UNIVERSITY OF CALIFORNIA
SANTA CRUZ

MEDIUM ACCESS CONTROL PROTOCOLS FOR MULTI-HOP WIRELESS AD HOC NETWORKS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Venkatesh Rajendran

December 2006

The Dissertation of Venkatesh Rajendran
is approved:

Professor Katia Obraczka, Chair

Professor J.J. Garcia-Luna-Aceves

Professor Mario Gerla

Dean Lisa C. Sloan
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Medium Access Control Protocols For Multi-hop Wireless Ad Hoc Networks

by

Venkatesh Rajendran

Wireless networking has become an increasingly active research area over the past decade. Recent advances in silicon manufacturing technology have enabled a steady increase in wireless communication capabilities yet still reducing form-factor. For instance, highly integrated, small foot-print, single-chip CMOS radios that can support very high data-rates (up to 480Mbps) are now commercially available [1] and the industry is moving towards multi-gigabit data rates using advanced coding/modulation techniques and new technologies [56, 65].

Such technological advances will enable new classes of applications for MANETs that require high application-level throughput and quality-of-service (QoS). Some examples include multimedia streaming, large content transfer, wireless storage area networks, etc. Consequently, the current trend is that new physical layer (PHY) technologies have been moving the fundamental limits challenging development and deployment of high data-rate, QoS-sensitive applications from the PHY to the medium access control (MAC) layer.

On the other hand, these advanced PHY techniques often result in higher energy requirements. For instance, the use of multiple RF chains and advanced decoding techniques to improve PHY performance increase power consumption of the receive operation significantly. To support better energy management, recent radios provide finer-grained power modes that selectively turns on portions of the radio transceiver. Radio power mode control at the MAC layer is thus critical in improving the battery life of the node. The state-of-the-art in medium access for MANETs is far from addressing the requirements imposed by upcoming applications and PHY technological advances.

Our research focus is to bridge this gap and develop a medium access scheduling framework that: (1) is application aware by adapting to changing traffic patterns and satisfying QoS requirements, (2) is self-organizing by adapting to current node state, and connectivity, (3) improves channel utilization and spatial re-use by multi-channel usage and collision-avoidance, and (4) is energy efficient.

The initial motivation for our research on MAC protocols is derived from our work on reliable multicast transport in MANETs. In this work, we introduce the Reliable Adaptive Congestion-controlled Transport protocol, or ReACT, that demonstrates the importance of optimizations at the MAC layer to improve the reliability at the transport layer. ReACT, combines source-based congestion- and error control with receiver-initiated localized recovery. While the latter attempts to recover localized losses (e.g., caused by transmission errors), the former is invoked only for losses and congestion that could not be recovered locally (e.g., caused by global congestion). Loss differentiation is an important component of ReACT and uses medium access control (MAC) layer information to distinguish between different types of losses. Through simulations, we evaluated ReACT's performance and compared it with RALM, a strictly source-based protocol. As our simulation results indicate, significant improvement in throughput and reliability could be achieved by using feedback from the MAC layer to differentiate congestion losses from other losses.

The Traffic-Adaptive Medium Access (TRAMA) protocol [42, 43] was the first proposal to implement energy-aware schedule-based medium access. TRAMA reduces energy consumption by ensuring that unicast and broadcast transmissions incur no collisions, and by allowing nodes to assume a low-power, idle state whenever they are not transmitting or receiving. TRAMA assumes that time is slotted and uses a distributed election scheme based on information about traffic at each node to determine which node can transmit at a particular time slot. Using traffic information, TRAMA avoids assigning time slots to nodes with no traffic to send, and also allows nodes to determine when they can switch off to idle mode and not listen to the channel. TRAMA is shown to be fair and correct, in that no idle node is an intended receiver and no receiver suffers collisions. An analytical model [43] to quantify the performance of TRAMA is presented and the results are verified by simulation. The performance of TRAMA is evaluated through extensive simulations using both synthetic- as well as sensor-network scenarios. The results indicate that TRAMA outperforms contention-based protocols (CSMA, 802.11 and S-MAC) and also static scheduled-access protocols (NAMA) with significant energy savings.

FLAMA [41] avoids explicit traffic information exchange and employs a much simpler election algorithm than TRAMA. FLAMA does not require explicit schedule announcements during scheduled access periods. Alternatively, application-specific traffic information is

exchanged among nodes during random access to reflect the driving application's specific traffic patterns, or *flows*. This allows FLAMA to still adapt to changes in traffic behavior and topology (e.g., node failure). FLAMA uses flow information to establish transmission schedules for each node. Additionally, FLAMA achieves traffic adaptiveness by assigning slots to a node depending on the amount of traffic generated by that node. This is accomplished by assigning *node weights* based on the incoming and outgoing flows. Nodes with more outgoing flows are given higher weights (i.e., more slots); the net effect is that nodes that produce/forward more traffic are assigned more slots.

FLAMA is simple enough so that it can be run by nodes with limited processing, memory, communication, and power capabilities. We evaluate the performance of FLAMA through simulations and test-bed experimentation. Simulation results indicate that, in terms of reliability, queuing delay and energy savings, FLAMA outperforms TRAMA, the first traffic-adaptive, schedule-based MAC proposed for sensor networks, and S-MAC, a contention-based energy-efficient MAC. FLAMA achieves significantly smaller delays (up to 75 times) when compared to TRAMA with significant improvement in energy savings and reliability, demonstrating the importance of application-awareness in medium access scheduling. Our simulation and test-bed results show that FLAMA achieves better end-to-end reliability with significant energy savings compared to S-MAC.

All previously mentioned protocols are designed to work with a single channel. Given that most commercially available radios to-date provide multiple orthogonal channels, protocols should make use of this feature to schedule parallel transmissions within a two-hop neighborhood, thus improving channel utilization. We introduce the Multi-Channel FLAMA (or mFLAMA), that extends the scheduling algorithm of FLAMA to support multiple channels. We compare the performance of mFLAMA with that of FLAMA by simulations, to illustrate the benefit in channel utilization when multiple channels are used for communication.

Finally, we present a new framework for energy-efficient channel access. One of the main features of the proposed framework, or DYNAMMA for DYNAmic Multi-channel Medium Access, is its ability to accommodate and adapt to different application traffic patterns in an efficient fashion, i.e., minimizing protocol overhead and delivery delay. This is an important contribution as it addresses a drawback inherent to scheduled-access MAC protocols. In DYNAMMA's current implementation, traffic adaptation is done by explicit traffic

announcements (in a considerably more efficient way, than existing scheduled-access protocols such as TRAMA [42]). Besides "explicit" adaptation, the flexibility provided by DYNAMMA's framework allows it to accommodate "implicit" traffic adaptation strategies, for example, using learning algorithms, which will further reduce protocol overhead.

We evaluate the performance of DYNAMMA by extensive simulations for different application scenarios. The results from our simulation study shows that DYNAMMA achieves significantly lesser queueing delay than TRAMA and provides high channel utilization and energy savings when compared to TRAMA and 802.11. We also present a generic MAC development test-bed for evaluating scheduled-access MAC protocols using UWB physical layer and we evaluate the performance of DYNAMMA using our FPGA-based test-bed.

In future work, DYNAMMA framework can be used for incorporating traffic prediction to establish the flow information. This can potentially reduce the queueing delay introduced due the scheduling. Another direction of work is to improve the transmission scheduling algorithm presented in the DYNAMMA framework to provide guarantied delivery delays.

To my parents, and friends.

Acknowledgments

I would like to convey my sincere and special thanks to my advisor Katia for guiding me throughout this work, and providing me constant encouragement and support both technically and personally. My thanks to JJ and Mario for being in my dissertation committee and providing valuable comments and suggestions. Thanks to Carol and Jodi for their support and advice throughout the program. My thanks to Wionics Research (Realtek) for providing valuable experiece with the wireless industry, supporting my studies, and providing the UWB radio for my research.

Thanks to my wonderful lab mates at INRG and CCRG, my friend Saro, my collegemates from India, and all my friends at UCSC. My thanks to Steve, Turi, Ravi, Ran, Ping-Wen, Sean and other collegueges at Wionics for providing a wonderful work environment in the later half of my study.

Last but not the least, my special thanks to my mom Rajalakshmi, dad Rajendran, sister Uma, brother-in-law Mahesh, wonderful nieces Mahima & Madhumita, aunt Shantha, uncle Jambunathan, and my entire family for all their help, support, and encouragement.

The entire work presented in this thesis is in collaboration with K. Obraczka, and J.J. Garcia-Luna-Aceves. In addition, I would like to acknowledge the collaboration M. Gerla, Y. Yi, K. Tang, and S.J. Lee in the work related to reliable multicasting, T. Aytur, R.-H, Yan, S. ten Brink, R. Mahadeveppa, and P.-W. Ong in the work related to UWB test-bed and UWB PHY simulation model, E.B.Decker in the work related to MFLAMA.

Chapter 1

Introduction

Multi-hop ad hoc wireless networks, or MANETs [38], do not rely on any fixed network infrastructure. They can then be deployed impromptu to provide networking services in areas where wireline networks do not exist or have been (permanently or temporarily) out of service. Consequently, MANETs have emerged as an ideal solution to a number of applications with significant scientific and societal relevance. Such applications include mission-critical applications such as emergency response, disaster recovery and rescue. Wireless sensor networks, a special case of MANETs, also find applications in environmental monitoring, surveillance, and tracking.

MANETs typically interconnect a collection of nodes that often have limited capabilities when compared to wireline network nodes. One main limitation is power as wireless devices are typically battery-powered; consequently, energy efficiency is critical to maximize the nodes' and the whole network's operational lifetime.

Besides power limitations, wireless nodes are usually limited in their communication capabilities. However, recent advances in silicon manufacturing technology have enabled a steady increase in wireless communication capabilities yet still reducing form-factor. For instance, highly integrated, small foot-print, single-chip CMOS radios that can support very high data-rates (up to 480Mbps) are now commercially available [1]. Figure 1.1 illustrates the data rate supported by different wireless technologies. As we can see the wireless industry is moving towards multi-gigabit data rates using advanced coding/modulation techniques and new technologies [56, 65].

Such technological advances will enable new classes of applications for MANETs

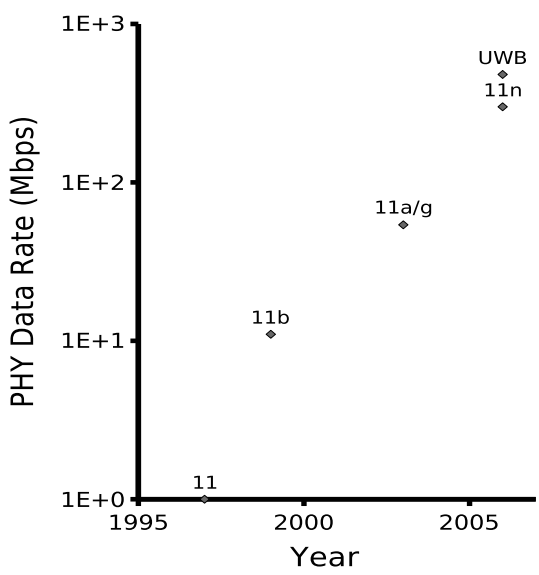


Figure 1.1: Wireless physical layer technologies and the supported data rates

that require high application-level throughput and quality-of-service (QoS). Some examples include multimedia streaming, large content transfer, wireless storage area networks, etc. Consequently, the current trend is that new physical layer (PHY) technologies have been moving the fundamental limits challenging development and deployment of high data-rate, QoS-sensitive applications from the PHY to the medium access control (MAC) layer.

On the other hand, these advanced PHY techniques often result in higher energy requirements. For instance, the use of multiple RF chains and advanced decoding techniques to improve PHY performance increase power consumption of the receive operation significantly. To support better energy management, recent radios provide finer-grained power modes that selectively turns on portions of the radio transceiver. Radio power mode control at the MAC layer is thus critical in improving the battery life of the node.

The state-of-the-art in medium access for MANETs is far from addressing the requirements imposed by upcoming applications and PHY technological advances.

Our research focus is to bridge this gap and develop a medium access scheduling framework that: (1) is application aware by adapting to changing traffic patterns and satisfying QoS requirements, (2) is self-organizing by adapting to current node state, and connectivity, (3) improves channel utilization and spatial re-use by multi-channel usage and collision-avoidance,

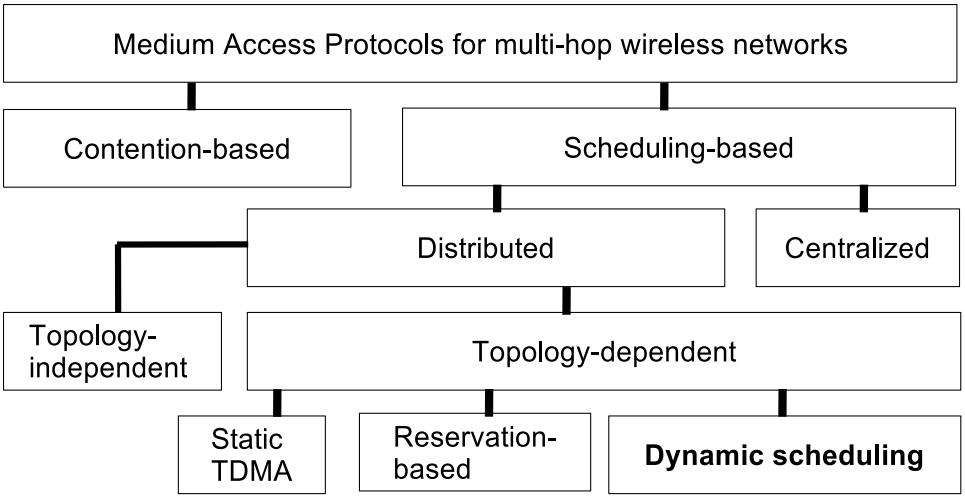


Figure 1.2: Medium Access Control protocols for multihop wireless networks

and (4) is energy efficient.

The following section presents a brief overview of various medium access approaches in MANETs and highlights the contribution of this work in energy-efficient, application-aware medium access.

1.1 MAC Overview

There is an extensive body of work on medium access control (MAC) protocols for multihop wireless networks, dating back to DARPA's packet radio program (e.g., [11, 13, 17, 28, 29, 31]). These MAC protocols can be broadly categorized as contention- and schedule-based as shown in Figure 1.2.

1.1.1 Contention-based Medium Access

Contention-based MACs have been widely used in wireless ad hoc networks due to their simplicity and ease of implementation. The best-known example of contention-based MAC protocol is the distributed coordination function (DCF) of the IEEE 802.11b standard [24]. 802.11's DCF uses the carrier sense multiple access (CSMA) technique combined with a four-way handshake that attempts to avoid collisions of data packets. However, the simplicity of the contention-based approach comes at the price of energy-efficiency and channel utilization.

The probability of collisions in a contention-based approach increases with the offered load. The upper bound on channel utilization (ρ) can be computed as follows: $\rho = t_p/(t_h + t_p + t_{ifs} + t_{bo} + t_{ctrl})$, where, t_p is the total time spent in transmitting useful payload, t_h is the total time spent in physical layer / MAC layer headers, t_{ifs} is the total time spent in inter-frame spacing(IFS), t_{bo} is the average time spent in back-off, t_{ctrl} is the total time spent in control packets (RTS / CTS / ACK).

As the PHY data rate increases, t_p decreases and to maintain high channel utilization, the amount of time spent in header, IFS, back-off, and control packet exchanges should be reduced. For contention-based MACs, t_{bo} and t_{ctrl} is much larger when compared to t_p and this contributes to a significant degradation in channel utilization [63, 66]. Some of the techniques used to improve the channel utilization include:

- Reduce the header and IFS overhead by using short preambles and burst mode transmissions with minimal inter-frame spacing (MIFS).
- Extend the amount of time spent in useful data transmissions with minimal control packet exchanges using extended transmit opportunities and block acknowledgements.

Even though these techniques can be used to reduce the overhead in contention-based MACs [22], the back-off periods can still lead to poor channel utilization. This was one of the main motivation behind migrating to schedule-based MACs, which eliminate the need for back-offs.

In traditional contention-based approaches, where there is no co-ordination between the nodes about the exact start time of the transmissions, nodes need to continuously listen to the medium to maintain active MAC layer connectivity. This introduces a key limitation that nodes consume energy needlessly when they are idle (i.e., not transmitting or receiving) as well as when collisions occur.

Some of the commonly employed techniques to improve the energy efficiency of a contention-based MAC protocol are as follows:

- Avoid overhearing using in-band / out-of-band signaling [47].
- Establish periodic listen / sleep cycles [36, 62, 67] and announce the availability of transmissions using special information elements [61, 64].

However, synchronized listen periods increases the channel contention significantly and also increases the overall noise floor during transmissions, leading to a degradation in the link quality. This motivated research into conflict-free mechanisms, in particular, scheduled-based medium access, where transmission schedules are established to allow nodes to receive data packets without collisions.

1.1.2 Scheduled-based Medium Access

Scheduling-based medium access protocols can be broadly categorized into topology-independent [9, 10, 26, 52] and topology-dependent [3, 17, 44] scheduling.

In a topology-independent scheduling approach, the transmission schedules are established such that each node is guarantied to have a conflict-free channel access to its receiver(s) within a finite number of medium access slots. Time-spread Multiple-Access (TSMA) protocol [10] is an example of topology-transparent scheduling approach where every node is assigned an unique binary code and the nodes behave deterministically on a time frame basis according to the code assignment. The number of nodes and the maximum node degree is used as a design parameter to establish the binary code and the corresponding channel access schedule for the time frame. However, nodes may collide in the channel access as they do not use the topology information. This reduces the channel utilization and energy efficiency of the topology-independent scheduling approaches [30].

In a topology-dependent scheduling approach, the node connectivity information is used to establish transmission schedules that are conflict-free in the time, frequency, code or space divisions of the channel. The problem of conflict-free transmission slot assignment can be compared to the graph coloring problem and is proven to be NP complete [12]. There are several proposals presenting maximal scheduling solutions using a centralized or distributed approach.

Centralized scheduling solutions require global topology information, which is prohibitive in a large network. Distributed approaches require a minimum of two-hop topology information for conflict-free scheduling and lack of complete topology information may affect the maximal property of the scheduling algorithm.

1.2 Research Focus

The focus of research on topology-dependent approaches has been (1) improving the channel utilization of the scheduling with limited topology information, and (2) reducing the communication overhead in exchanging topology information. However, the key limitations of wide-spread adaptation of transmission scheduling are:

- Degradation of channel utilization due to slot assignments to nodes that do not have any data to send.
- Scheduling ambiguities in determining the intended receivers leading to increased energy consumption due to idle listening.

This motivates our research on developing a medium access scheduling framework that: (1) is application aware by adapting to changing traffic patterns and satisfying QoS requirements, (2) is self-organizing by adapting to current node state, and connectivity, (3) improves channel utilization and spatial re-use by multi-channel usage and collision-avoidance, and (4) is energy efficient.

1.3 Contributions

In this section we summarize the main contributions of this work.

- **ReACT**:The initial motivation for our research on MAC protocols is derived from our work on reliable multicast transport protocols in MANETs. Reliable Adaptive Congestion-controlled Transport protocol, or ReACT, combines source-based congestion- and error control with receiver-initiated localized recovery. While the latter attempts to recover localized losses (e.g., caused by transmission errors), the former is invoked only for losses and congestion that could not be recovered locally (e.g., caused by global congestion). Loss differentiation is an important component of ReACT and uses medium access control (MAC) layer information to distinguish between different types of losses. Through simulations, we evaluated ReACT's performance and compared it with RALM, a strictly source-based protocol. As our simulation results indicate, significant improvement in throughput and reliability could be achieved by using feedback from the MAC layer to

differentiate congestion losses from other losses. This illustrates the importance of MAC layer support in providing desired quality of support at the application layer.

- **TRAMA**: The Traffic-Adaptive Medium Access (TRAMA) protocol [42,43] was the first proposal to implement energy-aware schedule-based medium access. TRAMA reduces energy consumption by ensuring that unicast and broadcast transmissions incur no collisions, and by allowing nodes to assume a low-power, idle state whenever they are not transmitting or receiving. TRAMA assumes that time is slotted and uses a distributed election scheme based on information about traffic at each node to determine which node can transmit at a particular time slot. Using traffic information, TRAMA avoids assigning time slots to nodes with no traffic to send, and also allows nodes to determine when they can switch off to idle mode and not listen to the channel. TRAMA is shown to be fair and correct, in that no idle node is an intended receiver and no receiver suffers collisions. An analytical model [43] to quantify the performance of TRAMA is presented and the results are verified by simulation. The performance of TRAMA is evaluated through extensive simulations using both synthetic- as well as sensor-network scenarios. The results indicate that TRAMA outperforms contention-based protocols (CSMA, 802.11 and S-MAC) and also static scheduled-access protocols (NAMA) with significant energy savings.

- **FLAMA & MFLAMA**: Flow-aware medium access protocol (FLAMA) illustrates the importance of the application-awareness in medium access protocols. FLAMA [41] avoids explicit traffic information exchange and employs a much simpler election algorithm than TRAMA. FLAMA does not require explicit schedule announcements during scheduled access periods. Alternatively, application-specific traffic information is exchanged among nodes during random access to reflect the driving application's specific traffic patterns, or *flows*. This allows FLAMA to still adapt to changes in traffic behavior and topology (e.g., node failure). FLAMA uses flow information to establish transmission schedules for each node. Additionally, FLAMA achieves traffic adaptiveness by assigning slots to a node depending on the amount of traffic generated by that node. This is accomplished by assigning *node weights* based on the incoming and outgoing flows. Nodes with more outgoing flows are given higher weights (i.e., more slots); the net effect is that nodes that produce/forward more traffic are assigned more slots.

FLAMA is simple enough so that it can be run by nodes with limited processing, memory, communication, and power capabilities. We evaluate the performance of FLAMA through simulations and test-bed experimentation. Simulation results indicate that, in terms of reliability, queuing delay and energy savings, FLAMA outperforms TRAMA, the first traffic-adaptive, schedule-based MAC proposed for sensor networks, and S-MAC, a contention-based energy-efficient MAC. FLAMA achieves significantly smaller delays (up to 75 times) when compared to TRAMA with significant improvement in energy savings and reliability, demonstrating the importance of application-awareness in medium access scheduling. Our simulation and test-bed results show that FLAMA achieves better end-to-end reliability with significant energy savings compared to S-MAC. We also present the multi-channel version of FLAMA that establishes collision-free transmission schedules across multiple channels.

- **DYNAMMA**: DYNAmic Multi-Channel Medium Access, an energy-efficient, scheduling-based multi-channel medium-access control (MAC) framework designed for multi-hop wireless ad hoc networks (MANETs). One of the main features of the proposed framework, is its ability to accommodate and adapt to different application traffic patterns in an efficient fashion, i.e., minimizing protocol overhead and delivery delay. This is an important contribution as it addresses a drawback inherent to scheduled-access MAC protocols. In DYNAMMA's current implementation, traffic adaptation is done by explicit traffic announcements (in a considerably more efficient way, than our previous approaches to medium access scheduling such as TRAMA [42]). Besides "explicit" adaptation, the flexibility provided by DYNAMMA's framework allows it to accommodate "implicit" traffic adaptation strategies, for example, using learning algorithms, which will further reduce protocol overhead.

  We evaluate the performance of DYNAMMA by extensive simulations for different application scenarios. The results from our simulation study shows that DYNAMMA achieves significantly lesser queueing delay than TRAMA and provides high channel utilization and energy savings when compared to TRAMA and 802.11.

- **MAC Development Test-bed**: A flexible MAC development platform using FPGA has been developed to evaluate the performance of scheduling-based MACs on the UWB

physical layer. The platform uses a readily available Xilinx embedded development board using PowerPC hardcore and custom lower MAC FPGA hardware providing precise radio mode control and transmission scheduling. The MAC platform uses the MAC-PHY interface to communicate with the UWB radio as the daughter card. We present a sample implementation of DYNAMMA on our MAC development platform and evaluate its performance.

1.4 Publications

- V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "DYNAMMA: DYNAmic Multi-channel Medium Access Protocol for Wireless Sensor Networks," Under Submission.

- V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," Wireless Networks, Vol. 12, pages 63-78, Feb 2006.

- V.Rajendran, J.J. Garcia-Luna-Aceves, and K.Obraczka, "Energy-Efficient, Application-Aware Medium Access for Sensor Networks," Proceedings of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), Washington D.C, November 7-10, 2005.

- V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," Proc. ACM SenSys 03, Los Angeles, California, 5-7 November 2003.

- V. Rajendran, K. Obraczka, Y. Yi, S.J. Lee, K. Tang and M. Gerla, "Combining Source- and Localized Recovery to Achieve Reliable Multicast in Multi-Hop Ad Hoc Networks," Proceedings of the Networking' 04, May 2004.

- V. Rajendran, Y. Yi, K. Obraczka, S.J.Lee, K.Tang and M. Gerla, "Reliable, Adaptive, Congestion-Controlled Adhoc Multicast Transport Protocol: Combining Source-based and Local Recovery," UCSC Technical Report, 2003.

- V. Rajendran, J.J. Garcia-Luna-Aceves, and K. Obraczka, "An Energy-Efficient Channel Access Scheduling for Sensor Networks," Proc. The Fifth International Symposium on Wireless Personal Multimedia Communication (WPMC), October 27–30, 2002, Honolulu, HI.

1.5 Organization

The remainder of the thesis is organized as follows. Chapter 2 motivates the important of MAC protocols in deciding the application-level quality of service in MANETs. We present a reliable multicast transport protocol that uses information from the MAC layer to perform loss differentiation. Chapter 3 presents the related work in energy-efficient medium access and presents our first attempt in energy-efficient channel access, DEANA, a simple dynamic energy-aware scheduling protocol for channel access in sensor networks and discuss potential gains in energy that could be achieved by using scheduled-based channel access protocols. Chapter 4 presents a more sophisticated scheduling algorithm that takes into account current traffic information in the MAC layer queue. Chapter 5 presents flow-aware scheduling protocol, a simple but effective mechanism to do traffic-aware scheduling in data gathering applications. Chapter 6 outlines the proposed framework for energy-efficient, multi-channel, channel access and also presents the MAC development test-bed using UWB radios. Chapter 7 concludes the thesis work and outlines the directions for future research on medium access protocols.

Chapter 2

Reliable Multicasting in MANETs

The initial motivation for our research on MAC protocols is derived from our work on reliable multicast transport in MANETs. In this chapter we demonstrate the importance of optimizations at the MAC layer to satisfy the application quality of service requirements. We present a novel reliable multicast transport protocol using cross-layer optimization for multi-hop, wireless ad hoc networks (or MANETs).

2.1 Reliable Multicast Transport

To recover from the different types of losses that may occur in MANETs, our Reliable Adaptive Congestion-controlled Transport protocol, or ReACT, combines source-based congestion- and error control with receiver-initiated localized recovery. While the latter attempts to recover localized losses (e.g., caused by transmission errors), the former is invoked only for losses and congestion that could not be recovered locally (e.g., caused by global congestion). Loss differentiation is an important component of ReACT and uses medium access control (MAC) layer information to distinguish between different types of losses.

Both unicast- and multicast routing in MANETs have been well-studied and, as a result, a number of protocols have been proposed [6, 34]. Several research efforts have also focused on transport-layer approaches to achieve end-to-end reliable point-to-point communication. This includes the work on improving TCP performance in "last-hop" wireless networks and MANETs [27, 35, 50, 51].

However, the types of scenarios targeted by MANETs make group-oriented services

such as data dissemination and teleconferencing a key application domain. In particular, the mission-critical characteristics of a number of these applications (e.g., emergency response, special civilian or military operations) call for efficient **reliable** multi-point communication protocols for MANETs. Undoubtedly, "network-supported" multicast communication is an efficient means of supporting group-oriented applications. This is especially true in MANETs where nodes are energy- and bandwidth limited. In these resource-constrained environments, reliable point-to-point protocols (that may be viable in wired networks) can get prohibitively expensive: the convergence of multiple requests to a single node typically causes intolerable congestion, violating the reliability and time constraints of a critical mission, and may drain the node's battery, cutting short the network's lifetime.

The Reliable Broadcast Protocol [37] addresses the problem of reliable atomic delivery of messages. While this protocol may work well in stable networks with low mobility and low failure rates, its performance will likely degrade in dynamic MANET scenarios where topology changes are frequent. Anonymous Gossip (AG) [7] recovers from losses by having pairs of multicast group participants exchange information on messages they have received or lost. AG uses solely local recovery from nearby members for error control. As expected, mainly due to the fact that AG does not implement congestion control, we observe that its performance deteriorates under heavy load.

Congestion-controlled Adaptive Lightweight Multicast (CALM) [55] is a multicast transport protocol that tries to achieve reliable delivery strictly through congestion control. This work demonstrates the importance of congestion control in improving reliability. The Reliable Adaptive Lightweight Multicast (RALM) protocol [53] uses a congestion control scheme similar to that of CALM and recovers from losses using source-based retransmissions. It requires multicast group member information to perform congestion control and error recovery. In an extended version of RALM [54], there is no need to maintain group membership information at the source. A combined source- and local recovery scheme purely based on number of packet losses is presented in [40]. The scheme does not account for the varying loss environment in wireless networks.

Several features unique to MANETs make the design of MANET reliable multicast transport mechanisms quite challenging. Among these features, we highlight: (1) MANET's heterogeneous loss characteristics due to factors such as mobility, node density, time-varying

channel conditions, (2) effects of lower layer protocols, e.g., inherent unfairness and unreliability of contention-based medium access control protocols (e.g., IEEE802.11 [24] uses plain CSMA when broadcasting packets and thus do not provide reliable broadcast delivery), and (3) MANET's extreme sensitivity to offered load.

These MANET features render design choices used in reliable multicast protocols for wired networks not at all applicable to MANET environments. Based on observations from the prior work [55], we argue that multicast reliability in MANETs cannot be achieved solely by retransmission of lost packets as is typically done in wired networks with protocols such as Scalable Reliable Multicast (SRM) [19]. Our premise is that, besides error control, effective reliable multicast delivery in MANETs must also perform congestion control. As demonstrated in previous studies [53–55], a simple congestion control scheme results in significant increase in delivery guarantees.

MANETs' complexity also calls for revisiting the layered system design argument which claims that, in a system, the design and implementation of each one of its layers should not be exposed to higher layers. We argue that in MANETs, information obtained from lower layers of the protocol stack is crucial for adequate performance at higher layers.

This motivated us to explore cross-layer mechanisms to achieve efficient reliable multicast transport. More specifically, we use information from lower layer protocols (in particular the MAC layer) to perform loss differentiation addressing MANETs' heterogeneous loss characteristics. Thus, some of the distinguishing features of ReACT are that (1) it combines source-based rate control with local error recovery and (2) uses loss differentiation to trigger either source-based control or local recovery. The goal is to recover from localized losses (e.g. due to node mobility, link quality, channel contention) using nearby group members, while congestion losses are reported to the source, triggering error- as well as congestion recovery.

Through extensive simulations, we evaluate ReACT's performance under a wide range of MANET conditions. In order to demonstrate the benefits of ReACT's loss differentiation and local recovery mechanisms, we also compare its performance against a strictly source-based control scheme (RALM [54]). In our experiments, as the underlying routing mechanism, we use a mesh-based multicast protocol, more specifically the On-Demand Multicast Routing Protocol (ODMRP) [33].

The remainder of this chapter is organized as follows. Section 2.2 presents a detailed

description of ReACT's source-based and local recovery mechanisms. Performance evaluation and simulation results follow in Section 2.3. Section 2.5 presents our concluding remarks and directions for future work in MAC layer multicasting.

2.2 ReACT

2.2.1 Overview

Our premise when designing ReACT is that in wireless environments losses may be caused by various factors and should be handled differently. For example losses caused by transmission errors (e.g., due to factors such as noise, interference, etc.) or hidden terminal collisions may be affecting only a small number of nodes in a neighborhood and thus can be recovered locally using a (non-congested) near-by member, i.e., without the involvement of the source. There is no need to trigger congestion control and slow down the source because these losses are not indicative of "global" congestion. Furthermore, by recovering locally, feedback and retransmissions are kept in the affected neighborhood and do not add to traffic destined to the source, hence improving protocol efficiency. On the other hand, congestion losses should be reported to the source triggering reduction of the sending rate as well as error recovery. However, special care should be taken as local recovery can exacerbate congestion if the network neighborhood performing recovery is already congested.

ReACT performs receiver-based loss differentiation to distinguish congestion- from local losses. A multicast receiver samples its MAC queue to detect congestion building up. Receivers also detect congestion building up anywhere on the path from the source by having intermediate nodes set a "congestion" flag in multicast data packets they forward. The congestion flag is set by any intermediate node whose MAC queue grows beyond a certain fraction of the maximum MAC queue size. By detecting incipient congestion (instead of waiting to take action until actual packet drops occur), ReACT tries to avoid persistent congestion conditions.

ReACT ensures that only multicast members that are situated in a non-congested area will be used to perform local recovery. This avoids contributing to congestion in an already congested neighborhood. The remainder of this section describes ReACT in detail by presenting its two main components, namely source-based (error and congestion) control and receiver-based error recovery.

2.2.2 Source-Based Control

ReACT employs a rate-based congestion control scheme that has two main modes of operation: initial rate set-up (i.e., determining the initial sending rate) [1], and congestion control. ReACT tries to determine the appropriate sending rate in order to avoid (1) initial bandwidth under utilization by starting too low, and (2) congestion by starting at too high of a rate.

One approach at setting up the initial rate is to probe the entire network and then decide on the rate based on the aggregate network condition. Though this approach can provide information on the overall state of the network, it is not scalable. Alternatively, in ReACT we establish the initial rate based on the set of members that are directly connected to the source. This provides the source with an estimate of its neighborhood's current conditions. The rate is decided so as to satisfy the worst receiver in this neighborhood.

The first data packet sent by the multicast source serves as a probe packet and each directly connected member replies with a *PROBE_REPLY* packet.After sending the first packet, the source waits for *PROBE_WAIT_TIME* to receive replies to its probe. *PROBE_WAIT_TIME* is set based on the network diameter (*NET_DIAMETER*) and an estimate of average time to traverse one hop (*NODE_TRAVERSAL_TIME*) accounting for queuing and transmission delays (similar to the route reply timeout of AODV [39]). If the source does not hear from any receiver in response to the probe packet, it will continue to send (probe) packets every *PROBE_WAIT_TIME* interval. The source then computes the inverse of the largest round-trip time reported during the initial probing period and uses that as its initial sending rate.

The rate is periodically updated once every *PROBE_INTERVAL* by directly connected neighbors. If no feedback has been sent to the source for the last *PROBE_INTERVAL* seconds, the receiver generates an explicit *PROBE_REPLY* packet. Receivers only send an update to the source if they detect significant changes to the time it takes them to get packets from the source. *PROBE_INTERVAL* is set sufficiently large to prevent oscillations in the source sending rate and also to reduce feedback overhead due probing. The source continues to send at this rate, until it hears a negative acknowledgment (NACK) from any receiver experiencing congestion. In that case, it reverts to congestion control.

[1] In our previous approaches [53–55], we start at the application sending rate and then react to congestion based on receiver feedback. Our experiments indicate that setting an initial rate too high may lead to extreme (sometimes unrecoverable) congestion and thus numerous packet losses (e.g., if the feedback path from the receivers to the source gets blocked.)

ReACT's congestion control works as follows. The source initially multicasts data packets at the rate decided using initial probing as described above. Upon reception of a NACK, the source adds the NACK sender to its *Receiver List* and enters loss recovery. The missing sequence numbers reported by the NACK are added to a global retransmission list, which is an aggregate of lost sequence numbers from all reporting receivers. This list is updated whenever the source retransmits a packet to prevent duplicate retransmissions. In addition, the source keeps track of the end-to-end latency between itself and each receiver that sent NACKs.

The source initiates loss recovery by selecting a receiver from the *Receiver List*, which we call *Feedback Receiver*. The source then retransmits a lost packet requested by the *Feedback Receiver* or multicasts a new packet (e.g., if all lost packets requested by that receiver had already been retransmitted). The packet header includes information instructing the *Feedback Receiver* to reply via "unicast" with a (positive) acknowledgment (ACK) indicating that all packets have been successfully received or specifying the sequence number(s) of packets that are still missing. All other receivers process the packet without replying to the source.

The source then responds by retransmitting the requested packets one at a time until the *Feedback Receiver* receives all packets (i.e., send-and-wait). The design philosophy behind retransmitting one packet at a time is to slow down the source when congestion is detected. Since only the *Feedback Receiver* replies to the source, the ACK/NACK implosion problem is avoided. NACKs are rate-limited to prevent excessive feedback overhead. The mechanism for controlling NACK generation is described in Section 2.2.3.

Once the *Feedback Receiver* obtains all packets, it unicasts an ACK to the source indicating successful reception of all packets. Upon reception of the ACK, the source removes the node from the *Receiver List*, chooses a new *Feedback Receiver* in a round robin fashion, and repeats this process until the *Receiver List* is empty.

When the *Receiver List* is empty, the source reverts to the latest sending rate decided based on periodic probe packets. If, however, the source does not receive a NACK or ACK from the *Feedback Receiver* within the time interval given by the measured round-trip time from the source to the *Feedback Receiver*, the source backs off and tries again up to a maximum number of times (which is three in our simulations) before removing it from the *Receiver List*. The removed receiver may later re-synchronize with the source through the normal NACK mechanism.

The round-robin send-and-wait approach does not require retransmissions of the same lost packets multiple times to each receiver. In the best-case scenario, lost packets are retransmitted only once by the source since retransmissions are multicast. For instance, if a set of receivers lost the same packet, it is retransmitted only once assuming the retransmitted packet is received by all the receivers. In the worse-case scenario, each receiver Gexperiences different packet losses. In this case, all lost packets must be retransmitted to each receiver.

2.2.3 Receiver-Based Error Recovery

The main goal of ReACT's receiver-based recovery mechanism is to detect losses that can be recovered locally avoiding source involvement (and hence avoid triggering congestion control). Congestion losses, however, should be reported to the source so that it knows to slow down.

In order to recover from losses "locally", nodes must obtain information about other group members as potential *Recovery Nodes*. Our scheme gathers member information using multicast data packets as they get forwarded over the multicast tree or mesh. Hence it is independent of the underlying multicast routing protocol. We are only interested in *Recovery Nodes* that are in the forwarding path from the source. More specifically, ReACT only uses immediate upstream member node(s) for recovery.

The way recovery requests and replies (or retransmissions) are routed has significant impact on the overall performance of the reliable multicast mechanism. If the underlying unicast routing protocol does not have a valid path for the recovery request and performs flooding for route discovery, significant additional load may result. Our simulation study indicates that local recovery based protocols that do not address this problem (e.g., AG) suffer from congestion even at moderate loads. ReACT employs a source routing approach that makes use of valid cached paths. The main advantage of this approach is that it makes ReACT independent of the underlying unicast routing protocol. The tradeoff is the overhead involved in maintaining source routes, especially in highly mobile environments. ReACT restricts the maximum distance between a member and its *Recovery Node* to LR_ROUTE_LEN hops to reduce the failure probability (e.g., due to node mobility) of source route.

Every node maintains a *Member Table* that stores information about current *Recovery Nodes*. To account for route volatility, *Member Table* entries are assigned an expiration time

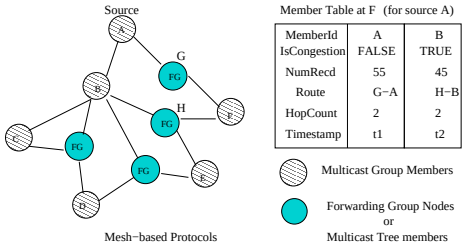


Figure 2.1: Member table maintained for local recovery

(*LR_VALID_TIME*). Additionally, each node maintains a metric of *reliability*, i.e., the rate at which it receives multicast data packets from *Recovery Nodes*. This information is used in selecting a *Recovery Node* if multiple ones exist. Each entry also has a flag to indicate if the path to the *Recovery Node* is congested. This flag is set if any intermediate node on the path to the *Recovery Node* has MAC queue size beyond the *CONGESTION_THRESHOLD*.

The IP option fields in the multicast data packet is used to carry route, hop count and congestion information. These fields are updated as the packet is forwarded to the group. The *route* field contains the path traversed by the packet from the upstream multicast group member. The *hopCount* field carries the length of the path. The *isCongestion* field denotes if any of the node in the path is congested. Whenever a node decides to perform local recovery, it selects a non-congested member that has the highest receive rate, lowest hop-count, and latest timestamp. Figure 2.1 shows a sample *Member Table* maintained by member node *F* when using either a tree- or mesh-based protocol. Mesh-based protocols may yield more than one upstream member because of path redundancy. As tree-based protocols also use broadcast for delivery, it is possible that a receiver might receive a packet from a node other than its parent node. Selecting an upstream *Recovery Node* based on its reliability and proximity increases the likelihood of successful local packet recovery.

Feedback generation is rate-limited to once every *MIN_FEEDBACK_INTERVAL* seconds to prevent excessive feedback overhead. Thus, every *MIN_FEEDBACK_INTERVAL*, receivers check if they need to perform error recovery by sending a NACK to a near-by member. There is a tradeoff in setting *MIN_FEEDBACK_INTEVAL*. When smaller intervals are used, we observe higher packet delivery ratios at the expense of higher overhead and lower throughput.

A NACK packet consists of an request array that is filled with the node's missing sequence numbers. The NACK is then sent to the selected *Recovery Node* if losses are found to

Table 2.1: Simulation parameters

ReACT Parameters	**Value**
LR_ROUTE_LEN	3
LR_VALID_TIME	3 s
MIN_FEEDBACK_INTERVAL	5 s
NODE_TRAVERSAL_TIME	50 ms
NET_DIAMETER	35
PROBE_INTERVAL	50 s

be localized. Nodes use cached source routes to communicate with the *Recovery Node*. On the other hand, if losses are due to "global congestion" or if the node finds that it is experiencing congestion, then it sends the NACK to the source using the underlying unicast routing protocol.

A node checks if losses are due to "global congestion" by examining the paths to *Recovery Nodes*. If all paths are congested, then all valid *Recovery Nodes* in the *Member Table* will have the *isCongestion* flag set. Additionally, the node also examines its queue to check if it is congested. If any of the above conditions is true, then losses are classified as due to "global" congestion and feedback is sent to the source directly triggering congestion control.

Besides missing sequence numbers, a NACK packet destined to the source also includes the average delay multicast packets take to reach the node from the source. Receivers update the average delay to a multicast source every time they receive a data packet from that source. The average delay is computed as an exponential average with more weight to recent measurements. Receivers sending NACKs to the source are placed in the *Receiver List*. The source then transmits to each *Receiver List* receiver based on its reported delay using a send-and-wait approach as described in Section 2.2.2.

2.3 Experimental Setup

As our simulation platform, we use the QualNet network simulator [60]. ODMRP [32] and AODV [39] are used as the underlying multicast and unicast routing protocols, respectively. The transmission range for the radio is $447.807m$ with a data rate of $2Mbps$. The MAC protocol used is IEEE 802.11 DCF [24].

We evaluate the performance of ReACT in comparison with plain Reliable Adaptive Lightweight Multicast (RALM) [54], a strictly source-based control scheme. We evaluate

ReACT's performance subject to a wide range of network conditions. We are particularly interested in how ReACT performs under various offered loads, and what is the impact of node mobility. Table 2.1 shows the values of the parameters used by ReACT.

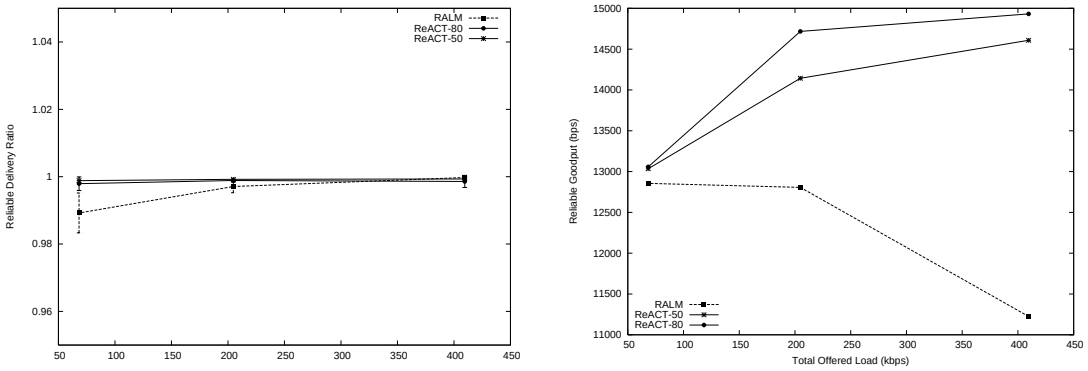
As we target applications that require the highest possible delivery guarantees, protocol reliability is a critical performance metric. We define *Reliable Delivery Ratio* as the fraction of packets successfully (or reliably) delivered to ALL receivers over the total number of packets sent. We also measure *Reliable Goodput* defined as the throughput of packets reliably delivered, i.e., packets that are received by all members. Finally, we measure the overhead incurred by the protocols. To account for control packets sent by underlying unicast/multicast protocols, we measure the total number of packets sent by each node at the MAC layer. *Normalized Overhead* is thus computed as the ratio of total packets sent at the MAC layer to total data packets delivered to all members. This measures the total number of packets transmitted to successfully deliver one data packet to all members.

First, we study the importance of congestion control by simulating a scenario with multiple sources generating different traffic loads and then we analyze the impact of node mobility. For these sets of experiments, 50 nodes are placed randomly in a $1500m \times 1500m$ field and 10 randomly chosen nodes join the multicast group. These group nodes join at the start of the simulation and stay subscribed to the group till the end of the simulation. Five randomly selected members continuously send CBR traffic throughout the whole duration of the simulation with payload size of 512 bytes. The results are averaged over several runs and presented with 95% confidence.

2.4 Simulation Results

2.4.1 Effect of Congestion

Figure 2.2(a) shows reliable delivery ratio under different loads. The error bars correspond to reliable delivery ratio's 95% confidence intervals. As the load increases, so does packet loss due to congestion and hidden terminal collisions. Both RALM and ReACT perform error recovery and congestion control and hence they achieve very high reliability. RALM employs strictly source-based error recovery using NACKs. NACKs are generated when lost sequence numbers are detected upon arrival of a new data packet. Hence, if a node stops receiving data



(a) Reliable delivery ratio

(b) Average reliable goodput

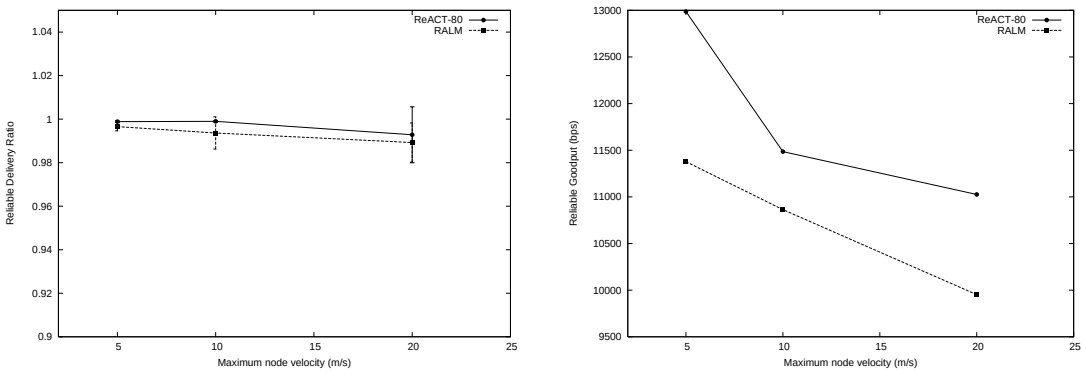
Figure 2.2: Effect of congestion

from a particular source, it will never generate NACK to recover lost packets. This leads to the reduced reliability of RALM at low loads and also higher reliability variance when compared to ReACT. On the other hand, ReACT achieves perfect reliability under various loads due to its robust error recovery mechanism.

We show results for two different versions of ReACT: one that detects congestion when queues grow above 80% of their maximum size, while the other version uses 50% as the queue threshold indicating congestion. As observed from Figure 2.2(a), both versions deliver almost perfect reliability.

Figure 2.2(b) illustrates the impact of ReACT's combined local- and source-based recovery mechanisms on goodput. We observe that ReACT achieves considerably higher (reliable) goodput when compared with RALM. Furthermore, ReACT is able to keep its goodput steady even at higher loads, while RALM suffers severe degradation at higher traffic rates. This is mainly because local recovery prevents the source from backing off its rate when packet losses are recovered locally. It should also be noted that RALM starts sending at the application rate and then performs congestion control when it receives feedback from the receivers. This aggressive behavior can potentially lead to severe congestion preventing NACKs from receivers to reach the source. This is one of the reasons for RALM's reliable goodput degradation as we increase the load. On the other hand, ReACT's initial setting of the sending rate also contributes to its high reliable goodput.

Figure 2.2(b) also illustrates the effect of *CONGESTION_THRESHOLD*, which is set



(a) Reliable delivery ratio

(b) Average reliable goodput

Figure 2.3: Effect of mobility

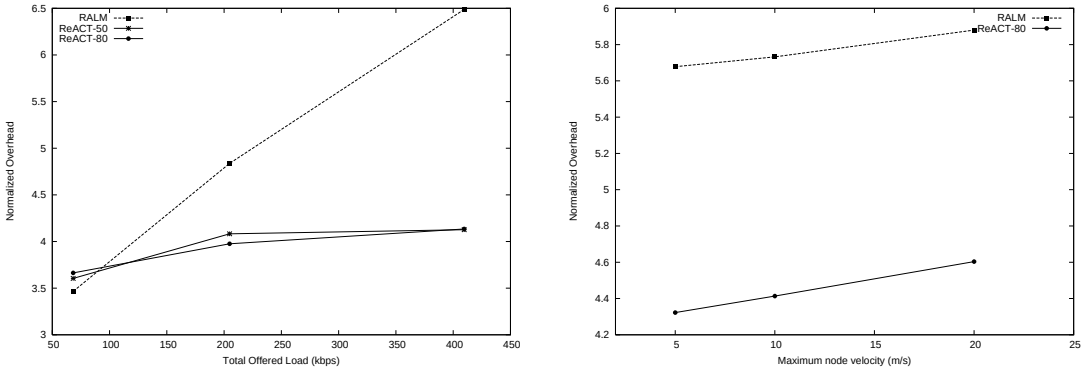
to 80% and 50% of the maximum MAC queue size. As expected, goodput is lower at 50% as ReACT becomes more conservative, generates feedback sooner and thus causes more frequent rate decreases.

Normalized overhead for RALM and ReACT are depicted in Figure 2.4(a). ReACT incurs significantly lesser overhead than RALM due to its local recovery mechanism at higher loads. It prevents unnecessary source retransmissions (which are multicast) for errors that are recoverable locally. Source route caching used by local recovery also helps to reduce the overhead incurred by local recovery. Otherwise, route discovery flooding by the underlying unicast routing protocols can significantly increase the total overhead. However, at low loads, probe replies sent by receivers for updating the source sending rate and the corresponding route discovery initiated by AODV slightly increases the overall normalized overhead.

2.4.2 Effect of Node Mobility

In these experiments, we use the random-way-point mobility model with no pause time and $0m/s$ minimum speed. We vary maximum speed from $5m/s$ to $20m/s$. The total network load injected in these mobile scenarios is $200Kbps$.

Figure 2.3(a) shows the reliable delivery ratio achieved by RALM and ReACT for different node velocities. Both RALM and ReACT are able to achieve perfect reliability even at high mobility. As previously discussed, the slight variation in RALM's reliable delivery ratio is due to the NACK generation mechanism driven by received data. As expected, Figure 2.3(b)



(a) Effect of congestion

(b) Effect of mobility

Figure 2.4: Normalized overhead

shows that both protocols exhibit degradation in goodput as we increase node mobility. In ReACT, the sending rate is updated based on the measured delay reported by the probed set of receivers. As we increase node mobility, the delay experienced by nodes becomes highly variable. ReACT uses the highest delay reported to update the sending rate, which can substantially reduce throughput. Thus, probing more frequently can improve the goodput with increased node mobility, at the expense of increased overhead due to probe replies. ReACT's local recovery mechanism is able to recover locally some mobility-induced losses, and thus achieves higher goodput than RALM.

As shown in Figure 2.4(b), ReACT's overhead is significantly lesser than RALM for all mobility conditions. This is mainly due to ReACT's local recovery mechanism which recovers mobility losses locally. As we increase mobility, ReACT's overhead increases as its local recovery effectiveness decreases: mobility leads to frequent timeouts of source routes maintained in the *Member Table*. This invalidates potential *Recovery Nodes* and forces receivers to resort to the source for error recovery. As previously discussed, this effect also contributes to the reduction in goodput with increased mobility. In our future work, we plan to overcome this problem by expanding cross-layer interaction and using more information from the lower layers (e.g., unicast routing).

2.5 Conclusion

In this chapter, we presented ReACT, an adaptive, congestion-controlled multicast transport protocol for reliable and timely multicast delivery in MANETs. One of ReACT's main distinguishing feature is its combination of source-based control and local recovery. ReACT's source-based control includes initial rate setup and congestion recovery which adjusts the sending rate using a simple stop-and-wait mechanism based on receivers' feedback.

Through simulations, we evaluated ReACT's performance and compared it with RALM, a strictly source-based protocol. Our results show that ReACT significantly improves both goodput and packet delivery with lower overhead. By way of its congestion control mechanism, ReACT is able to deliver perfect reliability under a wide range of conditions. We also demonstrate the benefits of ReACT's local recovery mechanism which prevents the source from reducing its rate unnecessarily and restricts the scope of receiver feedback yielding reduced protocol overhead.

As our simulation results indicate, significant improvement in throughput and reliability could be achieved by using feedback from the MAC layer to differentiate congestion losses from other losses. This illustrates the importance of MAC layer support in providing desired quality of support at the application layer. This also motivates our research on MAC layer, that provides reliable conflict-free delivery of multicast and broadcast frames.

Chapter 3

Dynamic Energy-aware Channel Access Protocol

In this chapter, we present a brief overview of the related work in medium access. We then introduce the Distributed Energy-Aware Node Activation (DEANA) channel access protocol for power-constrained networks, our first attempt in evaluating the benefit of energy-efficient channel access in MANETs.

Most channel access protocols (either contention-based or scheduling-based) are not power-aware, i.e., they provide no explicit mechanisms to achieve energy efficiency. The main source of energy consumption is idle listening, in which a node is in receive mode[1] even though it is not scheduled to receive or transmit any data. In addition, at high offered load, contention-based channel access protocols spend energy unnecessarily when collisions occur.

DEANA (Distributed Energy-Aware Node Activation) is a TDMA-based MAC protocol that adapts the Neighborhood-aware Contention Resolution (NCR), and node activation approach used in the Node Activation Multiple Access (NAMA) protocol [3]. The novel feature introduced by DEANA is that whenever a node is not scheduled for transmission or reception, the node's radio is switched to a low-power mode. It addresses the energy consumption caused by collisions and idle listening: (1) using the Neighborhood-aware Contention Resolution (NCR) mechanism [3] for collision-free channel access, and (2) selecting the state of the radio transceiver based on three different node states, namely transmitter, receiver, or idle in

[1] It has been shown that the energy spent in receive mode is 50-100% of energy spent in transmit mode for standard IEEE802.11 radios [49]

order to conserve energy. Since sensor networks are usually static, a simple neighbor discovery protocol could be used to distribute the two-hop neighbor information needed for the contention resolution protocol.

3.1 Related Work

PAMAS [47] is one of the earliest contention-based proposals to address power efficiency in channel access. PAMAS saves energy by attempting to avoid over-hearing among neighboring nodes. To achieve this, PAMAS uses out-of channel signaling. Woo and Culler [64] address variations of CSMA tailored for sensor networks, and propose an adaptive rate control mechanism to achieve fair bandwidth allocation among sensor network nodes. In the power save (PS) mode in IEEE 802.11 DCF, nodes sleep periodically. Tseng *et al.* [61] investigated three sleep modalities in 802.11 DCF in multi-hop networks. The sensor-MAC protocol [67], or S-MAC, exhibits similar functionality to that of PAMAS and the protocol by Tseng *et al.*. Like the other approaches, S-MAC avoids overhearing and nodes periodically sleep. However, unlike PAMAS, S-MAC uses in-line signaling, and unlike modalities of the PS mode in 802.11 DCF, neighboring nodes can synchronize their sleep schedules. T-MAC [62] is an improvement over S-MAC that adapts the duty cycle based on traffic. However, synchronized listen periods increase channel contention significantly and also increases the overall noise floor during transmissions leading to degradation in link quality.

D-MAC [36] is a contention-based medium access protocol optimized for data gathering applications over unidirectional trees. It schedules transmissions at each hop so that the latency in data collection is reduced. However, D-MAC assumes fixed topology and does not allow multiple data gathering trees. It cannot adapt to other sensor network applications. All of the above mentioned protocols improve energy efficiency by avoiding idle listening. However, they waste energy in (1) collisions due to hidden terminals and (2) carrier-sensing.

The WiMedia MAC targets UWB-based PHY [56] by defining a distributed, time-slotted medium access mechanism [56]. All nodes transmit beacons periodically and the medium access scheme is based on distributed reservations. Applications that require guaranteed service rates can take advantage of the reservation-based structure. However, static reservation-based approaches are not suited to applications with variable service rate. Reservation-based approaches may also lead to fairness problems and increased overhead in creating and maintaining

reservations.

All previously mentioned protocols are designed to work with a single channel. Given that most commercially available radios to-date provide multiple orthogonal channels, protocols should make use of this feature to schedule parallel transmissions within a two-hop neighborhood, thus improving channel utilization.

The work by So and Vaidya describes a multi-channel MAC for ad hoc networks (MMAC) using a single transceiver [48]. It is a contention-based medium access protocol similar to IEEE 802.11 and it uses the ATIM window in IEEE 802.11 PSM for announcing channel switching information. In MMAC, every node must listen in a default channel during the ATIM window. Nodes negotiate channels to transmit or receive by exchanging Preferred Channel Lists (PCLs). Another recent example of a multi-channel MAC is the Slotted Seeded Channel Hopping (SSCH) [2] protocol. SSCH is an improvement over SEEDEX [46] for scheduling across multiple channels. However, both approaches do not consider energy efficiency.

3.2 Dynamic Energy-aware Node Activation

We assume there is only a single channel for communications (both data and signaling) and the channel is time-slotted. DEANA adopts the neighborhood-aware contention resolution (NCR) node activation scheme [3] used by NAMA. The time division structure shown in Figure 3.1 is the basic (energy-unaware) node activation scheme derived from NAMA. Data packet transmission happens during a *transmission slot*. Time slots are grouped into sections and sections are grouped into blocks. In each block, the last section is reserved for updating node neighborhood information. NAMA's distributed node selection scheme guarantees collision freedom based on two-hop neighborhood information at each node. When a node is activated for transmission in a particular time slot, it requires all its neighbors to be in receive mode and thus can communicate with any one of them during that slot without conflict. Note that only for broadcast communication, a transmitter needs all its neighbors to be in receiver mode. Thus, when multicasting or (especially) unicasting data, considerable energy is wasted due to idle listening.

DEANA aims at reducing energy consumption in nodes that are not the intended receivers in a particular time slot. We divide NAMA's transmission slot into a control and data portion. During the control slot, the node activated by NAMA transmits a control packet

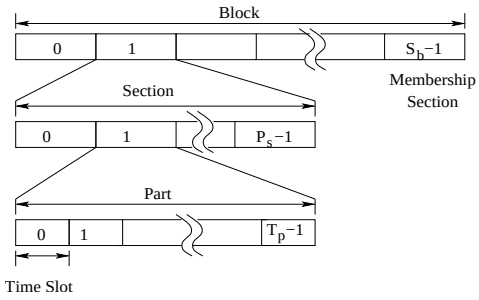


Figure 3.1: Time division structure for NAMA

with the identity of the intended receiver(s) and the actual data is transmitted during the data slot. Hence, all the one-hop neighbors of the selected node must be in receive mode during the control slot. During the data slot, only the intended receiver(s) need to be in receive mode while all other neighbors could be switched to low-power, standby mode. The modified slot structure is shown in Figure 3.2. Time is divided into cycles of scheduled access and random access. The scheduled access period is used for data transmission using the scheme described above and the random access period is used for transmission of small packets of signaling information containing neighbor information. We also allow some guard period for switching the radio in between transmission slots. The length of the scheduled and random access periods depend on the mobility of the network and the size of the signaling frame. In the case of sensor networks, there is very little or no mobility depending on the type of application. Hence, the main function of the random access period is to permit node additions (deployment of new nodes) or deletions (failure of nodes). Time synchronization could be done during this period. During the random access period, all participating nodes in the sensor processing task must be in either transmit or receive mode. Hence, the duration of the random access also plays a significant role in energy consumption. Slot synchronization and energy efficiency during random access are directions for future work.

Our energy conservation heuristics consists of the following rules:

- If the winner of the current transmission slot is in a node *A*'s one-hop neighborhood, then set node *A* to receive mode during the control slot. If node *A* is selected as a receiver during the control slot, then keep node *A* in receive mode during the data slot; else switch node *A* to standby mode.

- If the winner of the current transmission slot is in node A's two-hop neighborhood, then set node A to sleep mode for the entire transmission slot.
- If node A is the winner of the current transmission slot and has packets to send, then set node A to transmit mode, inform the intended receiver(s) during the control slot, and transmit the data packet(s) during the data slot.
- If node A is the winner of the current transmission slot and does not have a packet for transmission, set node A to standby mode for the entire transmission period.

3.3 Energy Model

In this section we present the mathematical model quantifying DEANA's energy consumption. As performance baseline, we also derive the expression for energy consumption of a protocol with no standby mode. We use *the expected energy consumption of a node during one transmission period* as performance metric.

The notation used in the remainder of the chapter are summarized in Table 3.1.

Table 3.1: DEANA Notations

N_2	: Node's number of two-hop neighbors
N_1	: Node's number of one-hop neighbors
q	: Channel access probability
p	: Probability that the selected node has a packet to transmit
p_{me}	: Probability that a node is selected as receiver by its one-hop neighbor
T_c	: Length of the control slot in seconds
T_d	: Length of the data slot in seconds
T_t	: Length of the transition period between transmission slots
P_{tx}	: Average power consumption in transmit mode
P_{rx}	: Average power consumption in receive mode
P_{st}	: Average power consumption in standby mode
$P_{x,y}$	: Average power consumption in transition from mode x to mode y
E_c	: Average power consumption during the control slot
E_d	: Average power consumption during the data slot
E_t	: Average power consumption during the transmission slot

The channel access probability q is the probability that a node is selected as a winner for a time slot. We assume that q is independent across nodes and is a function of the number

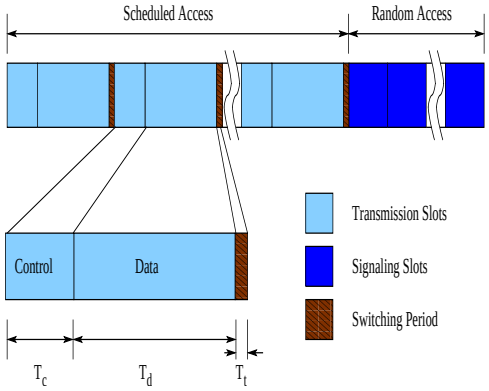


Figure 3.2: Time division structure for DEANA

of two-hop neighbors (N_2) of a node. We also assume that all the nodes have the same number of two-hop neighbors.

The power consumed when switching from a state to another is set to the higher value of power consumption when the transition is from the lower- to the higher power mode and is set to the average value when the transition is from the higher- to the lower power mode. Hence, the power consumption while in transition state is given by the following equations:

$$P_{tx,st} = (P_{tx} + P_{st})/2 \tag{3.1}$$

$$P_{st,tx} = P_{tx} \tag{3.2}$$

$$P_{rx,st} = (P_{rx} + P_{st})/2 \tag{3.3}$$

$$P_{st,rx} = P_{rx} \tag{3.4}$$

$$P_{tx,rx} = (P_{tx} + P_{rx})/2 \tag{3.5}$$

$$P_{rx,tx} = P_{tx} \tag{3.6}$$

The probability that a node is selected as the receiver by its one-hop neighbor is given by:

$$\begin{aligned} p_{me} &= \sum_{n \in N_2} q.(1/N_1).(N_1/N_2) \\ &= q \end{aligned} \tag{3.7}$$

For simplicity, in the following analysis we consider only unicast communication. Though we switch a node to idle mode for the entire transmission slot if the selected transmitter for the slot is a two-hop neighbor, we assume that a node could be either transmitting

or receiving during the control slot for our analysis. The probability of a node in transmit or receive mode during the control slot is given by q and $(1-q)$, respectively. Hence the energy consumption during the control slot is given by:

$$E_c = T_c.[(1-q).P_{rx} + q.P_{tx}] \tag{3.8}$$

During the data slot, a node could be in either transmit, receive, or standby mode. Nodes take a finite period of time to switch modes when switching from the control to the data slot. Therefore, the energy spent in switching should also be accounted. The energy consumption during the data slot is thus given by:

$$\begin{aligned} E_d = {} & (1-q).\{p.p_{me}.T_d.P_{rx} + \\ & (1-p_{me}+1-p).[P_{st}.(T_d-T_t)+P_{rx,st}.T_t]\} \\ & + q.\{p.T_d.P_{tx} + \\ & (1-p).[P_{st}.(T_d-T_t)+P_{tx,st}.T_t]\} \end{aligned} \tag{3.9}$$

The first term in Eq. 3.9 corresponds to the case where the node is not a winner of the current transmission slot. This occurs with probability $(1-q)$. The node can either switch to receive mode if it is the chosen receiver or to standby mode if it is not the intended receiver or if the transmitter does not have a packet to send. The second term in Eq. 3.9 corresponds to the case where the node is the winner of the transmission slot. Here the node can transmit or go to standby mode depending on whether it has a packet to send or not.

The energy consumption during the switching period T_t depends on the state of the node in the previous transmission slot and the state of the node in the next transmission slot. It is given by:

$$\begin{aligned} E_t = {} & T_t.\ \{q.[\ p.(q.P_{tx}+(1-q).P_{tx,rx}) \\ & +(1-p).(q.P_{st,tx}+(1-q).P_{st,rx})\ ] \\ & +(1-q).[p_{me}(p.(q.P_{rx,tx}+(1-q).P_{rx}) \\ & +(1-p).(q.P_{st,tx}+(1-q).P_{st,rx})) \\ & +(1-p_{me}).(q.P_{st,tx}+(1-q).P_{st,rx})]\} \end{aligned} \tag{3.10}$$

In the above equation all the possible state transitions are taken into account with the corresponding transition probabilities.

The average energy consumption during a transmission slot, E_{std}^{tot}, can be derived as:

$$E_{std}^{tot} = E_c + E_d + E_t \tag{3.11}$$

When using radios with no standby mode, a node could be either in transmit or receive mode. Therefore, the total energy consumption in this case is:

$$\begin{aligned} E_{nostd}^{tot} = T_t.\{&q.[q.P_{tx} + (1-q).P_{tx,rx}] \\ &+(1-q).[q.P_{rx,tx} + (1-q).P_{rx}]\} \\ &+(T_d + T_c).\{q.P_{tx} + (1-q).P_{rx}\} \end{aligned} \tag{3.12}$$

3.4 Performance Comparison

In this section the performance of DEANA is compared with that of regular NAMA. The performance metric used is the percentage of savings in energy and is computed as:

$$savings = [E_{nostd}^{tot} - E_{std}^{tot}]/E_{nostd}^{tot} \tag{3.13}$$

Note that nodes have to be in either receive or transmit mode during the random access period. Thus, the energy savings achieved by switching radios to standby mode only happen during transmission slots.

The radio we consider in our analysis, RFM TR1000 [59], has three modes of operation. Transmit-, receive-, and standby mode. The average power consumption and the latency involved in switching from one mode to another are given in Table 3.2 and Table 3.3 respectively. The data rate is 19.2*KBPS* and the length of the control and data slots depend on the size of the control and data packets and the channel propagation delay.

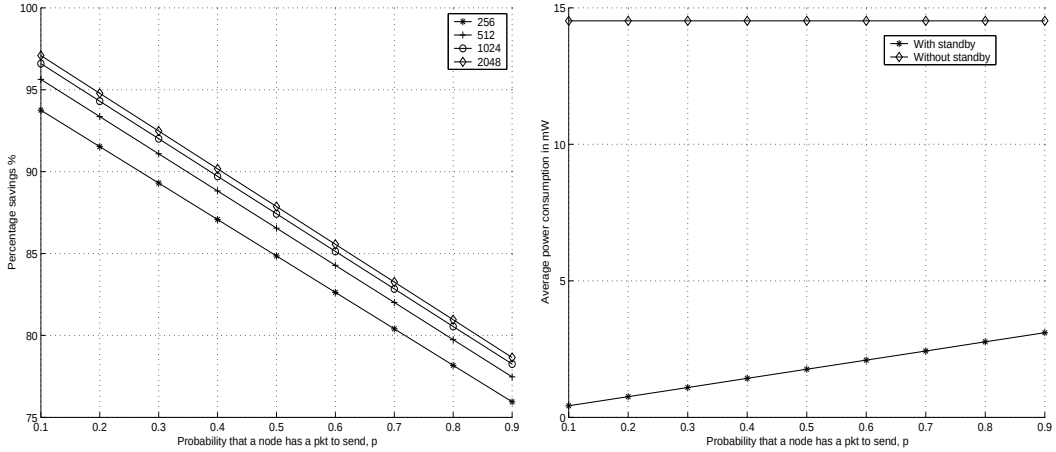
Figure 3.3(a) shows how energy savings is affected by p. For this analysis, The size of the control packet is fixed to 10 bytes and q is fixed at 0.1. We observe that energy savings are higher for lower values of p. For instance, for data packet size of 1024 bytes, energy savings

Table 3.2: Average power consumption in different modes

Mode	Power consumption in mW
Transmit	24.75
Receive	13.5
Standby	15×10^{-3}

Table 3.3: Transition times in μs

From/To	Transmit	Receive	Standby
Transmit	0	20	10
Receive	12	0	10
Standby	16	20	0



(a) Energy savings: different curves correspond to different sizes of the data packet (in bytes).

(b) Average power consumption

Figure 3.3: How p affects energy efficiency.

is around 78% for $p = 0.9$ and the savings are 85% for $p = 0.5$. This is due to the fact that if the selected node does not have any packets to send, then DEANA switches the node to standby mode for the entire transmission period. This improves the savings as the transmit mode power consumption is much higher than the standby mode power consumption. We also show the effect of control slot overhead in savings. Because nodes are either transmitting or receiving during control slots, the savings depends on the percentage of data slot during a transmission slot. For a fixed control slot size of 10 bytes, we vary the data packet size. As the size of the data packet increases, we observe increased savings. In other words, energy saved is mainly dependent on the percentage of data in the transmission slot.

The average power consumption for different values of p is plotted in Figure 3.3(b). In schemes without standby mode, power consumption is constant, whereas power consumption increases with p when standby mode is available since nodes switch to standby state if it does not have a packet to send.

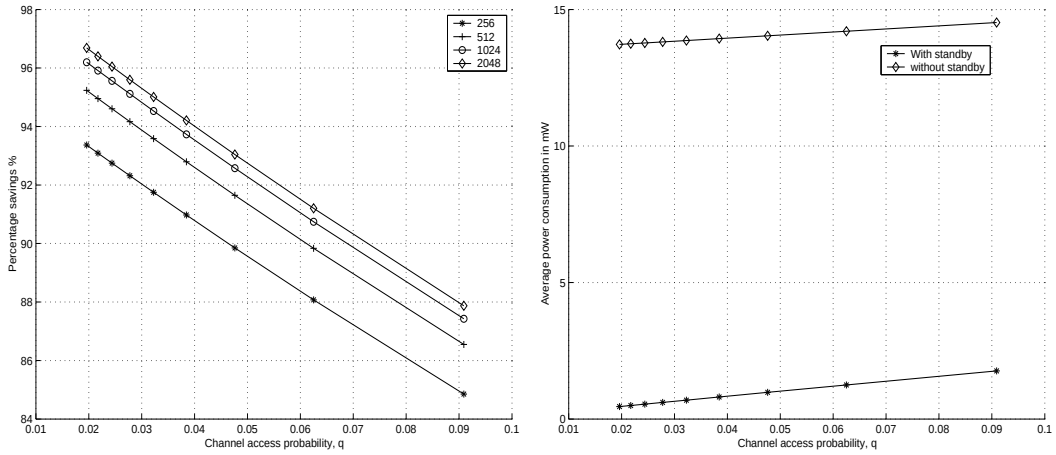
Figure 3.4(a) shows the effect of the channel access probability q with p fixed at 0.5. The energy savings decrease for higher q because nodes spend more time transmitting and receiving. Figure 3.4(b) shows the average power consumption for different values of q. As expected, the average power consumption increases for both schemes. As previously pointed out, this is due to the fact that as q increases the proportion of time spent by a node in transmit mode increases. Because power consumed in transmit mode is higher than in receive mode, the average energy consumption is higher when standby mode is not available.

Finally, we study the energy savings for different network densities and for different traffic conditions. The channel access probability q is dependent on the network density and p is dependent on the traffic conditions. For this experiment, we fix the data and control packet size at 512 and 10 bytes, respectively. Figure 3.5 shows the variation of energy savings with different traffic loads for different network densities. As we can observe, for a given network density the savings increases with a decrease in load. Also for the same traffic load, the savings increases with an increase in network density. As the network density is increased, the channel access probability decreases which leads to increased savings. For a moderately loaded system, the number of two-hop neighbors $N2$ ranges from 10 to 20. This gives a channel access probability of 0.091 and 0.047, respectively, assuming uniform distribution for channel access. The corresponding energy savings ranges from 77 to 88 % for $p = 0.9$.

3.5 Conclusion

In this chapter, we introduced the Distributed Energy-Aware Node Activation (DEANA) channel access protocol which uses a set of heuristics to select nodes to be switched to low-power, standby mode. Using an analytical model, we show that DEANA can achieve significant energy savings (up to 95%).

One of the main disadvantages of the proposed protocol is that, in practice, frequent switching between radio modes can lead to unnecessary power consumption and cancel out the benefits of switching nodes to standby mode. To address this problem, a link activation scheme or a scheme that can elect both transmitter and receiver in a distributed fashion could be used. In the next chapter we present traffic-adaptive scheduling protocol that eliminates the need for control slots within transmission slots.



(a) Energy savings: Different curves correspond to different sizes of the data packet (in bytes).

(b) Average power consumption

Figure 3.4: How q affects energy efficiency.

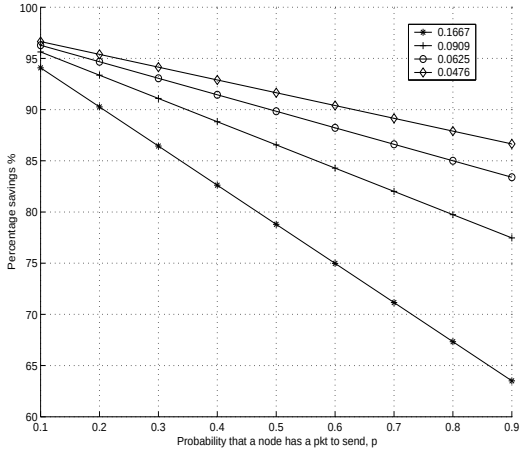


Figure 3.5: Effect of network density and traffic characteristics

Chapter 4

Traffic-adaptive Channel Access Protocol

In this chapter the traffic-adaptive medium access protocol (TRAMA) is introduced for energy-efficient collision-free channel access in wireless sensor networks. TRAMA reduces energy consumption by ensuring that unicast and broadcast transmissions incur no collisions, and by allowing nodes to assume a low-power, idle state whenever they are not transmitting or receiving. TRAMA assumes that time is slotted and uses a distributed election scheme based on information about traffic at each node to determine which node can transmit at a particular time slot. Using traffic information, TRAMA avoids assigning time slots to nodes with no traffic to send, and also allows nodes to determine when they can switch off to idle mode and not listen to the channel. TRAMA is shown to be fair and correct, in that no idle node is an intended receiver and no receiver suffers collisions. An analytical model to quantify the performance of TRAMA is presented and the results are verified by simulation. The performance of TRAMA is evaluated through extensive simulations using both synthetic- as well as sensor-network scenarios. The results indicate that TRAMA outperforms contention-based protocols (CSMA, 802.11 and S-MAC) and also static scheduled-access protocols (NAMA) with significant energy savings.

Section 4.1 introduces the TRaffic-Adaptive Medium Access (TRAMA) protocol, which provides energy-efficient conflict-free channel access in wireless sensor networks. Channel access in TRAMA is energy efficient while maintaining good throughput, acceptable latencies, and fairness. Energy efficiency is attained by (i) transmission schedules that avoid collisions of data packets at the receivers, and (ii) having nodes switch to low power radio mode when there is no data packets intended for those nodes. Adequate throughput and fairness is achieved by means of a transmitter-election algorithm that is inherently fair and promotes chan-

nel reuse as a function of the competing traffic around any given source or receiver. TRAMA derives collision-free transmission schedules based on (i) the identifiers of nodes one- and two hops away, (ii) the current time slot, and (iii) traffic information that specifies which node intends to transmit to which other node. Hence, the "sleep schedule" of a node is a direct function of the traffic going through the node and its neighbors, and is synchronized automatically when nodes exchange information about their identifiers and their traffic.

TRAMA is similar to the Node Activation Multiple Access (NAMA) protocol [3] as it provides conflict-free transmission by scheduling access among two-hop neighboring nodes during a particular time slot. However, NAMA does not address energy efficiency, i.e., nodes that are not transmitting switch to receiver mode. In contrast, TRAMA has energy efficiency as one of its main goals: it allows nodes to switch to sleep mode if they are not selected to transmit and are not the intended receivers of traffic for a particular time slot. Furthermore, by building schedules using traffic information, TRAMA, unlike NAMA, adapts to the application at hand. For instance, an event-tracking application will likely generate data only when an event is detected. On the other hand, monitoring applications may generate data continuously. In either case, TRAMA can adapt its schedules accordingly, delivering adequate performance and energy efficiency.

In contrast to prior MAC protocols proposed for sensor networks, TRAMA provides support for unicast, broadcast, *and* multicast traffic (i.e., transmitting to only a set of one-hop neighbors). TRAMA differs from S-MAC (which also provides explicit energy conservation mechanisms) in two fundamental ways: (i) TRAMA is inherently collision-free as its medium access control mechanism is schedule-based as opposed to S-MAC's which is contention-based; and (ii) TRAMA uses an adaptive, dynamic approach based on current traffic patterns to switch nodes to low power mode, while S-MAC's scheme is static based on a pre-defined duty cycle.

In Section 4.2, we show that TRAMA is fair and provides transmission schedules in which no collisions, idle listening, or idle senders occur. Section 4.3 presents an analytical performance comparison of TRAMA against a non-adaptive scheduled access MAC protocol. We evaluate the performance of TRAMA through extensive simulations using the Qualnet network simulator [60]. We compare the performance of TRAMA against three contention-based MAC protocols, namely CSMA, 802.11's DCF, and S-MAC, and also against NAMA, as a representative of collision-free channel access based on dynamic schedules. Section 4.4 describes our

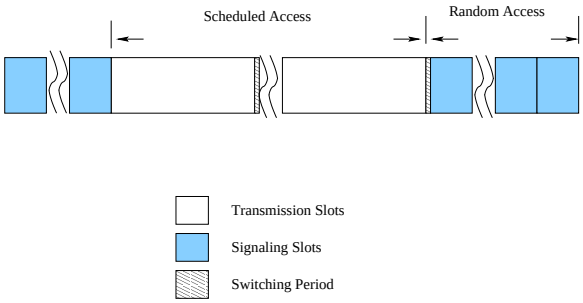


Figure 4.1: Time slot organization

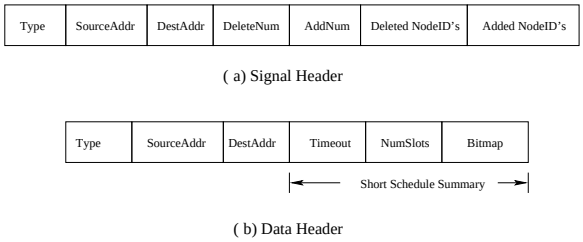


Figure 4.2: Signaling and data packet header format

simulation setup, and Section 4.5 presents simulation results. Our simulation results show that TRAMA exhibits superior end-to-end throughput (around 40% over S-MAC and CSMA and around 20% for 802.11) for both synthetic traffic models and traffic models that are sensor-network specific. This is because TRAMA avoids collisions due to hidden terminals. Our results also show that TRAMA achieves significant energy savings (since nodes can sleep for up to 87% of the time) and higher throughput. Section 4.6 presents concluding remarks.

4.1 TRAMA

4.1.1 Protocol Overview

TRAMA employs a traffic adaptive distributed election scheme that selects receivers based on schedules announced by transmitters. Nodes using TRAMA exchange their two-hop neighborhood information and the transmission schedules specifying which nodes are the intended receivers of their traffic in chronological order, and then select the nodes that should transmit and receive during each time slot. Accordingly, TRAMA consists of three components:

the Neighbor Protocol (NP) and the Schedule Exchange Protocol (SEP), which allow nodes to exchange two-hop neighbor information and their schedules; and the Adaptive Election Algorithm (AEA), which uses neighborhood and schedule information to select the transmitters and receivers for the current time slot, leaving all other nodes in liberty to switch to low-power mode.

TRAMA assumes a single, time-slotted channel for both data and signaling transmissions. Figure 4.1 shows the overall time-slot organization of the protocol. Time is organized as sections of random- and scheduled-access periods. We refer to random-access slots as *signaling slots* and scheduled-access slots as *transmission slots*. Because the data rates of a sensor network are relatively low, the duration of time slots is much larger than typical clock drifts. For example, for a 115.2 $Kbps$ radio, we use a transmission slot of approximately 46ms to transmit 512-byte application layer data units. Hence, clock drifts in the order of ms can be tolerated, and yet typically clock drifts are in the order of microseconds or even less. This allows very simple timestamp mechanisms (e.g., [16]) to be used for node synchronization. When much smaller clock drifts must be assumed and more expensive nodes can be used, nodes can be time synchronized using techniques such as GPS [15]. Accordingly, in the remainder of our description of TRAMA, we simply assume that adequate synchronization is attained.

NP propagates one-hop neighbor information among neighboring nodes during the random access period using the *signaling slots* to obtain consistent two-hop topology information across all nodes. As the name suggests, during the random access period, nodes perform contention-based channel acquisition and thus signaling packets are prone to collisions.

Transmission slots are used for collision-free data exchange and also for schedule propagation. Nodes use SEP to exchange traffic-based information, or schedules, with neighbors. Essentially, schedules contain current information on traffic coming from a node, i.e., the set of receivers for the traffic originating at the node. A node has to announce its schedule using SEP before starting actual transmissions. SEP maintains consistent schedule information across neighbors and updates the schedules periodically.

AEA selects transmitters and receivers to achieve collision-free transmission using the information obtained from NP and SEP. This is the case, because electing both the transmitter and the receiver(s) for a particular time slot is a necessity to achieve energy efficiency in a collision-free transmission schedule. Random transmitter selection leads to collisions, and

electing the transmitters and not the receivers for a given time slot leads to energy waste, because all the neighbors around a selected transmitter have to listen in the slot, even if they are not to receive any data. Furthermore, selecting a transmitter without regard to its traffic leads to low channel utilization, because the selected transmitter may not have any data to send to the selected receiver. Hence, AEA uses traffic information (i.e., which sender has traffic for which receivers) to improve channel utilization.

The length of a transmission slot is fixed based on the channel bandwidth and data size. Signaling packets are usually smaller than data packets and thus transmission slots are typically set as a multiple of signaling slots to allow for easy synchronization. In our implementation, transmission slots are seven times longer than signaling slots.

4.1.2 Access Modes and the Neighbor Protocol

In sensor networks, nodes may fail (e.g., power drained) or new nodes may be added (e.g., additional sensors deployed). To accommodate topology dynamics, TRAMA alternates between random- and scheduled access.

TRAMA starts in random access mode where each node transmits by selecting a slot randomly. Nodes can only join the network during random access periods. The duty cycle of random- versus scheduled access depends on the type of network. In more dynamic networks, random access periods should occur more often. In more static scenarios, the interval between random access periods could be larger, because topology changes need to be accommodated only occasionally. In the case of sensor networks, there is very little or no mobility, depending on the type of application. Hence, the main function of random access periods is to permit node additions and deletions. Time synchronization could be done during this period. During random access periods, all nodes must be in either transmit or receive state, so they can send out their neighborhood updates and receive updates from neighbors. Hence, the duration of the random access period plays a significant role in energy consumption.

During random access periods, signaling packets may be lost due to collisions, which can lead to inconsistent neighborhood information across nodes. To guarantee consistent neighborhood information with some degree of confidence, the length of the random access period and the number of retransmissions of signaling packets are set accordingly. In [4], it is shown that, for a network with an average of N two-hop neighbors, the number of signaling packet re-

transmissions should be 7 and the retransmission interval $1.44 * N$ to guarantee packet delivery of 99%. Thus, the length of the random access period will then be $7 * 1.44 * N$.

NP gathers neighborhood information by exchanging small signaling packets during the random access period. Figure 4.2(a) shows the format of the header of a signaling packet. Signaling packets carry incremental neighborhood updates and if there are no updates, signaling packets are sent as "keep-alive" beacons. Each node sends incremental updates about its one-hop neighborhood as a set of added and deleted neighbors. These signaling packets are also used to maintain connectivity between the neighbors. A node times out a neighbor if it does not hear from that neighbor for a certain period of time. The updates are retransmitted such that we ensure 0.99 probability of success. Because a node knows the one-hop neighbors of its one-hop neighbors, eventually consistent two-hop neighborhood information makes its way across the network.

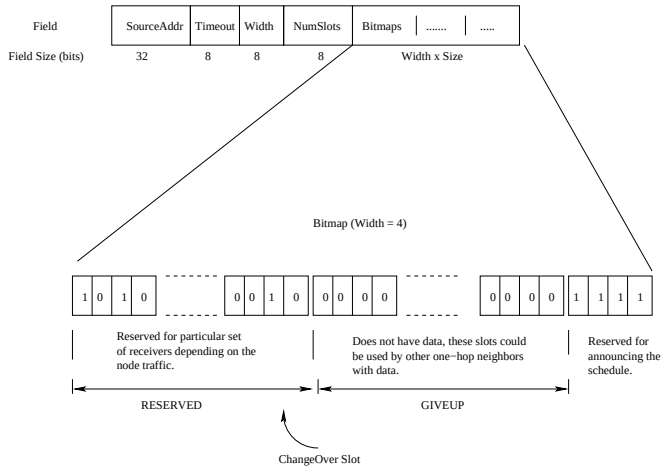


Figure 4.3: Schedule packet format

4.1.3 Schedule Exchange Protocol

SEP establishes and maintains traffic-based schedule information required by the transmitter (i.e., slot re-use) and receiver (i.e., sleep state switching) selection. A node's schedule captures a window of traffic to be transmitted by the node. This information is periodically broadcast to the node's one-hop neighbors during scheduled access.

Each node computes a *SCHEDULE_INTERVAL* based on the rate at which packets are

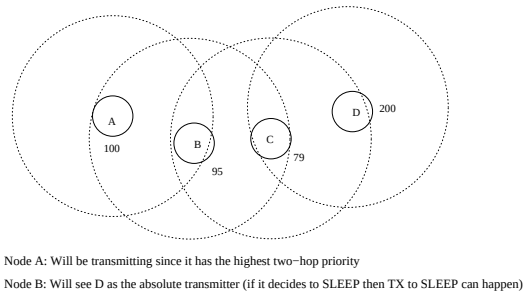


Figure 4.4: Inconsistency problem

produced by the higher layer application. The *SCHEDULE_INTERVAL* of a node represents the number of slots for which the node can announce the schedule to its neighbors according to the current state of its MAC-layer queue. The node then pre-computes the number of slots in the interval $[t, t + SCHEDULE\_INTERVAL]$ for which it has the highest priority among its two-hop neighbors (contenders), which we call "winning slots". Because these are the slots for which the node will be selected as the transmitter, the node announces the intended receivers for these slots. Alternatively, if a node does not have enough packets to transmit, it announces that it gives up the corresponding slot(s). Other nodes that have data to transmit can make use of these "vacant" slots. A node's last winning slot in this interval is reserved for broadcasting the node's schedule for the next interval. For example, suppose that node *u*'s *SCHEDULE_INTERVAL* is 100 slots. During time slot 1000, *u* computes its winning slots between $[1000, 1100]$. Let us assume that these slots are 1009, 1030, 1033, 1064, 1075, and 1098. Node *u* uses slot 1098, its last winning slot in this interval, to announce its next schedule by looking ahead from $[1098, 1198]$, and so on. The time corresponding to the last winning slot is fixed as the lifetime for the schedule.

Nodes announce their schedule via *schedule packets*. Because nodes have two-hop topology information obtained through NP, there is no need to send receiver addresses in the schedule packet. Instead, nodes convey intended receiver information using a bitmap whose length is equal to the number of one-hop neighbors. Each bit in the bitmap corresponds to one particular receiver ordered by their identities. The total number of receivers supported by this scheme depends on the size of the data slot and the number of slots for which receivers are announced. [1] For example, a node with four one-hop neighbors with identities 14, 7, 5 and 4

[1] Assuming the schedule is announced for 16 slots, the scheme can support 256 or 512 neighbors for a 512 or

will have a bitmap of size four with first MSB corresponding to node 14, second MSB to node 7. An advantage of using bitmaps is the ease with which broadcast and multicast communication can be supported. To broadcast a packet, all bitmap bits are set to 1, indicating that all one-hop neighbors are intended receivers of the packet. If the packet needs to be multicast to just 14 and 4, then only these bits are set in the bitmap. A node forms the bitmap for the winning slots based on the current traffic information for its queue. If the node's queue size is smaller than the number of bitmaps contained in the schedule, some of the winning slots will go unused. For these "vacant" slots, the node announces a zero bitmap. Slots with zero bitmaps could potentially be used by some other node in the two-hop neighborhood. The slot after which all the winning slots go unused is called *ChangeOver* slot. All unused slots happen contiguously toward the end before the last winning slot, which is reserved for announcing the next schedule. This maximizes the length of sleep periods.

Figure 4.3 shows the schedule packet format. *SourceAddr* is the address of the node announcing the schedule, *timeout* is the number of slots for which the schedule is valid (starting from the current slot), *width* is the length of the neighbor bitmap (i.e the number of one-hop neighbors), and *numSlots* is the total number of winning slots (i.e., the number of bitmaps contained in the packet). The last winning slot is always reserved for announcing the next schedule.

Additionally, a summary of a node's schedule is sent with every data packet. Schedule summaries help minimize the effects of packet loss in schedule dissemination. As shown in Figure 4.2(b), the summary includes the schedule's *timeout*, *numSlots*, and a bitmap corresponding to the winning slots in the current interval. The size of the bitmap is *numSlots* and is used to indicate whether the node is transmitting or giving up the corresponding slot. Note that, in order not to incur excessive overhead,[2] schedule summaries do not carry intended receiver information. They are meant to maintain synchronization[3] among one-hop neighbor schedules even in the face of losses. For example by inspecting the values of *numSlots* and the *bitmap* in the schedule summary, the receiving node can update or re-synchronize its stored schedule information. Each schedule has an associated timeout and nodes are not allowed to change the schedule until this timeout expires. This is required to ensure consistency across one-hop

1024 byte transmission slot size respectively.

[2]The overhead in the current implementation due to schedule summary is 6 bytes per data packet.

[3]As demonstrated in Section 4.2, TRAMA's correctness is not affected by unsynchronized schedules.

neighborhood schedules.

Nodes maintain schedule information for all their one-hop neighbors. The schedule information is consulted whenever a node has the highest two-hop priority to decide if the node will actually transmit (i.e., it has data to send and thus will use the slot) or will give up the slot to another node in the neighborhood. Based on this decision, the schedule information for the node is updated either using the short summary from the data packet (if the node is receiving), or assuming transmissions (if the node is sleeping since it is not the intended receiver of transmitter). In the latter case, its schedule is in an unsynchronized state until the node verifies or updates it based on the schedule summary piggybacked in a future data packet from that transmitter.

All nodes listen during the *ChangeOver* slot of the transmitter to synchronize their schedule. For instance, if a node *u* keeps assuming transmissions for a particular neighbor *tx* at different timeslots and the neighbor does not transmit any packets due to a contender that is hidden from *u*, then the schedule at node *u* for node *tx* will be unsynchronized. If node *u* does not listen during the *ChangeOver* slot, which is the last slot in the current schedule interval that will be used by node *tx* for transmission, it may assume that the node *tx* is transmitting the data corresponding to the *ChangeOver* slot and update the corresponding schedule. From now on until the schedule announced by node *tx* expires, node *u* considers that node *tx* gives up winning slots for re-use. This can lead to collisions if node *u* tries to reuse the winning slot of node *tx* that is actually used for transmission. Hence, schedules among neighbors could be unsynchronized only until the *ChangeOver* slot and a node has to listen during the *ChangeOver* slot of the transmitter.

It is possible for a node to get some extra slots for transmissions in addition to the original winning slots computed while announcing the schedule. To prevent inconsistencies and collisions when transmitting the schedule packet, a node should always send the schedule packet only on the previously announced timeout. Because all the slots after the *Changeover* slot are assumed as give-up slots by the neighbors, the schedules might be unsynchronized. Hence, transmitting a schedule before the timeout can potentially cause collisions with neighbors. In the following section, we describe how these schedule information is used to adaptively decide the node state.

4.1.4 Adaptive Election Algorithm

In the original NCR algorithm [3], a node is selected to transmit if it has the highest priority among its *contending set*. Node u's *contending set* is the set of all nodes that are in u's two-hop neighborhood. Node u's priority at time slot t is defined as the pseudo-random hash of the concatenation of node u's identity and t, or

$$prio(u,t) = hash(u \oplus t) \tag{4.1}$$

Assuming that node identities are unique and nodes are synchronized, all nodes compute the same priority value at any given time slot. However, if the selected node does not have any data to send, then the slot is wasted. Furthermore, nodes are free to transmit to any one-hop neighbor, because there is no sleep state in NCR.

For energy efficiency, TRAMA switches nodes to sleep state whenever possible, and attempts to re-use slots that are not used by the selected transmitter for bandwidth efficiency. A selected node may give up its transmission slot if it does not have any packets to send; this slot could then be used by another node. Nodes exchange current traffic information with their neighbors to make effective use of low-power, idle radio mode and accomplish slot re-use.

At any given time slot t during the scheduled access period, the state of a given node u is determined based on u's two-hop neighborhood information and the schedules announced by u's one-hop neighbors. Possible states are: transmit (TX), receive (RX), and sleep (SL).

At any given slot t, a node u is in the TX state if: (1) u has the highest *priority*, i.e., $prio(u,t)$ among its *contending set* and (2) u has data to send.

A node is in the RX state when it is the intended receiver of the current transmitter. Otherwise, the node can be switched off to the SL state, because it is not participating in any data exchange. This means that, if a node is not the selected transmitter, it will decide whether it needs to be in RX state by consulting the schedule sent out by the selected transmitter. If the transmitter does not have traffic destined for that node in the current slot, the node can then sleep.

Each node executes AEA to decide its current state (TX, RX, or SL) based on current node priorities (within its two-hop neighborhood) and also on the announced schedules from one-hop neighbors. The algorithm's pseudo-code is provided in Figure 4.15. Table 4.1 lists some basic terminology and notation used in the description of AEA.

Table 4.1: Notations and terminologies

$\mathbf{N2}(u)$	Set of neighbors of node *u* which are two-hops away.
$\mathbf{N1}(u)$	Set of neighbors of node *u* which are one-hop away.
$\mathbf{CS}(u)$	*u*'s *Contending Set* is the set of nodes in *u*'s two-hop neighborhood such that $\{u \cup \mathbf{N1}(u) \cup \mathbf{N2}(u)\}$.
$tx(u)$	*Absolute Winner* is the node with the highest priority in $\mathbf{CS}(u)$.
$atx(u)$	*Alternate Winner* is the node which has the highest priority among *u*'s one-hop neighbors, i.e., over the set $\{u \cup \mathbf{N1}(u)\}$.
$\mathbf{PTX}(u)$	*Possible Transmitter Set* is the set of all nodes in $\{u \cup \mathbf{N1}(u) - atx(u)\}$ that satisfy the condition given in Equation 4.2.
$\mathbf{NEED}(u)$	*Need Contender Set* is the set of nodes in $\{\mathbf{PTX}(u) \cup u\}$ that are in need of additional transmission slots.
$ntx(u)$	*Need Transmitter* is the node with the highest priority among the set of nodes $\mathbf{NEED}(u)$ containing valid synchronized schedule.

The state of a node depends on the *Absolute Winner* and the announced schedules from its one-hop neighbors. From node *u*'s point of view, the *Absolute Winner* at any given time slot *t* can be: (1) node *u* itself, (2) node *v* that lies in the two-hop neighborhood of node *u* in which case the *Alternate Winner* $atx(u)$ needs to be accounted for if hidden from node *v*, or (3) a node *w* that lies in node *u*'s one-hop neighborhood.

Whenever a node becomes an *Absolute Winner* for a particular timeslot and has announced a non-zero bitmap for this slot, it knows that no other node in its two-hop neighborhood will be transmitting in this slot. Thus, the node can transmit collision-free to its intended receiver(s). When a node is not an *Absolute Winner*, it is not certain who the actual transmitter for a particular slot is. For example, consider the topology shown in Figure 4.4. Let *D* be the node with highest priority in node *B*'s two-hop neighborhood in a given time slot and let *A* be the highest 2-hop priority node in node *A*'s two-hop neighborhood. Both *A* and *D* could transmit in the time slot because they are *Absolute Winners*; the *Absolute Winner* to node *B* is node *D*. Therefore, if *B* looks at its schedule information for *D* and finds out that it is not *D*'s intended receiver for the current slot, it will decide to switch to *SL* mode. However, if it happens to be *A*'s intended receiver, it will end up missing *A*'s transmission. Hence, before switching to *SL* mode, a node must also account for the *Alternate Winner*. This potential inconsistency occurs only if the *Alternate Winner* is hidden from the *Absolute Winner* i.e., they are three hops away.

To avoid wasting slots when the *Absolute Winner* has no data to send, TRAMA keeps track of nodes that could use extra slots to send their data. It first computes the set of nodes

that can possibly transmit at the current time slot. They are kept in the *Possible Transmitter Set*, which contains all nodes in the one-hop neighborhood that can possibly transmit without any collision. A node can transmit without collisions only if it has the highest priority in the two-hop neighborhood. Hence, a node checks for possible transmitters in the one-hop neighborhood using the available information. Because a node cannot know the entire two-hop neighborhood of its one-hop neighbors, it can only check if this neighbor has the highest priority among the nodes that are known to be the neighbor's two-hop neighbor. In other words, for a one-hop neighbor of node *u*, say node *y*, the following condition should be satisfied to be in the $\mathbf{PTX}(u)$:

$$prio(y) > prio(x) \forall x, x \in \mathbf{N1}(\mathbf{N1}(y)) \text{ and } x \notin \mathbf{N1}(y)) \tag{4.2}$$

The *Need Contender Set* is the subset of the *Possible Transmitter Set* and contains only those nodes that have data to send. Nodes for which node *u* does not have (valid) schedules are also included in this set as node *u* does not know whether these nodes have data to send.

The *Absolute Winner* is the assumed transmitter for a node, unless the *Alternate Winner* is hidden from *Absolute Winner* and it belongs to the *Possible Transmitter Set*. In the latter case, the *Alternate Winner* is the assumed transmitter. Whenever the assumed transmitter gives up, the *Need Contender Set* is checked and the node with the highest priority within this set is selected as the *Need Transmitter* $ntx(u)$. Nodes that are not in the schedule listed by the assumed transmitter can switch to *SL* mode to save energy. This is especially beneficial in scenarios in which only a few nodes generate data at a time and data are destined to small subset of receivers.

4.2 TRAMA Correctness

TRAMA is correct if it avoids collisions and transmissions to a sleeping node, both of which can cause packet losses. A node can, however, assume that some of its neighbors is transmitting when the transmission does not actually happen. Though this will lead to increased energy consumption because nodes may be in receive mode unnecessarily, it does not affect the correctness of the algorithm.

Arguing for collision freedom is simple. The only two ways in which a node *u* can be a transmitter is through line 4 in the pseudo-code, where the node is the *Absolute Winner* and line 34 in the pseudo-code, where the node is the *Need Transmitter*. In both cases, there cannot

be a node two-hops away from node *u* and transmitting. This follows from Equation 4.2 and by the definition of *Absolute Winner*. Hence, there can be no collisions due to transmissions from two-hop neighbors. Assuming that schedules are synchronized (which allows a node to know exactly whether the elected one-hop neighbor uses the slot or gives it up for re-use) and by virtue of the election mechanism, no other node that is one-hop away from node *u* can transmit. Hence, there can be no collisions due to a neighbor transmitting at the same time and the protocol maintains collision freedom at all the times.

To show that TRAMA never looses a packet due to an invalid state assignment (i.e., a node transmitting to a sleeping node), it is enough to show that, whenever a node *u* goes to sleep assuming that some node *v* is transmitting in its one-hop neighborhood (given that node *u* is not the intended receiver of node *v*), then no other node except node *v* can transmit in the one-hop neighborhood. A node always considers the transmitter to be either the *Absolute Winner*, the *Alternate Winner*, or the *Need Transmitter*. Because a node can receive only from a node that is one-hop away, the *Absolute Winner* should be a one-hop neighbor. Hence, if a node assumes that a neighbor is transmitting, it is either the *Alternate Winner* or the *Need Transmitter*.

Consider the case in which node *u* decides to sleep during a time slot *t* assuming that node *v* is the transmitter. Hence, node *v* has the highest priority among the two-hop neighbors of node *v* known to node *u* and among the one-hop neighbors of node *u* that have data to send. Let node *w* be the actual transmitter for the time slot *t* in the one-hop neighborhood of node *u*. This means that node *w* has the highest priority among the two-hop neighbors of node *w* and it has the highest priority among the one-hop neighbors of node *w* that have data to send. Node *w* can either be a one-hop neighbor to node *v* or a two-hop neighbor to node *v*. If node *w* is a one-hop neighbor of node *v*, then node *w* should have higher priority than node *v*. Because, node *v* and node *w* are neighbors of node *u* and the schedules are synchronized, it contradicts the fact that node *v* has the highest priority among the one-hop neighbors of node *u* that have data to send. Hence, node *w* cannot be the actual transmitter. When node *w* is a two-hop neighbor to node *v*, it should have higher priority than node *v*, which contradicts the fact that node *v* has the highest priority among the two-hop neighbors of node *v* known to node *u*. Hence, there cannot be a transmitter, node *w* in the one-hop neighborhood of node *u*.

Schedules can get unsynchronized for different reasons, and TRAMA is also correct in the face of unsynchronized schedules. For example when a node assumes that node *v* is

transmitting and decides to sleep, node *v* may not transmit. This will make the schedules unsynchronized and the *ChangeOver* slot for node *v* is reached earlier. The requirement that all the nodes should listen during the *ChangeOver* slot of the neighbors, prevents a node going to sleep or transmit state assuming that the neighbor is giving up and ensures correctness. Whenever a node assumes that a neighbor is transmitting a data to it, the schedules are updated only after receiving the data from the neighbor using the schedule summary. Hence, packet losses due to transmission errors can cause the schedules to be unsynchronized and forces a node to listen whenever the unsynchronized neighbor is elected for transmission. This continues until the node receives a data packet from the unsynchronized neighbor, and also prevents invalid state assignment. Hence, TRAMA is correct even when the schedules are not synchronized.

4.3 Analytical Model

In this section we present an analytical model for the delay performance of scheduling-access protocols (viz. NAMA and TRAMA) and use it to validate our simulation results.

The main goal of TRAMA's traffic-adaptive channel access is to achieve energy savings by making use of low-power, stand-by radio mode. The price to pay is the overhead involved in schedule propagation and also the latency inherent to scheduled-access MACs. Network latency consists of propagation, transmission and queuing delay; in the case of scheduled-access MACs, the latter component overshadows the other two. Scheduled-access using random priority for transmitter selection introduces a fixed delay irrespective of the traffic pattern. This *node activation* latency is mainly dependent on the number of contenders, which in turn depends on node density. In TRAMA, packets could be sent only after announcing the schedule. Hence, packets that arrive after the schedule announcement have to wait until the current schedule expires and a new schedule is announced. This introduces an additional *scheduling latency*, which, on average, is equal to $\frac{SCHEDULE\_INTERVAL}{2}$. The schedules are transmitted during a transmission slot in the scheduled access period. This increases the latency for data packets because a transmission slot is wasted for announcing the schedule. On the other hand, adaptive slot reuse allocates slots more frequently than plain node activation schemes (e.g., NAMA). This increases the effective channel access probability for TRAMA when compared to plain node activation.

We assume that nodes are uniformly placed over the given area and all nodes have

equal number of contenders. The channel access probability is the probability that a node wins a contention context and is given by,

$$q = \frac{1}{Number\ of\ contenders} \tag{4.3}$$

We also assume that channel access probability is uniform across nodes, packet arrivals follow a Poisson distribution. and all nodes generate equal length packets. The objective of the model is to determine a packet's average waiting time (before it gets serviced). We first derive this waiting time for scheduled access protocols and then extend it to NAMA and TRAMA.

A node can transmit only if it wins the contention context and this happens with probability q, the channel access probability, as given in Equation 4.3. A packet arriving at a node has to wait until the node succeeds in getting the channel. This can be modeled as the service time for a data packet. This service time could be modeled by a geometrically distributed random variable X with parameter q, which is the channel access probability. The random variable X is the number of the first winning slot. The mean and second moment of X are given by,

$$\bar{X} = \frac{1}{q},\ \bar{X^2} = \frac{(2-q)}{q^2} \tag{4.4}$$

Once the node services all the packets in the queue and goes idle, new arrivals to the idle system have to wait till the next winning slot. In other words, the node takes a vacation until the next winning slot. The interval until the next winning slot is again a geometrically distributed random variable V with parameter q. The random variable V is the number of failed slots before the first success. Thus, the mean and second moment of the vacation interval are:

$$\bar{V} = \frac{1-q}{q},\ \bar{V^2} = \frac{2-3q+q^2}{q^2} \tag{4.5}$$

This system can be modeled using a M/G/1 service model with vacations [5]. Both the service time and vacation time follow a geometric distribution. The waiting time can be readily obtained [5] as:

$$W = \frac{\bar{X^2} \cdot \lambda}{2(1-\lambda \cdot X)} + \frac{\bar{V^2}}{2V} \tag{4.6}$$

The values of $\bar{X}, \bar{X^2}, \bar{V}, \bar{V^2}$ can be substituted in Equation 4.6 to get the average waiting time in the queue.

For node activation (i.e., NAMA), the channel access probability depends only on the contending set which is the entire two-hop neighborhood i.e., $q_{NAMA} = \frac{1}{N1+N2+1}$. In TRAMA,

the channel access probability is also affected by adaptive slot reuse and schedule transmission. Adaptive slot reuse depends on the traffic in the one-hop neighborhood because we reallocate slots that are not used by winning nodes. These additional slots gained by slot reuse reduce the contender size by eliminating the set of one-hop nodes that do not have any data to send.

Thus, the effective contender set for calculating TRAMA's channel access probability is $(\mathbf{N2}(u) + \mathbf{N1}(u) + \mathrm{u} - \{onehop\ nodes\ without\ traffic\})$. We represent this effective channel access probability taking into account channel reuse by q_e.

The reduction in the channel access probability due to schedule transmission can be quantified as:

$$q_{TRAMA} = \frac{SCHEDULE\_INTERVAL \cdot q_e - 1}{SCHEDULE\_INTERVAL}, \tag{4.7}$$

where q_{TRAMA} is the actual channel access probability. Equation 4.7 based on the observation that one slot is wasted for schedule propagation every *SCHEDULE_INTERVAL*. As it is difficult to calculate q_e analytically, we use the original channel access probability q_{NAMA} to get a lower bound for TRAMA's channel access probability. The total waiting time for TRAMA also involves the *scheduling delay*. It is given as:

$$W = \frac{\bar{X^2} \cdot \lambda}{2(1 - \lambda \cdot \bar{X})} + \frac{\bar{V^2}}{2\bar{V}} + \frac{SCHEDULE\_INTERVAL}{2} \tag{4.8}$$

The term $\frac{SCHEDULE\_INTERVAL}{2}$ accounts for the additional delay faced by a packet as it has to wait until the schedule is announced.

Figure 4.5 shows the average delay (in number of slots) for NAMA and an upper bound [4] for the average delay for TRAMA for different two-hop neighborhood sizes. The number next to the protocol in the legend represents the total number of one-hop and two-hop neighbors (i.e. $N1 + N2$). The *SCHEDULE_INTERVAL* for TRAMA is fixed as 100 slots. As can be observed, TRAMA's average queueing delay is higher than NAMA. This is mainly due to the overhead introduced by the adaptive scheduling mechanism. In the next section we validate our analytical model by comparing to simulation results. We also show the improvement due to the adaptive slot reuse mechanism.

[4] The delay is the upper bound as we assume there is no slot reuse i.e. $q_e = q_{NAMA}$.

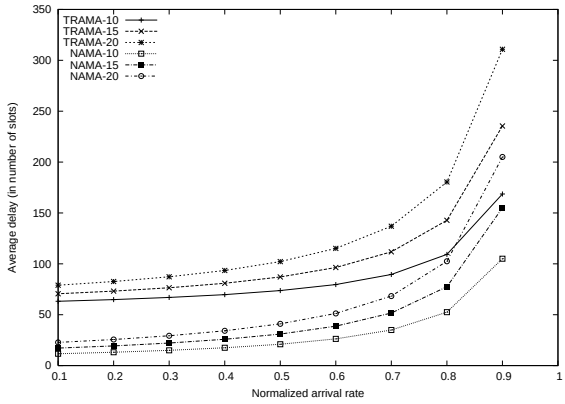


Figure 4.5: Average queueing delay for NAMA and TRAMA

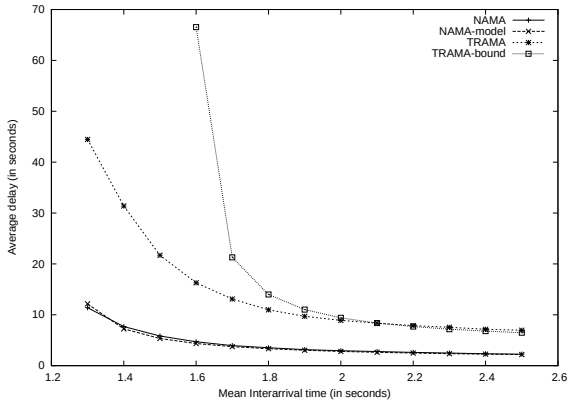


Figure 4.6: Comparison between average queueing delay obtained from analytical model and simulation for NAMA and TRAMA

4.3.1 Simulation Results

The results from the analytical model are verified by simulation using the Qualnet [60] network simulator. The simulation parameters are set to satisfy the assumptions made in deriving the model. To achieve a uniform channel access probability across nodes (i.e., all nodes have equal number of contenders, which is an assumption we make in our model), 100 nodes are deployed in a 650x650m square grid topology. The transmission range of the nodes is set to 104m [5] and the grid unit is 65m. On average, nodes in the center of the grid have a con-

[5]This value is set based on the type of radio we assume for sensor networks. This is discussed in more detail in Section 4.4.

tender size of 25 neighbors and this results in s a channel access probability of $q_{NAMA} = 0.04$ for NAMA. The $SCHEDULE\_INTERVAL$ is fixed at 100 slots for TRAMA. The nodes in the edge have lesser number of neighbors and hence higher channel access probability. Therefore, these nodes are not considered for gathering the delay measurement. Traffic is generated based on Poisson arrivals and the queueing delay is measured at the MAC layer queue. Traffic is generated throughout the simulation period of 600 seconds. Results are averaged over multiple runs and are compared results from our analytical model.

Figure 4.6 shows the average delay for different inter-arrival times for TRAMA and NAMA. The analytical results coincide with the simulation results for NAMA. The actual delay for TRAMA based on simulation is less than the delay obtained from the analytical model. This is consistent with the fact that the analytical model provides an upper bound for TRAMA's average delay and does not account for slot-reuse. We can clearly see the improvement in delay due to the adaptive slot reuse in TRAMA. For lower traffic rates, delay is mainly dominated by the *scheduling delay*. Hence, there is no improvement in the delay due to slot reuse. For higher traffic loads, the delay due to the contention for channel access dominates. Here the slot reuse mechanism improves the channel access probability and the total observed delay is less than the modeled delay.

4.4 Experimental Setup

Through simulations, we evaluate TRAMA and compare its performance against both contention- and scheduling-based protocols. While we consider Carrier-Sense Multiple Access (CSMA) [28], IEEE802.11 DCF [24] and S-MAC [67] as example contention-based protocols, we use Node Activation Multiple Access (NAMA) [3] as example scheduling-based protocol.

We used Qualnet [60] as our simulation platform and we present the results for a variety of scenarios. The underlying physical layer model used for all the experiments was based on the TR1000, a typical radio used in sensor networks. The TR1000 [59], the radio used by the UC Berkeley Motes [57], are short range, low data-rate (a maximum of 115.2KBPS) radios with built-in support for low-power sleep state. The average power consumption in transmit, receive and sleep modes is $24.75mW$, $13.5mW$ and $15\mu W$, respectively. The maximum transition time for switching is $20\mu S$. The modulation type used in the physical layer is ASK and the receiver threshold is $-75dBm$. Fifty nodes are uniformly distributed over a 500m x 500m area in all

the experiments. The transmission range of each node is 100m and the topology is such that the nodes have 6 one-hop neighbors on average. The average size of the two-hop neighborhood for this network is 17 nodes. Two different types of traffic load are considered in our study. We used a scenario in which node traffic is statistically generated based on a exponentially distributed inter-arrival time. We chose this to stress-test protocol performance for different arrival rates. We also test TRAMA's performance when driven by data gathering applications, which are considered typical of sensor networks. Below, we describe these traffic scenarios as well as other simulation parameters in detail.

4.4.1 Protocol Parameters

In both scenarios we fixed up the $SCHEDULE\_INTERVAL$ to be 100 transmission slots for TRAMA. The maximum size of a signaling packet is fixed at 128 bytes which gives to a slot period of $6.82ms$ with guard time to take care of switching. Transmission slots are seven times longer than the signaling slots supporting a maximum data fragment size 896 bytes. The random access period is fixed to 72 transmission slots and is repeated once every 10000 transmission slots.

S-MAC is a contention-based channel access protocol and it uses periodic sleep intervals to conserve energy. Sleep schedules are established using $SYNC$ packets which are exchanged once every $SYNC\_INTERVAL$. The duty cycle determines the length of the sleep interval.

We set $SYNC\_INTERVAL$ as $10sec$ and we varied the duty cycle (10% and 50%). All the nodes are time synchronized and hence we favored S-MAC by allowing the listen and sleep periods synchronized across the entire network. We also observed that S-MAC needed larger

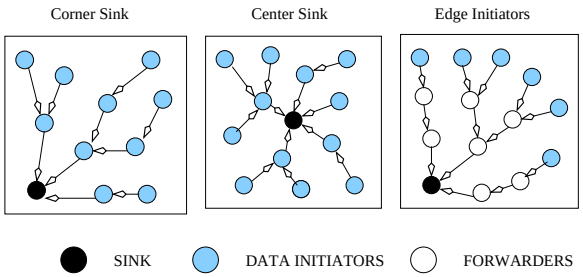


Figure 4.7: Data gathering application

time to set up the listen/sleep schedules with the neighbor. This is because S-MAC does not have a proper neighbor discovery protocol, it has to rely on the *SYNC* packets for doing this. *SYNC* packets are transmitted only once and are transmitted unreliably. Hence, a large warmup time of 20*sec* is allowed for the neighbor information to settle down. Because the queuing delay for the scheduling-type MAC's is higher, we allowed some more time for delivering the queued packets before ending the simulation. The simulation is run for 400*sec* and the results are averaged over multiple runs.

4.4.2 Synthetic Data Generation

The objective of this experiment is to measure the performance of TRAMA when all the nodes in the network generate traffic based on some statistical distribution. We used exponential inter-arrival for generating data and varied the rate from 0.5 to 2.5 seconds. A neighbor is randomly selected as a next-hop every time a node transmits a packet. We tested both unicast and broadcast data generation separately. The performance metrics are:

- **Average Packet Delivery Ratio:** It is the ratio of number of packets received to the number of packets sent averaged over all the nodes. For broadcast traffic a packet is counted to be received only if it is received by all the one-hop neighbors.
- **Percentage Sleep Time:** It is the ratio of the number of sleeping slots to the total number of slots averaged over the entire network.
- **Average queuing Delay:** Average delay for the packet to be delivered to the receiver
- **Average Sleep Interval**: This is the average length of sleeping interval. This measures the number of radio mode switching involved. Frequent switching can waste energy due to the transient power consumption involved in switching.[6]

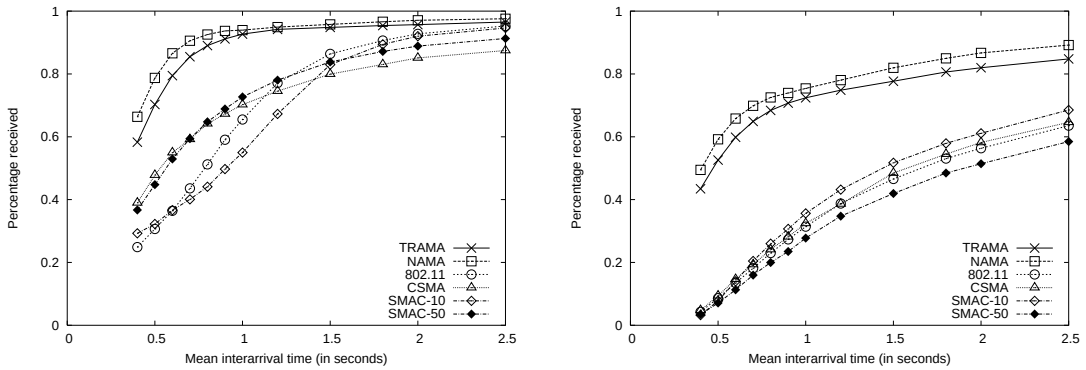
4.4.3 Data-Gathering Application

We assume a sink is collecting data from all the sensors for these experiments. The sink sends out a broadcast query requesting data from all the sensors. The sensors respond back with the data, which are generated periodically to the sink. We implemented a simple

[6]Measurements for 802.11 based radios are available in [18].

reverse-path routing to forward the data from the sensors to the sink. Figure 4.7 shows the three different scenarios considered for this study. Data-collection node or sink is placed in the corner for the first case and in the middle for the second case.

All the sensors respond with periodically generated data in both cases. Because data aggregation [25] or grouping data to minimize traffic, are advantageous, we also emulated data aggregation in a third case. Here, only the nodes at the edge generate traffic and we assume that the nodes do data aggregation and appends its reading to the parent node. To measure performance in these experiments we use the metrics defined for the synthetic case. The average packet delivery ratio is measured as the ratio of total number of data received by the sink to the total number of data sent by all the sensors, unlike the per-hop delivery ratio used for synthetic traffic generation.



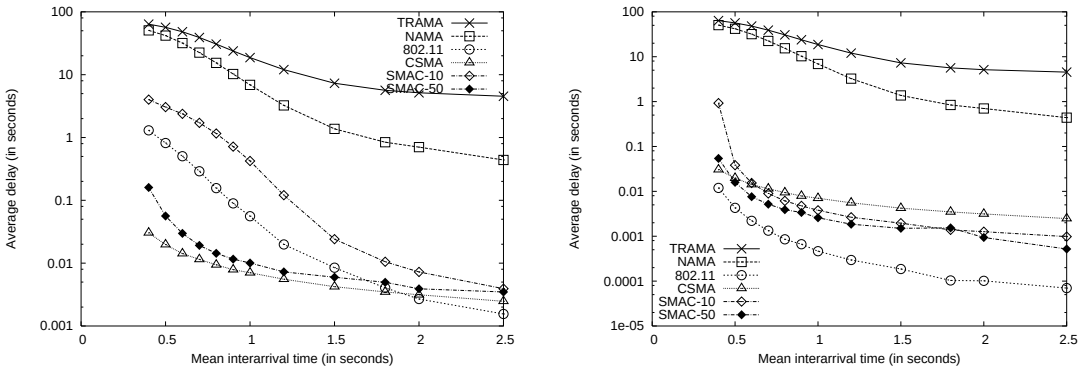
(a) Unicast Traffic (b) Broadcast Traffic

Figure 4.8: Average packet delivery ratio for synthetic traffic

4.5 Simulation Results

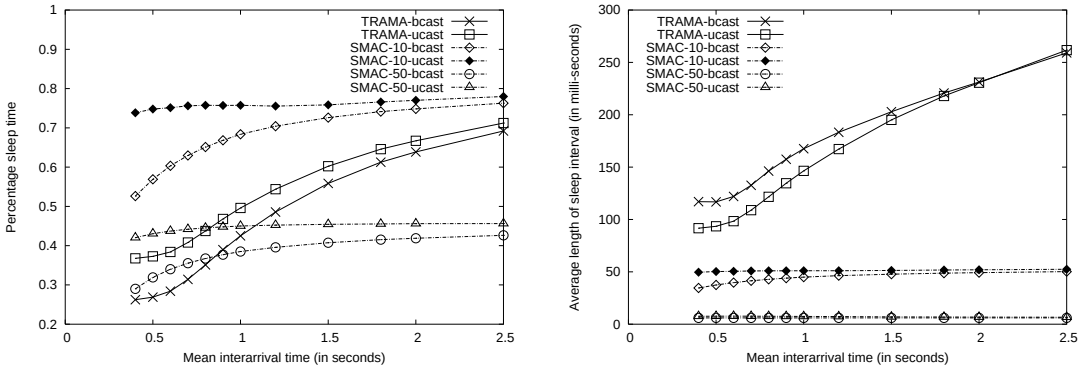
4.5.1 Synthetic Traffic

The packet delivery ratio, average queuing delay, percentage of sleep time, and average length of sleep intervals for synthetic traffic scenarios are shown in Figure 4.8, Figure 4.9, Figure 4.10(a) and Figure 4.10(b), respectively. We present results for S-MAC using two different duty cycles, namely 50% and 10%. Our results indicate that, in general, schedule-based MACs based on NCR achieve better delivery than IEEE802.11, CSMA and S-MAC. The main



(a) Unicast Traffic (b) Broadcast Traffic

Figure 4.9: Average queuing delay for synthetic traffic



(a) Percentage Energy Savings (b) Average Sleep Interval

Figure 4.10: Energy savings and average sleep interval for synthetic traffic

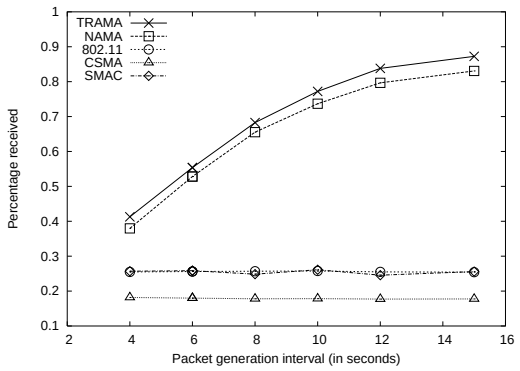
reason for the improvement in delivery is the collision freedom guaranteed at all times during data transmission. The effect is more noticeable when all nodes generate broadcast traffic. In CSMA, S-MAC and IEEE802.11, broadcasting is unreliable and susceptible to hidden-terminal collisions. This reduces broadcast delivery significantly when we increase the load as our results indicate. For IEEE802.11 and S-MAC, the RTS/CTS/DATA/ACK exchange for unicast traffic improves delivery when compared to broadcast traffic because it reduces hidden-terminal collisions by doing collision avoidance.

Schedule-based MACs, on the other hand, incur higher average queuing delays. We should point out that when measuring average delay, we account for the delay of packets suc-

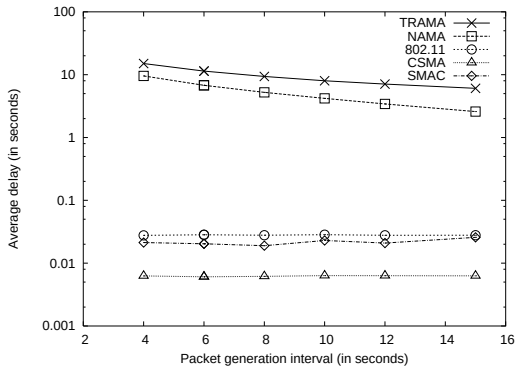
cessfully delivered. However, TRAMA and NAMA deliver more packets than contention-based MACs and this will reduce the retransmissions at the higher layers. Hence, the end-to-end delay perceived by the application will be comparable to that of contention-based protocols.

The average queuing delay for TRAMA is higher than that of NAMA due to the overhead involved in propagating scheduling information. Once every *SCHEDULE_INTERVAL*, a transmission slot is used for announcing schedules. This decreases the effective channel access probability for data transmission. This scenario is not a favorable scenario for traffic-adaptive elections, because the traffic is homogeneous across the network and all nodes periodically generate traffic. The throughput of TRAMA is comparable with that of NAMA and is significantly better than contention-based protocols for both unicast and broadcast traffic scenarios. The performance of the only other energy-efficient protocol, S-MAC is comparable with IEEE802.11 in terms of throughput. However, the delay is slightly higher than that of IEEE802.11 or CSMA due to the sleep periods, and it increases as the duty cycle of the listen periods is decreased. For a duty cycle of 50%, the delay of S-MAC is smaller than that of IEEE802.11 for the unicast-data generation scenario. This is because S-MAC frequently switches between sleep and listen modes(average sleep interval plotted in Figure 4.10(b) reflects this) for that duty cycle. This is equivalent to a node being awake most of the time, and the delay is less because S-MAC does not have any contention resolution algorithm.

The energy savings of TRAMA depend mainly on the traffic pattern, while the energy savings of S-MAC are dependent on the duty cycle. The total energy savings depend on both percentage sleep time and average length of sleep interval. Percentage sleep time metric does not account for the performance loss that is possible due to frequent radio mode switching. The average length of sleep interval quantifies the amount of radio-mode switching involved. A higher value of average sleep length is preferred because this implies less radio-mode switching and hence more savings. The results indicate that the percentage of sleep time is less for broadcast traffic when compared to unicast traffic, which is intuitive. The percentage sleep time of S-MAC increases as the duty cycle decreases. The price paid for decreasing the duty cycle is an increase in latency. The throughput decreases more steeply for a lower duty cycle as traffic increases. For the broadcast packets, a 50% duty cycle achieves less throughput than 10% duty cycle. Broadcast is done by plain carrier sensing and is more prone to hidden terminal collisions. The timing structure for 512 bytes data that favors low duty cycle as it reduces the

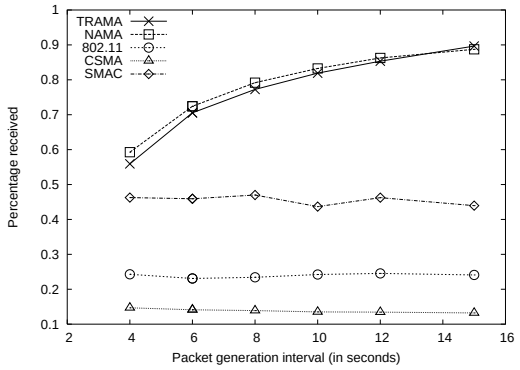


(a) Average delivery ratio

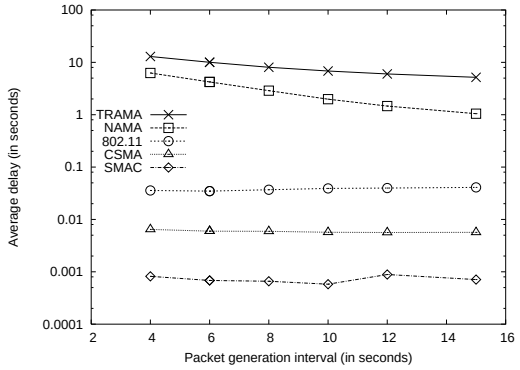


(b) Average queuing delay

Figure 4.11: Corner Sink



(a) Average delivery ratio

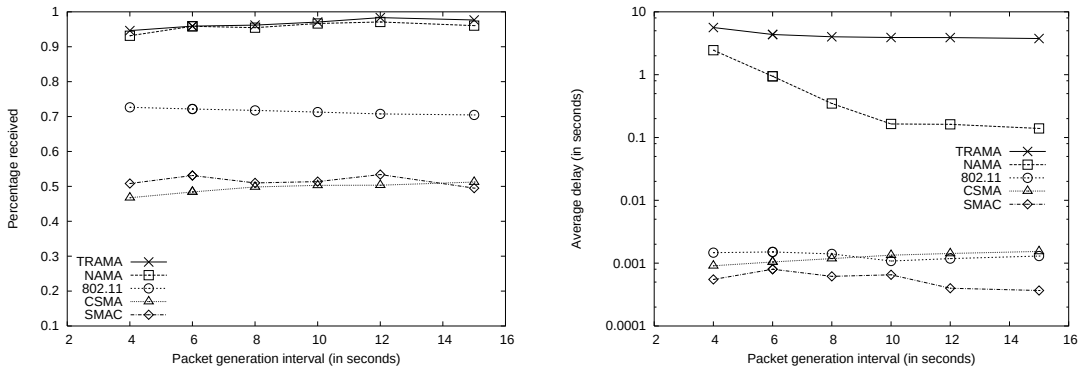


(b) Average queuing delay

Figure 4.12: Center Sink

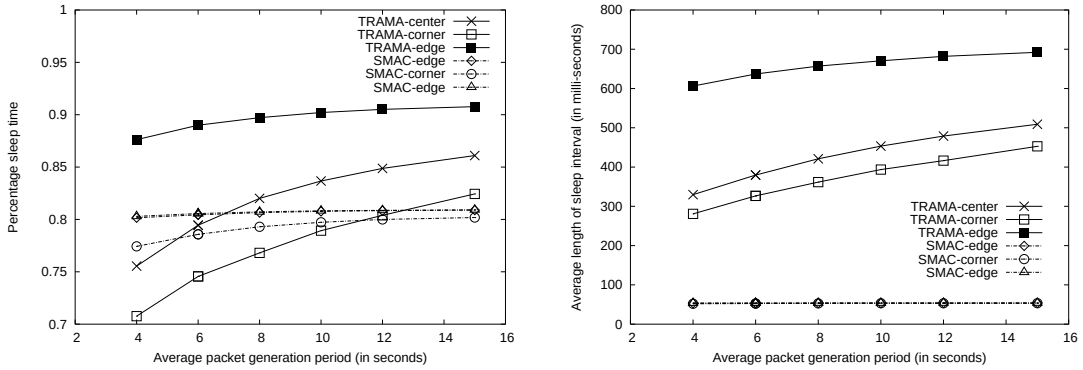
channel contention and hence the collisions are reduced. Note that a broadcast packet is counted as delivered ONLY if it is delivered to all the neighbors.

Compared with TRAMA, S-MAC with 10% duty cycle exhibits higher percentage of sleeping time. But the average length of sleep intervals is much lower for S-MAC when compared to TRAMA. This reduces the overall energy savings due to the overhead involved in mode switching. This is the case even though S-MAC is being favored by assuming that synchronized listen/sleep schedules are established across the nodes due to the simulation setup. The performance of TRAMA is not affected by nodes joining at discrete intervals because it does not require any synchronization of listen/sleep schedules.



(a) Average delivery ratio (b) Average queuing delay

Figure 4.13: Edge Sink



(a) Percentage Energy Savings (b) Average Sleep Interval

Figure 4.14: Energy savings and average sleep interval for sensor scenarios

In the subsequent experiments, we only consider S-MAC with 10% duty cycle, which has better performance than with a 50% duty cycle.

4.5.2 Sensor Network Application

We tested the protocols using a sensor network data gathering application. One of the nodes in the network is designated as the sink and the sink starts sending a broadcast query. All nodes receiving a non-duplicate query add the sender of the query as the next hop for data forwarding, establishing a reverse-shortest path tree with the sink node as the root. Figures 4.11(a), 4.12(a), and 4.13(a) show the average packet delivery ratio for the corner sink, center

sink, and edge sink scenarios respectively. Schedule-based MAC protocols outperform the contention-based MAC protocols in all the cases. The delivery is highest for the scenario in which the edge nodes are generating traffic. This is because the overall load in the network is low and well within the capacity of the protocols. Delivery to the center sink is slightly higher than when corner sink is used because the packets need to go through fewer number of hops to reach the sink. TRAMA performs much like NAMA and the decrease in throughput due to scheduling overhead is overcome by TRAMA's adaptive scheduling approach. The average delivery ratio is nearly constant for contention-based protocols as the variation in the offered load is not high.

Figures 4.11(b), 4.12(b), and 4.13(b) show the average per-hop delay for all the protocols for the sensor scenarios. Contention-based protocols outperform scheduling-based protocols in terms of delay. This is due to the latency introduced by random scheduling. Finally we show the percentage sleep time achieved by TRAMA and S-MAC in Figure 4.14(a). The percentage of time nodes can be put to sleep increases with decrease in traffic load. The percentage sleep time is quite high (as high as 85%) for the edge sink scenario which has the lowest load. Again the average length of the sleep interval is also the highest for this case. This clearly shows the benefit of TRAMA's traffic adaptability when compared to S-MAC. The average sleep interval of TRAMA is significantly higher than that of S-MAC. When compared to the edge sink scenario, the percentage sleep time is less for the center and corner sink scenarios due to the increased load. In the corner sink case, data forwarded by the nodes which are closer to the sink is heavier than data forwarded by nodes farther away. This reduces sleep time for these nodes and hence the overall percentage sleep time is lesser than the case where the sink is in the center. This also applies to the average length of sleep period shown in Figure 4.14(b).

4.6 Conclusion

In this chapter we presented TRAMA, a new energy-aware channel access protocol for sensor networks. TRAMA uses traffic-based scheduling to avoid wasting slots when nodes do not have data to send and to switch nodes to a low-power standby radio mode when they are not intended receivers of traffic.

Through extensive simulations, we compared TRAMA's performance against a number of contention- and a scheduled-based MACs. It is evident from the simulation results that

significant energy savings (since nodes can sleep for up to 87% of the time) can be achieved by TRAMA depending on the offered load. TRAMA also achieves higher throughput (around 40% over S-MAC and CSMA and around 20% over 802.11) when compared to contention-based protocols because it avoids collisions due to hidden terminals.

In general, scheduled-based MACs exhibit higher delays than contention-based MACs. In the case of TRAMA, the delay is higher than random-selection protocols (e.g., NAMA) due to the scheduling overhead. We presented an analytical model to quantify the delay for scheduling-based MACs and verified the model by simulations. TRAMA is well suited for applications that are not delay sensitive but require high delivery guarantees and energy efficiency. A typical example is sensor networks used for periodic data collection and monitoring applications.

```
1 Compute tx(u), atx(u) and ntx(u)
2 if (u = tx(u)) then
3   if (u.isScheduleAnnouncedForTx = TRUE) then
4     let u.state = TX
5     let u.receiver = u.reported.rxId
6     Transmit the packet and update the announced schedule
7   else if (u.giveup = TRUE) then
8     call HandleNeedTransmissions
9   endif
10 else if (tx(u) ∈ N1(u)) then
11   if (tx(u).announcedScheduleIsValid = TRUE AND tx(u).announcedGiveup =
TRUE) then
12     call HandleNeedTransmissions
13           else if (tx(u).announcedScheduleIsValid =
FALSE OR tx(u).announcedReceiver = u) then
14     let u.mode = RX
15   else
16     let u.mode = SL
17     Update schedule for tx(u)
18   endif
19 else
20   if (atx(u) hidden from tx(u) AND atx(u) ∈ PTX(u)) then
21     if (atx(u).announcedScheduleIsValid = TRUE AND atx(u).announcedGiveup =
TRUE) then
22       call HandleNeedTransmissions
23           else if (atx(u).announcedScheduleIsValid =
FALSE OR atx(u).announcedReceiver = u) then
24       let u.mode = RX
25     else
26       let u.mode = SL
27       Update schedule for atx(u)
28     endif
29   else
30     call HandleNeedTransmissions
31 endif
32 procedure HandleNeedTransmissions
33 if (ntx(u) = u) then
34   let u.state = TX
35   let u.receiver = u.reported.rxId
36     Transmit the packet and update the announced schedule
37           else if (ntx(u).announcedScheduleIsValid =
FALSE || ntx(u).announcedReceiver = u) then
38   let u.mode = RX
39 else
40   let u.mode = SL
41   Update the schedule for ntx(u)
42 endif
```


Figure 4.15: Pseudo-code description of AEA

Chapter 5

Flow-aware Channel Access Protocol

In the previous chapter we introduced TRAMA, which addresses energy efficiency by having nodes going into sleep mode if they are not selected to transmit and are not the intended receivers of traffic during a particular time slot. TRAMA uses traffic information to establish transmission schedules which are propagated to one-hop neighbors. This information is then used to define when nodes need to be in receive mode and when they can switch to low-power sleep mode. Besides its energy efficiency benefits, the use of traffic information also makes TRAMA adaptive to the sensor network application at hand. However, TRAMA's adaptiveness comes at a price, namely the complexity of its election algorithm and scheduling overhead for announcing traffic information. Schedule-based protocols exhibit inherently higher delivery delays when compared to contention-based approaches. In TRAMA, this is exacerbated by the need to propagate schedule information. This motivates the need for eliminating explicit traffic information exchange.

In this chapter, we introduce the FLow-Aware Medium Access (FLAMA) protocol, a schedule-based MAC protocol that leverages traffic predictability in sensor network applications. Traffic information can be determined by having the application explicitly specify its traffic characteristics, or by using traffic prediction techniques at each node. Depending on the application at hand, traffic prediction can be relatively simple. For instance, periodic data gathering (e.g., environmental monitoring) generates data streams over a collection tree rooted at the information sink and spanning all relevant nodes. When sending data, each node transmits to the upstream next-hop towards the sink. This information could be used to determine the next-hop node for a node's transmission.

Section 5.1 describes FLAMA in detail. FLAMA uses the concept of *flows* to characterize application traffic patterns. Flows represent one-hop traffic information and specify the transmitter, the receiver(s), and the rate at which packets are sent.

FLAMA uses flow-based traffic information to determine transmission schedules, as well as when nodes should be in receive mode or can switch to low-power sleep state.

Its main features are: (a) the distributed maintenance of energy-efficient, collision-free transmission schedules based on two-hop neighborhood information and implicit traffic information, (b) low transmission delays with limited processing and storage requirements, and (c) robust operation that accommodates topology changes.

We evaluate the performance of FLAMA through simulations and test-bed experimentation. Section 5.2 presents simulation results comparing the performance of FLAMA against two other MAC protocols. We uses the QualNet network simulator [60] for our simulation experiments, and compared the performance of FLAMA against TRAMA [42], an existing schedule-based MAC, and S-MAC. The results from our simulation study show that FLAMA achieves significantly lesser delay (up to 75 times) when compared to TRAMA, with significant improvement in energy savings and reliability when compared to TRAMA and S-MAC, demonstrating the importance of application-awareness in medium access scheduling.

Section 5.2.4 describes our implementation of FLAMA on TinyOS [21] for the Mica2 Motes platform [57] and presents experimental results comparing FLAMA and S-MAC in a sensor network test-bed. The results of our experiments show that FLAMA achieves 100% delivery compared to 75% for S-MAC at low offered loads (which are scenarios that favor contention-based MACs) and the average service time for FLAMA is an order of magnitude less than that of S-MAC.

Section 5.4 introduces the multi-channel extension to FLAMA and Section 5.5 presents simulation results quantifying the advantage of using multiple orthogonal channels for medium access. Finally, Section 5.6 presents concluding remarks.

5.1 Flow-Aware Medium Access

FLAMA uses a simple traffic adaptive, distributed election scheme for energy-efficient channel access. It requires two-hop neighborhood and flow information in the neighborhood to perform the election. Using only two-hop neighborhood information makes FLAMA scalable.

Time is organized in periods of random- and scheduled-access intervals as shown in Figure 5.1. We assume a single channel for data and signaling; however, FLAMA can be easily extended to handle multiple channels and Section 5.4 presents the multi-channel extensions. Channel access is contention-based during random-access and time-slotted during scheduled-access periods. During random access, neighbor discovery, time synchronization and implicit traffic information exchange are performed. Data transmission happens during scheduled access. Using periodic random-access periods allows FLAMA to adapt to topology and traffic changes in the network.

Unlike previous attempts at achieving adaptive scheduling in sensor networks (e.g., TRAMA [42]), FLAMA does not require explicit schedule announcements during scheduled access periods. Alternatively, application-specific traffic information is exchanged among nodes during random access to reflect the driving application's specific traffic patterns, or *flows*. This allows FLAMA to still adapt to changes in traffic behavior and topology (e.g., node failure). FLAMA uses flow information to establish transmission schedules for each node. Additionally, FLAMA achieves traffic adaptiveness by assigning slots to a node depending on the amount of traffic generated by that node. This is accomplished by assigning *node weights* based on the incoming and outgoing flows. Nodes with more outgoing flows are given higher weights (i.e., more slots); the net effect is that nodes that produce/forward more traffic are assigned more slots.

The implementation of FLAMA we showcase in this paper is customized for data gathering applications, an important class of sensor network applications. In data gathering scenarios, the information sink(s) sends out a query for a given sensor reading. When relevant sensors reply, a tree rooted at the sink is established. FLAMA uses this tree to define the corresponding flows. We discuss FLAMA's flow discovery mechanism in detail (and illustrate it with examples) in the remainder of this section.

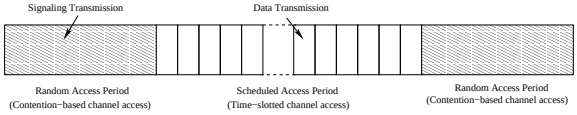


Figure 5.1: FLAMA's time organization.

5.1.1 Application Overview

We assume that in data gathering applications, the sink initially sends out a query requesting data from sensing nodes. As the replies from the sensors are forwarded back, a tree rooted at the sink spanning all relevant nodes is established. Sensor nodes then sample readings periodically and send them to the sink over the collection tree. On its way to the sink, data might be aggregated [25] to minimize energy consumption. We use the sink as the synchronization point for the other nodes (e.g., the sink may be connected to a backbone network and is synchronized to it).

For this type of data gathering scenarios, traffic is predictable and exhibit regular patterns, which can be exploited when designing MAC layer protocols tailored for these application scenarios. Since data is sent back to the sink along the forwarding tree, nodes can easily determine incoming and outgoing flows. More specifically, a node has incoming flows from all its children in the tree and it has only one outgoing flow to its parent. The sink does not have any outgoing flows.

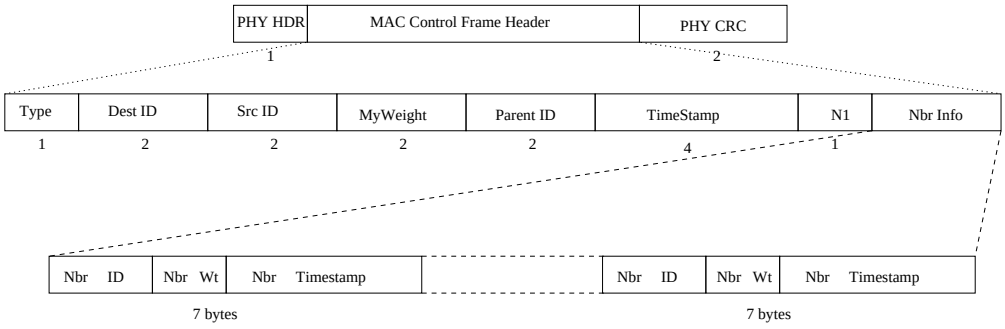
If R is the rate at which sensed data is generated at sensor nodes, all the nodes in the network except for the sink have an originating flow with data rate of R that exists for a period specified by the sink. A node has to either forward or aggregate flows that are incoming from its children. The outgoing flow rate is the sum of the incoming flow rates from the children and the originating flow rate R (if no data aggregation is employed [1]). FLAMA assigns node weights based on the resulting flow rates and performs traffic-adaptive scheduling. Section 5.1.2 describes how FLAMA acquires this information during the random access period.

5.1.2 Random-Access Period

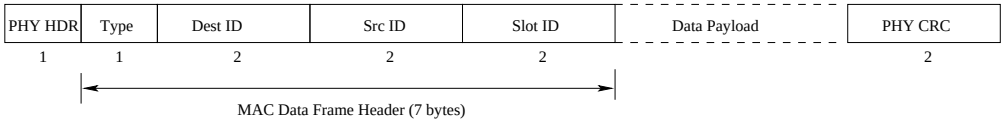
In FLAMA, random access is used for time synchronization, exchanging neighbor information, and establishing flow information. In the specific case of data gathering applications, establishing flow information is essentially forming the data forwarding tree. The data gathering node, or sink, initiates tree formation and time synchronization. Every node in the tree synchronizes with its parent using a pair-wise time synchronization algorithm based on timestamps.

Hence, during the random access period the following tasks that are necessary for

[1] If data aggregation is used, then the outgoing flow rate remains constant at R.



(a) MAC control frame



(b) MAC data frame

Figure 5.2: Frame formats.

FLAMA's operation are performed: (1) network-wide time synchronization, (2) data forwarding tree formation, (3) traffic flow information exchange and weight computation for traffic-adaptive election, and (4) two-hop neighborhood information and corresponding node weight exchange.

Nodes running FLAMA start in random access mode and the radio is in either transmit or receive state. Control frames are exchanged every $SYNC\_INTERVAL$. Two types of control frames ($SYNC$ and $SYNC\_REQ$) are exchanged during random access and channel access is based on carrier sensing. Figure 5.2(a) illustrates FLAMA's control frame format. Other than the source and destination information, the control frame also includes the node's outgoing flow weight , the node's parent, timestamp and a neighbor update list. The neighbor update list contains node identifiers for one-hop neighbors, their announced weights, and receive timestamps. In the case of data gathering applications, each node has only one outgoing flow towards the parent. Hence, it suffices to announce a single weight for the node. Other applications might need to announce multiple node weights based on the number of outgoing flows.

FLAMA requires time synchronization between two-hop neighbors. There are a number of known algorithms for time synchronization in ad hoc networks [14, 16, 20, 45] that can provide accuracy in the order of microseconds. The basic idea behind all these algorithms is time-stamping the packet at the lowest possible level and using these timestamps to calculate

clock drifts. We follow a similar approach to achieve time synchronization.

We employ a sender-initiated time synchronization mechanism where a node can send a *SYNC* frame only after synchronizing the clock with its parent. Otherwise, nodes send *SYNC_REQ* frames to discover parents. The sender (or the parent) initiates time synchronization by sending a *SYNC* frame with its local timestamp ($T1$). The receiver receives the frame at its local time $T2$. Now, $T2 = T1 + \delta + \tau$, where δ is the clock drift and τ is the propagation delay. The receiver replies with *SYNC_REQ* to the parent with its local timestamp ($T3$) and the sender receives the packet at its local time $T4$. Now, $T4 = T3 - \delta + \tau$. Using $T1, T2, T3$, and $T4$, both δ and τ can be calculated. As we require the receiver to adjust its clock based on the sender, the sender sends back a *SYNC* frame announcing the timestamp $T4$ to the receiver. The receiver computes the clock drift δ using the following expression and adjusts its clock:

$$\delta = (T2 - T1 + T3 - T4)/2 \tag{5.1}$$

Once a node becomes synchronized with its parent, it can start sending *SYNC* frames and synchronize downstream nodes. This process eventually synchronizes the entire network. Timestamps are generated at the physical layer to improve the accuracy. A node updates its child information whenever it receives *SYNC* frames with its node identifier as the parent. The length of the random access period is fixed based on the time required to complete the synchronization and tree formation processes.

During random access periods, signaling packets may be lost due to collisions. Hence, the interval should be long enough to accommodate signaling retransmissions. In general, the length of the random access period is $NUM\_RETX \times SYNC\_INTERVAL \times NET\_RADIUS$, where NUM_RETX is the desired number of retransmissions and NET_RADIUS is the network radius.

In our FLAMA implementation, we try to minimize state information exchanged and kept by nodes. Each node maintains the following neighborhood information: parent identifier (2 bytes), clock drift information ($T1$, $T2$, $T3$, $T4$, and $offset$, for a total of 20 bytes), one-hop neighbor table where each entry has 8 bytes of information (namely, node identifier, isChild flag, receive timestamp, node weight), and two-hop neighbor table (namely, node identifier, and node weight) with each entry having 3 bytes of information.

FLAMA uses node weights to adjust transmission schedules based on how much traffic individual nodes generate. Node weight calculation is illustrated using the example shown

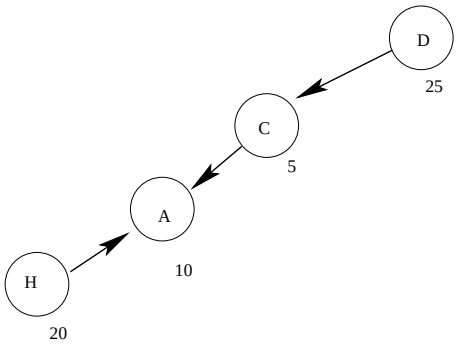


Figure 5.3: Topology.

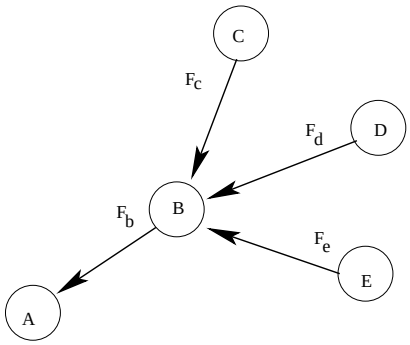


Figure 5.4: Traffic Flows.

in Figure 5.4, where arrows represent traffic flows with certain rates. For example, node B has three incoming flows (from nodes C, D, and E with rates F_c, F_d, and F_e, respectively) and a single outgoing flow to node A with a rate F_b. The outgoing flow rate F_b is a function of incoming flow rates and is given by:

$$F_b = F_{origin} + c \times F_c + d \times F_d + e \times F_e \tag{5.2}$$

where c, d and e denote the fraction of the flow that is forwarded. If the flows are "terminal flows" than c, d and e are 0. F_{origin} denotes the rate of the originating flow (if any) from node B. Node weights are directly proportional to the outgoing flow rate. Hence, node B's weight is decided based on F_b and is announced during random access.

5.1.3 Scheduled-Access Period

Setting the Slot Size

During scheduled-access, channel access is time-slotted. The slot interval is fixed based on a maximum physical layer frame size. In our implementation we used a packet size of 128 bytes which is the maximum physical layer packet size for TinyOS's CC1000 physical radio module. A guard interval is added to the time slot duration to account for synchronization errors and radio mode switching, and is set to a multiple of the maximum possible clock drift. The data frame format is shown in Figure 5.2(b). The number of slots in the scheduled-access period is decided based on the duty cycle for scheduled access. The distributed election algorithm described below is used to decide the state of each node at every slot.

Distributed Election Algorithm

FLAMA uses a distributed election algorithm to schedule collision-free transmissions. The design of the election algorithm is driven by the assumption that sensing nodes are typically limited in terms of processing and memory resources. Essentially, for each node, the election algorithm decides which radio mode to use in the current slot. The choices are transmit, receive, or sleep. FLAMA ensures that there is only one transmitter in the two-hop neighborhood and thus avoids hidden-terminal collisions. FLAMA's election algorithm requires that each node maintains a list of one- and two-hop neighbors and their corresponding weights, and parent information.

A node can transmit if it has the highest two-hop priority for the given time slot and it has data to send. A node should be in receive mode if it is not the highest two-hop priority node and its highest one-hop priority node is a child. Otherwise, a node can go to sleep. While in receive mode waiting for data, the node can switch to sleep mode if it does not start receiving data for $PREAMBLE\_INTERVAL$. Node weights computed during the random access period are incorporated into the election algorithm to provide more channel access for nodes with higher traffic rate. This makes FLAMA traffic-adaptive while maintaining the simplicity of the election algorithm.

Node priorities are calculated based on a pseudo-random function using the node

identifier (n), time-slot identifier (t) and node weight (*weight*) as shown below:

$$prio(n,t,weight) = pseudorandom(n+t) + weight \times C \tag{5.3}$$

where C is a constant multiplier. The pseudo-random function could be implemented using linear shift registers and $(n+t)$ determines the initial state of the register.

FLAMA achieves collision-freedom by allowing only one transmitter in the two-hop neighborhood. Due to limited neighborhood information and the distributed nature of the algorithm, special care should be taken to prevent a node from sleeping when a neighbor is transmitting a data packet destined to this node.

For example, consider the network shown in Figure 5.3. The priority values computed for the nodes are shown next to the node. According the node H it has the highest priority in two-hop neighborhood and will transmit to node A. However, highest priority two-hop node in node A's neighborhood is node C. If node A decides to switch its radio to stand-by mode, it will miss the data transmission from node H. This leads to transmission to a sleeping node.

To prevent this, the election algorithm can identify highest priority one-hop flows that are hidden from the highest priority two-hop flow and listen if needed. However, to identify hidden flows, a node should maintain complete topology information for the two-hop neighborhood. This is expensive when the available processing and memory resources are low. Alternatively, a node can just listen for a short interval during the start of the time slot to determine whether the highest priority one-hop flow is an incoming flow. If the node receives a start symbol during this period, it continues to listen and receives the packet. Otherwise, the node switches it radio to sleep mode. This method is easily implementable in today's radios and does not require maintenance of complex state information. In our implementation of FLAMA for MICA2 Motes we use this optimization. In case of powerful nodes (more processing power and memory) one can improve the efficiency of scheduling by maintaining more state information.

In the absence of flow information, the election could be carried out with one- and two-hop node identifiers. In this case, the receiver for the elected transmission is not known as the nodes do not have flow information. Energy-efficiency could be still achieved by using the technique mentioned in the previous paragraph.

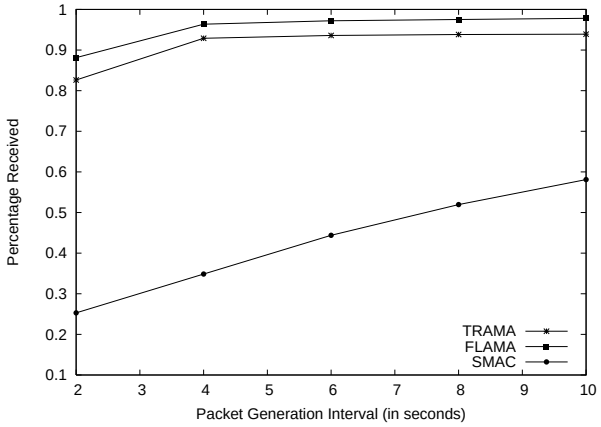


Figure 5.5: Average delivery ratio.

5.2 Performance Evaluation

FLAMA's performance is evaluated both by simulation and test-bed experimentation. The main goal of the simulation experiments is to highlight the importance of application-awareness in channel access scheduling. TRAMA is designed for general applications and hence, has to propagate traffic information explicitly and periodically. FLAMA, on the other hand, establishes flows based on traffic patterns exhibited by sensor network applications and need not propagate traffic information explicitly. S-MAC is also designed for general applications and does not account for application-specific traffic patterns. The main goal of our test-bed experiments is to establish the feasibility of implementing a TDMA-based, application-aware MAC protocols on sensor nodes and also to establish the advantages of FLAMA over traditional contention-based channel access protocols.

5.2.1 Performance Metrics

The following metrics are used to assess the performance of the protocols:

- **Average Packet Delivery Ratio** is the ratio of number of packets received at the sink to the number of packets sent by all sensor nodes. For broadcast traffic, a packet is counted to be received only if it is received by all the one-hop neighbors.

- **Percentage Sleep Time** is the ratio of the time spent in low-power sleep mode to the total experiment run time.
- **Latency** is computed as the average per-hop latency for the network.
- **Average Queue Drops** provides the average number of packets dropped at the MAC-layer queue.

5.2.2 Simulation Setup

To establish the importance of application-awareness, FLAMA's performance is compared against that of TRAMA and S-MAC. Qualnet [60] is used as the simulation platform. A physical layer model based on Mica2 motes' Chipcon CC1000 radio is implemented to accurately model the operating environment. The radio's data rate is $19.2Kbps$ and its range is around 300 feet.

Sensor network deployments for data gathering are often hierarchical, where there are some more capable data gathering nodes, each of which collect data from a subset of sensor nodes. We try to mimic this kind of deployment by using a grid topology with 16 nodes with the sink in the corner periodically issuing queries to the network to gather requested information. Nodes in the grid are separated by a distance of $75m$. All sensor nodes participating in the network report to the sink sending the requested information at the rate specified in the query. In our simulations, sensor nodes generate periodic 128-byte packets after an initial warmup time. This initial warmup period is needed to allow for neighbor discovery and is fixed at $50S$. The data generation rate is varied over multiple trials.

In FLAMA, flow discovery is done during the random-access period and this effectively establishes the data gathering tree. Since TRAMA and S-MAC do not perform flow discovery, we hard-coded the data collection tree for the simulation experiments involving TRAMA and S-MAC. The duty cycle for S-MAC is fixed at 10% and nodes are allowed to do adaptive listen at end of data transmission. S-MAC's synchronization interval is set to $10S$ and the contention window for data and synchronization packets are set to 31 and 15 slots, respectively. The simulation is run for 2000 seconds and results are averaged over multiple runs.

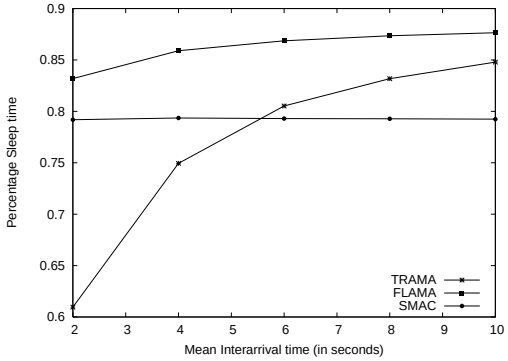


Figure 5.6: Energy savings.

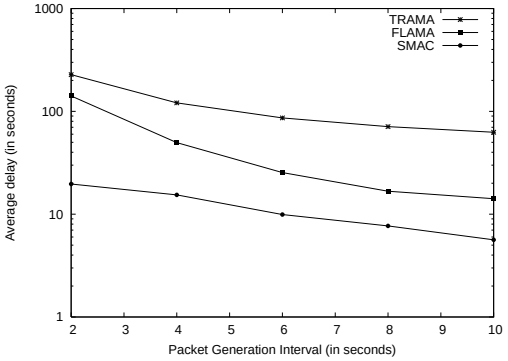


Figure 5.7: Average queueing delay.

5.2.3 Simulation Results

Figure 5.5 shows the average packet delivery ratio at the sink for different traffic generation intervals. FLAMA achieves better delivery ratio than TRAMA and S-MAC. This is due to the fact that FLAMA performs traffic adaptive scheduling without incurring much overhead. Nodes that are near the sink have a larger outgoing flow-rate and these nodes are favored in the election process. Whereas, TRAMA needs to propagate traffic information periodically and this is a significant overhead during scheduled access period. Hence, for the given simulation duration, FLAMA is able to service more packets than TRAMA.

We observed that the synchronized listen- and sleep cycles of S-MAC affect neighbor discovery and data throughput in multi-hop forwarding. This is because of the fact that S-MAC restricts transmitting or receiving packets to a specific (small) window of time. Depending on the contention window size for transmitting data and synchronization packets, collisions occur due to hidden terminals. This affects neighbor discovery significantly as the synchronization packets are sent by unreliable broadcasts. Hence, the average delivery ratio at the sink is significantly less for S-MAC when compared with scheduling-based protocols.

Figure 5.7 presents the average per-hop delay for FLAMA, TRAMA, and S-MAC. Overhead in periodic traffic announcements leads to higher queueing delay at intermediate nodes running TRAMA. The queueing delay for FLAMA is significantly lesser than that of TRAMA (up to 75 times). S-MAC achieves lesser delay than FLAMA in this topology. This is due to the delay involved in the election algorithm, which is dependent of the two-hop neighborhood size. However, it should be noted that FLAMA achieves much higher reliability than

S-MAC. Hence, end-to-end application perceived delay is much higher for S-MAC due to re-transmissions.

Energy efficiency for FLAMA, TRAMA, and S-MAC are shown in Figure 5.6. We observe that FLAMA achieves significant energy savings when compared to TRAMA and S-MAC. This is because FLAMA exchanges lesser information than TRAMA during scheduled access periods. For both FLAMA and TRAMA the energy savings are proportional to the offered load as expected. For S-MAC, energy savings depends on the fixed duty cycle.

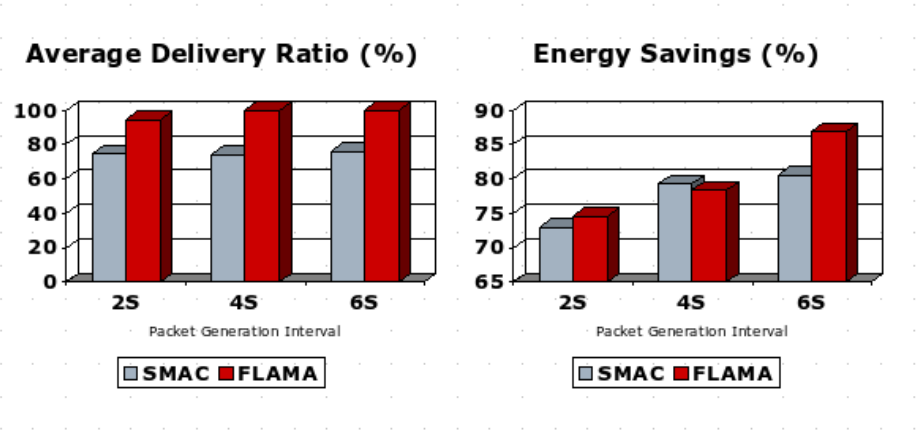


Figure 5.8: Test-bed results

Table 5.1: Average queue drops

Packet generation Rate (seconds)	Average Drops (FLAMA)	Average Drops (S-MAC)
2	0	13.66
4	0	9
6	0	0.33

5.2.4 Test-bed Experiments

We implement FLAMA on TinyOS for Mica2 motes and its performance is compared with that of S-MAC. Similarly to S-MAC, FLAMA is implemented on top of ISI's radio communication stack [68] for the Mica2 platform. ISI's S-MAC implementation is used for the experiments. Both for S-MAC and FLAMA, there is no MAC layer buffer to queue up frames. Hence, a frame from the application is dropped if the send buffer is full.

In the topology chosen for initial evaluation, a sink collects periodic data generated by sensor nodes and all the nodes are directly connected to the sink and the placements are such that hidden terminals exist. The main goals of the test-bed experiments are: to showcase that FLAMA can be implemented on sensor network platforms and also compare FLAMA's performance with S-MAC in a sensor network test-bed (instead of just through simulations).

During the experiments, each node maintains statistics about the number of data packets generated, number of data packets forwarded, number of data packets dropped due to buffer overflow, average service time for the packets, and radio statistics (i.e., time spent in transmit, receive, and standby mode). Statistics information is sent to the sink periodically along with the data. The sink is connected to an end host, and forwards all packets received to the host. Statistics are collected and processed by the host computer for every packet received at the sink.

We considered 128 bytes of data payload for both S-MAC and FLAMA. The routing information is hard-coded for S-MAC and the experiments are run multiple times to average the results. Identical operating environments are ensured for both S-MAC and FLAMA to avoid measurement errors. For S-MAC we considered 90% duty cycle for sleeping and for FLAMA we used 90% duty cycle for the scheduled access period. The length of the random-access period is fixed to 55s. The experiments are run for 400s so that there is enough time spent in scheduled access for FLAMA.

We consider different data rates for traffic generation and the results of our experiments are summarized in Figure 5.8. As we can observe, FLAMA significantly outperforms S-MAC in terms of delivery ratio, drop rate, and energy efficiency. For this topology, the average service time for S-MAC is on the order of 700ms, while for FLAMA the service time is around 100ms. Hence, the number of packets dropped for S-MAC is significantly higher than that of FLAMA as shown in Table 5.1. This affects the end-to-end reliability measured at the sink. It should be noted that FLAMA's delay is dependent on the number of two-hop nodes.

For low data rates, FLAMA achieves perfect reliability while S-MAC's reliability is 75%. This is because FLAMA avoids collision and transmissions to sleeping node. Also it does not exchange any control packets during the scheduled access period and hence the channel contention level is less. On the other hand, even though S-MAC uses RTS/CTS handshakes to avoid hidden-terminal collisions, it loses data packets due to increased service time and also due to RTS/CTS handshake failures. Note that low offered loads tend to benefit contention-based

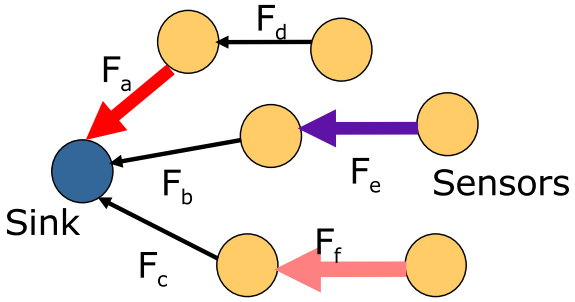


Figure 5.9: Multi-channel scheduling in data gathering applications.

protocols.

The average number of drops reported in the Table 5.1 reflects the number of packets dropped at the network layer (i.e., the buffer is full when a new packet arrives). In addition to these losses, there can be losses due to collisions, transmission errors, and RTS/CTS handshake failures. Hence, there is no direct correlation between the number of losses reported in Table 5.1 and the end-to-end delivery ratio. The results also indicate that FLAMA achieves energy savings comparable to S-MAC. This is in spite of the fact that FLAMA has the radio on during the entire random access period. This clearly demonstrates the importance of using an adaptive scheduling approach for channel access in sensor networks.

5.3 Multi-channel Medium Access

Commercial sensor network radios [8] often support multiple orthogonal communication channels. As shown in Figure 5.9, channel utilization can be improved by scheduling transmissions across multiple channels. This particular example illustrates a data gathering application, in which a sink is collecting data from all the sensors using a data forwarding tree. For the given traffic flow pattern, only one of the flows can be scheduled for transmission without hidden-terminal collisions using a single channel. However, with multiple channels, three flows (i.e., F_d, F_e, and F_f) can be scheduled concurrently without hidden-terminal collisions.

In previous sections, we established the importance of application-aware medium access in sensor networks. In the following section, we introduce the Multi-Channel FLow-Aware Medium Access Control protocol, or MFLAMA, a multi-channel extension of FLAMA proto-

col. MFLAMA leverages both the traffic predictability in some sensor network applications and also the availability of multiple orthogonal channels.

We evaluate the performance of MFLAMA through extensive simulations. Section 5.2 presents our simulation results which quantify the benefits of multi-channel scheduling. We observe that as we increase the number of orthogonal channels used for communication, there is significant improvement in channel utilization and queueing delay. However, we notice a "diminishing returns" effect as we increase the number of channels, i.e., the performance improvements observed decrease with the number of channels beyond a certain threshold. This threshold depends on the topology and traffic flow patterns being used.

5.4 MFLAMA

MFLAMA uses a distributed algorithm to establish transmission schedules across multiple channels. We assume that (a) all nodes are equipped with a single radio that can be tuned to transmit/receive in different orthogonal channels and (b) channel access is time-slotted. MFLAMA extends the FLAMA [41] approach to support scheduling across multiple channels. While the neighbor discovery, traffic characterization, and time structure organization of MFLAMA is similar to that of FLAMA, MFLAMA uses a novel distributed algorithm to schedule transmissions across multiple channels that guarantees collision freedom as well as no transmissions to sleeping nodes. For that reason, MFLAMA also requires additional signaling information as will be described in detail below.

MFLAMA requires consistent two-hop neighborhood and flow information to establish data transmission schedules. Similar to FLAMA, time is organized in periods of random- and scheduled-access intervals. Channel access is contention-based during random-access and time-slotted during scheduled-access periods. During random access all the nodes listen in the same channel (control channel). Neighbor discovery, time synchronization, and implicit traffic information exchange are performed during this period. Data transmissions are scheduled across multiple channels during scheduled access.

One notable difference when compared to FLAMA is that MFLAMA requires additional signaling information to accommodate collision-free, multi-channel scheduling. Table 5.2 presents the signaling information that is exchanged during MFLAMA's random-access period. Unlike FLAMA, MFLAMA requires the parent identifiers for all its one-hop neighbors.

Table 5.2: MFLAMA Signaling Information

Size (bytes)	Field	Description
1	len	physical layer length
1	type	packet type, SYNC or SYNC_REQ
2	dst	destination node address
2	src	source node address
4	st	start time (Sched Access)
4	ts	time stamp of this packet
2	parent	parent of src node
1	weight	cumulative weight
1	nn	numNodes, num of one hop neighbors
1	seq	seq num of this update
2 * nn	oh	one hop node ids.
2 * nn	oh_p	one hop parent ids
4 * nn	oh_ts	time stamp last heard from
1 * nn	oh_wt	node weights
1 * nn	oh_seq	last seen seq num

This information will be used by the distributed scheduling algorithm for collision freedom and correctness.

For each node at every slot, the election algorithm decides which radio mode, (*transmit, receive, or sleep*), as well as which channel to use. The algorithm ensures that there is only one transmitter in the two-hop neighborhood per channel and thus avoids hidden-terminal collisions. It also ensures that if a transmitter is elected to transmit in a particular channel, then the intended receiver listens in the same channel without any inconsistency.

Equation 5.3 is used to compute the node priorities. Each node is assigned a unique transmission channel from the set of available channels based on the node identifier (n) and time-slot identifier (t) using the same pseudo-random function.

Due to limited neighborhood information and the distributed nature of the algorithm, special care should be taken to prevent a node from sleeping or listening to another channel when it is the intended receiver of a a neighbor's transmission. To ensure this, for a given *parent* (receiver node), only the highest priority one-hop *child* is allowed to transmit. Hence, a node always listens to its highest priority one-hop child (if it is not a transmitter) on the channel chosen by the transmitting child.

```
1  Compute SortedOneHop(u,t) based on descending order of node priorities.
2  Initialize parentAvailable = TRUE; UsedChannelList = ∅; u.state = UNKNOWN;
3  foreach (node ∈ SortedOneHop(u,t)) begin
4    if (node == u) then : Out-going flow to parent
5      foreach (twoHop ∈ TwoHopList(u) begin
6        if PriorityHigh(twoHop, u) then : TwoHop higher priority
7              if (TXCHANNEL(u) == TXCHANNEL(twoHop) || u.parent ==
twoHop.parent) begin
8                let u.state = SLEEP; break ;
9            endif
10         endif
11     end
12       if (u.state == UNKNOWN && parentAvailable && TXCHANNEL(u) ∋
UsedChannelList) then
13         let u.state = TX; u.txchan = TXCHANNEL(u); u.rx = parent;
14     else let u.state = SLEEP; break ;
15   end
16   if (node == CHILD(u)) then : Incoming flow from child
17     let u.state = RX; u.rxchan = TXCHANNEL(node); u.tx = node;
18   else
19     let UsedChannelList = {UsedChannelList, TXCHANNEL(node)};
20     if (node == u.parent) then let parentAvailable = FALSE endif
21 end
22 if (u.state == UNKNOWN) let u.state = SLEEP endif
```

Figure 5.10: MFLAMA election algorithm pseudo-code

A node can transmit if it has the highest two-hop priority for that given time slot using the selected transmission channel and the receiver (*parent*) is available to receive.

A node can turn off its radio, i.e., go to sleep, if (a) it is an elected transmitter and does not have data to send, or (b) it is an elected transmitter with receiver conflict. While in receive mode waiting for data, the node can switch to sleep mode if it does not start receiving data for $PREAMBLE\_INTERVAL$.

The pseudo-code of the election algorithm is presented in Figure 5.10. Node weights are computed during the random access period and are incorporated into the election algorithm to provide more channel access for nodes with higher traffic rates. This makes MFLAMA traffic-adaptive while maintaining the simplicity of the election algorithm.

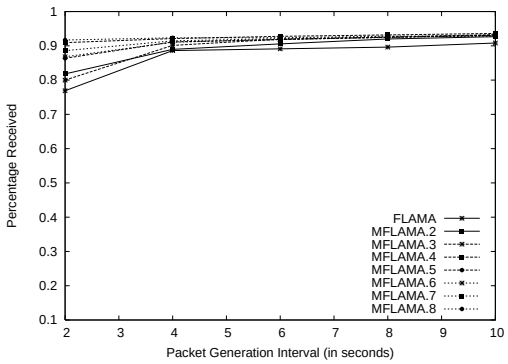


Figure 5.11: Average delivery ratio

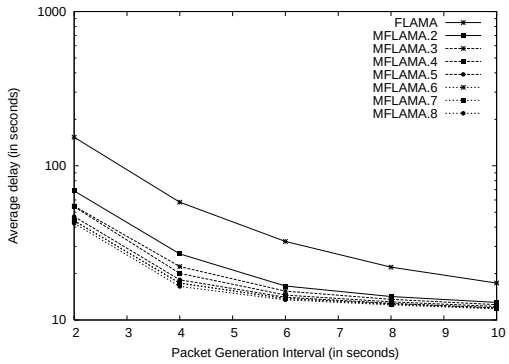


Figure 5.12: Average queueing delay

5.5 MFLAMA Performance Evaluation

MFLAMA's performance is evaluated through extensive simulation experiments using Qualnet [60]. The goal of the simulation study is to quantify the performance improvements that can be gained by using multi-channel when compared to single channel scheduling (FLAMA). Hence, we used the same simulation setup followed for the evaluation FLAMA as described in Section 5.2.2.

5.5.1 MFLAMA Simulation Results

Figure 5.11 shows the average packet delivery ratio at the sink and Figure 5.12 presents the average per-hop queueing delay for different traffic generation intervals for FLAMA and MFLAMA (with different number of total available channels). As we increase the offered load, the delivery ratio of the scheduling-based protocols decreases and is mainly caused by packet losses due to buffer overflows. As the number of orthogonal channels available for communication increases, the per-hop queueing delay decreases and this leads to an improved delivery ratio at higher offered loads.

This is due to the fact that with multiple channels available for communication, simultaneous transmissions can be scheduled without collisions. However, the improvement in channel utilization is limited by the availability of the simultaneous transmitter(s) and receiver(s) in the two-hop neighborhood. This depends on the node density and the traffic flow pattern of the application. For example, in a data-gathering application, near the data collection node (sink), the main bottle-neck for channel utilization is the availability of the sink rather than the

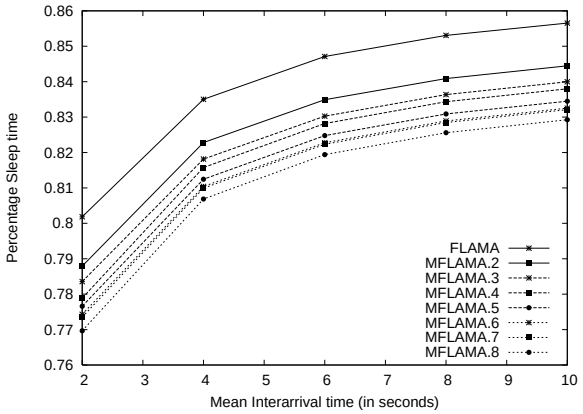


Figure 5.13: Percentage sleep time

availability of orthogonal communication channels.

Figure 5.13 shows how MFLAMA's energy efficiency compares to FLAMA. We observe that there is a slight decrease in percentage sleep time as the number of communication channels increase. When more channels are available for communication, nodes spend more time listening to the medium. This is due to the simple election algorithm employed in the MFLAMA approach that forces a node to listen to its child, if it has the highest one-hop priority without channel or transmit conflicts. As the offered load increases, nodes also spend more time on transmitting frames due to the increase in channel access probability due to multiple channels.

5.6 Conclusion

This chapter introduced an energy-efficient, application-aware medium access control protocol specifically designed with sensor network applications in mind. The proposed protocol is named FLAMA for FLow-Aware Medium Access and uses application-specific traffic information to adapt to the sensor network scenario at hand. Using traffic information, FLAMA is able to establish transmission schedules as well as determine which nodes should be in transmit or receive mode, or can switch their radios to low-power sleep state. This feature is instrumental in achieving energy efficiency without compromising the simplicity of the protocol.

The performance of FLAMA was evaluated using simulations and testbed experi-

mentations and demonstrated the importance of application-awareness in medium access. Simulation results indicate that FLAMA outperforms TRAMA and S-MAC in terms of reliability, and energy savings. FLAMA achieves significant improvement in delay performance for scheduling-based protocols. Our test-bed experiments showcase FLAMA's deployment on MICA2 Motes. They also show that FLAMA can achieve better end-to-end reliability with significant energy savings when compared to a contention-based protocol such as S-MAC.

This chapter also introduced MFLAMA, a multi-channel extension for the FLAMA protocol. MFLAMA improves channel utilization and queueing delay by scheduling transmissions across multiple channels, while maintaining energy efficiency. MFLAMA's performance is evaluated by simulations using data gathering scenarios, an important sensor network application domain. Our results indicate that an increase in the number of orthogonal channels results in a significant improvement in channel utilization and queueing delay. However, the benefits of using multiple channels are limited by the topology and traffic flow patterns. In the specific scenarios we used in our simulations, the number of channels threshold beyond which performance improvements start to decrease is two channels.

Chapter 6

Framework for Energy-aware Channel Access

In both TRAMA and FLAMA, transmission schedules are established by electing the highest priority node as the transmitter. The intended receivers for the schedule are decided based on the traffic schedule announced (TRAMA) or the pre-established forwarding node (FLAMA). However, the traffic characteristics of the application are best modelled using directed flows. Each flow can be quantified based on its arrival rate, and its relationships with other incoming flows as explained in the previous chapter. This motivates the need for a transmission scheduling approach that is flow-based rather than node-based.

In both TRAMA and FLAMA, the topology information is gathered during the random-access period by exchanging signaling packets. The signaling packet exchange is based on contention-based channel access and is prone to collisions due to hidden-terminals. As the topology information is critical to establish collision-free transmission schedules, the random-access period should be long enough to accommodate signaling packet retransmissions. During the random-access period all the data packet arrivals from the higher layer are queued and all the nodes should have their radio in transmit or receive state. Hence, a longer duration of the random-access period leads to a proportional increase in the power consumption and buffer requirements. The frequency of the random-access period directly impacts the amount of time needed for the network to re-configure whenever there is a topology change.

This motivates the need for an improved topology discovery mechanism that: (1) facilitates collision-free signaling exchange, (2) reduces power consumption and buffer size requirements, and (3) allows for quick re-configuration.

In FLAMA, the transmission schedules are established based on the traffic flow in-

formation obtained during the random-access period. This eliminates the overhead due to explicit traffic schedule announcements and improves the channel utilization. However, the traffic characterization mechanism used in FLAMA i.e. data forwarding tree establishment, is very dependent on the data gathering application.

This motivates the need for a flexible scheduled-access MAC framework that can be used to characterize any application-specific traffic patterns, as a set of directed flows, and exchange this information periodically to keep track of the changing traffic patterns.

In this chapter, we introduce DYNAMMA, a DYNAmic Multi-channel Medium Access framework whose goal is to provide a single framework for energy-efficient, traffic-adaptive, scheduled-based multi-channel medium access in high data-rate MANETs.

DYNAMMA provides a flexible, collision-free signaling structure for gathering topology and flow information, and employs a flow-based distributed scheduling algorithm to establish collision-free transmissions across multiple channels.

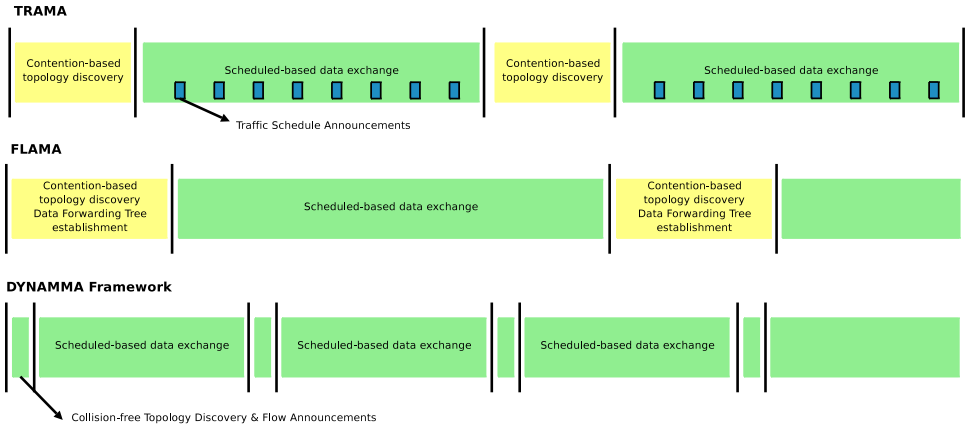


Figure 6.1: Time structure organization

Figure 6.1 illustrates the time structure organization of DYNAMMA when compared to TRAMA and FLAMA. DYNAMMA employs a collision-free, time synchronized channel access mechanism and eliminates the need for a large random-access period for signaling and neighbor/traffic discovery. Hence, the amount of time spent in exchanging signaling information is significantly reduced when compared to TRAMA or FLAMA. This reduces the data buffering requirements and reduces packet losses due to buffer overflows, which is the major source of packet loss in collision-free scheduling approaches. Also, shorter signaling durations

enables more frequent exchange of signaling packets leading to an efficient tracking of traffic and topology changes.

DYNAMMA employs a simple flow characterization model to establish and maintain traffic flow information and can be extended to support advanced traffic prediction models. The flow characterization model enables DYNAMMA to dynamically adapt to the application-specific traffic patterns.

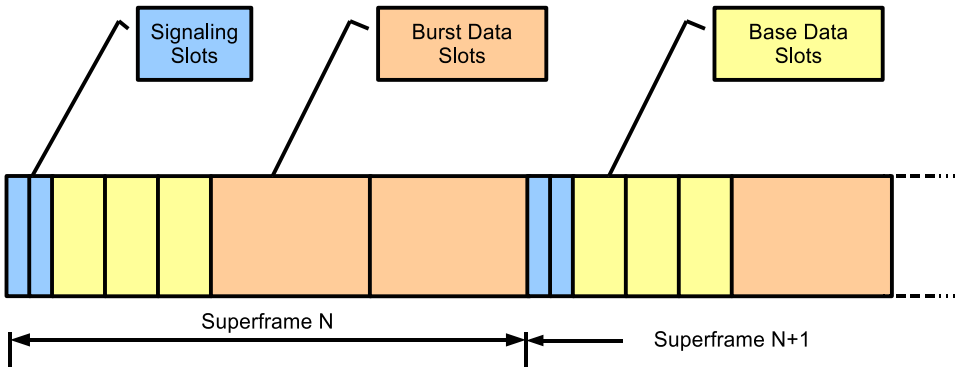


Figure 6.2: Time-slot organization of DYNAMMA

We evaluate the performance of DYNAMMA by extensive simulations for different application scenarios. We also evaluate the performance of DYNAMMA through test-bed experiments using UWB radios.

The results from our simulation study shows that DYNAMMA achieves significantly lesser queueing delay than TRAMA and provides high channel utilization and energy savings when compared to TRAMA and 802.11.

The remainder of the chapter is organized as follows. Section 6.1 describes the DYNAMMA framework, Section 6.2 describes the MAC development platform using UWB radios, Section 6.3.1 presents the simulation results, Section 6.3.3 presents the test-bed results and Section 6.4 concludes the paper with directions for future work.

6.1 DYNAMMA

We summarize the notations used in the description of the DYNAMMA framework in Table 6.1.

Table 6.1: DYNAMMA Notations

Number of channels, M	Total number of channel available.
One-hop Neighbors, $\mathbf{N1}(u)$	Set of neighbors of node *u* that are one-hop away.
Two-hop Neighbors, $\mathbf{N2}(u)$	Set of neighbors of node *u* which are two-hops away.
Active Flow Set, $\mathbf{AF}(u,t)$	Set of all flows that are active in the two-hop neighborhood of node *u* in timeslot *t*.
Required Access Slots, $S_r(f,n)$	Required number of access slots in superframe *n* for the flow f
Expected Access Slots, $E_r(f,n)$	Expected number of channel access slots in superframe *n* for the flow f
Channel Utilization Factor, $U(f,n)$	Channel utilization factor in superframe *n* for the flow f
Channel Utilization Threshold, TH_p	Channel utilization threshold for the flow class *p*

DYNAMMA's time-slot organization is illustrated in Figure 6.2. Time is divided into equally sized time units called *superframes*. DYNAMMA's superframe concept is similar to that of IEEE 802.15.3 [23] and WiMedia MAC [56].

Every superframe consists of a fixed number of time slots. DYNAMMA's time slots fall into three different categories, namely: *signaling slots, base data slots* and *burst data slots*. *Signaling slots* are used for neighbor/traffic information exchange, while *base data slots* and *burst data slots* are used for data exchange. The channel used for communication is dynamically assigned for every base– or burst data slot.

The duration of *base data slots* and *burst data slots* are fixed based on: the physical layer transmission rate, data packet size, number of data packets to be transmitted within the burst, channel switching time, and radio turn-on time. The duration of a *signaling slot* is based on the maximum signaling frame duration. The proposed superframe structure provides ample support and flexibility for neighbor discovery, traffic adaptation, and dynamic radio mode control to enable system-level energy optimizations.

6.1.1 Signaling slot assignment

Every node in a two-hop neighborhood is assigned its own signaling slot for collision-free signaling information exchange. DYNAMMA's signaling slot assignment is similar to the

beaconing slot assignment in the WiMedia MAC [56].[1] A certain number of free signaling slots, called the *Announce Period*, are maintained for new nodes joining the network. The *Announce Period* is a parameter of the protocol that can be adjusted based on network topology dynamics.

Whenever a new node joins the network, it listens for a certain number of superframes to determine the network's current state, i.e., the start of the superframe, location of the signaling slots, the *Announce Period*, and the signaling slot assignments in the neighborhood. A node randomly selects a free slot in the superframe for signaling and announces it using the *Announce Period*. Signaling announcements allow for dynamic expansion of the signaling period based on the two-hop neighborhood size.

If multiple nodes join the network at the same time, more than one can choose the same signaling slot and can lead to signaling packet collisions. However, a node can determine signaling packets collisions based on the signaling packets transmitted by its one-hop neighbors and move to a different signaling slot to resolve the conflict.

During the initial join period, if the node did not find any signaling transmissions, it can start a superframe structure by selecting the start of the superframe, and the location of the signaling slots. The node starts sending signaling packets in the selected signaling slot periodically. If two nodes start at the same time and start sending signaling packets at the same time, they may not discover each other. To prevent this, nodes are required to skip sending signaling packets periodically and listen for any activity during its signaling period.

If a signaling packet is not received from a neighboring node for a certain number of superframes, the node is considered to be inactive and is removed from the neighbor list. Time synchronization across the two-hop neighborhood is achieved using accurate timestamps provided by the PHY. Various known techniques (e.g., [15, 16]) can be used for time synchronization.

Signaling frames are transmitted on a well-known channel every superframe. DYNAMMA's current framework can be easily extended to support dynamic hopping of the channels used for signaling exchange. For example, the signaling channel for a particular superframe can be based on a pseudo-random function of the superframe identification number, or *superframeId*.

The total duration of the superframe used for signaling is directly proportional to the

[1] However, unlike WiMedia, there is no restriction on the position of the signaling slots.

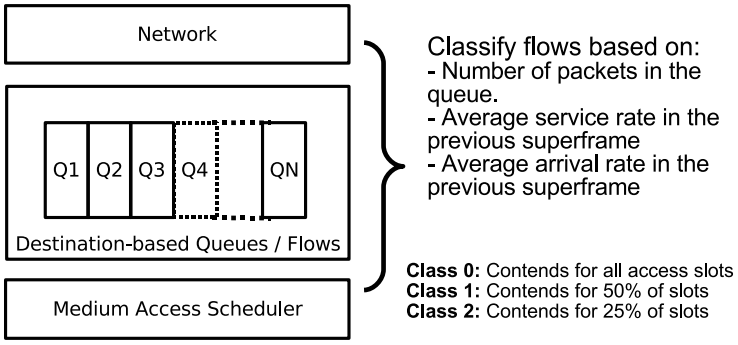


Figure 6.3: Traffic Classification

size of the two-hop neighborhood and duration of the *Announce Period*. This duration is much smaller when compared to that of TRAMA or FLAMA.

6.1.2 Traffic and Neighbor Discovery

All nodes are required to be active during signaling slots to gather neighbor– and traffic information. The following signaling information are encoded into the signaling packet: (1) the superframe identifier, or *superframeId*, (2) location of the signaling slot within the superframe, (3) one-hop neighborhood–, and traffic information.

Traffic information is modeled as a set of one-hop flows [41] directed to– or originating from the node. A flow is nothing but a stream of packets originating from a node destined to one of its one-hop neighbor(s) at the link layer. A flow, which can be unicast, multicast, or broadcast, is characterized by its originating node (or transmitter), destination node(s) and a unique flow identifier.

Flow information is gathered at each node based on the current traffic characteristics and is propagated every superframe using the signaling packets. This provides a flexible mechanism to adapt channel access based on the current traffic information.

The number of channel access slots required to service a flow is dependent on its arrival rate and its service rate. The flow arrival rates and service rates are dependent on the application traffic characteristics and the network topology. To improve the channel utilization, channel access slots should not be allocated to a flow that does not have any data to send.

Accordingly, traffic flows are classified into different classes depending on its arrival

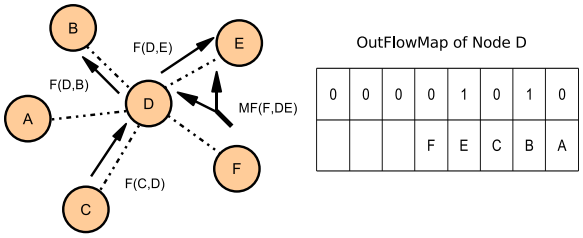


Figure 6.4: Traffic discovery

and service rates. The number of channel access slots that a flow can contend is decided probabilistically based on its class identifier. In the current implementation we use three flow classes, with class identifiers ranging from 0-2. Class 0 flows are the high-traffic flows and they contend for all the channel access slots in the superframe. Class 1 and class 2 flows are flows with reduced traffic and on an average they contend for one-half and one-quarter of the superframe, respectively.

All nodes maintain a set of destination-based queues (corresponding to outgoing flows), as shown in Figure 6.3. Flow classes are assigned based on the number of packets in queue, the average service rate in the previous superframe, and the average arrival rate in the previous superframe. The channel access probability of a flow (f) can be approximated as $1/NumberOfContendingFlows$. The expected access slots for the flow, $E_r(f)$ is computed as the product of the channel access probability of the flow and the number of slots in the superframe. The required access slots for the flow, $S_r(f)$ is computed based on the current MAC layer queue parameters. Using the expected access slots and the required access slots, the fractional usage, $U(f) = S_r(f)/E_r(f)$, is computed for all the outgoing flows. The flow classes are then assigned based on a threshold on the A flow belongs to class p, if $U(f) > TH_p$, where p is the smallest integer for which the inequality holds. For the current implementation, the class thresholds are fixed at $TH0 = 0.95$, $TH1 = 0.65$, and $TH2 = 0$.

Flow information is encoded in a *flow bitmap* format to reduce overhead in flow announcement. The position of the bit in the bitmap is used as the *flow identifier* and the bit is set to 1 to indicate that the flow exists. The originating node identifier and the *flow identifier* is used to uniquely identify a flow. The destinations for the flow are determined based on the *flow identifier* and the ordering of the announced one-hop neighbor list.

The most significant bit of the bitmap is reserved and is used to indicate a broadcast

flow. Multicast destination identifiers are sent as an extension of the one-hop neighbor list and the corresponding bit positions are used for announcing multicast flows.

The bit-width of the *flow bitmap* determines the maximum number of flows that can be announced by a node. Figure 6.4 illustrates a simple out-flow bitmap and the corresponding node ordering for a node with two outgoing flows.

Nodes announce both their outgoing- and incoming (originating from a one-hop neighbor) flows. Additionally, nodes also announce all active outgoing flow identifiers of their two-hop neighbors (encoded as a bitmap). This provides all the information required to uniquely identify a two-hop originating flow and is required to avoid hidden terminal collisions.

Figure 6.5 illustrates the format of a signaling packet showing its fields and the number of octets used for each field.

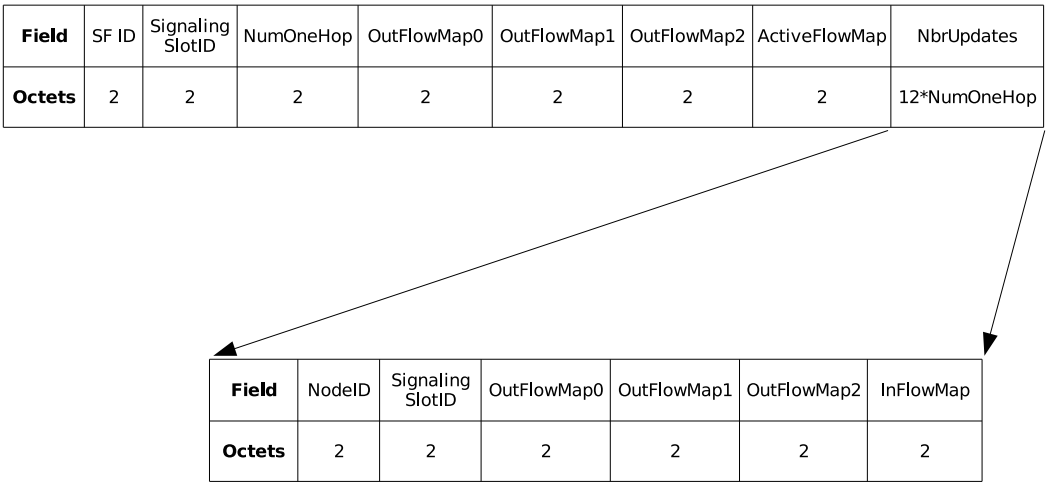


Figure 6.5: Signaling packet format

6.1.3 Distributed Scheduling Algorithm

The flow and the neighborhood information gathered using the signaling packet exchange are used by the distributed scheduling algorithm for establishing collision-free transmission schedules for the base and burst data slots.

Whenever a new flow is added to the announcement, a node should ensure that the flow information is propagated to the two-hop neighborhood before activating the flow for distributed election and this can take up to two superframes.

At the start of the base data slot or burst data slot (say t in superframe n), every node executes the election algorithm to determine its state as transmitter, receiver, or sleeping by electing flows from the set of contending flows. The transmission channel is determined using a pseudo-random function (PRF). The algorithm ensures that the receivers of the elected flows are listening on the particular channel decided by the transmitter.

The steps involved in the election process at node u are described below:

- Gather all active contending flows $AF(u,t)$ for the current timeslot t. This includes all the outgoing flows of node u, all the outgoing flows of $N1(u)$, and all the outgoing flows of $N2(u)$ that are currently active. Class 0 flows are active for any timeslot t. For class 1 and class 2 flows, a random number is generated using a pseudo-random function $PRF(flow.srcId,t)$, which is used to decide if the flow contends in the current slot or not.

- Flow priorities are computed as $PRF(flow.srcId, flow.flowId,t,n)$ and the transmission channel for the flow is computed using $PRF(flow.srcId)\%M$.

- Flows are examined starting from the highest priority flow from the set $AF(u,t)$. The flow can be scheduled for communication if the transmission channel and the initiator/destination(s) are not allocated to a higher priority flow. If the node is the intended originator or destination of a flow that cannot be scheduled due to a conflict, the node can set the state to sleep mode. Additionally, the node elected as a receiver may enter sleep mode if no transmission starts within a preset idle time.

The pseudo code of the distributed election algorithm used in DYNAMMA is presented in Figure 6.8. The algorithm ensures that when a flow $F(X,Y)$ is activated in channel n, then: (1) any node that is one-hop to the node X cannot receive from in channel n to any other flows, (2) any node that is one-hop to the node Y cannot transmit in channel n, and (3) there is only one transmitter in channel n in the two-hop neighborhood.

As every node has limited information on the flow / topology, the set of inputs for the algorithm running on different nodes are different. Hence, there can be some inconsistency in the decision made by each node on a particular flow. It is important to ensure that these inconsistencies do not affect the correctness of the algorithm.

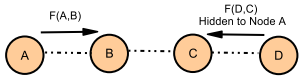


Figure 6.6: Hidden flow problem

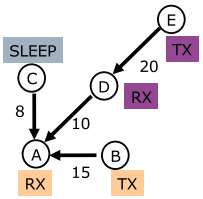


Figure 6.7: Example: Transmission schedule

For example, consider a linear topology shown in Figure 6.6. Assume that the flows $F(D,C)$ and $F(A,B)$ are allocated to the same channel m. In this case, node B knows about the flow $F(D,C)$ and marks the channel m as used by $F(D,C)$. However, the same channel could be re-used by flow $F(A,B)$ and node A is unaware of flow $F(D,C)$ and will initiate a transmission. If node B enters the sleep state assuming that there is a channel conflict, it will miss the transmission from node A. Fortunately, these conditions can be identified during the election process and the correctness can be ensured by trading off the efficiency of the election process.

This problem is resolved by forcing a node to receive mode if there is channel conflict due to a previously elected transmitter, which is two-hops away and hidden from the current one-hop transmitter.

Figure 6.7 illustrates the transmission schedule establishment for a particular time slot. Flows are indicated using directed arrows and the numbers next to the flows are the flow priorities computed for the current time slot. The highest priority flow $F(E,D)$ is elected first and is a transmit channel based on a PRF. The next higher priority flow $F(B,A)$ can be activated as long as the assigned channel is different from the channel assigned to $F(E,D)$. The flow $F(C,A)$ cannot be activated as the destination of the flow is already activated.

```
1  Compute AF(u,t) and sort AF(u,t) based on descending order of flow priorities.
2  Initialize BlackListNodes = ∅; UsedChannelList = ∅; u.state = UNKNOWN;
3  foreach (flow ∈ AF(u,t)) begin
4    if (flow.srcId == u) then : Outgoing flow
5      if (TXCHANNEL(u) ∋ UsedChannelList && flow.destId ∋ BlackListNodes) begin
6          let u.state = TX; u.txchan = TXCHANNEL(u); u.txflow = flow;
7      else let u.state = SLEEP;
8      endif
9    else if (flow.destId == u || flow.destId == ANY_DEST) then : Incoming flow
10     if ( (TXCHANNEL(flow.srcId) ∋ UsedChannelList) OR
       (CONFLICTTX hiddenfrom flow.srcId)) then
11         if (flow.destId ∋ BlackListNodes) then
12             let u.state = RX;u.rxchan = TXCHANNEL(flow.srcId); u.rxflow = flow;
13         else u.state = SLEEP; endif
14     else u.state = SLEEP; endif
15   else if (flow.srcId ∈ N1(u)) then : One-hop Originated Flow
16     let UsedChannelList = {UsedChannelList, TXCHANNEL(flow.srcId)};
17     let BlackListNodes = {BlackListNodes, flow.srcId, flow.destId};
18   else : Two-hop or Three-hop Originated Flow
19     let UsedChannelList = {UsedChannelList, TXCHANNEL(flow.srcId)};
20     let BlackListNodes = {BlackListNodes, flow.destId};
21     if (flow.srcId ∋ {N1(u), N2(u)}) then set hidden usage flag; endif
22   endif
23 if (u.state == UNKNOWN) then continue ; else break ;
24 end
```

Figure 6.8: DYNAMMA election algorithm pseudo-code

6.1.4 Correctness

In this section we establish that: DYNAMMA transmission schedules are collision-free, and DYNAMMA transmission schedules ensure that nodes do not transmit to a sleeping node or to a node listening on another channel.

For collision freedom, it is enough to ensure that two nodes in the in the two-hop neighborhood does not transmit during the same timeslot in the same channel. If two flows are contending for the same channel in the two-hop neighborhood, the flow with the highest priority is elected as the transmitter. This is ensured by steps 19 and 5 in the pseudo-code description of the algorithm. As there is only one transmitter in a particular channel in the two-hop neighborhood, transmission schedules established by DYNAMMA are collision-free.

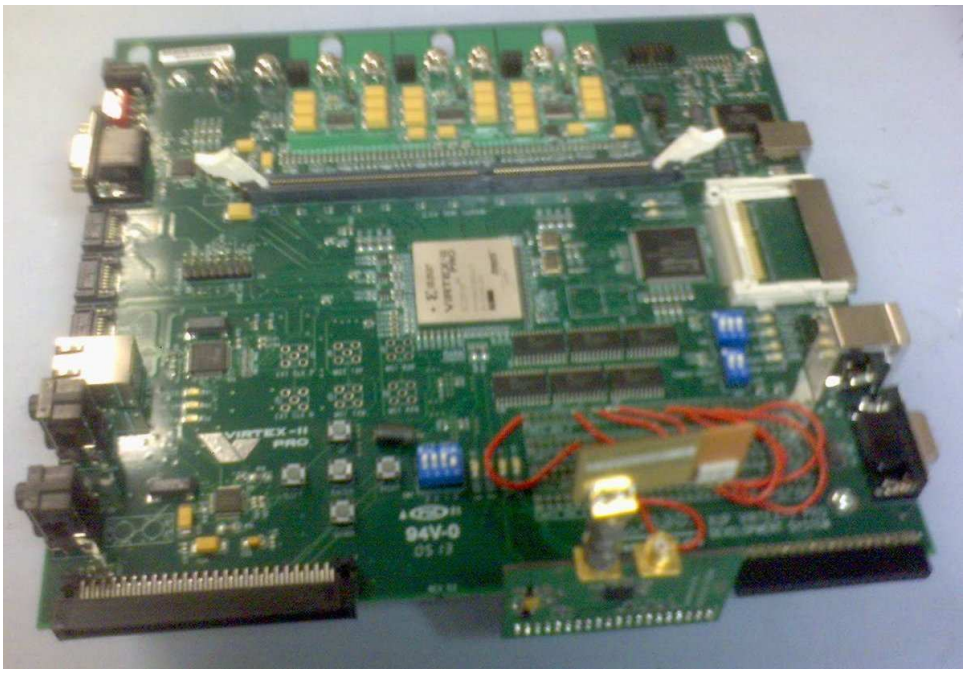


Figure 6.9: Xilinx FPGA Evaulation Board

In the election process, receiver always listens to the highest priority flow that does not have any channel or transmitter conflicts. A lower priority flow involving this receiver (in the same or different channel), will not not be elected as the receiver (and/or channel) will be black-listed by steps 15 to 21 of the pseudo-code description of the algorithm. Hence, the receiver always listens to the exact flow that can be elected for communication. Note that the flow that the receiver is listening may not be chosen for activated by the transmitter due to a conflict that is hidden from the receiver. This does not affect the correctness of the algorithm and only impact the channel utilization.

A receiver enters sleep mode only if there are no incoming flows or the highest priority incoming flow has a channel or transmitter conflict that is not hidden from the transmitter. This prevents transmissions to a sleeping node.

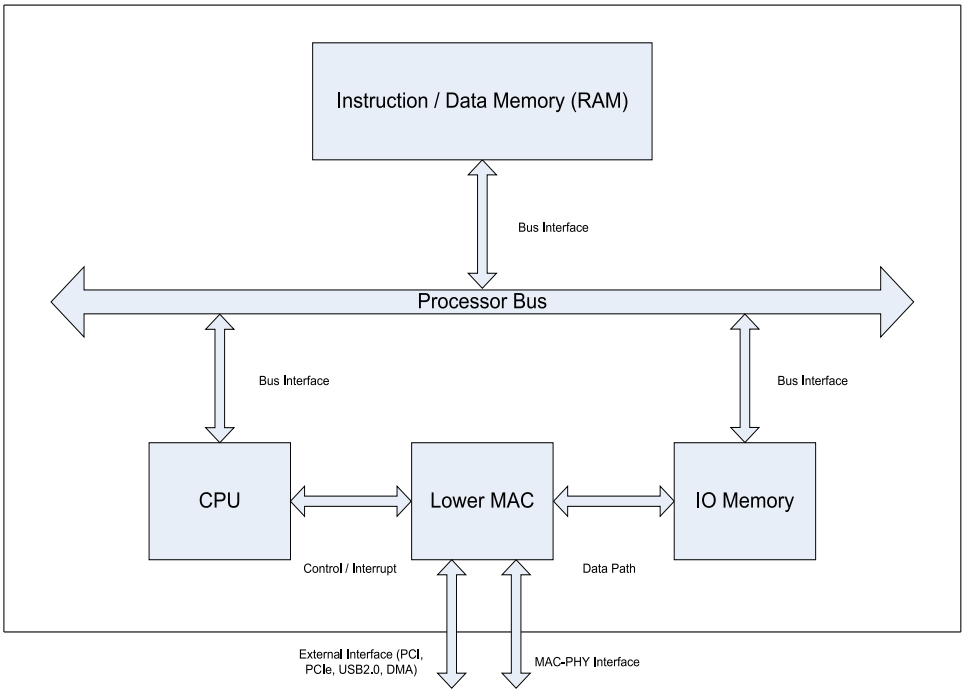


Figure 6.10: Generic MAC Development Architecture

6.2 DYNAMMA Implementation

In this section we describe our FPGA-based test-bed developed for evaluating the scheduled-based protocols on a UWB physical layer. Wide-spread availability of FPGAs with an embedded processor as a hard macro, simplifies the implementation and testing of MAC protocols in real-world. In our setup, we used a Xilinx evaluation board with a Virtex-II pro FPGA. The FPGA has a PowerPC(PPC405) hard macro that can be clocked up to 350MHz and the board supports several customizable interfaces.

The platform uses the UWB radio (RTU7010) daughter board developed by Realtek Semiconductors as shown in Figure 6.9. The radio implements the WiMedia physical layer standard [56] and supports data rates from 53.3$Mbps$ upto 480$Mbps$. The radio is controlled through the MAC-PHY interface (MPI) specified by the WiMedia [58].

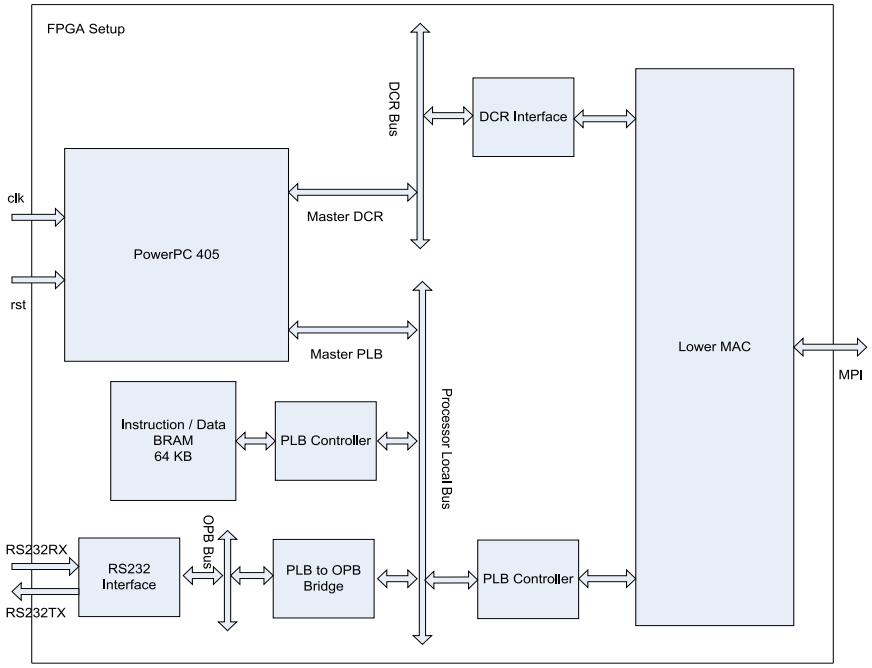


Figure 6.11: PowerPC-based MAC Architecture

6.2.1 Hardware Architecture

Figure 6.10 presents a generic processor-based MAC development architecture implemented in our hardware. In this architecture the MAC functions are partitioned into hardware and firmware components. The hardware component implements all the time critical functions such as radio mode control, transmit / receive scheduling, and transmit / receive DMA. The firmware component implements all the protocol functions, signaling, queue management, and the flow-based distributed scheduling. The hardware also provides accurate timestamps for packet transmission and reception using a 66MHz oscillator.

We instantiate a processor-based system using the Xilinx embedded development kit as shown in Figure 6.11. The system consists of a PowerPC processor, instruction and data memory for MAC firmware, IO memory for storing incoming / outgoing packets, RS232 UART interface for debugging and a lower MAC IP developed at Realtek. The lower MAC IP implements all the hardware MAC components and is controlled using the device control register (DCR) interface. The processor is clocked at 264MHz and the system is designed to handle very high data throughput to support 480$Mbps$ physical layer data rate.

The lower MAC IP implements the following functions:

- A programmable timer to generate periodic timer interrupts and superframe start interrupts.
- Hardware scheduler to transmit signaling packets periodically at a particular time slot.
- Hardware scheduler to queue radio mode control commands: transmit, receive and sleep.

The hardware scheduler is responsible for switching the radio mode at appropriate times with resolutions upto 15*ns*. The initial join sequence is implemented as discussed in Section 6.1.1. As the physical layer data rates are high, time synchronization in the order of *us* is required to maintain synchronization with the neighbors. The clock drift that can be tolerated over a superframe is in the order of *us*. To achieve this, we use a 66MHz crystal with less than 10ppm offset.

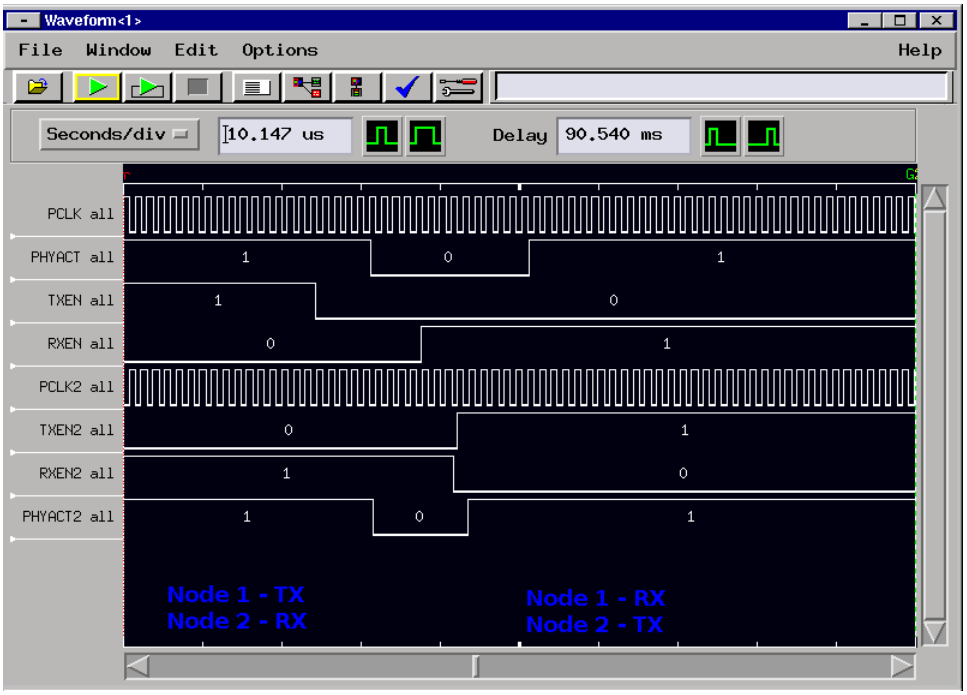


Figure 6.12: Transmit to receive turnarounds

Time synchronization is performed every superframe based on the timestamp of the received signaling information from the neighbors. UWB radios are short range and hence, the

propagation delay is negligible when compared to the required synchronization resolution. This facilitates the use of one way timestamps for time synchronization. A node always synchronizes to the slowest neighbor in the one-hop neighborhood.

The difference between the actual signaling packet arrival timestamp and the expected signaling packet arrival timestamp is used to determine the clock offset of a neighbor. If the difference is positive, then the neighbors clock is slower than this nodes clock. If this is the slowest node in the neighborhood the node delays its superframe by this offset.

Signaling packets with flow and neighbor information is prepared at the start of the superframe inside the superframe start interrupt handler. Once the node establishes the superframe structure, the periodic time slot interrupt timer is activated. During the time slot interrupt, the distributed scheduling algorithm is executed based on the flow and neighbor information. As this can take some time for processing, we use the result of the algorithm to schedule the radio mode for the next time slot. This allows for a large processing time to establish the transmission schedules.

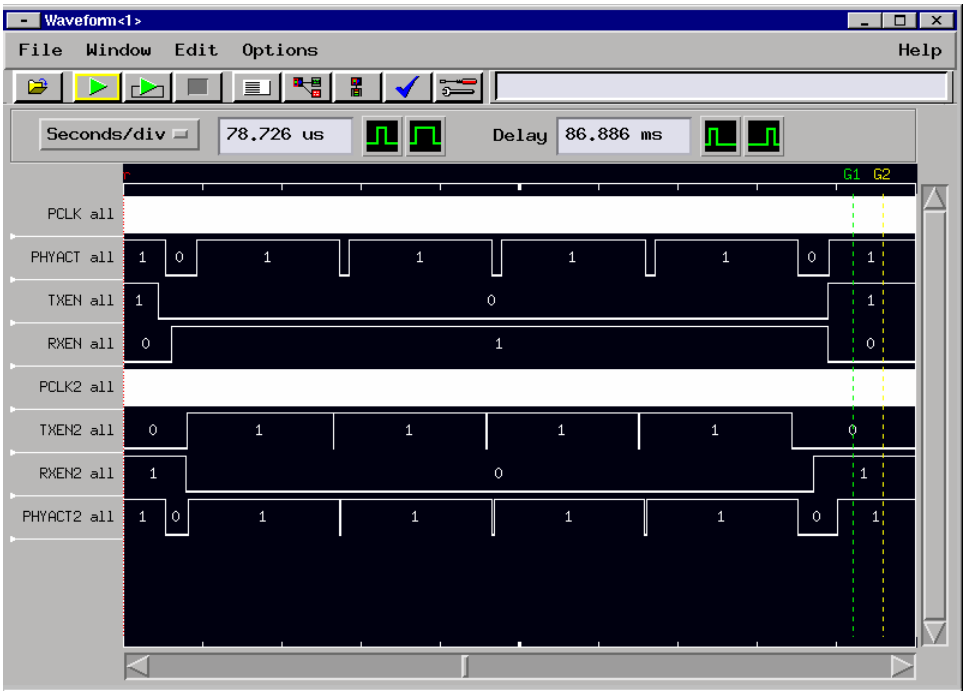


Figure 6.13: Data exchange during burst slots

The transmit and receive timings are verified by monitoring the MPI signals using a

Logic Analyzer (LA). The PHY silicon provides a 66MHz clock reference (PCLK) and all the MPI signals are synchronous to PCLK. The signals TX_EN and RX_EN are used by the MAC to initiate transmissions and receptions respectively. The PHY silicon indicates start of the transmission or reception using the PHY_ACTIVE signal. The start of the over-the-air preamble can be determined using the rising edge of the PHY_ACTIVE. The detailed information on the MPI signals can be found in [58].

Figure 6.12 presents the LA capture of the MPI signals (PCLK, TX_EN, RX_EN and PHY_ACTIVE) illustrating a slot turn-around between two nodes. As we can see, node 1 is the transmitter in slot *n* and is the receiver in the next slot. Node 1 enters the receive mode by asserting the RX_EN signal. Node 2 transitions from receive to transmit and it asserts the signal TX_EN2 to enter into transmission mode. We allow a short guard interval between the slot transitions to ensure that the receiver packet is received properly.

The RTU7010 radio supports the burst mode operation as defined in the UWB physical layer standard [56]. In this mode, packets are transmitted as a burst with minimal inter-frame spacing (MIFS). The MIFS duration 1.875us between the packets is maintained by the PHY. Figure 6.13 shows a set of burst mode data exchange between nodes at physical layer data rate of 200$Mbps$.

6.3 Performance Evaluation

We evaluate the performance of DYNAMMA by both simulations and test-bed experiments.

6.3.1 Simulation Setup

We compare the performance of DYNAMMA against both contention- and scheduling-based MACs. We use IEEE 802.11 DCF [24] and TRAMA [42] as representatives of contention-based and scheduling-based protocols, respectively. Qualnet [60] is used as the simulation platform. The radio model employed is based on the WiMedia physical layer specification [56] for UWB networks. We implemented the radio model based on BER lookup tables from Matlab simulations. The UWB physical layer is designed for short range, and high data-rate applications. The physical layer supports different data rates ranging from 53.3$Mbps$ to 480$Mbps$ and

seven channels (out of which only three are orthogonal). The radio range depends on the data rate and in our simulation we use the base rate of 53.3$Mbps$ with a radio range of about 20m.

We also evaluate the performance of DYNAMMA using the UWB MAC development test-bed. A prototype implementation of the DYNAMMA framework based on the UWB MAC development platform described in Section 6.2 is used for the test-bed experiments.

6.3.1.1 Traffic Generation

We consider two different traffic scenarios to illustrate DYNAMMA's application-awareness, i.e., its ability to adapt to different application traffic patterns. In the first scenario, node traffic is statistically generated based on exponentially distributed packet arrivals. For our experiments, we vary arrival rates generating more or less traffic. We chose this traffic pattern as a way to stress-test protocol performance given that a node has flows to all its neighbors and thus contention for the channel is high (especially for flow-based election protocols like DYNAMMA). The second traffic scenario is based on a data gathering application in which all nodes periodically send data to a sink node. For this traffic pattern, data follows a reverse tree (from the leaves to the root) and thus exhibits less contention.

UWB radios are short-range with very tight SNR requirements when compared to a standard 802.11 radios. With short-range radios, the deployments are often hierarchical with a small group of nodes and there exists a long-range backbone to provide connectivity across the groups. Hence, a topology size of 16 nodes with multiple hops is used for all the simulation experiments. To ensure connectivity between nodes, a square grid placement with 18m separation is followed. The data forwarding tree is hard-coded with a static route for data gathering application.

6.3.1.2 Protocol Parameters

For our simulation experiments, we set DYNAMMA's parameters as follows. The duration of the *base data slot* is based on the duration of the maximum PHY payload size of 4095 bytes and SIFS (10us). The *burst data slot* is set based on a burst size of 2 transmitted with short preamble. Thus, the base and burst data slots are set to 638.125us and 1268.125us, respectively. In our simulation experiments we followed a static assignment of signaling slots for the ease of implementation. The signaling slots are also grouped together in the superframe

and each base slot into divided in to 16 slots. Hence, the signaling overhead is is one base slot per superframe. This can be reduced by using a dynamic signaling scheme, where slots are assigned dynamically based on the network size. The superframe consists of 16 base slots, 238 burst slots, and 16 signaling slots. We vary the the number of channels available for use in DYNAMMA from 1 to 3 in order to quantify the effect of multi-channel scheduling.

TRAMA's parameters are optimized to fit the high data rate physical layer. TRAMA's $SCHEDULE\_INTERVAL$ is set to be 100 transmission slots. The maximum size of a signaling packet is fixed at 96 bytes which results in a slot period of $28.25us$ with guard time to take care of switching. Transmission slots are fixed to support a maximum data fragment size 4095 bytes which results in a slot period of $630.75us$. The random access period is fixed to 10000 signaling slots ($0.2825s$) and is repeated once every 10000 transmission slots ($6.3075s$). TRAMA incurs overhead due to random access period every $6.3075s$ and no data communication takes place during this interval. This can lead to increased queueing drops in TRAMA during random access period.

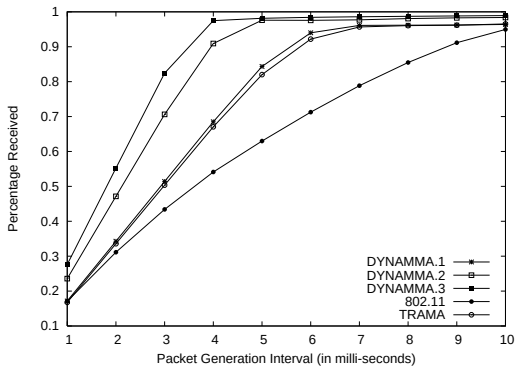
The following metrics are used to assess the performance of the protocols:

- **Average Packet Delivery Ratio** is the ratio of number of packets received to the number of packets sent,
- **Average Queuing Delay** is computed as the average per-hop latency for the network, and
- **Percentage Sleep Time** is the ratio between the time spent in low-power sleep mode to the total experiment run time. It provides a way to evaluate the protocol's energy efficiency.

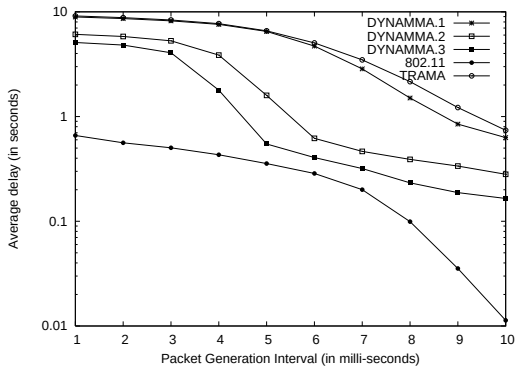
6.3.2 Simulation Results

6.3.2.1 Synthetic Traffic

In this scenario, all nodes generate unicast traffic to a randomly selected next-hop node. The data generation interval is varied from $1ms$ to $12ms$ and the results are averaged over several runs. Figure 6.14(a) shows the average packet delivery ratio at each node and Figure 6.14(b) shows the average per-hop queuing delay. The main observation here is that the major source of packet loss in scheduled-access MAC protocols is the packet drops due to buffer overflows. While, the major source of packet loss in 802.11 are hidden terminal collisions.

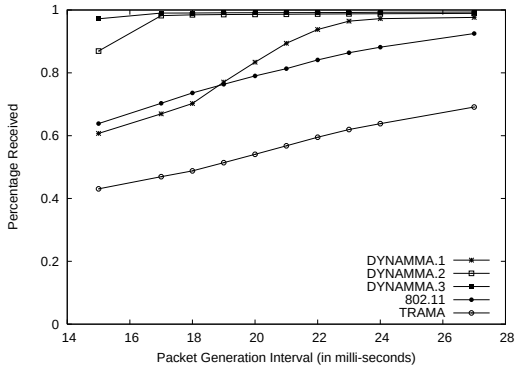


(a) Average Delivery Ratio

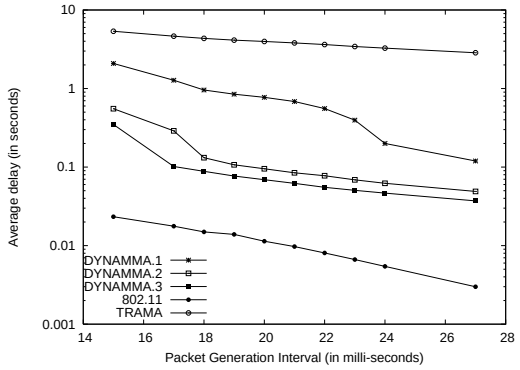


(b) Average Queueing Delay

Figure 6.14: Synthetic Traffic



(a) Average Delivery Ratio



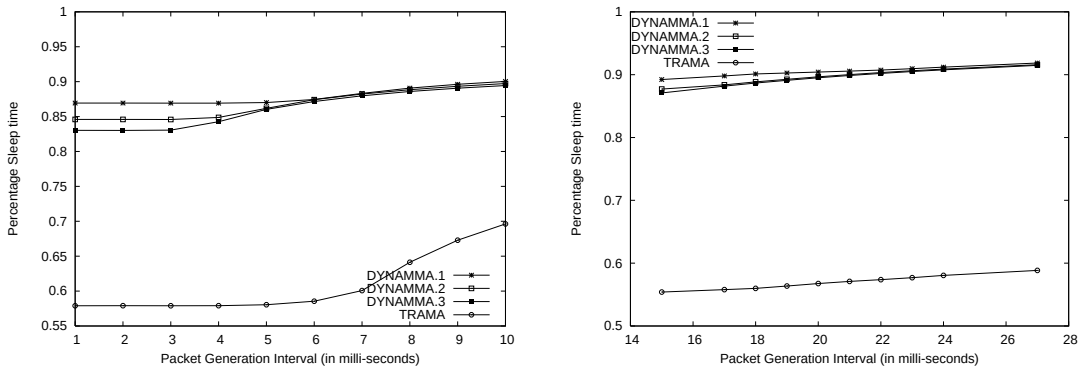
(b) Average Queueing Delay

Figure 6.15: Data Gathering

Hence, the packet delivery ratio decreases with increase in offered load for all the protocols. Due to the collision-freedom in transmission schedules both TRAMA and DYNAMMA exhibits higher delivery ratio than the 802.11.

DYNAMMA with single channel (DYNAMMA-1) achieves better queueing delay when compared to TRAMA and this leads to an improved delivery ratio. However, the queueing delay of DYNAMMA is higher than that of the 802.11, which is inherent to scheduled-access MAC protocols.

As we increase the number of channels (DYNAMMA-2 and DYNAMMA-3 for 2– and 3 channels, respectively), there is an improvement in the delivery ratio and queuing delay,



(a) Synthetic Traffic (b) Data Gathering

Figure 6.16: Energy Savings

more pronounced when the offered traffic is high. However, we noticed that as we increase the number of channels, the efficiency of the scheduling algorithm decreases due to transmitter/receiver conflicts i.e. flows cannot be scheduled as the transmitter or receiver is already a part of another flow in a different channel. However, due to the limited topology and flow information, there is no guarantee that the neighboring flow will be activated. Hence, there is a potential for wasted channel access slots in which none of the flows are active.

Figure 6.16(a) shows the percentage of time spent by nodes in sleep mode. DYNAMMA implements idle receive timeouts i.e. the receiver is turned off if the transmission does not start within a timeout. This leads to an increased energy savings in DYNAMMA when compared to TRAMA. We also notice that the energy savings decreases as we increase the number of channels. This is due to the fact that nodes spend more time transmitting or listening with more channel available for communication.

6.3.2.2 Data Gathering Application

In this scenario, we place a data gathering sink node in the corner of the grid. All the nodes generate traffic that is routed to the sink using a data forwarding tree. The routing table for data forwarding is hard-coded for this scenario. The goal of this experiment is to analyze the performance of DYNAMMA when there is regular flow pattern. The results highlight the application adaptiveness of DYNAMMA when compared to TRAMA.

Figure 6.15(a) illustrates the average delivery ratio at the sink node and Figure 6.15(b)

shows the average per-hop delay. TRAMA suffers heavily in this topology due to the high per-hop queueing delay. Periodic random access periods also affect packet delivery and large number of packets are dropped due to MAC layer queue overflow. This leads to a significant decrease in delivery ratio at the sink. DYNAMMA's inline signaling mechanism and the ability to adapt traffic announcement improve the queueing delay significantly when compared to TRAMA. DYNAMMA also outperforms 802.11 in average delivery ratio due to its collision-free scheduling algorithm.

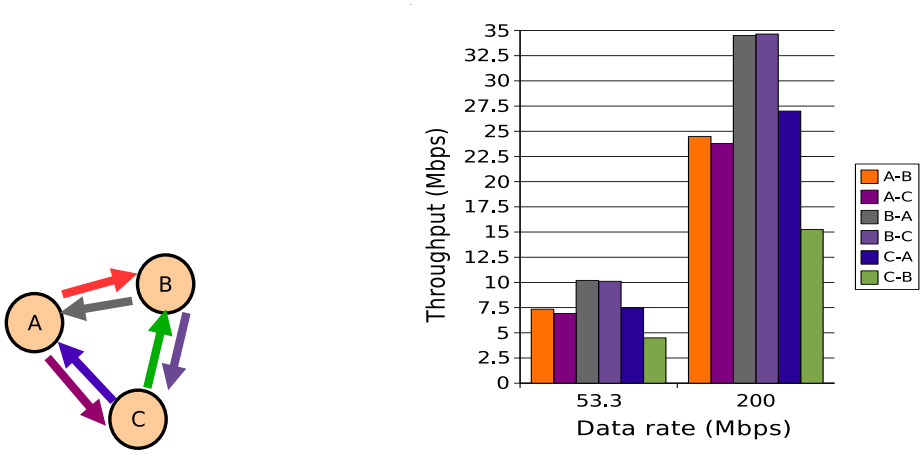
Figure 6.16(b) compares the percentage sleep time of DYNAMMA and TRAMA. DYNAMMA significantly improves the energy savings as the scheduling algorithm ensures that nodes sleep when they are not part of an active flow. As we increase the number of channels, the delivery ratio and queueing delay are improved due to multiple transmission schedules in the two-hop neighborhood on orthogonal channels. As expected, we see a big improvement in the performance as we go from one channel to two channels. However, beyond two channels the improvement is limited due to the transmitter / receiver conflicts.

In single channel scheduling approaches, it is not necessary for the receiver to know the exact transmitter in the one-hop neighborhood. It is enough to ensure that the receiver is listening for transmissions and any node in the one-hop neighborhood of the receiver can transmit. However, in the multi-channel scheduling the channel used for communication depends on the transmitter. Hence, the receiver should exactly know the transmitter in the one-hop. This restriction prevents any arbitrary node that is one-hop to the receiver re-using the medium and leads to some reduced channel re-use.

6.3.3 Test-bed Experiments

The main goal of the test-bed experiment is to provide a proof-of-concept implementation of DYNAMMA framework on a high data-rate physical layer such as UWB. The MAC development platform described in Section 6.2 is used for our test-bed experiments.

The base slot duration is fixed at $644us$ such that the maximum payload size (4095 bytes) can be supported at the base data-rate ($53.3Mbps$). With in the same duration, a burst of transmissions can be supported with higher physical layer data-rate. This eliminates the need for burst data slots and makes the implementation easier. For example with the same base slot duration, a 3400 byte data burst of size 4 can be supported at $200Mbps$. The signaling slot



(a) Topology (b) Per-flow Throughput

Figure 6.17: Test-bed Experiments

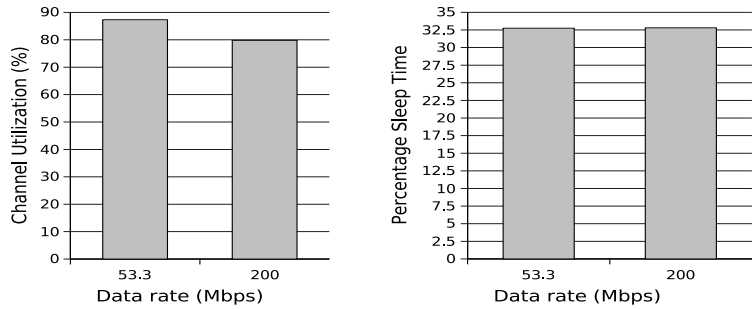


Figure 6.18: Channel utilization and energy savings

duration is 161*us* i.e. one fourth of the base slot. The superframe is made up of 16 signaling slots and 256 base slots, totaling to 167.440*ms*.

A simple topology of three nodes as shown in Figure 6.17(a) is considered and single channel is used for communication. The nodes are joined to the network one by one and a node establish an out-going to all the one-hop neighbors. The data is generated at the MAC layer and the saturated throughput is measured for all the flows. All the nodes stop and report the statistics after 10000 superframes. The experiment is repeated with 53.3*Mbps* and 200*Mbps* physical layer data-rate. For 200*Mbps* data-rate, packets are transmitted as a burst of size 4 during every channel access slot.

Figure 6.17(b) presents the per-flow throughput attained by the flows. As illustrated in Figure 6.18, DYNAMMA achieves very high channel utilization 87.33% and 79.8% for 53.3$Mbps$ and 200$Mbps$ data rates respectively. The channel utilization is slightly lower of 200$Mbps$ data-rate and this is due to the fact that the overhead due the header increases as the data-rate is increased. As expected, nodes on an average sleep for one thirds of the time as illustrated in Figure 6.18.

6.4 Conclusion

In this chapter we introduced DYNAMMA, a framework for DYNAmic Multi-channel Medium Access. DYNAMMA models application behavior using directed flows and schedules collision-free transmissions across multiple channels. DYNAMMA reduces energy consumption by switching the radio to low-power standby node whenever the nodes are intended participants of the flows.

We compared DYNAMMA's performance against TRAMA and 802.11 by extensive simulations for a two different application scenarios. It is evident from the simulation results that significant energy savings (upto 90%) with higher delivery ratio when compared to TRAMA and 802.11. DYNAMMA's multi-channel approach improves the channel utilization while reducing energy consumption. DYNAMMA can adapt to application traffic using minimal signaling exchange in the form of flow bitmaps.

We also presented a prototype implementation of DYNAMMA on a UWB MAC test-bed. Test-bed results indicate that DYNAMMA framework can achieve very high channel utilization (upto 87.33%) with considerable energy savings on a high-date physical layer technology.

Chapter 7

Conclusion and Future Work

The significantly diverse characteristics of wireless environments (when compared to wire-lined networks) coupled with a wide range of application requirements and device capabilities pose significant challenges to the design of efficient wireless communication systems.

This thesis dealt with various aspects of energy-efficient medium access in ad hoc wireless networks. We established the importance of cross-layer optimization in wireless networks and demonstrated its performance benefits using reliable multicast as an example.

We then presented a new approach to energy-efficient medium access that is traffic-adaptive and distributed. TRAMA high-lighted the importance of traffic-adaptiveness and energy-efficiency in the medium access. The scheduling delay and the computational overhead introduced by the traffic adaptiveness of TRAMA led to research in simpler, but efficient scheduling approaches taking advantage of the application behavior.

Flow-aware medium access (FLAMA) is presented for data gathering application in a sensor network. An implementation of FLAMA on TinyOS using the Mica2 motes is presented. The FLAMA approach is extended to support energy-efficient, conflict-free scheduling across multiple channels.

A low overhead, medium access framework for energy-efficient, multi-channel scheduling (DYNAMMA) is presented. A generic MAC development test-bed for evaluating scheduled-access protocols over UWB physical layer is developed and the performance of DYNAMMA is evaluated using the test-bed.

7.1 Future Work

- **Traffic Prediction**: Scheduled-based MAC protocols have several advantages over the contention-based protocols in terms of energy efficiency and channel utilization. However, the delay incurred due to the scheduling is still non-trivial. This necessitates some intelligence at the medium access layer to learn the traffic patterns and adapt the schedules accordingly to minimize the scheduling delay. Our DYNAMMA framework provides a baseline for traffic classification into flows and can be used to implement some adaptive learning algorithms to predict the flows and improve the efficiency of channel access scheduling.

- **Delay Guaranties**: Dynamic scheduling protocols using a pseudo-random function does not guarantee a fixed access time to the medium. However, delay guarantees are important to serve the application level QoS. Another direction of research is to improve the DYANAMMA election algorithm to guarantee a minimum QoS for each of the flows.

Bibliography

[1] T. Aytur, H-C. Kang, R. Mahadevappa, M. Altintas, S. ten Brink, T. Diep, C.-C. Hsu, C.-C. Lee, R.-H. Yan, and B. Razavi. A fully integrated UWB PHY in 0.13um CMOS. In *Solid-State Circuits Conference, Digest of Technical Papers, ISSCC 2006*, Feb 2006.

[2] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *ACM MOBICOM 2004*, 2004.

[3] L. Bao and J. J. Garcia-Luna-Aceves. A new approach to channel access scheduling for ad hoc networks. In *The seventh annual international conference on Mobile computing and networking 2001*, pages 210–221, 2001.

[4] L. Bao and J.J. Garcia-Luna-Aceves. Hybrid channel access scheduling in ad hoc networks. *Proc. IEEE Tenth International Conference on Network Protocols (ICNP)*, November 2002.

[5] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 2 edition, 1992.

[6] J. Broch, D. A. Maltz, D. B. Johnson, Y-C Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, October 1998.

[7] R. Chandra, V. Ramasubramanian, and K. P. Birman. Anonymous Gossip: Improving multicast reliability in mobile ad-hoc networks. *International Conference on Distributed Computing Systems*, pages 275–283, April 2001.

[8] Chipcon Corporation. *CC1000 Single Chip Very Low Power RF Transceiver*, 2004.

[9] I. Chlamtac and A. Farago. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions on Networking*, 2(1):23–29, February 1994.

[10] I. Chlamtac, A. Farago, and H. Zhang. Time-spread multiple-access (tsma) protocols for multihop mobile radio networks. *IEEE/ACM Trans. Netw.*, 5(6):804–812, 1997.

[11] I. Chlamtac and A. Lerner. Fair algorithms for maximal link activation in multihop radio networks. *IEEE Transactions on Communications*, 35(7):739–746, July 1987.

[12] I. Cidon and M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Transactions on Computers*, pages 1353–1361, october 1989.

[13] I. Cidon and M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Transactions on Computers*, 38(10):1236–1361, October 1989.

[14] H. Dai and R. Han. TSync: a lightweight bidirectional time synchronization service for wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(1):125–139, 2004.

[15] E.D.Kaplan. *Understanding GPS: Principles and Applications*. Artech House, 1996.

[16] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *IPDPS 2001*, April 2001.

[17] A. Ephremides and T.V. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on on Communications*, 38(4):456–460, April 1990.

[18] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, 2001.

[19] S. Floyd, V. Jacobson, C-G Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, 1997.

[20] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 138–149. ACM Press, 2003.

[21] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS 2000*, November 2000.

[22] IEEE. IEEE Standard for Information Technology - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements. IEEE Standard 802.11e-2005.

[23] IEEE. IEEE Standard for Information Technology - Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for high rate Wireless Personal Area Networks (WPANs). IEEE Std 802.15.3-2003.

[24] IEEE. IEEE Standard for Information Technology: Wireless LAN Medium Access Control (MAC) and Physical Layer Specifications. ANSI/IEEE Standard 802.11, 1999 Edition, 1999.

[25] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks, 2001.

[26] J.H. Ju and V.O.K. Li. An optimal topology-transparent scheduling method in multihop packet radio networks. *IEEE/ACM Transactions on Networking*, 6(3):298–306, June 1998.

[27] D. Kim, C.-K. Toh, and Y. Choi. TCP-BuS: Improving TCP performance in wireless ad hoc networks. *Journal of Communications and Networks*, 3(2), June 2001.

[28] L. Kleirock and F.A. Tobagi. Packet Switching in Radio Channels, Part 1: Carrier Sense Multiple-Access Models and their Throughpput-delay Characteristics. *IEEE Transactions on Communications*, 23(12):1400–1416, 1975.

[29] L. Kleirock and F.A. Tobagi. Packet Switching in Radio Channels, Part 2: Hidden-terminal Problem in Carrier Sense Multiple Access and the Busy-tone Solution. *IEEE Transactions on Communications*, 23(12):1417–1433, 1975.

[30] R. Krishnan and J. Sterbenz. An evaluation of the TSMA protocol as a control channel mechanism in MMWN, 2000.

[31] S. Lam. A carrier sense multiple access protocol for local networks. *Computer Networks*, 4:21–32, 1980.

[32] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-demand multicast routing protocol. *Proceedings of IEEE WCNC*, 1999.

[33] S.-J. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol for multihop wireless mobile networks. *ACM/Kluwer Mobile Networks and Applications*, 7(6):441–453, December 2002.

[34] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *INFOCOM (2)*, pages 565–574, 2000.

[35] J. Liu and S. Singh. ATCP: TCP for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1300–1315, July 2001.

[36] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks. In *Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Santa Fe, NM, April 2004.

[37] E. Pagani and G. P. Rossi. Reliable broadcast in mobile multihop packet networks. In *Proceedings of the third annual ACM/IEEE international conference on Mobile computing and networking*, pages 34–42. ACM Press, 1997.

[38] C. E. Perkins. *Ad Hoc Networking*. Addison Wesley, 2001.

[39] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of IEEE WMCSA*, pages 90–100, New Orleans, LA, February 1999.

[40] V. Rajendran. Reliable multicasting in ad hoc networks. Master's thesis, University of California, 2003.

[41] V. Rajendran, J. J. Garcia-Luna-Aceves, and K. Obraczka. Energy-efficient, application-aware medium access for sensor networks. In *Proceedings of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems*. IEEE, 2005.

[42] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 181–192. ACM Press, 2003.

[43] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. *Wireless Networks*, 12:63 – 78, Feb 2006.

[44] S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5(2):81–94, 1999.

[45] K. Romer. Time synchronization in ad hoc networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 173–182. ACM Press, 2001.

[46] R. Rozovsky and P. R. Kumar. SEEDEX: a MAC protocol for ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 67–75, New York, NY, USA, 2001. ACM Press.

[47] S. Singh and C. Raghavendra. PAMAS: Power aware multi-access protocol with signaling for ad hoc networks, 1999.

[48] J. So and N. H. Vaidya. Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 222–233, New York, NY, USA, 2004. ACM Press.

[49] M. Stemm and R. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. In *Proceedings of 3rd International Workshop on Mobile Multimedia Communications*, september 1996.

[50] D. Sun and H. Man. ENIC - an improved reliable transport scheme for mobile ad hoc networks. In *Proceedings of IEEE GLOBECOM 2001*, San Antonio, TX, November 2001.

[51] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar. ATP: A reliable transport protocol for ad-hoc networks. In *Proceedings of ACM MobiHoc*, Annapolis, MD, June 2003.

[52] V. R. Syrotiuk, C. J. Colbourn, and A. C.H. Ling. Topology-transparent scheduling for manets using orthogonal arrays. In *DIALM-POMC '03: Proceedings of the 2003 joint*

workshop on Foundations of mobile computing, pages 43–49, New York, NY, USA, 2003. ACM Press.

[53] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla. A reliable, congestion-controlled multicast transport protocol in multimedia multi-hop networks. In *Proceedings of IEEE WPMC 2002*, October 2002.

[54] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla. Reliable adaptive lightweight multicast protocol. In *Proceedings of IEEE ICC 2003*, May 2003.

[55] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla. Congestion controlled adaptive lightweight multicast in wireless mobile ad hoc networks. *Proceedings of IEEE ISCC*, July 2002.

[56] High rate ultra wideband PHY and MAC standard. ECMA Internal Standard ECMA-368, December 2005.

[57] http://www.cs.berkeley.edu/ awoo/smartdust/.

[58] MAC-PHY interface for ECMA-368. ECMA Internal Standard ECMA-369, December 2005.

[59] Product specification, http://www.rfm.com/products/data/tr1000.pdf.

[60] Scalable networks, http://www.scalble-networks.com.

[61] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In *Proceed-ings of the IEEE Infocom*, June 2002.

[62] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 171–180. ACM Press, 2003.

[63] A.L. Wijesinha, Yeong tae Song, M. Krishnan, V. Mathur, J. Ahn, and V. Shyamasundar. Throughput measurement for udp traffic in an IEEE 802.11g WLAN. In *First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005*, pages 220 – 225, May 2005.

[64] A. Woo and D.E. Culler. A transmission control scheme for media access in sensor networks. *ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom) 2001*, 2001.

[65] Y. Xiao. IEEE 802.11n: enhancements for higher throughput in wireless LANs. *Wireless Communications, IEEE*, 12:82 – 91, Dec 2005.

[66] Y. Xiao and J. Rosdahl. Throughput and delay limits of IEEE 802.11. In *IEEE Communications Letters*, volume 6, pages 355 – 357, Aug 2002.

[67] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE Infocom 2002*, June 2002.

[68] W. Ye, J. Heidemann, and D. Estrin. A flexible and reliable radio communication stack on motes. Technical report, USC Information Sciences Institute, September 2002.