeneous networks. Eventually, through the merging at the individual devices, and with the introduction of techniques to enable cross-layer/cross-node interactions, we can provide benefit to all devices in the co-located networks. This will have the following advantages: i) more efficient use of scarce network resources such as bandwidth and device energy, ii) providing seamless network connectivity, and iii) enabling the optimization of different application QoS objectives, such as low delay.

In order to support not only current but also emerging and future wireless networks and protocols, a protocol stack should be flexible, easy to use and to update, and simple. Flexibility is important, as new protocols must be able to operate within existing networks, and the amount of work necessary to migrate protocols into any new stack framework should be low. Thus, a good protocol stack should be designed to support protocol swapping. In order to ensure fast adoption, the protocol stack should be simple enough to make it easy to use and update. We believe that these goals can be met through a layered protocol stack that provides unified access to its data structures and does not include hidden operations.

1.3 Motivation for Receiver-Based Protocols

Dynamic networks are common in the pervasive wireless deployments of emerging networks. The source of these dynamics varies for different applications and different networks. For example, node mobility is one cause of network dynamics, imposing restrictions on the routing protocol. Frequent route requests caused by topology changes can result in a considerable amount of overhead and will affect the performance of the network. Commonly used duty-cycling (sleep-awake cycling of the nodes) in wireless sensor networks also contributes to the dynamic nature of the networks. Sensor nodes may temporally go off-line to save energy, and hence all nodes need to repeatedly poll the neighbors to avoid transmission failures and excessive delays.

All of these issues arise due to the attempt to maintain updated network information in dynamic networks. In proactive and reactive protocols, decisions about routing and medium access control are made by the sender, using whatever state information the sender has available, which may be stale in dynamic wireless networks. Receiver-based protocols represent a fundamentally different way of making decisions, where the potential receivers of a packet make decisions in a distributed manner, rather than the sender making decisions. Thus, the current spatio-temporal neighborhoods participate in the decision making processes to avoid frequent updates of information necessitated by traditional sender-based protocols.

As we will show in our proposed Receiver-Based Multicast (RBMulticast) protocol, by ensuring that neighbors that make the most forward progress to the destination have a higher priority to become the next hop node, we are able to remove the need for costly state maintenance (e.g., tree/mesh/neighbor table maintenance). This is a crucial property, making receiver-based protocols ideally suited for dynamic networks.

1.4 Research Contributions

This thesis aims to understand the issues associated with wireless communication protocol stack design and to develop a protocol stack and associated protocols that can better support current and emerging networks that require 1) unified access to data structures, 2) support for multiple protocols in the same stack layer, and 3) efficient protocols that support network dynamics. The specific contributions of my work include the following:

- The concept and benefits/limitations of a traditional layered approach and of cross-layer interactions are studied, and the requirements of new emerging networks are discussed, leading to a set of goals for a new protocol stack architecture [14].

- As part of the work in [14], a single layer cross-layer protocol XLM [15] is implemented into the X-Lisa information-sharing protocol stack architecture. The advantage of modularized protocol design through switching the MAC protocol is demonstrated through swapping the existing XLM MAC with a new Low Power Listening MAC protocol called XLM/LPL. Simulation results shows that X-Lisa can be successfully used as a flexible information sharing stack protocol, but it lacks the ability to support multiple protocols in the same layer simultaneously, and it does not provide a universal information-sharing interface.

- A new protocol stack architecture called UPS (Universal Protocol Stack) is proposed [16]. UPS standardizes a modular protocol stack that supports concurrent protocol operation and information sharing. This architecture offers a new perspective on protocol selection by implementing protocol switches between layers. Each protocol thus focus on its core task, and the information sharing and packet path through the protocol stack are handled by the stack.

- The use cases of UPS are demonstrated through simulations of mixed mobile and static networks as well as through a TinyOS sensor network implementation of UPS [16]. The results demonstrate that the UPS framework can be applied to existing protocols with no interference between different protocol modules in the same layer.

- A virtual interface approach is proposed that abstracts all the available interfaces using a single virtual interface. No modifications to the layer 3 (L3) routing protocol and above layers are required, and packets can be seamlessly transmitted from any of the available interfaces. This provides the opportunity for performing smart physical interface selection at the virtual interface to improve the network performance. Results from these simulations show that the use of a virtual interface can improve the network performance.

- A stateless receiver-based multicast protocol is developed that simply uses a list of the multicast members' (e.g., sinks') addresses, embedded in packet headers, to enable receivers to decide the best way to forward the multicast traffic [17, 18]. This protocol, called RBMulticast (Receiver-Based Multicast), exploits the knowledge of the geographic locations of the nodes obtained through UPS information sharing to remove the need for costly state maintenance (e.g., tree/mesh/neighbor table maintenance), which makes RBMulticast ideally suited for multicasting in dynamic networks.
- An adaptation method for distance-based duty cycling is proposed for receiver-based convergecast networks [19]. Based on local observed traffic, we derive a closed-form formula for the duty cycle that minimizes the expected energy consumption at a given distance while ensuring packet delivery remains high. Performance evaluations of the proposed duty cycle assignment method show that it greatly improves the energy efficiency without sacrificing packet delivery ratio or delay significantly.

1.5 Thesis Organization

This thesis is organized as follows: Chapter 2 elaborates on the related work in the area of protocol stacks, receiver-based routing, multicast routing, and node duty cycling. Chapter 3 evaluates the X-Lisa information-sharing architecture with the implementation of a layer-fusion protocol XLM. Chapter 4 proposes a universal protocol stack, UPS, for emerging wireless networks. Chapter 5 proposes a virtual interface to enable heterogeneous devices in UPS. Chapter 6 proposes a stateless multicast routing protocol, RBMulticast, for dynamic networks. Similarly, Chapter 7 analyzes and optimizes receiver-based convergecast with an energy-efficient duty cycle schedule assignment. Chapter 8 finally concludes the thesis and provides thoughts on future research in this area.

Chapter 2

Related Work

This chapter provides background on the issues addressed by this thesis. We survey the literature related to cross-layer designs, receiver-based protocols, multicast routing, and duty cycle control.

2.1 Protocol Stacks for Multiple Protocols

Due to the inadequacies of the traditional OSI and TCP/IP stack approaches, there have been several efforts to revise these stack structures. The *x*-Kernel protocol architecture proposed in [20] defines a uniform set of abstractions for encapsulating protocols. The dynamic network architecture described in [21], built on *x*-Kernel, replaces the layered approach by a complex protocol graph, where a protocol can have interfaces to multiple other protocols. Although the protocol graph idea and the set of abstractions provide similar functionality to the UPS Protocol Interface approach we propose in Chapter 4, there is no information-sharing interface or unified packet access mechanism defined in [21]. Moreover, an important contribution of this thesis work is the investigation of the execution of multiple protocols simultaneously, which is not covered in either [20] or [21].

The Recursive Network Architecture (RNA) [22] defines dynamic protocol stacks that are recursively adjusted and expanded at runtime through protocol discovery of self and neighbor nodes' protocol stacks. However, such dynamic configuration mechanisms require reprogramming of existing protocols. On the other hand, UPS maintains the integrity of protocols and does not intervene with the protocols outside the well-defined UPS interfaces.

MobileMan [23] is proposed for MANETs where an abstracted database, called the network status (NeSt), is used by all layers of the stack through a publish/subscribe API. The protocols that need information from other layers have to register with NeSt using a subscribe API for event notifications regarding this information. A publish API is used by protocols to notify NeSt of the occurrence of an event that may be of interest to other protocols. Hence, MobileMan provides the analogous functions to our UPS information sharing interface, although it lacks the UPS protocol interface functions to offer flexible stack layer design and to support the co-existence of multiple protocols.

In [24], a staircase protocol stack is proposed for VANETs. Due to the wide spectrum of applications (vehicle-to-vehicle and vehicle-to-roadside communication) and various types of communication (uni- and multicast, WiFi and sensor networks), services in VANETs tend to be application-oriented. The staircase approach provides a method whereby the application layer can directly bypass certain layers to control the lower layers. The staircase approach is a special case of the UPS framework by placing dummy bypass protocol modules into layers, as illustrated in Section 4.3.2.

Kumar et al. propose SensorStack [25], a five layer protocol stack for wireless sensor networks where a “Data Fusion Layer” is employed instead of the traditional “Transport Layer” in the OSI protocol stack [26]. SensorStack focuses on the exchange of information across stack boundaries. Cross-layer modules share information through system-wide management modules in order to retain modularity. The services of SensorStack are outside the protocol stack, since the design does not define a stack that shares protocol modules. These services send packets directly to the radio, bypassing the routing and MAC layer. This stack approach is not flexible and hard to employ within a network since it defines direct access to the Physical layer without going through the MAC, which results in channel access without any control for the services.

X-Lisa [14] uses a similar concept of information-sharing to SensorStack, using system-wide management modules to unify an information-sharing interface. However, X-Lisa inserts an additional layer between the Routing and MAC layers in the original OSI protocol stack. This appended layer, called CLOI, exchanges information between neighbor nodes through piggy-backing information onto broadcast packets. CLOI inserts and retrieves shared information from every broadcast packet that is passed through the Routing and MAC layers. In this thesis proposal, we show that two cross-layer protocols (DAPR [27] and XLM [15]) can be modified for the X-Lisa architecture, and we show the qualitative and quantitative benefits brought by using X-Lisa. X-Lisa also enables the exchange of information between the nodes through piggy-backing the information on outgoing packets. However, X-Lisa does not consider the possibility of employing multiple network modules *at the same time*, hence we developed the UPS protocol stack architecture.

The Chameleon architecture is proposed for wireless sensor networks to build generic MAC modules in order for a routing protocol to access the heterogeneous underlying radio devices [28]. Chameleon decomposes the entire MAC layer and part of the routing layer functionalities into basic functional blocks to build these generic MAC modules. This approach is specifically developed for multiple radio scenarios where one routing protocol is associated with multiple MAC protocols. However, UPS enables a more generalized method of protocol interface that can be used for all layers. Chameleon also only provides limited cross-layer information-sharing support, where information is exchanged through appending the intended information to packets.

TinyOS is an open-source operating system designed for wireless sensor networks. Primitive support for multiple protocols is achieved via the Active Message mechanism in [29]. However, Active Messaging is not a protocol stack architecture, instead it provides a tool for packet handling. This mechanism can be used for implementing the *UPS-PI*. The implementation of UPS in TinyOS includes a Message Pool with internal packet layout to support packet switching within the protocol stack

2.2 Receiver-Based Routing

Receiver-based communication is an opportunistic way of thinking about routing in that decisions are not required to be made at the sender side but instead are made at the receiver side. For example, a source node in ExOR [30] broadcasts packets that include a potential forwarders' list inside the header, and these potential forwarders contend to forward the packet through the use of different back-off times, which depend on the network distance to the destination. A source node in XLM [15] broadcasts packets with the destination's geographic location in the header, and every receiver contends to forward the packet through the use of different back-off times, which depend on the geographic distance to the destination. SOAR [31] uses the same idea, but in addition supports multiple paths for strategically selecting relay nodes. In other words, in receiver-based routing, decision-making is deferred to the possible receivers, that make decisions in a distributed manner.

Receiver-based routing is different from "On-demand" [32] or "Reactive" routing in that reactive routing calculates a route at the time a packet is sent down to the MAC layer. For example, AODV [33] begins transmission by first sending a "RouteRequest" to create temporary routes among intermediate nodes and then transmits data packets through the route established by the RouteRequest.

The ability to transmit data without requiring a route to be formed is enabled via extra knowledge in the MAC layer and the joint decisions of receiver nodes. Nodes could be assigned an ID in a structured manner and hence next hop nodes are implied in the destination address itself. For example, a node's ID can be assigned as the hop count from the sink to the node, assigned from an initial flooding control packet. In this case, DATA packets are broadcast by the MAC layer, and only potential next-hop nodes (smaller ID nodes) relay it to the destination. As another example, nodes may have statistics (e.g., energy, channel quality) that could assist in making forwarding decisions. A source node can send a request-to-send (RTS) packet, enabling potential receivers to contend for the ability to forward the packet, with the receiver node that has the best route being the first to return a clear-to-send (CTS) to receive this packet.

Researchers have analyzed the performance of receiver-based routing through mathematical models [34] [35] and shown that receiver-based routing protocols perform well in terms of hop distance, energy and latency. Unicast traffic is assumed in these works, hence, for convergecast and multicast traffic, further studies are required.

2.3 Multicast Routing

Existing multicast protocols for WSNs and mobile ad-hoc networks (MANETs) generally use a tree to connect the multicast members [36] [37] [38] [39] [40] [41]. For example, the Takahashi-Matsuyama heuristic can be used to incrementally build a Steiner tree for multicast routing [42] [43]. Additionally, multicast algorithms rely on routing tables maintained at intermediate nodes for building and maintaining the multicast tree [44] [45]. ODMRP [36], CAMP [37] and PUMA [38] are suitable for high mobility rates when a large number of nodes are needed to forward data messages. MAODV [39], ADMR [40] and AMRIS [41] require fewer nodes but more reconstructing of trees for forwarding data messages.

In location-based approaches to multicast routing [46] [47] [48], nodes obtain location information by default as an application requirement (e.g., a home fire detection sensor would know where it is located) or as provided by a system module (e.g., GPS or a location-finding service). If location information is known, multicast routing is possible based solely on location information without building any external tree structure. For example, PBM [49] weights the number of next hop neighbor nodes and total geographic distance from the current node to all destination nodes and compares this to a predefined threshold to decide whether or not the packet should be split. PBM is a generalization of GFG (Greedy-Face-Greedy) [50] routing to operate over multiple destinations. GMR [51] selects neighbors based on a cost over progress framework integrated with greedy neighbor selection. Geocast [52] delivers multicast packets by restricted flooding. Nodes forward multicast packets only if they are in the Forwarding Zone calculated at run time from global knowledge of location information.

In this thesis, we propose RBMulticast, which differs from these location-based approaches in that it is completely stateless and hence no costly state maintenance is required. The state maintenance of conventional multicast protocols requires extra traffic to keep the state information up to date, as well as requiring processing of the state information communicated and storage of this state information. On the other hand, in RBMulticast, only the node's own location and the location of the multicast members are needed for multicast packet routing.

2.4 Duty Cycle Control

Duty cycling is an aggressive means to save the energy of battery-operated devices. The duty cycling method shuts down radio subsystems whenever nodes are not participating in radio communication. For resource limited applications, it is intuitive to make nodes sleep whenever possible. For example, wireless sensor nodes with low data rates will spend most of their time in the idle state, and thus large gains can be achieved by reducing the energy wasted in idle listening.

Most often, all nodes in the network have the same duty cycle, in order to facilitate communication among the nodes. In some cases, all nodes' duty cycles are synchronized [53] [54], while in other cases, the nodes' duty cycles are asynchronous, so the nodes wake up at random times in relation to each other [55] [56] [57].

Recent work has looked at adapting the duty cycle to the local network conditions. For example, adapting the duty cycle to the local traffic was proposed in PMAC [58], where the sleep-wakeup schedule is represented by a string of bits that are updated each period using local traffic information available at the node. These schedules are exchanged at the end of each period, so that neighboring nodes are aware of each others' schedules. Another adaptive duty cycle approach, ALPL, adjusts a node's duty cycle according to the node's neighbors' duty cycles in order to support the data flows it receives [59]. However, none of these approaches optimize the duty cycle for convergecast data patterns and receiver-based routing.

In convergecast communication, the packet traffic observed around the sink node is much higher than the traffic observed far from the sink, i.e., nodes with different distances to the sink node receive and must relay different amounts of traffic. It is clear that a network-wide fixed duty cycle will not provide the optimal trade-off between energy efficiency and latency.

In this thesis, we utilize receiver-based protocols, which enable nodes to communicate with no synchronization or neighbor information, and hence do not require all nodes in the network to have the same duty cycle.

Chapter 3

Evaluation of the X-Lisa Cross-Layer Protocol Architecture

In this chapter, we examine an existing information-sharing layered architecture called X-Lisa [14] to evaluate its ability to support information-sharing among protocol layers. We select a layer-fusion cross-layer protocol called XLM [15] as our target communication protocol. XLM is of particular interest because it represents an extreme in cross-layer designs, consisting of only a single layer that carries out all of the functionality from the application layer to the MAC layer. Hence, it is a good candidate to test the benefits and limitations of a layered information-sharing architecture such as X-Lisa.

We show that the XLM single layer protocol can be implemented in a layered stack structure with information-sharing with no performance degradation. Additionally, we show that we can easily swap MAC protocols if we properly design a communication protocol using a layered approach.

The results of this chapter reveal some design limitions of the X-Lisa protocol stack architecture that we discovered during the implementation of the XLM/X-Lisa protocol. Therefore, a new universal protocol stack UPS will be introduced in the next chapter as a solution for emerging wireless networks.

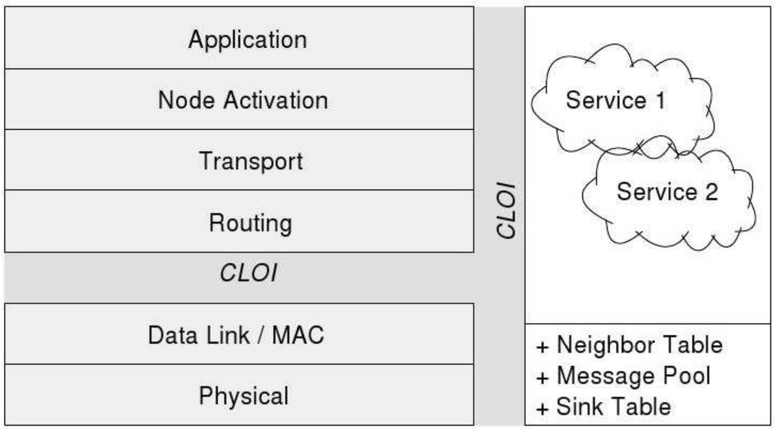


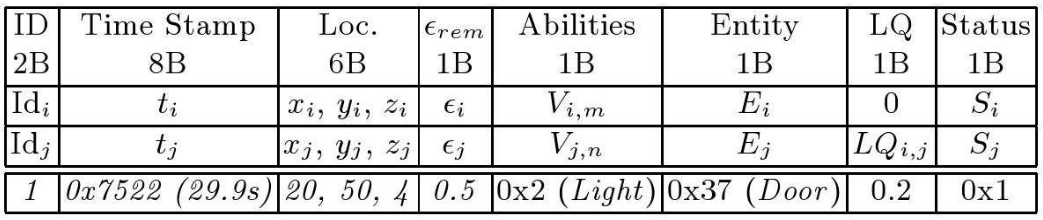
Figure 3.1: X-Lisa architecture.

3.1 X-Lisa: Cross-Layer Information-Sharing Architecture

The goal of X-Lisa is to provide protocol stack infrastructure support for information exchange, either between layers within a node or among neighbors. X-Lisa combines simplicity with support for cross-layer interactions, services, information propagation and event notification.

X-Lisa maintains three data structures that are used to store shared information: neighbor table, message pool, and sink table. Shared data structures are accessed through an additional interface called CLOI (Cross-Layer Optimization Interface). CLOI also includes an extended pseudo layer between the MAC and network layers to intercept packets. This process is done implicitly to piggyback/exchange extra information among neighbor nodes. The MAC and network layers do not need to know the underlying manipulations behind CLOI and X-Lisa. The X-Lisa protocol architecture is shown in Fig. 3.1.

The neighbor table stores shared information about the node itself and about neighbor nodes. The categories of shared information are predefined though CLOI interfaces at compile time. A default set of important parameters is identified as necessary for improving the performance of many sensor network protocols. The default neighbor table is shown in Fig. 3.2. Each row in the neighbor table stores the information shared with a target neighbor node, and table entries are updated through information piggy-



ID 2B	Time Stamp 8B	Loc. 6B	ϵ_{rem} 1B	Abilities 1B	Entity 1B	LQ 1B	Status 1B
Id$_i$	t_i	x_i, y_i, z_i	ϵ_i	$V_{i,m}$	E_i	0	S_i
Id$_j$	t_j	x_j, y_j, z_j	ϵ_j	$V_{j,n}$	E_j	$LQ_{i,j}$	S_j
1	*0x7522 (29.9s)*	*20, 50, 4*	*0.5*	0x2 (*Light*)	0x37 (*Door*)	0.2	0x1

Figure 3.2: A neighbor table is kept at every node i with information about itself and each of its neighbors j.

backed onto other packets, which are intercepted by the CLOI layer between the MAC and network layers.

The Message pool records information about data packets, including the type and priority of the packets. The message pool can be used with the neighbor table to help each layer make decisions about routing and sleep schedules. The Sink table stores required critical information about the various sinks in the network, including distance to the sink and attributes of the sink nodes. Both the message pool and the sink table need to be defined at compile time, and are not used in the implementation of XLM described in this chapter. Detailed definitions of the data structures can be found in [14].

An important service provided by X-Lisa is the location service. The location service of X-Lisa periodically updates a node's coordinates, and CLOI exchanges this information with the node's neighbors. Thus the neighbor table is kept up-to-date on the location of the node and each of its neighbors, removing the burden of performing this location service from the routing protocol. As will be introduced in the next section, XLM packet forwarding is based on the geographic location of the nodes, and hence this location information is critical in the XLM protocol design.

3.2 XLM: A Fused-Layer Protocol for Wireless Sensor Networks

The intention of XLM [15] is to avoid the issues in the traditional layered protocol approach with a complete unified protocol. All the functionalities of the traditional communication stack layers are converged into a single module in XLM. Hence, XLM belongs to the the layer-fusion cross-layer approach. Even with a single layer design, XLM provides the routing/MAC functionality and supports unicast communication.

By allowing all functionality to reside within a unified layer, routing decisions can be made with the information exchange of MAC layer control packets. The exposure of the lower layer information to the routing layer reduces the overhead of extra information exchanges. Furthermore, because components can access any necessary information directly, efficient energy consumption mechanisms and congestion control can be implemented in a straight forward manner.

Network communication in XLM is based on a MACAW [60] style hand-shaking protocol, where data transmission is based on 4 types of packets: RTS (request-to-send), CTS (clear-to-send), DATA, ACK. The XLM message transmission process is determined by the following hand-shake steps:

1. If node A wants to transmit a message to the sink, node A will first broadcast an RTS packet. Other nodes that receive the RTS packet acknowledge with a CTS packet to node A after a delay determined by a cost function that is based on their distance to the sink.

2. Node A will choose the first returned CTS packet as the winner of this hand-shake. This will trigger node A to start sending DATA packets to the winner, node B.

3. When node B receives a DATA packet, it responds with an ACK packet corresponding to the received DATA packet. Thus, XLM implements a stop-and-wait acknowledgement protocol, where each ACK packet corresponds to a particular DATA packet, and the transmitter (node A) does not send another DATA packet until receiving an ACK for the previous DATA packet. This procedure is repeated until all DATA packets are successfully received and acknowledged by node B.

4. The rest of the nodes covered in the radio range of any of the CTS/DATA/ACK packets know that there are active on-going communications, and hence they enter the sleep state to save energy and to avoid packet collisions.

In this routing/MAC mechanism, the sender node passively chooses the first responding node as the winner node to establish the connection. We classify this to be receiver-based communication because, instead of the sender making the decision about the next-hop node, the packet route is determined by the CTS backoff time chosen by the potential receiver nodes.

The potential receiver nodes' CTS return back-off delay times are a function of distance from the node to the sink node. This back-off time is determined as follow. The RTS packet coverage range is divided into N_p priority regions, in increasing distance order from the transmitter. Each priority region corresponds to a backoff window size, CW_i. Nodes in the feasible region, which is the region where forward progress to the sink is ensured, will send CTS packets with back-off time $\sum_{i=1}^{N_p-1} CW_i + cw_i$, where cw_i is a random generated number number $\in [0, CW_i]$. Thus, nodes nearer to the destination will become the winner of the current contention due to smaller back-off times. Neighbor nodes not in the feasible region are further away form the sink node than the transmitter and will not return CTS packets in response to RTS packets.

XLM is also classified as a geographic routing protocol, because distance is used to calculate the back-off delay function and to make routing decisions. In our X-Lisa implementation, this location information is stored and maintained by the neighbor table in X-Lisa for every node.

3.3 XLM/X-Lisa: Example of Layer Fusion Decomposition

XLM exhibits total layer fusion and is the counter-part of a layered protocol stack. It is desirable for an information-sharing cross-layer approach (such as X-Lisa) be able to replicate the performance achieved by XLM while maintaining the convenience of modular protocol layers. In our work, we show that this is possible. Furthermore, we illustrate the advantage of using a modular scheme by swapping the MAC protocol from the original XLM's MAC functions to a Low-Power-Listening (LPL) scheme [55].

3.3.1 Layer Implementation

In the initial protocol suite, we decomposed XLM into the layer modules shown in Fig. 3.1 and included them in the X-Lisa architecture. The new entity, called XLM/X-Lisa, is shown in Fig. 3.3(b) and is the layered version of XLM.

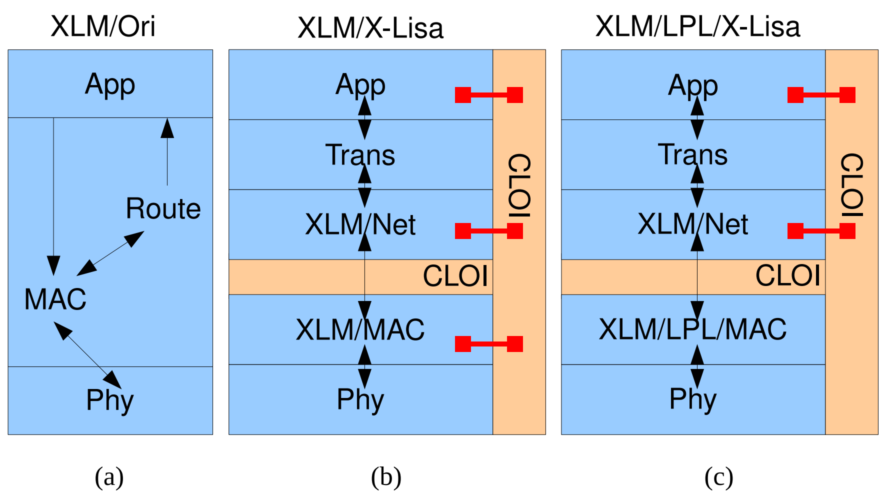


Figure 3.3: The original XLM (a), was broken into a layered scheme (XLM / X-Lisa) (b), and its MAC layer was replaced (c). Arrows show packet exchanges between layers, and squares show information exchanges.

From the application layer's perspective, XLM/X-Lisa is designed to have the same send/receive function call interface. With no change of the existing code, the application layer is able to switch from the original XLM to XLM/X-Lisa and expect the same packet send/receive behavior.

The transport layer extracts information from the user data and segments long data packets. Long data will be decomposed into several small segments of data that can fit into the network layer packet payload. A transport layer header will be added on these data chunks before they are delivered to the lower layer.

In this suite, the network layer XLM/Net has limited roles. Because the role of routing is transfered to the Linker layer, XLM/Net does not need to maintain a routing table. Hence, there are no extra control packets generated from the Network layer. Since there is no need to exchange information between network layers among different nodes, XLM/X-Lisa minimizes the packet size by ignoring the Network layer's header. XLM/Net simply maintains a queue of stored packets for the Link layer.

The Link layer XLM/MAC performs the RTS/CTS/DATA/ACK packet handshaking and controls the radio on/off for duty-cycling. Packet delivery failure information is shared with the Application layer by X-Lisa's CLOI interface to control traffic congestion.

3.3.2 Information Exchange with X-Lisa

In XLM and XLM/X-Lisa, receiver nodes make routing decisions with an initiative concept. According to dynamic statistics and an initiative $\imath$, receiver nodes can calculate and decide if they should accept a given RTS packet. The following is the calculation of initiative $\imath$:

$$\imath = \begin{cases} 1, \quad if \begin{cases} \xi_{RTS} \geq \xi_{Th} \\ \lambda_{relay} \leq \lambda_{relay}^{Th} \\ \beta \leq \beta^{max} \\ E_{rem} \geq E_{rem}^{min} \end{cases} \\ 0, \quad otherwise \end{cases}$$

ξ_{RTS} is the SNR of the received RTS packet, and ξ_{Th} is the SNR threshold. RTS packets with lower ξ_{RTS} than the threshold will be ignored by the receivers. λ_{relay} is the current relay rate, which represents the number of packets that have been routed in a unit time period. λ_{relay}^{Th} is the relay rate threshold. A node with too many packets in the routing queue results in lager relay rate than the threshold; receiver nodes will stop relaying more packets by not joining the current routing contention competition. β is the current packet memory buffer usage, and β^{max} is the buffer usage threshold. E_{rem} is the remaining energy in the node, and E_{rem}^{min} is the minimum remaining energy required. Nodes with low energy will not join the contention competition and route packets. $\imath = 1$ means that the receiver is allowed to return a CTS packet, and $\imath = 0$ means that it is not allowed.

Congestion control is accomplished by coordinating the data generation rate λ of the nodes. Data generation rate will vary according to the feedback of the sending and receiving packets from the XLM/MAC layer. The Application layer utilizes a mechanism that increases the data generation rate λ when an ACK packet is received. This implies that the data rate can be increased whenever a data packet is successfully delivered. Additionally, λ decreases if no CTS packet is received after sending an RTS packet. It is assumed that the cause of absent CTS packet is due to the busy states of all neighbor nodes.

X-Lisa uses the neighbor table as a central storage of the nodes' positions for location look-up. Because X-Lisa piggy-backs necessary information to packets, the packet

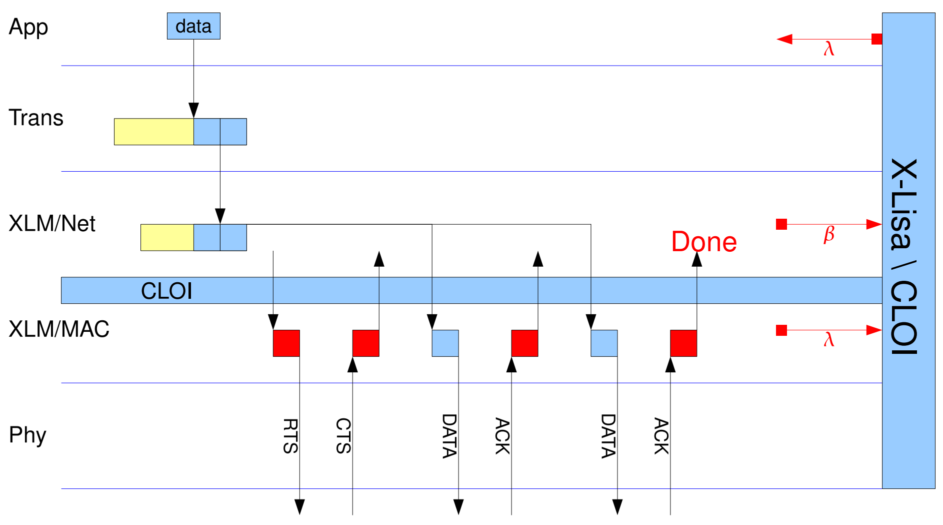


Figure 3.4: The flow graph for sending a packet. The arrows of β, λ, and d indicate the packet buffer usage, packet rate, and distance information exchanged through the X-Lisa interfaces.

payload of the Link layer will have the extra information added by CLOI.

3.3.3 Input/Output Packet Flow

The sender side of the XLM/X-Lisa protocol is shown in Fig. 3.4. To initiate a transmission, the sender side of the application generates a new request to send data. The data will be forwarded to the Transport layer. The data generation rate λ is adjusted according to the collision information of the MAC layer.

Data from the user will be segmented into small data packets and delivered to the Network layer packet queue, and the Network layer will then issue a send-request to the MAC layer. If the MAC layer is idle when the Network layer send the request, it will accept the request and start the first RTS packet for the RTS/CTS/DATA/ACK hand-shaking process.

If the MAC layer receives a response CTS packet, it knows that the relay node is decided. Then, it will get the first data segment from the XLM/Net protocol, add a link layer header, send out this DATA packet to the receiver node, and wait for an ACK packet for this DATA packet. Every DATA packet has a corresponding ACK packet. After the MAC receives the last ACK packet, it returns a *done* event to XLM/Net for

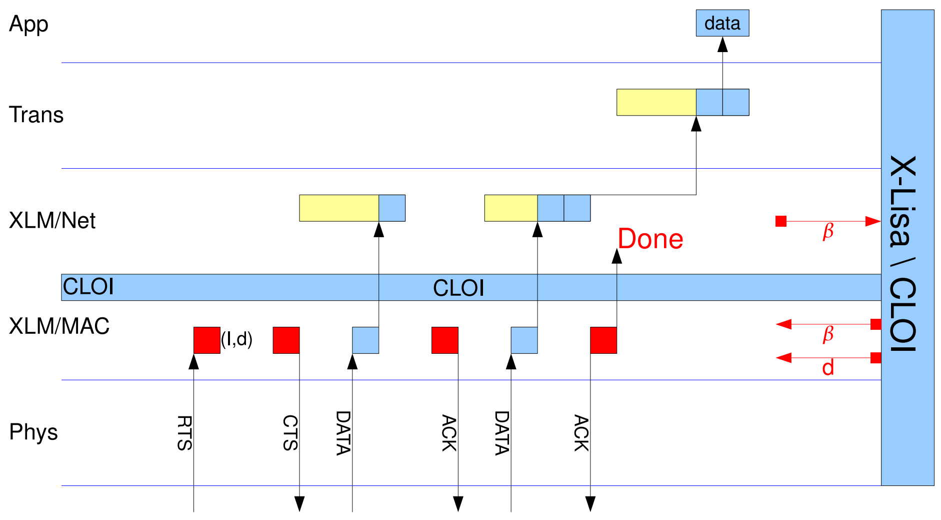


Figure 3.5: The packet flow graph of a node acting as the receiver node. The arrows of β, λ, and d indicate the packet buffer usage, packet rate, and distance information exchanged through the X-Lisa interfaces.

cleaning the packet queue. The value of λ will be updated by Link layer according to the CTS and ACK packets' status, as described in Sec. 3.3.2.

The sender node will go to the idle state if there are no future packets in the XLM/Net queue. Otherwise, it will wait a random period before sending the next packet to avoid network congestion.

A packet flow graph of a receiver node is presented in Fig. 3.5. When the node receives an RTS packet, it uses CLOI to get the location of the sender and the destination node. It then calculates the distance to the destination node and the current feasible region. If the receiver node is not inside the feasible region, the RTS packet will then be dropped. The receiver node will go to the sleep state in order to save energy. Otherwise, the initiative $\imath$ is calculated from the current statistics of the node.

If $\imath$ equals 0, the node will go to the sleep state. Otherwise, if $\imath$ equals 1, then the node will return a CTS packet with a backoff time according to its distance to the destination node. The reciever node will go to the idle state and wait for the first DATA packet. After the requested DATA packet is received, it will be forwarded to the upper layer to a receiving buffer inside the XLM/Net layer. Immediately, the XLM/MAC layer sends back an ACK packet. Every DATA packet has a corresponding ACK packet. If it is the last ACK packet, the XLM/Net layer will be notified to do either of the following two steps according to whether the reciever node is the destination node. If the receiver node is the destination, it will upload all the data packets to the transport layer. Otherwise, the receiver node will redirect the data packets to the output packet queue inside the XLM/Net protocol. Fig. 3.5 assumes the reciver node is the destination node of the arriving packet.

3.4 XLM/LPL: Example of Protocol Swapping

Modular designs allow replacing a protocol with little or no difficulties, otherwise, the benefit of including a new protocol may be outweighed by the amount of work necessary to incorporate them into an existing framework.

X-Lisa enables the insertion of new protocols through the appropriate use of the CLOI interface. To show the flexibility of X-Lisa, we experiment with swapping the MAC protocol from the XLM/MAC to a low-power-listening (LPL) MAC protocol.

3.4.1 Protocol Swapping Implementation

The new protocol suite is a variant of XLM/X-Lisa: the original MAC layer was replaced by the LPL MAC protocol SpeckMAC-D [55], as illustrated in Fig. 3.3(c). We named the new entity XLM/LPL/X-Lisa.

In SpeckMAC-D, every node sleeps for t_i s between wake-ups, where t_i is the inter-listening time. In order to guarantee that the receiver will wake up at some point during a transmission, a sender must repeat the same packet for t_i s. If a node wakes up and receives a packet, its MAC protocol forwards it to the network layer before sleeping for the rest of the cycle. In SpeckMAC-D, no RTS/CTS/ACK control packets are used in the packet delivery.

Because packet delivery of XLM/X-Lisa cross-layered design is based on location information, SpeckMAC-D is modified to support location information by making receiver nodes route packets only if they are closer to the destination node. This is similar to restricted flooding of packets in Geocast [52].

Similarly to XLM, we modified XLM/LPL/X-Lisa to route packets only if the node is closer to the destination, which is determined via the “feasible region.”

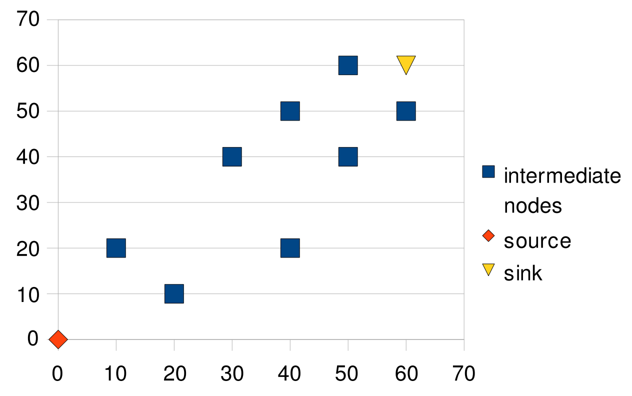


Figure 3.6: The source node is at (0,0) and the sink node is at (60,60). Radio range is 30m.

Table 3.1: Simulation results of XLM/LPL/X-Lisa and XLM/X-Lisa.

	XLM/LPL/X-Lisa	XLM/X-Lisa
Received Packets	165	2036
Transmitted Packets	3674	378
Goodput	0.75	1.0
Latency	1.503	0.683

The two suites of protocols were implemented in TinyOS and emulated with TOSSIM. We conducted emulations on 10 nodes with the topology shown in Fig. 3.6. The source node at location (0, 0) sends a packet to the sink node at location (60, 60) every 5 s, for a total simulation time of 100 s. The results are shown in Table 3.1.

These results show that XLM/LPL/X-LISA sends more packets than its original counterpart because the LPL scheme repeatedly sends packets over t_i s. The RTS/CTS handshake accounts for over a third of the 378 packets transmitted in the XLM/X-Lisa suite.

On the other hand, XLM/X-Lisa receives more packets because every communication requires a hand-shake, and because many nodes receive RTS/CTS/DATA/ACK packets even though they have lost the contention to other nodes and are not part of the communication. Finally, both suites exhibit high goodput (greater than 75%), with

XLM/X-Lisa showing better performance. Likewise, XLM/X-Lisa yields lower latency.

These results show that the replacement of the original XLM/MAC protocol by SpeckMAC-D is feasible and X-Lisa did not degrade the performance of the protocol: according to a set of quantifiable metrics, the two suites behave according to our expectations.

3.5 Conclusions

In this chapter, we showed that X-Lisa is expressive enough to support the decomposition of the layer-fusion protocol XLM. We also showed that the design of a new protocol suite is possible through "swapping" different protocol modules in a layered protocol stack. A new protocol suite XLM/LPL/X-Lisa was proposed without modifying the rest of the system, by simply swapping the MAC protocol modules.

Even though we demonstrated the strength of the X-Lisa information-sharing layered architecture, there are still many questions that remain. Does X-Lisa have the complete/right support for networks beyond sensor networks? X-Lisa provides neighbor tables for information exchange between neighbor nodes. Should we also maintain records for nodes further away? Should we have the information table format the same for all nodes and network protocols/applications? Different protocols require different services, and there are potentially an infinite number of possible protocols. Should CLOI keep growing to include all the unforeseen new services, or should we unhook CLOI from the stack structure in order to reasonably maintain CLOI?

More importantly, recent devices developed for emerging wireless networks support multiple communication substrates and require the execution of multiple protocols within a layer. The CLOI layer between the routing and MAC layers does not provide adequate support for simultaneous execution of protocols.

In the next chapter, the experience learned from our work with X-Lisa is used to develop a new protocol stack architecture. The proposed Universal Protocol Stack (UPS) discussed in the next chapter intergrates the idea of "protocol switching" and achieving the ability to execute multiple different protocols in the same layer at the same time, which is a suitable approach for emerging wireless networks.

Chapter 4

UPS: Universal Protocol Stack for Emerging Wireless Networks

Recent devices developed for emerging wireless networks, such as 4G cellular networks, wireless mesh networks, and mobile ad hoc networks, support multiple communication substrates and require the execution of multiple protocols within a layer, which cannot be supported efficiently by traditional, layered protocol stack approaches. While cross-layer approaches can be designed to support these new requirements, the lack of modularity makes cross-layer approaches inflexible and hence difficult to adapt for future devices and protocols. Thus, there is a need for a new protocol architecture to provide universal support for cross-layer interactions between layers, while also supporting multiple communication substrates and multiple protocols within a stack.

In this chapter, we propose Universal Protocol Stack (UPS), which provides such support in a modular way through packet-switching, information-sharing, and memory management. To show that UPS is realizable with very low overhead and that it enables concurrent and independent execution of protocols in the same stack layer, first, we present a wireless sensor network test-bed evaluation, where UPS is implemented in TinyOS and installed on individual sensor motes. Two cross-layer routing protocols are implemented and evaluated with UPS and without UPS. We also implemented UPS in the OPNET simulator, where the IP and AODV routing protocols are executed concurrently to support networks with both static and mobile wireless nodes. Our implementation shows that the overhead incurred to implement UPS is very low, and little or no modifications are required to adapt existing protocols to the UPS framework. Both the implementation and the simulation studies also show the advantages of enabling concurrent protocol execution within a stack layer, improving the successful packet delivery ratio or the total number of packets sent for the investigated scenarios.

4.1 Introduction

Emerging wireless networks such as 4G cellular networks, wireless mesh networks, and wireless ad hoc networks aim to effectively utilize recent devices that employ multiple communication substrates, such as laptops with Ethernet and WiFi network interfaces, or cellular phones with GSM and WiFi network interfaces. Existing protocol architectures, which include the traditional layered approach (e.g., the OSI or TCP/IP layered protocol stacks), cross-layer information-sharing, and layer fusion architectures cannot provide sufficient support for these emerging networks. The traditional layered approach does not provide support for multiple protocols in a layer; hence supporting multiple MAC/PHY for different communication substrates requires that the protocol designer make ad hoc modifications of the stack. While the fused-layer and information-sharing cross-layer approaches can be designed to support multiple communication substrates, the lack of modularity makes these approaches inflexible and hence difficult to adapt to support future devices and protocols. Thus, there is a need to investigate a new protocol architecture that has universal support for emerging wireless networks with multiple communication substrates and that supports cross-layer interactions and information-sharing to enable more efficient network operation.

Several wireless mesh network (WMN) studies assume routers have the ability to switch among multiple radios. In [9], the authors investigated the implementation of interface switching on multiple radio devices for WMNs, and they found that existing hardware and protocol stacks do not provide sufficient support for effective switching. The authors indicated that there was excess delay due to slow hardware radio switching, and high packet loss rate because packets in the old queue were discarded after switching to the new radio. The authors also found that when implementing routing protocols for multiple radio interfaces in the Linux operating system, modifications of the stack are needed because there is an implicit assumption that each interface is associated with exactly one channel. The authors propose to alter the TCP/IP stack by adding a channel abstraction module to switch packets between multiple interfaces and the routing layer.

Another implementation of multiple radio devices is investigated in [61] for Windows XP. The authors propose similar modifications to the TCP/IP stack as proposed in [9], where a virtual device driver runs between the Network layer and the Data Link layer. This additional module performs multiplexing and demultiplexing across multiple physical interfaces to imitate a single network interface and MAC address to the upper layer protocols. As shown by the approaches of [9] and [61], current operating systems, which implement the layered protocol stack, lack support for general packet switches, which is necessary for supporting devices with multiple radios.

Similarly, studies on the implementation of multiple MAC protocols sharing a single radio [10,62,63] have found that traditional stack approaches are not sufficient. The idea behind such systems is that the MAC protocol can be selected out of a set of MAC protocols at run time according to real time network conditions. To achieve this goal, two general interfaces supporting the set of MAC protocols are proposed between the Network and the Data Link layer, and between the Data Link layer and the Physical layer. These general interfaces provide fast switching for ensuring that each packet is sent to the appropriate MAC [62]. Inter-device control information exchange is important in addition to information exchange among protocol layers, because different devices must use the same MAC protocol in order to communicate.

In this thesis proposal, we propose Universal Protocol Stack (UPS), a new protocol stack architecture where multiple radios and multiple concurrent protocols within a layer are supported with very low overhead. Protocol IDs are used to identify packets of different protocols and to enable general protocol switches, as detailed in Section 4.2. Additionally, UPS enables support for cross-layer interactions and information exchange between different layers in order to optimize network performance.

UPS is illustrated in Fig. 4.1 for a node that runs multiple protocols in each layer concurrently. Different protocols and services are executed as modules and managed by the *UPS protocol interfaces* (*UPS-PIs*) with very low overhead incurred. UPS, thus, can efficiently execute multiple protocols in the same stack layer for different tasks, for different communication substrates and different applications. For example, the protocols can be IP using an IEEE 802.11 radio and AODV [33] using an 802.15.4

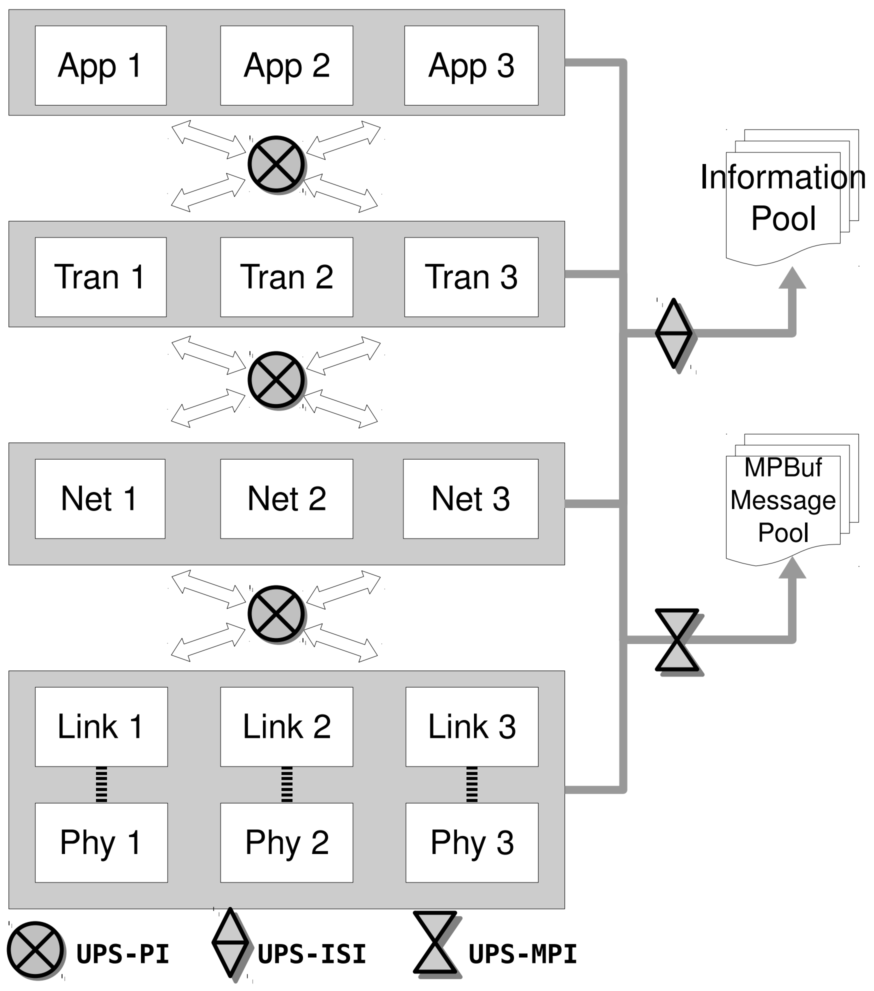


Figure 4.1: An example of the high level system block diagram of the UPS framework.

radio, where both networks are shared by the upper layer protocols (e.g., TCP/UDP) and applications.

UPS defines three interfaces, *UPS protocol interface* (*UPS-PI*), *UPS information-sharing interface* (*UPS-ISI*), and *UPS Message Pool interface* (*UPS-MPI*), along with a packet memory management scheme, namely UPS Message Pool. The *protocol interface* (*UPS-PI*), which is a protocol switch between stack layers, selectively delivers packets to the correct upper or lower target protocol module. With very low overhead incurred, UPS can efficiently execute multiple protocols in the same stack layer for different tasks. The same functionality could be achieved with specific customization of layered stack approaches as proposed in [9, 61], however, UPS provides a flexible, modular, and easily portable approach, enabling a well-defined framework for protocol execution.

At the same time, UPS provides a universal means for information exchange among the protocol modules through an *information-sharing interface* (*UPS-ISI*), making UPS ideally suited for complex heterogeneous networks. UPS also defines a packet memory management scheme, *Message Pool*, which is accessed via a *UPS Message Pool Interface* (*UPS-MPI*). This scheme is required to manage the available memory segments for packet storage and to provide a universal interface to protocols for accessing packet memory storage dynamically, e.g., to access the vital packet specific information such as the source or destination addresses.

We show that UPS is flexible and achievable in practice by implementing UPS on physical sensor network devices. The results show that cross-layer interaction is achieved with UPS easily and with very low overhead, while the well-defined interfaces of UPS protect the independence and modularity of separate protocol modules.

Due to the numerous advantages of enabling the co-existence of multiple protocols in a stack layer and of information-sharing, UPS is ideally suited for emerging wireless networks that require multi-functional or cross-layer support. Different protocols are responsible for different tasks, and they share vital information among themselves. The resulting output packets gracefully co-exist in the network. As an example, in this chapter we show that having UPS to support concurrent execution of IP and AODV routing protocols on a network with both static and mobile devices achieves better overall performance than running only one of these protocols.

We demonstrate the gain of this novel approach by comparing it with previous single protocol stack designs through detailed simulations and physical experiments. Two different networking paradigms, namely wireless sensor networks (WSNs) and mobile ad hoc networks (MANETs), are investigated with UPS. For each network, two routing protocols are used as protocol modules to show that different types of network layer traffic can co-exist with UPS while sharing the same MAC layer in a predictable manner: i) WSNs with the XLM [15] cross-layer protocol and the RBMulticast [17] multicasting protocol, and ii) MANETs with AODV and TCP/IP. More specifically, results show that utilizing UPS with two Network layer protocols running simultaneously and independently, the successful packet deliveries can be increased significantly while reducing the network traffic compared to using a traditional stack approach with a single Network layer protocol. For the scenarios investigated in this thesis proposal, the successful packet delivery ratio is increased up to 36% compared to using a traditional layered stack with TCP/IP/802.11, and up to 23% compared to using a traditional layered stack with AODV/802.11.

Using UPS, complex network algorithms can be divided into independent pieces for easy coding and analysis. As an example, we show in this thesis that for our sensor network implementation with UPS, a location service can be independently designed as a network protocol instead of being affiliated to a specific routing protocol, and locally provided radio power management of the CC2420 radio device is shared by all communication protocols via the *UPS-ISIs*, i.e., the UPS information-sharing interfaces.

The rest of this chapter is organized as follows. Section 4.2 provides a detailed description of UPS, including the *protocol interface* (*UPS-PI*), the *information-sharing interface* (*UPS-ISI*), and the *message pool interface* (*UPS-MPI*). Section 4.3 details results from our wireless sensor network implementation with Tmote Sky motes with UPS and from our OPNET simulations of UPS. Finally, Section 4.4 provides conclusions.

4.2 UPS Framework

UPS (Universal Protocol Stack) defines three groups of *interfaces* for interconnecting protocol modules and for enabling information-sharing among the protocol modules.

Implementations of the individual protocol modules are outside the scope of UPS; however, the protocol modules should work with the universal interface provided by UPS, or have protocol switch layers implemented, in order to interconnect with each other. Examples of protocol switch layers are shown in Section 4.3.3.

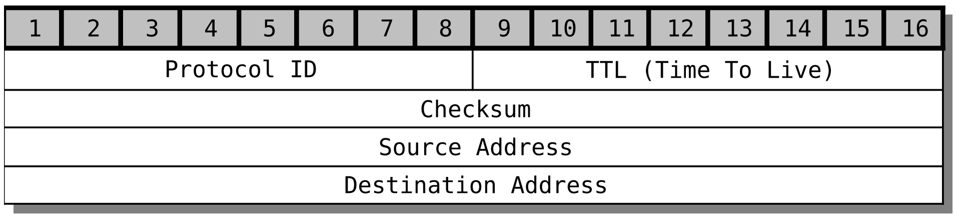
The first group of interfaces, *UPS protocol interfaces* (*UPS-PI*), enable multiple protocol modules to co-exist in the same stack layer and run concurrently and independently. This is done using an ID-based packet-switching mechanism to send packets to the correct upper or lower protocol modules.

The second group of interfaces, namely *UPS information-sharing interfaces* (*UPS-ISI*), provide a general means for a module to access another module's information stores, such as a neighbor table or location information, using unique information IDs that specify the particular information store with the data of interest.

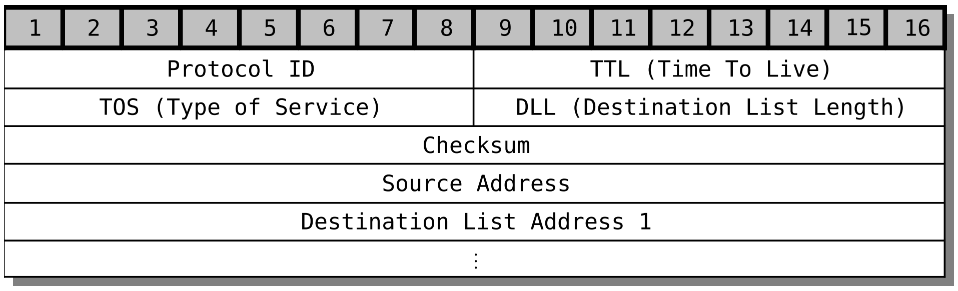
The last group of interfaces, *UPS Message Pool interfaces* (*UPS-MPI*), are used to enable multiple protocols to access common packet memory structures maintained in a *UPS Message Pool*. This section details these three groups of interfaces along with the proposed *Message Pool* system.

The basic requirement of the UPS framework on protocol design is in the protocols' packet header format. A leading header field that contains the unique Protocol ID is required by UPS in the protocol's packet header for packet switching. This Protocol ID is analogous to the port number necessary for Internet services, and it is chosen by the protocol designer. Each protocol is assigned a system-wide unique Protocol ID, as packet switches perform multiplexing and demultiplexing of packets according to the Protocol ID field. For example, the Network layer packet headers of the Network layer modules XLM/Net [15] and RBMulticast [17], which are implemented for our sensor network experiments, are shown in Fig. 4.2, where the first byte of both headers are the protocols' respective Protocol IDs.

The choice for the Protocol ID field size is a system design criteria. For the same purpose, Ethernet uses a two byte field called *EtherType*, to indicate which Network layer protocol is being used in an Ethernet frame. TCP/IP also uses the same header field, where the leading 4 bits *Version* [64] field in an IP header indicates different modules in the IP protocol, and a one byte *Protocol* [65] field indicates the next layer, i.e., Transport layer protocol.



(a) XLM/Net packet header



(b) RBMulticast packet header

Figure 4.2: Packet header format of XLM/Net and RBMulticast packets.

Using the extra first leading byte for all protocols allows UPS to perform packet switching on a per-packet basis. This enables packets from different protocols to mix together seamlessly, and it provides a generalized means to enable protocols to operate concurrently. While this does require that existing protocols be modified to provide this leading byte Protocol ID, as we will show in Section 4.3.3, with additional intermediate layers, existing protocols like IP and AODV can be integrated into UPS without any modification to the protocol itself. However, the additional bytes of Protocol ID may cause extra packet fragmentation in the other layers (e.g., TCP, IP, or 802.11) due to the packet size being over the maximum transmission unit (MTU). Therefore, the system default MTU should be adjusted accordingly when the UPS framework is enabled. One point worth noting is that 6LoWPAN [66] satisfies the Protocol ID requirement, since the *Dispatch Value* field used in the header can be used as the UPS Protocol ID.

In order to support legacy devices in networks using UPS, we can use a home agent technique as used in Mobile IP [67] or gateway devices to enable indirect connections to non-UPS-enabled devices. For example, a traffic flow coming from a non-UPS-enabled device can be sent to a UPS-enabled home agent or gateway device and then translated and redirected to the UPS-enabled device. These techniques may require some methods to differentiate packet formats in order to deduce the appropriate Protocol IDs; how to do this is outside the scope of our work.

4.2.1 UPS Protocol Interface (*UPS-PI*)

Unlike the TCP/IP stack model with five layers, UPS provides a layered protocol model without any restrictions on the number of layers. We assume the Physical layer is controlled by the Data Link layer, and there are no specific interfaces in between these layers. For the other layers, each layer is connected to a *Packet Switch* through the *UPS Protocol Interface* (*UPS-PI*). A *Packet Switch* is a generalized extension of Logical Link Control in the IEEE 802 family of standards. Its purpose is to multiplex packets passed from the upper layer (when transmitting) and demultiplex packets from the lower layer (when receiving). This is the key to the co-existence of multiple protocols in the same layer, since protocol modules receive the correct packets without knowledge of other modules in the protocol stack. Thus, instead of ad-hoc system components, protocol modules can be easily identified by multiple protocols in other layers, and protocols become tools with well-defined unified interfaces and predictable behavior. Building a new system becomes an easy procedure of combining protocols from a well defined toolbox without the need for customized protocol interfaces

UPS-PI consists of *Input* and *Output* system calls. *Input* is the interface that a protocol module would use for sending a packet up to a higher layer protocol module, and *Output* is the interface that a protocol module would use for sending a packet down to a lower layer protocol module. The function calls for these two interfaces are as follow:

```
Input(Packet);
Output(Lower layer Protocol ID, Packet);
```

We demonstrate the use of these interfaces using a schematic diagram in Fig. 4.3. The *Output* interface requires the Protocol ID of the next (lower) layer module for multiplexing of the output packets from different modules. The *Output* interface is

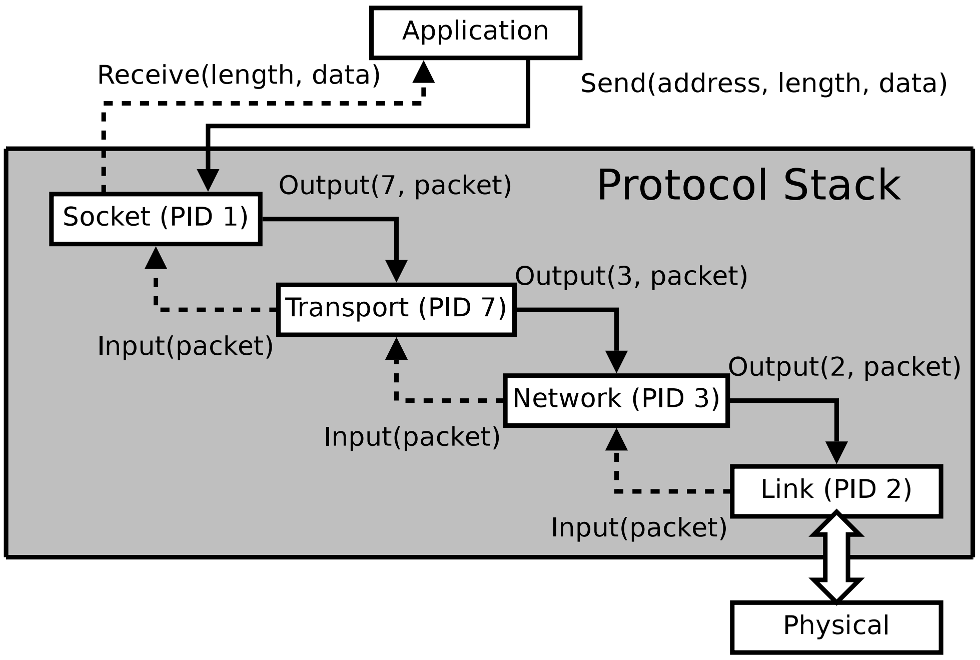


Figure 4.3: Packet flow diagram example for the UPS framework.

necessary not only to ensure that this packet reach the correct lower layer module, but also to enable the multiplexing of output packets from different modules by appending the correct Protocol IDs in front of every packet. The `Input` interface does not need the Protocol ID parameter because it is indicated by the leading byte of the packet header (the upper layer protocol module's packet header).

One issue that occurs when including the next lower layer Protocol ID as an argument of the *Output* interface is that we cannot select protocols that are not directly below the current layer. This one-layer only association is not necessarily a drawback, because it limits the protocol dependency to adjacent layers, and is also employed by current protocols (e.g., a selected socket implies TCP/UDP, TCP/UDP implies IP, etc.). The implications of this can be seen in the Section 4.3 case studies, where protocols implicitly assume the next layer protocol. An alternative is to have the application decide which protocols to employ at each layer, and have the Protocol IDs of all selected protocols stored within a general internal packet header, as will be introduced in the next section. Another alternative is to provide smart switches [68], leaving the protocol selection decision to be made by the smart switches on the fly and without input from the user. We will explore this approach in the future work that provides cross-layer interactions of the MAC and routing layers to best support application QoS goals. The best network/interface combinations for networks that support multi-radio devices will be explored.

Protocol IDs should be uniquely assigned without duplication. For example, Protocols IDs can be managed by a central authority similar to the way the Internet Assigned Numbers Authority (IANA) is responsible for maintaining the assignment of TCP/UDP port numbers.

4.2.2 UPS Message Pool and Message Pool Interface (*UPS-MPI*)

Even if protocols conform to the same *Input* and *Output* sending interfaces, different protocols still need a standardized memory structure for packets. One of the reasons is to support packet switching, since Protocol ID fields should be identified by the switches in the memory representation of the packet. Another reason is to provide different protocols access to the vital packet specific information such as the source or destination addresses. Moreover, to efficiently achieve access to a packet by multiple protocols, a common memory space should be used for the packet to avoid multiple memory copy operations.

The common memory structure is for internal packet processing within a specific system, and it can vary among different UPS implementations and devices. For example, a UPS Message Pool implementation for Linux and Windows can use the standard packet memory structure defined by the corresponding operating system. On the other hand, the packet layout definition is identical according to the specific protocol specification.

In UPS, the minimum information required for the common packet memory structure is encapsulated into a *Message Pool*, which manages available memory segments, and provides interfaces to protocols for accessing memory storage dynamically.

The *Message Pool* is accessed via the *UPS Message Pool Interface* (*UPS-MPI*), whose pseudo-code is as follows:

```
Memory Block = get(): get a memory block from the pool.
put(Memory Block): return Memory Block to the pool.
```

The *Message Pool* consists of memory blocks as shown in Fig. 4.4. As each memory block provides only a small amount of memory, to create a full packet, memory

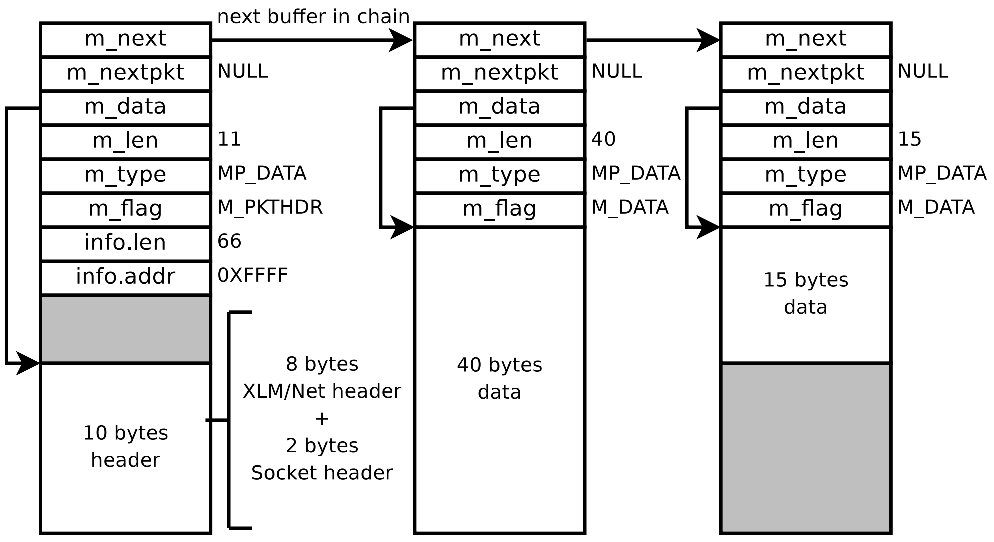


Figure 4.4: The internal packet structure of *UPS Message Pool*. The first memory block stores the packet header, and the packet data payload starts from the second block. This is a snapshot of an output packet inside the XLM/Net module. XLM/Net will be illustrated in Section 4.3.1.

blocks must be combined together. We call this structure a *Message Chain* because the individual memory blocks from the *Message Pool* are chained together by the `m_next` pointer to create the full memory structure for a packet, and different packets are chained together by the `m_nextpkt` pointer to form a chain of packets. This *Message Chain* idea imitates the implementation of packets within the Internet stack in Linux and UNIX [69].

This *Message Chain* packet format is independent from the packet data formats specified by the individual protocols in the stack, and is crucial to provide space for stackable protocol headers.

In actual *UPS-MPI* implementations, wrapper routines can be provided as assistant functions. For example, manipulation functions like `getNext(previousBlock)` or `putChain(blockHead)` can be added to the programming interface. The get() and put() functions described above are listed as the minimum necessary interfaces and thus must be provided to enable the Message Pool support. For the *UPS-MPI* implementation for systems that have their own message pool accessing interfaces, UPS should reuse their interfaces as much as possible for maximum compatibility to existing codes.

Although most operating systems such as Linux and UNIX already provide a similar packet memory structure [69], simpler operating systems such as TinyOS require implementation of the *Message Chain*. By default, the internal packet structure of TinyOS is a continuous memory space with the first 11 bytes dedicated to the Data Link layer header, the last 7 bytes dedicated to the meta data, and a fixed-sized payload in between. This packet structure does not support stackable protocol headers.

Protocol headers in the UPS stack model are organized as an *onion skin*, whereby each layer adds to or tears off the outermost skin. In the leftmost memory block in Fig. 4.4, headers are stored at the bottom of the memory block in a bottom-up manner. The data payload of a packet is stored in the second memory block in the chain and expanded to additional blocks if the payload is too large for a single memory block. Free space in the first block, shown in gray, is reserved for lower layer headers (e.g., the Link layer header), and space in the third block is unused and left empty (as the data payload did not utilize a full two memory blocks but required more than one memory block).

Because unused memory spaces in blocks are wasted, there is a trade-off between the size of the memory blocks and how many blocks are needed for a packet. Protocol designers can adjust the size of the memory blocks to meet their goals. On the other hand, variable block size can be provided to avoid wasted memory if the target operating system provides variable size dynamic memory allocation.

4.2.3 UPS Information-Sharing Interface (*UPS-ISI*)

Information-sharing based cross-layer design has often been proposed for wireless communication protocols due to dynamic radio conditions. UPS supports such cross-layer information sharing by providing the *UPS information-sharing interface* (*UPS-ISI*). Because of the variety of information shared among protocol modules, (e.g., see Table 4.1) it is difficult to manage all of this information in one centralized data storage as proposed in [25] [14].

UPS avoids centralized information storage by not providing storage explicitly. Instead, UPS assumes that the responsibility of information storage is on the provider side, and thus each module independently manages its own information stores. UPS

Table 4.1: Selected information-sharing protocols with their shared information and optimization goals. For more generalized classification, see [1, Table 1].

Model	Shared Information			Optimization Goals		
	App	Net	Link/Phy	App	Net	Link/Phy
Sichitiu [70]		***Schedule***: The actions and times of Application and MAC layers, ● *Sample*, Application layer generates data sample and MAC turns off radio power. ● *Transmit*, MAC turns on and transmits a packet. ● *Receive*, MAC turns on and receives a packet. ***Priority***: MAC sends data packet immediately. Control packet uses RTS/CTS in MAC to guarantee transmission and has lower priority than data packet.	***Exception***: Route table reset if collision occurs, packet lost, and Synchronization fails		Reduce buffer size because schedule reduces packet latency.	Link schedules to eliminate idle listening, packet collision, and delay.
SP [71]	***Urgency bit***: This bit in the packet notifies the Link layer to treat the packet as higher priority. ***Reliability bit***: This bit in the packet notifies the Link layer to acknowledge the packet and retransmit in case of failure.	***Schedule***: The MAC schedules are set to listen, receive, transmit, and sleep. ***Neighbor table update***: Routing cooperates with MAC layer to maintain neighbor table.	***Neighbor table update***: Updates to the neighbor table on different Link layer connections. Updates include address of neighbor, link quality, and scheduling information. ***Message pool***: Control feedback to Network layer, indicating the next sent packet in Network layer. ***MDU***: Link's maximum data unit of a specific Link protocol. ***Congestion status***: Feedback to Network layer for routing decision making.	Providing options of urgent packet and reliable communication.	Better packet buffer control by using message pool.	Using Link schedules to eliminate idle listening, packet collision, and delay.
RF [72]	***Data generation frequency***: Used for computing routing cost.	***Number of descendants***: Descendants in the routing tree for MAC to decide duty cycle.	***Radio duty cycle***: Used in computing routing cost.		Route selected to reduce routing cost.	Adapt radio duty cycle according to number of descendants and neighbors' duty cycle.
XLM/X-Lisa [14]		***Node location*** and ***buffer size***: For decision making in Link layer packet contention.	***Congestion status***: Allows Application layer to adjust the data generation rate.	Adjusting of data generation rate to avoid congestion.		Using Network layer information in packet contention decision.

simply provides a common interface for accessing the information stores of other modules. Modules offering information must thus provide the storage for their data, yet allow other modules to be able to access these data stores through the *access* interface of UPS-ISI. When a protocol is unloaded from the stack, the corresponding data storage will be removed as well, and thus there is no wasted storage space for unnecessary protocols.

Furthermore, we refine the concept of centralized information-sharing services by separating the concepts of "information" and "service". Services should be self-sufficient protocol stack modules working side-by-side with other stack modules instead of integrated into a centralized unit outside the protocol stack or integrated within the protocols themselves, and information should be retrieved from service modules through a unified interface. Thus information exchange can be imagined as inter-process communication in an operating system, where processes (services, protocol modules) get information directly from each other through well-defined interfaces with low overhead. UPS provides such a unified means for information exchange among the protocol modules through *UPS-ISI*. On the other hand, inter-node information exchange is implemented as separate services accessed via *UPS-PI*, enabling the modules to easily exchange information as required by cross-layer protocol designs.

For example, our Location Service module implementation has a location lookup hash table for fast location inquiry, and can be accessed by any module through the following pseudo-code:

```
access( InformationID, Operation, Methods, Argument ),
where:
   InformationID: ID of the Location Service Table
   Operation: Get or Set
   Methods: Location, Distance, Add or Remove
   Argument: a memory space where the first two bytes is the hash
             key, which is the target node address, followed by
             space for a return value
```

Similarly, as will be introduced in Section 4.3.1, our implementation of the RBMulticast [17] protocol maintains a multicast group table using a two layered double-list, which can be accessed by any module through the following pseudo-code:

```
access( InformationID, Operation, Methods, Argument ),
where:
   InformationID: ID of the RBMulticast group table
   Operation: Get or Set
   Methods: Look Up Node, Add or Remove Node
   Argument: a memory space that has a node address as the first two
             bytes and space for a return value following the node
             address
```

Although our examples show only *get* and *set* operations, in general, access control is necessary for information data storage. In addition, a publish/subscribe mechanism can be implemented to avoid the constant polling of information provider modules. Protocol modules act as subscribers, registering and unregistering from the information providers. The information providers act as publishers and maintain the list of subscribers. The information providers send messages with updated data to the subscribers. Both the publish and subscribe functionality can be supported through the access interface:

```
access( InformationID, Operation, Methods, Argument ),
where:
   InformationID: ID of the information of interest
   Operation: Pub or Sub
   Methods: Update, and Reg or UnReg
   Argument: a memory space for the information
```

The idea behind *UPS-ISI* is that the interface should be simple, with the data structure and control complexity left to the protocol designers. UPS provides the protocol switch and information sharing architectures but not the protocol switch decisions. For example, a system that uses a mix of CSMA/CA and TDMA MACs needs tight coordination. Via *UPS-ISI*, vital information like the TDMA schedule, CSMA/CA backoff status, and queue length can be easily obtained by the other MAC protocol, enabling the designer to develop the protocols to achieve the desired coordination.

As we show by the location lookup hash table and RBMulticast group table examples, *UPS-ISI* is general enough to support a variety of different types of information, each of which is identified via a globally unique ID (Information ID). A module only needs to know the particular “Information ID” of the information store it wants to access, as well as the “Operations” and “Methods” supported by that information store. The unique Information ID values can be assigned using the same method as for Protocol ID assignment.

4.2.4 Power Manager

Duty Cycling is a common strategy for energy conservation in many wireless MAC layer protocols, especially for wireless sensor networks. However, in the study of UPS, we observed that future energy conservation in wireless networks will likely include the joint optimization of different protocol layers. Use of a commonly employed randomized *Duty Cycle* will not meet future requirements. Furthermore, current energy conservation techniques in wireless networks generally focus on optimizing a specific protocol or specific application, and they inevitably have conflicts between different approaches. Thus a centralized *Power Manager* is urgently needed, with different protocol modules acting as *clients* of the *Power Manager*. UPS provides access to such a *Power Manager* via an interface with the following functions:

```
sleep_request(): ask the Power Manager to
                 turn off the radio
wakeup_request(): ask the Power Manager to
                  turn on the radio
awake(): called by the Power Manager at
         the moment power is turned on
```

`Sleep_request( )` and `wakeUp_request( )` are called by clients to inform the *Power Manager* of a request to change the radio state. These requests are considered in a centralized way by the *Power Manager* for different purposes (e.g., power conservation, least delay, reduce network traffic). `Awake()` is a function of the clients called by the *Power Manager* for the case when clients need to know that there was a transition of the radio state (e.g., the MAC starts sending a packet immediately when the radio power is turned on).

We designed a policy-based *Power Manager* in our UPS implementation (although any type of *Power Manager* can be used in UPS). The policy that we implemented follows these guidelines: a) turn off the radio as much as possible, and b) if any of the client protocol modules request a wake-up at a given time, then turn the radio back on. This policy-based *Power Manager* is used in our Link layer protocol design. In our implementation of the MAC protocol XLM/MAC, we made the communication behave as if it were duplex by abstracting the MAC send and receive functions as two independent components, with both acting as clients of the *Power Manager*. Under this setting, the radio power will only be turned off if both are not transmitting or receiving (from the radio or the upper layer protocol), which conforms to the intuition of how to transmit packets correctly.

4.3 UPS Architecture Case Studies

To show that the proposed UPS architecture is beneficial to real world applications, it is evaluated through both physical experiments and simulations. Meanwhile, two different networking paradigms are investigated to indicate the wide applicability of UPS and the potential gains achieved in different networks. A wireless sensor network (WSN) test-bed is formed that implements the UPS architecture on the Tmote Sky wireless sensor nodes [73]. To achieve this, the UPS architecture is implemented in TinyOS. A realistic scenario is defined where two cross-layer routing protocol modules are executed simultaneously, XLM [15] proposed for unicast data transmissions and RBMulticast [17] proposed for multicast data transmissions. In addition, a mobile ad hoc network (MANET) is simulated using the OPNET simulator [74]. In this scenario, two well-known routing protocols, AODV [33] and IP, are run simultaneously using UPS in a network with both static and mobile wireless devices.

4.3.1 TinyOS Experiments for UPS with Wireless Sensor Networks

In this case study, we implemented UPS in TinyOS [11], a widely used sensor node operating system. UPS is used to build the protocol stack shown in Fig. 4.5. XLM/Net and XLM/MAC are the Network layer and Data Link layer modules extracted from the cross-layer protocol XLM [15], which is a hand-shake receiver-based unicast protocol. The XLM/MAC controls the CC2420 radio chip and records network statistics into the information store called XLM Statistics, which provides experimental data collection that is then accessed by the Application layer through the *UPS-ISI*. RBMulticast [17] is a Network layer multicast protocol that also makes use of XLM/MAC to provide multicast communication. RBMulticast stores information about the multicast members in an RBMulticast Group Table, which is accessible via the *UPS-ISI access* function. An MPBuf Message Pool provides dynamic memory access to packet storage space for all stack protocols. The Host Addr module manages different host addresses for different protocols. The Location Service module in the Network layer provides location

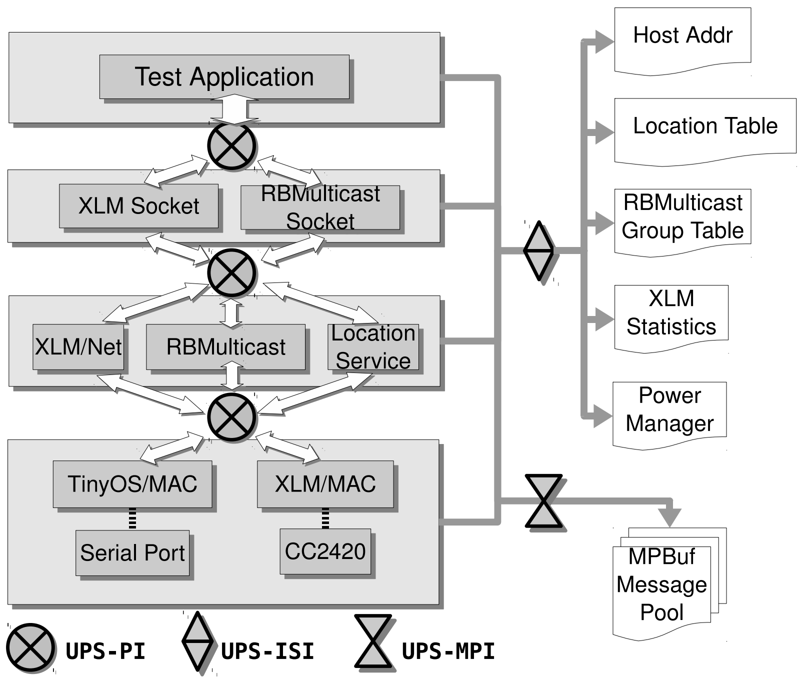
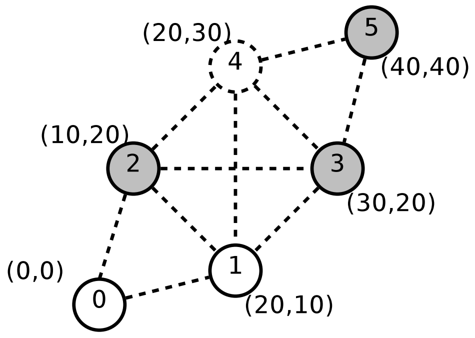


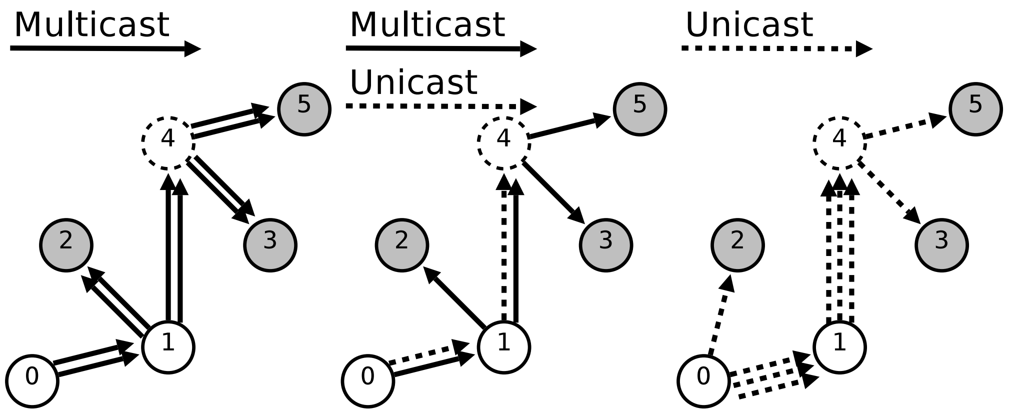
Figure 4.5: Block diagram of the UPS framework and protocol modules developed for the WSN environment.

information, which is stored inside a Location Table, accessible via the *UPS-ISI access* function described in Section 4.2.3. An additional MAC protocol, TinyOS/MAC, sends data from the serial port to a computer for data collection. Finally, we implemented a CC2420 Power Manager, which is a power manager that turns off the radio as much as possible when the radio is not used.

We implemented UPS and all the protocols in TinyOS, and we tested the performance of the protocols using both the TOSSIM emulator as well as an implementation on Tmote Sky motes. In the simulations and experiments, six sensor motes are deployed in the network as shown in Fig. 4.6(a). The locations of these motes are shown in parentheses as X and Y coordinates. The mote parameters are set to give a radio range of 30m, and dotted lines in Fig. 4.6(a) indicate possible connections between motes. Mote 0 sends both unicast and multicast packets and thus plays the role of a source node. Unicast mote 4 receives unicast data from the source node, and the three multicast receivers, motes 2, 3, and 5, receive multicast data from the source node. For easier explanation and comparison, node 4 is called the *unicast node* and node 3



(a)



(b) MC (c) MU (d) UC

Figure 4.6: a) The experimental network in this chapter. The dotted node is the unicast destination, and the shaded nodes are multicast receivers. b) MC: Multicast only, c) MU: Mixed Multicast-Unicast, and d) UC: Unicast only.

is called the *multicast node* in the following discussions. The remaining mote 1 does not require data from the source node and simply acts as a router in this network. The source node sends one unicast and one multicast packet, separated by a long period of time to guarantee no congestion in the network. This sequence of unicast followed by multicast packet transmissions is repeated 20 times in every simulation.

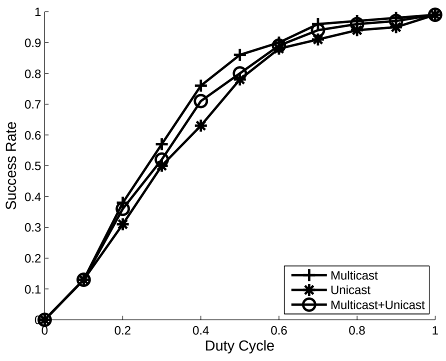
The number of transmitted packets in the following results includes all MAC layer packets (e.g., RTS, CTS, DATA, ACK), and the success rate and the packet delay are calculated in the application layer of the sender and the receiver. Duty cycle is calculated as the duration of time the sensor's radio is turned on in each 100 ms duration. For example, a duty cycle of 0.2 (20%) means that in every 100 ms, nodes will turn their radios on for 20 ms and then go to sleep for the remaining 80 ms if not transmitting/receiving; otherwise, motes will turn off their radios after completion of the transmission/reception.

4.3.1.1 TOSSIM UPS Simulations

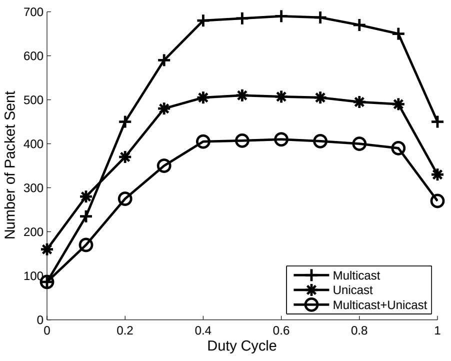
In this experiment, we aim to show that the UPS framework enables independent execution of multiple protocols in the same stack layer and does not introduce any interference between different protocols. We define three scenarios. In the first scenario, motes can only run the multicast protocol (i.e., RBMulticast), and hence unicast communication must be done using the multicast protocol. We set all unicast and multicast destination motes to be in the same multicast group, and the Application layers in the sensor motes have a filter that only accepts specific packets. This means that all packets still need to be received and handled by the application layer. In the second scenario, motes can run both the unicast and the multicast protocols. In the last scenario, motes can only run the unicast protocol, and multicasting is done by consecutive unicast transmissions. Packets to the unicast destination will be sent prior to the three consecutive unicast packets for the multicast destinations. All packets are separated at least 20 seconds to ensure there is no congestion in the network under low duty cycles. Some possible routes for these three cases are shown in Fig. 4.6.

All results in this section are the averages of 40 simulations. We show the overall success rate, the number of packets transmitted, and the average packet delay observed in Fig. 4.7. As seen in Fig. 4.7(a), the success rates of the three protocols only differ slightly, which indicates that this performance metric is dominated by the performance of the MAC layer, since both XLM/Net and RBMulticast rely on the same MAC protocol, XLM/MAC. Similar success rate results also show that the UPS-enabled concurrent execution of multiple protocols introduces no interference in protocol executions. The packet success rate values shown in Fig. 4.7(a) are observed to be correlated with the nodes' duty cycle, since the number of retransmissions are limited and the lower the duty cycle the less likely a packet will be successfully delivered.

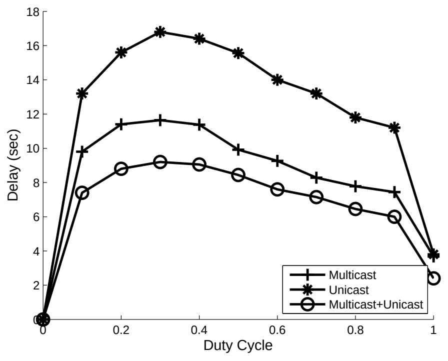
The total number of packets sent by all six nodes are shown in Fig. 4.7(b). The results clearly show the benefits of using a UPS-enabled concurrent protocol approach, where the total number of packets sent for the UPS approach is less than the other two



(a) Success rate.



(b) Total number of packets sent.



(c) Average packet delay.

Figure 4.7: Statistics of the experimental network in different scenarios.

single protocol execution cases. The non-zero numbers of packets sent when the duty cycle is zero in all three cases are due to the fact that the source sends RTS packets but all other nodes are always sleeping. Our XLM/MAC implementation sends RTS packets four times before dropping the packet.

The average total packet delay observed is given in Fig. 4.7(c). The delay results again show the same characteristics due to using the same MAC layer, but with some drifts due to the different Network layer routing protocols, where multicast packets have more delay than unicast packets. We do not count the delay time of the interval between consecutive packets. Otherwise, the unicast only scenario would have excessive delay overhead.

These results validate that when co-existing multicast and unicast protocols share the same MAC protocol module, their packet success rates are the same as the cases where only a multicast or a unicast protocol exist, which provides strong evidence that there is no disruptive interference among the protocols from the UPS protocol stack. This means that UPS supports highly modularized, interference-less interfaces for protocols and provides an efficient infrastructure for multi-functional WSNs.

4.3.1.2 Performance of UPS on Tmote Sky Motes

In this experiment, we consider the same three scenarios of Section 4.3.1.1, but reduce the separation time between packets to introduce congestion into the network, which better represents a practical scenario found in real applications. We measure the data from the Tmote Sky motes and compare to TOSSIM simulations, both running the same code to verify that UPS introduces low overhead and matches the requirements of resource limited wireless sensor networks.

In the first scenario, motes can only run the multicast protocol, with all unicast and multicast motes in the same multicast group and the application layers in the sensor motes filtering packets so as to only accept specific packets. In this congested scenario, two consecutive multicast packets, separated by one second, are sent every 5 seconds. This means that both packets will exist in the network for some period of time. In the second scenario, motes can run both the unicast, and the multicast protocols and the multicast packet will be sent 1 second after the unicast packet, with 5 seconds between the unicast packet transmissions. In the last scenario, motes can only run the unicast protocol, and multicasting is done by consecutive unicast transmissions separated by 100 milliseconds. Packets are sent to the unicast destination prior to packets being sent to the multicast destinations in all three cases. The experimental results using both Tmote Sky motes and TOSSIM simulations are shown in Fig. 4.8.

Fig. 4.8(a) presents the packet success rate results. The average success rates are still roughly the same for all scenarios, however larger variations are observed due to packet congestion. As seen in the figure, the success rate is more closely correlated to the duty cycle than in the previous simulations. The similar results of all three scenarios show that the common MAC protocol used is dominant in this performance metric. The results of the Tmote Sky motes and the TOSSIM simulations are also shown to be very close, which validates the simulations performed.

As shown in Fig. 4.8(b), the UPS-enabled mixed multicast-unicast approach still

Table 4.2: The overhead of context switches in TinyOS (and hence UPS) and of sending a 36 byte packet using the Tmote Sky motes. This table is a modified version from P. Levis [2, Fig. 10]. Packet time for sending a 36 B packet is measured from our experiments.

Overhead	**Time** (clock cycles)
Interrupt Switching	8
Interrupt Handler Cost	26-74
Task Switching	108
Sending a 36B Packet	32,258

requires fewer packets than the other two cases, revealing the advantages of using UPS-enabled multiple protocols for complex tasks. On the other hand, the unicast only approach does not achieve better performance than the multicast only approach as opposed to the scenarios evaluated in Section 4.3.1.1. This is because the congestion in the unicast only case is more severe compared with the scenarios evaluated in Section 4.3.1.1, as more packets are in the network at a given time.

The experimental results of Figs. 4.8(a) and 4.8(b) show almost identical behaviors of UPS in simulation and in real sensor nodes. This gives us strong confidence that UPS introduces low overhead on hardware requirements, preserves the independence of the protocols executed concurrently, and hence, is truly an applicable framework for protocol design in wireless sensor networks.

4.3.1.3 Overhead

One chief concern of the UPS stack model is the additional overhead among module calls (or *context switch* overhead in OS terminology) for supporting multiple protocols. Generally this overhead is insignificant compared to the cost of sending a packet in WSNs. For example, the context switch overhead and packet transmission time for TinyOS (and hence UPS) are given in Table 4.2. The hundredfold difference means that the cost of module calls in UPS is negligible compared to cost of the communication. UPS can even achieve near zero overhead under the TinyOS environment. This is because the UPS interfaces consist of groups of interface functions, and TinyOS uses *procedure inlining* by default to remove (expand) small function calls. All the indirect calls through UPS are changed into direct calls in the final execution code, and hence

Table 4.3: Comparison of execution code size.

	ROM Usage (Bytes)
Dummy TinyOS App	1398
Dummy TinyOS App (UPS)	1608
XLM (Single Layer)	25,252
XLM (UPS)	29,808
TinyOS App (BlinkToRadio)	11,516

the context switch delay is minimized.

Another overhead of interest is the code size. For the purpose of comparison, the code size for different TinyOS programs with and without UPS are listed in Table 4.3. The dummy TinyOS application is a primitive application that only counts OS overhead (e.g., modules without radio communication). The second program is the dummy application equipped with the UPS interfaces. The resulting code size is only 210 B larger. Single layer implementation of XLM [15] in TinyOS is found to occupy 25,252 bytes of ROM, whereas the XLM protocol within the UPS architecture is found to occupy 29,808 bytes of ROM. As expected, since UPS provides the message pool structure and more sophisticated services, it requires more memory space, however the increase is not prohibitive for most applications of UPS. As a reference, the basic radio application (BlinkToRadio) in TinyOS requires 11,516 bytes of ROM. Considering the multiple protocols implemented, the relative overhead of UPS will be much smaller.

4.3.2 Simulation Results for UPS with MANETs

The aim of this case study is to show that the UPS framework can be applied to off-the-shelf protocols with no interference between different protocol modules in the same layer. We consider a scenario where a mobile ad hoc network and a static wireless network are co-located as shown in Fig. 4.9. The nodes that are represented as squares are static, and the nodes that are represented as circles are mobile nodes. Two applications are considered to be running on the central node, where the first application requires multicast data transmission to a subset of static nodes that are defined as the static destinations (*SD*s), and the second application requires unicast data transmissions to a subset of mobile nodes that are defined as the mobile destination nodes (*MD*s).

Considering two common routing protocols used for static and ad hoc networks, namely IP and AODV, for this specific scenario, there are two possible stack implementation alternatives without UPS: The first one is to design a new routing protocol to incorporate these two routing protocols, which is not acceptable, since it is not feasible to develop a new routing protocol for every possible case. The second possibility is to require all nodes to run IP or to run AODV. We will show, however, that neither of these approaches are ideal. Note that running two separate routing protocols is infeasible without an approach like UPS, since a received packet from the MAC protocol cannot be directed without an explicit indication of the routing protocol.

In UPS, the solution is simply to run both protocols at the same time. Hence, we investigate three protocol stack approaches: a TCP/IP/802.11 stack, an AODV/802.11 stack, and the UPS-enabled stack with two concurrent routing protocols with a common 802.11 MAC protocol, as shown in Fig. 4.10. The Bypass module is included because we assume no Transport layer protocol for AODV.

Simulations are performed for 16 static nodes, among which 4 nodes are defined as *SD*s, and 32 mobile nodes, among which 4 nodes are defined as *MD*s. The three different stacks are evaluated as follows. In the first case, the source node has UPS-enabled TCP/IP and AODV co-existence, where multicast communication to *SD*s is done through IP multicast and the unicast communication to *MD*s is done through AODV. In the second case, all nodes, including the source node, run the TCP/IP/802.11 stack. In this case, the communication to *SD*s is done though IP multicast, and the communication to *MD*s is done though multiple IP unicasts. In the third case, all nodes can only run the AODV/802.11 stack, where both the static multicast communication to *SD*s and the mobile unicast communications to *MD*s are done by individual AODV unicast packets. The comparison of the stacks are done for two performance metrics: success rate of the packet transmissions and the total traffic load generated in the network.

The simulation is run for a 1 hour period, with all results being the average over this 1 hour. The source node sends one packet/s to the *SD*s and one packet/s to each of the *MD*s. The mobile nodes follow the Random Waypoint Mobility (RWPM) Model where the mobile speeds are varied among the following intervals: [0, 0-5, 5-10, 10-15, 15-20] m/s. The interval represents the minimum and maximum mobile speeds that are used as the parameters of the RWPM model. The corresponding performance results are shown for the maximum speed of the interval, e.g., 10 represents the interval of 5-10 m/s. The simulation region is a 200m by 200m area, and the radio uses a free space propagation model where the transmission power and the reception power threshold are set to achieve a maximum of three hops distance between the nodes.

The simulation results of the performance under different mobility levels are shown in Fig. 4.11. The average success rate of packets is shown in Fig. 4.11(a), where the UPS enabled multiple protocol execution gives the highest success rates. The reason why the nodes running only the TCP/IP/802.11 or the AODV/802.11 protocol stacks have lower success rate is because IP cannot handle mobile nodes successfully, and AODV cannot handle static nodes successfully. Although both the AODV/802.11 stack and the UPS-enabled dual protocol approach use AODV to unicast packets, the success rates are different. The AODV/802.11 stack performance is down about 10% compared to the performance of the UPS-enabled dual protocol approach. The reason is that AODV sends route requests to all nodes and builds a route via both the static nodes and the mobile nodes. When the mobile nodes move, the routes to static nodes are also broken, thus reducing the success rate of packet delivery even to static nodes. On the contrary, for the UPS-enabled dual protocol case, routing for static and mobile nodes are independent, and thus a static node will not become a relay for a mobile node in AODV.

Fig. 4.11(b) shows the total traffic generated in the network for the three different cases. The AODV/802.11 stack induces a huge amount of traffic because it transmits many control packets for route discovery to static *SD*s. The TCP/IP/802.11 stack, on the other hand, has the lowest traffic because IP assumes a static network, and packets to unreachable nodes are simply dropped without routing discovery overhead. The deficiency of IP is the low success rate of the packets transmitted to *MD*s.

In conclusion, in the simulations, the UPS-enabled multiple protocols approach has the highest packet success rate and a low traffic load for a mixed static and mobile network. This approach benefits from the strengths of both the IP and the AODV routing protocols, that is, low overhead in the static network and high success rate in the mobile network, and hides the drawbacks by the auxiliary protocol. The spirit of the UPS framework is that we do not design a new protocol for the special scenario of mixed static and mobile networks. What a designer needs to do is to choose the right protocols from the existing protocol pool and use the UPS interfaces to glue them all together.

4.3.3 Notes on Implementing UPS for Off-the-Shelf Protocols

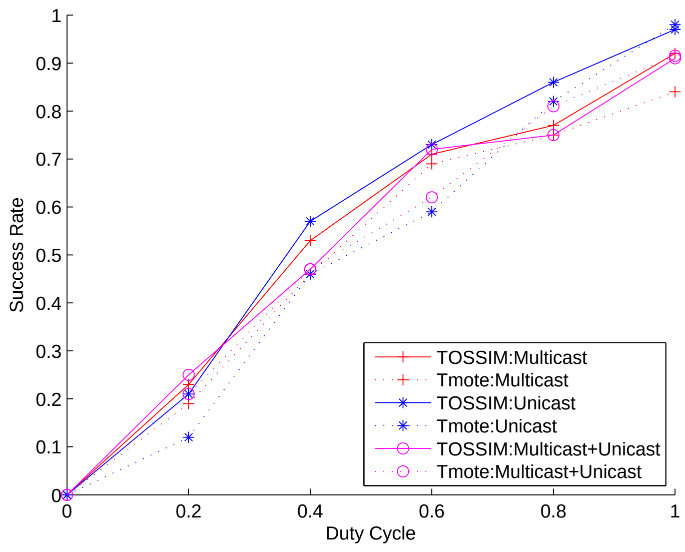
In order to not change the internal algorithms of IP and AODV, we implemented the switch function as additional layers between the OSI layers. The switch layer only appends a Protocol ID to multiplex a packet to the lower layer, and subtracts the Protocol ID to demultiplex a packet to the upper layer. With this approach, all protocol details of the original protocols remain the same.

4.4 Conclusions

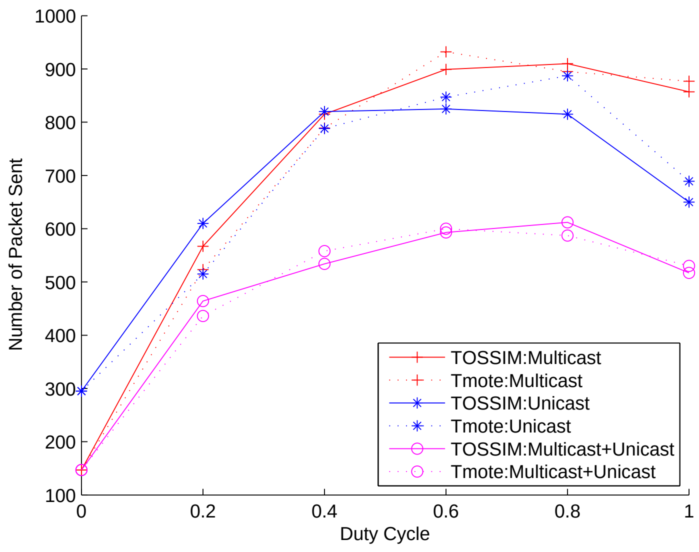
In this chapter, we presented our Universal Protocol Stack (UPS) architecture, which considers the needs of emerging wireless networks, where network efficiency is the primary concern. Previous stack approaches have not considered the possibility of running different protocols in the same stack layer concurrently, combined with cross-layer information-sharing between modules or layers. Our approach, UPS, uses the “Packet Switch” concept to provide support for both of these features observed in emerging wireless networks. The UPS protocol interface, *UPS-PI*, provides packet switches between stack layers that selectively deliver packets to the correct upper (input) or lower (output) target protocol modules, enabling the modules in different layers to seamlessly work together. The design of UPS also considers the need for cross-layer cooperation and hence enables cross-layer information-sharing among different protocol modules through the *UPS-ISI* interface. In addition, a common packet memory structure is presented, namely by the *UPS Message Pool*, to enable multiple protocols to have unified access to internal data of a packet via the *UPS Message Pool Interface* (*UPS-MPI*).

A test-bed was built to investigate the UPS framework on wireless sensor motes. Our physical experiments with a TinyOS implementation of UPS, that are verified by the results of TOSSIM emulations, show that UPS indeed enables the co-existence of multiple protocols in the same stack layer, with very low system overhead, and furthermore UPS can provide the benefit of concurrent operation of multiple protocols and cross-layer information exchange. Thus, UPS has been shown to provide a promising architecture for emerging wireless networks with complex multi-functional applications.

We also implemented the UPS framework in the OPNET simulator, which enabled the simultaneous execution of the off-the-shelf TCP/IP and AODV protocols with an underlying IEEE 802.11 MAC protocol, and the results show that the UPS-enabled stack with *multiple-protocol* layers has excellent performance in a mixed static-dynamic network compared to the conventional TCP/IP/IEEE 802.11 stack and to the AODV/IEEE 802.11 stack. More importantly, while achieving this performance, no modification of the investigated protocols is needed to integrate them in UPS. Hence, UPS provides a generic and universal way to support multiple protocols within a layer, which is an important requirement of emerging wireless networks. Note that, these advantages can be generalized to wired networks.



(a) Success rate.



(b) Total number of packets sent.

Figure 4.8: a) Packet success rate. b) Total number of packets sent by all nodes. This number counts all MAC layer packets (e.g., RTS/CTS/DATA/ACK). Packet delay is not considered here because it is difficult to calculate time difference without synchronization among sensor nodes.

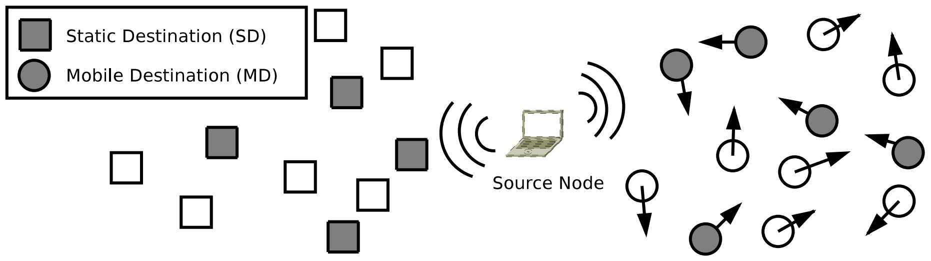


Figure 4.9: The simulation scenario of a mixed static and mobile network.

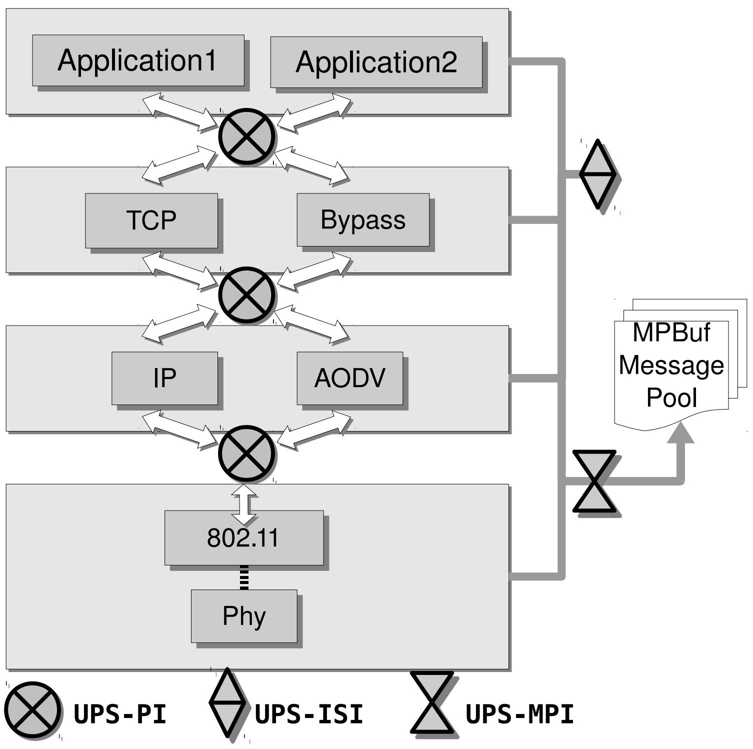
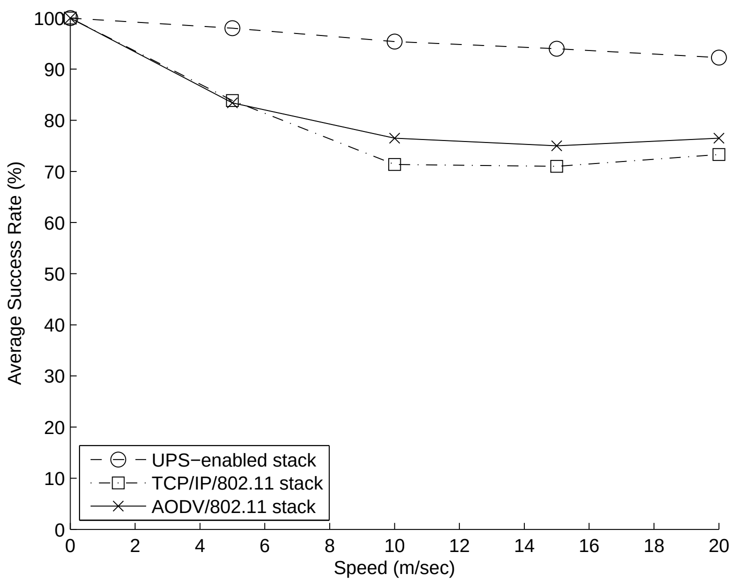
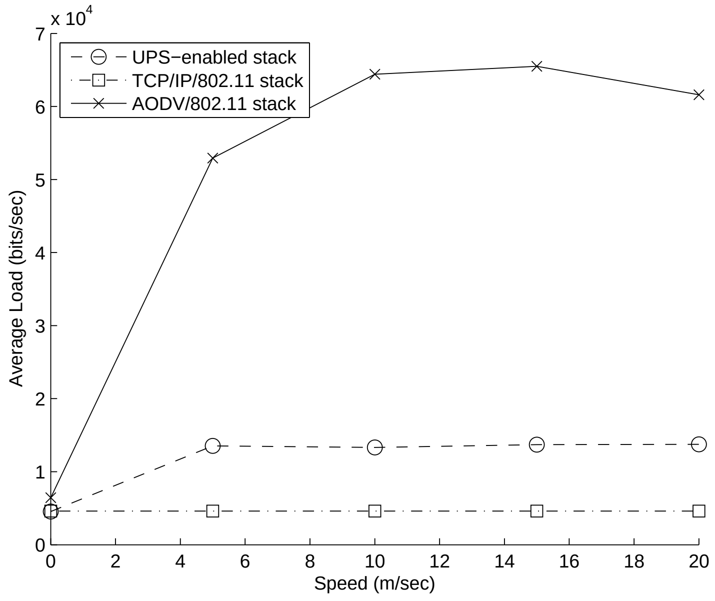


Figure 4.10: The high level system block diagram of the UPS stack consisting of the TCP/IP and AODV ad hoc routing protocols.



(a)



(b)

Figure 4.11: The performance comparison of the three protocol stacks for a) average packet success rate, and b) average traffic load of the network.

Chapter 5

Enabling Heterogeneous Devices using Virtual Interfaces

A majority of existing communication protocols are developed with the assumption of a single radio. While there has been some initial work on protocols that use multiple radios, existing approaches are limited to specific application domains and specific radio interface implementations. In this thesis, we propose a new approach to supporting multiple radio interfaces that abstracts all the available interfaces using a single virtual interface. The selection of the specific physical interface to use per packet is done by the virtual interface, thus ensuring that no modifications of the upper layer protocols are required. This provides the opportunity for algorithms at the virtual interface to optimize the selection of the physical interface to improve the network performance. To test the virtual interface approach, we evaluate scenarios with multi-radio devices that support LTE, WiFi, and a CSMA network through simulations in ns-3. Results from these simulations show that the use of a virtual interface is feasible and can improve the network performance. Different interface selection algorithms as well as the limitations of the virtual interface approach are discussed.

5.1 Introduction

Since packet radio networks were introduced, numerous wireless communication protocols have been designed to enhance network efficiency. However, a majority of the proposed protocols are developed with the assumption of single radio devices. For instance, MAC protocols IEEE 802.11 [75] and IEEE 802.15.4 [76] assume each device has exactly one radio interface. Similarly, routing protocols such as AODV [33] and Directed Diffusion [77] also assume one radio interface for each node, and thus the adaptations of these protocols to multi-radio devices are not trivial.

Recent research has proposed the use of homogeneous multi-radio devices to increase connectivity, where all devices have the same set of radios. For example, multi-radio solutions are widely recognized as the standard approach for wireless mesh networks [5, 78, 79]. In these multi-radio scenarios, the approach is to use channel assignment for all of the radio interfaces in order to reduce transmitter/receiver interference [80] [81]. While these approaches use multiple radios, nevertheless, all the results are limited to specific application domains and specific radio interface implementations.

Recently, with the prevalence of wireless hand-held devices, several communication standards including GSM, CDMA, LTE, WiFi, Bluetooth, ZigBee, and RFID have been developed. Most mobile devices, such as cellular phones, tablet computers, and laptop computers, are manufactured with many of these heterogeneous radio communication substrates, and the challenge of managing these multi-radios has begun to receive attention. A common method to enhance the connectivity of current multi-radio devices is to provide a prioritized list of networks to access. For example, when a user is within WiFi coverage, the multi-radio device uses WiFi for connectivity, and only if WiFi coverage is not available, then the device switches to a cellular network [12]. As another example, 4G introduces the concept of Always Best Connected (ABC) [8], which claims to integrate different radio access techniques, such as 2G, 3G, WLAN, WMAN, etc., into a common network.

Many innovative projects have started to develop solutions for supporting these heterogeneous radio networks. For example, multiple parallel MAC protocols [82] [83] and routing protocols [84] can be adaptively selected according to the network conditions. However, current methods require architectural modifications of the existing communication protocols.

In this thesis, we propose an approach that abstracts all the available interfaces using a single virtual interface. Since we assume that all the interfaces are connected to the same IP network, no modification to the layer 3 (L3) routing protocol or layers

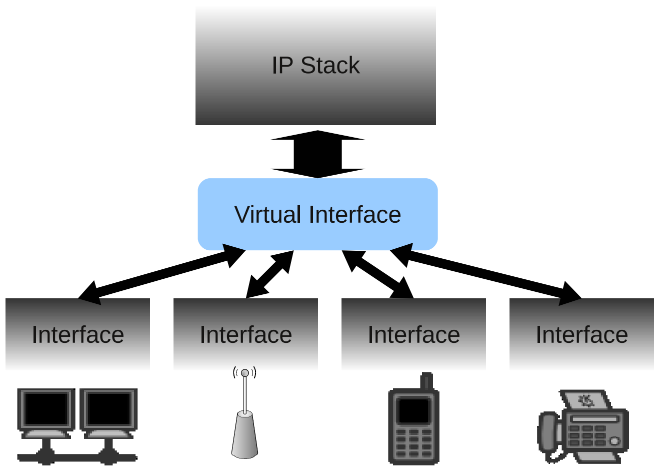


Figure 5.1: Virtual interface block diagram.

above is required, and packets can be seamlessly transmitted from any of the available interfaces. We evaluate a scenario with multi-radio devices that support LTE, WiFi, and a CSMA network through simulations in ns-3 [85]. The key technique we use is to expose a virtual interface to control all the physical interfaces, with the virtual interface controlled by the IP layer. Because of this additional virtual layer, the selection of the physical interface is completely hidden from the IP layer, and the actual physical interface selected has no impact on the functionalities of the upper layer. This provides the opportunity for performing smart physical interface selection at the virtual interface to improve the network performance.

The rest of this chapter is organized as follows. We describe the virtual interface system model in Section 5.2, and we present the results compared with using a single interface in Section 5.3. Section 5.4 discusses an interface selection algorithm that uses a weight function. Adaptive random selection schemes are discussed in Section 5.5. The use of a virtual interface with TCP traffic is described in Section 5.6. Limitations of the virtual interface approach are discussed in Section 5.7. Conclusions are provided in Section 5.9, and future work is presented in Section 5.10.

5.2 The Virtual Interface

The virtual interface is proposed to enable the aggregation of all the available interfaces to a single virtual interface. Instead of the conventional solution, where all the physical interfaces are directly exposed to the layer 3 (L3) routing protocol, the proposed virtual interface disconnects the existing physical interfaces from L3. Fig. 5.1 shows the virtual interface system block diagram. The virtual interface provides access to the physical interfaces and seamlessly incorporates all of the physical interfaces into the network. All of this can be achieved without user intervention. The Internet stack requirements and the packet send/receive flow will be discussed in this section.

5.2.1 Internet Stack

The physical interfaces controlled by the virtual interface do not need to be customized. Any layer 1 (L1) and layer 2 (L2) interfaces that are compatible with the L3 IP protocol requirements can be used as a target interface. We call the physical interfaces controlled by the virtual interface as *client interfaces*. Only the virtual interface is assigned an IP address, the client interfaces do not have individual IP addresses. We assume that all the target nodes are equipped with the same set of client interfaces. The impacts of this assumption will be discussed later in Section 5.7.

The protocols above L3 consist of the TCP/UDP protocols, which implies that ARP/IPv4/IPv6 are used by all the nodes in the L3. Since TCP/UDP and the application layers are above L3, from the L2 point of view, they are irrelevant to the virtual interface. Any protocol that is compatible with the Internet stack can be used with the virtual interface without modification. In our ns-3 implementation, the entire Internet stack and physical interfaces are used as is from the ns-3 code repository.

5.2.2 Sending Packets

When the virtual interface receives an outgoing packet from the L3 layer, a client interface must be selected from the candidates. The selection method can use information from the physical interfaces to make this decision. The procedure for sending a packet is summarized by the pseudo code in Algorithm 1.

Algorithm 1 Send via a Virtual Interface

```
Input: Packet from upper layer
Output: Packet inserted to a client interface
1: Select an interface $i$ according to
       • Available interfaces
       • Available information
2: Send the packet to interface $i$
```

In the Internet protocol stack, only minimal information is shared between layers. For example, the only information shared between the L2 interface and the routing layer is the packet itself and the destination address. Unless there are custom modifications, such as using link quality at the routing protocol, no information such as packet success or delay is available to the routing protocol from the L2 interface. Therefore, for the typical case of using the client interface on a device as is, i.e., without modification, we expect that the virtual interface cannot obtain any extra information from the client interfaces.

Due to this lack of information, the best interface selection method that the virtual interface can implement is to evenly distribute the load to all the available client interfaces. We refer to this as *uniform random selection method* in this chapter. This can be improved if some form of user input is available or modifications of the client interfaces are allowed. In Sections 5.4 and 5.5, we will discuss how we use extra information to design intelligent client interface selection algorithms.

We assume that all the L2 MAC protocols are tightly coupled to the corresponding radio devices. Therefore, the selection of a specific client interface means the selection of the corresponding L2 MAC protocol and the L1 physical device.

5.2.3 Receiving Packets

Since we assume all nodes are running IP protocols with the Internet stack, all the incoming packets are in the same IP format. Therefore, received packets are simply forwarded from the client interface through the virtual interface to the L3 layer.

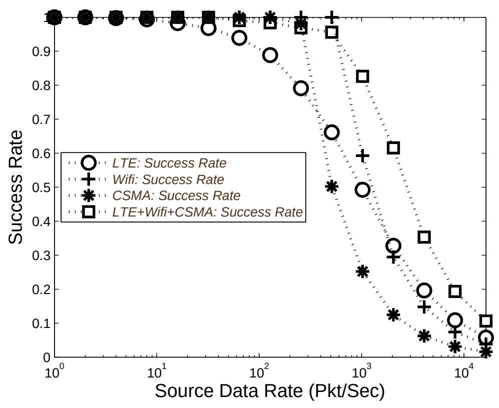


Figure 5.2: Success rate using three interfaces compared with one interface. WiFi is configured with OFDM rate 6 Mbps and CSMA is configured with channel bandwidth 5 Mbps and delay 2 ms.

5.3 One vs. Multiple Interfaces

We first consider the experiment of using multiple interfaces with the uniform random interface selection method compared with using a single interface. Intuitively, as more interfaces leads to higher overall bandwidth, throughput should be increased compared with using only a single interface.

Using ns-3, we simulate a scenario with a pair of source-destination nodes in the network. IPv4 is used as the L3 protocol, and the source application sends UDP traffic with Poisson distribution. Note that, with this setup, packet success rate is directly proportional to the resulting throughput because there is no interfering traffic.

Fig. 5.2 shows the success rates when using one client interface versus three client interfaces, LTE + WiFi + CSMA. As expected, the packet success rate is higher when more interfaces are used in the communication. From the results, we can see that the success rates are virtually the same under low traffic. This implies that if the target application is for a low traffic network, using multiple interfaces does not necessarily increase the network performance, although there is no harm in terms of success rate in using more interfaces.

Fig. 5.3 shows the application layer's average delay from the simulations. The results indicate that average delay increases while the packet success rate increases with multiple client interfaces. The LTE only scenario has lower delay than using multiple

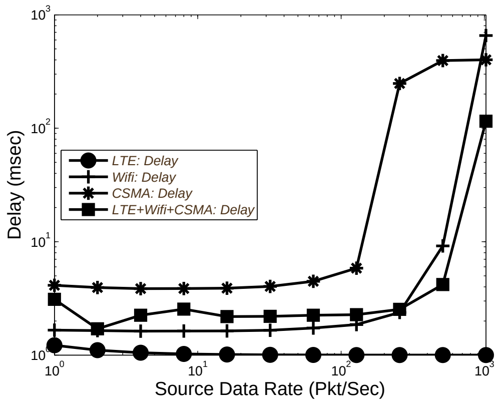


Figure 5.3: Delay using three interfaces compared with one interface.

interfaces. The reason for the lower delay in the simulation is that LTE uses 1 ms time frame for Transmission Time Interval (TTI), and hence the delay is a constant 1 ms for all source data rates, plus a small delay of routing overhead (ARP packets). LTE in the ns-3 implementation does not support burst packets, and packets are dropped if multiple packets are waiting to be sent in a TTI frame. Dropped packets are not included in the delay calculation.

5.4 Weighted Random Interface Selection

A natural enhancement of the uniform random interface selection method is to weight the probability of selecting a particular client interface using input from the user. In this experiment, we choose the expected channel capacity as the weight, so higher bandwidth client interfaces will receive a higher portion of the traffic load. The experimental throughput value of maximum packets per second are used as the weights of the interfaces:

	LTE	WiFi	CSMA
Weight (pkt/s)	508	604	256

Note that the actual throughput depends not only on the channel bandwidth but also characteristics of the protocol, for example, back-off time, control packets (e.g., RTS, CTS, and ACK packets), and queue length. In our simulations, all the protocol parameters are set to the ns-3 default values.

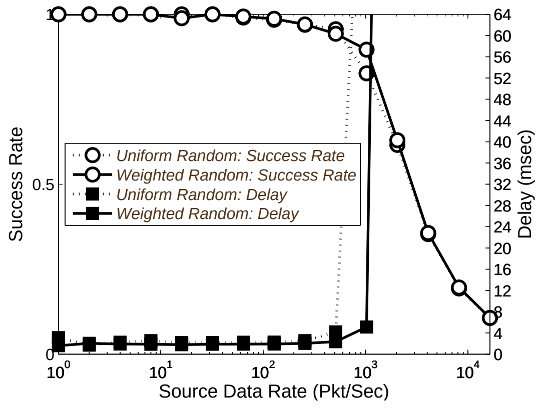


Figure 5.4: Success rate and delay using three interfaces and the weighted random selection method compared with the random selection method.

The procedure of the weighted random selection method is summarized by the pseudo code in Algorithm 2.

Algorithm 2 Weighted Random Interface Selection

```
Input: Available client interface list $I$ and list of weights $w_i$
Output: Interface $i \in I$ is selected
1: Calculate $W_{total} = \sum w_i$
2: Select interface $i$ with probability $p_i = \frac{w_i}{W_{total}}$
```

The LTE client interface in our experiment transmits packets via an up-link channel, from a user node to a base station. This is a lower throughput channel compared with the down-link channel, and it is slower than the WiFi channel in our environment. Fig. 5.4 shows the simulation results. Although the user input channel capacity provides direct information about the client interfaces' performance, the packet success rate is only marginally increased. The reason is that there is a non-linear relationship between the application input data rate and the success rate. The success rate is 100% for low data rate values, however, it suddenly drops after a threshold data rate. This indicates that using a constant weighted selection algorithm is not necessarily helpful in making the client interface selection decision, especially considering that the user may enter incorrect weight information. This constant weight method is also vulnerable to changes in channel conditions, which will be considered in Section 5.5.

The resulting delays for the uniform random and weighted random selection algorithms are also shown in Fig. 5.4. We can see that weighted random selection of the client interfaces decreases the delay. The reason for the lower delay in the simulation using the weighted random selection algorithm is because more packets are sent via LTE (since it has a higher weight than CSMA), the packet delay is reduced.

The resulting delays in Figs. 5.3 and 5.4 show that optimization of more than one metric is much more difficult for heterogeneous interfaces. For example, no set of weights can provide both the best success rate and delay at the same time in these experiments. The optimal weights for delay are (1, 0, 0), i.e., all packets are sent through LTE, but this will result in lower packet success rate, as shown in Fig. 5.2.

5.5 Adaptive Random Interface Selection

As discussed in Section 5.2.2, uniform random selection is the best approach we can use without supplying user input or modifying the client interfaces to provide extra information to the virtual interface. From Section 5.4, we see that user input may improve the performance, but using constant weights suffers from any changes in channel conditions for the different client interfaces.

In this section we propose an *adaptive random interface selection method* to overcome these problems. In this approach, information is extracted from client interfaces to calculate the weights for each client interface, and the weights are adjusted with changes in the run time client interface conditions. This algorithm uses channel congestion information from the client interfaces to adaptively balance the traffic loads. For system support of information sharing, the Universal Protocol Stack (UPS) [86] is selected as the infrastructure for the ns-3 implementation.

For the LTE interface, because the ns-3 implementation does not have an ARQ Indicator Channel [87], there is no NACK to identify packet losses. In order for the virtual interface to obtain packet drop information, a callback function is added to the LTE protocol to inform the sender that a packet was dropped.

For the WiFi interface, a notification event is added when the sender node performs an RTS retransmission, as this indicates a packet collision for one of the RTS/CTS/DATA/ACK packets.

For the CSMA interface, a notification event is added when the sender detects the channel to be non-idle while trying to send a packet. The packet will be assigned to another back-off slot and retransmitted.

Each interface uses a counter to record the number of events of that interface. This counter is used to calculate the runtime weight of the interface, and the resulting weight is used for the interface selection probability calculation. The higher the count, the lower the probability of that client interface being selected, so a congested client interface will receive less traffic from the virtual interface. We also use the TCP-analogous moving average technique that decreases the count over time, so the lower weight interface will start receiving more traffic until congestion finally occurs again. The procedure of adaptive random selection is summarized by the pseudo code in Algorithm 3. Default weight $w_{i_{def}}$ in Algorithm 3 is the weight for the weighted random interface selection provided by the user, as discussed in Section 5.4.

Algorithm 3 Adaptive Random Interface Selection

```
Input: Available client interface list $I$
Output: Interface $i \in I$ is selected
1:  for interface $i$ in list $I$ do
2:      /* Count of congestion event for interface $i = c_i$ */
3:      /* For each congestion event, increase $c_i$ by 1 */
4:      /* For each predefined period $T$, decrease $c_i$ by 1 */
5:      /* Weight of interface $i = w_i$ */
6:      /* Default weight of interface $i = w_{i_{def}}$ */
7:      if $c_i \geq 10$ then
8:          $w_i = 0$
9:      else
10:         /* Weight calculated as a moving average that is decreased by 10% */
11:         $w_i = w_{i_{def}} \times (1 - c_i \times 0.1)$
12:     end if
13: end for
14: Calculate $W_{total} = \sum w_i$
15: Select interface $i$ with probability $p_i = \frac{w_i}{W_{total}}$
```

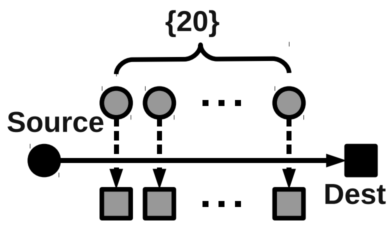


Figure 5.5: Additional traffic to saturate the WiFi channel.

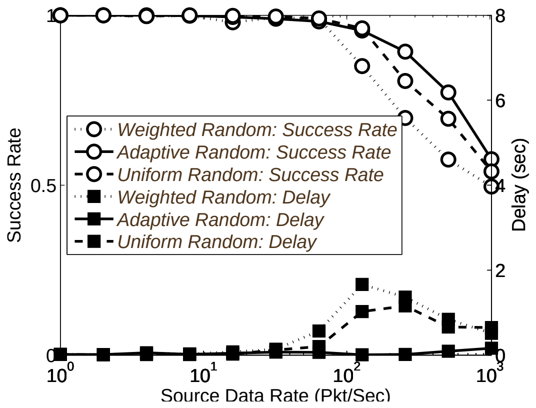


Figure 5.6: Success rate and delay using the adaptive random, weighted random and uniform random client selection methods.

In order to simulate runtime changes in channel conditions, the original source and destination nodes in the previous experiments are surrounded by an additional 20 pairs of source/destination nodes, as shown in Fig. 5.5. These 20 source nodes generate Poisson traffic to saturate the WiFi channel. The weighted random interface selection method is expected to perform poorly because it unconditionally sends packets to the congested WiFi interface.

Fig. 5.6 shows the simulation results, where the resulting success rate for the weighted method is worse than that of uniform random interface selection method. This is because the weighted random selection does not consider the dynamic channel information and results in high number of packets sent to congested WiFi interface. The optimal weights for success rate in this experiment are (508, 0, 256), i.e., avoid packets sent through WiFi, but this will result in lower packet success rate in different traffic patterns. The adaptive random interface selection method successfully avoids the WiFi interface and achieves higher packet success rate for all source packet loads.

The spike and decrease of the resulting delay for the weighted random interface selection is due to the packet drops in the WiFi and CSMA channel. Since dropped packets from WiFi are not included in the delay calculation, the overall average delay for all three interfaces is reduced when a higher proportion of packets are sent through LTE.

5.6 Interface Selection while Using TCP

In order to determine the effects of TCP's congestion control algorithm on the selection of the physical interface, we ran experiments using all of the previous interface selection algorithms as well as the individual physical interfaces for a single source-destination pair utilizing TCP with Poisson traffic distribution. Fig. 5.7 shows the success rate for these simulations. The delay results are shown in Fig. 5.8. No background traffic is considered in these simulations.

The results of these simulations show that LTE has the lowest success rate. As discussed in Section 5.3, the LTE implementation in ns-3 drops some packets when sending burst traffic, and this results in long delays due to TCP re-transmissions. The CSMA success rate drops sharply from the input rate 512 pkt/sec. This is because high

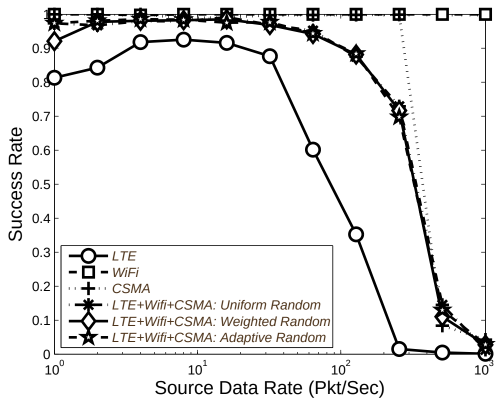


Figure 5.7: Success rate of the different interface selection algorithms using TCP.

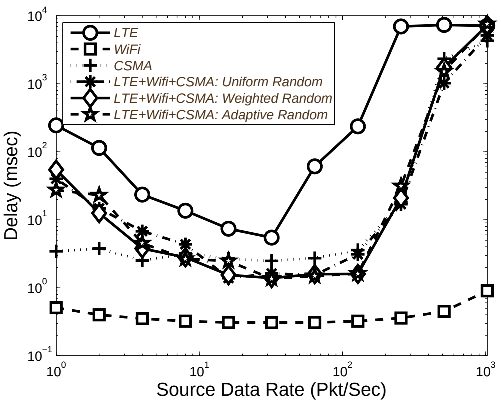


Figure 5.8: Delay of the different interface selection algorithms using TCP.

source data rate overflows the CSMA internal packet queue, and the TCP protocol tries to retransmit the dropped packets. However, retransmission results in more traffic to the interface, which leads to even worse queue overflow. The queue length is 100 packets in this experiment.

From Fig. 5.7, we can see that WiFi and CSMA have the highest success rates, and all the random selection methods have sub-optimal and similar results. This means that for TCP traffic, the optimal per-packet selection rule is to use only one interface (e.g., WiFi or CSMA), instead of multiple interfaces. Although the random interface selection methods provide smoothed results to avoid the worst case of concentrated LTE interface usage, they do not achieve the optimal packet success rate.

The reason for the reduction in success rate is due to out-of-order packet delivery in the multiple interface scenarios. TCP performs flow control and packet retransmission with the use of “Acknowledgement Number” for tracking the order of packets [88]. If the number of the out-of-order packets is over a threshold, TCP will assume that a segment has been lost, and TCP then performs a retransmission without waiting for a retransmission timer to expire. In the case of multiple interfaces, if a sequence of packets is randomly distributed to the WiFi and CSMA interfaces, although the WiFi interface might contain packets later in the order, since WiFi has higher bandwidth and shorter delay, the packets will reach the destination earlier than the CSMA packets, and hence this results in receiving packets out-of-order.

The delay results of the TCP experiments also confirm the same observation. Even in the low source data rate and 100% success rate regions, all the multiple interfaces selection methods have higher delays due to TCP retransmission. This indicates that the performance of interface selection methods changes with the traffic pattern, and the optimal interface selection methods for UDP cannot be directly applied to TCP.

To enhance the success rate, per-stream interface selection methods should be used. To retain the packet order, all traffic of a TCP stream should be forwarded to one designated interface, and different TCP streams should use different interfaces to increase the throughput. The design of a per-stream interface selection method is left as future work.

5.7 Limitations

The virtual interface has some limitations. First, in the current implementation, all the nodes need to have the same set of physical interfaces. Otherwise, a packet may be initially routed to a relay node in the IP routing table, but at the virtual interface it is forwarded to a client interface that is not connected to that relay node. That means that the virtual interface can only select the interfaces that are connected to the destination node. This limitation can be solved by adding a nodes-interfaces table to each virtual interface to record the interfaces of neighbor nodes.

The second limitation is the client interfaces’ MAC addresses. All the client interfaces of a node need to have the same MAC address. The problem is explained in Fig. 5.9. Because the virtual interface can select an interface with the wrong destination

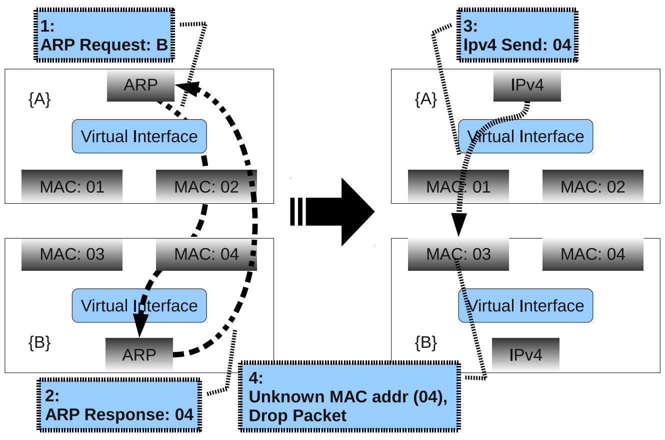


Figure 5.9: Situation showing the need for all physical interfaces to have the same MAC address.

MAC address, the destination node will assume this is an invalid packet and drop it. By assigning all client interfaces of a node with the same MAC address, all the destination interfaces will have the correct addresses and the packet can be processed.

There may be additional application specific limitations. For example, if the WiFi hotspot needs login information input from the user, the virtual interface may fail to complete the authentication process and can impact the resulting performance. In general, any dependencies between the network interfaces and the rest of the protocol layers (i.e., cross-layer interactions) have to be carefully examined when deploying virtual interfaces in the target system.

5.8 Machine Learning Interface Selection

We have done some preliminary MATLAB experiments with machine learning techniques for the interface selection, including Saturating Counter [89], Learning automata [90], Optimal stopping [91], Bayesian Learning [92], Multiarmed Bandit [93], Kalman Filter [94], and Particle Filter [95]. The results show only marginal differences in the performance.

All the methods are similar in their ability to track the network status, but the simulation results are dominated by the traffic patterns and algorithm parameters. Algorithm parameters optimized for one traffic pattern have failed on other traffic patterns, and no single method can out-perform the others in all conditions. The results show that unless the learning method is custom designed to a specific device interface setup and for an application traffic pattern, simple random selection provides adequate performance for general use.

5.9 Conclusions

In this chapter, we proposed an approach that abstracts all the available interfaces using a single virtual interface. By using the proposed virtual interface, packets can be transmitted from any of the available interfaces, without modifying the Layer 3 routing protocols and the Layer 2 MAC protocols. Because this additional virtual layer hides the physical interfaces from the IP layer, novel physical interface selection algorithms can be applied at the virtual layer to improve the network performance. Three interface selection algorithms were proposed in this chapter: uniform random interface selection method, weighted random interface selection method, and adaptive random interface selection method. We evaluated UDP and TCP traffic scenarios with multi-radio devices that support LTE, WiFi, and a CSMA radio interface through simulations in ns-3. The results showed that supporting multi-radio devices with a virtual interface improves the network performance, but the performance of the interface selection algorithms is influenced by the traffic pattern and algorithm parameters.

5.10 Future Work

Our future work will focus on the limitations of the virtual interface implementation. We also plan to add more interface selection algorithms to the virtual layer as libraries, and to implement the virtual interface on physical mobile devices to make it a viable option for future multi-radio device deployments.

Chapter 6

Stateless Multicast Protocol for Dynamic Networks

The previous chapters have explored novel protocol stack architectures to provide support for cross-layer information sharing, as well as the execution of multiple protocols in the same stack layer. Such architectures are important to support the goals of emerging wireless networks. Additionally, emerging wireless networks have several features that require a re-thinking of protocols. Specifically, these networks are often dynamic, due to node mobility, limited node energy requiring duty-cycling of the nodes, and channel conditions. Thus, it is important to consider how best to support the protocol goals in these dynamic conditions. Specifically, in this chapter, we take a fresh look at multicasting in dynamic networks, while in the following chapter we explore the appropriate setting of node duty-cycles in convergecast transmission scenarios.

Multicast routing protocols typically rely on the a-priori creation of a multicast tree (or mesh), which requires the individual nodes to maintain state information. In dynamic networks with bursty traffic, where long periods of silence are expected between the bursts of data, this multicast state maintenance adds a large amount of communication, processing and memory overhead for no benefit to the application. Thus, we have developed a stateless receiver-based multicast protocol that simply uses a list of the multicast members' (e.g., sinks') addresses, embedded in packet headers, to enable receivers to decide the best way to forward the multicast traffic. This protocol, called RBMulticast (Receiver-Based Multicast), exploits the knowledge of the geographic locations of the nodes to remove the need for costly state maintenance (e.g., tree/mesh/neighbor table maintenance), making it ideally suited for multicasting in dynamic networks. RBMulticast was implemented in the OPNET simulator and tested using a sensor network implementation. Both simulation and experimental results confirm that RBMulticast provides high success rates and low delay without the burden of state maintenance.

6.1 Introduction

In our daily life, several applications require data delivery to multiple destination nodes, where the use of multicast routing is an ideal approach to manage and reduce network traffic. These applications range from member-based TV/Video broadcasting to push media such as headlines, weather and sports, from file distribution and caching to monitoring of information such as stock prices, sensors and security. Oftentimes these services are required over highly dynamic networks, such as mobile ad hoc, vehicular, or wireless sensor networks. These networks are dynamic due to the mobility of the nodes in the network and/or the random sleep/awake cycles that are often utilized to minimize energy dissipation of the devices. Providing robust multicast routing in such dynamic network environments is an important design challenge for supporting these applications.

In some wireless multicast applications, the source and intermediate nodes are mobile, but the multicast recipients' locations are fixed and known. For example, fixed, road-side stations may require traffic updates from cars in a vehicular ad hoc network. Similarly, applications including habitat monitoring, wildfire detection, and pollution monitoring utilize data from mobile sensors that must be sent to stationary sinks in the region. In all of these applications, the locations of the particular set of destinations for some data are fixed and known a-priori by the nodes in the network. In other wireless multicast applications, all nodes, including the multicast destinations, are mobile. In this case, in order to support any type of multicast service to particular devices, the source nodes must know the locations of the multicast destination nodes. This can be provided by a service discovery protocol that sits outside the routing protocol, updating the source(s) with the current location of the sink nodes. In either case (fixed sink nodes or mobile sink nodes with a service discovery protocol providing updates on the sinks' locations), the routing protocol can assume knowledge of the sinks' locations. We can exploit this knowledge to design a stateless multicast routing protocol.

In this thesis, we propose a Receiver-Based Multicast protocol, RBMulticast, which is a stateless cross-layer multicast protocol where packet routing, splitting packets into multiple routes and the medium access of individual nodes rely solely on the location information of multicast destination nodes [17]. RBMulticast includes a list of the multicast members' locations in the packet header, which prevents the overhead of building and maintaining a multicast tree at intermediate sensor nodes, because all the necessary information for routing the packet is included within the packet header. Additionally, the medium access method employed does not require any state information such as neighbor wake-up time or any a-priori operations such as time synchronization. No tree creation or maintenance or neighbor table maintenance is required, making RB-Multicast require the least state of any multicast routing protocol, and it is thus ideally suited for dynamic networks.

RBMulticast is a *receiver-based* protocol, which means that the relay node of a packet transmission is decided by the potential receivers of the packet in a distributed manner. This routing approach does not require routing tables and enables the use of the current spatio-temporal neighborhood; this can be compared to proactive and reactive routing protocols where the route is decided using the latest available information, which can be stale. This is a crucial property, especially for dynamic networks. In RBMulticast, receivers contend for the channel based on their potential contribution towards forwarding the packet, which is inspired by the cross-layer protocol XLM [15], a receiver-based unicast protocol designed for wireless sensor networks (WSNs). Nodes that make the most forward progress to the destination will contend earlier and hence have a higher chance to become the next-hop node. In RBMulticast, the multicast routing uses the concepts of "*virtual node*" and "*multicast region*" for forwarding packets closer to the destination multicast members and determining when packets should be split into separate routes to finally reach the multicast members.

The total number of hops that packets travel to reach their destination is an important performance metric for routing protocols, as it provides an indication of bandwidth usage and of the energy efficiency of the protocol. In this chapter, we derive a mathematical model for the lower and upper bounds on average hop count realized by RBMulticast given the network parameters: target area, node density, duty cycle of the nodes, number of multicast members and the communication range. These analytical bounds are validated by simulation runs of RBMulticast performed in Matlab.

To detail the network performance of the RBMulticast, packet level simulations are performed using the OPNET simulator [74]. The performance of RBMulticast is compared to that of the XLM unicast protocol to show the performance gain achieved by the proposed multicast routing protocol. The results show that RBMulticast achieves much better performance in terms of latency and network traffic. Physical experiments of RBMulticast are also conducted using Tmote Sky motes to validate the simulations. Results of these experiments show that RBMulticast achieves high packet delivery success rate even in highly dynamic networks, e.g., over 90% where relay nodes move at speeds up to 30 m/s. Such high performance is not realizable for highly dynamic networks using other multicast approaches, since nodes must keep updated information about the network. RBMulticast is lightweight and robust, making it ideally suited for multicast applications in ad-hoc networks such as WSNs and MANETs.

6.2 RBMulticast Protocol Description

RBMulticast is a receiver-based cross-layer protocol that performs multicast routing based on receiver-based geographic unicast protocols such as XLM [15]. The receiver-based unicast only needs the sender node's location and the final destination node's location, which are provided in the MAC packet, to decide the next hop along the route. We assume that the "void" (hole) problem in geographic routing is solved implicitly, for example, using the right-handed rule as in GPSR [96].

Throughout this chapter, we will assume that the multicast members are stationary, such as multiple stationary sinks in WSNs or stationary road side access points in vehicular ad-hoc networks. The intermediate nodes can be either static or mobile. Although mobile intermediate nodes result in route breaks in conventional multicast protocols, since no multicast tree or mesh is used in RBMulticast, mobile intermediate nodes are supported at no additional cost in RBMulticast. Mobile destinations (multicast members) create a challenging problem for multicast protocols, and its solution is out of the scope of this work.

6.2.1 RBMulticast Overview

Nodes in RBMulticast create what we call “multicast regions” centered around themselves. There are several ways to create these regions (see Section 6.2.2). However, we use a quadrants approach due to its simplicity and good performance, where each multicast region corresponds to one quadrant of the network, for a grid centered at the node.

When a user initiates a request to send a packet to a multicast group, data is passed down to the RBMulticast module in the protocol stack. Once the RBMulticast module gets this packet, it retrieves the group list from its group table, assigns the group nodes to the multicast regions based on their locations, and using these locations, calculates a “virtual node” location for each multicast region. RBMulticast replicates the packet for each multicast region that contains one or more multicast members and appends a header consisting of a list of destination nodes (multicast members) in that region, TTL (Time to Live) value, and a checksum value. The destination of a replicated packet is the “virtual node” of the corresponding multicast region, which can be determined in several ways (see Section 6.2.4), e.g., as the geometric mean of the locations of all the multicast members in that multicast region. In the end, all packets for all multicast regions are inserted in the MAC queue, and are then broadcasted to the neighborhood. The node closest to the virtual node (within the available relay nodes as determined by receiver-based contention at the MAC layer) will take responsibility for forwarding the packet. The procedure for transmitting packets is summarized in pseudo code in Algorithm 4.

When a node receives a multicast packet, RBMulticast first examines the checksum in the packet header, and drops the packet if any corruption exists in the packet. It also drops the packet if it is not in the forwarding zone. The forwarding zone is the area within the radio range of the sender that has a smaller distance to the destination than the sender-destination distance.

After a node receives a multicast packet, it then retrieves the destination node list from the RBMulticast packet header. If this node is inside the destination list, it removes itself from the list and passes a copy of the packet to the upper layers in the protocol

Algorithm 4 RBMulticast Send

```
Input: Packet output from upper layer
Output: Packets inserted to MAC queue
 1: Get group list $N$ from group table
 2: for node $n$ in group list $N$ do
 3:     for multicast region $r$ in 4 quadrants regions $R$ do
 4:         if $n \in r$ then
 5:             Add $n$ into $r.list$
 6:         end if
 7:     end for
 8: end for
 9: for $r \in R$ do
10:     if $r.list$ is non-empty then
11:         Duplicate a new packet $p$
12:         Add RBMulticast header ($TTL$, $checksum$, $r.list$) to $p$
13:         Insert $p$ to MAC queue
14:     end if
15: end for
```

stack. RBMulticast then checks the TTL value and drops the packet if the TTL is lower than 0. Finally, if there still remain nodes in the destination list, multicast regions and virtual nodes are recalculated, and new packets are generated if required. The packets (one per multicast region that contains multicast members) are then inserted in the MAC queue for transmission. The procedure executed after receiving packets is summarized in pseudo code in Algorithm 5.

Fig. 6.1 gives an example of how RBMulticast is employed. The two multicast regions, the south-west and north-west quadrants, contain only one multicast member each, and thus a packet is sent directly to these multicast destinations. The north-east multicast region has three multicast members, and thus a single packet is sent to the virtual node located at the geometric mean of the locations of the multicast members (dotted circle with label 3 in the figure). The south-east multicast region has no multicast members, and hence no packet is transmitted into this region. Once a packet sent towards a virtual node reaches an intermediate node for which the multicast members are no longer in the same multicast region, the node will split off packets to each of the multicast regions accordingly.

6.2.2 Multicast Regions

Once a node receives a multicast packet (from the application layer or from a previous hop node), it divides the network into multicast regions, and it will split off a copy of the packet to each region that contains one or more multicast members. We show two possible divisions of the network into multicast regions in Fig. 6.2(a) and 6.2(b).

There is no method that is clearly best. Influencing factors include the sink node locations and how the relay nodes are distributed. For the quadrants approach, the multicast region decision only needs two comparisons (X and Y axes) for each multicast member and is extremely fast. We believe that it is preferable for systems with low computational capacity such as wireless sensor nodes.

6.2.3 Packet Splitting

In Algorithms 4 and 5, we describe the RBMulticast method that splits packets at relay nodes for which the multicast destinations reside in different regions. This method is

Algorithm 5 RBMulticast Receive

```
Input: Packet input from lower layer
Output: Forwarded packets inserted to MAC queue
 1: Calculate checksum. Drop packet if error detected
 2: Drop packet if not in Forwarding zone
 3: Get destination list $D$ from packet header
 4: for node $d$ in destination list $D$ do
 5:     if I am $d$ then
 6:         Duplicate the packet and input to upper layer
 7:         Remove $d$ from list $D$
 8:     end if
 9: end for
10: if $TTL$ in header = $0$ then
11:     Drop the packet
12:     return
13: end if
14: for $d \in D$ do
15:     for multicast region $r$ in 4 quadrants regions $R$ do
16:         if $d \in r$ then
17:             Add $d$ into $r.list$
18:         end if
19:     end for
20: end for
21: for $r \in R$ do
22:     if $r.list$ is non-empty then
23:         Duplicate a new packet $p$
24:         Add RBMulticast header ($TTL - 1$, $checksum$, $r.list$) to $p$
25:         Insert $p$ to MAC queue
26:     end if
27: end for
```

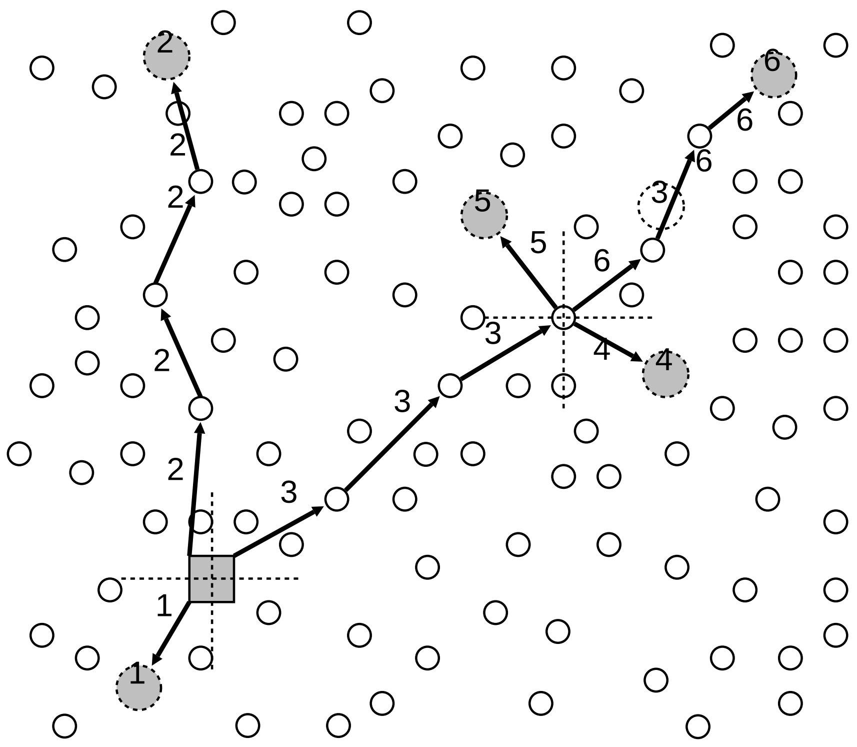


Figure 6.1: Example showing how RBMulticast delivers multicast packets. The source node is the square node. Multicast members are shaded circles, and virtual nodes are dotted circles. Because every destination node will become a virtual node at the end, they are all shown with dotted circles. The number on the side of the lines indicate the destination of that packet.

used in the protocol description due to its simplicity.

In a variation of this method, namely RBM-V, the packets are instead split off at the neighbor nodes of the virtual node, which delays splitting the packets compared to the former method. Hence, in RBM-V, certain packets need to travel backwards after splitting, which may increase the total hop count. However, as will be shown in Section 6.3.3, this variation of RBMulticast requires similar or lower average number of hops to reach all members.

6.2.4 Virtual Node

Network layer multicast protocols, which require multiple destinations, are built on top of Link layer protocols that typically allow only a single (unicast) or all (broadcast)

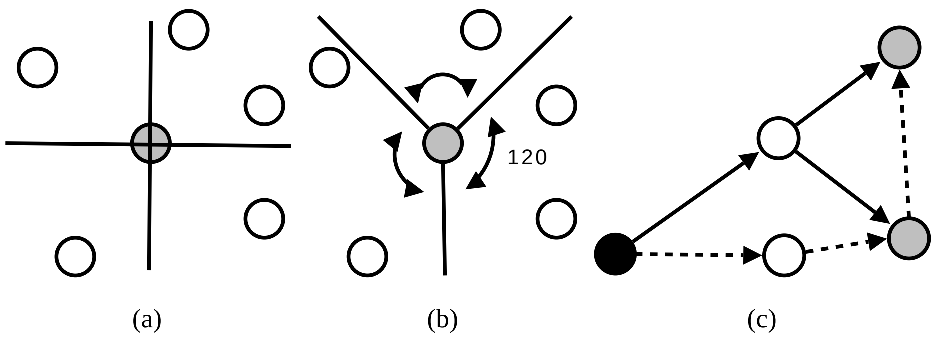


Figure 6.2: Two possible ways to divide a space into multicast regions: a) dividing the space into four quadrants, and b) dividing the space into three 120° regions. c) demonstrates how to choose a next hop node. The solid node is the source node, and the gray nodes are the multicast members. The solid line is the route when choosing a target node near the geographic mean of the multicast members, and the dotted line is the route when choosing a target node close to the nearest multicast member. We can see that the longest distance is two hops distance in the first case, and it is three hops distance in the second case.

destinations. Possible ways to adapt the need for multiple multicast destinations to a MAC layer that can only handle a single destination are choosing a node that is close to the geographic mean of the multicast members, or close to the nearest multicast node, as shown in Fig. 6.2(c).

In RBMulticast, because we assume no knowledge of neighbor nodes and no routing tables, we assign a "virtual node" located at the geographic mean of the multicast members for each multicast region. This virtual node is used as an imaginary destination for the multicast packet in that region. The virtual nodes are not necessarily reachable or even physically exist as illustrated in Fig. 6.1. The idea behind this is that even if a virtual node does not exist, we can still find a route using the assumed receiver-based MAC protocol to get the packet closer to the location of the virtual node. On the other hand, when using the nearest multicast node as the destination, all node addresses physically exist and virtual nodes are not necessary.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Protocol ID								TTL (Time To Live)							
TOS (Type Of Service)								DLL (Destination List Length)							
Checksum															
Source Address															
Destination List Address 1															
⋮															

Figure 6.3: Packet header of the RBMulticast protocol.

6.2.5 RBMulticast Header

The goal of a stateless approach is to keep intermediate nodes from having to store any data for routing and medium access. This is possible only if all information required to multicast a packet is carried along with the packet. The question is how much information the multicast packet needs to carry for successful delivery to all multicast members. Fig. 6.3 shows the structure of an RBMulticast header.

The first byte *Protocol ID* is for protocol identity in the protocol stack [86]. *TTL* (Time To Live) provides a maximum time, in hop number, that a packet should last in the network. *TOS* (Type Of Service) indicates four kinds of packets in RBMulticast, which are “data”, “join”, “leave”, and “update” packets. The update packets are used in group management and periodic group list updates. *DLL* (Destination List Length) indicates how many nodes are in the node list, and thus will determine the length of the header. The RBMulticast header size is not fixed since the destination list length is variable. *Source Address* is the address of the source node, which equals the RB-Multicast group ID of this packet, and *Destination List Address* stores the locations of the *DLL* destination nodes. The RBMulticast group ID is not actually needed in this protocol since all the multicast members are included in the packet header.

Because we assume a receiver-based MAC layer, the next hop is determined by a joint decision among potential receivers. Hence, the RBMulticast header does not need to carry any state for routing the packet. However, we still need to decide when the packet must be split off to different destinations. This is usually implied by tree branches in tree-based multicast approaches. Because of the location information assumption, we can use multicast regions to decide when packets must be split off without any tree structure. A packet will be split off to each multicast region if multicast members exist in these regions. Therefore, a *destination list* is the only requirement for multicast packet delivery: this destination list must be carried inside the packet header.

As with any multicast protocol that uses a destination list, the packet header length will increase linearly with the number of destination nodes. The maximum number of multicast members allowed in a group is restricted by the packet size. For packets in the IEEE 802.15.4 standard of Wireless Sensor Networks, the maximum packet size is 128 bytes, and hence the maximum number of nodes in the destination list is around 50, which is sufficient for practical purposes. The impact of packet length on energy consumption can be reduced by adjusting the power control of the MAC protocol, as shown in [97]. The idea is to compensate for the increased packet collision rate due to long packet lengths by increasing the transmit power.

6.2.6 Group Management

RBMulticast supports multicast group management where nodes can join or leave any multicast group. Some nodes manage the multicast groups and act as the group heads. Nodes join and leave a group by sending “join” and “leave” packets to the group head. Join and leave packets are multicast packets with destination lists that contain only the group head address.

RBMulticast supports Many-to-Many multicast mode, and thus every node in a multicast group can multicast packets to all other nodes in the same group. The extra burden is that the node must maintain group node lists for groups it has joined. In the case of nodes joining or leaving, the group head must send “update” packets including a list of its updated multicast group members to all group nodes. Nodes send “join” packets periodically to the group head, and nodes that die without sending ”leave” packets are removed from the list after a time-out period.

6.2.7 Summary

In summary, RBMulticast uses virtual nodes to embed multicast packets into a receiver-based unicast protocol, and it uses multicast regions to split off packets to different multicast members. All the required information is carried by the packet headers, and hence no state is stored by the intermediate relay nodes. The only required information for packet delivery is the location of the relay source node and the multicast members, thus the size of the packet header grows with the number of multicast members. For a multicast group, a group head maintains the group list, which is the only state information of our RBMulticast protocol.

Compared to non-stateless multicast protocols that need to update the routing table at intermediate nodes, RBMulticast stores no state at intermediate nodes, making it ideally suited for multicasting in dynamic networks.

6.3 Analytical Bounds on Average Hop Count Performances

Average hop count performance of a protocol is an important performance metric, since it provides information about the network traffic generated by the protocol and the total energy consumed for packet delivery.

6.3.1 RBMulticast Performance

For the sake of brevity in the analysis, we consider the variation of RBMulticast where the packet splits are not done until the packets reach a relay node within one hop distance of the virtual node, which is named RBM-V. As will be shown by simulations in Section 6.3.3, the hop count performance of RBM-V is similar to or better than that of RBMulticast.

RBMulticast is investigated under the conditions that nodes are assumed to know their own location information and to use a perfect MAC with no packet collisions. Nodes are uniformly deployed with density ρ in a $2R$ diagonal square area. The source node is located at the center of the area, and M multicast destinations are uniformly distributed in the square area. Quadrant multicast regions are used. Transmission ranges of the nodes are normalized to 1 in the analysis. All nodes except the source and the multicast members employ a fixed duty cycle, d. The average number of available relays in the radio coverage area is defined as $N = d\rho\pi$. Symbols used in the multicast analysis are defined in Table 6.1.

We calculate the expected number of packet splits and the expected packet distance from the relay nodes to the virtual nodes iteratively, where at each iteration packets are split and are sent to the updated virtual nodes. The total expected number of hops $E[m]$ is the iterative summation of the multiplication of the expected number of packet splits with the expected number of hops for each split packet.

Since the multicast members are uniformly distributed among all multicast regions, the virtual node location of a region is the geometric mean of the multicast members' coordinates, i.e., the center of mass of the multicast members.

For the case of quadrant multicast regions, with the radius of the experiment area R, the expected location of the virtual node in the first quadrant is,

$$\vec{r} = \frac{\int_S \rho_S \vec{r} dS}{\int_S \rho_S dS} = \frac{\int_0^{\frac{R}{\sqrt{2}}} \int_0^{\frac{R}{\sqrt{2}}} \rho_S \binom{x}{y} dxdy}{\rho_S (\frac{R}{\sqrt{2}})^2} = \begin{pmatrix} \frac{R}{2\sqrt{2}} \\ \frac{R}{2\sqrt{2}} \end{pmatrix}, \tag{6.1}$$

where ρ_S is the density of sink nodes (multicast members) in the multicast region S. The expected distance from the relay node to the virtual node is hence $|\vec{r}| = \frac{R}{2}$, and the average advancement ratio for the first virtual node is $\alpha = \frac{|\vec{r}|}{R} = \frac{1}{2}$.

At iteration i, N_{Ri} multicast packets are sent to all virtual nodes, each of which is at average distance $D_i = \alpha D_{i-1}$ due to symmetry. The average distance to a virtual node at iteration i is thus

$$D_i = (\alpha)^i R = (\frac{1}{2})^i R. \tag{6.2}$$

The iteration ends if the distance from the relay nodes to the virtual nodes is less than 1, i.e., if $D_i < 1$. Hence, from (6.2),

$$(\alpha)^i R < 1 \Rightarrow i < \left\lceil \frac{log\frac{1}{R}}{log\alpha} \right\rceil. \tag{6.3}$$

We ignore the expected number of hops $E[n_i']$ for $D_i < 1$ for simplicity.

For the case of four $90°$ quadrant multicast regions, given that a total of M sink nodes are uniformly distributed in all four regions, the probability mass function of the number of regions having at least one sink node, N_R, can be written as

$$P_{N_R}(N_R = k|M) = (\frac{1}{4})^M \binom{4}{k} \sum_{i=1}^{k} (-1)^{k-i} \binom{k}{i} i^M. \tag{6.4}$$

Table 6.1: Definition of symbols in the multicast analysis.

Symbol	Definition
ρ	Density of relay nodes.
d	The probability of a node to have its radio turned on at any given time, corresponding to the duty cycle.
N	$N = d\rho\pi$ is the average number of available relays (neighbors) in the radio coverage area.
R	The radius of the experiment area.
M	The number of multicast members.
M_d	The number of target destination nodes for a packet.
α	The average advancement ratio: the ratio of the distance to the virtual node to the radius of the multicast region at each iteration.
N_{Ri}	Number of multicast regions that have at least one sink node at the ith iteration.
$A(r, D)$	The forwarding zone area at distance $D \geq 1$. $A(r, D) = 2\int_{D-1}^{r} a \arccos(\frac{a^2+D^2-1}{2aD})da.$
ζ	The advancement towards the destination.
n'	Number of hops until one-hop distance to the sink.
D	The distance from the source to the sink node.
D_i	Avg. distance between relay and virtual nodes at iteration i.
n'_i	The number of hops to reach the virtual node at iteration i.
i_{max}	Total number of iterations.
m'	Total number of hops for all packets to be one hop distance to sinks.
m	Total no. of hops for all packets to reach the sinks. $m = m' + M$.

It can be verified that $\sum P_{N_R}(\cdot|M)$ is equal to 1. For a packet with M_d destination nodes, the average number of target regions $E[N_R]$ is

$$E[N_R|M_d] = \sum_{k=1}^{4} k * P_{N_R}(k|M_d). \tag{6.5}$$

Note that M_d might not be an integer when we consider the average number of destination nodes at a given iteration. We use (6.4) as an approximation in this case. It can be verified that this approximation holds in the boundary conditions

$$\begin{aligned} \lim_{M_d \to 0} E[N_R|M_d] &= 0 \quad (6.6)\\ \lim_{M_d \to \infty} E[N_R|M_d] &= 4. \quad (6.7) \end{aligned}$$

The average number of target multicast regions at iteration i is an iterative function given by

$$E[N_{R_i}] = E[N_R|M_{d_{i-1}}] \times E[N_{R_{i-1}}], \tag{6.8}$$

where M_{d_i} is the number of destinations of a packet at iteration i, which is found by

$$M_{d_i} = \frac{M}{E[N_{R_i}]}. \tag{6.9}$$

The initial conditions of the iterative function are $E[N_{R_0}] = 1$ and $M_{d_0} = M$. The iterations end when the average number of destination nodes at iteration i is less than 2, that is, $M_{d_i} < 2$. In this case, 1 out of M_{d_i} packets is directly sent to the destination node and the remaining packets $M_{d_i} - 1$ require one further iteration of packet splitting. We do not count hops $E[n_i']$ for iterations where $M_{d_i} < 2$ to simplify the equations and assume each of the destinations that are left can be reached in one additional packet transmission.

Based on the limits of $D_i \leq 1$ and $M_{D_i} \geq 2$, the total number of iterations i_{max} is the solution of the maximization problem

$$\begin{aligned} &\text{maximize} \quad i \\ &\text{subject to} \quad \lceil \tfrac{log\frac{1}{R}}{log\alpha} \rceil > i \\ &\qquad\qquad\quad \tfrac{M}{E[N_{R_i}]} \geq 2, \end{aligned} \tag{6.10}$$

where $i \in \mathbb{N}$ and $E[N_{R_i}]$ is the iterative function given in (6.8).

The bounds of the expected number of hops $E[n']$ that first leads to a point within one-hop distance to the destination node is given in [34], which is

$$\frac{D-1}{E[\zeta]} \leq E[n'] < \frac{D}{E[\zeta]}, \tag{6.11}$$

where $E[\zeta]$ is the average one-hop advancement in distance toward the sink and D the distance from the source to the destination node. Thus,

$$E[\zeta] \quad = \quad \begin{cases} 1 - \int_0^1 e^{-NA(D-a,D)/\pi} da, & D \geq 1. \\ \text{undefined}, & \text{otherwise.} \end{cases} \tag{6.12}$$

Here, $A(r, D)$ is the forwarding zone area comprised of the intersection of two circles with radii 1 and r with centers at distance $D \geq 1$, and N is the average number of available relays.

We modify the upper bound of (6.11) for the case $0 \leq D < 1$ and the lower bound of (6.11) for the case $1 \leq D < 2$ because hop count should always be larger than 1, which means at least 1 additional hop is needed before reaching the multicast members. The distance D is also calculated from (6.2) for each iteration. Equation (6.11) is thus expanded to include these boundary conditions as

$$\max\Big\{\frac{\alpha^i R - 1}{E[\zeta]}, 1\Big\} \leq E[n_i'] < \max\Big\{\frac{\alpha^i R}{E[\zeta]}, 1\Big\} \tag{6.13}$$

The average number of hops is aggregated over the iterations of packet splitting. The average hops in each iteration is the average number of required multicast regions $E[N_{R_i}]$ times the average number of hop advancements $E[n_i']$ for each split packet. The expected total number of hops such that all packets lead to points within one hop distance of sink nodes is then

$$E[m'] = \sum_{i=1}^{\infty} E[N_{Ri}] E[n_i']. \tag{6.14}$$

Hence, the expected total number of hops is

$$E[m] = E[m'] + M. \tag{6.15}$$

Finally, incorporating (6.12) and (6.13) into (6.14), the upper and lower bounds for the expected total number of hops $E[m]$ can be calculated.

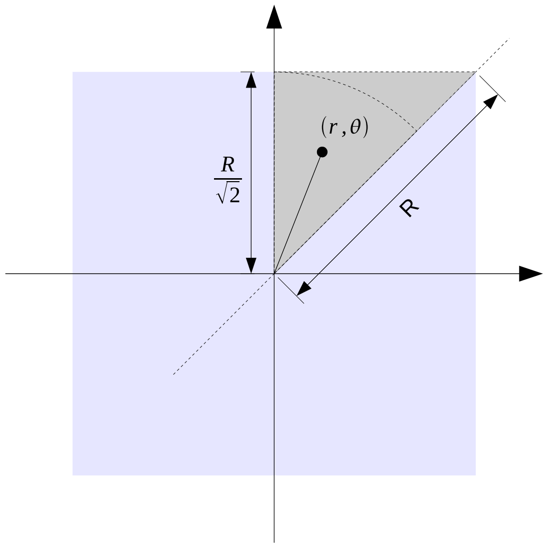


Figure 6.4: The region considered in the multiple unicast analysis.

6.3.2 Analysis for Using Multiple Unicast as Multicast

For comparing the performance gain of RBMulticast, this section investigates the expected total number of hops for the packet transmissions when using separate unicast delivery to each multicast member instead of using RBMulticast. The idea behind this analysis is that unicast can be regarded as a special case of RBMulticast, where each destination node is in a different multicast region in the first iteration. That is, $E[N_{R_1}] = N$ and $E[N_{R_i}] = 0$ for $i \geq 2$.

To calculate the expected distance from the source to the uniformly distributed destination nodes in the region, we need to calculate the probability density function f_D for the distance from the source to the destination nodes. Because of the symmetry of the regions, we only need to consider the shaded triangle region shown in Fig. 6.4. The function f_D is the same for the remaining areas.

Let the two uniformly distributed random variables $\hat{X}$ and $\hat{Y}$ represent the coordinates of destination nodes in a region. The probability density function $f_{\hat{X},\hat{Y}}$ of a destination node to be located within the triangle shown in Fig. 6.4 is $\frac{4}{R^2}$. In order to calculate the expected location of the destination nodes, we use the Jacobian determinant J to transform $\hat{X}$ and $\hat{Y}$ into random variables $\hat{R}$ and $\hat{\Theta}$ in the polar coordinate system as

$$\hat{R} = \sqrt{\hat{X}^2 + \hat{Y}^2}, \;\; \hat{\Theta} = \tan^{-1}\frac{\hat{Y}}{\hat{X}}. \tag{6.16}$$

The joint probability distribution function is then

$$f_{\hat{R},\hat{\Theta}}(r,\theta) = |J| \cdot f_{\hat{X},\hat{Y}} = r\frac{4}{R^2}. \tag{6.17}$$

The marginal probability density function $f_{\hat{R}}(r)$ should be calculated separately for the two intervals $0 \le r < \frac{R}{\sqrt{2}}$ and $\frac{R}{\sqrt{2}} \le r \le R$.

For $0 \le r < \frac{R}{\sqrt{2}}$, r and θ are independent. Hence,

$$f_{\hat{R}}(r) = \int_{\frac{\pi}{4}}^{\frac{\pi}{2}} f_{\hat{R},\hat{\Theta}}(r,\theta)d\theta = \frac{\pi r}{R^2}, \; 0 \le r < \frac{R}{\sqrt{2}}. \tag{6.18}$$

For $\frac{R}{\sqrt{2}} \le r \le R$, r and θ are dependent and $\frac{\pi}{4} \le \theta \le \sin^{-1}\frac{R}{\sqrt{2}r}$. Hence,

$$f_{\hat{R}}(r) = \int_{\frac{\pi}{4}}^{\sin^{-1}\frac{R}{\sqrt{2}r}} f_{\hat{R},\hat{\Theta}}(r,\theta)d\theta \tag{6.19}$$

$$= \frac{4r}{R^2}\left[\sin^{-1}\frac{R}{\sqrt{2}r} - \frac{\pi}{4}\right], \; \frac{R}{\sqrt{2}} \le r \le R. \tag{6.20}$$

The expected number of hops to reach a point within one-hop distance of the sink node is

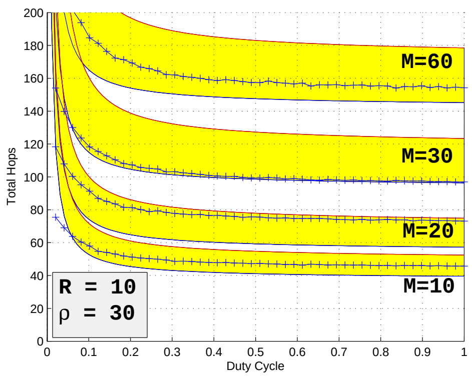
$$E[n'] = \int_0^R f_{\hat{R}}(r) \cdot E[n']\big|_{D=r} dr. \tag{6.21}$$

From the multicast equation (6.14) with splitting the packet to M regions (sinks) in the first iteration and the distance distribution function $f_{\hat{R}}(r)$,

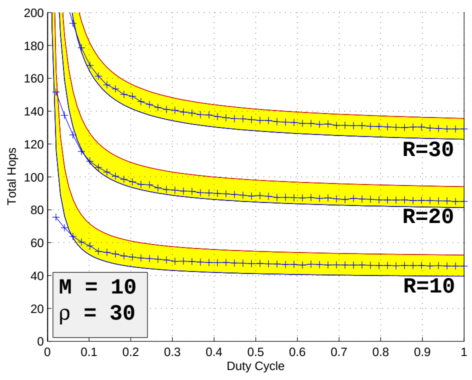
$$\begin{aligned} E[m'] &= \sum_{i=1}^{\infty} E[N_{Ri}]E[n_i'] = E[N_{R_1}]E[n'] \\ &= N \cdot \left(\int_0^R f_{\hat{R}}(r) \cdot E[n']\big|_{D=r} dr\right). \end{aligned} \tag{6.22}$$

Replacing the upper bound $\dfrac{D}{E[\zeta]}$ and lower bound $\dfrac{D-1}{E[\zeta]}$ with $E[n']$, we can find $E[m']$. Finally,

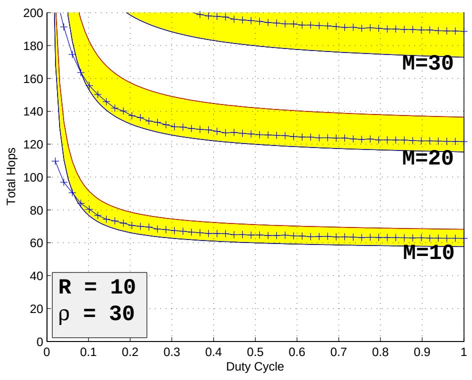
$$E[m] = E[m'] + M. \tag{6.23}$$



(a) Analytical bounds and simulation results for total number of hops with different number of sink nodes M.



(b) Analytical bounds and simulation results for total number of hops with different network area R.



(c) Analytical bounds and simulation results using unicast for multicast with different number of sink nodes M.

Figure 6.5: Performance comparisons for RBMulticast with perfect location information and no packet collisions.

6.3.3 Validation of Hop Distance Analysis and Performance Comparisons

We conducted Matlab simulations of RBM-V to validate the analysis presented above. For all experiments, node density is $\rho = 30$ nodes/m^2, where nodes are uniformly randomly distributed in a square area with diagonal $2R$.

The upper bound and lower bound analytical results and the corresponding Matlab simulations are shown in Fig. 6.5. Results for fixed R and various M values are shown in Fig. 6.5(a), and results for fixed M and various R are shown in Fig. 6.5(b). All the results show that $E[m]$ provides good lower and upper bounds estimates for the average total number of hops required for RBM-V for all M and R value pairs.

The performance using unicast instead of multicast is shown in Fig. 6.5(c). The result for $M = 60$ is not shown in this figure because the total hop count exceeds the boundary of the figure. The results, as seen from the comparison of Fig. 6.5(a) and 6.5(c), show that for the same network range R, RBM-V dramatically decreases the total number of hops compared with using multiple unicast transmissions.

All analytical results overestimate the number of hops for very low duty cycle regions. In the analysis, a packet can traverse an unlimited number of hops to reach the destination, because each individual hop can advance infinitesimal distance according to (6.12). In the simulations, however, node locations are discretely distributed, and total hop count to the destination cannot exceed the number of relay nodes between the source and the destination. Consequently, the bounds are skewed when the duty cycle d (hence the number of available relays $N = d\rho\pi$) is very low, but they hold for most d values.

The simulation results for RBMulticast, where packets are split whenever target destination nodes reside in different multicast regions are given in Fig. 6.6. We show the standard deviation in this case. The results show that the delayed splitting version, RBM-V, uses fewer hops when M is large. This shows that on average RBM-V performance is better in terms of total hops if sink nodes are uniformly distributed. When RBMulticast is deployed in practice, the performance is determined by the actual network topology and the underlying MAC protocol.

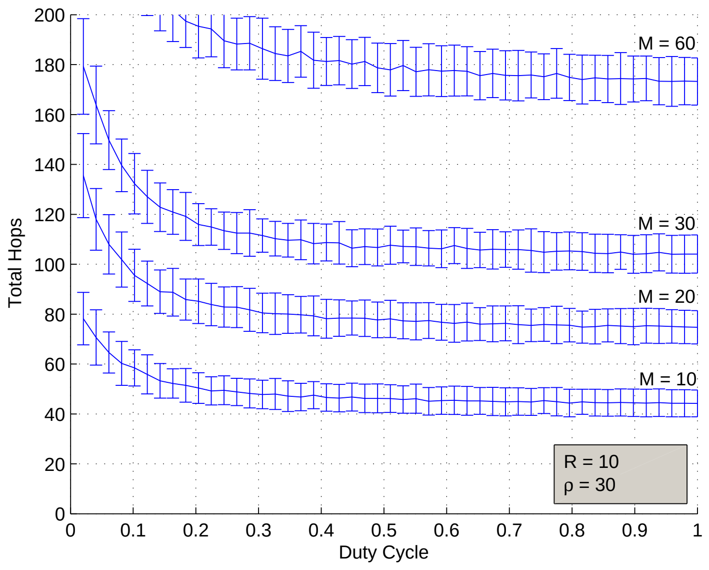


Figure 6.6: Matlab simulation results of total number of hops with different number of sink nodes M for the original version of RBMulticast.

6.4 RBMulticast Performance Evaluation

We implemented RBMulticast in the OPNET Simulator [74] to investigate its network performance with the underlying receiver-based MAC protocol in both static and mobile scenarios. Results of detailed packet-level OPNET simulations with high densities showed that RBMulticast suffers from packet collisions when packets are split. The reason for these collisions and a proposed MAC-level improvement for reducing the effect of collisions is described in this section, along with the simulation results for the improved RBMulticast.

6.4.1 Outline of Implemented MAC Unicast Protocol

In our receiver-based routing, all receivers contend by sending CTS (clear to send) packets to the transmitter to be the next-hop router when they hear an RTS (request to send) from a transmitter. The CTS contention is decided by the distance from the receivers to the sink, that is, receivers send CTS packets back to the transmitter with a backoff time proportional to their distance to the sink. The first node that sends a CTS

Table 6.2: Packet Delivery Ratio in terms of SPTI.

SPTI (sec)	0	0.05	0.08	0.125	0.25
PDR	0.75	0.88	0.94	0.95	0.95

packet is selected as the next hop by the transmitter, and the transmitter forwards the DATA packet to that node. The communication ends with a final ACK packet from the next hop node to the transmitter.

6.4.2 Split Packet Contention Problem in RBMulticast and the Proposed Solution

RBMulticast requires splitting the packet at a node, if the locations of the multicast members, which are listed in the header of the packet, reside in different regions for that node. The packet splitting creates replicas of a packet, updating the destination node locations in each packet accordingly. After replication, all the packets generated are immediately inserted to the node's buffer to be transmitted. However, this creates a burst of packet traffic and congestion within the transmission range of the splitting node, since the relay nodes receiving the packets will contend with the splitting node with the remaining split packets. The problem is more severe for large interference-to-transmission range ratios, because of the higher number of relay nodes contending with the splitting nodes and with each other.

To decrease the contentions and the possibility of collisions after packet replications, we first define a transmission order for the replicated packets based on the region. For example, the packets destined to the northeast region are transmitted first, the ones destined for the northwest region are transmitted second, etc. Then, a certain splitting packet time interval (SPTI) is used between transmissions destined to different regions. One disadvantage of delaying the packet transmissions by the duration SPTI is the increase in the end-to-end delay, i.e., the latency. To determine a reasonable value for SPTI that achieves a high packet delivery ratio with an acceptable latency, we conducted simulations with 5 sinks and 200 nodes with 100% duty cycle. The effect of SPTI is shown in Table 6.2. It is clearly seen that the packet delivery ratio is increased when SPTI is used compared to not using it. An SPTI value of 80 ms can achieve al-

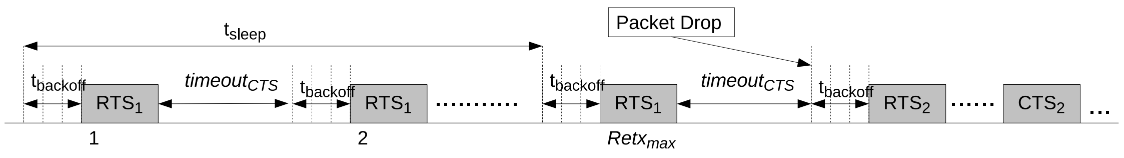


Figure 6.7: Adjustment of $timeout_{CTS}$ for MAC improvement.

most as high packet delivery ratio as that with high SPTI. That means a low latency can still be achieved using SPTI without sacrificing the packet delivery ratio performance.

6.4.3 MAC Level Improvements

To achieve a high packet delivery ratio, at least one relay candidate should be awake and listening to the channel during an RTS packet transmission or any of its retransmissions. Each RTS transmission requires a preceding random backoff and, for the case of no CTS reply, the CTS timeout duration. To guarantee reaching a relay candidate if one exists, all retransmissions should be spread to a duration larger than the maximum sleep duration of the nodes, i.e.,

$$t_{sleep} \leq Retx_{max} \times (t_{backoff} + timeout_{CTS}) \tag{6.24}$$

where t_{sleep} is the sleep duration, $Retx_{max}$ is the maximum number of RTS retransmissions, $t_{backoff}$ is the expected backoff time for contention, and $timeout_{CTS}$ is the timeout duration a node will wait to receive a CTS. $t_{backoff}$ can be calculated as

$$t_{backoff} = E[SlotNumber] \times t_{slot} \tag{6.25}$$

where $SlotNumber$ is the selected backoff slot number, and t_{slot} is the duration of a slot. The idea of spreading RTS retransmissions to a duration larger than t_{sleep} is illustrated in Fig. 6.7.

The parameter $timeout_{CTS}$, which is enlarged by 10 times compared with the default value of the implementation of the original XLM MAC, is used as the duration between the end of an RTS transmission and its retransmission when no CTS reply is received for the previous RTS. Thereby the backoff duration for CTS contention of the relay candidates is enlarged by 10 times correspondingly, which allows relay candidates to have more time listening to the channel and to have their schedule canceled if they hear an ongoing CTS transmission.

6.4.4 Simulation Results

We define multiple scenarios for RBMulticast simulations to evaluate the three performance metrics: packet delivery ratio, latency and the average traffic generated to transmit one data packet to all multicast members. In all scenarios, the area is a $150m \times 150m$ square. The transmission range is $30m$ and the interference range is approximately $80m$. Based on the Tmote Sky sensor node specifications [73] and the observations from experiments, we set the channel data rate to be $220Kbps$, the length of RTS, CTS, and ACK packets to be 78 bits and of raw data packets to be 400 bits. The SPTI is set to $0.1\ s$, and the maximum number of retransmissions is set to 25. The source packet generation rate is 0.2 pkts/sec. Each parameter set is evaluated with 10 simulation runs whose averages are displayed in the figures. The standard deviation of the results are observed to be small, ranging from 1% to 7%.

It is difficult to compare RBMulticast with existing multicast routing protocols, as any protocol that requires state maintenance will not be able to function in the dynamic environments tested here, such as with node duty cycles as low as 20% and with high mobility nodes as high as 30 m/s. Hence, we compared RBMulticast to using stateless unicast protocols to send the packets individually to each multicast member. Two unicast protocols are compared to illustrate the advantage of RBMulticast. The first protocol is Unicast based on the original XLM MAC, which is denoted as UOX, and the second protocol is Unicast based on the improved XLM MAC, denoted by UIX, where the MAC-level improvements proposed in Section 6.4.3 are applied. Both unicast protocols are run with 100% duty cycle and are compared to RBMulticast with 20%, 60% and 100% duty cycle.

6.4.4.1 Static Nodes, Five Sinks

The first set of simulations are investigated to evaluate the performance of RBMulticast using static nodes with the source located at (0, 0) and 5 sinks located at the edge of the target area, as shown by the rectangular nodes in Fig. 6.8. The average packet delivery ratios observed for varying numbers of nodes are shown in Fig. 6.9(a). As seen in

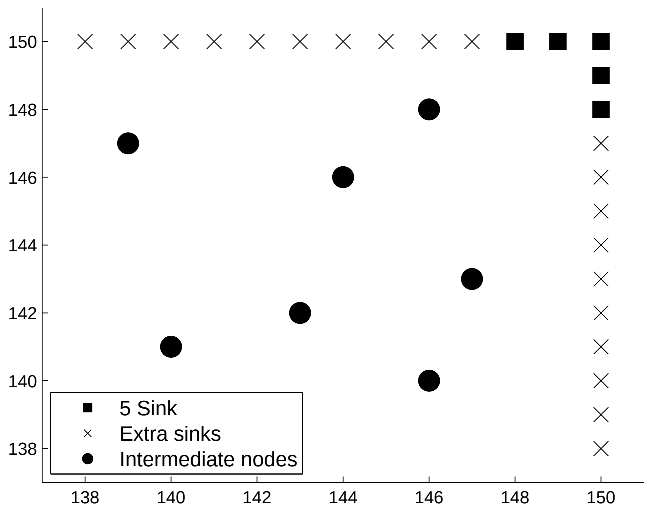


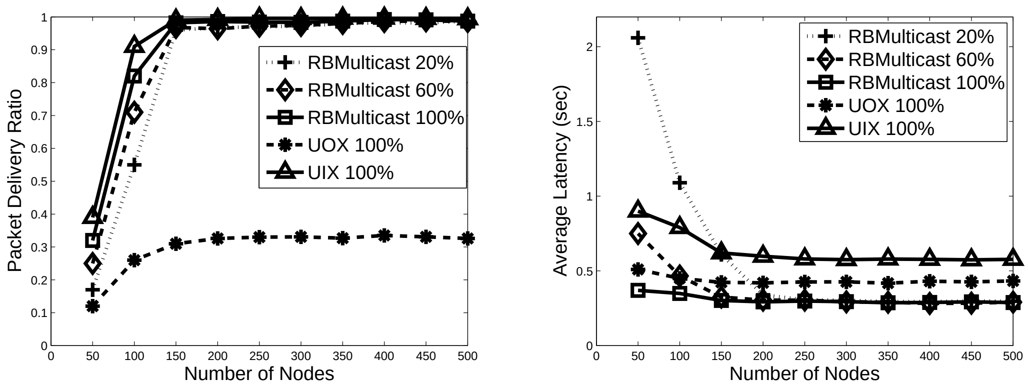
Figure 6.8: Multicast member (sink) locations in the simulation. The additional members align along the boundaries for different simulations. Note that this figure shows a blow-up of the north-east quadrant of the simulation area.

the figure, the packet delivery ratio is very low for a small number of nodes, which is due to the high probability of holes in the network. When there are no holes in the area, which is achieved with high density, the packet delivery ratio is close to 100% for RBMulticast, independent of the duty cycle value. This interesting result is due to the improvements proposed for the MAC: SPTI efficiently reduces the contention of multiple splitting packets, and the extended CTS timeout enables finding a relay node, even when the nodes spend much time sleeping. It is also shown that the packet delivery ratio is not reduced as density increases, which is usually the case due to multiple CTS replies causing congestion in the network.

From Fig. 6.9(a), we can see that UIX performs slightly better than RBMulticast in terms of packet delivery ratios and both perform much better than UOX. For UIX, packets are deferred before their transmission due to the timer on the buffer in XLM. Therefore, before the transmitter node's timer reaches its timeout to send the next packet to a different multicast member, it is quite possible that the previously transmitted packet has reached the destination node and thus will not create interference for the new packet. With a reasonable node density such that no holes exist in the network, the packet delivery ratio is expected to be high as shown in Fig. 6.9(a). Although the same improved MAC guarantees the packet delivery ratio performance of RBMulticast to be high, due to the fixed value of SPTI, multiple replicated packets still exist in the network and successively contend for the channel within a short period of time. As a result, more back-offs or collisions occur than that of UIX and contribute to the similar but slightly worse performance. For UOX, once the burst of packets is inserted into the buffer of the MAC layer with no time interval before the transmission attempts, the relay nodes receiving packets would contend with the remaining packets in the buffer of the source node and lead to a low packet delivery ratio.

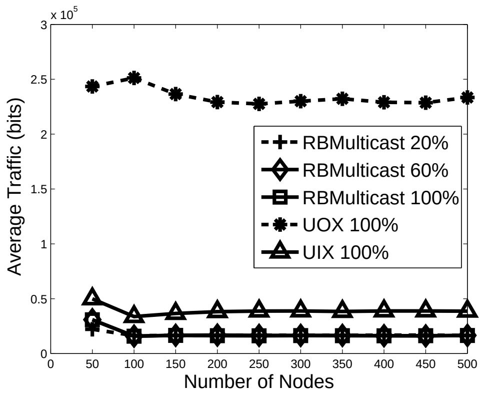
Fig. 6.9(b) shows the latency as a function of the number of nodes. Under low duty cycle and low node density of RBMulticast, since the sleeping times are not synchronized, it is very possible that no relay node candidate can be found in the first attempt, and multiple retransmissions are needed to find a relay node. As the duty cycle and the density increase, more relay node candidates are available and fewer retransmissions are needed, which leads to a decrease in the latency. Fig. 6.9(b) also shows that for low density values, the average latency is high for all three protocols. With an increase in the density, the average latency becomes constant. Since RBMulticast reduces the total number of transmissions to reach all multicast members, the average latency is lower than the other two protocols. Having more time for retransmissions in the improved MAC layer, UIX has a higher average latency than UOX.

The average traffic generated to transmit one data packet to all multicast members is shown in Fig. 6.9(c). It is calculated by dividing the total number of traffic generated to transmit one data packet (RTS/CTS/DATA/ACK) by the packet delivery ratio. Since RBMulticast requires fewer packet transmissions, it generates the least traffic for the delivery of a data packet among the three methods (under 100% duty cycle, UIX generates average traffic roughly 2.3 times compared with RBMulticast). By having fewer retransmissions due to the advantage of the improved MAC, UIX generates less traffic than UOX. In low densities, more retransmissions occur and more packets are dropped because no relay node is found for forwarding. Hence, the average traffic for successfully transmitting one packet to all multicast members is higher than that of higher densities. Also the fact that the average traffic for the three different duty cycles does



(a) Packet delivery ratio vs. number of nodes. (static nodes, 5 sinks)

(b) Average latency vs. number of nodes. (static nodes, 5 sinks)



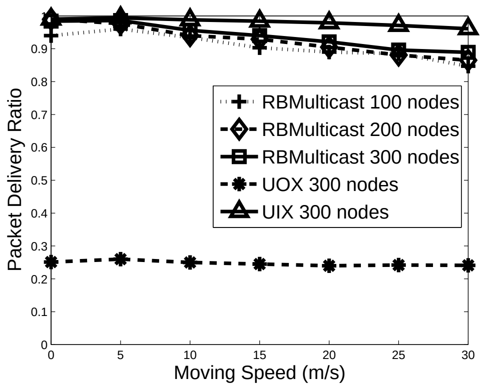
(c) Average traffic for transmitting one data packet vs. number of nodes. (static nodes, 5 sinks)

Figure 6.9: Performance comparisons for RBMulticast: static scenario, 5 sinks.

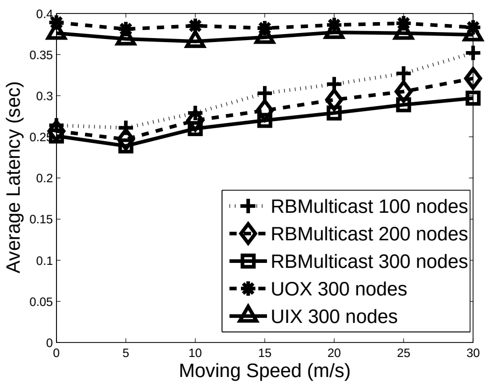
not differ significantly is due to the improved MAC where not many retransmissions are needed to accomplish the delivery of a data packet, even with low duty cycle.

6.4.4.2 Mobile Nodes, Five Sinks

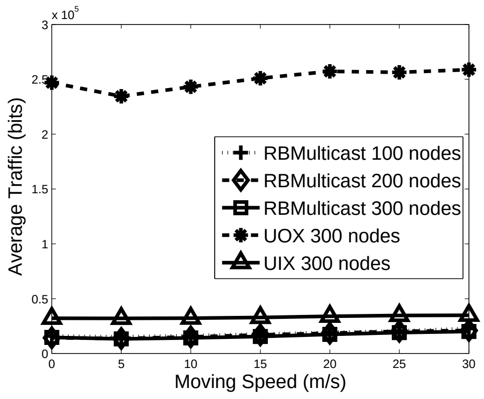
The second set of simulations are performed to investigate the performance of RB-Multicast in mobile scenarios. All intermediate nodes move according to the Random Waypoint mobility model with a certain speed. The source and multicast members are moved inward 25m as compared to Fig. 6.8 to avoid the issues with the ”cluster into the middle” effect of the Random Waypoint model [98, 99]. A duty cycle of 100% is



(a) Packet delivery ratio vs. mobile speed. (mobile nodes, 5 sinks)



(b) Average latency vs. mobile speed. (mobile nodes, 5 sinks)



(c) Average traffic for transmitting one data packet vs. mobile speed. (mobile nodes, 5 sinks)

Figure 6.10: Performance comparisons for RBMulticast: mobile scenario, 5 sinks.

investigated for three different numbers of nodes: 100, 200 and 300.

Fig. 6.10(a) shows the packet delivery ratio as a function of mobile speed. Note that the data points corresponding to 0 m/s show the performance of static networks. All three curves indicate that when the intermediate nodes are moving at low speeds and the node density is low, the performance is slightly better than that when they are static. The reason is that the ”empty holes” that exist in the static scenario when the density is low, can be eased when the nodes move into the ”empty holes” and become relay candidates. When nodes move fast, more link breaks can occur because a receiver moves out of the transmission range of the transmitter. Fig. 6.10(a) shows that UIX performs the best among the three protocols and RBMulticast performs better than UOX due to the same reason as that of the static scenario.

Fig. 6.10(b) shows the average latency as a function of mobile speed. When density is increased, less time is required to finish the transmission. As seen in the figure, RBMulticast has the least latency among the three protocols, for the same reason as in the static scenario.

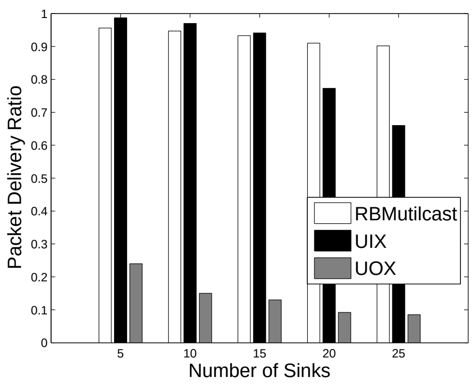
Fig. 6.10(c) shows the average traffic generated to transmit one data packet as a function of mobile speed. When the speed of mobile nodes increases, the average traffic generated per transmission becomes higher due to the increase in the number of retransmissions caused by more link breaks. Note that RBMulticast has the least traffic since it requires the smallest number of hops among the three protocols (under the same number of nodes, UIX generates average traffic of 1.7 to 2.2 times that of RBMulticast).

6.4.4.3 Effect of Number of Sinks

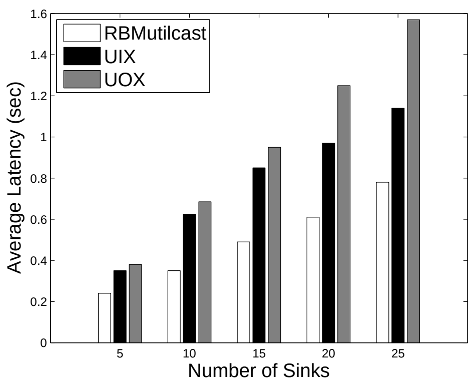
To further test the robustness of RBMulticast, we develop the third simulation scenario where performance is evaluated in terms of the number of sinks when under mobile scenarios. The source node is located at (25, 25). Sinks are located around the upper-right corner of the inner $100m \times 100m$ area with the distance of $1m$ between the adjacent sinks as the rectangular and cross nodes seen in Fig. 300 intermediate nodes move with a speed of 10 m/s inside the $150m \times 150m$ area. The Random Waypoint model is applied. The duty cycle investigated is 100%. The simulation results for the static scenario are similar to that of the mobile scenario, and hence are omitted.

Fig. 6.11(a) shows the packet delivery ratio as a function of the number of sinks. With an increase in the number of sinks, all three protocols have lower packet delivery ratios. In RBMulticast, because more sinks require more replicas of packets, fiercer contention occurs for the channel. Therefore it is expected that the packet delivery ratio will decrease with an increase in the number of multicast members. As seen in Fig. 6.11(a), RBMulticast is more robust to an increase in the number of multicast members. Above 20 members, RBMulticast gives better packet delivery ratio compared to UIX. It is expected that with an increase in the number of sinks, the advantage of RBMulticast over UIX will be larger.

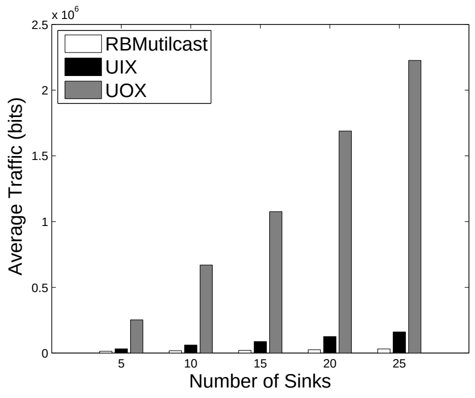
Fig. 6.11(b) shows the average latency as a function of the number of sinks. More



(a) Packet delivery ratio vs. number of sinks. (mobile scenario, 300 nodes)



(b) Average latency vs. number of sinks. (mobile scenario, 300 nodes)



(c) Average traffic for transmitting one data packet vs. number of sinks. (mobile scenario, 300 nodes)

Figure 6.11: Performance comparisons for RBMulticast: mobile scenario, varying number of sinks.

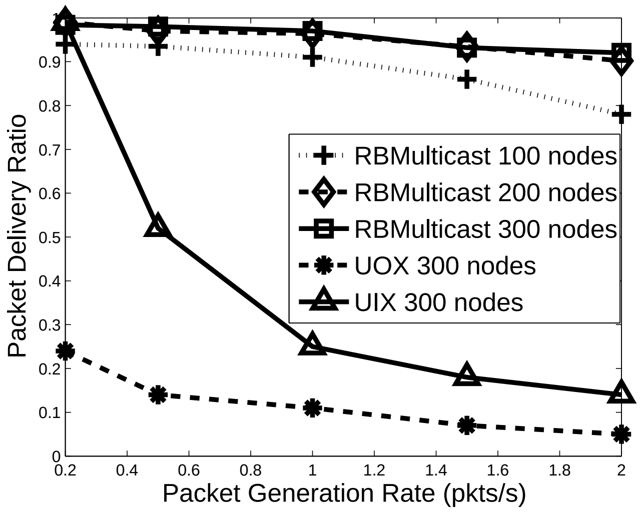


Figure 6.12: Packet delivery ratio vs. data generate rate. (mobile scenario with speed 10 m/s, 5 sinks).

replicated packets lead to more contention and retransmissions, which results in higher latency when the number of sinks increases in RBMulticast. However, since packets in RBMulticast travel through fewer hops, RBMulticast latency is much lower than the other two protocols. Note that when the number of sinks is small, UOX performs better than UIX due to the smaller maximum retransmission count, while when the number of sinks increases, the MAC improvement reveals its advantage by reducing the latency through avoiding contentions.

Fig. 6.11(c) shows the average traffic generated to transmit one data packet as a function of the number of sinks. Because unicast needs to send separate packets to each sink, many paths are repeated and redundant. Hence, with the increase of the number of sinks, RBMulticast has a greater advantage in terms of average traffic. Note that due to the large number of retransmissions with the original XLM MAC, UOX always has a much larger traffic generation than the other two protocols, which verifies the necessity of the MAC improvements.

6.4.4.4 Effect of Packet Generation Rate

To evaluate the performances of the protocols under different data traffic rates, another set of simulations is conducted, where the data generation rate is increased from 0.2

pkts/sec to 2 pkts/sec. All the other parameter values are kept the same as with the mobile scenario investigated in Section 6.4.4.2, fixing the speed of the mobile nodes to 10 m/s. The results, shown in Fig. 6.12, indicate that the packet delivery ratio of UIX drops sharply as the packet generation rate is increased from 0.2 pkts/sec. UOX performs very poorly for all of the packet generation rates investigated. However, as seen in the figure, RBMulticast is much more resilient to changes in the packet generation rate, and provides significant benefits under more saturated traffic conditions in terms of packet delivery ratio.

6.4.4.5 Uniformly Distributed Sinks, Mobile Nodes

A new scenario is developed to further illustrate the performance of RBMulticast. 300 mobile intermediate nodes are randomly deployed in a $150m \times 150m$ scenario with a moving speed of 10 m/s. Sinks are uniformly distributed along the upper edge and right edge (e.g., as in [100]) of the $100m \times 100m$ inner area with the source located in $(25, 25)$, i.e., at the lower-left corner of the inner area.

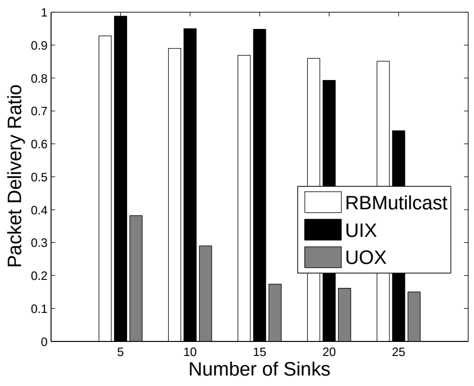
Fig. 6.13(a) shows that when the number of sinks are 10 and 15, RBMulticast performs worse compared with the previous scenario, and UIX has a larger advantage in packet delivery ratio. This is because when multicast members are sparsely distributed, RBMulticast requires splitting more packets, leading to more contentions.

Fig. 6.13(b) and 6.13(c) show that RBMulticast's advantage over UIX in average latency and average traffic is not that evident compared with the previous scenario because more contention leads to a larger delay, and the decrease of the packet delivery ratio directly increases the average traffic to successfully transmit one packet to all multicast members.

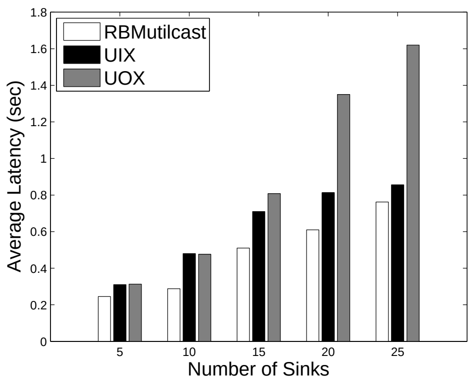
These simulation results shows that the performance of RBMulticast is tightly connected with the location of the sinks. Generally, RBMulticast has a larger advantage compared with unicast when sinks cluster than when they are sparsely distributed.

6.4.4.6 Effect of Location Errors

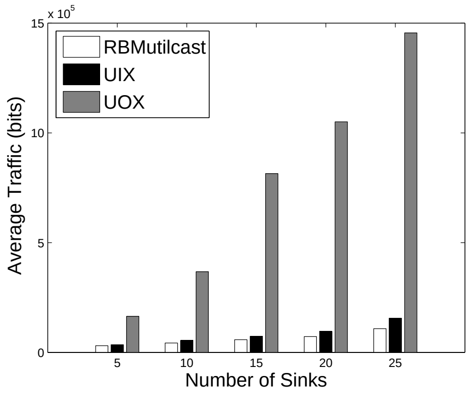
The previous performance evaluations are based on the assumption that nodes can obtain accurate information about their own location, which cannot be provided by many existing locationing systems. In this section, we investigate the effect of location er-



(a) Packet delivery ratio vs. number of sinks. (mobile scenario, 300 nodes).



(b) Average latency vs. number of sinks, mobile scenario, 300 nodes.



(c) Average traffic vs. number of sinks, mobile scenario, 300 nodes.

Figure 6.13: Performance comparisons for RBMulticast: mobile scenario, uniformly distributed sinks.

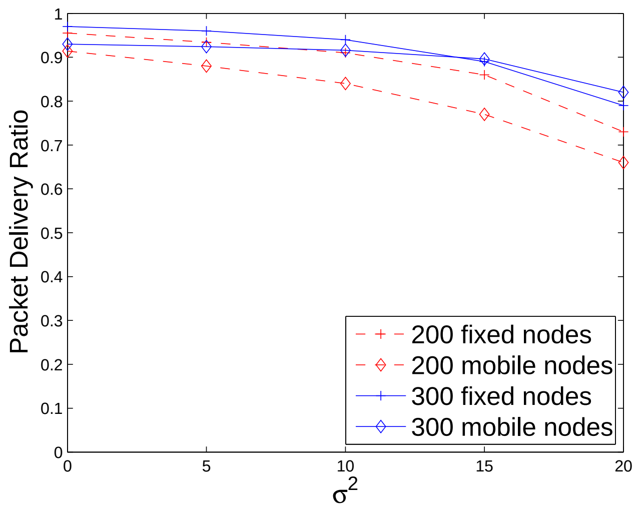


Figure 6.14: Packet delivery ratio with location estimation error of $\mathcal{N}(0, \sigma^2)$: 5 uniformly distributed sinks.

rors on the performance of RBMulticast. We assume the drift between the actual node location and the estimated node location follows a normal distribution $\mathcal{N}(0, \sigma^2)$. Both static and mobile scenarios are implemented, with the same parameter values defined in Sections 6.4.2 and 6.4.4.5.

Fig. 6.14 shows that the packet delivery ratio remains high when variance σ^2 is low (i.e., the estimated location is close to the actual location), and it decreases as variance increases, as expected. Additionally, Fig. 6.14 shows that RBMulticast is more robust to location errors with 300 nodes than with 200 nodes. This is because the nodes in the forwarding regions may misjudge their location as outside of the forwarding regions. In this sense, the number of nodes participating in the contention for data forwarding is less than that when the nodes have accurate location information. This results in potentially longer hop distances and hence worse packet delivery ratio performance.

6.5 Tmote Sky Implementation

We show the realization of RBMulticast through a test-bed implementation using Tmote Sky motes [73] and through simulation using TOSSIM [101].

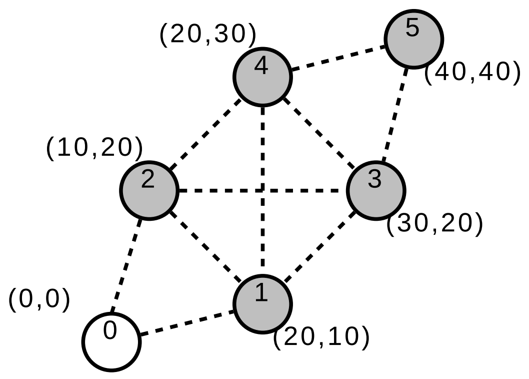


Figure 6.15: The experimental network for comparing results in the Tmote Sky implementation and the TOSSIM simulations. Node 0 is the source node and the shaded nodes are multicast destination nodes.

We test the RBMulticast protocol in a highly dynamic scenario, where nodes have a very short frame time of 100 ms. For example, a duty cycle of 0.2 (20%) means that in every 100 ms, nodes will turn their radios on for 20 ms and then go to sleep for the remaining 80 ms if not transmitting or receiving. We use this highly dynamic scenario to demonstrate the advantages of stateless multicast. That is, RBMulticast can achieve high success rates and low latency in a highly dynamic scenario where structured (e.g., tree) approaches are difficult to employ.

For the experiment, we collect data from six Tmote Sky sensor nodes arranged in the topology shown in Fig. 6.15. The statistics shown in Fig. 6.16 compare the results of TOSSIM simulations with the results of the Tmote Sky experiments. In this figure, we see that the RBMulticast implementation results match closely with the simulation results that assuming ideal conditions. The physical experimental results show that RBMulticast is lightweight in computation even for low processing power sensor devices. The success rate is slightly higher in the TOSSIM simulations than in the Tmote Sky implementation. This is because the Tmote Sky mote (which contains a CC2420 radio chip) requires 8 ms plus some CPU overhead to send/receive a packet, and this is on the same order of the time it takes the radio to change its status in our experiments (e.g., 20 ms for a duty cycle of 0.2). In this experiment, we set the MAC layer retry limit to 4 times before a packet is dropped, and the increased number of packets sent for the Tmote Sky implementation is hence due to the extra resent RTS packets.

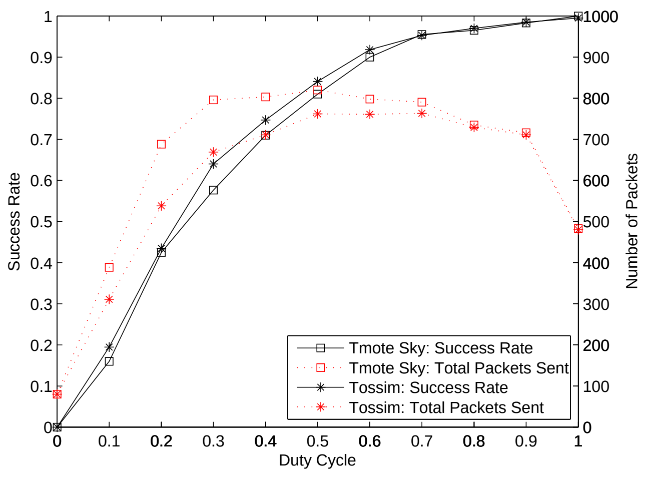


Figure 6.16: Comparison of Tmote Sky implementation to TOSSIM simulations.

6.6 Conclusions

Current multicast protocols generally rely on various tree structures and hence intermediate nodes need to maintain tree states or routing states for packet delivery. In this chapter, we presented a new stateless multicast protocol for ad-hoc networks called Receiver-Based Multicast (RBMulticast). RBMulticast uses geographic location information to route multicast packets, where nodes divide the network into geographic "multicast regions" and split off packets depending on the locations of the multicast members. RBMulticast stores a destination list inside the packet header; this destination list provides information on all multicast members to which this packet is targeted. Thus, there is no need for a multicast tree and therefore no tree state is stored at the intermediate nodes.

RBMulticast also utilizes a receiver-based MAC layer to further reduce the complexity of routing packets. Because we assume that the receiver-based MAC protocol can determine the next hop node in a distributed manner, the sender node does not need a routing table or a neighbor table to send packets but instead uses a "virtual node" as the packet destination. Thus RBMulticast requires the least amount of state of any existing multicast protocol.

Our simulations and implementation of RBMulticast showed that it can achieve high success rates, low latency and low overhead in terms of the number of bits transmitted in the network for both static and dynamic scenarios, making RBMulticast well suited for both mobile and stationary ad-hoc network environments.

Chapter 7

Energy-Efficient Duty Cycle Assignment for Receiver-Based Convergecast

Wireless sensor networks (WSNs) are one type of emerging network that require new protocols to support their unique features. In particular, sensor networks are extremely resource limited, hence requiring protocols to make optimal use of this resource. Duty cycling is often used to reduce the energy consumption caused by idle listening in WSNs. Most studies on WSN protocols define a common duty cycle value throughout the network to achieve synchronization among the nodes. On the other hand, a few studies propose adaptation of the duty cycle according to uniform traffic conditions, which is beneficial assuming one-to-one traffic patterns that result in evenly distributed packet traffic.

In this work, we consider the convergecast communication pattern commonly observed in WSNs. In convergecast communication, the packet traffic observed around the sink node is much higher than the traffic observed far from the sink, i.e., nodes with different distances to the sink node receive and must relay different amounts of traffic. Additionally, we utilize receiver-based protocols, which enable nodes to communicate with no synchronization or neighbor information, and hence do not require all nodes in the network to have the same duty cycle.

In this chapter, we model the expected energy consumption of nodes utilizing receiver-based protocols as a function of their duty cycle and their distance to the sink node. Using this analysis, we derive a closed-form formula for the duty cycle that minimizes the expected energy consumption at a given distance. Moreover, we propose an adaptation method for the derived distance-based duty cycle, based on local observed traffic. Performance evaluations of the two proposed duty cycle assignment methods show that they greatly improve the energy efficiency without sacrificing packet delivery ratio or delay significantly.

7.1 Introduction

Duty cycling, where a node is periodically placed into the sleep mode, is an effective method of reducing energy dissipation in Wireless Sensor Networks (WSNs). The lower the duty cycle, the nodes can sleep longer and the more energy they will save, whereas the fewer nodes are available to participate in data routing at any given time, which will increase transmission latency and decrease the throughput. Thus, there is a trade-off between energy efficiency, transmission latency, and throughput, determined by the duty cycle used in the network.

Duty cycle is typically fixed throughout the network, with all nodes utilizing the same duty cycle. However, this may not provide the best overall performance for the network. Many sensor network applications require convergecast communication, where data from sensors are transmitted to a sink in the network. In this type of communication pattern, nodes close to the sink must transmit much more data than nodes far from the sink, and hence the duty cycles of the nodes should be adjusted appropriately to ensure energy efficiency while meeting traffic demands and keeping latency low.

Recently, a new class of protocols, called receiver-based routing, has been proposed as a means of allowing communication when nodes are not aware of the exact duty cycle of their neighbors. In receiver-based routing, receivers contend to be the next-hop router of a packet, and the transmitter selects the “best” receiver under a given optimality criteria to become the next hop for transmission. For example, in the receiver-based protocol Implicit Geographic Forwarding (IGF) [102], all receivers contend to be the next-hop router when they hear a packet route request, and the transmitter selects the receiver that is closest to the sink as the next hop. Specifically, the transmitter initiates communication by sending an RTS packet that indicates the transmitter's location and the location of the sink. Nodes that hear the RTS packet first determine whether they make forward progress to the sink, and, if so, they calculate their distance to the sink. After a delay proportional to their distance to the sink, nodes send a CTS packet back to the transmitter. The first node that sends a CTS packet is selected as the next hop by the transmitter, and the transmitter forwards the data packet to that node.

Researchers have analyzed the performance of receiver-based routing through mathematical models [34] [35] and shown that receiver-based routing protocols perform well in terms of hop distance, energy and latency. Unicast traffic is assumed in these works, hence, for convergecast traffic further studies are required. Extensions to traditional receiver-based routing have included providing information about link quality for making routing decisions [15], and supporting multiple paths by strategically selecting relay nodes and employing adaptive rate control [31]. Utilizing duty cycling with receiver-based routing and convergecast data patterns, it is clear that a network-wide fixed duty cycle will not provide the optimal trade-off between energy efficiency and latency.

Adapting the duty cycle to the local traffic was proposed in PMAC [58], where the sleep-wakeup schedule is represented by a string of bits that are updated each period using local traffic information available at the node. These schedules are exchanged at the end of each period, so that neighboring nodes are aware of each others' schedules. Another adaptive duty cycle approach, ALPL, adjusts a node's duty cycle according to the node's neighbors' duty cycles in order to support the data flows it receives [59]. However, none of these approaches optimize the duty cycle for convergecast data patterns and receiver-based routing.

In this chapter, we derive a mathematical model to determine the energy dissipation of a node as a function of its duty cycle and its distance to the sink for convergecast data patterns and receiver-based routing. Using this model, we find the duty cycle as a function of node distance to the sink to minimize the expected energy dissipation. Additionally, in order to balance energy efficiency and latency, we develop a traffic-adaptive duty cycle approach that begins with the distance-based duty cycle assignment and adapts the duty cycle based on current local traffic patterns observed by the node. In receiver-based protocols, the number of retransmitted RTS packets provides an indication of the traffic. Under heavy traffic, nodes must generate many retransmitted

RTS packets. If the number of retransmitted RTS packets outnumbers the number of original RTS packets, nodes should increase their duty cycle in order to alleviate the traffic congestion; otherwise, they should decrease their duty cycle to save energy. This approach allows the duty cycle to be tuned to trade-off energy and latency for observed local traffic patterns.

Specifically, two duty cycle assignment methods are proposed in this chaper: Distance-based Duty Cycle Assignment (DDCA), where the duty cycle is assigned to each node based on the node to sink distance, and Traffic-Adaptive Distance-based Duty Cycle Assignment (TDDCA), where the duty cycle is initialized to the one given by the DDCA method and adapted to the traffic as explained above. Simulation results show that DDCA and TDDCA reduce energy consumption compared with the commonly used, network-wide constant duty cycle method. Additionally, TDDCA reduces latency at the expense of a small increase in energy consumption compared with DDCA, which indicates that TDDCA is able to trade-off the lower latency of the network-wide constant duty cycle method and the energy efficiency of the distant-based duty cycle method (DDCA).

7.2 Distance-based Duty Cycle Assignment Methods

The traffic relayed at a node is related to its distance to the sink, the number of source nodes in the network, the packet traffic generated by each source node, and the node density. In this section, we present this relationship analytically, then, given the average traffic observed at a node, we derive the expected duty cycle for minimizing the expected energy consumption of the node.

7.2.1 Traffic Rate Analysis

For the analysis, we assume a circle area with the sink located in the center and the nodes including the sources uniformly randomly allocated as illustrated in Fig. 7.1, where r_T is the transmission range. We define the n^{th} ring to be the ring whose inner circle is $(n-1)r_T$ away from the sink with width r_T. Hence, the n^{th} ring contains the nodes that are n-hops away from the sink. Let there be N_n nodes in this ring. The average traffic that must be relayed by all of the nodes located in the n^{th} ring per unit

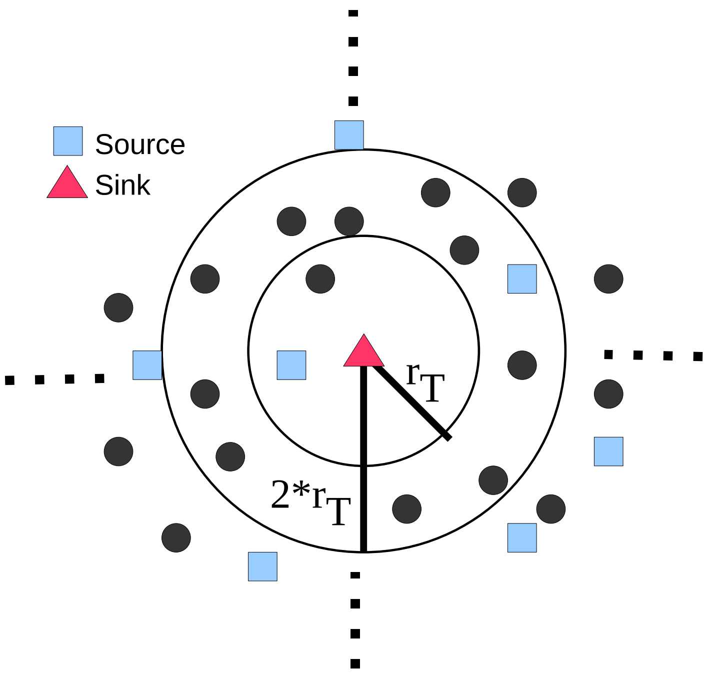


Figure 7.1: Sample network topology.

time, Γ_n, is the summation of the traffic generated by the source nodes in the n^{th} ring and within the rings outside of the n^{th} ring per unit time, i.e.,

$$\Gamma_n = \lambda_g \rho_s \pi (R^2 - [(n-1)r_T]^2), \tag{7.1}$$

where λ_g is the average traffic generation rate of the source nodes, ρ_s is the density of source nodes, and R is the radius of the network area.

Since Γ_n is the average traffic relayed by all nodes per unit time in the n^{th} ring, a node within that ring relays a traffic with a mean $\lambda_r = \Gamma_n / N_n$ packets per unit time. A node with a distance r to the sink resides in the $n = \lceil \frac{r}{r_T} \rceil$ ring and the number of nodes in the n^{th} ring is

$$N_n = \rho_r \pi \left\{ (nr_T)^2 - [(n-1)r_T]^2 \right\}, \tag{7.2}$$

where ρ_r is the density of nodes. Hence, the average traffic rate of a node at distance r, λ_r, is

$$\lambda_r = \frac{\lambda_g \rho_s \pi \left\{ R^2 - [(\lceil \frac{r}{r_T} \rceil - 1) r_T]^2 \right\}}{\rho_r \pi \left\{ [(\lceil \frac{r}{r_T} \rceil) r_T]^2 - [(\lceil \frac{r}{r_T} \rceil - 1) r_T]^2 \right\}}. \tag{7.3}$$

7.2.2 Duty Cycle for a Given Expected Traffic Rate

The time required for a transmission and the energy efficiency of the network is closely related to the duty cycle values used. Higher duty cycle values provide more nodes available for data routing, such that the possibility to have no relay nodes is decreased and a lower latency is achieved, yet they consume more energy. In this section, we derive the duty cycle that minimizes the energy consumption for a given traffic rate.

In [35], a similar derivation is done for unicast traffic, where every node can be a source or a destination. We adapt the analysis presented in [35] for the following MAC protocol modifications proposed. Although a receiver-based MAC protocol is analyzed in [35], our simulation results showed a high number of collisions and high CTS traffic load for the MAC protocol investigated therein. To reduce the number of collisions and the CTS traffic load, the MAC protocol is modified as follows. In [35], the relay region (locations with geographic advancement to the sink) is divided into N_p priority regions, and each region is assigned a contention slot such that priority region i is assigned the ith slot in the contention window. We assign each priority region N_r CTS contention slots, such that priority region i is assigned the slots $((N_i - 1) \times N_r, N_i \times N_r - 1)$. This reduces CTS collisions, as all nodes in priority region i can select one of the N_r CTS contention slots to send their CTS packet.

The following duty cycle analysis is based on the idea that the expected energy consumption of a sensor node is proportional to the expected total awake time, t_l, of the node. This is because, the radio idle listening power is approximately the same as the transmission and reception power in WSNs [103]. Hence, a constant power value P is assumed for idle listening, transmission, and reception.

Let N denote the average number of nodes within a node's transmission range, d denote duty cycle, and λ_r denote the average traffic rate of a node located at distance r to the sink node given in (7.3). Assuming a Poisson or uniform packet generation rate, the average traffic rate of a node follows the Poisson distribution. The probability that a node detects no traffic can be calculated to be $e^{-\lambda_r N T_L}$ where T_L is its listen period at each cycle and $\lambda_r N$ is the average packet arrival rate within its transmission range. Thus, the probability that a node detects any ongoing traffic is $p_0 = 1 - e^{-\lambda_r N T_L}$. If ξ is the ratio of the relay region (i.e., the region in which nodes make forward progress to the sink) to the transmission area, $p_0 \xi$ is the probability of a node detecting ongoing traffic and residing in the relay region of that traffic.

When a node has a packet to send, it sends an RTS packet and keeps retransmitting the RTS packet until receiving a CTS packet. The expected number of RTS transmissions needed before the first successful RTS/CTS handshake is

$$\begin{aligned} \textstyle\sum_{i=1}^{\infty} i(1-p_1)^i p_1 &= \tfrac{1-p_1}{p_1} \\ &= (e^{\xi dN}-1)^{-1} \end{aligned} \tag{7.4}$$

where $p_1 = 1 - e^{-\xi dN}$ is the probability that at least one node replies to the RTS packet, since the number of nodes residing in an area can be approximated by Poisson distribution for uniformly random deployment [104]. For each retransmission, the node sends out an RTS packet and waits for $N_p \times N_r$ CTS slot durations. The expected time needed before the first successful RTS/CTS handshake, t_H, is then

$$t_H = (e^{\xi dN}-1)^{-1}(T_{RTS} + N_p N_r T_{CTS}), \tag{7.5}$$

where T_{RTS} and T_{CTS} are the transmission delays for RTS and CTS packets, respectively.

As illustrated in Fig. 7.2, the expected total time for a complete RTS, CTS, DATA and ACK packet communication is

$$t_C = T_{RTS} + xT_{CTS} + T_{DATA} + T_{ACK}, \tag{7.6}$$

where x represents the number of CTS contention slots up to and including the first successful CTS packet, T_{DATA} and T_{ACK} are the required times for DATA and ACK packets, respectively. The formula for x can be calculated from a standard CSMA model, and we omit it here for the sake of brevity. Therefore, the expected total time for a node to transmit a packet, including all the failed RTS packets and the successful data exchange is $t_t = t_H + t_C$.

The expected time for a node to receive an RTS packet during a listening period is $\frac{T_L}{2}$. An approximation for the probability that a node wins the contention and is selected as the relay node is given in [35] as $\dfrac{1-e^{-\xi dN}}{\xi dN}$. Then, the average active time of a node that receives traffic and that resides in the relay region of the sender node is

$$t_1 = \tfrac{T_L}{2} + \frac{1-e^{-\xi dN}}{\xi dN} t_C + \left(1 - \frac{1-e^{-\xi dN}}{\xi dN}\right)\tfrac{T_L}{2}. \tag{7.7}$$

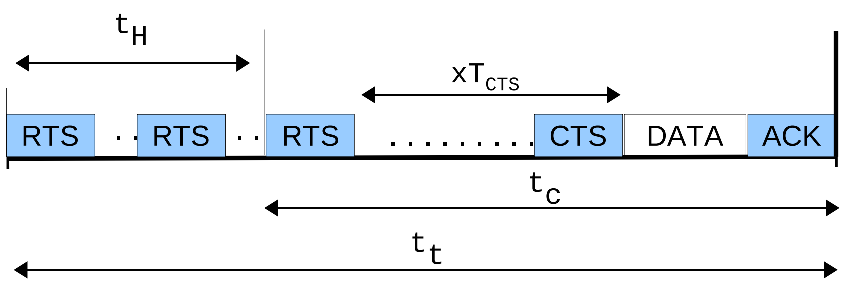


Figure 7.2: Representation of packet exchange durations.

Finally, the expected time a node is awake during one listen period is $t_l = (1 - p_0\xi)T_L + p_0\xi t_1$, where $(1 - p_0\xi)$ represents the probability that either a node hears no traffic or hears some traffic but is not in the relay region, in which case the node is awake for T_L time.

The expression for the expected energy consumption $\bar{P}$, then, can be derived as

$$
\begin{aligned}
\bar{P} &\simeq P \times \frac{t_l}{T_L/d} + \lambda_r P t_t \\
&\simeq P\{d + \lambda_r[(e^{\xi dN} - 1)^{-1}(T_{RTS} + N_p N_r T_{CTS}) \\
&\quad +(2 - e^{-\xi dN})(T_{RTS} + xT_{CTS} + T_{DATA} + T_{ACK})]\} \\
&\simeq P\Big\{d + \lambda_r\{[(e^{\xi dN} - 1)^{-1}(1 + N_p N_r) + 2+ \\
&\quad x]T_{CTS} + 2T_{DATA}\}\Big\} \\
&\simeq P\{d + \lambda_r[(e^{\xi dN} - 1)^{-1} N_p N_r T_{CTL} + 2T_{DATA}]\},
\end{aligned}
\tag{7.8}
$$

where $T_{RTS} \simeq T_{CTS} \simeq T_{ACK} = T_{CTL}$, and $1 - e^{-\lambda_r N T_L} \simeq \lambda_r N T_L$ when $\lambda_r N T_L << 1$. Since xT_{CTS} is dominated by the other components in the formula, it is eliminated as a simplification.

We take the derivative of the expected energy consumption function with respect to d and set it to zero to find the duty cycle that minimizes the expected energy consumption. The duty cycle resulting in E_{min} is $d_{opt} = \frac{\log[\frac{\alpha+2+\sqrt{\alpha(\alpha+4)}}{2}]}{\xi N}$ where $\alpha = \lambda_r \xi N N_p N_r T_{CTS}$. Finally, the mathematical relation between duty cycle and average traffic rate is derived. The value of λ_r for a node is found with the analysis presented in Section 7.2.1.

7.2.3 Duty Cycle Assignment Methods Proposed

The Distance-based Duty Cycle Assignment (DDCA) method defines the duty cycle of a node to be the duty cycle based on the analysis presented in Sections 7.2.1 and 7.2.2. Since analysis do not take packet contention and collision into consideration, we round up the duty cycle found by DDCA to be $\frac{\lceil 100d \rceil}{100}$. Although the analysis presented considers expected traffic observed by a node at a given distance, in practice the actual traffic loads vary per node and over time. Moreover, the entire analysis focuses on minimizing energy consumption while leaving the end-to-end delay performance as a later concern. Aiming to solve these problems, we also propose a distance-based duty cycle assignment scheme combined with the actual traffic pattern observed. In general, the receiver-based protocols do not exchange any traffic information between nodes to achieve stateless communication. However, RTS packets can be used to observe the traffic load. The number of retransmitted RTS packets increases either when a node's duty cycle is too low and no relay candidates can be found, or when the traffic load is too high and the high contention of nodes causes collisions of the RTS packets from different transmitters. For either case, increasing the duty cycle would increase the probability of successful communication.

We introduce a piggyback flag to the original packet header of the RTS packet to indicate whether this packet is being retransmitted or not. A counter is also set in every node to record the numbers of the initial and retransmitted RTS packets. If the total number of the received retransmitted RTS packets in the current cycle outweighs the total number of the received initial RTS packets, it indicates severe contention in the neighborhood, and the duty cycle of the node is increased to mitigate the traffic load. Otherwise, the duty cycle is decreased every cycle down to a minimum of 1% to minimize the energy consumption. This method is called Traffic-Adaptive Distance-based Duty Cycle Assignment (TDDCA). TDDCA is expected to improve the latency performance, since it takes into account not only the distance-based duty cycle assignment, but also the spatiotemporal traffic information in a particular network deployment.

7.3 Performance Evaluation of Duty Cycle Assignment

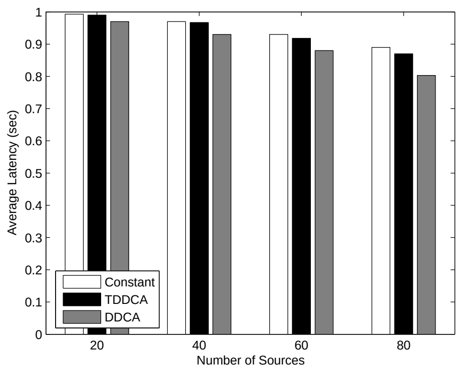
Simulations are performed using the OPNET simulator to compare the two methods proposed, namely DDCA and TDDCA, with the network-wide constant duty cycle assignment method. In the network-wide constant duty cycle method, the duty cycle is set to the duty cycle found by the DDCA method for the nodes one hop away from the sink, such that a high packet delivery ratio is guaranteed.

The performance metrics evaluated are packet delivery ratio, average energy consumption, and average latency. The radius of the target area R is set to be $90\,m$ and the transmission range r_T for all nodes is set to be $30\,m$. For simplicity, we assume the relay region ratio is constant and set to 0.4 when determining the DDCA duty cycle, and the power for transmission, reception and idle listening is set to 1 unit. The sink is located in the center of the area, where 400 nodes are uniformly randomly deployed. In TDDCA, the duty cycle is changed by 1% every listening interval based on the observed RTS retransmissions.

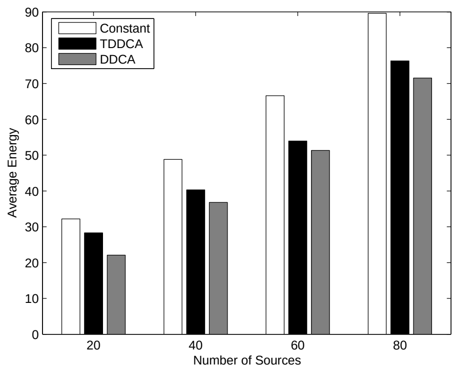
Two sets of simulations are performed to investigate the performance of the presented duty cycle assignment methods for a varying number of sources and a varying packet generation rate, λ_g. The effect of the number of sources is investigated for a packet generation rate of 0.5 packet/sec and the effect of the packet generation rate is investigated for 40 sources.

The packet delivery ratio (PDR) values achieved by the three methods are presented in Fig. 7.3(a) and Fig. 7.4(a). In all three methods, the PDR results are very close and higher than 97% for light traffic loads. With an increase in traffic load, the constant duty cycle method performs the best because its higher duty cycle can provide more awake nodes to participate in data routing. The slightly worse performance of TDDCA compared to the constant duty cycle method indicates that the fixed increments and decrements in duty cycle is not efficient in terms of PDR. One alternative is to use varying duty cycle increments and decrements as proposed in [105].

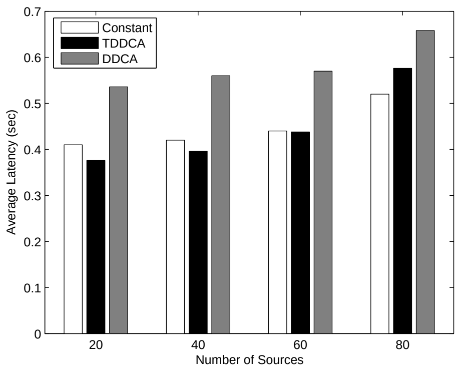
While PDRs are approximately the same using all three methods, Figs. 7.3(b) and 7.4(b) both show that TDDCA and DDCA are more energy-efficient than the constant duty cycle method, and that DDCA performs better than TDDCA. DDCA reduces energy dissipation between 21% and 32% compared to the constant duty cycle method, while TDDCA reduces energy dissipation between 12% and 19% compared to the con-



(a) Packet delivery ratio



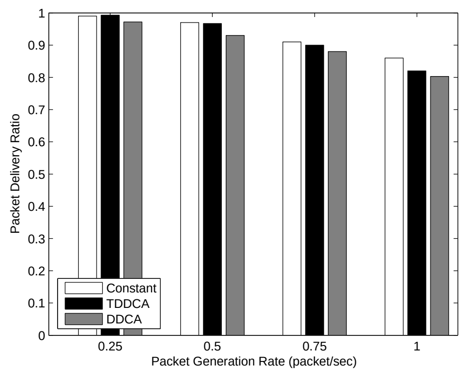
(b) Average energy consumption



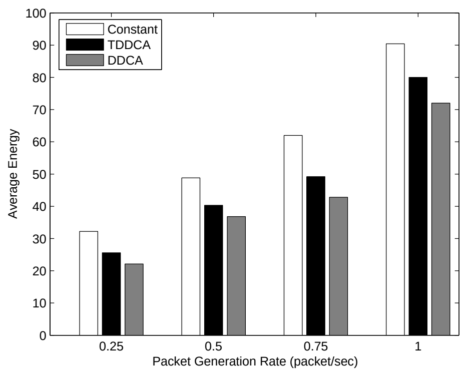
(c) Average latency

Figure 7.3: Simulation results in terms of the number of sources.

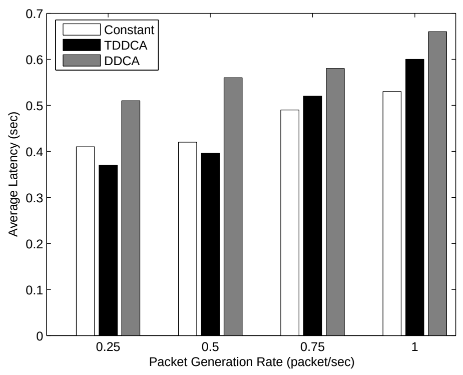
stant duty cycle method. Because the entire network is likely to generate more retransmitted RTS packets than original RTS packets, TDDCA increases duty cycle more often than decreasing it. The reason is as follows: in the area near the sink where traffic is heavy, available nodes that receive the first RTS packet turn to a busy state until they win the contention or receive a CTS packet from another node for the same RTS packet. In this busy state, receivers do not reply RTS packets from other transmitters, which results in retransmitted RTS packets even when there are awake nodes within nodes' transmission ranges. On the other hand, in the area far from the sink where traffic is light, the duty cycles of nodes are low such that it is possible that there are no awake nodes that can hear an RTS packet when it is broadcasted. Thus, retransmitted RTS packets are generated in this case as well. Generally the fact that TDDCA



(a) Packet delivery ratio



(b) Average energy consumption



(c) Average latency

Figure 7.4: Simulation results in terms of source packet generation rate λ_g.

increases the duty cycle more often than decreasing it leads to its larger average energy consumption than DDCA.

Figs. 7.3(c) and 7.4(c) show that TDDCA performs the best in terms of latency. In light traffic, TDDCA achieves better latency values compared with DDCA, e.g., latency using TDDCA is 30% less than latency using DDCA when the number of sources is 20. Since nodes are likely to increase their duty cycle rather than to decrease it, in TDDCA there are more nodes available to contend for the channel and latency is reduced compared with DDCA. It is also shown that in heavy traffic, TDDCA performs worse in terms of latency compared with the constant duty cycle method. This is because under the severe impact of packet collisions and contention, traffic patterns vary between every listening interval such that a simple comparison between the number of original

RTS packets and retransmitted RTS packets cannot reflect the current level of traffic accurately enough. Hence, the method of changing duty cycles by 1% in each listening interval is not effective to achieve a low latency in high traffic conditions.

In summary, both DDCA and TDDCA are more energy-efficient than the constant duty cycle method, while achieving similar packet delivery ratio performance. Compared with DDCA, TDDCA has an advantage in terms of latency.

7.4 Conclusions

In this chapter, we derived the duty cycle for a node as a function of its distance to the sink to minimize expected energy consumption for convergecast traffic patterns and receiver-based routing. Based on our analysis, we developed two duty cycle assignment algorithms. Simulation results show that both methods decrease energy consumption compared with the constant duty cycle method by up to 32% for the scenarios investigated. The traffic-adaptive distance-based duty cycle assignment method achieves energy improvements without sacrificing the latency and throughput significantly. The analysis can be extended as future work to improve the performance of distance-based duty cycle assignment in heavy traffic scenarios, by taking the packet collisions and contention into account.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis addresses some of the fundamental problems in stack architectures and protocol design for emerging wireless networks. We are convinced by our preliminary results that network dynamics can be supported. Through stacks that support multiple protocols and information sharing, and through the use of stateless network protocols, networks with multiple substrates can be efficiently implemented in our UPS framework. The contributions of this thesis are summarized as follows.

- A single layer cross-layer protocol XLM [15] is implemented into the X-Lisa information-sharing protocol stack architecture. Simulation results shows that X-Lisa can be successfully used as a flexible information sharing protocol stack, but it lacks the ability to support multiple protocols in the same layer simultaneously, and it does not provide a universal information-sharing interface.

- A new protocol stack architecture called UPS (Universal Protocol Stack) is proposed. UPS standardizes a modular protocol stack that supports concurrent protocol operation and information sharing.

- Examples of implementing UPS are demonstrated through simulations. The results demonstrate that the UPS framework can be applied to existing protocols with no interference between different protocol modules in the same layer.

- A new virtual interface within the UPS framework to support multiple radio interfaces is proposed. Results from simulations using this virtual interface show that the use of a virtual interface can improve the network performance.
- A stateless receiver-based multicast protocol (RBMulticast) is developed, where receivers decide the best way to forward the multicast traffic. RBMulticast removes the need for costly state maintenance, which makes RBMulticast ideally suited for multicasting in dynamic networks.
- An adaptation method for distance-based duty cycling is proposed for receiver-based convergecast networks. Based on local observed traffic, a closed-form energy formula for the duty cycle is derived.

8.2 Future Work

Possible future work includes designing approaches to select routers and interfaces given the multiple choices available in a device and supported by the UPS architecture. Specifically: 1) Layer 3/routing level decisions to choose the relays efficiently for multihop networks with the use of multi-radio devices; and 2) cross-layer interactions of the MAC and routing layers to best support application QoS goals. The goal is to explore the best network/interface combinations for networks that support multi-radio devices.

8.2.1 Routing Layer Protocol Selection

The terms *heterogeneous networks* and *hybrid networks* have been widely adopted in the literature to represent the integration of cellular networks and WiFi. The related studies target architectural support, or provide the throughput capacity of such integration.

The emerging IEEE 802.21 standard has the goal of supporting handovers between specific network types, specifically GSM, WiFi, Bluetooth, 802.11 and 802.16. This standard will provide a tool for packets to seamlessly switch networks among the investigated networks. However, how to select the best network is still an unsolved research problem.

Chapter 4 defined a protocol architecture that enables us to run multiple protocols in the same stack layer. Utilizing this architecture, we can support multi-radio devices with a common routing protocol. An important area of future work is to look at the selection of the route when links from different networks may be traversed by the packet. This type of routing enables the nodes in the network to utilize all resources available in any co-existing networks and bridge the different networks when beneficial.

8.2.2 Cross-Layer Protocol Selection

Once the layer 3 and layer 2 protocol selection method is decided, a generic solution that integrates multi-layer information can be designed accordingly. The generic solution will track such information to predict the network performance. A protocol architecture can be developed that incorporates both layer 3 and layer 2 cross-layer decisions for efficient resource utilization while supporting end-to-end application goals.

Chapter 9

Bibliography

[1] V. T. Raisinghani and S. Iyer, “Cross-layer design optimizations in wireless protocol stacks,” *Computer Communications*, vol. 27, no. 8, pp. 720–724, 2004.

[2] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, “TinyOS: An operating system for sensor networks,” *Ambient Intelligence*, pp. 115–148, 2005.

[3] A. Boukerche, H. A. B. F. Oliveria, E. F. Nakamura, and A. A. F. Loureiro, “Vehicular ad hoc networks: A new challenge for localization-based systems,” *Comput. Commun.*, vol. 31, no. 12, pp. 2838–2849, July 2008.

[4] K. Akkaya and M. F. Younis, “A survey on routing protocols for wireless sensor networks,” *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

[5] I. F. Akyildiz, X. Wang, and W. Wang, “Wireless mesh networks: a survey,” *Comput. Netw. ISDN Syst.*, vol. 47, no. 4, pp. 445–487, 2005.

[6] B. Zhen, H.-B. Li, and R. Kohno, “IEEE body area networks and medical implant communications,” ser. BodyNets ’08, 2008, pp. 26:1–26:4.

[7] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, “A multi-radio unification protocol for IEEE 802.11 wireless networks,” ser. BroadNets ’04, 2004, pp. 344 – 354.

[8] E. Gustafsson and A. Jonsson, “Always best connected,” *Wireless Communications, IEEE*, vol. 10, no. 1, pp. 49 – 55, feb. 2003.

[9] C. Chereddi, P. Kyasanur, and N. H. Vaidya, “Net-X: a multichannel multi-interface wireless mesh implementation,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, no. 3, pp. 84–95, 2007.

[10] A. Farago, A. Myers, V. Syrotiuk, and G. Zaruba, “Meta-MAC protocols: automatic combination of MAC protocols to optimize performance for unknown conditions,” *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 9, pp. 1670 –1681, sep. 2000.

[11] TinyOS, “http://www.tinyos.net/,” 2008. [Online]. Available: http://www.tinyos.net/

[12] A. Balasubramanian, R. Mahajan, and A. Venkataramani, “Augmenting mobile 3G using WiFi,” ser. MobiSys ’10, 2010, pp. 209 – 222.

[13] E. H. Ong and J. Khan, “Dynamic access network selection with QoS parameters estimation: A step closer to ABC,” may 2008, pp. 2671 –2676.

[14] C. J. Merlin, C.-H. Feng, and W. B. Heinzelman, “Information-sharing protocol architectures for sensor networks: The state of the art and a new solution,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 13, no. 4, pp. 26–38, 2009.

[15] I. Akyildiz, M. Vuran, and O. Akan, “A cross-layer protocol for wireless sensor networks,” in *Proc. of CISS 2006*, March 2006.

[16] C.-H. Feng, I. Demirkol, and W. B. Heinzelman, “UPS: Unified protocol stack for wireless sensor networks,” *MobiQuitous*, 2009.

[17] C.-H. Feng and W. B. Heinzelman, “RBMulticast: Receiver based multicast for wireless sensor networks,” *IEEE Wireless Communications and Networking Conference (WCNC ’09), April 2009.*

[18] C.-H. Feng, Y. Zhang, I. Demirkol, and W. B. Heinzelman, “Stateless multicast protocol for ad-hoc networks,” *IEEE Transactions on Mobile Computing*, vol. 99, no. PrePrints, 2011.

[19] Y. Zhang, C.-H. Feng, I. Demirkol, and W. B. Heinzelman, "Energy-efficient duty cycle assignment for receiver-based convergecast in wireless sensor networks," ser. GLOBECOM, 2010.

[20] N. C. Hutchinson and L. L. Peterson, "The X-Kernel: An architecture for implementing network protocols," *IEEE Trans. Softw. Eng.*, vol. 17, no. 1, pp. 64–76, 1991.

[21] S. W. O'Malley and L. L. Peterson, "A dynamic network architecture," *ACM Trans. Comput. Syst.*, vol. 10, no. 2, pp. 110–143, 1992.

[22] J. Touch and V. Pingali, "The RNA metaprotocol," *Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on*, pp. 1–6, Aug. 2008.

[23] M. Conti, G. Maselli, G. Turi, and S. Giordano, "Cross-layering in mobile ad hoc network design," *Computer*, vol. 37, no. 2, pp. 48–51, Feb 2004.

[24] H. Füßler, M. Torrent-Moreno, M. Transier, A. Festag, and H. Hartenstein, "Thoughts on a Protocol Architecture for Vehicular Ad-Hoc Networks," in *Proc. of 2nd International Workshop in Intelligent Transportation (WIT 2005)*, Hamburg, Germany, 03 2005, pp. 41–45.

[25] R. Kumar, S. PalChaudhuri, C. Reiss, and U. Ramachandran, "System support for cross-layering in sensor network stack," *Proceedings of the International Conference on Mobile Ad Hoc and Sensor Networks, Hong Kong*, pp. 793–807, 2006.

[26] H. Zimmermann, "OSI reference model–the ISO model of architecture for open systems interconnection," *Communications, IEEE Transactions*, vol. 28, no. 4, pp. 425–432, Apr 1980.

[27] M. Perillo and W. Heinzelman, "DAPR: A protocol for wireless sensor networks utilizing an application-based routing cost," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC*. IEEE, 2004, pp. 1528–1533.

[28] A. Dunkels, F. Österlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2007, pp. 335–349.

[29] P. Buonadonna, J. Hill, and D. Culler, "Active message communication for tiny networked sensors," in *IEEE INFOCOM*, 2001.

[30] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," *SIGCOMM Computer Communications Review*, vol. 34, no. 1, pp. 69–74, 2004.

[31] E. Rozner, J. Seshadri, Y. A. Mehta, and L. Qiu, "SOAR: Simple opportunistic adaptive routing protocol for wireless mesh networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 12, pp. 1622–1635, 2009.

[32] M. Gerla, Y.-Z. Lee, J.-S. Park, and Y. Yi, "On demand multicast routing with unidirectional links," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 4, March 2005, pp. 2162–2167 Vol. 4.

[33] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," Feb 1999, pp. 90–100.

[34] M. Zorzi and R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance," *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, pp. 337–348, Oct.-Dec. 2003.

[35] M. Zorzi and R. R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Energy and latency performance," *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, pp. 349–365, 2003.

[36] S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-demand multicast routing protocol," in *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, 1999, pp. 1298–1302 vol.3.

[37] J. Garcia-Luna-Aceves and E. Madruga, "A multicast routing protocol for ad-hoc networks," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE*

Computer and Communications Societies. Proceedings. IEEE, vol. 2, Mar 1999, pp. 784–792 vol.2.

[38] R. Vaishampayan and J. Garcia-Luna-Aceves, “Efficient and robust multicast routing in mobile ad hoc networks,” in *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, Oct. 2004, pp. 304–313.

[39] E. M. Royer and C. E. Perkins, “Multicast operation of the ad-hoc on-demand distance vector routing protocol,” in *MobiCom ’99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1999, pp. 207–218.

[40] J. G. Jetcheva and D. B. Johnson, “Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks,” in *MobiHoc ’01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2001, pp. 33–44.

[41] C. Wu and Y. Tay, “AMRIS: a multicast protocol for ad hoc wireless networks,” in *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE*, vol. 1, 1999, pp. 25–29 vol.1.

[42] K. Chen and K. Nahrstedt, “Effective location-guided tree construction algorithms for small group multicast in manet,” *INFOCOM. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1180–1189, 2002.

[43] W. Wang, X.-Y. Li, and Y. Wang, “Truthful multicast routing in selfish wireless networks,” in *MobiCom ’04: Proceedings of the 10th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2004, pp. 245–259.

[44] A. Okura, T. Ihara, and A. Miura, “Bam: branch aggregation multicast for wireless sensor networks,” *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pp. 10 pp.–, Nov. 2005.

[45] C.-C. Chiang, M. Gerla, and L. Zhang, “Shared tree wireless network multicast,” in *Computer Communications and Networks, 1997. Proceedings., Sixth International Conference on*, Sep 1997, pp. 28–33.

[46] D. Koutsonikolas, S. Das, H. Charlie, and I. Stojmenovic, “Hierarchical geographic multicast routing for wireless sensor networks,” in *Sensor Technologies and Applications, 2007. SensorComm 2007. International Conference on*, Oct. 2007, pp. 347–354.

[47] S. Das, H. Pucha, and Y. Hu, “Distributed hashing for scalable multicast in wireless ad hoc networks,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 3, pp. 347–362, March 2008.

[48] S. Basagni, I. Chlamtac, and V. R. Syrotiuk, “Location aware, dependable multicast for mobile ad hoc networks,” *Computer Networks*, vol. 36, no. 5/6, pp. 659–670, 2001.

[49] M. M. andand Holger Fuler, J. Widmer, and T. Lang, “Position-based multicast routing for mobile ad-hoc networks,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 53–55, 2003.

[50] P. Bose, P. Morin, and J. Urrutia, “Routing with guaranteed delivery in ad hoc wireless networks,” in *Wireless Networks*, 1999, pp. 48–55.

[51] J. Sanchez, P. Ruiz, and I. Stojmnenovic, “GMR: Geographic multicast routing for wireless sensor networks,” in *Sensor and Ad Hoc Communications and Networks, 2006. SECON ’06. 2006 3rd Annual IEEE Communications Society on*, vol. 1, Sept. 2006, pp. 20–29.

[52] Y.-B. Ko and N. H. Vaidya, “Geocasting in mobile ad hoc networks: Location-based multicast algorithms,” *wmcsa*, vol. 0, p. 101, 1999.

[53] W. Ye, J. Heidemann, and D. Estrin, “Medium access control with coordinated adaptive sleeping for wireless sensor networks,” *Networking, IEEE/ACM Transactions on*, vol. 12, no. 3, pp. 493–506, June 2004.

[54] T. van Dam and K. Langendoen, “An adaptive energy-efcient MAC protocol for wireless sensor networks,” *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[55] K.-J. Wong and D. K. Arvind, “SpeckMAC: low-power decentralised MAC protocols for low data rate transmissions in specknets,” pp. 71–78, 2006.

[56] M. Buettner, G. V. Yee, E. Anderson, and R. Han, “X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks,” pp. 307–320, 2006. [Online]. Available: http://dx.doi.org/10.1145/1182807.1182838

[57] A. El-Hoiydi and J. Decotignie, “WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks,” *Proceedings of the First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, July 2004.

[58] T. Zheng, S. Radhakrishnan, and V. Sarangan, “PMAC: An adaptive energy-efficient MAC protocol for wireless sensor networks,” in *IPDPS ’05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS’05) - Workshop 12*. Washington, DC, USA: IEEE Computer Society, 2005, p. 237.1.

[59] R. Jurdak, P. Baldi, and C. Videira Lopes, “Adaptive low power listening for wireless sensor networks,” *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 988–1004, 2007.

[60] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, “MACAW: A media access protocol for wireless lans,” *ACM SIGCOMM*, September 1994.

[61] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, “A multi-radio unification protocol for IEEE 802.11 wireless networks,” oct. 2004, pp. 344 – 354.

[62] C. Doerr, M. Neufeld, J. Fifield, T. Weingart, D. Sicker, and D. Grunwald, “MultiMAC - an adaptive MAC framework for dynamic radio networking,” nov. 2005, pp. 548 –555.

[63] K.-C. Huang, X. Jing, and D. Raychaudhuri, “MAC protocol adaptation in cognitive radio networks: An experimental study,” aug. 2009, pp. 1 –6.

[64] J. Postel, “Internet Protocol,” RFC 791 (Standard), Sep. 1981, updated by RFC 1349. [Online]. Available: http://www.ietf.org/rfc/rfc791.txt

[65] J. Arkko and S. Bradner, “IANA Allocation Guidelines for the Protocol Field,” RFC 5237 (Best Current Practice), Feb. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5237.txt

[66] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” RFC 4944 (Proposed Standard), Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[67] “IP mobility support for IPv4, RFC3344,” United States, 2002.

[68] M. Kassar, B. Kervella, and G. Pujolle, “An overview of vertical handover decision strategies in heterogeneous wireless networks,” *Comput. Commun.*, vol. 31, no. 10, pp. 2607–2620, 2008.

[69] W. R. Stevens and G. R. Wright, *TCP/IP illustrated (vol. 2): The implementation*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[70] M. Sichitiu, “Cross-layer scheduling for power efficiency in wireless sensor networks,” *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1740–1750 vol.3, March.

[71] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, “A unifying link abstraction for wireless sensor networks,” pp. 76–89, 2005.

[72] R. M. Jurdak, “Modeling and optimization of ad hoc and sensor networks,” Ph.D. dissertation, Irvine, CA, USA, 2005, co-Chair-Cristina Videira Lopes and Co-Chair-Pierre Baldi.

[73] Sentilla Corporation, “http://www.sentilla.com/.” [Online]. Available: http://www.sentilla.com/

[74] OPNET, “http://www.opnet.com/.” [Online]. Available: http://www.opnet.com/

[75] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Standard 802.11, June 1999.

[76] "Single-chip 2.4 ghz IEEE 802.15.4 compliant and ZigBee(TM) ready RF transceiver - CC2420." [Online]. Available: http://focus.ti.com/docs/prod/folders/print/cc2420.html

[77] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 2–16, 2003.

[78] W. Si, S. Selvakennedy, and A. Y. Zomaya, "An overview of channel assignment methods for multi-radio multi-channel wireless mesh networks," *J. Parallel Distrib. Comput.*, vol. 70, no. 5, pp. 505–524, 2010.

[79] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2004, pp. 114–128.

[80] M. Wang, L. Ci, P. Zhan, and Y. Xu, "Multi-channel MAC protocols in wireless ad hoc and sensor networks," in *CCCM '08: Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 562–566.

[81] J. Mo, H.-S. W. So, and J. Walrand, "Comparison of multi-channel MAC protocols," in *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. New York, NY, USA: ACM, 2005, pp. 209–218.

[82] C. Doerr, M. Neufeld, J. Fifield, T. Weingart, D. Sicker, and D. Grunwald, "MultiMAC - an adaptive MAC framework for dynamic radio networking," *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 548 –555, nov. 2005.

[83] X. Qin and R. Berry, "Exploiting multiuser diversity for medium access control in wireless networks," in *INFOCOM 2003. Twenty-Second Annual Joint Con-*

ference of the IEEE Computer and Communications. IEEE Societies, vol. 2, 30 2003, pp. 1084 – 1094 vol.2.

[84] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer, “SHARP: a hybrid adaptive routing protocol for mobile ad hoc networks,” in *MobiHoc ’03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2003, pp. 303–314.

[85] “The Network Simulator NS-3,” http://www.nsnam.org/.

[86] C.-H. Feng, I. Demirkol, and W. B. Heinzelman, “UPS: Universal protocol stack for emerging wireless networks,” *Ad Hoc Networks*, Aug. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2011.07.013

[87] “Long Term Evolution Protocol Overview,” freescale semiconductor, Tech. Rep., 2008. [Online]. Available: http://www.freescale.com/

[88] Shweta Sinha, “A TCP tutorial.” [Online]. Available: http://www.ssfnet.org/Exchange/tcp/tcpTutorialNotes.html

[89] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[90] Y. Song, Y. Fang, and Y. Zhang, “Stochastic channel selection in cognitive radio networks,” *Global Telecommunications Conference, 2007. GLOBECOM ’07. IEEE*, pp. 4878 –4882, nov. 2007.

[91] V. Kanodia, A. Sabharwal, and E. Knightly, “MOAR: a multi-channel opportunistic auto-rate media access protocol for ad hoc networks,” *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on*, pp. 600 – 610, oct. 2004.

[92] A. Motamedi and A. Bahai, “MAC protocol design for spectrum-agile wireless networks: Stochastic control approach,” *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, pp. 448 –451, april 2007.

[93] P. Varaiya, J. Walrand, and C. Buyukkoc, “Extensions of the multiarmed bandit problem: The discounted case,” *Automatic Control, IEEE Transactions on*, vol. 30, no. 5, pp. 426 – 439, may 1985.

[94] S. M. Kay, *Fundamentals of statistical signal processing: estimation theory*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[95] K. Huber and S. Haykin, “Improved bayesian MIMO channel tracking for wireless communications: incorporating a dynamical model,” *Wireless Communications, IEEE Transactions on*, vol. 5, no. 9, pp. 2458 –2466, september 2006.

[96] B. Karp and H. T. Kung, “GPSR: greedy perimeter stateless routing for wireless networks,” in *MobiCom ’00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000, pp. 243–254.

[97] H. Chen and Y. Li, “Performance model of ieee 802.11 dcf with variable packet length,” *Communications Letters, IEEE*, vol. 8, no. 3, pp. 186–188, March 2004.

[98] C.-f. Hsin and M. Liu, “Hitting time analysis for a class of random packet forwarding schemes in ad hoc networks,” *Ad Hoc Netw.*, vol. 7, no. 3, pp. 500–513, 2009.

[99] D. W. Gage, “Many-Robot MCM search systems,” in *Proceedings of Autonomous Vehicles in Mine Countermeasures Symposium*, 1995, pp. 9–55.

[100] Q. Cao, T. He, and T. Abdelzaher, “uCast: Unified connectionless multicast for energy efficient content distribution in sensor networks,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, no. 2, pp. 240–250, Feb. 2007.

[101] P. Levis and N. Lee, “TOSSIM: A simulator for TinyOS networks,” Tech. Rep., 2003.

[102] B. Blum, T. He, S. Son, and J. Stankovic, “IGF: A state-free robust communication protocol for wireless sensor networks,” In: Proceedings of the 3rd IEEE Workshop on Applications and Services in Wireless Networks, Tech. Rep., 2003.

[103] Crossbow Technology, “http://www.xbow.com/.” [Online]. Available: http://www.xbow.com/

[104] I. Demirkol, C. Ersoy, F. Alagoz, and H. Delic, “The impact of a realistic packet traffic model on the performance of surveillance wireless sensor networks,” *Computer Networks*, vol. 53, no. 3, pp. 382 – 399, 2009.

[105] C. J. Merlin and W. B. Heinzelman, “Duty cycle control for low power listening MAC protocols,” in *In Proc. MASS08*, 2008.