# ENERGY EFFICIENT MAC LAYER DESIGN FOR WIRELESS SENSOR NETWORKS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the

Graduate School of The Ohio State University

By

Sha Liu, B.S., M.E., M.A.S.

* * * * *

The Ohio State University

2008

# ABSTRACT

Energy efficient communication is a critical design objective for wireless sensor networks which are usually highly energy constrained. In addition, the throughput and latency performance is also important for several sensor network applications. To simultaneously achieve the seemingly contradictory goals, this dissertation identifies the three major sources of energy wastage in communications, i.e., idle listening, overhearing, and packet retransmissions, and proposes three mechanisms to optimize the energy consumption while maintaining high throughput and low latency.

To deal with the idle listening problem, we design a new low duty-cycle MAC layer protocol called **Convergent MAC (CMAC)**. CMAC can work at low duty cycles and requires no synchronization when there is no traffic. When carrying traffic, CMAC first uses anycast to wake up forwarding nodes, and then converges gradually from route-suboptimal anycast to route-optimal unicast. Experiments and simulations show that CMAC significantly outperforms other duty cycling protocols in terms of latency, throughput and energy efficiency.

The MAC layer anycast technique is also an efficient technique to cope with low link quality and interference in wireless sensor networks. By utilizing the nature of broadcast wireless communication medium and allowing multiple nodes in the forwarding set to compete to be the packet forwarder, links with poor reception quality but good progress in a given routing metric space can be opportunistically used to forward packets. However, the impact of unreliable communication in the reverse channel on anycasting has not been studied before. The second part of this dissertation analyzes the impact of unreliability of reverse links on the performance of existing anycast protocols, proposes a new metric characterizing the number of transmissions in the network for anycast based MAC protocols, and presents an efficient solution for computing the forwarding sets.

The third part of this dissertation is dedicated to optimizing the schedule of packet retransmissions. CSMA relies on carrier sensing to decide if retransmissions should be performed immediately. However, in cases where the poor channel quality persists or packet losses are due to interference undetectable by carrier sensing, the channel assessment alone is not a good indicator of successful transmissions. To schedule retransmissions at appropriate moments, we propose a new technique called **transmission pushback** to reduce such losses by delaying retransmissions. This technique overcomes periods of poor channel quality while ensuring a throughput matching the incoming packet rate. In order to determine the optimal pushback period, we devise an adaptive channel prediction technique based on estimating the parameters of a simple hidden Markov model (HMM) which represents the channel. We dynamically update the parameters of the HMM based solely on the ACK sequence for the previous packet transmissions. By considering both the packet incoming rate and the packet loss pattern, the appropriate pushback period is calculated and applied for future retransmissions.

To my parents and friends.

# ACKNOWLEDGMENTS

Upon completing my Ph.D. study, I would like to express my sincere gratitude to my advisor, colleagues, parents and friends. They have given me enormous help in both study and life.

It is my honor to be able to work with Dr. Prasun Sinha for the past four years. It is him who led me to the research area of wireless sensor networks, and it is also him who guided me the way to high quality research work. I sincerely appreciate the freedom Dr. Sinha gave me in research, and his countless insightful suggestions are what I can benefit from for life. Beyond the materials, Dr. Sinha also impresses me by his highly professional attitude in research, which is also critical for me to learn in the future.

I would like to express my gratitude to Dr. Anish Arora, Dr. Ten H. Lai, and Dr. Dong Xuan. They provided many valuable suggestions for my candidacy proposal and dissertation, which broadened my vision and encouraged much deeper thinking in my research area.

I would also like to thank my colleague Kai-Wei Fan, Zizhan Zheng, and Ren-Shiou Liu. They provided insightful comments and suggestions on my research. Special thanks must be expressed to Kai-Wei Fan, who provided enormous help in both ideas and simulations for my research.

Four-year Ph.D. study has not been easy. There have been many moments when I felt frustrated and depressed. It was my parents, Yangchun Liu and Shoushu Zheng, and friends, Jing Li, Na Li, Lijia Wei, Bo Gu, Jialiang Li, and many more, who helped me out during these times. I would like to thank them for being so supportive.

# VITA

| | |
|---|---|
| January 10, 1979 | Born - Chongqing, China |
| July 2001 | B.S. Statistics, University of Science and Technology of China, China |
| July 2004 | M.E. Computer Science, University of Science and Technology of China, China |
| August 2007 | M.A.S. Applied Statistics, The Ohio State University, U.S.A. |
| September 2004-present | Graduate Teaching & Research Associate, The Ohio State University |

# PUBLICATIONS

## Research Publications

Sha Liu, and Prasun Sinha. "Reverse Channel Aware MAC Layer Anycast for Wireless Sensor Networks". Under submission.

Sha Liu, Kai-Wei Fan and Prasun Sinha. "CMAC: Energy Efficient MAC Layer Design for Sensor Networks with Anycast". Under submission.

Sha Liu, Rahul Srivastava, Can Emre Koksal, and Prasun Sinha. "Achieving Energy Efficiency with Transmission Pushbacks in Sensor Networks". To appear in *Proc. IWQoS*, June 2008.

Kai-Wei Fan, Sha Liu, and Prasun Sinha. “Dynamic Forwarding over Tree-on-DAG for Scalable Data Aggregation in Sensor Networks”. To appear in *IEEE Transactions on Mobile Computing (TMC).*

Sha Liu, Kai-Wei Fan, and Prasun Sinha. “CMAC: An Energy Efficient MAC Layer Protocol Using Convergent Packet Forwarding for Wireless Sensor Networks”. In *Proc. SECON*, pages 11–20, June 2007.

Kai-Wei Fan, Sha Liu, and Prasun Sinha. “Scalable Data Aggregation for Dynamic Events in Sensor Networks”. In *Proc. SenSys*, pages 181–194, November 2006.

Kai-Wei Fan, Sha Liu, and Prasun Sinha. “Structure-free Data Aggregation in Sensor Networks”. In *IEEE Transactions on Mobile Computing (TMC)*, Volume 6, Issue 8, pages 929-942, August 2007.

Kai-Wei Fan, Sha Liu, and Prasun Sinha. “On the Potential of Structure-free Data Aggregation in Sensor Networks”. In *Proc. INFOCOM*, pages 1–12, April 2006.

Sha Liu, Kai-Wei Fan, and Prasun Sinha. “Dynamic Sleep Scheduling using Online Experimentation for Wireless Sensor Networks”. In *Proc. SenMetrics*, pages 166–174, July 2005.

Sha Liu, Shoubao Yang, and Weifeng Sun. “Collaborative SCTP: A Collaborative Approach to Improve the Performance of SCTP over Wired-cum-Wireless Networks”. In *Proc. LCN*, pages 276–283, November 2004.

Sha Liu, and Shoubao Yang. “The Impact of DAD on Handoff Performance of Mobile IPv6 and Test of MLD-Based DAD”. In *Proc. the 1st International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, pages 192–195, January 2004.

Sha Liu, Kai-Wei Fan, and Prasun Sinha. “Protocols for Data Aggregation in Sensor Networks”. Chapter in book titled *Wireless Sensor Networks and Applications*, Editors: Yingshu Li, My Thai, and Weili Wu, Springer Verlag’s book series Network Theory and Applications, 2005.

Kai-Wei Fan, Sha Liu, and Prasun Sinha. “Ad-hoc Routing Protocols”. Chapter in book titled *Algorithms and Protocols for Wireless and Mobile Networks*, Editor: A. Boukerche, CRC/Hall Publisher, 2004.

# FIELDS OF STUDY

Major Field: Computer Science and Engineering

Studies in:

| | |
|---|---|
| Computer Networking | Prof. Prasun Sinha |
| | Prof. Anish Arora |
| | Prof. Dong Xuan |
| | Prof. Steve H. Lai |
| | Prof. Can Emre Koksal |
| Probability Models | Prof. Douglas E. Critchlow |
| Statistics | Prof. Chris Hans |

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# Introduction

## 1.1 Challenges in Energy Efficiency

Energy efficiency is a critical performance metric in wireless sensor networks. Since the radio consumes much higher power than other components such as sensing or processing, it is critical to conserve energy spent on all radio activities especially on idle listening, overhearing and transmissions. Idle listening happens when nodes have their radio transceivers turned on but have no communication activities. Since idle listening consumes energy at almost the same rate as receptions, for applications incurring only sporadic traffic, having transceivers always on consumes significant amount of energy, and thus should be avoided. Overhearing happens when nodes are receiving packets not destined for them due to the broadcast nature of the wireless medium, which should also be avoided as much as possible. To ensure reliability, retransmissions are performed when transmissions fail or acknowledgements do not reach the sender. Due to the unreliability of wireless links, retransmissions are inevitable, but how to select the right moments for retransmissions is a challenging problem. In this dissertation, we try to address the challenges involving all three aspects above. More specifically, the challenges include:

**Using low duty cycling to reduce idle listening:** Since the power consumption during idle listening is of the same order as in transmitting and receiving states, saving energy during idle listening is critical. Among various techniques, radio duty cycling is considered as an effective way to reduce the power consumption when there is no traffic, and many MAC layer protocols have been proposed to deal with this problem [70, 63, 41, 48, 11, 51, 54]. However, existing MAC layer protocols usually cause performance degradation in throughput and latency which are critical metrics for various applications such as event tracking and surveillance. Current duty cycling MAC layer protocols for wireless sensor networks are either synchronized using explicit schedule exchanges or totally unsynchronized. However, both have their weaknesses and deficiencies. SMAC [70], TMAC [63] and DMAC [41] use periodic synchronization messages to schedule duty cycling and packet transmissions. Such message exchanges consume significant energy even when no traffic is present. BMAC [48] uses unsynchronized duty cycling and uses long preambles to wake up receivers. However, the long preamble mechanism leads to high latency and unnecessary preamble transmissions, receptions, and overhearing. Polastre et. al. propose a link abstraction called Sensornet Protocol (SP) [49] to adjust the preamble length by observing recent and nearby traffic. However, SP still relies on long preambles to initiate data flows, and it cannot dynamically select the next hop if the intended next hop is currently unavailable because of sleeping or interference.

**Combating unreliable links using MAC layer anycast:** Limited energy resources equipped in sensor nodes poses significant challenges in wireless communications. For low power devices high packet loss rates have been reported in various studies [69, 69]. In order to provide robustness against transmission failures, MAC layer anycast [35, 38, 77, 76, 27] based solutions have been explored. The key idea is to leverage the broadcast nature of the channel, the density of the network, and the lack of perfect correlation of the channels to the neighboring nodes. Anycasting generalizes the concept of a next-hop node to a subset of neighboring nodes, among which the forwarder is elected dynamically from the nodes that successfully receive the packet transmission. Such elections require reliable acknowledgments from the elected forwarders and a mechanism for arbitration among the receivers, both of which are heavily dependent on the reliability of the reverse links. Although it has been reported that the reverse link is often more reliable immediately after a packet transmission [58, 14, 73], we find that for anycast the performance impact of unreliability in the reverse link is significant. In anycast, an inaccurate arbitration may lead to the election of multiple forwarders which can result in packet duplication. The farther the point of duplication from the sink, the more the wastage of energy. Unreliability in the reverse link can also cause unnecessary packet retransmissions. Thus, reverse links are critical to consider in the design of anycast based MAC layer solutions.

The existing MAC layer anycast protocols ignore the quality of the reverse link. The selection of the forwarding set and the assignment of priorities for arbitration are based either on geographic distance to the sink [77, 76, 27, 38] or the delay metric advertised by the neighboring nodes [35]. A naive forwarding set computation technique is to include all nodes [77, 76] that are better than the current node according to a metric such as geographic progress or delay. In

[38], a simple threshold based approach is used for the selection where nodes with a certain minimum geographical progress are selected in the forwarding set. Observing the sub-optimality of such techniques, an algorithm for optimal selection of forwarding nodes is proposed in [35], which however assumes that the reverse link is reliable. Using data from the testbed for packets of 40 bytes transmitted at $-25dBm$, we construct the anycast forwarding sets and compute the number of transmissions for all source-destination pairs on the Motelab testbed [65]. The expected number of transmissions of a packet originating at a node is represented by the metric EATX (Expected number of Anycast Transmissions), which includes the duplicate copies that can be created due to poor reverse link qualities. As the baseline of comparison, unicast routes along with ETX metrics [20] are also computed. Fig. 3.2 shows the ratio of the EATX of the anycast protocol in [35] to that of unicast. It can be observed that for more than 50% of the source-destination pairs, the performance of unicast is better. For some cases, the anycast protocol in [35] may even result in up to 7 times more transmissions than unicast.

**Scheduling of retransmissions:** High variability in channel quality caused by factors such as fading, mobility, and time-varying multiuser interference makes it difficult to achieve both energy efficiency and high throughput. Without any effort for adapting to the variability, the system resources are consumed highly inefficiently. Due to high packet loss rates, a high fraction of the energy of a node is consumed by multiple retransmissions per packet. The prevailing CSMA based protocols use carrier sensing to avoid collisions and backoffs to address the problem of contention among nearby nodes. However, packet transmissions

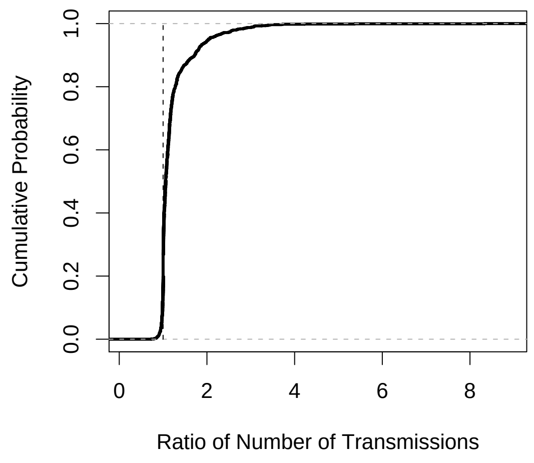


Figure 1.1: Performance degradation of the anycast protocol in [35] due to the ignorance of the reverse link quality).

may fail due to cumulative interference from other nodes in the network. Indeed in our testbed experiments with Mica2 nodes, we have observed that with interfering sources that are sufficiently far away, 69% of the packets for which the CSMA granted a transmission permit are lost. From this example, we can conclude that the combined effect of a large number of interfering sources can be very detrimental and the CSMA based protocols - designed to suppress collisions - are not effective in avoiding such losses. Immediate solution to this problem is reducing the carrier sense threshold that triggers a backoff and consequently increases the carrier sense range. This, in effect, would enable a node to sense this combined interference and hidden terminals to some extent, and avoid some of the losses. However, the increase in carrier sense range makes a node overly conservative with respect to interference and leads to lower effective throughput. Therefore, simple adjustment of the carrier sense range is not sufficient to avoid transmissions during poor channel conditions. To combat the highly variable wireless channel, rate adaptation techniques have been widely studied for wireless LANs. However, in the currently available sensor hardware platforms, the limited computational power rules out sophisticated control actions for adaptation, and thus only very simple strategies (e.g., transmit or do not transmit a packet at a given time) are implementable, which poses further challenges in protocol design.

## 1.2 Contributions

In answer to the challenges listed in Section 1.1, this dissertation makes the following contributions.

**Convergent MAC (CMAC):** The challenges in achieving both low duty cycling and high throughput motivate our design of an energy efficient MAC layer protocol called Convergent MAC (CMAC) (Chapter 2). CMAC uses unsynchronized sleep scheduling when there are no packets to transmit. While transmitting packets, CMAC first uses *aggressive RTS* to *anycast* packets to potential forwarders which wake up first and detect the traffic using a mechanism called *double channel check*. Once the sender is able to transmit packets to a node with an acceptable routing metric, CMAC *converges from anycast forwarding to unicast* to avoid the overhead of anycast. To characterize the operating region of the CMAC protocol, we analytically model the performance of anycast based forwarding and the performance of convergent packet forwarding. To validate the practicability of CMAC, we implement CMAC in TinyOS [7] and compare it with BMAC on the Kansei testbed [9]. We also evaluate CMAC in *ns2* [6] against SMAC [70], DMAC [41], XMAC [11], a variant of GeRaF [77, 76, 13], and an 802.11 based CSMA/CA protocol at 100% duty cycle. The results show that CMAC outperforms other duty cycle scheduling protocols in all aspects while providing throughput and latency performance comparable to the fully awake CSMA/CA protocol. The key contributions of the CMAC design are summarized as follows.

- We propose CMAC, a novel MAC layer protocol, which improves latency and energy efficiency by utilizing the proposed aggressive RTS, anycasting and convergent packet forwarding mechanisms.
- We analytically model the performance of both anycast and unicast based forwarding, and the performance of convergent packet forwarding.
- Using experiments on the Kansei testbed [9, 23], we validate the design goals for CMAC.

**Reverse Link Aware Anycast:** Motivated by the suboptimal performance of existing anycast protocols due to the ignorance of reverse link quality, in Chapter 3 we characterize the impact of unreliable reverse links on the performance of anycast protocols, introduce a new metric to characterize the number of transmissions in the network, and propose an efficient solution for computing the forwarding sets for all nodes.

- We propose a new anycast routing metric called EATX to guide forwarding set selection and route construction. We formulate the computation of EATX considering both forward and reverse link reliability.
- We propose an algorithmic framework for forwarding set selection which takes bidirectional channel quality into consideration. We also propose an anycast route construction mechanism suitable for the prevailing convergecast traffic pattern in wireless sensor networks.
- By analyzing data for unicast transmissions collected from the testbed, we exhibit the nature of unreliability of data packets and corresponding acknowledgments.
- Using simulations driven by data from the Motelab testbed [65], we show that our anycast protocol that uses the proposed forwarding set selection algorithmic framework and the route construction mechanism outperforms unicast and the anycast protocol in [35].

**Transmission Pushback:** To determine the right moments for packet retransmissions, we propose a binary control technique over CSMA. Our approach is based on exploiting the temporal correlations of the interference process. We introduce a new concept called **transmission pushbacks**, which refers to an appropriately computed delay introduced at the MAC layer in order to avoid periods with bad-channel quality while considering a node's throughput requirement. Therefore, we reduce the number of transmissions per packet as well as the number of transmission attempts per unit time. In case of bursty losses, avoiding the bad channel state may also lead to a higher throughput (visible at higher layers) despite lower number of transmission attempts. The main idea of transmission pushbacks is to defer packet transmission attempts for an appropriately selected period upon failed packet transmissions. Plain CSMA leads to failed transmissions, and thus wastes energy, during periods with poor channel quality. CSMA with exponential backoff may reduce such failed transmissions, but it also cuts down the transmission attempts, even at times of improved channel quality. In contrast, our proposed transmission pushback mechanism predicts the duration for which the channel quality will remain poor. Thus, unnecessary transmissions can be avoided to conserve energy and the good channel states are taken advantage of.

To determine the pushback time, we need to estimate the channel quality and how it varies over time. We use an adaptive channel prediction technique based on estimating the parameters of a simple hidden Markov model (HMM) which represents the channel. We dynamically update the parameters of the HMM based solely on the binary ACK sequence (transmission success or failure) for the previous packet transmissions. We choose the appropriate pushback period by considering the throughput requirement posed by the incoming data rate, and the predicted quality of the channel. The proposed approach is simple to implement over existing CSMA based MAC solutions, as well as queue and congestion control algorithms. Therefore it is highly suited for existing sensor network platforms. In summary, the following contributions are made by the design of the pushback technique:

- Using data collected from a sensor network testbed, temporal characteristics of channel variations and interference are studied.

- A novel concept called **transmission pushbacks** is introduced, that is used to increase the packet success rate while considering the throughput constraint at each node.
- Through simulations it is shown that significant gains in energy and/or throughput can be observed in all scenarios using the proposed technique.

## 1.3 Organization of The Dissertation

The rest of the dissertation is organized as follows. In Chapter 2, the design and evaluation of the CMAC protocol is presented. In Chapter 3, we formulate the performance of MAC layer anycast protocols in presence of reverse link losses and propose a forwarding set selection algorithm to optimize the required number of transmissions in the network. In Chapter 4, the transmission pushback technique is proposed and studied. Finally, Chapter 5 concludes the dissertation.

# CHAPTER 2

# CMAC: An Energy Efficient MAC Layer Protocol Using Convergent Packet Forwarding

Duty cycling the radio is important to achieve long lifetime in wireless sensor network, but it usually causes performance degradation in throughput and latency which are critical metrics for various applications such as event tracking and surveillance. These conflicting objectives motivate our design of a new MAC layer protocol called **Convergent MAC (CMAC)**. Compared to other MAC layer protocols like BMAC [48] and SMAC [70], CMAC can significantly reduce latency and improve throughput while supporting very low duty cycles.

## 2.1 Introduction

In this section, we present the motivation of the CMAC protocol design by analyzing the shortcomings of existing protocols. In Section 2.1.1, we discuss the synchronization overhead of existing duty cycling MAC layer protocols. In Section 2.1.2, we discuss the potential of utilizing spatial diversity in wireless sensor networks to reduce latency and improve energy efficiency, and the disadvantages of existing protocols.

### 2.1.1 Synchronization Overhead

Current duty cycling MAC layer protocols for wireless sensor networks are either synchronized using explicit schedule exchanges or totally unsynchronized. However, both have their weaknesses and deficiencies.

SMAC [70], TMAC [63] and DMAC [41] belongs to the category of synchronized protocols. SMAC [70] uses periodic synchronization messages to schedule duty cycling and packet transmissions. TMAC [63] uses the same mechanism as SMAC to synchronize nodes, but TMAC saves more energy by ending the listening period dynamically to reduce idle listening. DMAC [41] and the approach proposed in [60] schedule wake-up periods in a staggered fashion from sources to the sink such that packets can be forwarded without waiting for the next active period. In addition, the active period is divided into receiving and transmitting slots in DMAC to avoid interference with the upstream and downstream nodes. However, these approaches suffer unnecessary energy consumption on synchronization message exchanges when there is no traffic. OMAC [12] schedules transmissions according to receivers' wake-up schedules which are spread across time so that the contention and interference can be minimized. The analysis in [12] exhibits the advantage in energy efficiency of such a receiver-centric scheduling scheme, but this scheme does not take the latency performance into account. Simple calculation for SMAC on Mica2 [4] shows that synchronization messages consume almost 18% of the total energy in the absence of data traffic for 1% duty cycle, which implies 22% potential improvement to the lifetime by eliminating such message exchanges. Note that for duty cycle as low as 0.1%, this improvement could be 225%. Thus for operation at low duty cycles, synchronization messages should be eliminated.

Another paradigm of scheduling sleep and wake-up is to use unsynchronized duty cycling, and wakes up receivers by transmitting long preambles like in BMAC [48] and WiseMAC [22]. Using this technique, the synchronization overhead is delayed until there is traffic. However, the long preamble mechanism has the following three problems.

- The latency accumulated along multihop routes could be overwhelming due to the use of long preambles on each hop.
- The energy consumed on preamble transmission and reception after the receiver has woken up is wasted. This is due to lack of information at the sender side about the wake-up schedule of the receiver, and thus the preamble length is chosen conservatively.
- Neighbor nodes other than the intended receiver will also be kept awake by the long preamble until the data packet transmission finishes, which is also wasteful due to the overhearing of long preambles not destined to them.

Fig. 2.1 shows an example where node A intends to send packets to node C. Due to the long preamble mechanism, both node A and node B uses long preambles even though their intended receivers wake up much earlier than the end of the preambles. Hence, the latency for delivering one packet along the two hops is at least twice the preamble length. If retransmissions are performed, the latency further increases proportionally with the number of transmissions.

Polastre et al. propose a link abstraction called Sensornet Protocol (SP) [49] to adjust the preamble length by observing recent and nearby traffic. However, SP still relies on long preambles to initiate data flows, and long preambles have to be

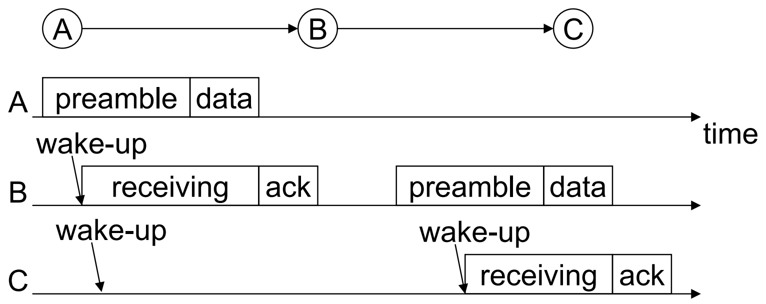


Figure 2.1: Illustration of the disadvantages of using long preambles

used again if such implicit synchronization is lost due to low data rate or interference. XMAC [11] reduces the overhead caused by long preambles by breaking a long preamble into small strobed packets, and thus allows receivers to send feedbacks quickly. But to receive the strobed packets, XMAC introduces much longer awake time than BMAC, which implies higher duty cycles or longer duty cycle length. SCP-MAC [71] is a hybrid protocol utilizing both explicit synchronization and the preamble based technique. In SCP-MAC, nodes exchange synchronization messages at a frequency lower than SMAC, and preambles that are long enough to overcome the clock drift are transmitted before the data. Since SCP-MAC also requires synchronization, energy is still consumed on message exchanges for this purpose. In addition, SP, XMAC and SCP-MAC cannot utilize the spatial diversity (Section 2.1.2) to dynamically select the next hop if the intended next hop is currently not available because of sleeping or interference.

### 2.1.2 Spatial Diversity

In a sensor network, usually multiple nodes are deployed within the transmission range of each node to improve robustness in communications and event sensing. By exploiting such a spatial diversity, it is possible to make quick routing progress by contacting a potential forwarder which wakes up earlier than the best one. GeRaF [77, 76, 13] is a typical example utilizing this idea. In GeRaF, the forwarding region (closer to the destination and within transmission range) is divided into a few sub-regions according to their distances to the destination. Upon forwarding a packet, the sender broadcasts an RTS packet to all nodes in the sub-region closest to the destination and expects a CTS reply. If there is no reply, the sender will broadcast another RTS packet to the sub-region that is the second closest to the destination. This process continues until a CTS is received or all sub-regions have been searched in which case the forwarding fails. If more than one node in the same sub-region happen to wake up and reply to the RTS packet at the same time, the sender detects the collision and directs nodes in that sub-region to send CTS packets probabilistically. After the RTS/CTS handshake, the sender can start to transmit the data packet to the one from which the valid CTS packet is received. Contention based forwarding protocols investigated in [27, 29, 30, 61, 16, 17, 15, 67] share similar idea as GeRaF, but they resolve the contention among receivers by letting receivers delay CTS transmissions for different amount of time and monitor the channel to decide if they should send CTS packets. To favor receivers closer to the destination, they are assigned higher reply priorities (shorter waiting time before sending CTS packets). These contention based forwarding protocols have smaller forwarding overhead compared to GeRaF, but are proposed to circumvent the hot spot of the network and focus on eliminating the state (location of neighbors) maintenance, and thus they have not been employed to cooperate with low duty cycling. There are some other anycast protocols studied in this context, which either list potential receivers and their CTS transmission priorities in RTS packets [33], or probe neighbors in a round-robin manner to find an awake one [19]. RAW [46] has similar idea as the basic anycast scheme described above, but it still requires explicit schedule exchange which should be avoided.

These anycast based approaches provides the potential of low duty cycles and low latency performance by exploiting the spatial diversity, but they still have the following three disadvantages.

- They either are not designed to work at low duty cycles [27, 29, 30, 61, 16, 17, 15, 67], or rely on the receivers to detect the start of an RTS transmission [77, 76, 13], which implies more frequent wake-up or longer active period than BMAC.
- The anycast route could be longer since the optimal forwarding nodes may be asleep during anycasts.
- The overhead of anycast RTS/CTS exchange is higher than unicast. Hence, although these anycast based protocols do not suffer from the overhead of synchronization messages, they incur higher overhead during data transmissions.

### 2.1.3 Contributions

The above problems with existing MAC layer protocols and anycast protocols motivate our design of an energy efficient MAC layer protocol called Convergent MAC (CMAC). We summarize the main contributions of the CMAC design as follows.

- We propose CMAC, a novel MAC layer protocol, which improves latency and energy efficiency by utilizing the proposed aggressive RTS, anycast and convergent packet forwarding mechanisms.

- We analytically model the performance of both anycast and unicast based forwarding, and the performance of convergent packet forwarding;

- We present details of the implementation and experimental evaluation of CMAC on the Kansei testbed [23, 9] to validate our design goals.

The rest of the chapter is organized as follows. Section 2.2 presents the design and analysis of the CMAC protocol. In Section 2.3, we discuss the implementation issues and experimental evaluations on Kansei testbed [9, 23] comparing CMAC with BMAC. Section 2.4 exhibits simulation results comparing CMAC with other protocols. Finally, Section 2.5 summarizes this chapter.

## 2.2 Convergent MAC (CMAC)

Motivated by the limitations of current approaches, we propose a MAC layer protocol called **Convergent MAC (CMAC)** that supports low latency and high throughput as well as low duty cycle operation. When there is no traffic in the network, CMAC uses unsynchronized wake-up scheduling with a pre-defined idle duty cycle. In this wake-up scheduling scheme, the duration between successive wake-ups is fixed according to the duty cycle and active period. However, to spread out the wake-ups of neighboring nodes across time such that the mechanisms in CMAC can benefit from it, we uniformly randomize the wake-up time of each node for the first times it goes back to sleep after receiving a packet. While transmitting packets, the transmitter uses *aggressive RTS* (Section 2.2.1) instead of a long preamble to activate the receiver. To detect aggressive RTS, nodes periodically wake up and "*double check*" the channel for activities (Section 2.2.1). Unlike other unicast MAC layer protocols, CMAC initially uses *anycast* (Section 2.2.2) to transmit the packet to a potential forwarder that wakes up first. Awake candidate receivers will contend to be the anycast receiver by prioritizing their CTS transmissions according to their routing metrics to the sink. After receiving a CTS packet, the data packet will be sent to the sender of the CTS packet immediately. Nodes will keep their radios "on" for a short duration anticipating more packets whenever they successfully receive data packets destined to them. This reduces the overhead of searching for awake forwarders in subsequent transmissions. To overcome the disadvantage of anycast such as higher RTS/CTS overhead and longer route stretch, CMAC *converges from anycast to unicast* once it establishes contact with a receiver having sufficiently good routing metric.

### 2.2.1 Aggressive RTS

The long preamble mechanism of BMAC incurs high latency in order to ensure that the receiver is awake before sending data packets. However, the receiver may wake up much earlier than the end of the preamble, which makes part of the preamble transmission wasteful. By observing such a disadvantage, we propose to use *aggressive RTS* to replace the long preamble, which breaks up the long preamble into multiple RTS packets (thus also called an *RTS burst*). The RTS packets do not use long preambles, and are separated by fixed short gaps each of which allows receivers to start sending back CTS packets. Once the transmitter receives a CTS packet, it sends the data packet immediately. Each RTS gap need not accommodate an entire CTS transmission as long as the RTS sender can detect the preamble and cancel the next RTS transmission accordingly. The number of RTS packets to be sent in one RTS burst depends on the duty cycle length. For the same duty cycle length, the duration of one RTS burst is roughly the same as the long preamble used by BMAC. If nodes uniformly randomly wake up, the expected latency at each hop can be roughly reduced by half. Using pseudo-code, the operations of the RTS sender are summarized as Algorithms 1 and 2.

**Algorithm 1**: OnBackoffEnds($pkt$)

```
   // Try to initiate an RTS burst for packet $pkt$.
1  $RSSI \leftarrow$ GetRSSI();
2  if ($RSSI < CSThreshold$) then
3      $rts.src \leftarrow$ GetMyLocation();
4      $rts.dst \leftarrow pkt.dst$;
5      $rts.dist \leftarrow$ Distance($rts.src$, $rts.dst$);
6      $rts.nav \leftarrow$ GetNav();
7      Send($rts$);
8      $rts\_cnt \leftarrow 1$;
9      InterRTSTimer.start($T_{inter-RTS}$);
10 else
11     BackoffTimer.start(RandomUnifrom(0,$maxBackoff$));
12 end
```

Aggressive RTS can provide the opportunity for receivers to send feedbacks quickly, but if the receiver wakes up and finishes the channel assessment within a single RTS gap, the RTS burst may become undetected. One way to resolve this issue is to lengthen the awake time at each node. But to maintain a certain duty cycle, longer

**Algorithm 2**: OnInterRTSEnds()

```
   // Decide if more RTS packets should be sent
1  $T_d \leftarrow$ GetDutyCycleLength();
2  $maxRTS \leftarrow \frac{T_d}{\text{TxTime}(rts)_{+T_{inter_RTS}}}$;
3  if $(rts\_cnt \leq maxRTS)$ then
4      if $(RadioState \neq IDLE)$ then //Already detected a valid preamble
           // Cancel all following RTS transmissions
5          return;
6      else
7          Send($rts$);
8          $rts\_cnt++$;
9          InterRTSTimer.start($T_{inter-RTS}$);
10     end
11 end
```

awake time also leads to longer sleep time, which can potentially introduce higher latency. Hence, we propose to use a mechanism called *double channel check* to reliably detect ongoing RTS bursts. Double channel check works by assessing the channel twice with a fixed short separation between them each time a node wakes up. The positive conclusion on busy channel from either check will keep the node awake anticipating an RTS. Between these two channel checks, the radio can be put to sleep mode to save energy.

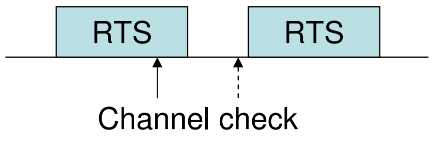
To ensure that the double channel check can reliably detect RTS bursts, it is important to examine the timings of channel assessments and RTS packets. Without any constraint, there are four possible interactions between double channel check and aggressive RTS.

1. The first channel check overlaps with an RTS packet. In this case, the second check is canceled (Fig. 2.2(a)) and the node keeps awake for a while expecting an RTS packet.

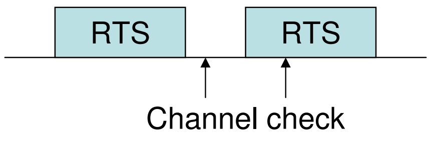
2. The first channel check falls into a gap of RTS packets, but the second check reports a busy channel (Fig. 2.2(b)). In this case, the node also keeps awake for a while expecting an RTS packet.

3. The two channel checks fall into two RTS gaps (Fig. 2.2(c)).

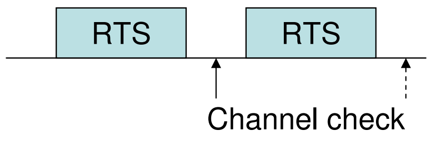
4. Both channel checks fall into the same RTS gap (Fig. 2.2(d)).



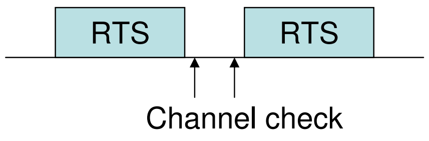
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Figure 2.2: The double channel check mechanism in CMAC.

To ensure the correctness of the aggressive RTS and double channel check mechanism, case 3 (Fig. 2.2(c)) and 4 (Fig. 2.2(d)) should be avoided. To prevent case 3 from happening, the interval between the two channel assessments must be shorter than the RTS transmission time. This can be satisfied by padding RTS packets with extra bytes if needed. We discuss the choice of these parameters in Section 2.3 where the implementation details are presented. To avoid case 4, firstly the RTS gap should be fixed, which is achieved in CMAC by sending all RTS packets without assessing the channel except the first one (Fig. 2.3). Secondly, the channel check interval should also be fixed and must be longer than an RTS gap. Such a "double-check" mechanism

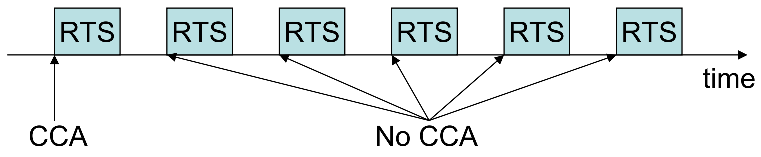


Figure 2.3: CCA in each aggressive RTS burst.

ensures that nodes will not miss any RTS burst in their vicinity. The operations of double channel check are summarized in the pseudo-code shown in Algorithms 3 and 4.

**Algorithm 3**: OnWakeUp()

```
   // Perform the first channel check after waking up.
1  $T_d \leftarrow$ GetDutyCycleLength();
2  $T_i \leftarrow$ GetDoubleCheckInterval();
   // Same as BMAC, use up to 5 samples to improve the robustness
3  $maxSamples \leftarrow 5$;
4  $i \leftarrow 1$;
5  repeat
6      $RSSI \leftarrow$ GetRSSI();
7      if ($RSSI < CSThreshold$) then
8          ChannelCheckTimer.start($T_i$);
9          return;
10     end
11     $i++$;
12 until ($i = maxSamples$) ;
   // No second channel check since the first one is positive
13 Recv(pkt);
```

To avoid the scenario where a node wakes up and detects the energy of the last RTS packet in a burst, the entire duration of an aggressive RTS burst is set to be one more RTS packet longer than receiver's duty cycle length. Let $d, r, g, n$ denotes

**Algorithm 4**: OnChannelCheckTimerExpire()

```
   // Perform the second channel check.
 1 $T_d \leftarrow$ GetDutyCycleLenght();
 2 $maxSamples \leftarrow 5$;
 3 $i \leftarrow 1$;
 4 repeat
 5     $RSSI \leftarrow$ GetRSSI();
 6     if $(RSSI < CSThreshold)$ then
 7         Sleep($T_d$);
 8         return;
 9     end
10     $i++$;
11 until $(i = maxSamples)$ ;
   // The second channel check is positive, start to receive the
      packet
12 Recv(pkt);
```

the duty cycle length, the time to transmit an RTS packet, the interval between two RTS packets, and the number of RTS packets in a burst, respectively. Then $n$ must satisfy

$$n \geq \lceil \frac{d}{r+g} \rceil + 1. \tag{2.1}$$

## 2.2.2 Anycast Based Forwarding

For unicast packet forwarding, the next-hop node is chosen among a set of nodes that can make routing progresses towards the destination (the sink in most sensor networks), and the node that can minimize some routing metric such as geographical distance is chosen as the next-hop node. But if nodes work at low duty cycles, the latency of waiting for the next-hop node to wake up may still be high even if Aggressive RTS is used. To mitigate such an impact on performance, we observe that among nodes that can make routing progresses, some of them may wake up much earlier than the next-hop node. Hence, if these nodes can forward packets before the next-hop node is available, the delay in packet forwarding can be reduced. For example, suppose the duty cycle length is 1 and there are $n$ such potential forwarders, it takes on average $\frac{1}{n+1}$ to get in contact with at least one of them. To exploit this kind of spatial diversity, the sender needs to be notified when such a node wakes up. CMAC achieves this by using a extended Aggressive RTS technique (Section 2.2.1 which allows potential forwarders to send CTS packets. We define the neighbor nodes with smaller routing metrics to the destination than the sender as a *forwarding set*.

However, more than one node in the forwarding set may try to reply to the same RTS, and the one closest to the destination should be elected to receive the data packet. In CMAC, the CTS transmissions are prioritized according to the routing metrics of contending nodes. Nodes with better routing metrics can send CTS packets earlier, while other overhearing nodes cancel their CTS transmissions accordingly. Our approach can work with routing metrics such as geographical distance, hop count, ETX[20], ETT[44] and PRR×Dist[59]. In this chapter, we only investigate the use of geographical distance to resolve CTS contentions.

CMAC partitions the forwarding region into $k$ subregions, $R_1, R_2, \ldots, R_3$, such that nodes in $R_i$ are closer to the destination than nodes in $R_j$ for $1 \leq i < j \leq k$ (Fig. 2.4). Each gap between two consecutive RTS packets is divided into $k$ sub-intervals called *CTS slots*. Nodes in region closer to the destination can send CTS packets in earlier CTS slots. Each CTS slot is further divided into several *mini-slots* to resolve the contention within each region, and each receiver will randomly choose one mini-slot to start its CTS transmission. On detecting a busy channel, pending

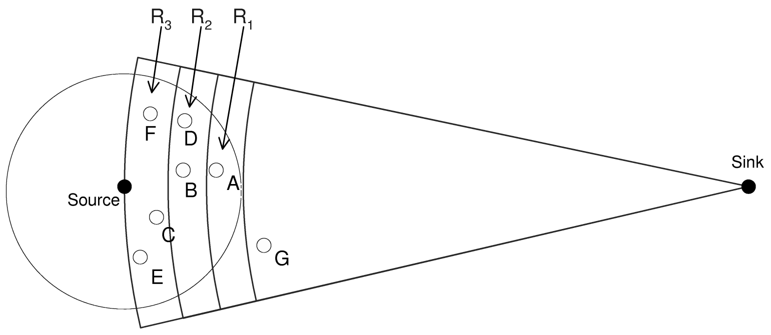


Figure 2.4: Subetting the forwarding region in CMAC.

CTS transmissions will be canceled assuming the existence of another CTS (Fig. 2.5). Note that even though the contention for sending CTS packets is low for very low duty cycles, the above scheme is still necessary as the number of awake nodes will increase with persistent traffic. The optimal selection of the number of CTS slots and the number of mini slots depends on various factors including node density and distance to the sink, and is an open problem. In the rest of this chapter, we use 3 CTS slots and 6 mini slots according to empirical performance measurement using XSM nodes [21] on the Kansei testbed [9, 23]. The operations of the receiver are summarized in pseudo-code in Algorithms 5 and 6.

### 2.2.3 Performance Analysis of Anycast

Anycast can establish contact with a forwarding candidate node faster than unicast, but this is achieved at the cost of higher overhead and less routing progress for each individual transmission. In this subsection, we analytically model the performance of anycast. The metric used in this analysis is the *normalized latency* representing the average latency of each transmission normalized by its geographical

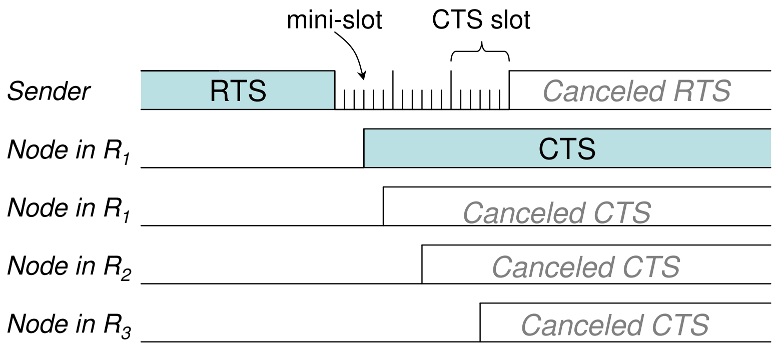


Figure 2.5: CTS contention resolution in CMAC.

**Algorithm 5**: OnRecvRTS($rts$)

```
   // Operations after receiving RTS packet $rts$.
1  $dst \leftarrow rts.dst$;
2  $src \leftarrow rts.src$;
3  $d_s \leftarrow rts.dist$; // Distance from RTS sender to the destination ;
4  $loc \leftarrow$ GetMyLocation();
5  $d \leftarrow$ Distance($loc$, $dst$);
6  if ($d > d_s$) then // No participation if farther away than RTS sender
7      $nav = rts.nav$; // Set network allocation vector (NAV) ;
8      return;
9  else// Participate the anycast
10     if (Receiver is in region $R_j$) then
11         $MiniSlots \leftarrow$ RandomUniform(0,$maxMiniSlots$);
12         $cts.src$ = GetMyAddress();
13         $cts.dst = src$;
14         $cts.nav = rts.nav$ - TxTime($rts$);
15         CTSBackoffTimer.start($j - 1 + MiniSlots$);
16     end
17 end
```

**Algorithm 6**: OnCTSBackoffEnds()

```
   // Operations when the CTS backoff ends
1  $RSSI \leftarrow$ GetRSSI();
2  if ($RSSI > CSThreshold$) then
3      return;
4  else
5      Send($cts$);
6  end
```

routing progress. For the rest of the analysis, the length of a duty cycle is normalized to 1, and the notations used are summarized as follows.

- $L$: normalized latency.
- $\rho$: node density.
- $S$: area of the forwarding region.
- $X$: geographical progress made by anycast.
- $Y$: the latency of finding the first awake node in the forwarding set. Its CDF is $F(y) = 1 - (1-y)^n$. Since the locations of nodes do no affect their wake-up scheduling, $Y$ is independent of $X$.
- $r$: transmission range.
- $r_0$: the minimum progress required for a neighbor node to be present in the forwarding set.
- $d$: distance from the transmitter to the destination.

Note that lower duty cycle leads to longer duty cycle length since the time to check the channel is fixed. Then for very low duty cycles, the RTS and data packet transmission times could be ignored. Hence, $E[Y] \approx \int_0^1 y dF(y) = \frac{1}{\rho S+1}$. Then the expected normalized latency could be expressed as

$$E[L] = E[\frac{Y}{X}] = E[Y]E[\frac{1}{X}] = \frac{1}{\rho S+1}E[\frac{1}{X}], \tag{2.2}$$

where the second equality is due to the independence of $X$ and $Y$.

To compute $E[\frac{1}{X}]$, consider the upper half of the forwarding region as region $OAR$ in Fig. 2.6, where $O$ is the sender and $D$ is the sink. As illustrated in Fig. 2.6, for

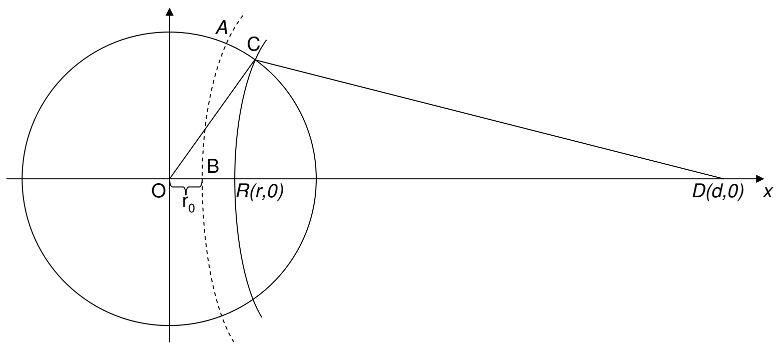


Figure 2.6: Calculation of $E[\frac{1}{X}]$ in the analysis of the anycast protocol in CMAC.

any point $B$ between $O$ and $R$ with $x$ coordinate no less than $r_0$, its weight is $\frac{1}{x}$ times the length of arc $\widehat{BC}$ (Fig. 2.6), while the length of $\widehat{BC}$ is

$$\begin{aligned} |\widehat{BC}| &= |CD| \times \angle CDO && (2.3) \\ &= |CD| \arccos \frac{|CD|^2 + |OD|^2 - |OC|^2}{2|CD||OD|} && (2.4) \\ &= (d-x) \arccos \frac{(d-x)^2 + d^2 - r^2}{2d(d-x)}. && (2.5) \end{aligned}$$

Hence,

$$\begin{aligned} S &= \int_{r_0}^{r} |\widehat{BC}| dx && (2.6) \\ &= \int_{r_0}^{r} (d-x) \arccos \frac{(d-x)^2 + d^2 - r^2}{2d(d-x)} dx, && (2.7) \end{aligned}$$

and

$$\begin{aligned} E[\frac{1}{X}] &= \frac{\int_{r_0}^{r} \frac{1}{x} |\widehat{BC}| dx}{S} && (2.8) \\ &= \frac{\int_{r_0}^{r} \frac{1}{x}(d-x) \arccos \frac{(d-x)^2+d^2-r^2}{2d(d-x)} dx}{\int_{r_0}^{r} (d-x) \arccos \frac{(d-x)^2+d^2-r^2}{2d(d-x)} dx}. && (2.9) \end{aligned}$$

There are three parameters affecting $E[L]$: $r_0$, $S$ and $\rho$. $S$ depends on $r_0$ and $d$. Fig. 2.7(a) plots $E[L]$ versus $r_0$ for various $d$ values when $\rho = 10$. It can be seen that for a certain node density, larger $d$ leads to smaller $E[L]$, but $d$ only affects $E[L]$ a little. In Fig. 2.7(b) $d$ is fixed to be 10, while $\rho$ is varied from 5 to 15. It can be seen that higher node density clearly leads to smaller $E[L]$. In addition, for certain $d$ and node density, there is an optimal value of $r_0$ to optimize $E[L]$ (the minimum points of the curves in Fig. 2.7(a) and 2.7(b)). This behavior is expected since small $r_0$ may lead to little routing progresses, while large $r_0$ may exclude too many good potential forwarders and thus may limit the benefit of spatial diversity.

After finding the optimal $r_0$, nodes still need to decide to use anycast or unicast. For unicast, the normalized latency is bounded by $\frac{1}{2r}$. Hence, for anycast to be superior than unicast on average, it should have lower expected normalized latency. Using Equation (2.2), this criterion leads to the following critical node density above which anycast is better

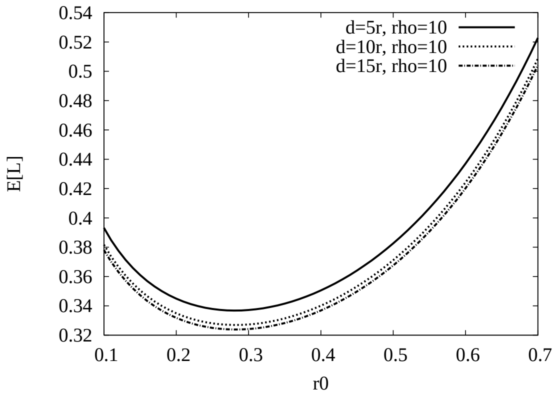
$$\rho > \frac{2rE[\frac{1}{X}] - 1}{S}. \tag{2.10}$$

Using this formula, we can compute if nodes should use anycast or unicast given the parameters of the network or a region of the network. Note that if a node knows the number of nodes in its forwarding set, it can locally make the decision using a similar approach. Specifically, for a node with $n$ neighboring nodes in its forwarding set with each node making $r_i$ progress ($1 \leq i \leq n$), it can decide if anycast is better if

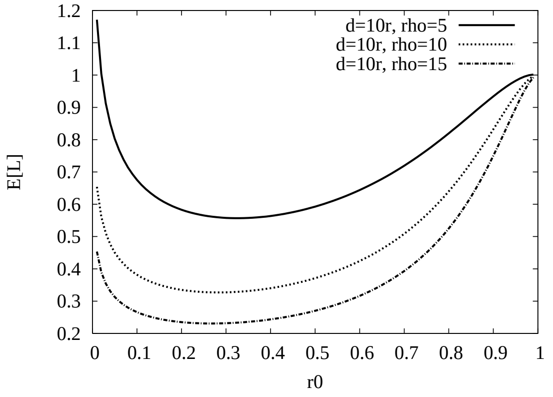
$$\frac{1}{n(n+1)} \sum_{i=1}^{n} \frac{1}{r_i} < \frac{1}{2\max_{1\leq i\leq n}\{r\}}. \tag{2.11}$$

### 2.2.4 Optimizing the Forwarding Set

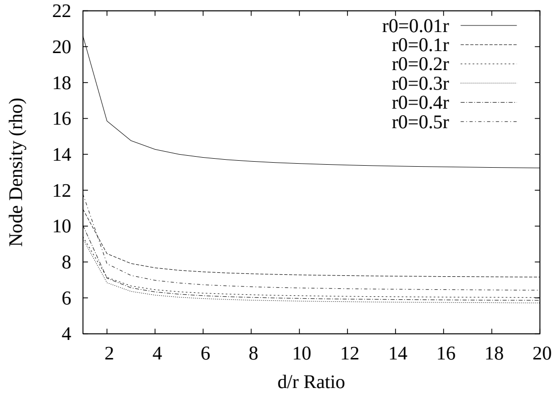
Using anycast, nodes that are closer to the destination and that wake up earlier can pick up the packet and make some progresses in routing. However, as shown



(a) Anycast Performance, vary $d$



(b) Anycast Performance, vary $\rho$



(c) Critical Density

Figure 2.7: Performance analysis of the anycast protocol in CMAC.

in Section 2.2.3, including every node in the forwarding region may not be optimal, and the performance can be improved if the nodes included in the forwarding set are chosen carefully. In this section, we present two algorithms to select the optimal forwarding set. The first one chooses the forwarding set to optimize the normalized latency defined in Section 2.2.3, and the second one attempts to optimize the end-to-end latency.

### Optimizing Expected Normalized Latency

Given $m$ nodes $\{N_{i_1}, N_{i_2}, \ldots, N_{i_m}\}$ in the forwarding set with node $N_{i_k}, k = 1, \ldots, m$, making geographical progress of $r_i$, the normalized latency is

$$E[L(\{r_{i_1}, r_{i_2}, \ldots, r_{i_m}\})] = \frac{1}{m(m+1)} \sum_{j=1}^{m} \frac{1}{r_{i_j}}. \tag{2.12}$$

Hence if a sender has $n$ candidate nodes in its forwarding region, there are totally $2^n - 1$ possible forwarding sets for this node. However, computing the expected normalized latency for all these sets are computational inefficient. To reduce such computation overhead, we propose Algorithm 7 to find the optimal forwarding set. This algorithm adds one node to the forwarding set each time and computes the expected normalized latency provided by the current set. The nodes are added into the set in a way such that nodes closer to the sink are added earlier. In other words, Algorithm 7 computes the normalized latencies of all prefix sets, instead of all subsets, of the sorted candidate set, and selects the prefix set that has the lowest expected normalized latency.

Even though Algorithm 7 does not search all possible subsets, it can find the optimal forwarding set. This can be seen simply by a substitution argument. Suppose the optimal set $S_1$ is not a prefix set, then there exists a node $N_j$ in the forwarding

**Algorithm 7**: Choosing forwarding set to optimize local expected normalized latency

```
   Input: Nodes $\{(N_1, r_1), (N_2, r_2), \ldots, (N_n, r_n)\}$ with $r_j > r_k$ for $j < k$
   Result: Optimal forwarding set in terms of expected normalized latency
1  $OptimalSet \leftarrow \{\}$;
2  $max \leftarrow \infty$;
3  $i \leftarrow 1$;
4  while $(i \leq n)$ do
5      if $(E[L(\{r_1, r_2, \ldots, r_i\})] < max)$ then
6          $max \leftarrow E[L(\{r_1, r_2, \ldots, r_i\})]$;
7          $OptimalSet \leftarrow L(\{r_1, r_2, \ldots, r_i\})$;
8      end
9  end
10 return $OptimalSet$;
```

region that is closer to the destination than at least one node, say node $N_k$, in $S_1$. Then if we substitute $N_k$ in $S_1$ by $N_j$ to form a new set $S_2$, then the expected normalized latency of set $S_2$ will be lower than that of set $S_1$. Hence, it follows that checking all prefix sets is sufficient for finding the optimal forwarding set as done by Algorithm 7.

### Optimizing Expected End-to-End Latency

The ultimate goal of finding optimal forwarding set is to minimize the end-to-end latency from the sender to the receiver. Hence, if the sender has the knowledge on the expected latencies from all nodes in its forwarding region to the destination, then it can use such information to select the optimal forwarding set. Given $m$ nodes $\{N_{i_1}, N_{i_2}, \ldots, N_{i_m}\}$ included in the forwarding set, and the average end-to-end latency from node $N_{i_k}, k = 1, \ldots, m$, to be $l_i$, then the expected end-to-end latency from the sender is

$$E[L(\{l_{i_1}, l_{i_2}, \ldots, l_{i_m}\})] = \frac{1}{m+1} + \frac{1}{m}\sum_{j=1}^{m} \frac{1}{l_{i_j}}. \qquad (2.13)$$

If the nodes included in the forwarding set are sorted in ascending order of their average end-to-end latency to the destination, it can be shown that checking all prefix sets is sufficient for finding the optimal forwarding set. The proof is also by the substitution argument similar as the one for Algorithm 7. Hence, we propose Algorithm 8 to choose the optimal forwarding set if the average end-to-end latencies from all nodes in the forwarding region are known.

**Algorithm 8**: Choosing forwarding set to optimize expected end-to-end latency

```
Input: Nodes $\{(N_1, l_1), (N_2, l_2), \ldots, (N_n, l_n)\}$ with $l_j < l_k$ for $j < k$
Result: Optimal forwarding set in terms of expected end-to-end latency
1  $OptimalSet \leftarrow \{\}$;
2  $max \leftarrow \infty$;
3  $i \leftarrow 1$;
4  while ($i \leq n$) do
5      if ($E[L(\{l_1, l_2, \ldots, l_i\})] < max$) then
6          $max \leftarrow E[L(\{l_1, l_2, \ldots, l_i\})]$;
7          $OptimalSet \leftarrow L(\{l_1, l_2, \ldots, l_i\})$;
8      end
9  end
10 return $OptimalSet$;
```

## 2.2.5 Converging from Anycast to Unicast

Although anycast obviates the need for synchronization messages and has better chance to make progress in packet forwarding than unicast, it has two main shortcomings. First, anycast may choose suboptimal routes because the best next hop is sleeping or due to interference. Second, the overhead of anycast RTS/CTS exchange is usually higher than its unicast counterpart. Hence, a mechanism is needed to reduce the overhead incurred by anycast, and we propose *convergent packet forwarding* to resolve these problems as follows.

In CMAC, the node will remain awake for a short duration after receiving a data packet (The choice of this duration is analyzed in Section 2.2.6). During this period, a node with better routing metric may wake up and become the receiver of the next anycast. If the latest anycast receiver has a routing metric close to the best one (both falling in the same CTS slot), CMAC will use unicast instead to reduce the overhead. Taking Fig. 2.4 as an example, node C might be the earliest to wake up, followed by B and then by node A. Since A is already in the optimal region (region $R_1$ in Fig. 2.4), the transmitter starts to unicast to A regardless if there is any other sleeping node in $R_1$ with greater progress. However, it is possible that there is no node in region $R_1$ in Fig. 2.4. Hence, if the transmitter cannot find a better next hop than the current one after a full duty cycle, it switches to unicast. In this way, the packet forwarding converges from anycast to unicast for each link. After some time without successful data packet reception, CMAC will timeout and nodes will again start following unsynchronized idle duty cycles.

The unicast after the convergence process may or may not use RTS/CTS. In our experiments, CMAC does not use RTS/CTS after convergence for fair comparison with BMAC. In our simulations, CMAC uses RTS/CTS that is similar to 802.11 after convergence for comparison with 802.11, SMAC and GeRaF.

If the sensed event moves, the source nodes may continuously change with each of them generating only a small number of packets. In this case, the convergence may still happen at places closer to the destination where the routes may be more stable.

For some other cases such as low data rates, the convergence may not happen, but CMAC can still use aggressive RTS and anycast to make quick progress towards the sink.

**Convergence Time**

If the sender is backlogged, the convergence from anycast to unicast either when the sender can not find a better node than current one, or when a node in the best forwarding subregion ($R_1$ in Fig. 2.4) wakes up and its CTS packet is received by the sender. For the former case, the convergence takes one duty cycle to finish since this duration is needed to learn that there are no better forwarders. For the latter case, using the notations in Section 2.2.3, the expected latency $L_c$ for at least one node is this region to wake up is

$$L_c = \frac{1}{\rho S_1 + 1}, \tag{2.14}$$

where $S_1$ is the area of the best forwarding subregion and is derived similarly as Eqn. (2.7). Let $r_1$ denote the minimum geographic progress made by nodes in this region, then

$$S_1 = \int_{r_1}^{r} (d-x) \arccos \frac{(d-x)^2 + d^2 - r^2}{2d(d-x)} dx. \tag{2.15}$$

Using the forwarding region division scheme in Fig. 2.4, the latency from sending the first RTS packet until the convergence finishes is plotted in Fig. 2.8. From the figure, it can be observed that the convergence latency decreases slowly with the increase in $d$ due to the slow change of the forwarding region shapes. Density $\rho$ has more significant impact on the convergence time with higher density leading to faster convergence.

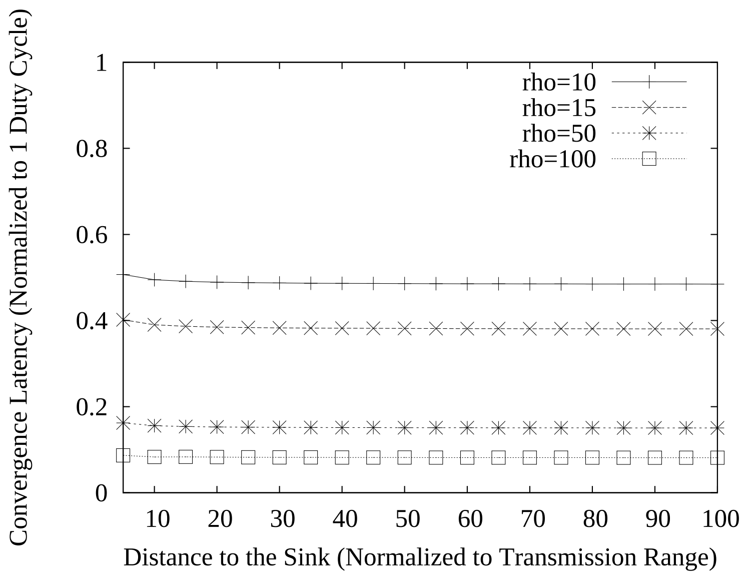


Figure 2.8: Numerical results for convergence latency.

### 2.2.6 Awake Period after Receiving a Packet

To reduce the packet delivery latency, a long awake duration after receiving each packet is preferred. But longer awake period also leads to more energy consumption. Hence, nodes need to optimize the length of this period based on the observed traffic information such as average packet arrival rate to accommodate latency or energy consumption requirements. In what follows, we use a simple model to analyze the latency and power consumption for different awake durations. The duty cycle length is normalized to 1, and other notations are listed below.

- $A$: active duration after receiving a packet.
- $Z$: packet arrival interval (a random variable).
- $G$: the CDF of $Z$.
- $\lambda$: average packet arrival rate.

- $p_r$: power for idle listening and receiving.
- $p_t$: power for transmitting.

If the next packet arrives at the sender before the active duration $A$ times out, which happens with probability $P\{Z < A\} = G(A)$, unicast will be used. Hence the latency is 0 (before transmitting the data packet), and the idle listening for period $Z$ is the single source of energy consumption. Here the energy consumption on the data packet transmissions and receptions is omitted since it is the same regardless of the choice of the awake period. Otherwise, if the next packet arrives at the sender after the active duration times out, which happens with probability $P\{Z \geq A\} = 1 - G(A)$, anycast will be used. Hence the average latency is the average time needed to contact at least one receiver which is $\frac{1}{n+1}$, and the average energy consumption has two components, idle listening for duration $A$ and transmitting aggressive RTS for $\frac{1}{n+1}$. Therefore, the average latency $L_a$ and average energy consumption $E_a$ are

$$\begin{aligned} L_a &= \frac{1}{n+1}(1 - G(A)), \\ E_a &= G(A)p_r E[z|z < A] \\ &\quad +(1 - G(A))(p_r A + p_t \frac{1}{n+1}). \end{aligned}$$

If the packet arrival process is Poisson with parameter $\lambda$, then $G(A) = 1 - e^{-\lambda A}$, and

$$E[z|z < A] = \frac{\int_0^A z\lambda e^{-\lambda z} dz}{\int_0^A \lambda e^{-\lambda z} dz} = \frac{\frac{1}{\lambda} - (A + \frac{1}{\lambda})e^{-\lambda A}}{1 - e^{-\lambda A}}. \tag{2.16}$$

Hence,

$$L_a = \frac{e^{-\lambda A}}{n+1}, \tag{2.17}$$

$$E_a = \frac{p_r}{\lambda} + (\frac{p_t}{n+1} - \frac{p_r}{\lambda})e^{-\lambda A}. \tag{2.18}$$

We plot in Fig. 2.9(a) and Fig. 2.9(b), $L_a$ and $E_a$ versus $A$ for different $\lambda$ and $n$ ($p_r = 1$ and $p_t = 1.5$ for simplicity). It can be seen that $L_a$ decreases with the increase of $A$ given a certain $n$, but $E_a$ has more complex variation patterns. Using Equation (2.18) we can see that there are three cases as follows.

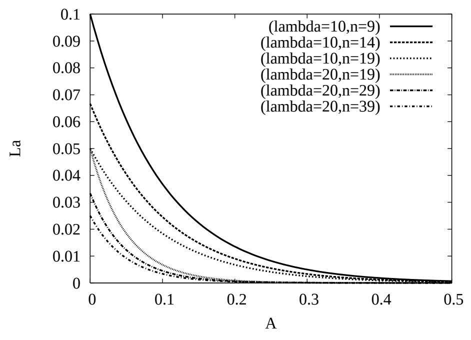
1. If $\frac{p_t}{n+1} < \frac{p_r}{\lambda}$, $E_a$ increases with $A$ up to $\frac{p_r}{\lambda}$. This is because with sufficient node density, the sender can get in contact with a receiver quickly, and thus keeping awake for long time after receiving a packet wastes energy.

2. If $\frac{p_t}{n+1} = \frac{p_r}{\lambda}$, $E_a = \frac{p_r}{\lambda}$, in which cases the packet arrival rate and node density reach the equilibrium related to the ratio between transmission power and reception power.

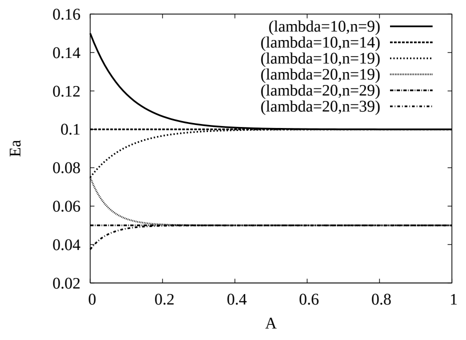
3. If $\frac{p_t}{n+1} > \frac{p_r}{\lambda}$, $E_a$ decreases with $A$ down to $\frac{p_r}{\lambda}$. This is because with packet arrival rate increases, the average idle listening time before receiving another packet decreases, and thus if the sender does not have a forwarding set that is sufficiently large (large $n$), the energy consumed in idle listening may become insignificant compared to the energy consumed by transmitting RTS packets.

### 2.2.7 Synchronized Wake-up Schedule

In order to save more energy after convergence, nodes can synchronize with their upstream and downstream nodes to use synchronized wake-up schedule instead of keeping fully awake. In this section, we present a CMAC variant called CMAC-S using a staggered scheduling idea similar to DMAC [41] after convergence. When the transmitter intends to converge from anycast to unicast, it synchronizes its schedule with the receiver. The two nodes will maintain the staggered schedule as long as



(a) Impact of $A$ on Latency



(b) Impact of $A$ on Energy

Figure 2.9: Numerical results for convergence performance analysis in CMAC.

there is traffic between them. After a certain duration without traffic, the nodes go back to using unsynchronized duty cycling.

In CMAC-S, the transmission and reception slots are split. The synchronized schedule is illustrated in Fig. 2.10. Each wakeup cycle has a sleep time slot and an active time slot. In active time slot, time is divided into a receiving slot and a transmitting slot. Nodes can only transmit data during the transmitting slot. Therefore, the downstream nodes must schedule their receiving time slot to match their upstream node's transmitting slot. Fig. 2.11 illustrates the synchronization schedules of a few synchronized nodes. The receiving and transmitting slots are staggered such that the upstream nodes can transmit to downstream nodes without contention between them. The staggered schedule allows nodes to forward packets from the source to the sink with low delay.

When two nodes agree to synchronize, they must schedule their wakeup periods in a staggered way. We consider the following two cases for synchronization.

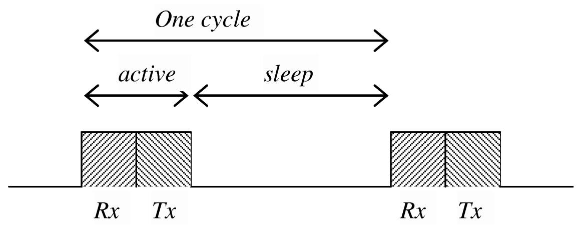


Figure 2.10: Synchronization Schedule in CMAC-S.

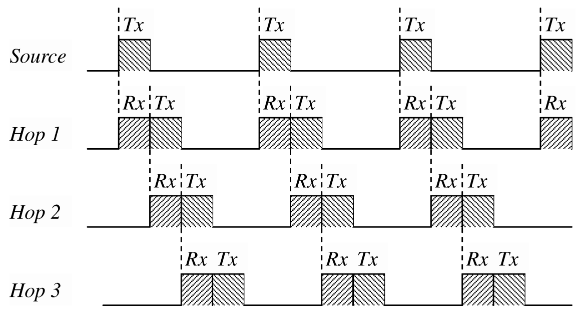


Figure 2.11: Staggered synchronization schedules in CMAC-S.

- *The sender is the source and is not synchronized.* When a sender is not synchronized, the schedule can be started at any time. Fig. 2.12 illustrates a scenario where an unsynchronized sender wants to synchronize with the receiver. The schedule will be started as of the time the sender sends the RTS packet for the first successful data transmission.

- *The sender is an intermediate node, and is already synchronized with its upstream node.* As the sender is already synchronized, it can not change its schedule. So when it needs to synchronize, it explicitly indicates the time elapsed since the beginning of the current transmitting slot (see Fig. 2.13). The receiver uses this offset to determine its staggered wakeup schedule to properly match

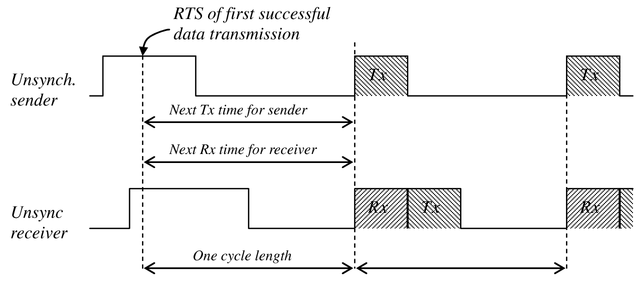


Figure 2.12: The synchronization initiated by the source in CMAC-S.

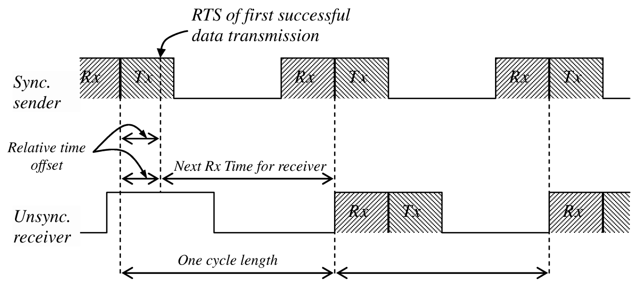


Figure 2.13: The synchronization initiated by intermediate nodes in CMAC-S.

the sender. Therefore even if the transmission failed for the first few tries, the receiver can still know the time to start the schedule.

Multiple sources may simultaneously send data in case of a static event that triggers multiple nodes or in case of a mobile event. When multiple sources need to report data to the sink synchronization needs to be managed across merging routes.

If a sender is not synchronized but the receiver is already synchronized with another sender (at the junction of two merging flows), it follows the receiver's schedule. The receiver indicates the time elapsed since the beginning of the last receiving time slot in the CTS header, and the sender learns when to start its transmitting slot. If

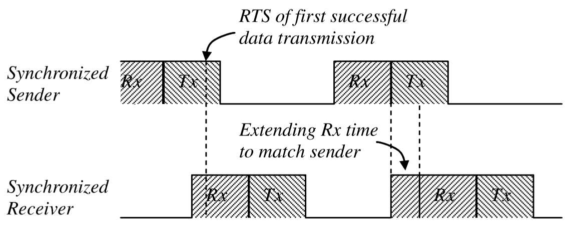

RTS of first successful data transmission
Synchronized Sender
Rx
Tx
Rx
Tx
Extending Rx time to match sender
Synchronized Receiver
Rx
Tx
Rx
Tx

Figure 2.14: Accommodation of upstream schedules in CMAC-S.

both the sender and the receiver are synchronized with their upstream nodes (receiver is synchronized with another sender) but they are not synchronized with each other, the following approaches can be used to adjust their synchronization:

- The sender can match the receiver's schedule and request its upstream nodes to adjust their wakeup schedules. However, this causes a ripple effect that needs to be propagated to the leaf nodes of the tree.
- The receiver switches to a wakeup schedule that satisfies the new sender as well as the old sender(s). However, this requires the receiver to maintain a higher active duty cycle (see Fig. 2.14).
- The sender splits its receiving and transmitting slots to match with its upstream as well as downstream nodes as shown in Fig. 2.15. We have chosen this approach to implement as it incurs lower overhead compared to the other approaches.

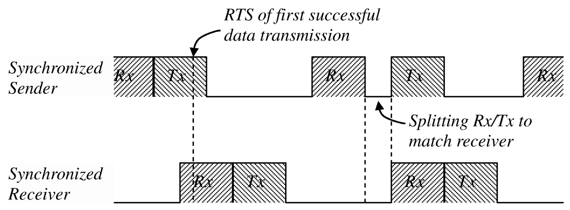

RTS of first successful data transmission
Synchronized Sender
Rx
Tx
Rx
Tx
Rx
Splitting Rx/Tx to match receiver
Synchronized Receiver
Rx
Tx
Rx
Tx

Figure 2.15: Accommodation of downstream schedules in CMAC-S.

## 2.3 Experimental Evaluation

Our TinyOS [7] implementation[1] of CMAC is based on XSM [21] which is similar to Mica2 mote [4] using the same CC1000 radio [1] and microcontroller. We set the mini-slot length to the transmission time of 1 byte on CC1000 radio which is $416\mu s$, a period long enough to accommodate the propagation delay and busy channel detection (One channel sampling takes about $265\mu$ to finish). Other parameters are summarized in Table 2.3. The Kansei testbed consists of 105 XSM nodes forming a $15 \times 7$ topology with node separation of 3 feet. The transmission range is set to 4 rows/columns in the testbed. Each XSM node is attached to a Linux-based stargate [5] through which command messages are sent to trigger the generation of packets.

We evaluate the throughput, latency, and energy efficiency of CMAC against BMAC for two basic event scenarios, static event and moving event. Here throughput refers to the total number of packets received at the sink in 600 seconds, latency is the average delay experienced by a packet, and energy efficiency refers to the energy consumption of the entire network for delivering one 36-byte packet to the sink (called

[1]Code available at `http://www.cse.ohio-state.edu/~liusha/cmac`.

| | |
|---|---|
| CTS-slot length | 7.488$ms$ |
| Number of CTS-slots | 3 |
| Mini-slot length | 416$\mu s$ |
| Number of mini-slots | 6 |
| RTS packet size | 44 bytes |
| Double channel check interval | 10$ms$ |
| Sensing range | 4$ft$ |

Table 2.1: Default experiment parameters for CMAC

normalized energy). We measure the energy consumption by keeping track of the duration nodes spend on idle, receiving, transmitting and sleeping states, and the power consumption rate presented in [48] is used to calculate the total energy consumption. Since CMAC uses geographical progress as the metric to classify potential forwarding nodes, we run greedy geographic routing on top of BMAC. Since the network topology on the testbed a grid, routing void in greedy forwarding is eliminated.

Note that the double channel check almost doubles the times of channel sampling in BMAC. Thus CMAC consumes more energy on channel assessment than BMAC if the duty cycle length is the same. To be fair, we evaluate CMAC with duty cycle length double that of BMAC in this section. For example, if BMAC uses 300$ms$ duty cycle length, CMAC will use 600$ms$. Since using 300$ms$ duty cycle length in BMAC is roughly 1% duty cycle, we denote it by BMAC 1%, and denote CMAC using 600$ms$ duty cycle length as CMAC 1%. To provide the baseline for throughput and latency evaluation, we also gathered the data for BMAC and CMAC without duty cycling, denoted by BMAC 100% and CMAC 100% respectively.

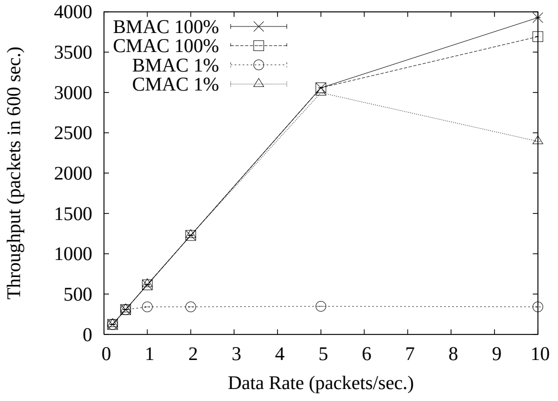
### 2.3.1 Static Event Scenarios

In this set of experiments, we emulate an event happening at one corner of the testbed. The source node sends all packets to the sink located at the diagonally opposite corner. We vary the data rate at source nodes, and the results are shown in Fig. 2.16.

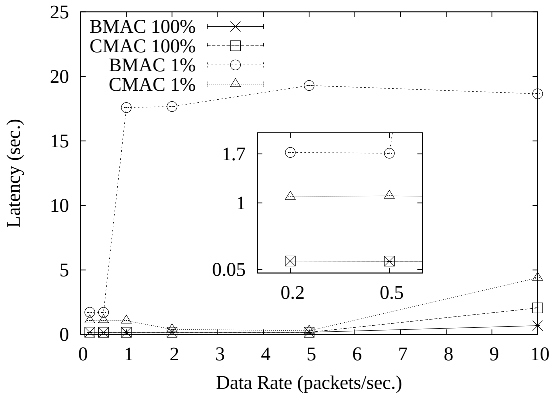
For low data rates (0.2 $\sim$ 0.5 packets/sec.), both CMAC 1% and BMAC 1% can deliver all packets (Fig. 2.16(a)), but Fig. 2.16(b) shows that CMAC 1% exhibits better latency performance than BMAC 1% due to the capability of aggressive RTS and anycast to discover awake potential forwarders.

Under high data rates ($\geq$ 1 packet per second), BMAC 1% can not deliver all packets to the sink, and the flat curve shows that the channel capacity is reached due to the use of long preambles and multihop contention. CMAC 1% saves unnecessary long preambles, and thus not only significantly outperforms BMAC 1% but also provides similar throughput as BMAC 100% and CMAC 100% (Fig. 2.16(a)). In some cases, e.g., data rates of 2 and 5 packets per second, CMAC 1% even provides latency performance very close to that of BMAC 100% (Fig. 2.16(b)). This is due to the convergence of CMAC from anycast to unicast and the saving on anycast overhead. At the data rate of 10 packets per second, CMAC 1% does not provide throughput and latency very close to BMAC 100% or CMAC 100% because the high contention leads to some convergence duration timeouts which result in more RTS/CTS packets, but CMAC 1% still exhibits significant improvement over BMAC 1%.

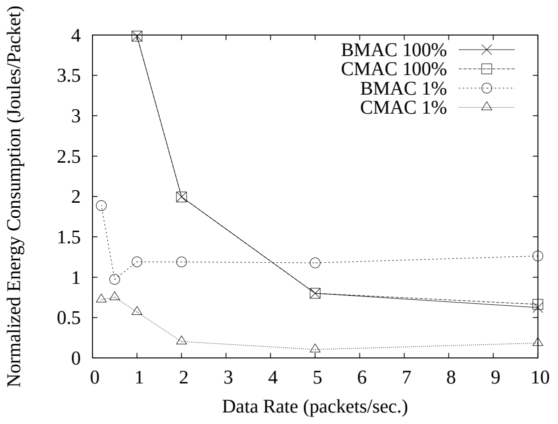
Fig. 2.16(c) shows CMAC 1% utilizes the energy more efficiently than BMAC 1% and BMAC 100%, and the energy efficiency becomes better as the data rate increases.



(a) Throughput



(b) Latency



(c) Energy Efficiency

Figure 2.16: Experiment results of throughput, latency and energy efficiency performance of CMAC and BMAC under different data rates.

Hence, we conclude that CMAC is more suitable for providing high throughput and low latency when the idle duty cycle is low.

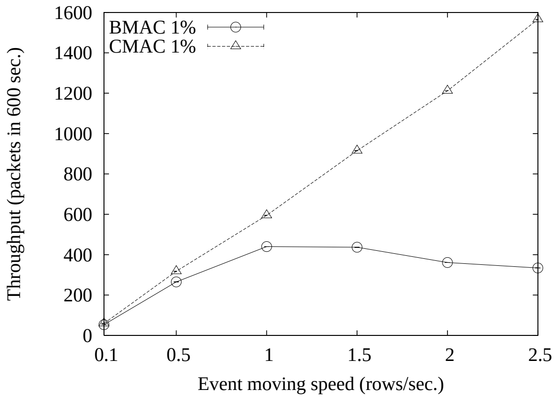
### 2.3.2 Moving Event Scenario

To evaluate the performance of CMAC for moving events, we let the emulated event move along the bottom edge of the testbed at different speeds where faster speed triggers more packets. The results of throughput, latency and energy efficiency are shown in Fig. 2.17.

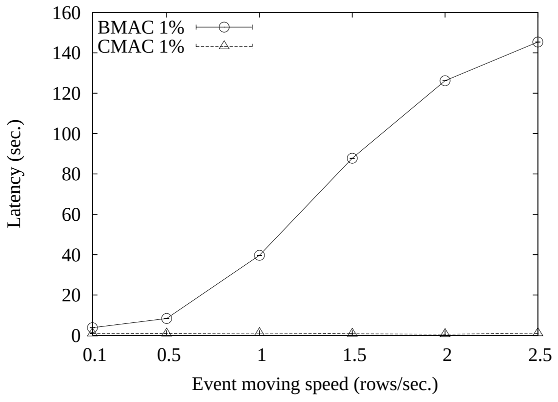
Fig. 2.17(a) exhibits the advantage of CMAC 1% over BMAC 1% in terms of throughput. The throughput of BMAC 1% increases with the increase of the moving speed for slow speeds, but it gradually drops after the moving speed exceeds 1 row/sec. However, the throughput increase of CMAC 1% shows that it can accommodate the packets generated due to faster event moving speed. Fig. 2.17(b) shows remarkable advantage of CMAC 1% over BMAC 1% in latency (less than $1s$ compared to more than $100s$). For BMAC 1%, the queueing delay contributes to most of the latency and is due to the use of long preambles. Fig. 2.17(c) shows the advantage of CMAC 1% in energy efficiency. CMAC 1% saves 75% $\sim$ 95% normalized energy of BMAC 1%. In addition, the normalized energy consumption of CMAC 1% decreases gradually with the increase of moving speed because there are more opportunities for CMAC to converge when there are more active flows. But for BMAC 1%, the normalized energy consumption increases sharply due to the inefficiency of long preambles.

### 2.3.3 Anycast Performance

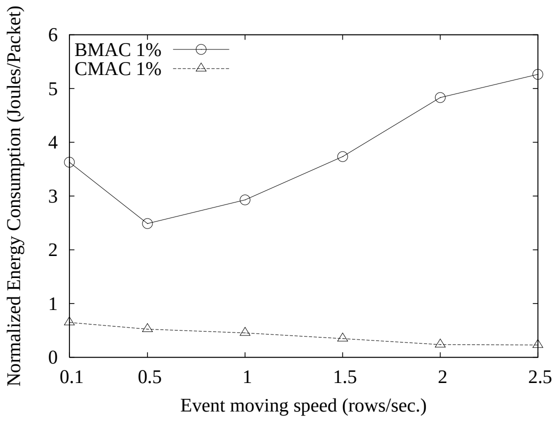
For low data rates, CMAC may not be able to converge from anycast to unicast because the traffic may not be enough. In such cases, the performance of CMAC



(a) Throughput



(b) Latency



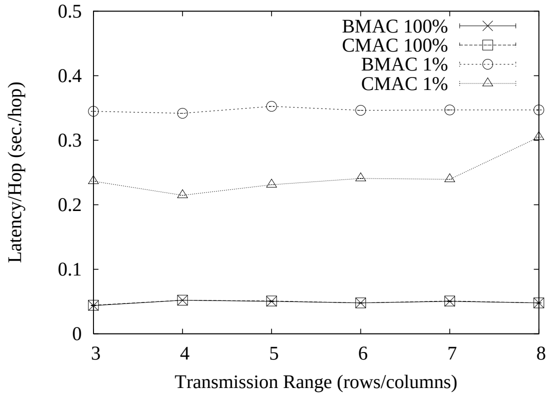
(c) Energy Efficiency

Figure 2.17: Experiment results of throughput, latency and energy efficiency performance of CMAC and BMAC for moving events.

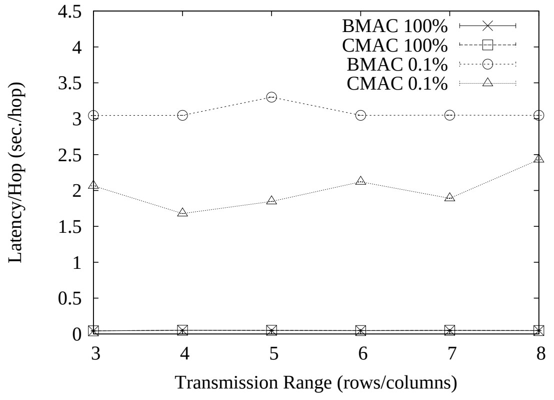
depends on the aggressive RTS and anycast mechanisms. So, we evaluate the performance of the aggressive RTS and anycast mechanism in this section. The duty cycles are 1% and 0.1%, where each cycle is $3000ms$ and $6000ms$ respectively for BMAC 0.1% and CMAC 0.1%. The source node is located at one corner, and the sink is at the diagonally opposite corner. We vary the node density by adjusting the transmission range from 3 rows/columns to 8 rows/columns and run each experiment for 600 seconds. The data rate is chosen such that every packet is purely anycast enroute without any convergence or queuing delay. Due to the limited size of the Kansei testbed, we present the latency normalized by the hop count of unicast, i.e., $\frac{Latency}{Hops}$, and the results are shown in Fig. 2.18(a) and 2.18(b) (Fig. for throughput are omitted since all protocols can deliver all packets to the sink).

CMAC reduces the latency of BMAC by about 33% at both 1% and 0.1% duty cycles except for transmission range of 8 rows with 1% duty cycle, where the improvement is not very significant. The reason for this is that the packet can take as few as 2 hops to reach the destination while the last-hop transmission does not use anycast since the destination is already in range.

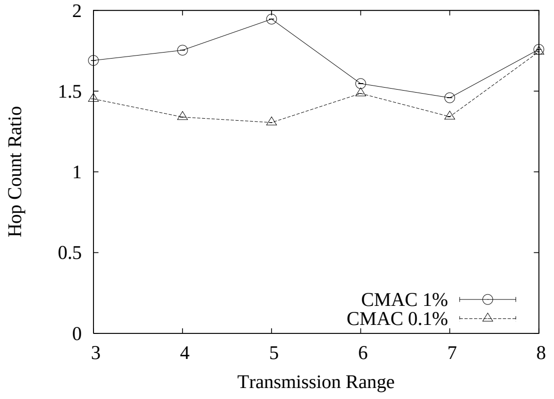
We also collect the route stretch of anycast, which is represented by the average number of hops of anycast normalized by the hop count of unicast. Fig. 2.18(c) shows CMAC 0.1% has larger stretch than 1%. This is because for higher duty cycles, there are more choices in next hop nodes and thus the elected next hop is also better in terms of the routing metric. As Figs. 2.18(a) and 2.18(b) show, even with route stretch, CMAC 1% can still outperform BMAC 1% due to the use of aggressive RTS and anycast.



(a) Per Hop Latency for 1% Duty Cycle



(b) Per Hop Latency for 0.1% Duty Cycle



(c) Route Stretch

Figure 2.18: Experiment results of anycast latency performance of CMAC 1% and CMAC 0.1% under different node densities.

## 2.4 Simulation Based Evaluation

We also conduct simulations[2] for large networks to compare the throughput, latency and normalized energy consumption of CMAC with other protocols using *ns2* [6]. Our study is based on the following six protocols:

- **CMAC:** Our proposed scheme described in Section 2.2.
- **CMAC-S:** Similar to CMAC, but a DMAC-like [41] staggered scheduling is used after convergence.
- **XMAC:** The XMAC protocol proposed in [11] without automatic duty cycle adaptation.
- **GeRaF:** Using the anycast protocol in Section 2.2.2 with $3ms$ active period, which is similar in essence to [77, 76].
- **SMAC:** A full SMAC implementation as proposed in [70] with adaptive listening enabled. The active period is set to accommodate one packet transmission, and thus the message passing is not activated.[3]
- **DMAC:** The DMAC protocol with all mechanisms proposed in [41] implemented, but the wake-up schedules along with the data collection tree are assumed to be predetermined.[4]
- **CSMA/CA:** Fully awake CSMA/CA protocol adapted from the 802.11 code in *ns2* distribution version 2.29.

[2]Code available at `http://www.cse.ohio-state.edu/~liusha/cmac`.

[3]The code is adapted from the *ns2* distribution version 2.29.

[4]The code is adapted from the one from the authors that is downloadable at `http://anrg.usc.edu/www/index.php/Downloads`.

- **Anycast:** Using the anycast mechanism described in Section 2.2.2 with radio fully awake.

All simulations are conducted in a grid network deployed in an area of $2000m \times 2000m$. The performance metrics include throughput, latency and energy efficiency (or normalized energy consumption) which is defined as the energy consumed by the entire network to deliver one byte of application data to the sink. We emulate an event moving randomly at $10m/s$ in this area. Once the event is within the sensing range, nodes continue sending reports about the event at the assigned data rate. We use $250m$ as the transmission range, but our protocol works for any radio transmission range. The routing protocol on top of unicast based MAC protocols is greedy geographic routing where the local minimum is avoided due to the use of grid topology. In this section, we present the performance comparison for various duty cycles, data rates, node densities, sensing ranges, and event moving speeds. Unless otherwise mentioned, the default parameters are set as shown in Table 2.2. All data points are averaged over 10 simulations with different random seeds, and we also plot the 95% confidence interval for each of them.

### 2.4.1 Duty Cycle

In this section, we evaluate the impact of the duty cycle on MAC layer protocols by varying it from 0.1% to 1%. (GeRaF and SMAC use 1% to 10% duty cycles instead because they can barely deliver any packet for lower duty cycles.) Fig. 2.19 shows the comparison in throughput, latency and energy efficiency for CMAC, CMAC-S, SMAC, DMAC, XMAC and GeRaF.

Table 2.2: Default Simulation Parameters

| | | | |
|---|---|---|---|
| Simulation Running Time | $400s$ | RTS size | 20 bytes |
| Bandwidth | $38.4Kbps$ | CTS size | 14 bytes |
| Tx power | $27mA$ | ACK size | 14 bytes |
| Rx power | $10mA$ | Data header | 28 bytes |
| Idle power | $10mA$ | Data payload | 50 bytes |
| CTS slot | $0.2ms$ | Anycast CTS | 20 bytes |
| Active period | $3ms$ | Preamble+PLCP | 24 bytes |
| Transmission Range | $250m$ | Interference Range | $550m$ |
| Sensing Range | $75m$ | Node Separation | $100m$ |
| Event Moving Speed | $10m/s$ | Number of Nodes | $20 \times 20$ |
| Duty Cycle | 0.1% | Data Rate | 1 packet/sec. |

DMAC has the best energy efficiency because the maximum duty cycle in DMAC is 40% [41] even when nodes are actively communicating. However, the advantage is obtained at the cost of low throughput and high latency. With the increase in duty cycle, DMAC's throughput also increases, but it is still significantly lower than CMAC and XMAC even for a duty cycle of 1%. Fig. 2.19(b) does not show the data points for SMAC and DMAC because they incur latency more than 10 times higher than CMAC and XMAC (not shown in for the sake of clarity). Such poor performance in latency is due to the use of large transmitting and receiving slots (about $48ms$ each) which leads to 32 times longer sleep time than CMAC and XMAC. SMAC has similar trend as DMAC but even worse performance in throughput and latency, and thus we exclude SMAC and DMAC in following evaluations to show clearer performance plots.

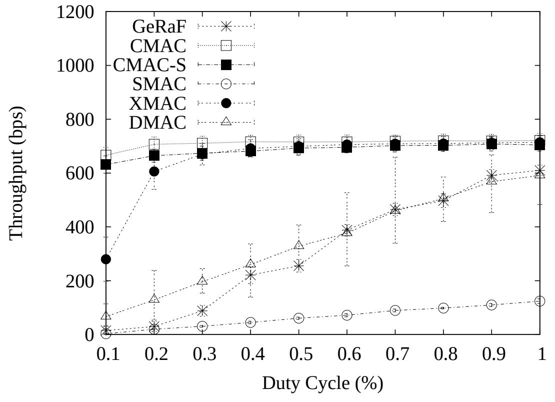
We also observe that GeRaF with 1% duty cycle has lower latency than CMAC and XMAC with 0.1% duty cycle, but the throughput of GeRaF is extremely low at such an low duty cycle. Due to the lack of mechanisms competitive to aggressive

RTS or strobed preambles [11], GeRaF may exhaust its RTS retry limit before any potential forwarder wakes up and thus drop the packet. Since GeRaF also performs much worse than CMAC or XMAC even though they use significantly higher duty cycles, We exclude it from following discussions on performance comparisons.

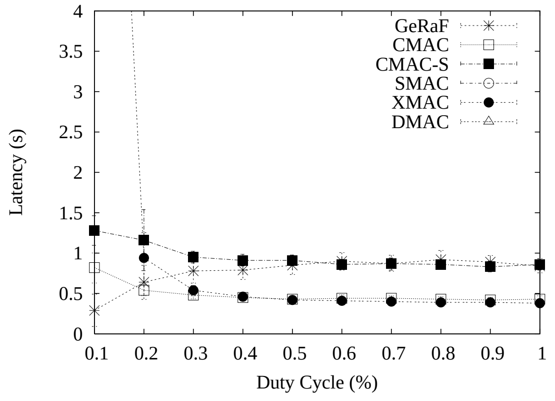
For CMAC, it can be observed that CMAC outperforms other protocols in terms of both throughput and energy efficiency (except for DMAC in energy efficiency). Compared to XMAC, the benefit of using anycast is more significant for lower duty cycles (from 0.1% to 0.3%), and such benefit also translates to more energy savings. CMAC has better performance in throughput and latency than CMAC-S, but CMAC-S is slightly more energy efficient due to the use of the staggered scheduling after convergence.

### 2.4.2 Data Rate

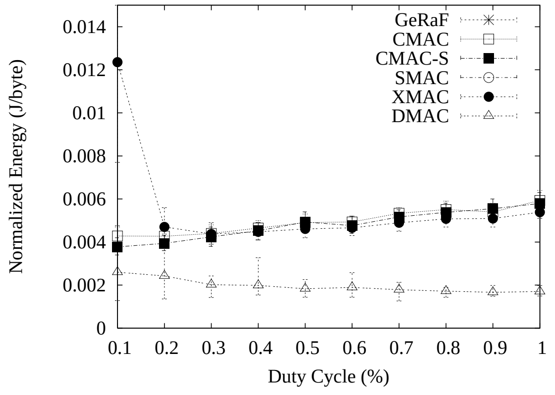
In this section, we evaluate the performance of CMAC for various data rates from 0.05 packet/sec to 1 packet/sec. Fig. 2.20 shows the simulation results for throughput, latency and normalized energy consumption for CSMA/CA, Anycast,CMAC, CMAC-S and XMAC. CSMA/CA and Anycast have the highest throughput and lowest latency, but they achieve these by using significantly more energy. XMAC can not achieve performance close to these two 100% awake protocols in throughput and latency because of the low duty cycle (1%), where the strobed preamble sequence consumes significant amount of time. CMAC and CMAC-S, however, can achieve throughput similar to CSMA/CA and Anycast. For latency performance, CMAC and CMAC-S not only significantly outperform XMAC but also approach the low latency provided by CSMA/CA and Anycast when the packet rate is higher than



(a) Throughput



(b) Latency



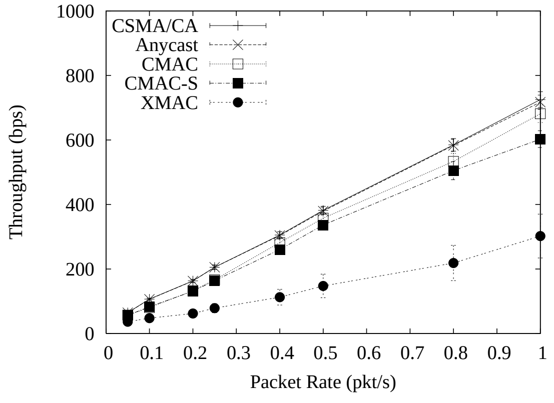
(c) Energy Efficiency

Figure 2.19: Simulation results for CMAC, SMAC, DMAC, XMAC and GeRaF under different duty cycles. The data points of GeRaF and SMAC have duty cycles 10 times of the corresponding X coordinates (due to their inability to deliver any packet for duty cycles lower than 1%. The data points on the latency of SMAC and DMAC are not plotted because it is more than 10 times higher than other protocols.).

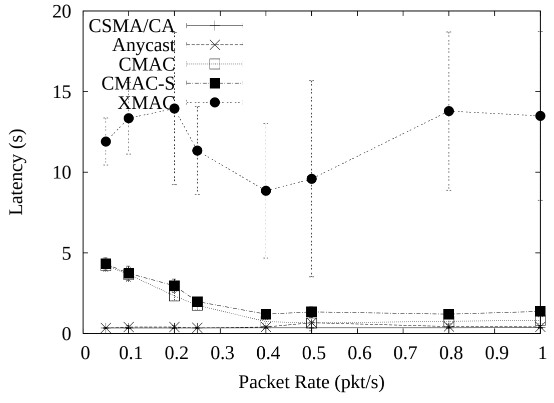
0.4 packets/sec. The performance gain in CMAC compared to XMAC is due to the anycast component which reduces the amount of time seeking for the receiver from $\frac{1}{2}$ to $\frac{1}{n+1}$ where $n$ is the number of nodes in forwarding region. When energy efficiency is considered, CMAC is also much more significant than other protocols (Fig. 2.20(c)). Similar to Section 2.4.1, CMAC has better throughput and latency performance than CMAC-S, but CMAC-S is more energy efficient.

### 2.4.3 Node Density

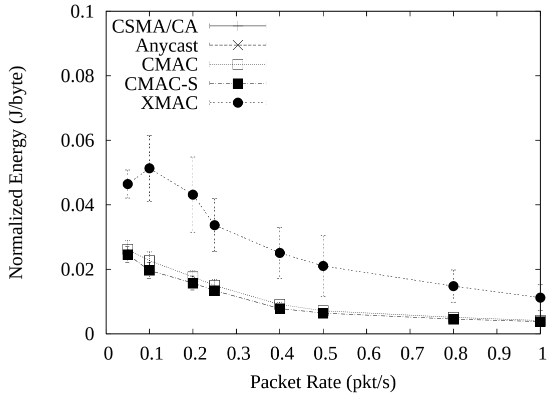
Out analysis results in Section 2.2.3 and 2.2.5 show that node density has significant impact on the performance of anycast and convergence from anycast to unicast. In this section, we evaluate the performance of CMAC in networks with different node densities. We vary the number of nodes in the network from 100 to 625 while keeping the area and sensing range unchanged. Fig. 2.21 shows that the throughput, latency and normalized energy consumption all increase with the increase of node density. This is because for the same sensing range, more nodes are generating packets with higher node density. Fig. 2.21(a)) shows the results in throughput, latency and energy efficiency for CMAC, CMAC-S and XMAC. It can be observed that the throughput for CMAC and CMAC-S increases with higher density, but the throughput of XMAC saturates the channel when the density is high (for scenarios with more than 400 nodes in the network). The trends in latency performance for CMAC and XMAC even diverges (Fig. 2.21(b)). The reason behind such different trends is because CMAC utilizes anycast which benefits from higher node densities, while it is difficult for XMAC to sustain increased traffic rate under high densities.



(a) Throughput



(b) Latency



(c) Energy Efficiency

Figure 2.20: Simulation results for throughput, latency and energy efficiency performance of CMAC, SMAC, GeRaF under different data rates.

Fig. 2.21(c) shows that CMAC and CMAC-S also significantly outperforms XMAC in terms of energy efficiency.

### 2.4.4 Event Size

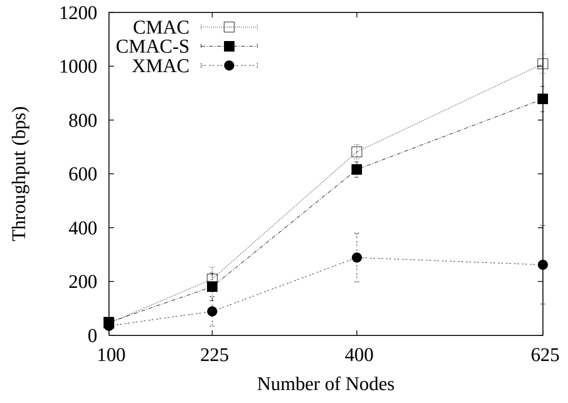
In this section, the performance of CMAC is compared to other protocols under different event sizes. The event radius is varied from $50m$ to $200m$ with larger event triggering more nodes to generate packets. The results for CMAC, CMAC-S and XMAC are shown in Fig. 2.22. We can see that CMAC and CMAC-S outperform XMAC in all aspects. Such performance gain can also be attributed to the benefit of using anycast to quickly forward packets, which reduces both the channel contention from excessive strobed preambles and queueing latency.

### 2.4.5 Event Moving Speed

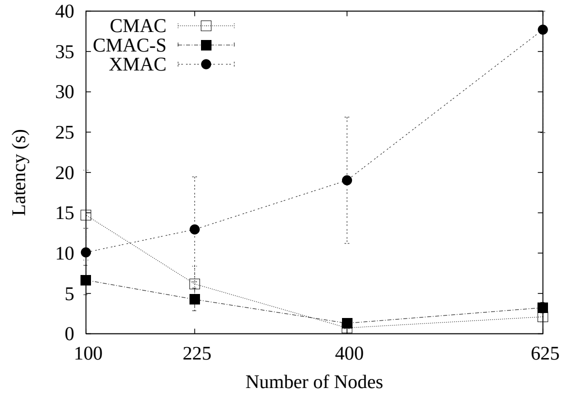
In this section, the performance of CMAC is compared to XMAC for various event moving speeds. The sensing range/event size is varied from $5m/s$ to $20m/s$. With increase in moving speed, a wider range of nodes are triggered to generate packets in any given period. Fig. 2.23 shows the performance comparison, from which it can be seen that for higher speed, CMAC has greater benefit over XMAC in terms of latency and energy efficiency. Again this can be attributed to faster routing progress achieved by anycast than strobed preambles, and the saving in strobed preambles also translates to lower energy consumption.

## 2.5 Summary

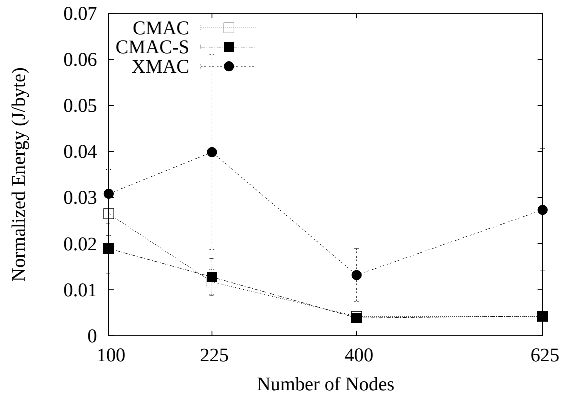
Existing MAC layer solutions for low duty cycling either consume a lot of energy on periodic synchronization messages or incur high latency due to the lack of



(a) Throughput

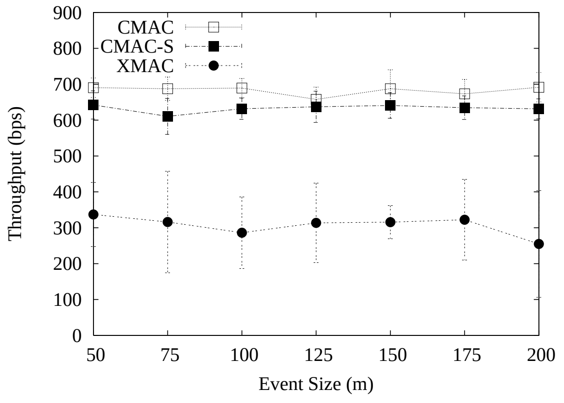


(b) Latency

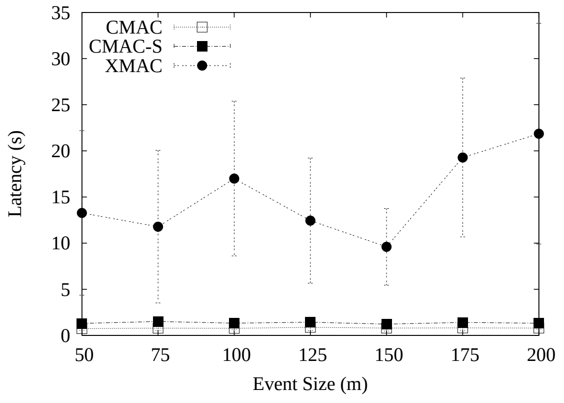


(c) Energy Efficiency

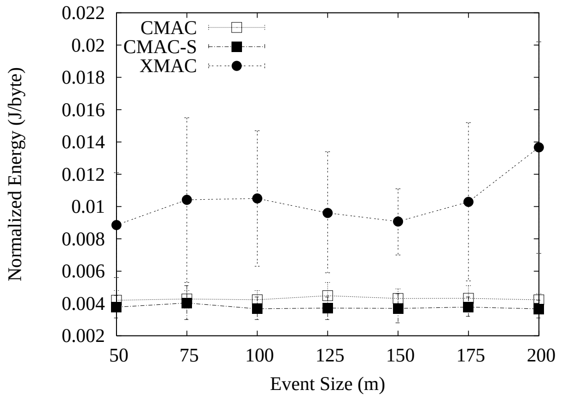
Figure 2.21: Simulation results for CMAC, CMAC-S, XMAC, anycast and CSMA/CA under different node densities.



(a) Throughput

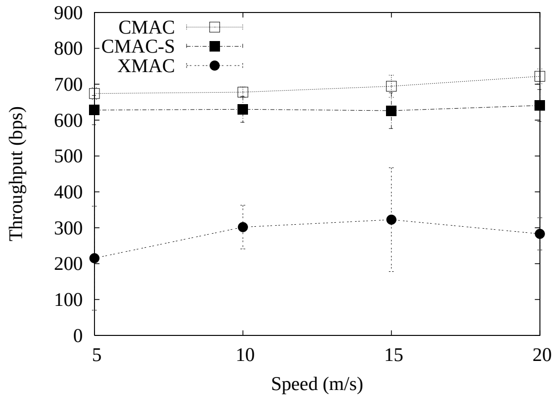


(b) Latency

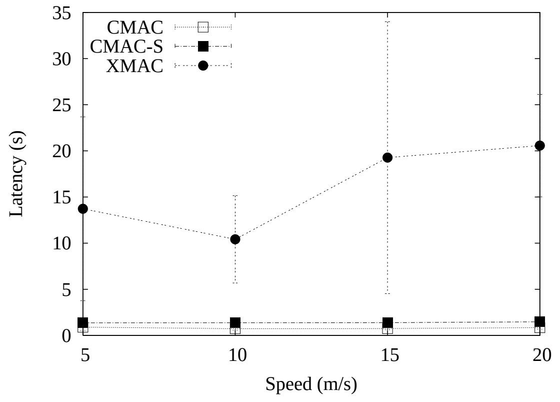


(c) Energy Efficiency

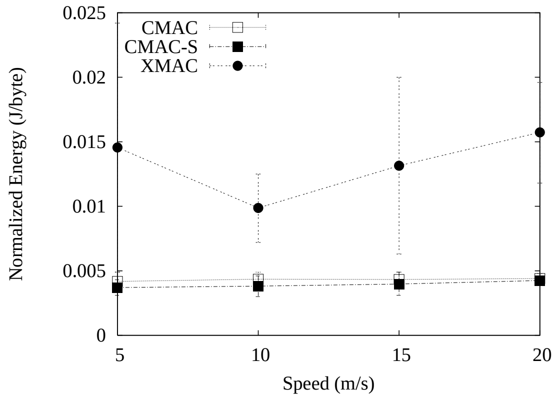
Figure 2.22: Simulation results for throughput, latency and energy efficiency performance of CMAC, CMAC-S and XMAC for different event size.



(a) Throughput



(b) Latency



(c) Energy Efficiency

Figure 2.23: Simulation results for throughput, latency and energy efficiency performance of CMAC, CMAC-S and XMAC under different event moving speeds.

synchronization. To address such problems, we propose a MAC layer protocol called CMAC that comprises of three mechanisms, aggressive RTS, anycast and convergence. We also implement CMAC in TinyOS and evaluate it extensively. The experiment and simulation results show that CMAC at low duty cycles can achieve comparable throughput and latency performance as fully awake CSMA protocol, while greatly outperforming other energy efficient protocols like BMAC, SMAC and GeRaF. Hence, we conclude that CMAC is highly suitable for wireless sensor networks that require low latency and high throughput as well as long network lifetime.

# CHAPTER 3

# Reverse Channel Aware MAC Layer Anycast

## 3.1 Motivation

The limitation in energy resources has posed significant challenges in deploying large-scale unattended sensor networks. One of the key causes for energy wastage is lost packet transmissions, which needs to be minimized to improve the network lifetime. Packet transmissions may be lost due to various phenomena including time-varying channel conditions, collisions, and interference. For low power devices high packet loss rates have been reported in various studies [69, 69]. In order to provide robustness against transmission failures, MAC layer anycast [35, 38, 55, 57, 33, 19, 77, 76, 67, 15, 16, 61, 30, 27, 29, 36, 42] based solutions have been explored. The key idea is to leverage the broadcast nature of the channel, the density of the network, and the lack of perfect correlation of the channels to the neighboring nodes. Anycasting generalizes the concept of a next-hop node to a subset of neighboring nodes, among which the forwarder is elected dynamically from the nodes that successfully receive the packet transmission.

Anycasting requires a reliable acknowledgment from the elected forwarder which is heavily dependent on the reliability of the reverse links. Even though it has been reported in the literature that the reliability of synchronous acknowledgements is higher than unicasting data packets through the same link [58, 14], which is confirmed in our experiments, we find that for anycasting the performance impact of unreliable reverse link is significant. In anycasting, failed acknowledgements and retransmissions may cause multiple nodes to forward the duplicates of the same packet. The farther the point of duplication from the sink, the more the wastage of energy since the total cost is the summed-up cost of all paths taken by all duplicates. In addition, the unreliability of the reverse link can also lead to unnecessary packet retransmissions. Therefore, it is critical to consider reverse links in the design of anycast based MAC layer solutions.

Fig. 3.1 gives an example illustrating how duplicates are created in MAC layer anycast, where the priority numbers decide the order in which nodes send acknowledgements. In Fig. 3.1, nodes B and C compete to forward packets for node A, and B sends acknowledgements earlier than C. The first transmission from A does not reach B, but C receives it and sends an acknowledgement. Even though the acknowledgement is not received by A, C still starts to forward it. A then retransmits the same packet, but this time B instead of C receives it, resulting in a duplication. Because node A still receives no response, it retransmits again. This time both B and C receive it, and they suppress the duplicates according to their packet caches in MAC layer. Suppose 40 transmissions are required for each copy to reach the sink from either B or C, the total cost from A in this case will be 83 transmissions (3 for A to anycast to B or C, 40 each from either B or C to the sink). Considering that further duplicates may be created by downstream nodes of B and C, the actual cost could be even higher.

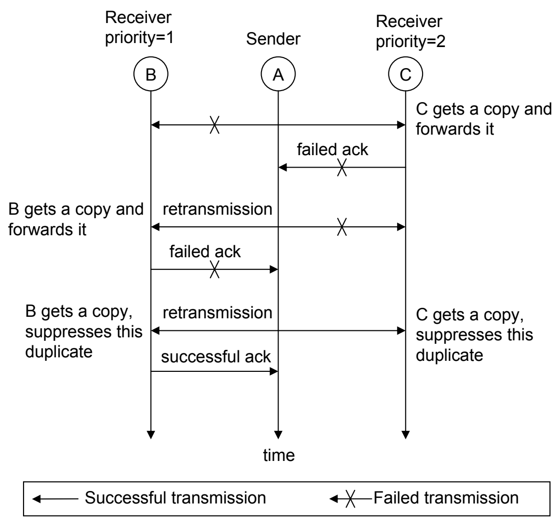


Figure 3.1: Illustration on how duplicates are created in MAC layer anycast.

The existing MAC layer anycast protocols ignore the quality of the reverse link. The selection of the forwarding set and the assignment of priorities for arbitration is based either on geographic distance to the sink [38, 57, 77, 76, 67, 15, 16, 61, 30, 27, 29, 36, 42] or the delay metric advertised by the neighboring nodes [35]. A naive forwarding set computation technique is to include all nodes [77, 76] that are better than the current node according to a metric such as geographic progress or delay. In [38], a simple threshold based approach is used for the selection where nodes with a certain minimum geographical progress were selected in the forwarding set. Observing the sub-optimality of such techniques, an algorithm for optimal selection of forwarding nodes is studied in [35], which however assumes that the reverse link is reliable. Using data from the testbed for packet size of 40 bytes transmitted at $-25dBm$ (details of the data are in Section 3.5.1), we construct the anycast forwarding sets and compute the number of transmissions for all source-destination pairs on the Motelab

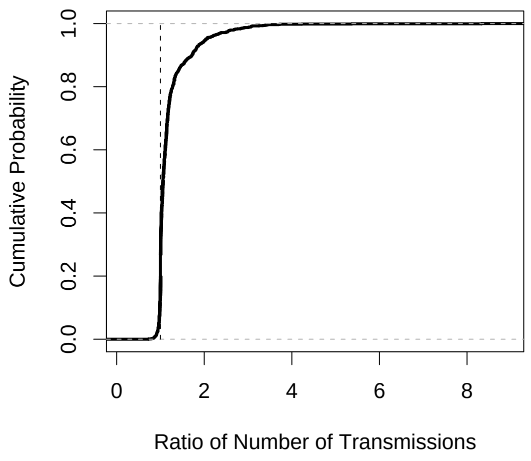


Figure 3.2: Ratio of EATX metrics (anycast in [35] versus unicast).

testbed [65]. The expected number of transmissions of a packet originating at a node is represented by the metric EATX (Expected number of Anycast Transmissions), which includes the duplicate copies that can be created due to poor reverse link qualities. As the baseline of comparison, unicast routes along with ETX metrics [20] are also computed. Fig. 3.2 shows the ratio of the EATX of the anycast protocol in [35] to that of unicast. It can be observed that for more than 50% of the source-destination pairs, the performance of unicast is better. For some cases, the anycast protocol in [35] may even result in up to 7 times more transmissions than unicast.

To understand the reason for the poor performance in Fig. 3.2, we use the scenario shown in Fig. 3.3 as an example. Node A needs to select its forwarding set from candidate pool {B, C, D}. The packet reception rate and acknowledgement reception rate pair (PRR, ARR) along with the EATX values of the candidates are also shown in

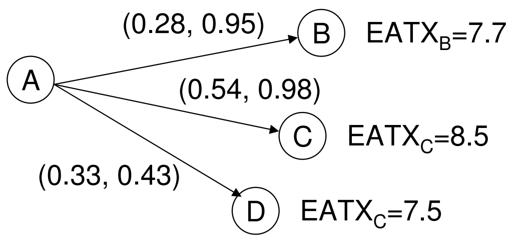


Figure 3.3: 3-node example to show why anycast algorithms that are reverse link unaware perform suboptimally.

the figure. The resulting EATX performance and forwarding sets of various algorithms are shown in Table 3.1. Using the algorithm in [35] which is reverse link unaware, the resulting forwarding set is {D, B, C} with nodes listed according to their priorities, and $EATX_A = 11.29$ (the calculation of EATX is discussed in Section 3.3). However, such performance is even worse than unicast using ETX [20] as the metric which gives $EATX_A = 10.38$ by using C as the next-hop node. In addition, examinations of all possible forwarding sets show that a forwarding set containing both B and C with B having higher priority is optimal with $EATX_A = 9.88$. The anycast forwarding set selection algorithm proposed in Section 3.4 can also successfully choose the optimal forwarding set. In fact, evaluations in Section 3.5 show that the proposed algorithm can achieve optimum for 80% of the cases. The reason for the poor performance of the reverse link unaware anycast in [35] is because node D, which has the lowest EATX value, is selected as the first node in the forwarding set regardless of its low reliability in acknowledgements (ARR=0.43). Hence, the probability of duplication is high once node D receives the packet.

| Protocol | Forwarding set sorted in priorities | $EATX_A$ |
|---|---|---|
| Unicast | C | 10.38 |
| Anycast in [35] | D B C | 11.29 |
| Anycast in Section 3.4 | B C | 9.88 |
| Optimal anycast | B C | 9.88 |

Table 3.1: Performance of various protocols for the 3-node example in Fig. 3.3.

*The objective of this chapter is to characterize the impact of the unreliability of reverse links on the performance of anycast protocols, and to design an efficient solution for computing the forwarding sets for all nodes.* To this end, we propose a new metric to characterize the number of transmissions in the network for anycast based MAC protocols, and design a forwarding set selection algorithm along with a distributed anycast route construction protocol based on it. The key contributions of this chapter are as follows.

- We propose a new anycast routing metric called EATX to guide the forwarding set selection and route construction. We formulate the computation of EATX considering forward and reverse link reliability.
- We propose an algorithm framework for forwarding set selection which takes bidirectional channel quality into consideration. We also propose an anycast route construction mechanism suitable for the prevailing convergecast traffic pattern in wireless sensor networks.
- By analyzing data for unicast transmissions collected from the testbed, we exhibit the nature of unreliability of data packets and corresponding acknowledgments.

- Using simulations driven by data from the testbed, we show that our anycast protocol that uses the forwarding set selection algorithm framework and the route construction mechanism outperforms unicast and the anycast protocol in [35].

The rest of this chapter is organized as follows. Section 3.2 summarizes related work. In Section 3.3, we present the EATX metric formulation. In Section 3.4, we propose the forwarding set selection algorithm framework and the route construction mechanism with the reverse link quality taken into consideration. Section 3.5 provides our study on the bidirectional link reliability and the performance evaluation comparing our algorithm with unicast and an adaption of the algorithm in [35]. Finally Section 3.6 concludes this chapter.

## 3.2 Related Work

MAC layer anycast is originally proposed for wireless ad hoc networks in which nodes are mobile [33, 19, 36, 27, 30, 61, 57]. To reduce the impact of outdated neighbor information due to mobility, nodes do not rely on beacon messages to learn their neighborhood or to determine the next hop forwarding nodes. Instead, neighboring nodes that are closer to the destination contend to send acknowledgments according to the geographical progresses they can make towards the destination. These protocols use the RTS-CTS-DATA-ACK 4-way handshake and are mainly designed to reduce the delay to learn that the current next-hop node has moved out of range. Similar schemes are also proposed for wireless sensor networks to combat low packet reception rate [15, 16, 29]. These papers also analyze various shapes of the forwarding region in which forwarding nodes should reside, but they do not take the bidirectional link quality into consideration.

MAC layer anycast is also proposed to work with low duty-cycling wireless sensor networks [35, 38, 77, 76], where neighbor nodes that wake up earlier and are closer to the sink can compete for the channel to send acknowledgments. To decide which nodes have shorter delays before sending acknowledgments, CMAC [38] and GeRaF [77, 76] divide the forwarding region into several subregions and allow nodes in subregions closer to the sink to respond earlier. GeRaF [77, 76] simply put all nodes closer to the sink into the forwarding region. CMAC [38] defines a minimum progress requirement for each node to be in the forwarding region. This requirement is based on several factors like node density and distance to the sink, but it still does not take the link reliability into consideration. In [35], the probability for each neighbor node to be awake is known by every node, which is used to compute the forwarding set that optimizes the end-to-end delay to the sink. To characterize the impact of packet losses, the algorithm in [35] can also be applied by converting neighbors' awake probabilities into packet reception rates. However, as shown in Section 3.1, the algorithm in [35] often performs worse than unicast due to the ignorance of the reliability in acknowledgements.

Opportunistic routing [10, 75] is another paradigm utilizing the broadcast nature of wireless transmissions. But instead of canceling acknowledgement transmissions based on carrier sensing, protocols in this category rely on the receptions of acknowledgements from nodes with higher priorities to do so. The difference in design is due to the different communication patterns they are designed for. Opportunistic routing is designed for wireless mesh networks where the traffic is bursty, and thus some batching techniques are used to improve the robustness as well as reducing the overhead. In contrast, for wireless sensor networks which usually have low data rates, the batched communication is not suitable, and thus the arbitration on packet forwarding needs to be done on packet-by-packet basis.

In the literature, the reliability of unicast transmissions has been shown to be quite different from the predicted values measured using broadcast packets [58, 73, 14]. More specifically, acknowledgments are most likely to be successful because of their smaller sizes and the higher likelihood of a clearer channel following data packet transmissions. Our bidirectional link quality data measured for the 154 nodes in the Motelab testbed are in agreement with [58] (Section 3.5.1), but we also reveal that the performance of anycast can still be severely impacted regardless of this common finding (Section 3.1 and 3.5).

## 3.3 Anycast Routing Metric EATX

The number of packet transmissions in a network is directly related to the energy consumption and the network lifetime. In addition, excessive unnecessary transmissions leads to high interference and thus high packet loss rate. Hence, it is critical to reduce the total number of transmissions in the network. Due to the lack of coordination among all potential forwarders in MAC layer anycast, packet duplicates may be created, and multiple paths may be taken by them to reach the sink. To design an efficient anycast forwarding set selection algorithm with this factor considered, we introduce a metric called EATX in this section to characterize the total number of anycast transmissions in the network. The ETX metric [20] was designed for a similar objective, but it is applicable only for unicast. To facilitate further discussion, we first introduce our anycast model and define several terms used in the formulation:

### 3.3.1 Model and Notations

We present the MAC layer framework which is similar to the framework used in [35].

Each sender maintains a subset of neighboring nodes, called the *forwarding set* in which each node is assigned a priority. The packet transmission is followed by a sequence of slots, one for each node in the forwarding set. A node with higher priority owns an earlier slot. Each potential forwarder except the first one monitors the energy level in the channel in slots preceding its own to determine if a higher priority forwarding node has acknowledged its reception of the packet. If so, it cancels its own acknowledgement. In [55], optimizations are proposed for the contention based arbitration approaches used in MAC layer anycast protocols, but the mechanism is orthogonal to the topic of this chapter since the causes of acknowledgement failures are different (ACK collision versus channel losses). For the sake of simplicity of modeling, like in [77, 76, 38, 35], we assume that all link loss probabilities (forward as well as reverse) are independent. We also assume nodes that receive the same packet for more than once can suppress the duplications by using some techniques such as packet caching at the MAC layer.

- $S_i$: forwarding set of node $i$.
- $r_{ij}^{S_i}$: number of slots delayed before sending acknowledgements. A node with smaller $r_{ij}^{S_i}$ has higher priority or earlier slot in $S_i$ to send acknowledgements.
- $p_{ij}$: packet reception rate (PRR) for link $i \rightarrow j$.

- $a_{ij}$: acknowledgment reception rate (ARR) for link $i \rightarrow j$ given that node $j$ receives the packet. Note that $a_{ij}$ is different from $p_{ji}$ as shown in [58, 14].

- $P_i^{S_i}$: probability for node $i$ to get a valid acknowledgment in one transmission given forwarding set $S_i$.

- $T_i^{S_i}$: number of transmissions from node $i$ until a valid acknowledgment is received given forwarding set $S_i$.

- $P_{ij}^{S_i}$: probability for node $j$ to get a packet and take the forwarding responsibility in one transmission from node $i$ given forwarding set $S_i$.

- $F_{ij}^{S_i}$: number of packet copies forwarded by node $j \in S_i$ in $T_i$ transmissions from node $i$. Note that $0 \leq F_{ij}^{S_i} \leq 1$.

- $EATX_i^{S_i}$: expected number of end-to-end anycast transmissions to forward one packet to the sink.

To improve the readability, when the forwarding set $S_i$ is clear from the context, $r_{ij}^{S_i}$, $P_i^{S_i}$, $T_i^{S_i}$, $P_{ij}^{S_i}$, $F_{ij}^{S_i}$ and $EATX_i^{S_i}$ can be written as $r_{ij}$, $P_i$, $T_i$, $P_{ij}$, $F_{ij}$ and $EATX_i$, respectively.

We derive the expression for EATX in three steps. First, we calculate the number of transmissions needed for the sender to transmit one packet. Second, the expected end-to-end number of transmissions for any forwarder is presented. Finally, we formulate the EATX metric of the sender.

### 3.3.2 Expected Number of Transmissions ($E[T_i]$)

The number of transmissions for one packet from node $i$ is geometrically distributed with success probability

$$P_i = \sum_{j \in S_i} \left[ p_{ij} a_{ij} \prod_{k \in S_i : r_{ik} < r_{ij}} (1 - p_{ik}) \right]. \tag{3.1}$$

Hence, the expected number of transmissions from node $i$ is

$$\begin{aligned} E[T_i] &= \frac{1}{P_i} & (3.2) \\ &= \frac{1}{\sum_{j \in S_i} \left[ p_{ij} a_{ij} \prod_{k \in S_i : r_{ik} < r_{ij}} (1 - p_{ik}) \right]}. & (3.3) \end{aligned}$$

### 3.3.3 Expected Number of Transmissions from Each Forwarder ($E[C_{ij}]$)

Node $j \in S_i$ forwards the packet received from node $i$ if and only if it has higher priority than all other nodes in $S_i$ that also receive this packet. Therefore, for one transmission from node $i$, the probability for node $j$ to become a forwarder is

$$P_{ij} = p_{ij} \prod_{k \in S_i : r_{ik} < r_{ij}} (1 - p_{ik}). \tag{3.4}$$

To calculate $E[F_{ij}]$, the expected number of packets forwarded by candidate $j$, we consider the conditional probability for node $j$ to forward one packet given that the total number of transmissions is $T$. In other words, for the first $T - 1$ transmissions, the sender (node $i$) gets no acknowledgment while getting a valid acknowledgment for the $T$-th transmission. For any of the first $T - 1$ transmissions to fail, the probability is $1 - P_i$. For node $j$ to forward one packet in any of the first $T - 1$ transmissions, the probability is $(1 - a_{ij})P_{ij}$ because its acknowledgement does not reach the sender according to the base of the conditioning. Hence, in each of the first $T - 1$ transmissions, the probability for node $j$ to forward one packet is $\frac{(1-a_{ij})P_{ij}}{1-P_i}$. Given that the sender gets a valid acknowledgment in the $T$-th transmission, which happens with probability $P_i$, the probability for node $j$ to forward the packet in this round is then $\frac{a_{ij}P_{ij}}{P_i}$.

Since each potential forwarder forwards the same packet at most once, for node $j$ to forward at least one packet ($F_{ij} = 1$), it must become the forwarder in at least one of the $T$ transmissions, the probability for which is thus given by:

$$P\{F_{ij} = 1 | T_i = T\} = 1 - \left[1 - \frac{(1-a_{ij})P_{ij}}{1-P_i}\right]^{T-1} \left(1 - \frac{a_{ij}P_{ij}}{P_i}\right), \tag{3.5}$$

and the expected number of packet copies forwarded by node $j$ is

$$\begin{aligned} E[F_{ij}] &= E[E[F_{ij}|T_i]] \\ &= E[P\{F_{ij} = 1|T_i\}] \\ &= E\left[1 - \left[1 - \frac{(1-a_{ij})P_{ij}}{1-P_i}\right]^{T_i-1} \left(1 - \frac{a_{ij}P_{ij}}{P_i}\right)\right] \\ &= 1 - \left(1 - \tfrac{a_{ij}P_{ij}}{P_i}\right) \textstyle\sum_{k=1}^{\infty} \left[1 - \tfrac{(1-a_{ij})P_{ij}}{1-P_i}\right]^{k-1} (1-P_i)^{k-1} P_i \\ &= 1 - \frac{P_i - a_{ij}P_{ij}}{1 - [1 - P_i - (1-a_{ij})P_{ij}]} \\ &= \frac{P_{ij}}{P_i + (1-a_{ij})P_{ij}} \end{aligned}$$

### 3.3.4 EATX

Given $E[T_i]$ and $E[F_{ij}]$, the EATX metric of node $i$ is

$$\begin{aligned} EATX_i &= E[T_i] + \sum_{j \in S_i} E[F_{ij}] EATX_j & (3.6) \\ &= \frac{1}{P_i} + \sum_{j \in S_i} \frac{P_{ij}}{P_i + (1-a_{ij})P_{ij}} EATX_j, & (3.7) \end{aligned}$$

which can be computed recursively from the sink by assigning $EATX = 0$ to it.

Note that if there is only one node in the forwarding set, anycast degenerates to unicast, and $EATX_i = ETX_i$. To see this, we have $P_i = p_{ij}a_{ij}$ and $P_{ij} = p_{ij}$ for such cases, and thus

$$\begin{aligned} EATX_i &= \frac{1}{p_{ij}a_{ij}} + \frac{p_{ij}}{p_{ij}a_{ij} + (1-a_{ij})p_{ij}} EATX_j \\ &= \frac{1}{p_{ij}a_{ij}} + EATX_j, \end{aligned}$$

which is the same formula as in unicast.

If the anycast protocol in [35] is adapted by converting the awake probability of forwarding candidates to packet reception rate, then the anycast metric therein is a special case of EATX when $a_{ij} = 1, j \in S_i$. In such cases,

$$\begin{aligned} P_i &= \sum_{j \in S_i} \left[ p_{ij} \prod_{k \in S_i : r_{ik} < r_{ij}} (1 - p_{ik}) \right] \\ &= 1 - \prod_{j \in S_i} (1 - p_{ij}), \end{aligned}$$

and thus,

$$\begin{aligned} EATX_i &= \frac{1}{P_i} + \sum_{j \in S_i} \frac{P_{ij} \times EATX_j}{P_i + (1-a_{ij})P_{ij}} \Big|_{a_{ij}=1} \\ &= \frac{1 + \sum_{j \in S_i} \left[ (EATX_j \times p_{ij} \prod_{k \in S_i : r_{ik} < r_{ij}} (1 - p_{ik})) \right]}{1 - \prod_{j \in S_i} (1 - p_{ij})}. \end{aligned}$$

## 3.4 Anycast Route Construction

To choose the optimal forwarding set, the sender needs to decide which neighboring nodes to be included as well as their priorities. For a neighborhood of $n$ nodes, there are $2^n - 1$ non-empty subsets, and for a non-empty subset of cardinality $k \leq n$, there are $k!$ possible priority assignments. Therefore, it is a computation intensive operation to compute the optimal solution by enumerating all possibilities. In this section, we first propose a forwarding set selection algorithm based on an ordering mechanism which sorts nodes according to certain criteria. By consulting the formulation of

EATX, we propose two ordering criteria for further evaluation in Section 3.5. In Section 3.4.2, we exhibit how to construct anycast routes for the entire network.

### 3.4.1 Forwarding Set Selection Algorithm

Once the EATX values of all neighbors are obtained, each node can use Algorithm 9 to decide which nodes should be included in its forwarding set and the priorities. For the sake of readability, the global ID's of neighbor nodes are renumbered locally. The principle of our forwarding set selection framework is to sort all neighbors according to some criteria that characterizes their *goodness to be an anycast forwarder*, where nodes with better goodness are placed in the front the sorted list. The metric to decide the ordering can be any cost metric suitable for reducing the EATX of the selected forwarding set. Algorithm 9 starts from an empty forwarding set (lines 1–3), and gradually adds nodes according to the ordering (lines 4–12). Each time a node is added, the EATX value of the resulting set is calculated (line 6). The set that has the minimum EATX value is chosen as the forwarding set, and the priorities are assigned according to the ordering of the nodes in the list (lines 7–11).

Algorithm 9 runs in $O(n)$ time dominated by the single loop examining all prefixes of the sorted node list. Note that the calculation of EATX for any prefix of the sorted node list can finish within $O(1)$ time. This is because even though it requires the calculation of the cumulative product term $\prod_{j=1}^{k}(1-p_j)$ which seems to need $O(n)$ time, the latest result at the end of each iteration of $k$ can be saved so that it takes only one instead of $k$ multiplications for the next iteration. But for better readability, this detail is not shown in Algorithm 9.

**Algorithm 9**: Forwarding-Set-Selection()

```
input  : $N = \{1, \ldots, n\}$: sorted neighbor list,
         $(p_1, a_1), \ldots, (p_n, a_n)$: bidirectional reliability
         $EATX_1, \ldots, EATX_n$: neighbors' EATX metrics
output: $S$: forwarding set
         $R = (r_1, \ldots, r_n)$: priorities
         $EATX$: EATX value of current node.
1  $S \leftarrow \Phi$ // Initialize the forwarding set
2  $R \leftarrow \Phi$ // Initialize the priority assignment
3  $EATX \leftarrow \infty$ // Initialize the EATX metric
4  foreach ($k = 1 \ldots n$) do
5      $S_{tmp} \leftarrow \{1, \ldots, k\}$
6      $EATX_{tmp} \leftarrow EATX(S_{tmp})$
7      if ($EATX_{tmp} < EATX$) then
8          $S \leftarrow S_{tmp}$
9          $R \leftarrow \{1, \ldots, k\}$
10         $EATX \leftarrow EATX_{tmp}$
11     end
12 end
13 return $(S, R, EATX)$
```

There are many choices for the criteria to sort the neighbor nodes, but the formulation of EATX in Equation (3.7) suggests that nodes with lower EATX values and higher acknowledgement reliability ($a_j$) are more preferred. The data packet reliability ($p_j$) is also important as it is related to the local number of retransmissions. Equation (3.7) also reveals that the EATX metric consists of two components, local number of transmission attempts and the expected total cost from nodes that forward packets. Hence, one natural candidate for the sorting criterion is to use each neighbor $j$'s $EATX_j + \frac{1}{p_j a_j}$ value. This metric also measures the goodness for each neighbor to be the unicast nexthop. However, combining the $p_j$ term with the $a_j$ term leads to underestimation of the chance of losing acknowledgements since $p_j a_j$ can still be large when $a_j$ is small. Since acknowledgement losses are the main reason for duplicated forwarding by multiple nodes, we also evaluate another metric for node ordering which considers neighbor $j$'s $EATX_j + \frac{1}{a_j}$ value. The reason for ignoring the $p_j$ term can also be explained using the data collected from the Motelab testbed [65] from which it can be seen that $p_j$'s and $a_j$'s are usually positively correlated, i.e., links with high $a_j$'s usually also have high $p_j$'s. In summary, we evaluate the following two metrics as the sorting criteria for Algorithm 9 in this chapter.

$$EATX_j + \frac{1}{p_j a_j}, j = 1, \ldots, n, \tag{3.8}$$

and

$$EATX_j + \frac{1}{a_j}, j = 1, \ldots, n. \tag{3.9}$$

### 3.4.2 Distributed Anycast Route Construction

For each node to compute its forwarding set using Algorithm 9, neighbors' EATX metrics must have been learned. But during network initialization, none of the nodes knows its EATX metric except the sink (EATX=0 for the sink). Hence, it is natural to build anycast routes from the sink in a recursive way similar to distance vector routing [47][20]. Algorithms 10 and 11 summarize the operations needed at each node upon different events. The neighbors of the sink will be the first ones to compute their forwarding sets and EATX metrics after receiving broadcast advertisements from the sink; nodes two hops away from the sink will do the computation next; and so on until all nodes in the network have their forwarding sets and EATX metrics computed. This process may take several iterations since nodes may learn more information about their neighborhood after sending their own advertisements.

**Algorithm 10**: On-Recv-New-Adv()

```
input  : $Seq_{curr}$: current latest sequence number
         $Seq$: sequence number in the advertisement
         $ID$: ID of the advertisement sender
         $EATX_{ID}$: EATX value of this neighbor
         $N$: known and sorted neighbors with the latest sequence number
         $EATX$: current EATX metric.
output: $S$: Forwarding Set
         $R$: Priorities of nodes in $S$
         $EATX$: new EATX metric.
1  if ($Seq > Seq_{curr}$) then
2      $Seq_{curr} \leftarrow Seq$
3      $N \leftarrow \{ID\}$
4      if (Packets pending) then
5          $(S, R, EATX) \leftarrow$ Greedy-Selection($N$,$\{(p_{ID}, a_{ID})\}$,$\{EATX_{ID}\}$)
6      end
7      StartTimer;
8  else
9      if ($Seq = Seq_{curr}$) then
10         $N \leftarrow N \cup \{ID\}$
11         if (Timer not running) then
12             call On-Timeout()
13         else
14             if (Packet pending) then
15                 $(S, R, EATX) \leftarrow$ Greedy-Selection($N$,$\{(p_j, a_j) : j \in N\}$,
                   $\{EATX_j : j \in N\}$)
16             end
17         end
18     end
19 end
```

**Algorithm 11**: On-Timeout()

```
input  : $S$: current forwarding set
         $R$: current priority assignment
         $EATX$: current EATX metric
         $N$: sorted neighbor nodes with the latest sequence number.
output : $S$: Forwarding Set
         $R$: Priorities of nodes in $S$
         $EATX$: new EATX metric.
1 $(S_{old}, R_{old}, EATX_{old}) \leftarrow (S, R, EATX)$
2 $(S, R, EATX) \leftarrow$ Greedy-Selection($N$,$\{(p_j, a_j) : j \in N\}$, $\{EATX_j : j \in N\}$)
3 if $((S, R, EATX) \neq (S_{old}, R_{old}, EATX_{old}))$ then
4     Broadcast $(S, R, EATX)$
5 end
```

It is important to notice that usually nodes can not expect advertisements from all neighbors because the information for some nodes (usually farther away from the sink) is simply unavailable or outdated. Hence, each node can compute its forwarding set and EATX metric after receiving any new information. However, the computation and the following advertisement broadcasts should be postponed in anticipation of more incoming advertisements to conserve the energy consumed on computations before sufficient information has been collected. The delay in broadcast is also necessary to reduce the amount of broadcast traffic and to dampen route fluctuations. To do this, during the route construction, each node waits on a timer while trying to receive more advertisements before starting computation and broadcasting (lines 7, 13–17 in Algorithm 10). The length for the timer can be learned from historical records of the delay between receiving the first advertisement and receiving all information from nodes currently in the forwarding set. If any packet is pending to be transmitted, the packet holder has to compute its forwarding set according to all the information collected so far, but advertisement can still be postponed until the timer fires (lines 4–6, 14–16 in Algorithm 10). When the timer set therein fires, the latest EATX metric and forwarding set are calculated and broadcast (Algorithm 11). Priority assignment is also included in the advertisement for receivers to learn their slots to send acknowledgements.

To ensure that the anycast routing paths are loop-free, we utilize a sequence number based technique similar to DSDV [47]. The sink, the single destination of the usual convergecast traffic pattern prevailing in wireless sensor networks, tags its advertisements with even sequence numbers. Other nodes compute their forwarding sets and EATX metrics using advertisements with the latest sequence number. For significant topology changes, odd sequence numbers will be used by non-sink nodes in advertisements to avoid conflicts with the information populated from the sink. Algorithm 10 summarizes the steps taken when receiving an advertisement. Each node only uses information tagged with the latest sequence number for forwarding set computation (lines 1, 9 in Algorithm 10).

Our anycast route construction procedure generates loop-free routes. To see this, assume there is at least one circular path between node A and B for the same sequence number, i.e., path $A \rightarrow N_1 \rightarrow \ldots \rightarrow N_k \rightarrow B \rightarrow N_{k+1} \rightarrow \ldots \rightarrow N_l \rightarrow A$ exists for some $k$ and $l$. However, Algorithm 9 ensures that for the same sequence number, $EATX_A > EATX_{N_1} > \cdots > EATX_{N_k} > EATX_B > EATX_{N_{k+1}} > \cdots > EATX_{N_l} > EATX_A$, which is self-contradictory.

## 3.5 Performance Evaluation

In this section, we evaluate performance of our anycast route construction using the two proposed ordering criteria. To reflect the true bidirectional link quality in real sensor networks, we use unicast to measure the link reliability on the Motelab testbed (Section 3.5.1) and use the data to drive our simulations. In Section 3.5.1, we present our measurement results and some discussion. In Section 3.5.2, we replicate the topology of the testbed and compare the EATX performance of our algorithms with the anycast algorithm in [35] and unicast.

### 3.5.1 Study on Bidirectional Link Reliability

In this section, we show and discuss the bidirectional unicast reliability measurements from the MoteLab testbed. This testbed consists of 184 Tmote Sky [8] nodes with 154 of them available during our experiments. The Tmote Sky node is equipped with a CC2420 [2] radio which is IEEE 802.15.4-2003 [3] compliant and supports hardware automatic acknowledgement transmssion. To obtain the unicast data, we firstly use broadcast commands through serial forwarder to discover the neighbors of each node with different transmission power levels. Then all discovered links are tested in a round-robin manner to avoid internal interference with different packet sizes and transmission power levels. We record the packet reception rates (PRR) and the acknowledgement reception rates (ARR) and show the results in Fig. 3.4-3.5 for two packet sizes, 40 bytes and 126 bytes, and two transmission power levels, 0dBm and -25dBm. The Loess curve fitting [43] is also plotted to show the variation pattern. Other packet sizes and power levels exhibit similar patterns.

From Fig. 3.4 and 3.5, we can have the following observations.

- ARR increases with PRR for most links, and on average ARR is higher than PRR.

- It can also be observed that no matter which packet size or transmission power level is used, there are always links with good PRR's but highly variable ARR's. Some nodes with near perfect PRR's still have poor ARR's even though the reverse channel is expected to be good after a packet transmission.

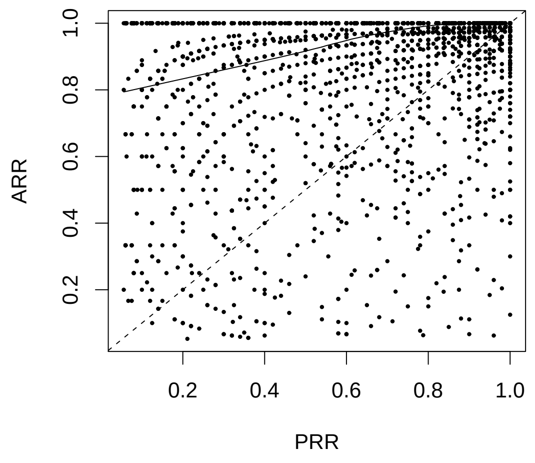
- The ECDF plots also suggest that there are significant number of links with imperfect acknowledgement reliability, where $15\% - 20\%$ links have ARR's lower than 80%.

- Compared to smaller packets (40 bytes) transmitted as the same power level, the reliability of acknowledgements are on average higher for larger packet size (126 bytes). This is because for larger packets to be received uncorrupted, they usually require better condition for the forward channel, and better forward channels usually correspond to better reverse channels.

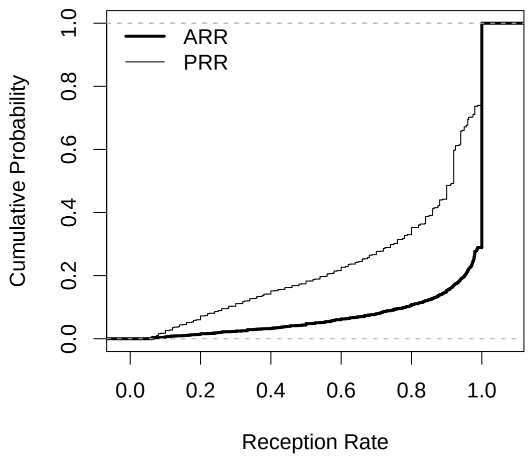
It is worthwhile noting that although the proportion of links with poor ARR is not high, even a small number of unreliable reverse links can cause performance degradation in MAC layer anycast protocols due to extra local retransmissions and duplicated forwarding.

### 3.5.2 Emulating the Motelab Testbed

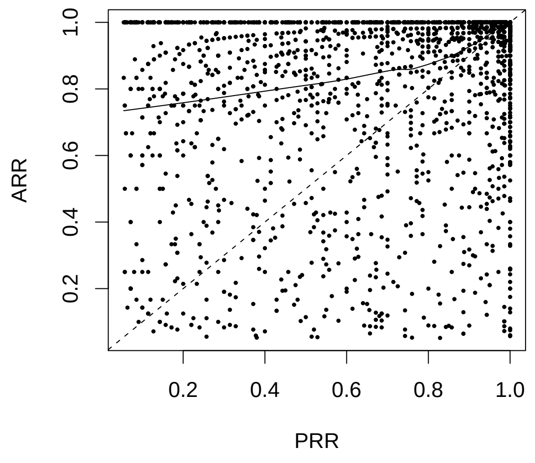
Using the link reliability data collected from the Motelab testbed (Section 3.5.1), we compare the EATX performance of our anycast construction with the two proposed metrics against with the anycast algorithm in [35] and unicast. The optimal solutions are also obtained and compared for certain cases. In summary, the evaluated algorithms include,



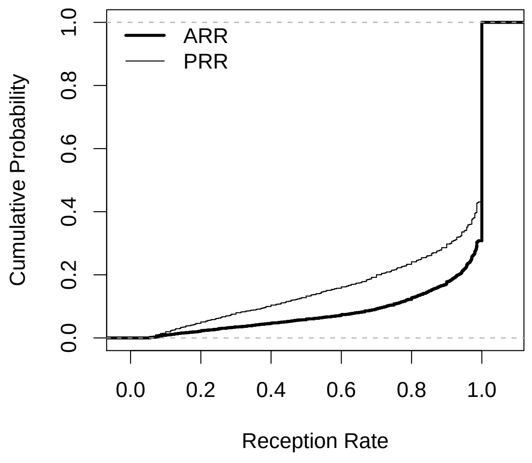
(a) Scatter plot (packet size: 126 bytes)



(b) ECDF (packet size: 126 bytes)

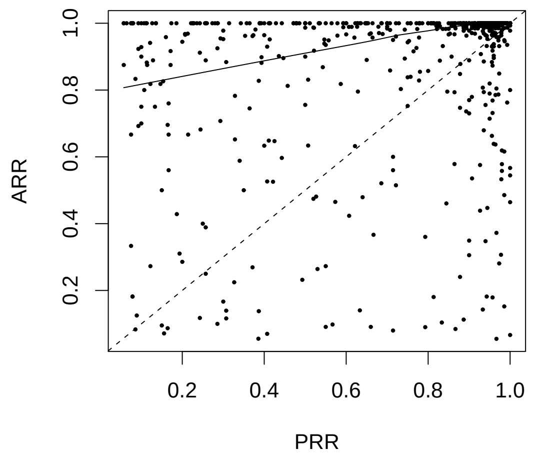


(c) Scatter plot (packet size: 40 bytes)

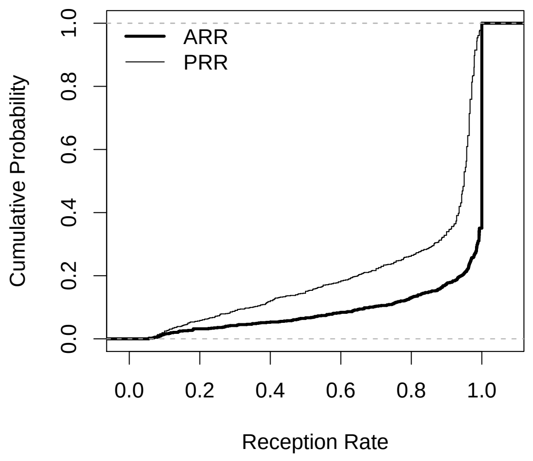


(d) ECDF (packet size: 40 bytes)

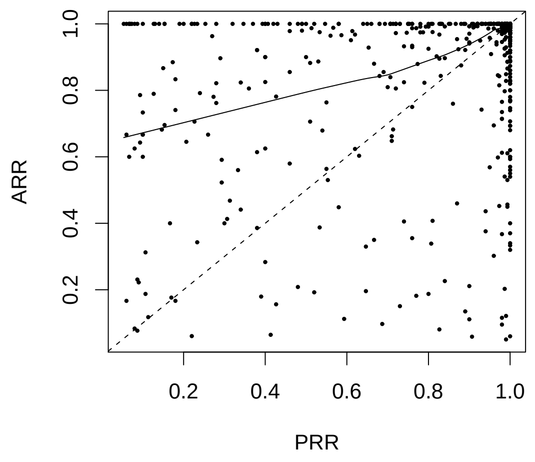
Figure 3.4: Data packet and acknowledgement reliability data from the Motelab testbed [65]. (Transmission Power Level: 0dBm.)



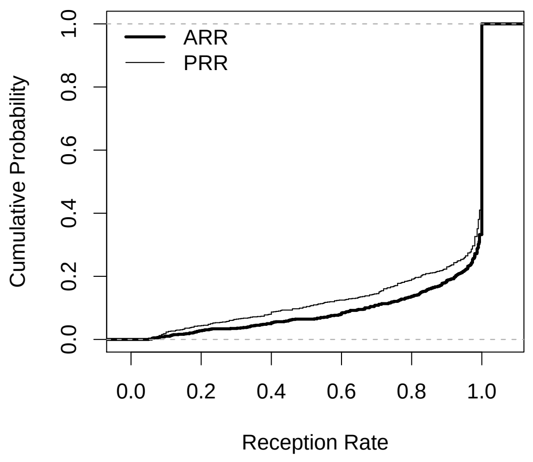
(a) Scatter plot (packet size: 126 bytes)



(b) ECDF (packet size: 126 bytes)



(c) Scatter plot (packet size: 40 bytes)



(d) ECDF (packet size: 40 bytes)

Figure 3.5: Data packet and acknowledgement reliability data from the Motelab testbed [65]. (Packet size: 40 bytes. Transmission Power Level: -25dBm.)

- **Anycast-EATX-1**: our anycast algorithm with $EATX + \frac{1}{ARR}$ as the metric in Algorithm 9.

- **Anycast-EATX-2**: our anycast algorithm with $EATX + \frac{1}{PRR \times ARR}$ as the metric in Algorithm 9.

- **Anycast-JK**: the anycast algorithm proposed by Kim et. al. in [35].

- **Anycast-Opt**: the optimal forwarding set computed by enumerating all non-empty subsets and all possible permutations.

- **Unicast**: distance-vector based unicast routing with ETX [20] as the routing metric.

All algorithms are evaluated by constructing exactly the same topology as in the Motelab testbed. Reliability for data packets and acknowledgements are retrieved from the experimental data described in Section 3.5.1.

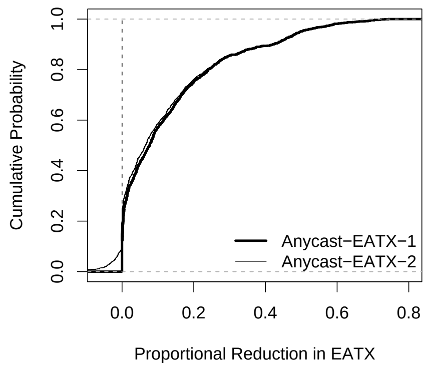
**Low Transmission Power**

On the Motelab testbed [65], because there are finite number of nodes deployed at fixed locations, the number of neighbor nodes decreases with the transmission power. In this section, we set the transmission power to the lowest level (-25dBm) supported by Tmote Sky motes [8], thus generating a topology with minimum average node degree. The simulation results for this topology are shown in Fig. 3.6, 3.7 and 3.8.

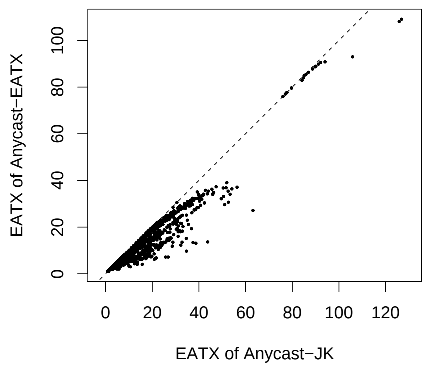
Fig. 3.6 shows the comparison among Anycast-EATX-1, Anycast-EATX-2 and Anycast-JK. The ECDF plot in Fig. 3.6(a) shows that compared to Anycast-JK, Anycast-EATX-1/2 bring reduction in EATX for more than 70% of the source-destination pairs. The median and mean reduction are about 10% and 20%, and the maximum reduction could be as high as 80%. When Anycast-EATX-1/2 both outperform Anycast-JK, they provide similar EATX reduction with Anycast-EATX-1 slightly performing better. Anycast-EATX-2 leads to higher EATX than Anycast-JK for less than 10% of the time, while Anycast-EATX-1 has only negligible number of such node pairs. Such a performance difference is attributed to the underestimation of reverse channel quality as discussed in Section 3.4.1. Fig. 3.7 and 3.8 show the scatter plots for Anycast-EATX-1/2 versus Anycast-JK, where it can be seen that most points are below the $y = x$ line, implying smaller number of required transmissions using our algorithm.

The comparison among Anycast-EATX-1/2 and unicast is shown in Fig. 3.7. Even though the scatter plots in Fig. 3.7(b) and 3.7(c) show slight improvement, Fig. 3.7(a) shows that for about 70% of the source-destination pairs, Anycast-EATX-1/2 performs better than unicast. The average saving is about 5%, and the maximum is about 15%. The marginal improvement for most cases is expected, since using transmission power level as low as -25dBm, nodes have small candidate pools and the link quality is typically poor. Due to the low degree, many nodes do not have more than one or two "good enough" candidates and thus the even optimal performance is close to unicast.

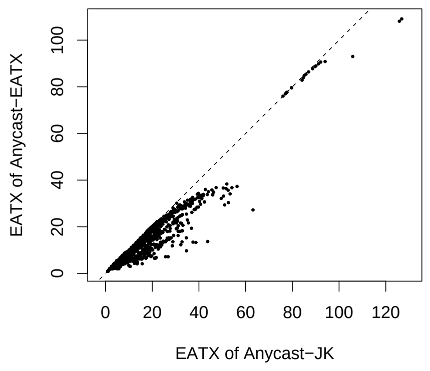
To understand what is the performance limit for MAC layer anycast, we calculate the optimal forwarding sets for nodes in the topology that have at most 5 nodes with smaller EATX values. By enumerating all subsets and their permutations, we calculate the optimal EATX values for them and compare the results with what we get from Anycast-EATX-1/2. Fig. 3.8(a) shows the ECDF functions for the percentages of extra transmissions needed compared to the optimal results. It can be seen that



(a) Anycast-EATX-1/2 vs. Anycast-JK

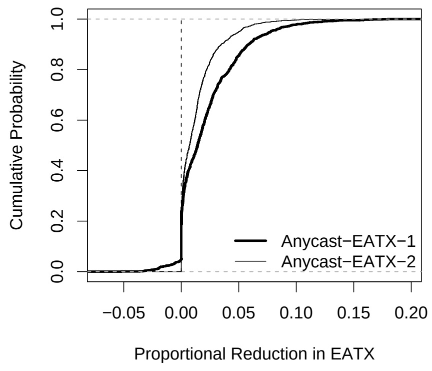


(b) Anycast-EATX-1 vs. Anycast-JK

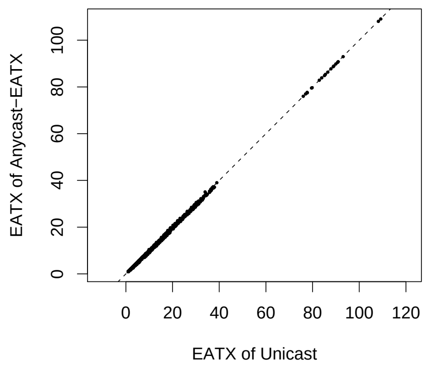


(c) Anycast-EATX-2 vs. Anycast-JK

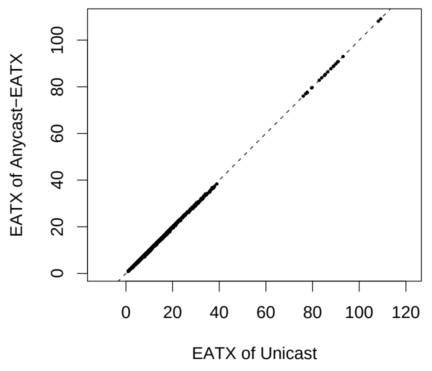
Figure 3.6: Comparison of Anycast-EATX and Anyast-JK (packet size: 40 bytes; power: -25dBm).



(a) Anycast-EATX vs. Unicast



(b) Anycast-EATX-1 vs. Unicast



(c) Anycast-EATX-2 vs. Unicast

Figure 3.7: Comparison of Anycast-EATX and unicast (packet size: 40 bytes; power: -25dBm).

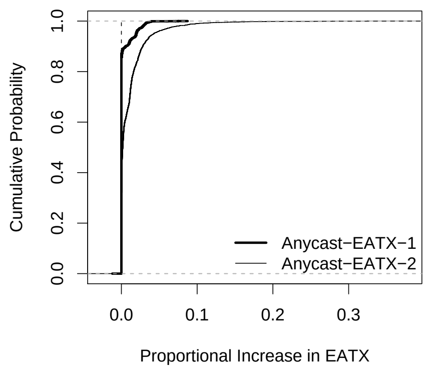
for nearly 90% of these node pairs, Anycast-EATX-1 achieves optimal results, and for nearly 100% of time Anycast-EATX-1 uses less than 5% extra transmissions. As for Anycast-EATX-2, it performs the same as optimum in 50% of the cases, but for 99% of the cases it uses only 10% more transmissions. The close performance of Anycast-EATX-1/2 and optimum also explains the closeness of Anycast-EATX-1/2 and unicast.

Overall, when the transmission power level is low, Anycast-EATX-1 performs better than Anycast-EATX-2, while both of them perform better than Anycast-JK and unicast for majority of the cases. Both Anycast-EATX-1 and Anycast-EATX-2 perform close to the optimum. For large packet sizes, we also conducted similar simulations. The results are similar and the ECDF plots of all comparisons are shown in Fig. 3.9.

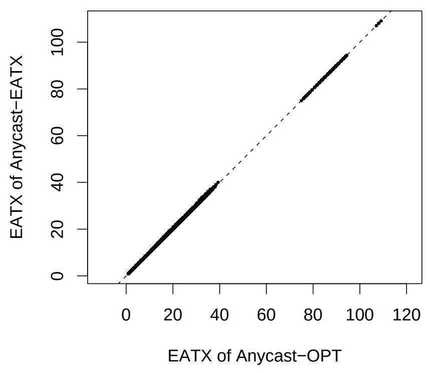
### High Transmission Power

In this section, we use the data for the highest transmission power level, 0dBm, supported by Tmote Sky motes [8]. In the resulting topology nodes have maximum achievable degrees. The simulation results for this topology and 40-byte packets are shown in Fig. 3.10, 3.11 and 3.12.

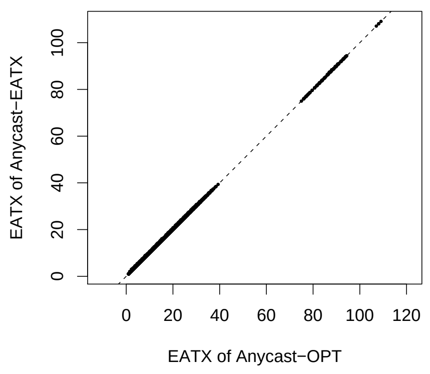
Fig. 3.10 shows the comparison among Anycast-EATX-1/2 and Anycast-JK. The ECDF plot in Fig. 3.10(a) shows that compared to Anycast-JK, Anycast-EATX-1/2 brings reduction in EATX for more than 50% of the source-destination pairs. The median and mean reduction are also about 10% and 20%, and the maximum reduction could be as high as 80%. When Anycast-EATX-1/2 both outperform Anycast-JK, they provide similar EATX reduction with Anycast-EATX-1 slightly performing better. Anycast-EATX-2 leads to higher EATX than Anycast-JK for less than 10% of the



(a) Anycast-EATX-1/2 vs. Anycast-Opt

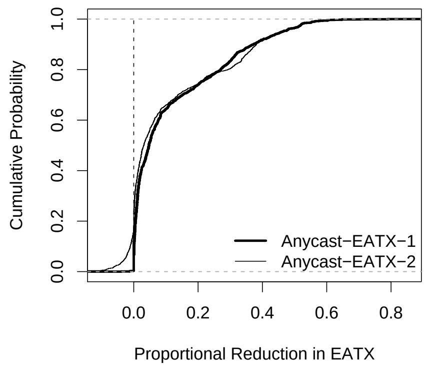


(b) Anycast-EATX-1 vs. Anycast-Opt

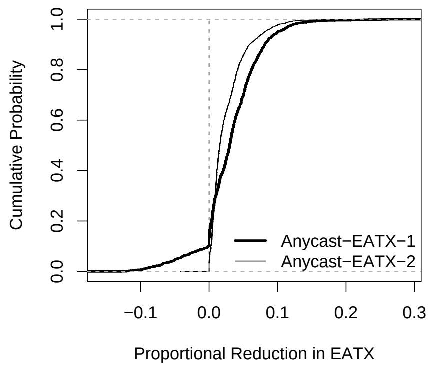


(c) Anycast-EATX-2 vs. Anycast-Opt

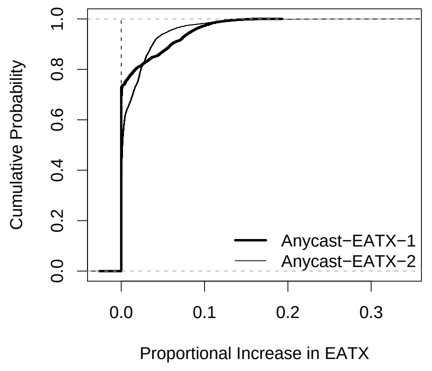
Figure 3.8: Comparison of Anycast-EATX-1/2 and optimum (packet size: 40 bytes; power: -25dBm).



(a) Anycast-EATX-1/2 vs. Anycast-JK



(b) Anycast-EATX-1 vs. Unicast



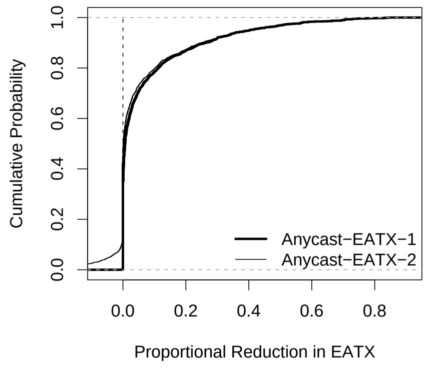
(c) Anycast-EATX-2 vs. Anycast-Opt

Figure 3.9: Comparison of Anycast-EATX-1/2 to Anyast-JK, unicast and optimum (packet size: 126 bytes; power: 0dBm).

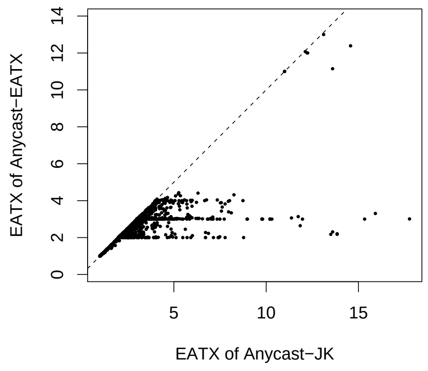
cases, while Anycast-EATX-1 has only negligible number of such node pairs. Such a performance difference can still be attributed to the underestimation of reverse channel quality as discussed in Section 3.4.1. Fig. 3.11 and 3.12 show the scatter plots for Anycast-EATX-1/2 versus Anycast-JK, where it can be seen that most of the points are below the $y = x$ line implying smaller number of required transmissions using our algorithm. Compared to the results for transmission power of -25dBm (Fig. 3.6(b) and 3.6(c)), the scatter plots show that Anycast-EATX-1/2 can achieve greater advantages for more nodes by leveraging multiple choices in the neighborhood.

The comparison among Anycast-EATX-1/2 and unicast is shown in Fig. 3.11. Compared to the results for transmission power level at -25dBm, the scatter plots in Fig. 3.11(b) and 3.11(c) show higher improvement. The percentage of source-destination pairs benefiting from Anycast-EATX-1/2 is about 50%, while the maximum savings in EATX can be as high as 60% (Fig. 3.11(a)). An important reason for the difference is the high node degree. Fig. 3.11(a) shows that Anycast-EATX-1 brings more benefit than Anycast-EATX-2. This is because Anycast-EATX-1 does not underestimate ARR as in Anycast-EATX-2 which couples ARR and PRR together.

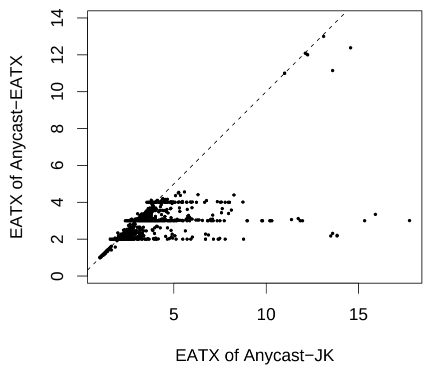
We calculate the optimal forwarding sets for nodes with at most 5 neighbors with smaller EATX values, and the results are compared with those of Anycast-EATX-1/2. Fig. 3.8(a) shows the ECDF functions for the percentages of extra transmissions needed compared to the optimum. It can be seen that for nearly 80% of the source-destination pairs, Anycast-EATX-1 achieves optimal results, and for 100% of the cases Anycast-EATX-1 uses less than 5% extra transmissions. As for Anycast-EATX-2, it performs much worse than Anycast-EATX-1 in this case. Anycast-EATX-2 can



(a) Anycast-EATX-1/2 vs. Anycast-JK

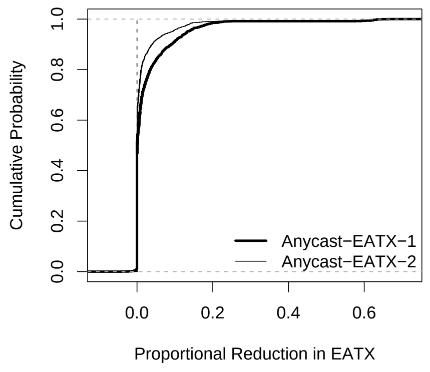


(b) Anycast-EATX-1 vs. Anycast-JK

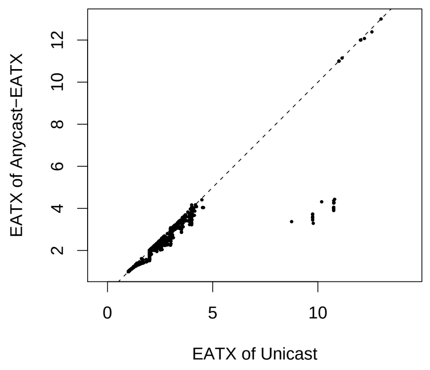


(c) Anycast-EATX-2 vs. Anycast-JK

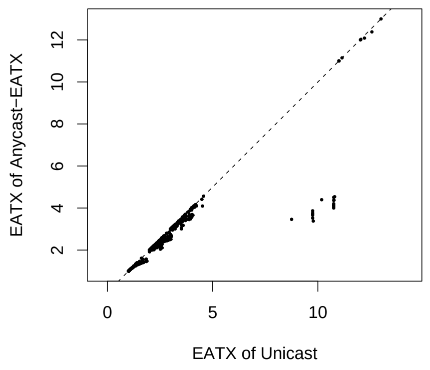
Figure 3.10: Comparison of Anycast-EATX and Anyast-JK (packet size: 40 bytes; power: 0dBm).



(a) Anycast-EATX-1/2 vs. Unicast



(b) Anycast-EATX-1 vs. Unicast



(c) Anycast-EATX-2 vs. Unicast

Figure 3.11: Comparison of Anycast-EATX and unicast (packet size: 40 bytes; power: 0dBm).

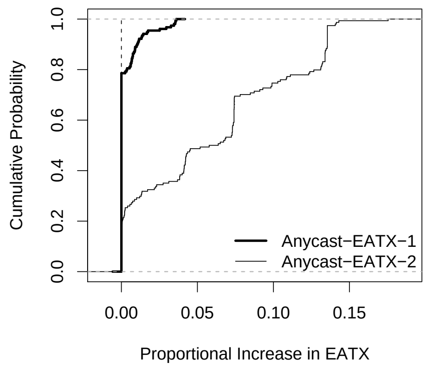
achieve optimum for only 20% of the cases, but it still uses less than 20% more transmissions than optimum. In Fig. 3.12(b) and 3.12(c), the performance of Anycast-EATX-1/2 is still very close to the $y = x$ line, but the difference is visually noticeable.

Overall, in the case of high transmission power, Anycast-EATX-1 performs better than Anycast-EATX-2, while both of them performs better than Anycast-JK and unicast for majority of the source-destination pairs. In such cases, Anycast-EATX-1/2 can also perform closely to the optimum. For large packet sizes, we also conducted similar simulations. The results are similar and the ECDF plots of all comparisons are shown in Fig. 3.13.

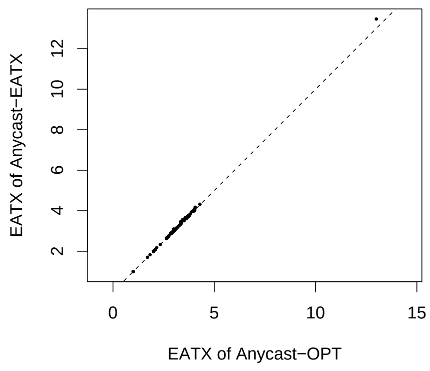
### 3.5.3 Random Topology

On the Motelab testbed 3.5.2, changing the transmission power level can result in different node densities in the neighborhood, but the network size is also reduced at the same time. In this section, we evaluate the performance of our algorithm framework using randomly generated topologies. The data packet loss probabilities are generated using the model in [78]. To generate acknowledgement loss probabilities in compliance to the data collected from the Motelab testbed (Section 3.5.1), we evenly divide the interval (0,1) into 20 subintervals, and then the reverse link data are placed into the 20 subintervals according to the corresponding forward link reliability data. For example, if a link has 0.33 and 0.8 forward and reverse link reliability respectively, it is put into the 7-th interval (0.3, 0.35].

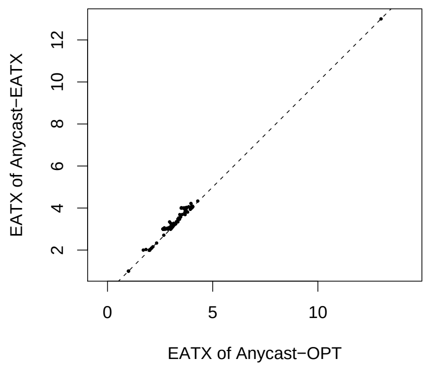
The evaluations for different densities are all conducted in a $50m \times 50m$ area with transmission range of $10m$. In this section, we present the results for densities of 5 and 9 nodes per $10m \times 10m$. To reduce the impact of randomness on the results, we



(a) Anycast-EATX-1/2 vs. Anycast-Opt

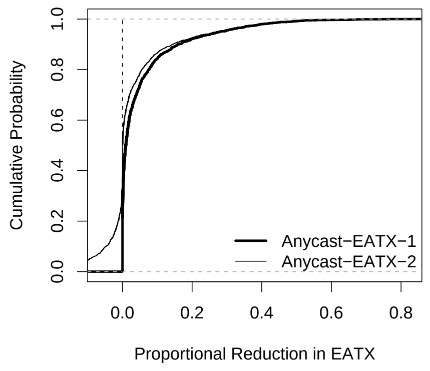


(b) Anycast-EATX-1 vs. Anycast-Opt

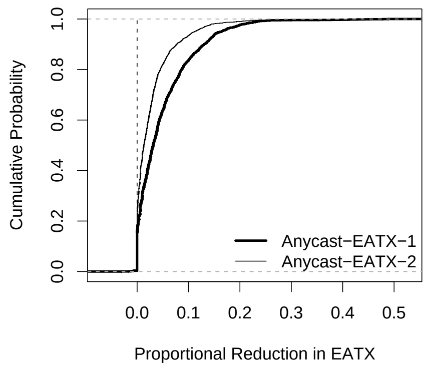


(c) Anycast-EATX-2 vs. Anycast-Opt

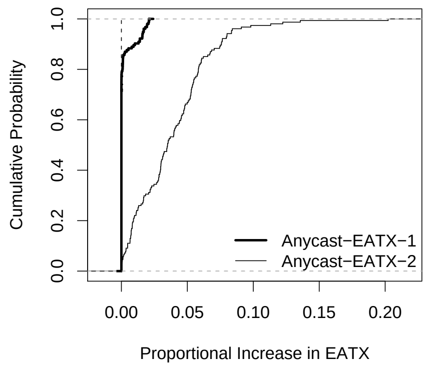
Figure 3.12: Comparison of Anycast-EATX and optimum (packet size: 40 bytes; power: 0dBm).



(a) Anycast-EATX-1/2 vs. Anycast-JK



(b) Anycast-EATX-1 vs. Unicast



(c) Anycast-EATX-2 vs. Anycast-Opt

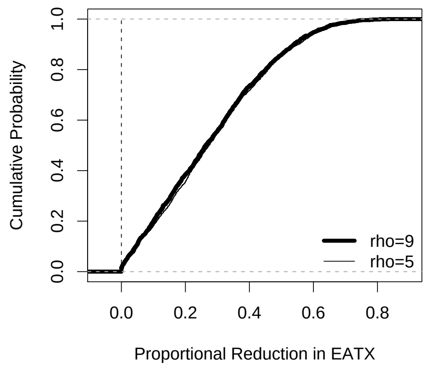
Figure 3.13: Comparison of Anycast-EATX to Anyast-JK, unicast and optimum (packet size: 126 bytes; power: -25dBm).

use the same 10 random seeds to generate topologies for all algorithms, and collect the data for all runs.

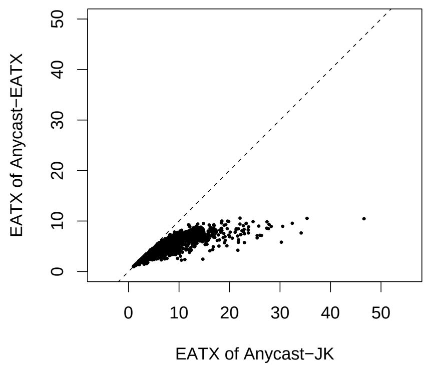
For the clarity of figures, we omit the performance curves of Anycast-EATX-2 in this section as it performs worse than Anycast-EATX-1 most of the time. Figs. 3.14 to 3.16 show the results. It can be observed that the improvement over Anycast-JK is significant and similar for both densities (5 and 9). But when compared to unicast, higher density leads to more improvement due to the existence of more candidates in the neighborhood. For the comparison with optimum, higher density also leads to performance closer to optimum, but all of them perform within 110% of the optimum.

## 3.6 Conclusion and Future Work

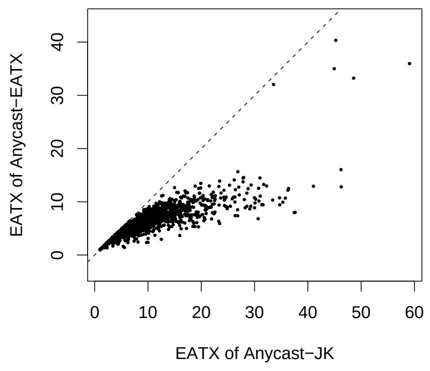
Existing MAC layer anycast protocols in the literature fail to consider the impact of acknowledgement losses, and thus lead to duplicated forwarding along multiple paths to the sink. In this chapter, we identify this problem and use experimental data to show that using the anycast protocols that are reverse channel unaware like in [35], the number of transmissions taken by the entire network to deliver one packet can be significantly more than unicast. For energy constrained wireless sensor networks, such duplicated transmissions are wasteful. In this chapter, we propose to use the expected number of end-to-end transmissions (EATX) to take the reverse link into consideration, and provide the formulation of the EATX metric. Based on this metric, we propose an anycast forwarding set selection algorithm which works by prioritize neighbor nodes in company with a distributed anycast route construction protocol. Through simulations driven by the data collected on the Motelab testbed, we evaluate two criteria based on the formulation of the EATX metric. The results show that



(a) Anycast-EATX-1/2 vs. Anycast-JK

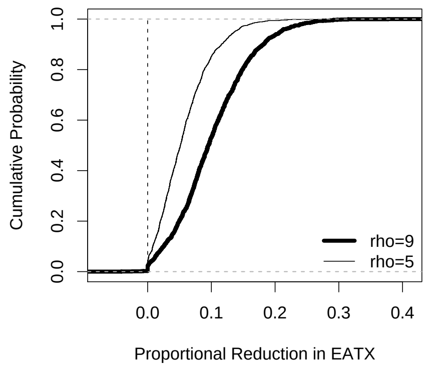


(b) Anycast-EATX-1 vs. Anycast-JK

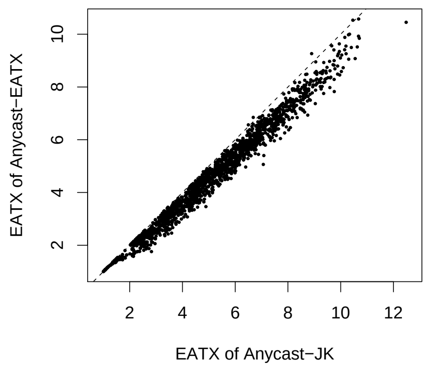


(c) Anycast-EATX-2 vs. Anycast-JK

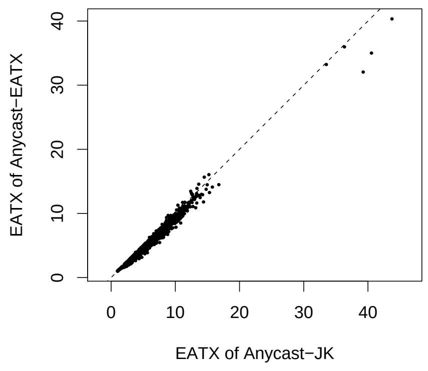
Figure 3.14: Comparison of Anycast-EATX and Anyast-JK in random topologies with different node densities.



(a) Anycast-EATX-1/2 vs. Unicast

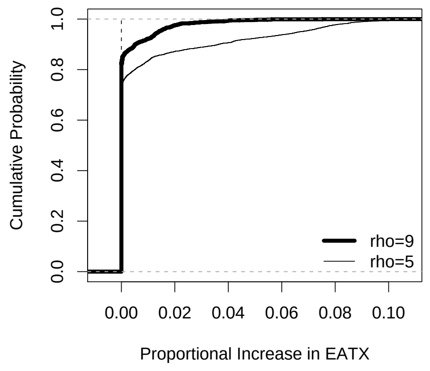


(b) Anycast-EATX-1 vs. Unicast

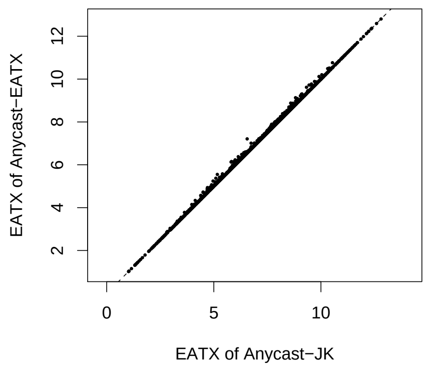


(c) Anycast-EATX-2 vs. Unicast

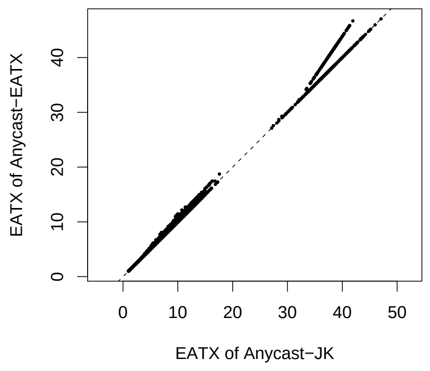
Figure 3.15: Comparison of Anycast-EATX and unicast in random topologies with different node densities.



(a) Anycast-EATX-1/2 vs. Anycast-Opt



(b) Anycast-EATX-1 vs. Anycast-Opt



(c) Anycast-EATX-2 vs. Anycast-Opt

Figure 3.16: Comparison of Anycast-EATX and optimum in random topologies with different node densities.

our algorithm performs better than unicast and the anycast algorithm in [35] for a majority of cases. We also present results showing that our algorithmic framework works close to the optimum. Especially when the criterion in Formula (3.8) is used, the selected forwarding sets agree with the optimum for at least 80% of evaluated cases.

The proposed algorithmic framework in this chapter assumes the knowledge of PRR and ARR to each neighbor node, but in a network with these parameters change over time, it is still an open problem to adapt quickly to such dynamics. Network dynamics may be caused by varying traffic patterns, node deployments and failures, and significant changes in link quality. Our current solution (Section 3.4.2) depends on broadcasting advertisements and to construct new forwarding sets in such cases, but the speed of convergence for large scale networks is still a challenging issue to address.

# CHAPTER 4

# Achieving Energy Efficiency with Transmission Pushbacks

## 4.1 Introduction

Energy consumption is one of the key considerations in the design of multi-hop wireless sensor networks. Since a high percentage of the energy is spent on data communication, support for efficient and reliable communication is critical. However, high variability in channel quality caused by factors such as fading, mobility and time-varying multiuser interference makes it difficult to achieve those objectives. Indeed, Woo et al. [69], and Zhao and Govindan [74] have both observed a significant variability in link quality in wireless sensor networks. The former paper points out that the instantaneous packet error probability varies by approximately 30% around its mean. The latter paper and [66] both show that the packet-error stochastic process exhibits significant long-term dependence.

Without any effort for adapting to the variability, the system resources are consumed highly inefficiently. Due to high packet loss rates, a large fraction of the energy of a node is consumed by multiple retransmissions per packet. However, in the currently available sensor hardware platforms, the limited computational power rules out sophisticated control actions for adaptation. Only very simple strategies (e.g., transmit or do not transmit a packet at a given time) are implementable.

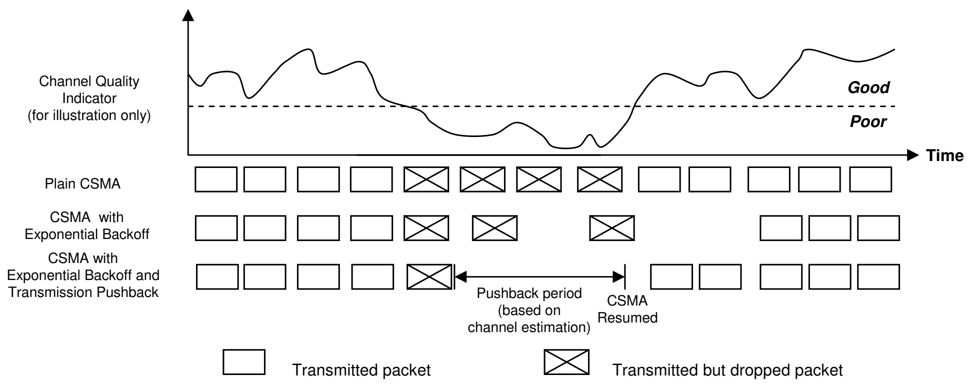

Channel Quality Indicator (for illustration only)
Good
Poor
Time
Plain CSMA
CSMA with Exponential Backoff
CSMA with Exponential Backoff and Transmission Pushback
Pushback period (based on channel estimation)
CSMA Resumed
Transmitted packet
Transmitted but dropped packet

Figure 4.1: Avoid periods with poor link quality using transmission pushbacks.

The prevailing CSMA protocol uses carrier sensing to avoid collisions and backoffs to address the problem of contention among nearby nodes. However, packet transmissions may fail due to cumulative interference from other nodes in the network. Indeed in our testbed experiments with XSM nodes [21], we have observed that with interfering sources that are sufficiently far away, 69% of the packets which CSMA grants to transmit are lost. From this example, we can conclude that the combined effect of a large number of interfering sources can be detrimental and the CSMA based protocols – designed to suppress collisions – are not effective in avoiding such losses. An immediate solution to this problem is to reduce the carrier sense threshold that triggers a backoff, and consequently increase the carrier sense range. This, in effect, would increase the capability of sensing interference and reduce packet losses. However, the increase of carrier sense range makes a node overly conservative with respect to interference and leads to lower effective throughput. Therefore, simple adjustment of the carrier sense range is not sufficient and a separate mechanism is needed over CSMA to avoid transmissions when the channel condition is poor.

In this chapter, we systematically study the problem of addressing packet losses due to cumulative interference, and propose a binary control technique over CSMA. Our approach is based on exploiting the temporal correlations of the interference process. We introduce a new concept called *transmission pushbacks*, which refers to an appropriately computed delay introduced at the MAC layer in order to avoid periods with bad-channel quality while considering a node's throughput requirement. Therefore, we reduce the number of transmissions per packet as well as the number of transmission attempts per unit time. In case of bursty losses, avoiding the bad channel state may also lead to a higher throughput (visible at higher layers) despite fewer number of transmission attempts.

The main idea of transmission pushbacks is to defer packet transmission attempts for an appropriately selected period upon failed packet transmissions. Fig. 4.1 illustrates the benefits of using transmission pushbacks in comparison with CSMA based approaches in the presence of time-varying channels. Plain CSMA leads to failed transmissions, and thus wastes energy, during periods with poor channel quality. CSMA with exponential backoff may reduce such failed transmissions, but it also cuts down the transmission attempts, even at times of improved channel quality. Our proposed transmission pushback mechanism predicts the duration for which the channel quality will remain poor. Thus, unnecessary transmissions can be avoided to conserve energy and the good channel states are taken advantage of.

To determine the pushback time, we need to estimate the channel quality and how it varies over time. We use an adaptive channel prediction technique, based on estimating the parameters of a simple hidden Markov model (HMM), which represents our channel. We dynamically update the parameters of the HMM based solely on the binary ACK sequence (transmission success or failure) for the previous packet transmissions. We choose the appropriate pushback period by considering the throughput requirement measured by the incoming data rate, and the predicted quality of the channel. Such an adjustment in rate, based on the throughput requirement is also seen in lazy packet schedulers [50]. The proposed approach is simple to implement over existing CSMA based MAC solutions, as well as queue and congestion control algorithms. Therefore it is highly suitable for existing sensor network platforms.

The design of the transmission pushback mechanism makes the following contributions:

- Using data collected from a sensor network testbed, temporal characteristics of the channel variations and the interference are studied.

- A novel concept called *pushbacks* is introduced, that is used to increase the packet success rate while considering the throughput constraint at each node.

- Through simulations it is shown that significant gains in energy and/or throughput can be observed in all scenarios using the proposed technique.

The rest of this chapter is organized as follows. Section 4.2 summarizes related work. Section 4.3 presents our approach to model the channel losses. Section 4.4 presents a description of our pushback algorithm. Section 4.5 presents evaluation of the proposed scheme. Finally, Section 4.6 concludes this chapter and presents pointers to future research directions.

## 4.2 Related Work

**Transmission Rate Adaptation:** Transmission strategies based on channel estimation has been considered in the context of 802.11 networks [34, 31, 56, 68]. More specifically, the past packet success and failure reports have been used to design strategies to dynamically adapt the physical layer transmission rate to optimize the throughput. ARF [34] uses a heuristic to predict the channel quality based on past transmission success and failure records, but it is ignorant of the underlying time-varying properties of the channel. RBAR [31] uses RTS/CTS to get immediate feedback from the destination to learn about the quality of the channel and determine the transmission rate. However, these packets have high overhead especially in sensor networks where the data packet sizes are comparable to the size of RTS/CTS packets. In OAR [56] during good channel periods the transmitter opportunistically transmits multiple packets back-to-back at a high data rate. In contrast to the opportunistic nature of [56], our solution uses a rigorous channel model to predict the duration for which the channel will continue in a poor state. In [68], authors present a history based mechanism to predict the quality of the channel and adjust the transmission rate. Our objective of optimizing energy consumption for a given throughput constraint is different from the past work. Our solution methodology is also different as it uses *rigorous estimation of dynamic channel properties.*

In order to address high interference and network congestion, several back-pressure based mechanisms have been proposed for sensor networks [64, 32, 52, 37]. CODA [64] uses a moving average of channel samples to detect the onset of congestion and sends back-pressure messages to control the data rate of upstream nodes. Fusion [32] uses the concept of hop-by-hop flow control and prioritized MAC along with token buckets to control the packet rate at each node. IFRC [52] and RAIN [37] use variations of the AIMD (Additive Increase Multiplicative Decrease) and the back-pressure mechanisms. Various schemes have been proposed that select the backoff counter [18] and the backoff window [45] based on the channel state.

We conclude this section by observing that these solutions are unaware of the time-varying characteristics of the channel, and hence still do blind retransmissions ignorant of current channel condition. In addition, MAC protocols with backoffs tuned to channel conditions do not address the time-scales associated with channel variations as shown in Figure 4.1. On contrary, the pushback algorithm proposed in this chapter can learn the channel characteristics and schedule retransmissions accordingly. In fact, our solutions can be used in conjunction with the above link layer mechanisms and bring extra benefit as shown in our simulation results.

## 4.3 Channel Modeling

In this section we describe our channel loss model and its parameters and give an experimental justification for our model. We use this channel loss model to derive the theoretical expressions for Packet Success Ratio (PSR) and throughput as functions of channel and system parameters. We describe how to estimate these parameters based on the available measurements at the sensor nodes in the next section.

### 4.3.1 Channel Model

One way to model (bit or packet level) errors in wireless communication channel is to use Hidden Markov Models (HMMs) [28, 26, 62]. While not as simple as a Bernoulli or an independent loss channel model, Markov models are more capable of characterizing the statistical dependencies which might occur in a wireless channel.

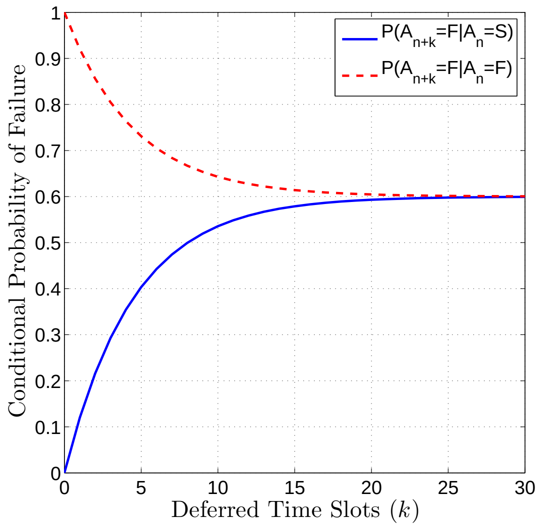


Figure 4.2: Conditional probability of failure as a function of deferred time slots setting $p = 0.6$ and $\alpha = 0.8$.

The Gilbert-Elliott channel model [28] was first used to derive the bit-level channel model. This model is a Markov chain which has two states namely, "Good" and "Bad", which have fixed channel characteristics, respectively. To analyze the channel, we use a modified version of the Gilbert-Elliott channel model with unknown channel parameters for the Good and Bad states. In addition, the transition probabilities are unknowns to be specified.

At this level of generality, given a set of channel observations, the calculation of these unknown parameters requires a substantial amount of computation. To simplify the model, we assume that the statistics of the packet error stochastic process is *wide-sense Markov* of order 1. A sequence $A_n, n \geq 1$ is said to be wide-sense Markov if the probability of an event $A_{n+m}$ is completely determined by its most recent value, i.e. $A_n$ and the time difference $(m)$ between the two events. The autocovariance structure for such a process is exponential. More specifically, if the unconditioned packet loss probability is $p$, then the autocovariance function of $A_n$ is defined as $K_A(m) = p(1 - p)\alpha^{|m|}$. For this wide-sense Markov process, the probability of successful (failed) transmission in a future time slot, conditioned on successful (failed) transmission in the present slot, is unique (Appendix I in [40]):

$$\begin{aligned} P(A_{n+m} = \mathrm{F} | A_n = \mathrm{S}) &= p(1 - \alpha^m) && (4.1) \\ P(A_{n+m} = \mathrm{S} | A_n = \mathrm{S}) &= 1 - p(1 - \alpha^m) && (4.2) \\ P(A_{n+m} = \mathrm{F} | A_n = \mathrm{F}) &= p + (1 - p)\alpha^m && (4.3) \\ P(A_{n+m} = \mathrm{S} | A_n = \mathrm{F}) &= 1 - p - (1 - p)\alpha^m, && (4.4) \end{aligned}$$

where $A_n$ is the event of a success ('S') or a failure ('F') in transmission, in the $n$th time slot; $p$ and $\alpha$ are the temporal parameters of the model to be estimated. For a multi-hop wireless network, $p$ is the parameter that indicates the effect of the total number of interferers for a given link.

For a larger number of interfering users, we will have a higher value of $p$ and consequently a higher failure probability due to interference. On the other hand, $\alpha$ is representative of the average burst length of the interfering users. Longer burst lengths will lead to larger values of $\alpha$ as there will be a stronger correlation between probability estimates over longer periods. For the purpose of illustration, the conditional probabilities of failure for $p = 0.6$ and $\alpha = 0.8$ are plotted in Fig. 4.2. The "deferred time slots" represents the number, $k$, of time slots waited after an event S or F. Here, one time slot is the duration it takes to transmit one packet over the channel.

Note that we reduced the number of unknown parameters to two with the first order Markov process. Compared to the aforementioned model, in our case a successful transmission corresponds to the Good state and a failed transmission corresponds to a Bad state. Now the model has become numerically tractable and flexible enough to handle the channel model.

From the two curves in Fig. 4.2, one can see the reasoning behind choosing a pushback duration conditional on an event F only. If we schedule the next packet for immediate transmission (i.e., $k = 1$) after an S, we have the best chance of observing another S. Intuitively, we are taking advantage of the good channel state. On the other hand, if we defer scheduling the transmission (i.e.,$k > 1$) of the next packet after an F, we lower the probability of failing in that transmission. The longer the deferral time, the higher the probability of an S in the next transmission. However, a long wait can cause the throughput to drop. Thus we need to strike a balance between the two requirements, throughput and probability of success. Hence, with the Markov channel model, the problem reduces to finding the appropriate pushback period after a failed transmission. In the next subsection, we will derive the expressions for the throughput and packet success rate for this scheme using our channel model.

### 4.3.2 Channel Parameters

To find the channel parameters, packet success ratio and throughput, we solve the Markov chain associated with our first order Markov process. Our main objective here is to find the throughput as a function of the number of deferred time slots, $k$, on a transmission failure[5]. Once we have this expression, we can choose $k$ according

[5]Upon a successful transmission, the deferral time is 1 (no deferral).

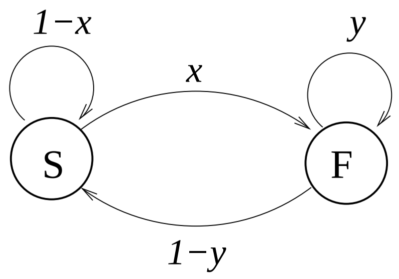


Figure 4.3: Markov chain representation of the channel.

to the desired throughput based on the incoming data rate. To validate this model using real data, we also find the expression for the PSR.

Our Markov chain has two states, S and F as illustrated in Fig. 4.3. The current state is S if the final packet transmission is successful and F, otherwise. Note that a transition does not necessarily occur in every time slot, rather it occurs for every packet transmission attempt. Since we schedule a transmission immediately after a successful event, the expression for $x$ is obtained by substituting $k = 1$ in (4.1). Direct application of (4.3) gives the expression for $y$.

$$x = P(A_{n+1} = \mathrm{F}|A_n = \mathrm{S}) = p(1-\alpha) \tag{4.5}$$

$$y = P(A_{n+k} = \mathrm{F}|A_n = \mathrm{F}) = p + (1-p)\alpha^k \tag{4.6}$$

Notice that the transition probabilities at state F are functions of $k$ as well as $p$ and $\alpha$. This is due to the effect of the pushback period of $k$ time slots after the failed transmission attempts. The associated steady state probabilities are therefore functions of $k$ as well and these probabilities for state S and state F are respectively,

$$\begin{aligned} \pi_{\mathrm{S}}(k) &= \frac{(1-p)(1-\alpha^k)}{p(1-\alpha)+(1-p)(1-\alpha^k)} \qquad (4.7) \\ \pi_{\mathrm{F}}(k) &= 1-\pi_{\mathrm{S}}(k). \end{aligned}$$

We define the packet success ratio (PSR) as the total fraction of the packets that are successfully transmitted, i.e., it is equal to the steady state probability, $\pi_{\mathrm{S}}(k)$ of state S.

To formulate an expression for throughput, consider the following: on a transmission attempt, the sender waits for $k$ time slots in state F and 1 time slot in state S for the next transmission attempt. Thus, the average number of slots per attempt is $\pi_S(k) + k\pi_F(k)$. Consequently the number of packet transmissions per slot,

$$X(k) = \frac{1}{\pi_S(k) + k\pi_F(k)} = \frac{p(1-\alpha) + (1-p)(1-\alpha^k)}{kp(1-\alpha) + (1-p)(1-\alpha^k)}. \tag{4.8}$$

The resulting throughput, $\rho(k)$, is thus

$$\begin{aligned} \rho(k) &= \pi_{\mathrm{S}}(k)X(k) \\ &= \frac{(1-p)(1-\alpha^k)}{kp(1-\alpha) + (1-p)(1-\alpha^k)}. \end{aligned} \tag{4.9}$$

We tested the validity of our Markov model using an experimental setup in the Kansei testbed [9] by setting up a wireless link between two sensor motes. This link was encircled with seven sensor motes spread around the perimeter of a 20 m$^2$ area. We measured the packet success rate between two motes, while the rest of the motes acted as sources of interference. For this experiment, the wireless link transmitted 36 byte packets at 100 ms interval. We programmed the nodes causing interference to transmit bursts of packets once every second. The burst size was selected according to a uniform distribution between [0,15]. We estimated the two parameters, $\alpha$ and $p$ using the ACK-level data of the entire trace of 30 mins. Note that the purpose of this experiment was to validate the model. In our actual algorithm, the estimation of the two parameters is much simpler and does not depend on a long trace of data. The theoretical packet success rate based on the estimated $\alpha$ and its actual experimental

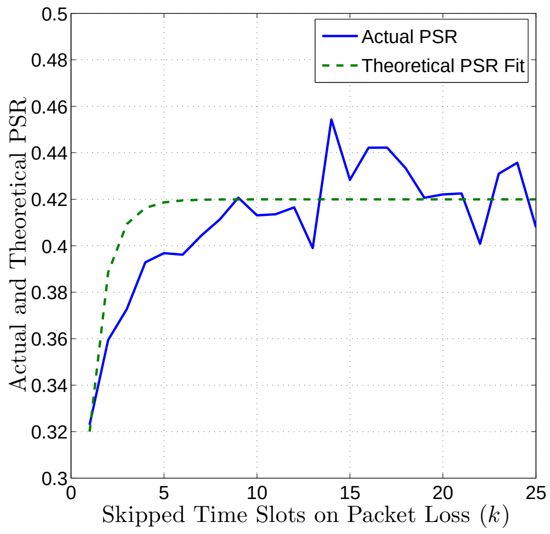


Figure 4.4: Comparison of the actual PSR gain achieved by our pushback algorithm and the theoretical fit using Eqn. (4.7).

value are plotted in Fig. 4.4 as a function of $k$. This plot gives credence to the Markov model.

## 4.4 The Pushback Algorithm

The objective of the pushback algorithm is to estimate the period for which the channel will remain in a poor state and defer retransmissions accordingly in order to conserve energy. In addition, the algorithm must provide similar throughput as CSMA for a reduced number of transmission attempts.

The proposed pushback algorithm is based on CSMA. If a transmission is successful, the next transmission is scheduled by CSMA. However, in case of a failed transmission, the next transmission is pushed back by $k$ slots. Note that the pushback "slot" is the time it takes to transmit a packet and therefore it is different from the contention slot of CSMA. When nodes boot up, an initial value $k_{\text{init}}$ is assigned to $k$, and then $k$ is recalculated periodically each time the $m$-th (predefined) transmission failure happens using a simple mechanism based on the estimates of the channel parameters and throughput constraint. Our computation is both practical to implement and is also shown to perform well using simulations in Section 4.5.

The proposed mechanism for computing $k$ is based on the formulation presented in Section 4.3.1. First, $k$ is set to an initial value $k_{\text{init}}$, and based on the ACK-level observations of success and failure in the recent past, a maximum likelihood (ML) estimation of parameters $\alpha$ and $p$ is made. In fact, estimating the transition probabilities of the Markov chain (parameters $x$ and $y$ in Fig. 4.3) is sufficient for the desired ML estimation. Indeed, given the ML estimates $\hat{x}$ and $\hat{y}$ (of $x$ and $y$ respectively), we show in Appendix B in [39] that the solution $(\hat{\alpha}, \hat{p})$ to the system of Eqns (4.5) and (4.6) gives the ML estimates of $\alpha$ and $p$ which characterize the channel. In addition, to ensure that a node can sustain the incoming rate of packets, the computation of $k$ must take the incoming packet rate into account. We use the running average of the incoming packet rate, $\rho_{\text{new}} = \gamma/t_e + (1-\gamma)\rho_{\text{old}}$, to represent the throughput constraint. Here $\rho_{\text{new}}$ ($\rho_{\text{old}}$) is the new (previous) estimate of the throughput requirement, $t_e$ is the time elapsed since the last packet arrival, and $\gamma$ is the smoothing factor. Using $\rho_{\text{new}}$ (throughput constraint), $\hat{\alpha}$ and $\hat{p}$, Equation (4.9) can be used to compute the new value of $k$.

To avoid the complexity of direct computation of $k$, we propose the use of look-up tables. The first table $T_{\hat{\alpha}}(x, y, k)$ contains the values of $\hat{\alpha}$ corresponding to $k$, and discretized $x$ and $y$. The second table $T_k(\hat{p}, \hat{\alpha}, \rho)$ contains the values of $\rho(k)$ corresponding to $k$, and discretized $\alpha$ and $p$. A brief description of the two tables

| Table | Equations | Purpose |
|---|---|---|
| $\mathrm{T}_\alpha(x, y, k)$ | Eqns. (4.5), (4.6) | To compute $\alpha$ for given $x$, $y$ and $k$. |
| $\mathrm{T}_k(p, \alpha, \rho)$ | Eqn. (4.9) | To compute $k$ for given $p$, $\alpha$ and $\rho$. |

Table 4.1: Lookup tables used in the pushback algorithm.

used is given in Table 4.4. These tables will not change during the operation of the node, so they can be computed offline and stored in all nodes. The available storage space on the nodes will determine the size of the tables. From our experimental experience, it should suffice to have a $10 \times 20 \times 20$ table. For instance, this table can have 10 values of $k$ (2 to 11), 20 values of $p$ (0 to 0.95 in increments of 0.05), and 20 values of $\alpha$ (0 to 0.95 in increments of 0.05). These numbers could be stored as integers between 0 and 100. Hence, the two tables would take 8K bytes.

In summary, upon the $m$-th transmission failure, function **Pushback()** (Algorithm 12) is called. In this algorithm, The ML estimates $\hat{x}$ and $\hat{y}$ are calculated in lines 3 and 4. A table lookup is employed to find the value of $\hat{\alpha}$ corresponding to $\hat{x}$, $\hat{y}$ and $k$, and then $\hat{p}$ can be calculated according to Equation (4.5). Finally the pushback period $k$ is estimated using another table lookup with the appropriate values of $\hat{p}$, $\hat{\alpha}$ and $\rho$.

## 4.4.1 Remedial Mechanisms

The pushback algorithm above can work well if the real packet loss pattern is captured well by our channel model introduced earlier and the transition probabilities are accurately measured. However, either of them may deviate from reality, in which

**Algorithm 12**: Pushback()

```
1 if ($failureCount = m$) then
2     $failureCount \leftarrow 0$;
3     $\hat{x} \leftarrow \frac{\text{Number of S}\rightarrow\text{F transitions}}{\text{Total number of stays in S states}}$;
4     $\hat{y} \leftarrow \frac{\text{Number of F}\rightarrow\text{F transitions}}{\text{Total number of stays in F states}}$;
5     $\hat{\alpha} \leftarrow \mathrm{T}_{\hat{\alpha}}(\hat{x}, \hat{y}, k)$;
6     $\hat{p} \leftarrow \hat{x}/(1 - \hat{\alpha})$;
7     $k \leftarrow \mathrm{T}_k(\hat{p}, \hat{\alpha}, \rho)$;
8 end
9 Delay the retransmission for $k$ slots;
```

case the throughput may not be maintained if $k$ is chosen too aggressively. Hence, we introduce two remedial mechanisms to solve such problems.

### Measuring Actual Pushback Amount

In our pushback algorithm, the delay amount, $k$ slots, is calculated according to the state transition probabilities and the throughput constraint. However, after delaying for $k$ slots, nodes may need to delay their retransmissions further due to contention from other senders. This could lead to loss in throughput since the delaying amount is longer than expected by the model. Hence, the running average of the difference between the calculated delay amount and the actual delay amount is maintained, and subtracted from the newly calculated $k$.

### Controlling the Pushback Amount at the Interface Queue

Once our channel model deviates from the actual channel, adjusting $k$ as in Section 4.4.1 may not work well. To cope with such situations, we let the interface queue impose a pushback control policy to speed up packet forwarding once the queue is backlogged. This policy simply commands the pushback algorithm to fall back to CSMA (using $k = 1$) if the queue length is above a certain threshold. In our evaluations, this value is set to half of the queue capacity.

## 4.5 Simulation Evaluation

We conduct simulations in *ns2* [6] to compare the performance of our pushback algorithm with plain CSMA with and without binary exponential backoff in wireless sensor networks. Here the CSMA without exponential backoff (denoted as CSMA) simulates BMAC, the default MAC layer protocol, while CSMA with exponential backoff (denoted as CSMA/EB) represents other general CSMA protocols. We also study the performance improvement when the pushback algorithm cooperates with other congestion control mechanisms such as rate limiting and back pressure. In this section, the radio propagation model used in our simulations is introduced, followed by the simulation results.

### 4.5.1 Radio Model

To make the wireless radio model and MAC layer in *ns2* more real, we make the following two modifications.

**Accumulative Interference and SNR calculation**

The CSMA (MAC and PHY) protocol simulated in *ns2* differs from reality in two ways. First, it fails to consider interference from nodes outside the carrier sensing range. However, the cumulative interference from more than one node sufficiently far away may still affect packet receptions. Second, it does not calculate the packet loss probability according to the Signal-to-Noise Ratio (SNR). In our simulations, we modified the physical layer of *ns2* to combine all sources of noise and interference to calculate the SNR and then use Equation (9) in [78] to calculate the packet success rate.

### Radio Propagation Model

In our simulations, we use a radio propagation model based on the shadowing model implemented in *ns2*. Consequently, the received power level at a receiver is determined by

$$\left[\frac{P_{\text{rec}}(d)}{\overline{P}_{\text{rec}}(d_0)}\right]_{\text{dB}} = -10\beta \log\left(\frac{d}{d_0}\right) + X_{\text{dB}},$$

where $P_{\text{rec}}(d)$ is the received power at this receiver which is a distance $d$ away from the sender, $\beta$ is the path loss exponent, $\overline{P}_{\text{rec}}(d_0)$ is the average received power level at a reference distance $d_0$, and $X_{\text{dB}}$ is a Gaussian random variable with mean 0 and standard deviation $\sigma_{\text{dB}}$ (called shadowing deviation).

In standard *ns2*, $X_{\text{dB}}$ is independent for different packets. However, $X_{\text{dB}}$ usually varies according to some random process [78, 53]. Hence, we use an order 1 autoregressive model (AR(1)) for $X_{\text{dB}}$ as follows.

$$X_{\text{dB}}(t) = \phi X_{\text{dB}}(t-1) + Z(t),$$

where $\phi$ is called the channel coherence coefficient which quantifies the memory in channel variations and $Z(t)$, the error term, is independently and identically distributed with normal distribution $\mathcal{N}(0, \sigma_Z^2)$. To make the variance of $X_{\text{dB}}(t)$ independent of $\phi$, we choose $\sigma_Z = \sigma_{\text{dB}}\sqrt{1-\phi^2}$. In our simulations, the time is discretized such that 1 slot is roughly equal to the average time to transmit a packet, and the value of $X_{\text{dB}}(t)$ is constant within a time slot. Note that in the modified shadowing model, if the autoregression coefficient $\phi = 0$, then the model just falls back to the default shadowing model provided by *ns2*.

### 4.5.2 Simulation Evaluations

We conduct simulation evaluations on our pushback algorithm in data gathering networks. The node located at one corner of the area serves as the sink, while all other nodes generate data periodically to be sent to the sink. We evaluate the performance of all the three protocols (CSMA, CSMA/EB and CSMA/EB with Pushback) under different data rates, channel coherence coefficients $\phi$, shadowing deviations, network sizes, node densities in grid and random topology, packet sizes, and packet transmission rates. The metrics focused on in this study include the following four.

- **Throughput**: number of packets received at the sink in 500 seconds.
- **Packet success rate (PSR)**: average success rate for each transmission attempt in the network.
- **Energy tax**: average number of transmissions needed to deliver one packet to the sink.
- **Normalized delay**: average delay per hop.

Each set of simulations is carried out for 10 times with different random seeds, and the error bar denoting the minimum and maximum values of each simulation set is also plotted. In all simulations, the default parameter values simulating the XSM nodes [21] are summarized in Table 4.5.2.

#### Data Rates

The pushback algorithm takes advantage of the flexibility provided by nodes that afford to delay the retransmissions (e.g., the ones away from the sink) without reducing the throughput. On the other hand, some nodes (e.g., those close to the sink)

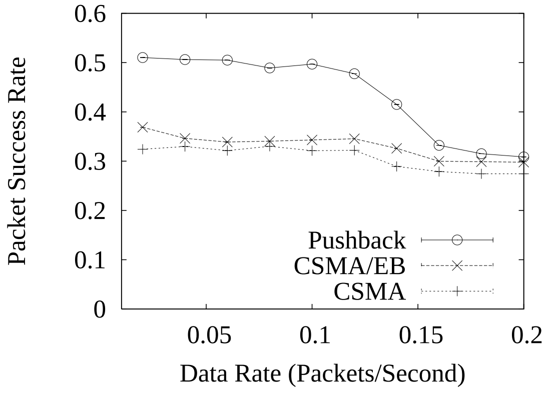
|  |  |  |  |
|---|---|---|---|
| Packet Size | 100 bytes | Ack Size | 5 bytes |
| Bandwidth | 19.2 Kbps | Transmit Power | 0 dBm |
| Backoff Slot | 0.4167 $\mu$s | Pushback Slot | 18.33 ms |
| $\beta$ | 4 [78] | $\sigma_{\mathrm{dB}}$ | 4 |
| $\phi$ | 0.8 | Data Rate | 0.1 packet/sec. |
| Number of nodes | 25 ($5 \times 5$) | Node separation | 45 m |

Table 4.2: Default simulation parameters for the pushback algorithm

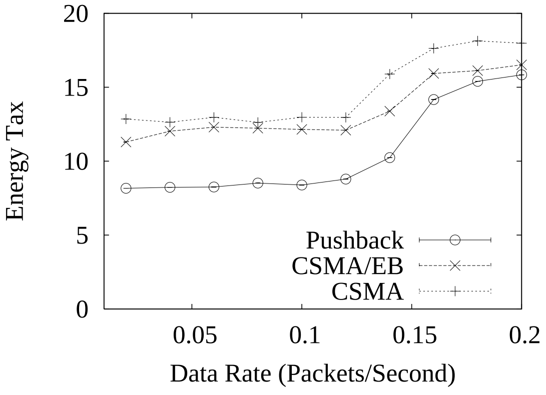
may have very small room for pushback (i.e., $k \approx 1$) since they need to accommodate higher data rates. In this section, we evaluate the pushback algorithm for data generation rate at each non-sink node from 0.01 packets/second (pps) to 0.2 pps. Fig. 4.5 shows the simulation results, from which it can be observed that for low data rate ($<$ 0.15 pps in this case), the pushback algorithm can improve the PSR by 51% and 71% when compared to CSMA/EB and CSMA respectively. For higher data rate, the improvement is less, but higher throughput is achieved at the same time. This implies that with pushbacks the queue drop rates are reduced. Similar improvement in energy tax can also be observed. Note that the higher normalized delay for data rates $\geq$ 0.18 pps is partially due to the queueing delay for delivering more packets.

**Channel Coherence Coefficient**

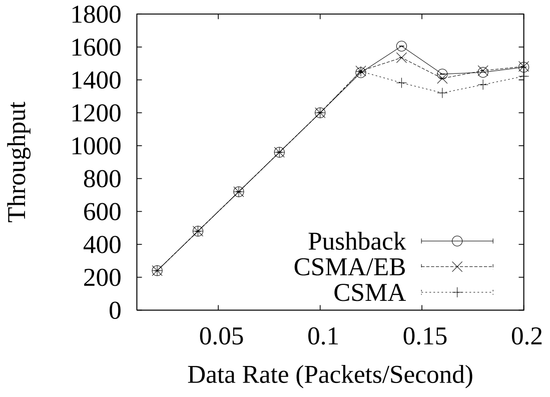
The pushback algorithm is very effective in the presence of temporal correlation in channel losses. Such correlations may be caused by channel coherence or correlated interference. In this section, we evaluate the performance of the pushback algorithm under channel coherence coefficients $\phi$ ranging from 0 to 0.8. Fig. 4.6 shows the simulation results. It can be seen that by utilizing the pushback algorithm, the improvement in PSR over CSMA/EB is between 46% to 63%, and it increases with



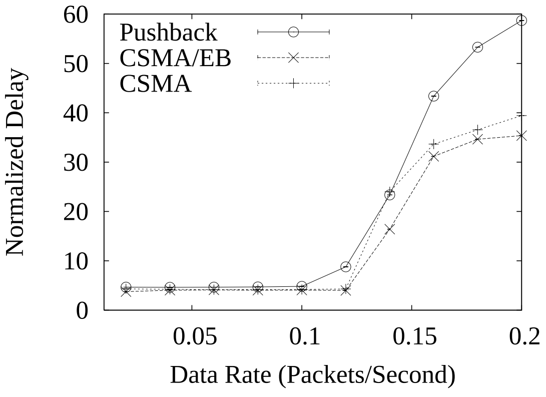
(a) Packet Success Rate



(b) Energy Tax

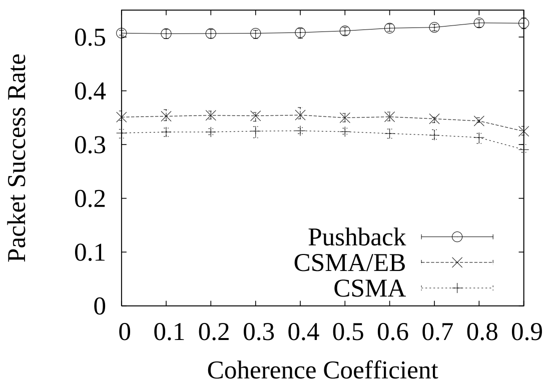


(c) Throughput

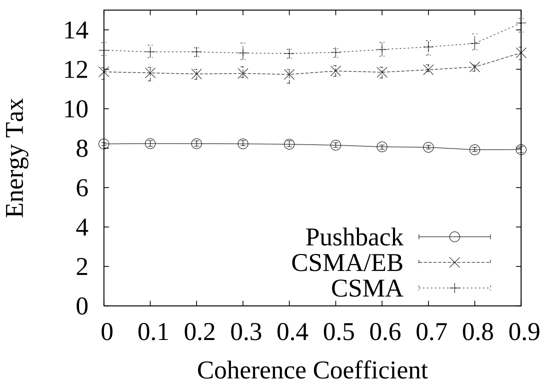


(d) Normalized Delay

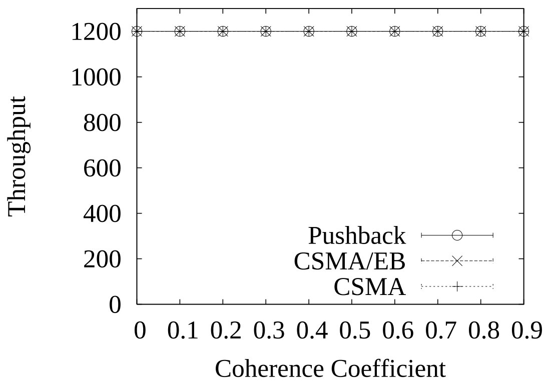
Figure 4.5: Simulation results for various data rates.



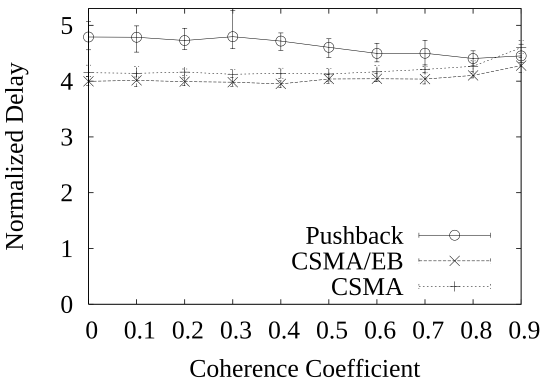
(a) Packet Success Rate



(b) Energy Tax



(c) Throughput



(d) Normalized Delay

Figure 4.6: Simulation results for various channel coherence coefficients $\phi$ in a $5 \times 5$ network.

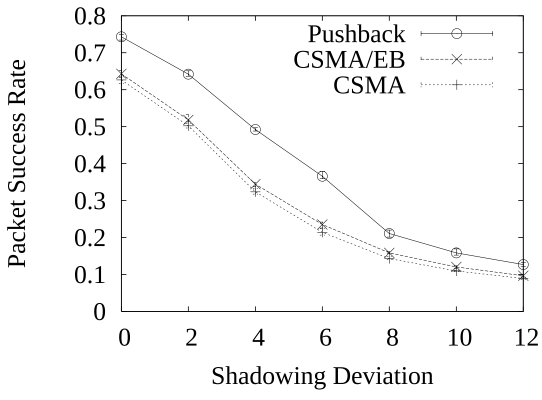
$\phi$. Similarly, there is a reduction in energy tax between 33% to 38%, which increases with $\phi$. The throughput is similar for various values of $\phi$. Although the latency is higher, the difference is small for large $\phi$. As expected, the pushback algorithm is more effective when the channel is more coherent. Moreover, the results also show that even in the case of low channel coherence, the pushback algorithm improves PSR and energy tax because the pushback mechanism also avoids retransmissions when the interference is high.

### Shadowing Deviation

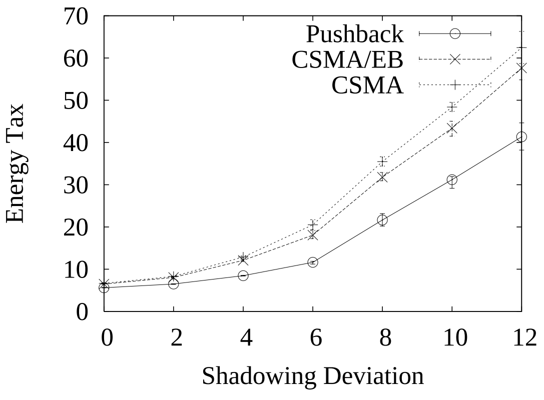
In our simulations, the shadowing deviation $\sigma_{\mathrm{dB}}$ is a major factor besides the interference that affects packet receptions. According to the shadowing radio propagation model, larger shadowing deviation causes more packet losses. In this section, we evaluate the performance of the pushback algorithm for various typical $\sigma_{\mathrm{dB}}$ values [53]. Fig. 4.7 shows the results for the four evaluation metrics. The results show that the PSR for all three protocols decreases with $\sigma_{\mathrm{dB}}$, but the pushback algorithm can provide improvement of up to 91% compared to CSMA/EB. For $\sigma_{\mathrm{dB}} = 0$ in which case the shadowing model does not directly cause packet losses, our pushback algorithm can still improve the PSR from 66% to 76% (or 15% improvement). For energy tax, the percentage reduction increases with $\sigma_{\mathrm{dB}}$ by up to a factor of 8 and becomes steady for higher values of $\sigma_{\mathrm{dB}}$. For throughput, we can observe that the channel capacity is reached for larger $\sigma_{\mathrm{dB}}$ with the pushback algorithm, which results in steady improvement of up to 27% over CSMA and CSMA/EB. It can observed that for large $\sigma_{\mathrm{dB}}$, CSMA/EB has lower throughput than CSMA due to the inefficiency of the exponential backoff mechanism. Fig. 4.7(d) shows the pushback algorithm does not result in deterioration of the delay performance. In fact, the normalized delay can be improved when $\sigma_{\mathrm{dB}}$ is around 8, where the throughput improvement is maximized. Even for $\sigma_{\mathrm{dB}} \geq 10$, the overlapping of the error bars indicates insignificant difference in delay.

### Network Size

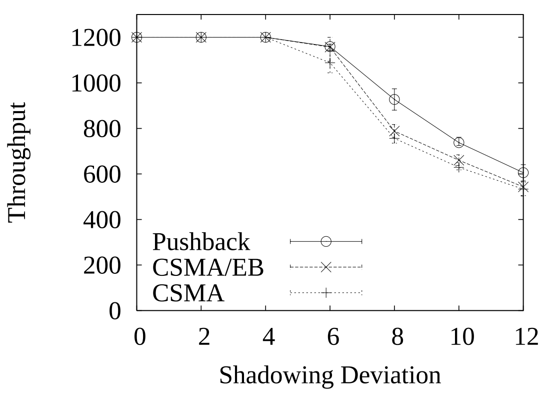
The benefit of the pushback algorithm depends on the amount of delay that nodes can afford before retransmissions as discussed in Section 4.5.2. In a large network,



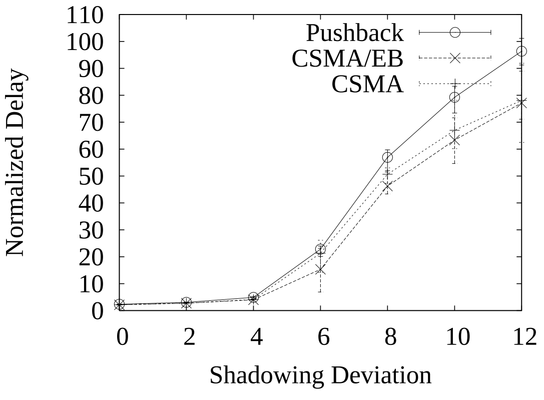
(a) Packet Success Rate



(b) Energy Tax



(c) Throughput



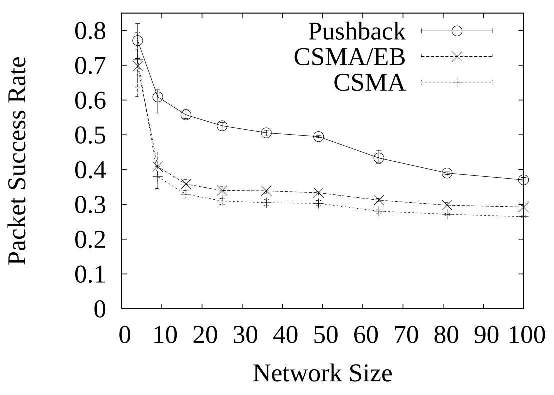
(d) Normalized Delay

Figure 4.7: Simulation results for various shadowing deviations $\sigma_{\mathrm{dB}}$ in a 5×5 network.

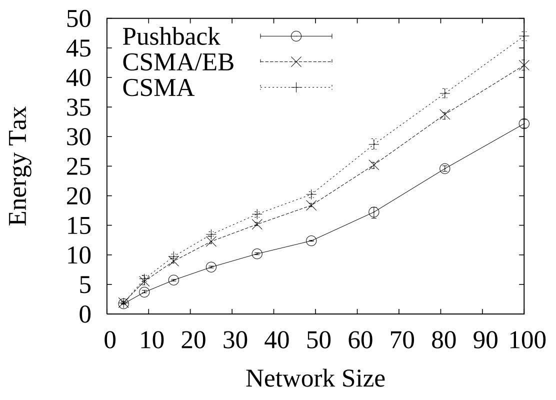
even though the rate of data generated at each node may be low, the cumulative data rates at nodes closer to the sink may still be high. Hence, we evaluate the pushback algorithm for various network sizes in this section. The number of rows and columns in a grid network is varied from 2 to 10 ,i.e., the network size varies from 4 to 100 nodes, while keeping other parameters fixed as in Table 4.5.2. The results are shown Fig. 4.8. Observe that the pushback algorithm results in higher benefit in PSR when the network size is smaller than 64. For larger networks, the pushback algorithm can still provide overall improvement due to the relatively lower data rates at nodes not in the hot-spots. Throughput for large network sizes can also be improved even though the delay is higher due to queueing at nodes for delivering more packets.

### Node Density in Grid Topology

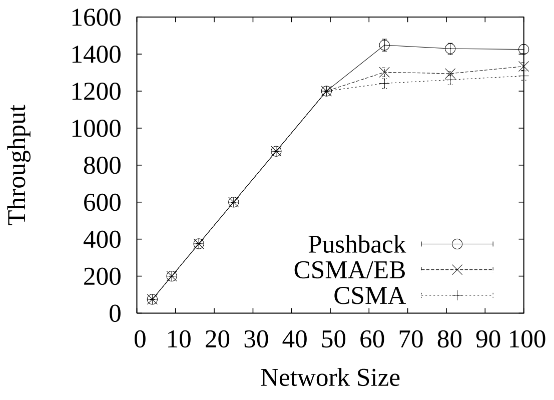
Node density is another potential factor that affects the cumulative data rates at nodes in the hot spots. In this section, we evaluate the performance of the three protocols with different number of nodes placed within a fixed area. With $\sigma_{\mathrm{dB}} = 8$, we vary the number of nodes from 25 to 100 by varying the node separation while keeping other parameters fixed as in Table 4.5.2. Fig. 4.9 shows the results. The lack of smoothness in performance curves is due to the grid placement of nodes. The improvement in PSR and energy tax decrease with the increase of node density while the throughput is still maintained as higher density also implies higher accumulated data rates at nodes in hot spots. Hence, it can be concluded that the pushback algorithm can fulfill the throughput constraint well, and take advantage of opportunities for pushback to save energy.



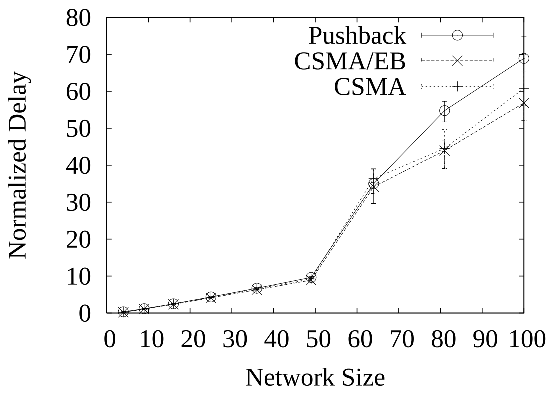
(a) Packet Success Rate



(b) Energy Tax

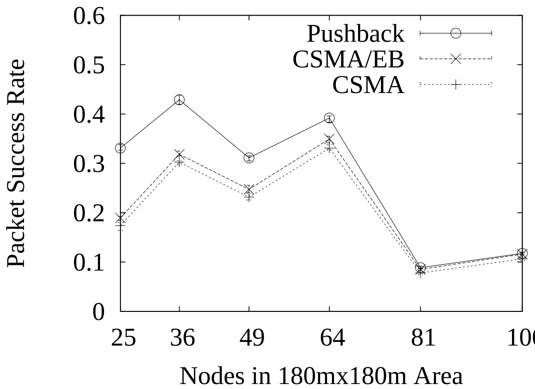


(c) Throughput

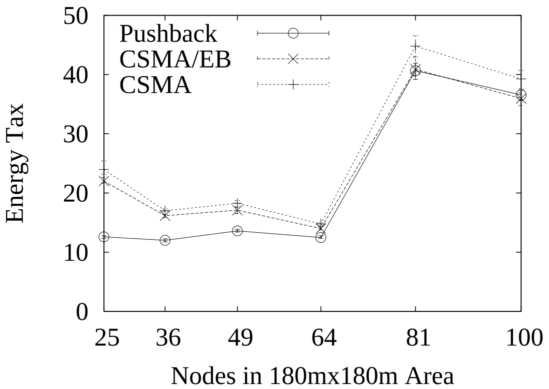


(d) Normalized Delay

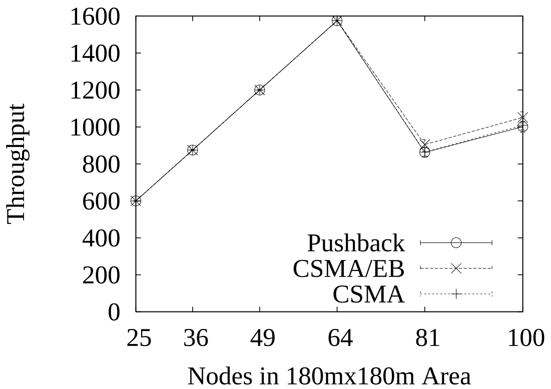
Figure 4.8: Simulation results for various network sizes.



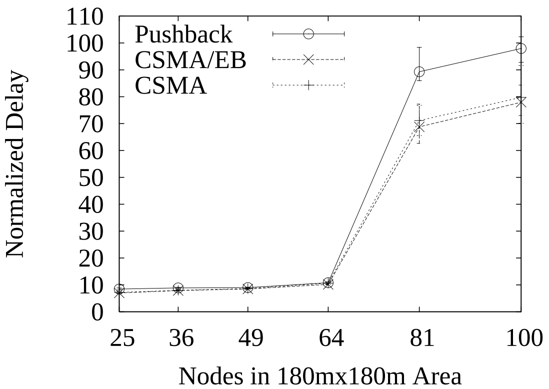
(a) Packet Success Rate



(b) Energy Tax



(c) Throughput



(d) Normalized Delay

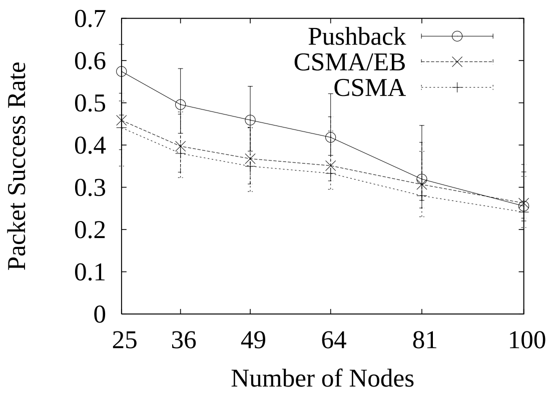
Figure 4.9: Simulation results for various node densities.

### Node Density in Random Topology

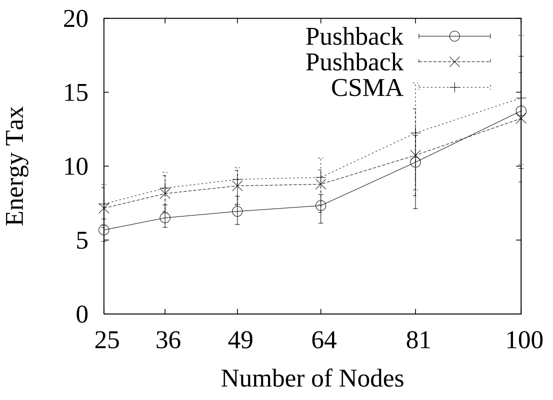
In a network with nodes placed in a grid, many nodes have homogeneous neighbor sets. Hence in this section, we evaluate the performance of the pushback algorithm with nodes randomly placed in a 150 m $\times$ 150 m area. We vary the number of nodes from 25 to 100 with other parameters fixed as in Table 4.5.2. As shown by the results in Fig. 4.10, even though the absolute improvement in PSR decreases with the increase of node density, steady improvement of about 33% is maintained except for the case of 100 nodes which represents quite high node density and thus high data rates at nodes in hot-spots. The absolute reduction in energy tax is stable except for the case of 100 nodes. In addition, the pushback algorithm can lead to higher throughput and similar delay as CSMA.

### Packet Size

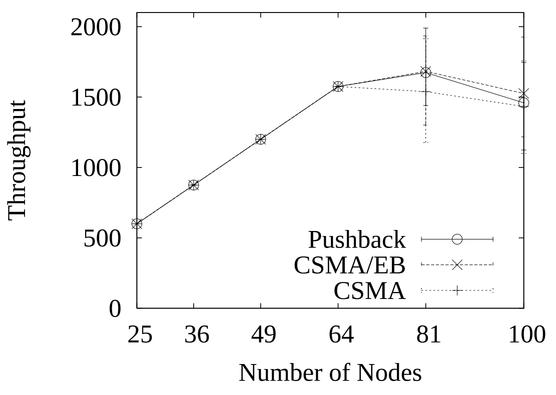
Larger packet size implies smaller control overhead of header size and acknowledgement. But using larger packet size usually also means the transmission is more vulnerable to errors. Hence, we evaluate the performance of the three protocols with various payload sizes in this section and the results are summarized in Fig. 4.11. It can be seen that with the increase in packet size, the PSR decreases and the energy tax increases for CSMA and CSMA/EB. But for the pushback algorithm, the PSR and energy tax are steady, except for packet sizes as large as 250 bytes, and the improvement is more significant for medium packet sizes. For packet size of 250 bytes, the SNR with fixed node separation is not sufficient to sustain a high success rate, which leaves smaller room for improvement.



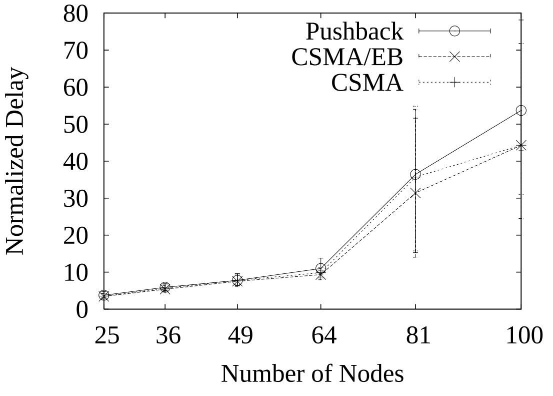
(a) Packet Success Rate



(b) Energy Tax

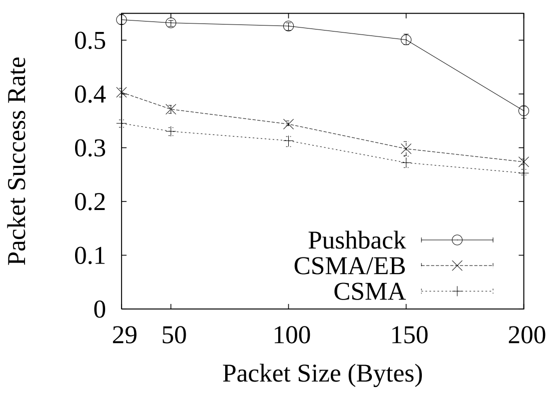


(c) Throughput

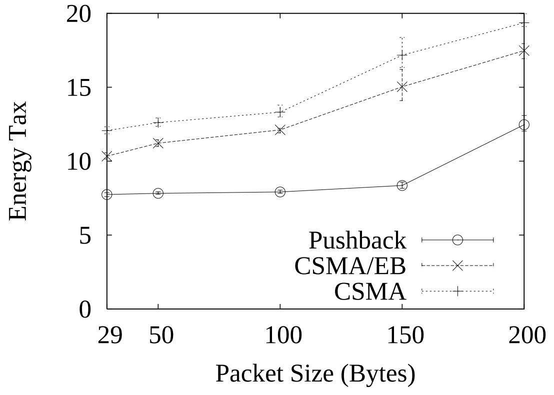


(d) Normalized Delay

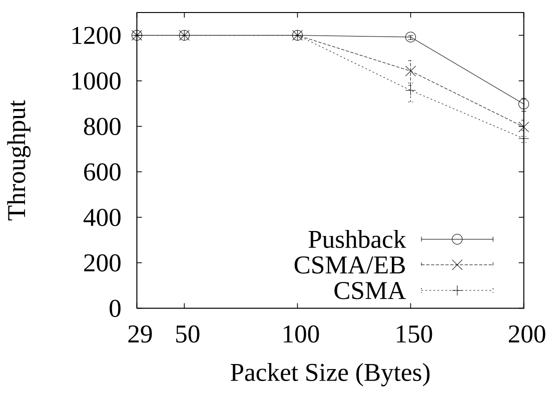
Figure 4.10: Simulation results for various node densities in a random topology.



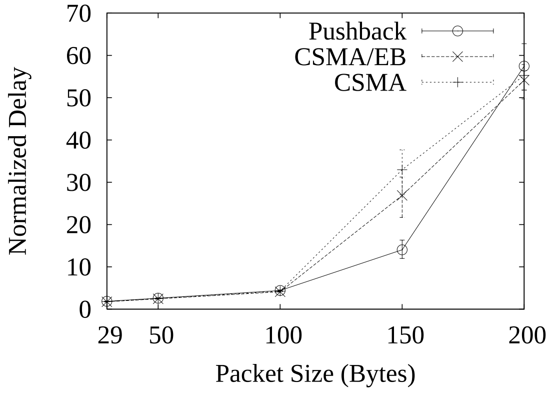
(a) Packet Success Rate



(b) Energy Tax



(c) Throughput



(d) Normalized Delay

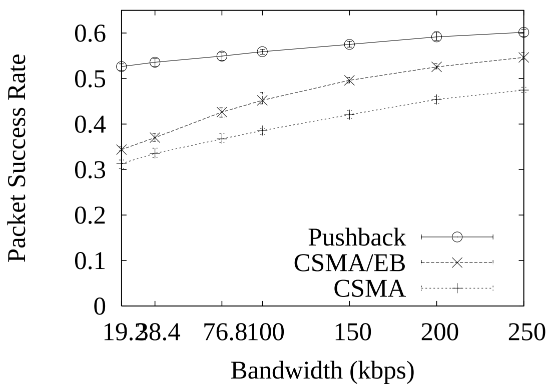
Figure 4.11: Simulation results for various packet sizes.

### Bandwidth

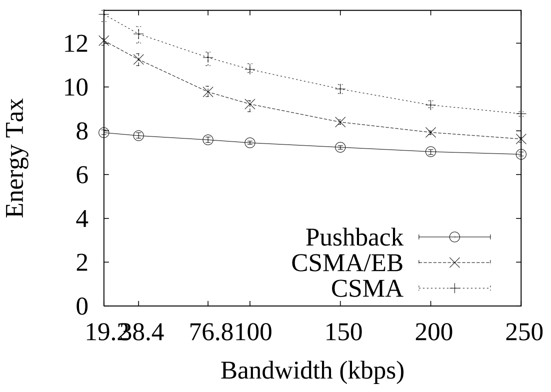
In this section, we evaluate the performance of the pushback algorithm with different bandwidths ranging from 19.2 kbps provided by XSM nodes [21] to 250 kbps supported by IEEE 802.15.4. The results plotted in Fig. 4.12 show that the pushback algorithm can provide improvement in PSR and energy tax, but the improvement is smaller for higher rates. To understand this, observe that even though the bandwidth increases, the size of backoff slots remains similar because it takes similar time to perform clear channel assessment. Therefore, the CSMA backoffs fulfill part of the task of pushbacks. However, for radios with higher rates, if the packet size is proportionally increased, we can still observe similar improvement as small packet sizes in Fig. 4.12.

### Cooperation with Rate Control and Back Pressure

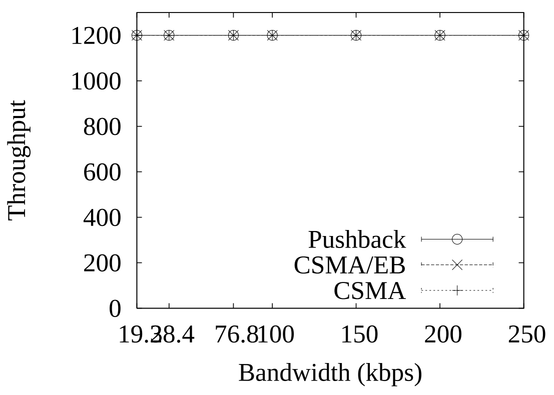
Many link layer protocols [64, 32, 52, 37] use packet rate control and back pressure techniques to mitigate the congestion in the network. These techniques can work in conjunction with and benefit from the pushback algorithm. To show this, we implement and test the rate limiting and back pressure mechanisms (denoted as RC/BP) proposed in [32] along with the pushback algorithm. Fig. 4.13 shows the PSR and energy tax when CSMA/EB and RC/BP are used with and without the pushback algorithm under different data rates. It can be seen that the two metrics have similar variation trends as in Fig. 4.5, which shows that the pushback algorithm can still result in significant benefit when other congestion control mechanisms are used in conjunction. The results for throughput and delay (Fig. 4.13 also shows similar trends as in Fig. 4.5.



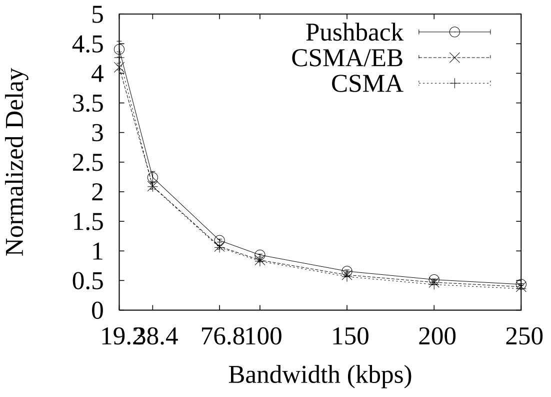
(a) Packet Success Rate



(b) Energy Tax

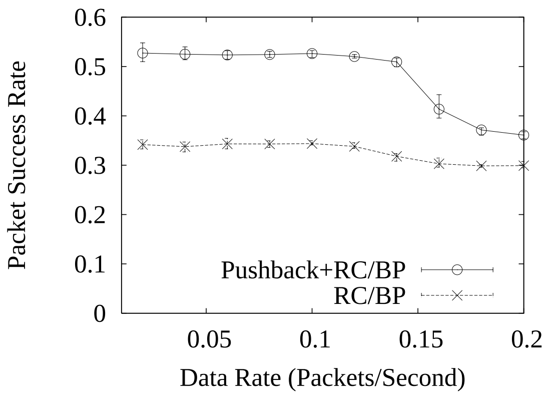


(c) Throughput

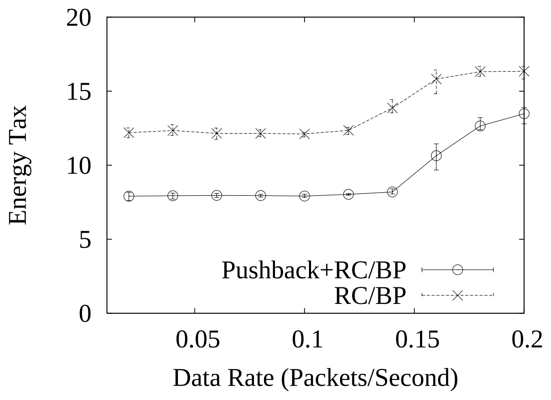


(d) Normalized Delay

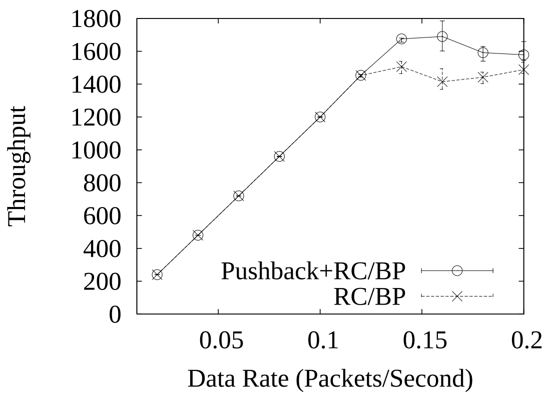
Figure 4.12: Simulation results for various bandwidth.



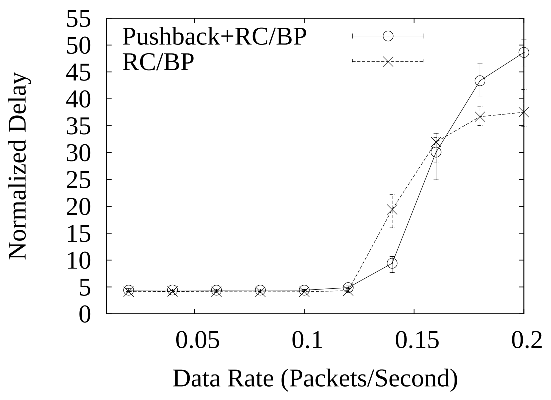
(a) Packet Success Rate



(b) Energy Tax



(c) Throughput



(d) Normalized Delay

Figure 4.13: Simulation results for the cooperation of the pushback algorithm and RC/BP.

## 4.6 Conclusions and Future Research

This chapter introduces a channel aware transmission pushback mechanism to optimize energy efficiency. Using a simple but effective packet loss model, this approach does not incur high computational overhead on the sensor nodes. Using simulations we show that the pushback algorithm can significantly improve the packet success rate and the energy tax without degrading the throughput. In addition, this algorithm is easy to implement over existing MAC and link layer protocols. Hence, we conclude that the pushback algorithm is highly suitable for energy constrained wireless sensor networks.

Future research directions based on the concepts introduced in this chapter are described below.

**Push-Backs in Other Networks:** This chapter has focused on the application of the novel concept of *pushbacks* in the context of sensor networks. However, the concept of *pushbacks* when applied to other networking scenarios such as ad-hoc networks and mesh networks, can be used to optimize other parameters such as the number of transmissions. In these networks, reduced interference due to reduction in number of transmission is expected to result in increased throughput.

**Joint Optimization of Transmission Parameters:** In this work we have used channel quality prediction to appropriately delay transmissions. However, channel quality prediction can be used to adjust other parameters such as physical layer data rate, transmission power, and carrier-sense threshold, some of which are inter-related.

# CHAPTER 5

# Conclusion and Future Work

In this dissertation, we identify the three major energy consumption sources, i.e., idle listening, overhearing, and retransmissions, and propose three approaches to achieve energy efficiency in wireless sensor networks. CMAC (Chapter 2) achieves low duty cycle, low latency and high throughput. The aggressive RTS and anycast in CMAC reduce the idle listening and overhearing, while convergent packet forwarding optimizes route stretch caused by anycast, thus saving the number of transmissions. By taking the reverse channel into consideration, our anycast forwarding set algorithm (Chapter 3) addresses the duplicated forwarding problem of existing anycast algorithms which are reverse link unaware. In Chapter 4, we exploit the temporal correlation in channel condition, and propose to use transmission pushbacks to avoid retransmissions when the channel quality is still poor. Experimental and simulation results have shown that the proposed approaches achieve the design goal and outperform other existing protocols.

Despite the performance improvement brought by the proposed approaches, there are still some open problems to be addressed, which are listed below:

**Optimal receiver contention scheme in anycast:** In Chapter 2, we choose the number of CTS slots and number of mini slots according to empirical measurements. But to make the algorithm more robust for various applications and scenarios, it is worthwhile to investigate the choices of such parameters. In addition, the basic question of how the contention resolution among receivers should be done leads to many other open problems. For example, in some protocols such as ExOR [10, 75], receivers cancel their forwarding tasks by receiving acknowledgements, while in the community of MAC layer anycast [35, 38, 77, 76, 27, 29, 30, 61, 16, 17, 15, 67], the cancellation is performed upon detecting a busy channel. Both mechanisms have their advantages and disadvantages, but rigorous measurements and analysis are needed for better designs.

**Adaptation to network dynamics in anycast forwarding set selection:** The algorithmic framework proposed in Chapter 3 assumes the knowledge of PRR and ARR to each neighbor node, but in a network with these parameters change over time, it is still an open problem to adapt quickly to such dynamics. Network dynamics may be caused by varying traffic patterns, node deployments and failures, and significant changes in link quality. Our current solution (Section 3.4.2) depends on broadcasting advertisements and to construct new forwarding sets in such cases, but the speed of convergence for large scale networks is still a challenging issue to address.

**Joint optimization of transmission parameters:** In Chapter 4, we use channel quality prediction to appropriately delay transmissions. However, channel quality prediction can be used to adjust other parameters such as physical layer data rate, transmission power, and carrier-sense threshold, some of which are inter-related. The joint optimization should consider various factors such as application types and traffic patterns.

Even though various protocols have been proposed to conserve the precious energy resource in sensor nodes in different ways, energy efficient MAC layer design for wireless sensor networks still faces a lot of challenges. Among all challenges, it is worthwhile pointing out that current protocol designs have the problem of integration with other communication components. More specifically, different design goals usually require conflicting assumptions or lead to incompatible features in protocols. For example, protocols for fairness, congestion control [32, 64, 52], data aggregation [25, 24] and our pushback algorithm (Chapter 4) are designed to minimize packet losses, but they also require the sender or receiver to wait a significant amount of time before carrying out further forwarding or transmission to obtain optimal performance. However, idle listening also consumes energy during such waiting periods, while the lack of coordination among the sender and the receiver makes it inefficient to apply duty cycling protocols. Another example is like the DMAC protocol [41] and its variants, which are designed to improve the performance of convergecast [72], but its staggered on wake-up scheduling also incurs high latency for traffic originating from the sink. Hence, protocol designs that optimize one aspect may have suboptimal performance when they are used to build a complete system. In addition, the various traffic patterns introduced by different application types also make the cooperation of protocols inefficient or suboptimal. Therefore, we believe that in the future, the protocol design for wireless sensor networks should take the whole system including all supportive features into consideration. When tradeoffs are necessary, the performance gain and loss from all components under different situations must be analyzed to optimize the entire system.

# BIBLIOGRAPHY

[1] CC1000. http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf.

[2] CC2420. http://focus.ti.com/lit/ds/symlink/cc2420.pdf.

[3] IEEE std. 802.15.4 - 2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs). http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf.

[4] Mica2. http://www.xbow.com/Products/productsdetails.aspx?sid=72.

[5] Stargate. http://platformx.sourceforge.net/home.html.

[6] The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns/.

[7] TinyOS. http://www.tinyos.net.

[8] Tmote sky. http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf.

[9] A. Arora, E. Ertin, R. Ramnath, W. Leal, and M. Nesterenko. Kansei: A High-Fidelity Sensing Testbed. *IEEE Internet Computing*, 10(2):35–47, Mar. 2006.

[10] S. Biswas and R. Morris. ExOR: opportunistic multi-hop routing for wireless networks. In *Proc. SIGCOMM*, pages 133–144, Aug. 2005.

[11] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. In *Proc. SenSys*, pages 307–320, Nov. 2006.

[12] H. Cao, K. W. Parker, and A. Arora. O-MAC: A Receiver Centric Power Management Protocol. In *Proc. ICNP*, pages 311–320, Nov. 2006.

[13] P. Casari, A. Marcucci, M. Nati, C. Petrioli, and M. Zorzi. A Detailed Simulation Study of Geographic Random Forwarding (GeRaF) in Wireless Sensor Networks. In *Proc. MILCOM*, pages 17–20, Oct. 2005.

[14] A. Cerpa, J. L. W. M. Potkonjak, and D. Estrin. Temporal Properties of Low Power Wireless Links: Modeling and Implications on Multi-Hop Routing. In *Proc. MobiHoc*, pages 414–425, May 2005.

[15] D. Chen, G. Cao, and L. Zuo. A Multihop Data Relay Scheme for Wireless Networked Sensors. In *Proc. VTC*, volume 3, pages 1814–1818, Sept. 2005.

[16] D. Chen, J. Deng, and P. K. Varshney. A State-Free Data Delivery Protocol for Multihop Wireless Sensor Networks. In *Proc. WCNC*, volume 3, pages 1818–1823, Mar. 2005.

[17] D. Chen, J. Deng, and P. K. Varshney. On the Forwarding Area of Contention-Based Geographic Forwarding for Ad Hoc and Sensor Networks. In *Proc. SECON*, pages 130–141, Sept. 2005.

[18] J.-G. Choi and S. Bahk. Channel aware MAC scheme based on CSMA/CA. In *Proc. VTC*, pages 1559–1563, May 2004.

[19] R. R. Choudhury and N. H. Vaidya. MAC-Layer Anycasting in Ad Hoc Networks. *SIGCOMM Computer Communication Review*, 34(1):75–80, Jan. 2004.

[20] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proc. MobiCom*, pages 134–146, Sept. 2003.

[21] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *Proc. IPSN*, pages 497–502, Apr. 2005.

[22] A. El-Hoiydi and J.-D. Decotignie. Low Power Downlink MAC Protocols for Infrastructure Wireless Sensor Networks. *Mobile Networks and Applications*, 10(5):675–690, oct 2005.

[23] E. Ertin, A. Arora, R. Ramnath, M. Nesterenko, V. Naik, S. Bapat, V. Kulathumani, M. Sridharan, H. Zhang, and H. Cao. Kansei: A Testbed for Sensing at Scale. In *Proc. IPSN SPOTS)*, pages 399–406, Apr. 2006.

[24] K.-W. Fan, S. Liu, and P. Sinha. Scalable Data Aggregation for Dynamic Events in Sensor Networks. In *Proc. SenSys'06*, pages 181–194, Nov. 2006.

[25] K.-W. Fan, S. Liu, and P. Sinha. Structure-Free Data Aggregation in Sensor Networks. *IEEE Trans. Mobile Computing*, 6(8):929–942, Aug. 2007.

[26] B. D. Fritchman. A Binary Characterization Using Partitioned Markov Chains. *IEEE Trans. Inform. Theory*, 13(2):221–227, Apr. 1967.

[27] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein. Contention-Based Forwarding for Mobile Ad Hoc Networks. *Ad Hoc Networks*, 1(4):351–369, Nov. 2003.

[28] E. N. Gilbert. Capacity of a burst-noise channel. *Bell Syst. Tech. J.*, 39:1253–1266, sep 1960.

[29] T. He, B. M. Blum, Q. Cao, J. A. Stankovic, S. H. Son, and T. F. Abdelzaher. Robust and Timely Communication over Highly Dynamic Sensor Networks. *Real-Time Systems*, 37(3):261–289, 12 2007.

[30] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli. BLR: Beacon-Less Routing Algorithm for Mobile Ad-Hoc Networks. *Computer Communications*, 27(11):1076–1086, July 2004.

[31] G. Holland, N. Vaidya, and P. Bahl. A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks. In *Proc. MobiCom*, pages 236–251, July 2001.

[32] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating Congestion in Wireless Sensor Networks. In *Proc. SenSys*, pages 134–147, Nov. 2004.

[33] S. Jain and S. R. Das. Exploiting Path Diversity in the Link Layer in Wireless Ad Hoc Networks. In *Proc. WoWMoM*, pages 22–30, Jun. 2005.

[34] A. Kamerman and L. Monteban. WaveLAN II: A High-performance Wireless LAN for the Unlicensed Band. In *Bell Labs Technical Journal*, volume 2, pages 118–133, 1997.

[35] J. Kim, X. Lin, N. Shroff, and P. Sinha. On Maximizing the Lifetime of Delay-Sensitive Wireless Sensor Networks with Anycast. In *Proc. INFOCOM*, Apr. 2008.

[36] P. Larsson. Selection Diversity Forwarding in a Multihop Packet Radio Network with Fading Channel and Capture. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):47–54, Oct. 2001.

[37] C. Lim, H. Luo, and C.-H. Choi. RAIN: A Reliable Wireless Network Architecture. In *Proc. ICNP*, pages 228–237, Nov. 2006.

[38] S. Liu, K.-W. Fan, and P. Sinha. CMAC: An Energy Efficient MAC Layer Protocol Using Convergent Packet Forwarding for Wireless Sensor Networks. In *Proc. SECON*, pages 11–20, June 2007.

[39] S. Liu, R. Srivastava, C. E. Koksal, and P. Sinha. Achieving Energy Efficiency with Transmission Pushbacks in Sensor Networks. OSU Technical Report, July 2007.

[40] S. Liu, R. Srivastava, C. E. Koksal, and P. Sinha. Achieving Energy Efficiency with Transmission Pushbacks in Sensor Networks. In *Proc. IWQoS*, June 2008.

[41] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks. In *Proc. IPDPS*, pages 224–231, Apr. 2004.

[42] L. G. Matheson and E. A. Whitehill. Probabilistic Contention Based Forwarding in Multihop Packet Radio Networks. In *Proc. 1994 Tactical Communications Conference*, volume 1, pages 355–364, May 1994.

[43] J. Neter, M. H. Kutner, W. Wasserman, and C. J. Nachtsheim. *"Applied Linear Regression Models"*. McGraw Hill, 4th edition, 2004.

[44] J. Padhye, R. Draves, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *Proc. MobiCom*, pages 114–128, Sept. 2004.

[45] P. Papadimitratos, A. Mishra, and D. Rosenburgh. A Cross-Layer Design Approach to Enhance 802.15.4. In *Proc. MILCOM*, pages 1719–1726, Oct. 2005.

[46] V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S. Iyengar. Random Asynchronous Wakeup Protocol for Sensor Networks. In *Proc. BroadNets*, pages 710–717, Oct. 2004.

[47] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proc. SIGCOMM*, pages 234–244, Aug. 1994.

[48] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. SenSys*, pages 95–107, Nov. 2004.

[49] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A Unifying Link Abstraction for Wireless Sensor Networks. In *Proc. SenSys*, pages 76–89, Nov. 2005.

[50] B. Prabhakar, E. U. Biyikoglu, and A. E. Gamal. Energy-Efficient Transmission over a Wireless Link via Lazy Packet Scheduling. In *Proc. INFOCOM*, pages 386–394, Apr. 2001.

[51] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves. Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks. In *Proc. SenSys*, pages 181–192, Nov. 2003.

[52] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware Fair Rate Control in Wireless Sensor Networks. In *Proc. SIGCOMM*, pages 63–74, Sept. 2006.

[53] T. S. Rappaport. *Wireless Communications: Principles and Practice.* Prentice Hall, second edition, 2001.

[54] I. Rhee, A. Warrier, M. Aia, and J. Min. ZMAC: a Hybrid MAC for Wireless Sensor Networks. In *Proc. SenSys*, pages 90–101, Nov. 2005.

[55] M. Rossi, N. Bui, and M. Zorzi. Cost and Collision Minimizing Forwarding Schemes for Wireless Sensor Networks. In *Proc. INFOCOM*, pages 276–284, Apr. 2007.

[56] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knighlty. Opportunistic Media Access for Multirate Ad Hoc Networks. In *Proc. MobiCom*, pages 24–35, July 2002.

[57] J. A. Sanchez, R. Marin-Perez, and P. M. Ruiz. BOSS: Beacon-less On Demand Strategy for Geographic Routing inWireless Sensor Networks. In *Proc. MASS*, pages 1–10, Oct. 2007.

[58] L. Sang, A. Arora, and H. Zhang. On Exploiting Asymmetric Wireless Links via One-Way Estimation. In *Proc. MobiHoc*, pages 11–21, Sept. 2007.

[59] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari. Energy-Efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks. In *Proc. SenSys*, pages 108–121, Nov. 2004.

[60] M. L. Sichitiu. Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks. In *Proc. INFOCOM*, volume 3, pages 1740–1750, Mar. 2004.

[61] P. Škraba, H. Aghajan, and A. Bahai. Distributed Passive Routing Decisions in Mobile Ad-Hoc Networks. In *Proc. VTC*, volume 4, pages 2814–2818, Sept. 2004.

[62] W. Turin and M. M. Sondhi. Modeling Error Sources in Digital Channels. *IEEE J. Select. Areas Commun.*, 11(3):340–347, Apr. 1993.

[63] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Procotol for Wireless Sensor Networks. In *Proc. SenSys*, pages 171–180, Nov. 2003.

[64] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks. In *Proc. SenSys*, pages 266–279, Oct. 2003.

[65] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: A Wireless Sensor Network Testbed. In *Proc. IPSN SPOTS*, pages 483–488, Apr. 2005.

[66] A. Willig, M. Kubisch, H. Christian, and A. Wolisz. Measurements of a Wireless Link in an Industrial Environment Using an 802.11-Compilant Physical Layer. *IEEE Trans. Ind. Electron.*, 49(6):1265–1282, Dec. 2002.

[67] M. Witt and V. Turau. BGR: Blind Geographic Routing for Sensor Networks. In *Proc. WISES*, pages 51–61, May 2005.

[68] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *Proc. MOBICOM*, pages 146–157, Sept. 2006.

[69] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. SenSys*, pages 14–27, Nov. 2003.

[70] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Trans. Networking*, 12(3):493–506, June 2004.

[71] W. Ye, F. Silva, and J. Heidemann. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling. In *Proc. SenSys*, pages 321–334, Nov. 2006.

[72] H. Zhang, A. Arora, Y. ri Choi, and M. G. Gouda. Reliable Bursty Convergecast in Wireless Sensor Networks. In *Proc. MobiHoc'05*, pages 266–276, May 2005.

[73] H. Zhang, A. Arora, and P. Sinha. Learn on the Fly: Data-Driven Link Estimation and Routing in Sensor Network Backbones. In *Proc. INFOCOM*, pages 1–12, Apr. 2007.

[74] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *Proc. SenSys*, pages 1–13, Oct. 2003.

[75] Z. Zhong and S. Nelakuditi. On the Efficacy of Opportunistic Routing. In *Proc. SECON*, pages 441–450, June 2007.

[76] M. Zorzi and R. R. Rao. Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance. *IEEE Trans. Mobile Comput.*, 2(4):349–365, Oct. 2003.

[77] M. Zorzi and R. R. Rao. Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance. *IEEE Trans. Mobile Comput.*, 2(4):337–348, Oct. 2003.

[78] M. Zuniga and B. Krishnamachari. Analyzing the Transitional Region in Low Power Wireless Links. In *Proc. SECON*, pages 517–526, Oct. 2004.