Title
Efficient Radio Communication for Energy Constrained Sensor Networks

Author
Liu, Tao

Publication Date
2013

Peer reviewed|Thesis/dissertation

University of California
Merced

Efficient Radio Communication for Energy Constrained Sensor Networks

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering and Computer Science

by

Tao Liu

2013

The dissertation of Tao Liu is approved.

John Heidemann

David Noelle

Stefano Carpin

Alberto E. Cerpa, Committee Chair

University of California, Merced

2013

Dedicated to my parents, Shiyi Liu and Yanping Zhang, and my wife, Li Zhang.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank my advisor Alberto Cerpa for his support and guidance throughout my graduate study. I am also thankful for the support of my dissertation committee, Stefno Carpin, John Heidemann and David Noelle whose advice and suggestions were very much appreciated.

A big thanks also go to my wife, Li Zhang, for making this work possible and my time at UC Merced all the more enjoyable. I would also like to express my gratitude to my parents, Shiyi Liu and Yanping Zhang, for their loving support. Finally, I am thankful for the delightful company of my colleagues, Lun Jiang, Ankur Kamthe, Varick Erickson, Stefan Achleitner, Benjamin Balaguer, Gorkem Erinc, Derek Burch, Chao Qin, Weiran Wang and Jimei Yang, and friends at UC Merced.

Vita

1978	Born, Wuhan, China.
2000	Bachelor of Engineering, Wuhan University, China.
2006	Master of Science, London South Bank University, UK.
2007–2013	Research Assistant, Electrical Engineering and Computer Science, University of California–Merced.

Publications

Tao Liu and Alberto E. Cerpa. TALENT: Temporal Adaptive Link Estimator with No Training. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (SenSys 2012)*, Toronto, Canada, November 2012.

Tao Liu and Alberto E. Cerpa. Foresee (4C): Wireless link prediction using link features. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN 2011)*, Chicago, IL, April 2011.

Lun Jiang, Jyh-How Huang, Ankur Kamthe, **Tao Liu**, Ian Freeman, John Ledbetter, Shivakant Mishra, Richard Han, Alberto E. Cerpa. SenSearch: GPS

and Witness Assisted Tracking for Delay Tolerant Sensor Networks. In *Proceedings of the 8th International Conference on Ad Hoc Networks and Wireless (ADHOCNOW 2009)*, Murcia, Spain, September 2009.

Tao Liu, Ankur Kamthe, Lun Jiang, Alberto E. Cerpa. Performance Evaluation of Link Quality Estimation Metrics for Static Multihop Wireless Sensor Networks. In *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2009)*, Rome, Italy, June 2009.

Weishuai Cheng, Jin Chen, **Tao Liu**. Method for Computing Correlation Coefficient between Structural Risks in Water Transfer Project. Engineering Journal of Wuhan University (Engineering Edition), Vol.37 No.1, 2004.

Qiang Liu, Jin Chen, Wei Huang, **Tao Liu**. Studies on Water Resources Bearing Capacity, China Water Resources, Vol. 10, 2003.

Abstract of the Dissertation

Efficient Radio Communication for Energy Constrained Sensor Networks

by

Tao Liu

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Merced, 2013

Professor Alberto E. Cerpa, Chair

Radio communication is an integral part of wireless sensor networks. This dissertation focuses on improving the energy consumption of radio communication in sensor networks by proposing novel approaches in two key aspects of low-power wireless communication, namely, wireless link quality estimation and low duty-cycle data forwarding protocol. I first motivate the research with a comparative study of the routing performance and energy consumption with respect to existing link quality estimation protocols. Then, I propose 4C, a data-driven approach to build link quality prediction models based on empirical data collected from the deployment site in order to address the problems of link quality estimation in low-power sensor networks. Furthermore, I improve this novel data-driven approach to predict short temporal link quality variations without prior collected training data by employing online learning techniques. Analytical and empirical evaluations show that the proposed link quality prediction approach can significantly reduce the cost of radio transmissions by utilizing long links with variable quality. Moreover, I proposed SAF, an energy-efficient data forwarding protocol that can effectively utilize the short term link quality prediction models in duty-cycled

networks. With the help of link quality prediction models, SAF not only minimizes the energy consumption spent on idle nodes, but also leverage the spatial diversity of wireless links via opportunistic routing. The dissertation concludes with a discussion of the potential improvements of the proposed approaches in low-power sensor networks as well as in other wireless networks.

CHAPTER 1

Introduction

In the recent years, wireless sensor networks (WSNs) have gained increasing attention from both the research community and actual users. Typically, WSNs consist of multiple sensor nodes deployed in the vicinity of the interested phenomenon in order to monitor the relevant physical or environmental conditions. A sensor node in WSNs is generally equipped with a low-power radio transceiver, small micro-controller, a power source (usually batteries) as well as various sensing components such that the node can be deployed at diverse locations to monitor the interested phenomenon. Depending on the sensing component, sensor nodes are capable of sensing many types of information from surrounding environments. As the sensor nodes are typically battery powered and communicate wirelessly, there is no requirement for any additional infrastructure to power or connect these nodes once they are deployed. This unobtrusive nature of the nodes enables autonomous sensing in conjunction with other nodes and facilitates large scale deployment of many sensor nodes in remote, mobile, or dangerous locations that is inconvenient for traditional, hardwired sensing devices. Moreover, the low power radio enables the nodes communicate with each other and transmit the acquired data cooperatively through ad-hoc, self-configured wireless networks to one or more gateway nodes or basestations [PSM04, XRC04, WLJ06]. With multiple sensor nodes deployed in the in the vicinity of the interested phenomenon, WSNs can monitor various physical parameters from multiple locations, sensing

and detecting relevant events with great detail across large areas. Thus, the power of WSNs arises not from the capabilities of each node but of the network comprising of hundreds of nodes as a whole. As such, WSNs are seen in many various applications, such as wildlife behavior monitoring in the natural habitat [MCP02, LSZ04], environmental parameters monitoring from hazardous locations [DS05, WLJ06], event detection in industrial applications [CCD11, ZSC11], unobtrusive human behavior sensing [ESK11, NDL12] and object detection, classification and tracking in outdoor [KLK07] and indoor [KJD09] environments.

The deployment requirements from various applications, however, impose unique challenges to the sensor nodes in terms of size, cost and capacity. In order to reduce deployment efforts, the size of sensor nodes are generally small, ranging from that of a shoe box [Sta05] to the size of a coin [WIS09]. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. Therefore, a common topic in WSNs is the design of efficient systems that balance the sensing performance, communication costs and system life time with limited energy, computation and communication capacities.

1.1 Problem Statement

Radio communication is an important component of any wireless distributed system, and it is critical for WSNs in particular. As sensor nodes are generally battery-powered devices, a fundamental concern in sensor network design is the energy consumption. The radio transceiver is often one of the most power hungry components in sensor nodes [PK00, Mot], and therefore minimizing the

energy spent on radio communications is one of the main goals of the communication protocols for WSNs. In order to reach this goal, many routing protocols [WTC03, GFJ09, DYM] designed for low-power sensor networks establish their network topology based on quality of the links in the data forwarding paths. By estimating the quality of the wireless links in the network, the routing protocol can select the data forwarding path from the source node to destination node with the best quality such that the radio transmission cost is minimized. In this regards, efficient and accurate link quality estimation is vital for the underlying routing protocols to calculate the quality of potential data forwarding paths and is directly connected to the energy consumption of radio communication.

However, the wireless link quality in WSNs is notoriously dynamic and unpredictable. As pointed out by a vast array of research [ZG03, ZHK04, LMH03, ABB04, CWP05, CWK, SL06, SKH06, SKA, SDT10], the propagation of the low-power radio signals is affected by many factors, resulting in wireless link quality fluctuation over time [CWP05, SDT10] and space [ZG03, ZHK04, CWK], and connectivity is often asymmetric [ZHK04, CWK]. Most of the prior research focuses on the empirical characterization of low-power links through real-world measurements in the link level and present radically different (sometimes contradicting) results, which raise the need for a comparative study that deeply analyzes the performance impact in terms of routing when using different link quality estimation techniques.

Moreover, recent research [ALB09, SDT10] indicates that the commonly used link estimators such as ETX [DAB03] or 4Bit [FGJ07] tend to capture the long term link quality instead of short term quality variations. As a result, routing protocols based on these link estimators often favor the stable high quality links and tend to ignore the links that have intermediate quality in long term, but

show continuous high quality in short periods. Prior work [CWP05] has shown that intermediate links usually cover longer distances than high quality links, and routing protocols could take advantage of the high quality periods of these links and use them when forwarding a packet. This strategy can reduce the number of hops in the path, and ultimately reduce the number of transmissions for delivering a data packet. Further, current link estimators often assume that the current link quality remains the same as the last estimation, but this assumption of stable link quality is often invalid due to the frequent variations of wireless links. Thus, accurate quality estimation of intermediate links remains a difficult task due the convergence time of ETX and the dynamic nature of wireless channels. A major goal of the proposed research is to design prediction based link quality estimation schemes that can accurately *predict* the wireless link quality under varying network conditions in order to improve the routing efficiency by using long, intermediate quality links.

1.2 Dissertation Goals

This work advocates a data-driven approach to the challenging problem of link quality estimation for wireless links with frequent quality variations. The basic idea of this data-driven approach is to utilizes machine learning methods to build models that predict the link quality with information from historical packet reception. With the prediction models, we propose model based link quality estimators that can accurately predict *when* the link quality is high with few inputs from past packet reception. Furthermore, we extend the modeling approach by incorporating online learning techniques to eliminate the need of off-line model training. Through extensive analytical and empirical evaluation, we argue that this data-driven approach can significantly improve routing efficiency and reduce

communication costs without incur much overhead. Overall, the goals of this work can be summarized as the following.

- To design a data-driven link quality prediction scheme suitable for both sparse and burst traffic patterns.
- To develop prediction models with online learning algorithms that can adapt to varying network conditions without off-line training.
- Validate the proposed approaches through comprehensive analysis and evaluation.

1.2.1 Summary of Work

This section summarize the four parts that contribute to the thesis. First, we highlight the challenges of link quality estimation by presenting a comprehensive study of the routing performance of common communication protocols with respect to existing link quality estimation techniques. Then, we propose 4C, a data-driven, model based hybrid link quality predictor for improving routing efficiency. Third, we present TALENT, which further extends 4C with online learning techniques to predict short temporal link quality variations for burst traffic patterns. Finally, we proposed SAF, a cross layer data forwarding protocol that addresses the energy consumption problems in duty-cycled sensor networks, and discusses the integration of TALENT in a duty-cycled scenario.

Specifically, Chapter 2 first introduces the related literature regarding radio communication in low-power sensor networks and discusses the state-of-art link estimation techniques in detail. Chapter 3 then motivate the research through a comprehensive study on the routing performance of existing link estimators in order to highlight the challenges in link estimation and data transmission in

WSNs.

Chapter 4 proposes 4C, a novel link estimator that applies link quality *prediction* along with link estimation. The approach is data-driven and consists of three steps: data collection, offline modeling and online prediction. The author shows the usefulness of link quality prediction based on different machine learning methods, such as, naive Bayes classifier, logistic regression and artificial neural networks. The prediction models take a combination of link layer and the physical layer information as input, and output the reception probability of the next packet. Analytical and empirical results show that logistic regression works well among the three models with the additional advantage of having the small computational cost.

Based on the modeling guidelines established in 4C, Chapter 5 presents TALENT, a self-learning, plug-and-play estimator to predict the quality of a wireless link in the near future using a combination of packet and physical level quality indicators. One of the main advantages of TALENT is the use of online learning techniques that are able to adapt to the wireless dynamics without the need for data collection and model re-training. When using TALENT together with CTP, experimental results show that on many different environments TALENT increases the delivery efficiency significantly in comparison to state-of-the-art link quality estimators.

In Chapter 6, we then address the problems in duty-cycled networks, i.e., additional energy consumption introduced by asynchronous wakeup schedule. The proposed Synchronized Anypath Forwarding (SAF) is a cross-layer data forwarding scheme that applies opportunistic routing techniques and trades off latency for a significant increase in data transmission efficiency with low duty cycle. SAF synchronizes the sender and receiver nodes locally and therefore eliminates the

need of idle sending/listening in typical duty cycle techniques such as Low Power Listening. Furthermore, it takes the advantage of the synchronous wake-up schedule to send the packets to multiple nodes, such that the recipient node(s) with the largest routing gain can forward the packet. This chapter also discussed the integration of SAF with link quality predictor such as 4C and TALENT. Finally Chapter 7 concludes the dissertation with possible improvements.

1.3 Contribution

The contribution of this thesis is three fold.

First, we are the first to propose the data-driven, model based link quality prediction approach (4C). With extensive evaluation, we show that these models, with the appropriate set of parameters, can be implemented in resource constrained nodes with very limited computation capabilities and small overhead. Also, we designed a receiver-initiated online prediction module that informs the routing protocol about the short temporal high quality links, enabling the routing protocol to select temporary, low-cost routes in addition to the stable routes. Empirical evaluation proves the effectiveness of the proposed approach in terms of transmission cost reduction.

Second, we show that by using online learning techniques, the TALENT prediction model can adapt to a wide range of network dynamics without prior training data and with fast convergence time. To our knowledge, this is the first attempt to introduce online learning techniques to adapt network link estimation parameters under environmental and network dynamics. Our implementation of TALENT in TinyOS integrates with LPL [PHC04], a low-power listening protocol for efficient communication and actual energy savings when using duty-cycled

radios. Furthermore, we applied the prediction approach of TALENT to empirical packet traces from 802.11 networks and confirmed that TALENT outperforms ETX based link estimators significantly even in 802.11 networks with much higher data rate. These results suggest the potential application of TALENT in much wider range of wireless networks.

Finally, we design a novel opportunistic anycast routing protocol, SAF, which actively regulates the traffic and takes the advantage of the predicable and bursty traffic pattern to forward data in an energy efficient manner. SAF maintains a sender-driven adaptive wakeup schedule, which allows synchronization between senders and multiple forwarding nodes. SAF is a practical realization of duty-cycled routing protocol that can utilize the full potential of the prediction based link estimators such as 4C and TALENT.

CHAPTER 2

Background and Related Work

Data collection and event detection is the basis of many sensor networks applications and deployments [PSM04, XRC04, WLJ06], therefore the most common network structure of WSNs is a multihop tree topology: nodes in the network connect to the root node (basestation) through one or more hops, forming a tree-like structure for data aggregation in the basestation. Routing protocols establish the routing tree based on the quality of the wireless links between nodes in the network such that the transmission cost of sending a packet from sensor nodes to the root is minimal. In this regard, accurate link quality estimation is vital to achieve optimal routing topology and greatly affects routing performance. However, due to the dynamic nature of wireless channels, accurate link quality estimation remains a challenging task. The following sections summarize the state-of-art in terms of link quality estimation metrics and techniques, as well as the common radio communication protocols in duty-cycled networks.

2.1 MAC Protocols in Wireless Sensor Networks

Media Access Control (MAC) protocols are low level channel access control mechanisms that control the network nodes to communicate within a shared medium, e.g. radio channel. As maximizing the network lifetime is a common objective of sensor networks, MAC protocols often incorporate duty-cycling, i.e., turn on the

radio only when there are wireless activities. In sensor networks, low-power radio communication in the MAC layer generally falls within two categories: asynchronous and synchronous. Synchronous low-power networks communicate on shared local or network-wide schedules, but many of these protocols often lean toward a hybrid approach. As indicated by Rhee *et. al* [RWA08], TDMA protocols suffer from many deficiencies when applied to ad-hoc wireless networks. Network scalability, mobility, efficiency, broadcasting capabilities, and robustness often severely degrade in performance due to the rigidity of pure TDMA protocols. Therefore, many protocols [DL03, YHE02, ED04, YSH06] adopt a hybrid approach to avoid the deficiencies of TDMA by combining TDMA with certain types of asynchronous support. In this work, we focus on asynchronous driven low-power protocols as asynchronous networks are easy to deploy, reliable, widely tested, and are very common in current WSN deployments.

2.1.1 Asynchronous MAC Protocols

Asynchronous MAC protocols can be further categorized as sender-initiated and receiver-initiated. In sender-initiated protocols, the sender nodes actively probe the radio channel such that the receiver nodes can detect the potential traffic, whereas in receiver-initiated protocols, the receiver nodes poll the channel and the sender nodes respond with data packets in local buffer. Low Power Listening [Dav07] (LPL) specified in TinyOS [LMP04] is an example of sender-initiated protocols and is implemented in multiple MAC protocols [PHC04,ED04,BYA06]. For example, X-MAC [BYA06] proposed by Buettner *et al.* is a commonly used low-power asynchronous MAC for energy constrained sensor networks. In X-MAC, nodes periodically wake up the radio to detect radio activity with clear channel assessment (CCA) and turns off the radio if there is no activity detected.

Upon detecting activity in the wireless channel, X-MAC keeps the radio on to receive the potential packet. Once the packet is received, the MAC protocol signals the receive event to upper layers, and puts the radio back to sleep after a short wait period. On the other hand, the nodes with packets to send need to continuously occupy the wireless channel, so that the intended recipient nodes can detect the incoming packets. BoX-MAC [ML08] is an improved version of X-MAC and is the default MAC protocol of CC2420 radio in TinyOS. As oppose to these sender-initiated MAC protocols, SCP-MAC [YSH06], RI-MAC [SGJ08] and A-MAC [DDC10] represent receiver-initiated MAC protocols which initiate packet transmissions from the receiver side. The main advantages of the receiver-initiated approach is that it can avoid the long preambles of receiver-initiated approach and can achieve ultra low duty cycles, but it also comes with the addition overhead of time synchronization and scheduling, and less efficient channel probing mechanism (probe/ack frame exchange) compared to simple channel polling used by sender-initiated approach.

As energy conservation is the main focus of the low-power sensor network protocols, cross layer adaptive data forwarding protocols with low duty cycle also received much attention. Y. Gu and T. He [GH07] proposed DSF which optimizes on the expected data delivery ratio, expected communication delay, and expected energy consumption to achieve the desired energy efficiency. Koala [MLT08] propose an ultra low duty cycle, reliable data retrieval system that maintains no persistent routing state on the nodes. In addition, adaptive MAC protocols such as PW-MAC [TSG11] and WiseMAC [ED04] propose predictive methods as well as MAC level synchronization to improve energy usage, throughput and reliability. SAF proposed in Chapter 6 also falls into this category, however it departs from previous work due to its unique design aspect: SAF regulates the traffic such that synchronous data forwarding and opportunistic routing can be

used together to minimize the energy consumption.

2.2 Routing Protocols in Sensor Networks

In this section, we review the prior research addressing the issues of routing in WSNs. We focus primarily on protocols that are implemented and tested in real-world environments.

Due to the ad-hoc nature of sensor networks, it make sense to inherit the existing routing protocols for ad-hoc networks. For example, TYMO [DYM] and NST-AODV [GSA06] both originate from the ideas behind DYMO and AODV, which are protocols tailored to mobile ad-hoc networks. However, there are basic problems that arise with these protocols in WSNs. First, the hop count metric used in these protocols does not provide good performance since it treats all hops as equal. Second, routes are based on the end-to-end principle, meaning that they are costly both to establish and to maintain in a lossy environment. And finally, the protocols do not exploit the fact that most traffic is destined to one node (i.e., the sink), which is a common practice in WSNs.

For many sensor networks applications and deployments [PSM04, XRC04, WLJ06], the basic network structure is a multihop tree topology: nodes in the network connect to the root node(s) through one or more hops, forming a tree-like structure. Routing protocols establish the routing tree based on the quality of the wireless links between the sender nodes and the forwarding nodes such that the path cost of sending a packet to the root is minimal.

Collection Tree Protocol (CTP) [GFJ09] is the default tree-based multihop collection protocol in TinyOS 2. It is is developed based on the experiences on other collection protocols such as MultihopLQI [Mul] and MintRoute [WTC03].

In CTP, every node maintains an ETX value for it self and uses a link estimator to estimate the ETX of nearby neighbors. The ETX value is used as the routing gradient: the root nodes have the lowest possible ETX, and nodes away from the root nodes will have additive ETX values of all the hops between the node and the nearest root node.

CTP consists of three main components: a link estimator that is responsible for estimating the quality of the links to single-hop neighbors; a routing engine that is in charge of choosing the next hop (parent) based on the link estimation as well as processing network-level information such as congestion detection; a forwarding engine that maintains a queue of data packets to send. The link estimator is loosely coupled with the routing engine and forwarding engine, therefore can be implemented with different link estimation techniques or even different metrics. Currently the default link estimator of CTP is 4Bit [FGJ07], but other link estimators are also available [GBP10, BKY10, BZV10].

2.2.1 Opportunistic Routing in Duty-Cycled Networks

ExOR [BM05] is the first to propose the concept of opportunistic routing (OR) in wireless networks. By taking advantage of the broadcast nature of the wireless medium, ExOR leverages the spatial diversity of the network to improve the throughput. The potential of OR has been confirmed by both theoretical analysis [ZLY07] and experimental evaluations [CJK07, LM07].

In the WSN field, numerous studies [SIV09, DGV11, PTD11] address the theoretical aspect of OR. The main focus of these studies are in terms of energy, reliability and delay due to the unique challenge of the energy constrained WSNs. Their models and simulations show that OR can be used to reduce energy consumption as well as improve routing efficiency. Landsiedel [LGD12] *et al.* incor-

porated the principals of opportunistic routing in duty cycled sensor networks to reduce the radio power consumption and the latency at the same time. Their results strongly motivated our research.

2.3 Link Quality Estimation in Sensor Networks

As discussed in the previous section, effective link quality measurement is a fundamental building block for reliable communication in wireless network, and numerous link estimation metrics and techniques have been proposed in this area. In this section we summarize the prior literature related to the topic of wireless link quality estimation in WSNs and highlight the main challenges of link estimation in the low-power, resource-constrained sensor networks.

2.3.1 PRR based Link Estimation Metrics

Due to the highly dynamic nature of wireless channel, traditional metrics used in wired networks, such as hop count, round trip time and latency generally fail to provide a highly reliable path estimation in WSNs. De Couto *et al.* [DAB03] proposed ETX, a widely used wireless link quality metric that uses the packet loss ratio to estimate the expected transmission cost over a wireless link. Their study show that this Packet Reception Rate (PRR) based metric can achieve better routing performance than the shortest hop count. Further comparisons by Draves *et al.* [DPZ04] conclude that in static wireless networks, ETX performs better than three other commonly used traditional metrics, namely, minimum hop-count, per-hop Round Trip Time and per-hop Packet Pair Delay.

Woo *et al.* [WTC03] outlined an effective design for multihop routing and confirmed that the PRR based metrics such as ETX are more suitable in energy-

sensitive routing scenarios. They also showed that window mean estimator with exponentially weighted moving average (WMEWMA) is superior to other well established estimation techniques such as moving average. Based on their design, Fonseca *et al.* [FGJ07] proposed the Four-Bit link estimator (4Bit) to provide well-defined interfaces that combine information from the physical, data-link and network layers using four bits. The seamless integration of the four bits of information makes 4Bit agile and lightweight on the link quality monitoring: an *ack* bit from the link layer indicates whether an acknowledgment is received for a sent packet, a *pin* bit and a *compare* bit from the network layer interact with the underlying routing protocols to keep important neighbor nodes monitored, and a *white* bit from the physical layer denotes the high quality wireless channel by checking the rate of the decoding error in the received packets. Although the white bit uses physical layer parameter, 4Bit only consider it as a quick indication of the overall wireless channel quality. In essence, 4Bit inherits the WMEWMA design proposed by Woo *et al.* and uses ETX as its link quality metric.

The stability of EWMA filters is determined by its weight, which is fixed in traditional EWMA filters. Kim *et al.* argued that it is possible to create an adaptive filter by combining EWMA filters with different weights. They proposed Flip-flop [KN01], a composition of an agile EWMA filter and a stable one that can be agile when possible but stable when necessary. Renner *et al.* proposed the Holistic Packet Statistics (HoPS) [REW11], which incorporates four quality metrics, namely, Short-term, Long-term, Absolute Deviation, and Trend estimation together to provide a holistic assessment of the link and its dynamic behavior. However, an intrinsic problem of these ETX based metrics is that the ETX value require several packet receptions to calculate, which limits the agility of the link estimators.

In addition to ETX, A. Cerpa *et al.* [CWP05] proposed Requested Number of Packets (RNP) in their study of temporal properties of low power wireless links. They discovered that among links with similar delivery rates, a link with discrete losses can deliver more data packets with the same number of send attempts than a link with consecutive losses over the same period of time. Therefore, RNP is designed to account for the distribution of packet losses in order to accurately estimate the total number of transmissions needed in an Automatic Repeat Request (ARQ) enabled network. We implemented RNP in TinyOS and compared its performance with 4Bit experimentally in Chapter 3. The results show that RNP performs similarly with 4Bit with a tendency of using long links.

2.3.2 Physical Layer Parameters as Link Quality Metrics

Other than metrics based on packet reception, the physical layer can provide immediate information on the wireless channel as well as the quality of received packets. In general, two kinds of information can be extracted from a received packet: Received Signal Strength Indicator (RSSI) and Signal to Noise Ratio (SNR). Radio chips that are based on the IEEE 802.15.4 standard such as TI CC2420 [CC2] also implements another metric called Link Quality Indicator (LQI).

The correlations between these PHY parameters and PRR have been well studied. Theoretically, PRR can be computed using SNR and other radio parameters such as the modulation scheme, encoding scheme, frame and preamble lengths [ZK04]. However, experimental work with early platforms [ZG03] showed that it is difficult to make good estimation for low and intermediate quality links using RSSI values. Their measurements show that links with PRR of at least 95% had high RSSI, but the converse was not true, meaning that high RSSI can not direct correlate to high quality. Lai *et al.* [LMH03] measured the wireless link

quality variation for several days in two different environments, and the results show that the expected packet success rate (PSR) can be approximated by SNR with a sigmoid function. Moreover, they found that the measured PSR vs SNR curves in different networks have similar shape but shift different amount with respect to SNR. Based on the observation, they proposed an energy efficient cost metric named link inefficiency, which is calculated with the reciprocal of PSR weighted by the distance between the measured SNR value and the "knee" point in the sigmoid curve.

Later work by Son *et al.* [SKH06] confirmed their findings in newer sensor platform. Their experimental study show that by using a regression model, the signal-to-interference-plus-noise-ratio (SINR) can be mapped to PRR with very high precision ($R^2 > 0.9$). They also confirmed that a high packet reception ratio is guaranteed if SINR is higher than a threshold. However, they also found a significant variation of about 6 dB in the threshold for different radios operating at different transmission powers. These hardware variations make it difficult to distill a universal SNR to PRR relationship that is applicable to networks with multiple nodes. Similarly, Senel *et al.* [SCL07] proposes a SNR based estimator which uses a pre-calibrated SNR-PSR curve to estimate the PSR with SNR processed by a Kalman Filter.

In addition to RSSI and SNR, another metric called Link Quality Indicator (LQI) is also available in the TI CC2420 radio chip. The availability of this metric has led several routing protocols [Mul,FGJ07,GBP10] to adopt LQI as the quality metric of choice. However, Srinivasan and Levis [SL06] soon discovered that despite its wide adoption, LQI does not correlate to packet reception as well as RSSI in their experimental work.

There are also attempts to create hybrid link estimators that employ the

physical layer parameters in addition to the PRR based metrics. For example, F-LQE proposed by Baccour *et al.* [BKY10] is a fuzzy logic link quality estimator that utilizes linear membership functions to compute the quality estimation based on the four characteristics: packet delivery (PRR), link asymmetry, stability and SNR. The aforementioned 4Bit [FGJ07] is another hybrid link estimator as it utilizes the white bit from the physical layer. Rondinone *et al.* [RAR08] also suggested a reliable and efficient link quality indicator by combining PRR and normalized RSSI. Boano *et al.* [BZV10] proposed the Triangle metric, a fast estimator suitable for mobile environments. It geometrically combines PRR, SNR, and LQI together and uses empirical-based thresholds to indentify the link quality. LETX [GBP10] proposed by Gomze *et al.* utilizes a pair-wise linear model to map LQI values to link reception ratio directly. The authors argued that although LQI is not reliable for intermediate quality links, it is adequate for a reactive routing approach. Many of the link estimators mentioned here are also discussed in the comprehensive survey by Baccour *et al.* [BKM12], along with a taxonomy of existing link quality estimation techniques and their performance analysis.

In general, it is hard to find a well defined correlation between the PHY parameters and packet reception over different links and even different networks. As a result, state of art routing protocols such as CTP [GFJ09] often utilize PRR based link estimation. In our work, we show that physical layer information is an important component in the prediction of future reception, and when combined together with PRR based metrics they yield improve performance results.

2.3.3 Link Estimation with Burst Link Behavior

Although WMEWMA is highly accurate and has a small settling time for low and high quality links, i.e. links with average PRR lower than 10% or higher than 90%, it does not perform well when monitoring intermediate links that often show short temporal quality variations. Prior research indicates that most of link quality variation is observed in links with intermediate quality[1] [ZG03, WTC03, ABB04, CWK]. Moreover, short time estimation on link quality often shows a bursty pattern on packets reception, which implies that packet loss is correlated [SKA, ALB09].

To quantitatively study the characteristics of the burst link behavior, Srinivasan *et al.* [SKA] defined a β factor that quantifies the burstiness of a link. The β of a link is computed using the Kantorovich–Wasserstein distance between this link and an ideal bursty link. Their study showed that β is affected by the time interval between each packet sent: high β is usually observed with short inter-packet intervals such as 10 milliseconds, whereas β is much lower on longer intervals such as 500 milliseconds. To exploit the bursty links, they propose opportune transmission, which transmits data packets as soon as possible until the first loss occurs and waits 500 milliseconds before retransmit. Their results showed that the routing protocol can potentially decrease the delivery cost over high β links when using opportune transmissions.

Alizai *et al.* [ALB09] proposed to apply a Bursty Routing Extension (BRE) to the existing routing protocols that utilizes a short term link estimator (STLE) to detect short-term reliable links. The STLE is designed based on the heuristic that any link, no matter of what quality, becomes temporarily reliable after three consecutive packets are received over that link. If an overhearing STLE node has

[1] We consider links with PRR between 10% and 90% are of intermediate quality.

a better path cost than the packet's destination after receiving three consecutive packets, the BRE notifies the sender to send the future packets to the overhearing node instead of the original next hop. The sender switches back to the original next hop immediately on packet loss.

The design of STLE is similar to the receiver-initiated link estimators proposed in Chapter 4 and 5, but the underlying estimation technique is completely different: our estimation is based on trained models that predict the link quality with PRR and parameters from physical layer, whereas the STLE deems a link reliable based on three consecutive packet receptions. Furthermore, we do not necessarily stop using a temporary parent based on a packet loss, but rather, make an informed decision based on the output of a cost function that considers the costs of sending both control and data packets and the expected probability of success output by our prediction model.

2.3.4 Model Based Link Quality Estimation

Applying data-driven methods on link quality prediction has been less studied. K. Farkas *et al.* [FHR06] made link quality predictions using a pattern matching approach based on SNR. The main assumption is that the behavior of links shows some repetitive pattern. The authors suggest that the above assumption is valid for 802.11 wireless ad-hoc networks. Furthermore, they proposed XCoPred (using Cross-Correlation to Predict), a pattern matching based scheme to predict link quality variations in 802.11 mesh networks in [FHL08]. 4C differs from XCoPred in several aspects: 4C combines both PRR and PHY parameters with prediction models trained off-line, whereas XCoPred considers SNR only and uses cross correlation without prior training. Wang *et al.* [WMP07] used a decision tree classifier to facilitate the routing process. Their approach is to train the decision

tree offline and then use the learned rules online. The results show machine learning can do significantly better where traditional rules of thumb fail. However, they only considered RSSI in their input features and overlooked other physical layer information, whereas our modeling explore much more parameters with different traffic patterns.

CHAPTER 3

Comparative Study of Routing Performance

As detailed in the previous sections, the routing performance of WSNs are heavily affected by the underlying link quality estimation metrics due to the dynamic and asymmetric nature of the wireless links. In previous research [DAB03, DPZ04, WTC03, GYH04], the link reliability estimation metrics such as ETX have been proved to have better performance than the conventional metrics, such as shortest path or minimum latency, in wireless networks. In this section, we further evaluate the impact of the routing performance with respect to three of the most commonly used link estimation techniques, namely, ETX [DAB03], RNP [CWP05] and 4Bit [FGJ07]. By conducting an extensive performance comparison between ETX, 4Bit and RNP in a static, multihop wireless sensor network testbed, we hope to understand the pros and cons of each link estimator, and more importantly, to find the potential improvements on the existing link metrics in order to further increase the routing performance in WSNs.

We also investigate the effects of blacklisting on ETX, 4Bit and RNP. A link estimator with a blacklisting policy will only consider links with quality above a threshold, and thus minimizes the potential costs for estimating a low quality link that will not be used in routing. Such a threshold is necessary to filter out neighbors with low quality links and avoid pollution of the neighbor table. However, the blacklisting policy may hide some neighbors to the routing protocol, reducing the number of available routing choices. To quantify the difference

in performance derived from blacklisting, we conducted experiments using the same link estimator under the same network configuration, with and without blacklisting.

Contribution:

In this chapter, we evaluated the performance of ETX, RNP and 4Bit in a variety of network configurations. We are the first to provide an implementation of the RNP metric in TinyOS. By applying different link quality metrics to the same routing protocol, CTP, we compared their performance with regards to path length, path quality, delivery rate, transmission overhead and network stability. In addition, we studied the effects of those metrics in the presence of different blacklisting policies and discovered several interesting properties.

In the following sections, we first introduce the three link quality estimator (ETX, RNP and 4Bit) in detail, and then present the experimental methodology of the performance evaluation in Section 3.2. Section 3.3 discuss the evaluation results and compare the performance of the link estimators. A discussion of the findings in this comparative study is presented in Section 3.4.

3.1 Link Quality Metrics

3.1.1 Expected Number of Transmissions

The purpose of ETX is to minimize the number of transmissions for data packets. ETX estimates the number of transmissions needed to send a unicast packet by measuring the delivery rate (or packet reception ratio) of beacon packets between neighboring nodes. The ETX metric for a link can be calculated as $\frac{1}{d_f \times d_r}$, where d_f and d_r are forward and reverse delivery rates for a link. To compute d_f and d_r, each node broadcasts beacon packets periodically. Every beacon packet contains

the reception rates of beacons received from each of its neighbors. From the beacon packets, a node can read d_f, the delivery rate from itself to its neighbors, and compute the reverse delivery rate d_r by counting the number of lost beacons from its neighboring nodes.

ETX is implemented in the link estimator of the Collection Tree Protocol (CTP) [GFJ09] in TinyOS. By default, the link estimator employs a blacklisting policy to filter out neighbors with low link quality.

3.1.2 Requested Number of Packets

Cerpa *et al.* [CWP05] proposed RNP in their study of temporal properties of low power wireless links. The goal of RNP is to account for the distribution of packet losses of a link when estimating link quality. In their study, they discovered that among links with similar delivery rates, a link with isolated losses can deliver more data packets with the same number of send attempts than a link with consecutive losses over the same period of time. The aim of RNP is to measure the total number of transmissions needed in an Automatic Repeat Request (ARQ) enabled network where the underlying packet loss distribution is known.

In this chapter, we provide the first actual implementation of RNP for TinyOS. With the same link estimator architecture implemented in CTP, we can compute the RNP of the links between a node and its neighbors. Like ETX, RNP estimator broadcasts beacon packets periodically. To compute RNP for a link, a node records the sequence number of beacons broadcasted by its neighbors and calculates the difference in sequence number (D) between the last two received beacons. D value greater than 1 indicates packet losses, creating a gap in the continuous sequence numbers. In an ARQ network, the lost packets will be retransmitted repeatedly until they are acknowledged by the recipient. Assuming

the gap indicates a loss period during which all transmission attempts will fail, then the number of retransmission attempts (R) for the lost beacons of the gap can be calculated as the following:

$$R = \frac{D \times (D-1)}{2}$$

To be generic, let us assume N beacons are sent during one estimation. Since multiple packets may be lost during the transmission of total N beacons, multiple gaps may exist. Let G be the number of gaps appearing during the reception of N beacons, and R_i be the number of retransmission attempts for the ith gap. The gaps indicate the underlying packet loss distribution during the transmission of N beacons, so we can calculate the number of transmission required for delivering one packet as the following:

$$r = \frac{N + \sum_{i=1}^{G} R_i}{N}$$

where r is the total number of transmissions attempts needed for reliably delivering one packet under the packet loss distribution detected by the reception of beacons. A node piggybacks the r value in its own beacon packets so that its neighbors know the forward direction link quality. Therefore, a node can estimate the backward direction link quality by counting the lost beacons and read the forward direction link quality from received beacon packets. For a link, we define the RNP metric used by the routing algorithm as follows:

$$RNP = r_f \times r_b$$

where r_f and r_b are link quality estimates in forward and backward direction, respectively.

RNP metric has the same range as ETX. A lower RNP value indicates better link quality, 1 meaning the link quality is 100%. But it differs from ETX in a

number of ways. Firstly, RNP is applicable when a retransmission mechanism is in place, at the MAC or the network layer. Secondly, RNP tends to be lower for links with isolated losses compared to links with consecutive losses. As a result, the RNP value of a link can be quite different from ETX.

3.1.3 Four-Bit

The 4Bit [FGJ07] link quality estimator (also referred as 4B) provides well-defined interfaces that combine information from the physical, data-link and network layers. 4Bit uses ETX as its link quality metric, but it also provides 4 bits of information compiled from different layers: a *white bit* from the physical layer, denoting the low probability of decoding error in received packets. An *ack bit* from the link layer to indicate whether an acknowledgment is received for a sent packet. The *pin bit* and the *compare bit* are from the network layer. Routing protocols use the *pin bit* to keep important nodes in the neighbor table maintained by the link estimator and the *compare bit* to indicate the importance of a link.

The 4Bit link estimator implemented in TinyOS operates as follows: the *compare bit* interface takes the beacon message received from a neighbor and the *white bit* as inputs, and finds the neighbor has better quality link, and, more importantly, the neighbor that is irreplaceable for routing proposes. The criteria to set the *white bit* differs for different platforms. For the CC2420 radio, the *white bit* is set when the Link Quality Indication (LQI) of a packet is higher than 105, which corresponds to a reception rate higher than 90%.

3.2 Experimental Methodology

3.2.1 Routing Protocol

Collection Tree Protocol (CTP) [GFJ09] is the default tree-based multihop collection protocol in TinyOS 2. We use it with the three link quality estimators to evaluate the routing performance.

In CTP, every node maintains an cost value for itself and uses a link estimator to estimate the transmission cost of the nearby neighbors. The cost value is used as the routing gradient: the root nodes (basestations) have the lowest possible cost (0 in the cases of ETX and RNP), and nodes away from the root nodes will have additive cost values of all the hops between the node and the nearest root node. To maintain the cost gradient, each node broadcasts beacon messages which contain its cost value. Upon receiving the beacon message from a neighbor, a node will calculate the path cost of the neighbor, i.e., the cost value of the neighbor plus the transmission cost of its link to this neighbor. A node will consider the neighbor with lowest path cost as its next hop (called parent in CTP) and set its cost gradient to the path cost of the parent node.

CTP has three components: a link estimator, a routing engine, and a forwarding engine. The link estimator is responsible for estimating the transmission cost of the links to single-hop neighbors. The routing engine is in charge of choosing the next hop (parent) based on the link estimation as well as network-level information such as congestion, whereas the forwarding engine maintains a queue of packets to send. In this comparative study, I reimplemented the ETX based link estimator to incorporate RNP as the link quality metric, and used ETX, RNP and 4Bit in the following evaluation.

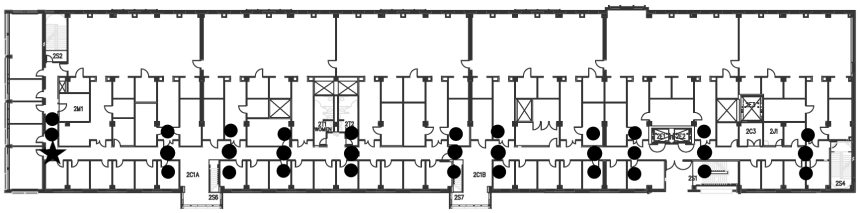


Figure 3.1: The testbed consists of 33 nodes, divided in 11 groups. The star denotes the root node (basestation).

3.2.2 Local Sensor Network Testbed

Our local sensor network testbed is located on one floor of an office building, with the nodes placed along the ceiling of a long corridor. The nodes locations are fixed for all our experiments as seen in Figure 3.1.

There are 33 nodes in total, organized into 11 groups of 3 nodes. The nodes are MoteIV Tmote Sky, which comprises of a TI MSP430 micro-controller and a Chipcon CC2420 radio for wireless communications. Within one group, nodes are separated by one foot distance. All the motes are connected to a central server, thus serial communication is enabled between the nodes and the server.

3.2.3 Experiment Settings

The experiments were conducted under four different network configurations: 11 nodes with 0 dBm and -10 dBm transmission power; 33 nodes with 0 dBm and -10 dBm transmission power. Under each configuration, we conducted one-hour experiments for all three metrics (ETX, RNP, 4Bit) with and without blacklisting.

In CTP, the root node acts as the sink for the routing tree. To increase the diameter of the network, the root node was set at one end of the corridor (depicted as a star in Figure 3.1). In the 11-node experiments, only one node in

each group was used. Except for the root node, all nodes send 1 packet/sec to the sink using CTP. In the 33-node experiments, all the nodes are used and the packet sending interval was set to 5 sec to avoid collisions. In all experiments, the payload of each packet was set to 100 bytes, resulting in a packet length of 117 bytes.

In summary, we used 2 power levels ($0dBm$ and $-10dBm$), 2 network densities (11 and 33 nodes), 2 blacklisting policies (with and without) for each link quality estimation metric. We conducted a total of 24 experiments covering the complete set of parameters for each link quality metric. In the 11-node experiments, each node sends about 3,600 data packets with 36,000 packets sent in total. In the 33-node experiments, each node sends about 720 data packets with 23,040 packets sent in total. For each packet, we kept a complete record of the path taken and the number of transmissions on each hop as it is being forwarded by CTP. Our evaluation is based on more than 700,000 packet traces we collected. Note that although the nodes are placed along a corridor, the actual topology is more like a mesh network due to the large number of links. The number of unique paths used by a node can exceed 40 during one experiment.

3.2.4 Data Collection

For each experiment, the end to end delivery rate, latency and path taken is recorded for each data packet. An exponentially weighted moving average (EWMA) is used to compute the reception rates of each link with non-zero throughput. This gives a good approximation of the instantaneous reception rate of the link at the time the packet was sent through it. To measure the quality of the path taken by the packet, we computed the product of the instantaneous reception rate of each link that constituted its path. For each path,

we also record the number of attempts made for sending the packet at each hop along the path. The sum of all the send attempts per hop is the the number of transmissions required for sending a packet through along that path.

3.3 Evaluation Results

In this section, we describe the results of our experiments. Firstly, we present the performance evaluation metrics used to analyze our results. Secondly, we compare the experimental results with the evaluation metrics. Note that the results presented in this section are from experiments with the presence of a blacklisting policy. Finally, we discuss the impact of blacklisting policy on routing performance.

3.3.1 Evaluation Metrics and Parameters

We used the following matrics to measure the routing performance.

Path Length is the number of hops along a path. It measures routing depth of nodes in the network. The path length affects end-to-end latency and energy usage.

Path Quality is the product of link quality for each hop along the path. To provide a uniform measurement for path quality, we define the link quality as the reception rate of a directed link at the time a packet is transmitted. Path quality reflects the end-to-end reliability of a path in the absence of retransmissions.

End-to-End Delivery Rate is the total number of packets received at the sink from a specific source node divided by the total number of packets originating from that source node.

Transmission Overhead refers to the number of send attempts needed to

deliver a packet to the sink via an established route. It can be considered as the cost of delivering a packet, which is the summation of the send attempts including retransmissions, at each hop along the path. Transmission overhead is proportional to the total power consumption for delivering a packet in the network, as well as the end-to-end latency.

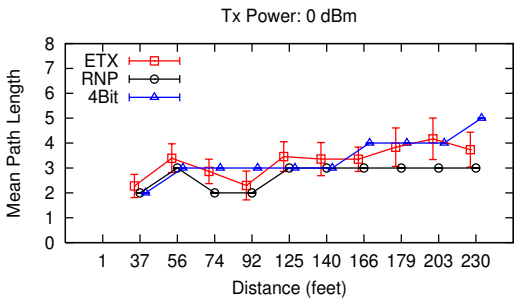
Routing Overhead is the cost of maintaining a connected network. It can be represented by the number of beacon packets sent during a experiment because the CTP maintains its routing tree by broadcasting beacons.

Stability measures the total number of routing topology changes in the network over a period of time.

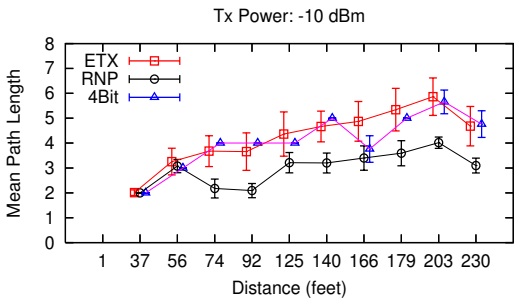
Most graphs are plotted with the average value and error bars with standard deviation. The values of the different routing metrics are slightly shifted on the x axis to improve readability with overlapping error bars. We also use boxplots in some graphs showing max, min, 1st, 2nd, 3rd quartiles, and average values, so we can provide a better understanding of the underlying distribution of the results. We now proceed to discuss the performance the different link quality estimation metrics with respect to these evaluation metrics.

3.3.2 Path Length

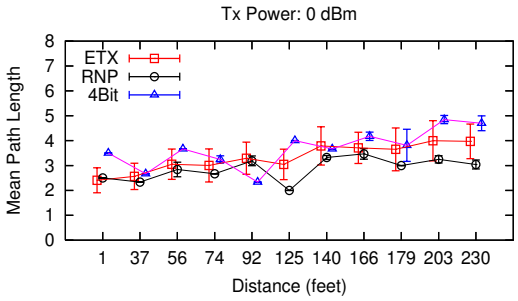
Figure 3.2 shows the average path length as a function of the distance between source nodes and the sink. In general, the higher the transmission power, the shorter the average path length for any source-sink pairs since high power links can cover a long distance. This can be seen by comparing Figures 3.2(a) with 3.2(b) and Figures 3.2(c) with 3.2(d) respectively. In the 33-node experiments this tendency is less apparent because the high network density leads to a larger number of available paths compared to the sparse network in the 11-node



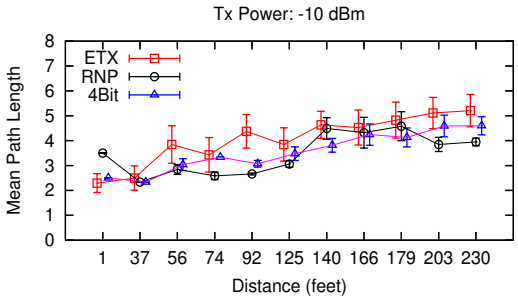
(a) High Power, 11 Nodes



(b) Low Power, 11 Nodes



(c) High Power, 33 Nodes



(d) Low Power, 33 Nodes

Figure 3.2: Mean path length with respect to node-sink distance.

experiments.

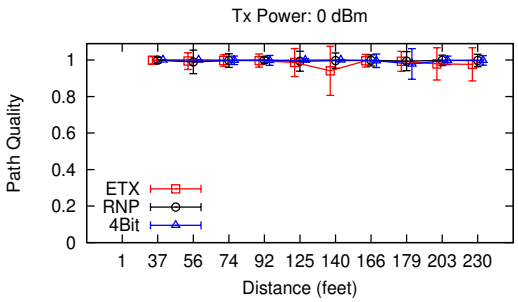
With some exceptions, there is a general trend of increase in path length as the distance from the source to the sink increases. The exceptions indicates that some links are considerably longer than the others, and they are quite stable. This can be explained with multipath effect. For example, in Figure 3.2(b), the nodes farthest from the sink on average have shorter path length than the second farthest nodes since they are at the end of a corridor.

Moreover, we can see that the average path length chosen by ETX and 4Bit is very close. This observation is not surprising since ETX is part of the 4Bit metric for link estimation. However, RNP chooses shorter paths than both ETX and 4Bit in the most cases. It is because RNP is less sensitive to sparse packet losses, allowing the routing protocol to select links with reasonably high quality that cover longer distances. This is an important characteristic of RNP, and we will discuss its impact on routing performance in the following sections.

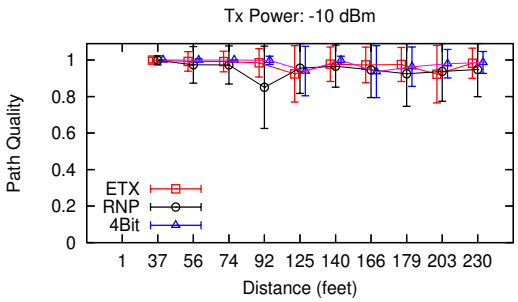
3.3.3 Path Quality

Figure 3.3 shows the average path quality as a function of the distance between source nodes and the sink. In almost all scenarios, the path qualities are above 90%. Only a few exceptions exist in the low power, high network density experiment with RNP. This indicates the most of the paths are constituted of high quality links.

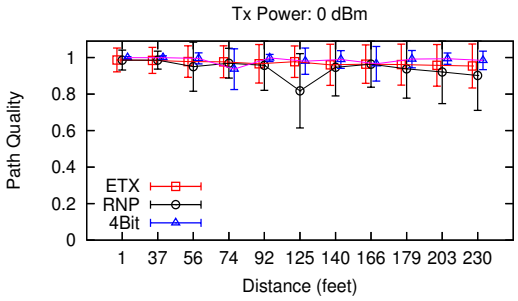
For low network density, the three metrics do not show significant differences and tend to pick high path quality links for both low and high power levels, as seen in Figures 3.3(a) and 3.3(b). For high network density, RNP tends to pick paths that do not necessarily have high path quality whereas ETX and 4Bit tend to pick paths with better quality than RNP, as shown in Figures 3.3(c)



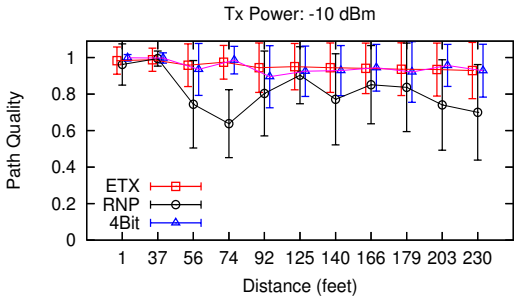
(a) High Power, 11 Nodes



(b) Low power, 11 Nodes



(c) High Power, 33 Nodes



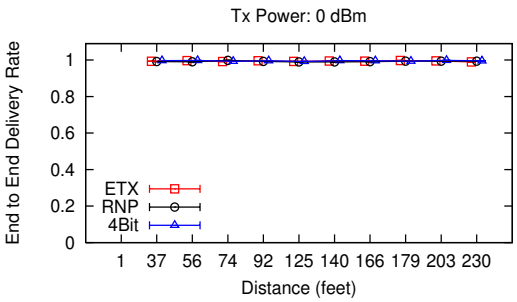
(d) Low power, 33 Nodes

Figure 3.3: The average path quality with respect to node-sink distance.

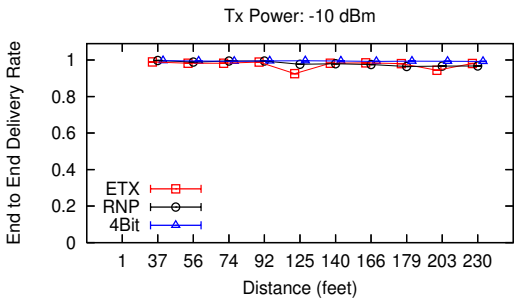
and 3.3(d). In order to investigate the reason for this behavior, we need to look into the definition of RNP. Compared with ETX, RNP gives higher estimation to links with sparse packet losses. When the average reception rate of a link is moderate, e.g., above 70%, the number of lost packets is small compared to the number of delivered packets, and number of consecutive losses should be even smaller. In this case, RNP will give high estimate to the moderate links, making the routing protocol to consider them as well as the high quality links. Since the moderate links can likely cover longer distance than the high quality links do, the routing protocol will be able to select paths with less number of hop using RNP. Note that moderate link quality does not necessarily mean moderate delivery rate due to the retransmission mechanism.

3.3.4 Delivery Rate

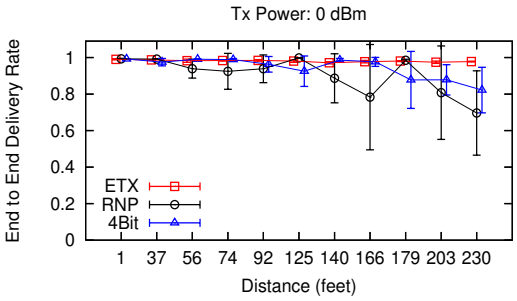
Figures 3.4(a)– 3.4(d) show the average end to end delivery rate as a function of the distance between source nodes and the sink. For low network density (see Figures 3.4(a) and 3.4(b)), the delivery rate is higher than 90% and remains unaffected by the distance and the metric used. In particular, 4Bit achieves an impressive near 100% delivery rate. In high network density and low power scenario, the average delivery rate remains above 90%, except for a few nodes, as seen in Figure 3.4(d). For high power and high network density scenario in Figure 3.4(c), the differences are more apparent. ETX keeps a high delivery rate for all the nodes, RNP shows some lower delivery rate in some nodes, and the delivery rate for 4Bit drops as the distance increases. The worst average delivery rate for any distance using ETX is 96%, whereas the worst average delivery rate for any distance using RNP and 4Bit is 70% and 82% respectively. Investigation of the cause revealed that the delivery rate drop is caused by hot-spots in



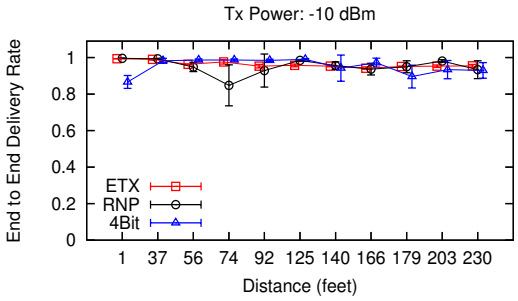
(a) High Power, 11 Nodes



(b) Low power, 11 Nodes

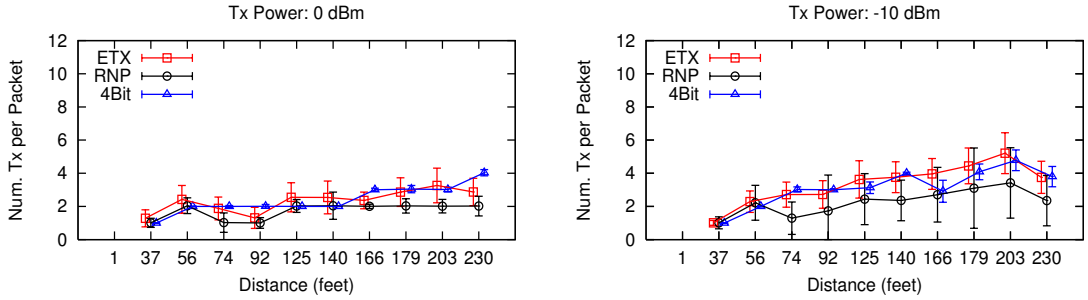


(c) High Power, 33 Nodes



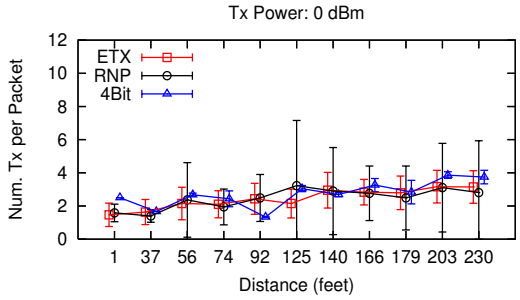
(d) Low power, 33 Nodes

Figure 3.4: The average delivery rate with respect to node-sink distance.

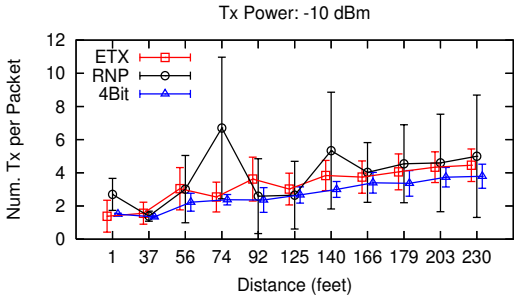


(a) High Power, 11 Nodes

(b) Low Power, 11 Nodes



(c) High Power, 33 Nodes



(d) Low Power, 33 Nodes

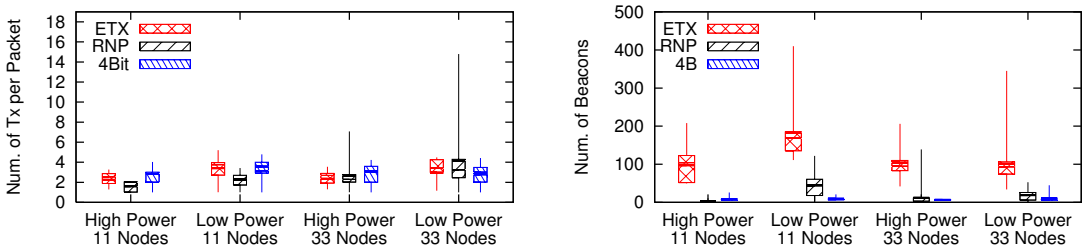
Figure 3.5: The total number of transmissions with respect to node-sink distance.

the network topology. The reason of the formation of hot-spots is explained in subsequent sections.

3.3.5 Transmission Overhead

The transmission overhead is measured in terms of transmission attempts per packet. Figures 3.5(a)-3.5(d) show the average transmission overhead as a function of the distance between the source nodes and the sink.

In general, the transmission overhead is very close to corresponding the path length, suggesting that most packets are successfully received in the first send attempts. This is consistent with the high path quality observed in Figure 3.3. An exception is the low power, high network density scenario in Figure 3.5(d),



(a) Transmission Overhead

(b) Routing Overhead

Figure 3.6: Boxplots of overall transmission overhead and routing overhead

Transmission Overhead	**High Power 11 Nodes**	**Low Power 11 Nodes**	**High Power 33 Nodes**	**Low Power 33 Nodes**
RNP vs. ETX	**0.0048**	**0.0027**	0.2693	0.1649
RNP vs. 4Bit	**0.0028**	**0.0013**	0.1356	0.1291

Table 3.1: P-values from t-tests run on the transmission overhead results of RNP vesus ETX and 4Bit. A p-value smaller than 0.05 indicates the cost difference between 4C and 4Bit are significant with 95% confidence.

during which the transmission overhead of RNP increases drastically for some nodes. It can be explained by RNP's low path quality in this case, as depicted in Figure 3.3(d).

Note that although the number of transmissions required per hop are close to one for all three metrics, the total costs for sending a packet to the sink varies because of the different path lengths and qualities. Figure 3.6(a) plots of the overall transmission overheads in all four scenarios. In low density scenarios, the overhead of RNP is lower than ETX and 4Bit as it sends more packets through short paths than ETX and 4Bit. Even though these paths have slightly lower path quality comparing to those chosen by ETX and 4Bit, the overall cost of sending a packet is still *significantly lower* in the low density cases. However, in high

density scenarios with 33 nodes, RNP perform on par or slightly worse compared with ETX or 4Bit. Table 3.1 lists the p-values of t-tests on the RNP with respect to ETX and 4Bit results. The p-values present the statistical significance of these results, i.e., if the p-value is smaller than 0.05, the transmission cost restuls are significantly different from each other with 95% confidence. As presented in Table 3.1, RNP shows significant difference with respect to ETX and 4Bit in low density scenarios with 11 nodes, whereas in high density case the transmission overhead of RNP is not statistically different from ETX or 4Bit.

Also note that in low power scenarios (see Figures 3.5(b) and 3.5(d)), 4Bit showed a slight improvement over ETX. The reason behind the 4Bit's superior performance might be the interoperability between link estimator and the routing protocol introduced by the *pin bit* and *compare bit*, which enable the estimator to be aware of the important neighbors in the routing points of view. Given the small neighbor table in the estimator and the large number of neighbors available with different link qualities in a high density network, such interoperability can provide valuable information for a better neighbor management.

3.3.6 Routing Overhead

The routing overhead is the number of beacon packets sent during an experiment. The beacon packets are used by both CTP and the link estimator: CTP broadcasts beacons to proactively maintain a routing tree, while the link estimator use the same beacons to piggyback link quality information. Link estimators rely on the beacon packets to perform link quality measurements, but the beacon broadcasting is fully controlled by CTP. CTP immediately broadcasts a new beacon when the next hop of the current node changes or it detects a better path than the existing one. Otherwise, CTP will broadcast beacon packets periodically.

Parameters			Unique Path			Tree Change		
Power	Nodes	BL	ETX	RNP	4Bit	ETX	RNP	4Bit
0 dBm	11	Y	157	11	10	310	1	0
-10 dBm	11	Y	273	44	18	585	47	8
0 dBm	33	Y	795	51	39	1296	19	7
-10 dBm	33	Y	874	79	59	1159	53	27
0 dBm	11	N	152	11	39	325	1	38
-10 dBm	11	N	369	16	10	927	6	0
0 dBm	33	N	366	203	64	495	211	41
-10dBm	33	N	950	136	11	1296	123	3

Table 3.2: Routing topology changes during the experiments.

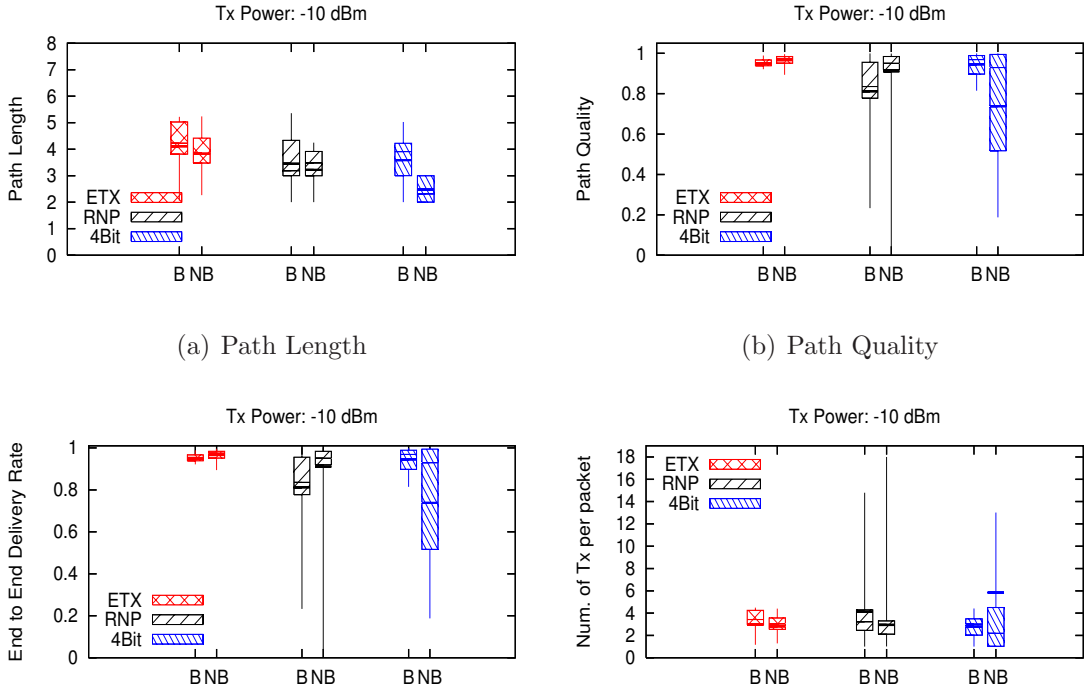
Figure 3.6(b) illustrates the number of beacons sent in all four scenarios. When using ETX as link estimator, CTP sends significantly more beacons than it does with RNP or 4Bit. The routing overhead of RNP is slightly larger than 4Bit (although not statistically significant). The high routing overhead of ETX indicates frequent routing changes, whereas the low routing overhead of 4Bit means CTP rarely changes the routing tree with 4Bit. In the case of ETX, it advise CTP changes its next hop whenever a better path is available, causing CTP constantly send beacons to inform the neighboring nodes about the route change. However, with 4Bit, routing changes hardly occur because CTP can use *compare bit* to keep important neighbors in the neighbor table. RNP reacts to link quality changes slower than ETX, but still more responsive than 4Bit.

3.3.7 Stability

The stability of a routing topology is a important factor for high level operations like scheduling and aggregation. In CTP, every node in the routing tree has one and only one parent node. Whenever the parent node changes, the routing tree will change. Table 3.2 lists the number of routing topology changes and the number of unique path taken during each experiment. We observe that nodes with ETX change parents much more frequently than RNP or 4Bit. 4Bit exhibits a very stable routing topology due to the fact that the routing protocol can direct 4Bit to keep certain nodes in its neighbor table for better estimation. RNP is slower to react to quality changes and hence has lower route changes than ETX but slightly higher than 4Bit. The high number of route changes for ETX reflects its tendency to pick perfect quality links at all times. These greedy approach leads to higher routing overhead, as shown in the previous section.

Having a stable routing tree has both advantages and disadvantages. On the one hand, higher-level applications can take advantages of a stable routing tree for in-network data processing, but on the other hand, high density networks with stable routing trees will result in the formation of hot-spots. Since RNP and 4Bit are more stable with regards to the routing topology, most of the packets are sent along the same route during the entire experiment despite the existence of available alternatives routes to the sink. This leads to formation of hot-spots in the network. Nodes in the hot-spots are overwhelmed by incoming packets, forcing them to drop some packets, hence increasing the packet losses that lead to a drop in path quality and end-to-end delivery rate.

For example, in the high power, high network density experiment (see Table 3.2, third line), when using ETX the routing topology changed 1296 times in total and almost all the nodes participated in forwarding packets. In contrast,



(a) Path Length

(b) Path Quality

(c) Delivery Rate

(d) TX Overhead

Figure 3.7: Path length, path quality, delivery rate and transmission overhead for experiments with and w/o blacklisting in 33-node network and low power.

RNP had 51 unique paths, suggesting that each node uses less than 2 alternative routes to forward packets on average. Similarly, 4Bit had 39 unique paths, and there are only 7 routing changes, which means most nodes used only one route to forward packets. In the case of ETX, the load of forwarding packets is evenly distributed on the whole network. However, in the cases of RNP and 4Bit, most of the packets were forwarded along fewer (or almost one in the case of 4Bit) several high quality paths, creating hot-spots in the network and degrading the delivery rates, as seen in Figure 3.4(c).

3.3.8 Impact of Blacklisting

In this section, we evaluate the effect of imposing a blacklisting policy. All results presented in previous sections were collected with blacklisting activated. By default, the TinyOS link estimator for ETX and 4Bit employs a blacklisting policy with a threshold of delivery rate 18%. For RNP, the threshold cannot be represented simply as a fraction of delivery rate because the RNP value changes as the distribution of losses change. In our implementation, RNP blacklists a link when delivery rate drops below approximately 20%.

Our experimental results do not exhibit obvious differences in low network density configurations regardless of the presence or absence of blacklisting. This is because the link estimator has the capacity to handle all 11 nodes in the network. However, in the 33-node experiments, RNP and 4Bit perform quite differently with or without blacklisting whereas ETX is not affected by the absence or presence of blacklisting. For the sake of brevity, we omitted the figures in high network density and high power since they present similar behavior than the low power scenarios.

Figure 3.7 shows the three routing metrics operating with (B) and without (NB) blacklisting in the terms of overall path quality, path length, delivery rate and transmission overhead. Figure 3.7(a) shows an improvement in all three metrics by reducing the path length, with the most notable improvement by 4Bit. However, this improvement comes at a cost for the 4Bit case, since there is a significant path quality and end-to-end delivery degradation as seen in Figures 3.7(b) and 3.7(c). Both ETX and RNP actually improve path quality without blacklisting. Similarly, in Figure 3.7(d) the overall transmission overhead is improved in both ETX and RNP without blacklisting, while the 4Bit case makes the average case worse.

3.4 Discussion

As discussed in Section 3.3, RNP is more tolerant to losses when the link quality is moderate. So, RNP gives the better estimated value to links with a wider range of delivery rates as compared to ETX. From the routing perspective, RNP allows the routing protocol to select paths with less total numbers of hops to set up a route with reasonable quality. Although packet losses occur more frequently in longer links of moderate quality, the retransmission mechanism in the routing protocol can still ensure a high delivery rate with a few retransmissions. This is reflected in our experiments where RNP chooses paths with less hops than the other two metrics as seen in Figure 3.2. This characteristic of RNP leads to a smaller overhead for delivering a packet, while maintaining a high end to end delivery rate that is comparable to that of ETX and 4Bit as seen in Figures 3.4(a) and 3.4(b). However, RNP's tolerance of sparse losses make it less responsive to changes in link quality than ETX. This causes RNP to change route when the link quality to the next hop drops to a very low level. In this case, the retransmission can no longer compensate for the high packet losses. Moreover, in high network density environments, excessive retransmissions increase contentions in the network, lowering the end to end delivery rates of RNP in comparison to ETX and 4Bit, as seen in Figures 3.4(c) and 3.4(d).

ETX and 4Bit exhibit the same preference in choosing paths with near perfect links. As a result, the average path length for ETX and 4Bit is almost the same. Since ETX selects parent nodes based solely on the quality of its neighbors, it has better path quality and is more adaptive to changes in link quality in a variety of scenarios. This very greedy adaptivity comes at a cost, since ETX significantly increases the routing overhead of the routing algorithm by constantly trying to pick the best instantaneous neighbor, as shown by Table 3.2. While this behavior

has the advantage of implicitly spreading the load among all nodes, we believe load balancing schemes should be implemented in higher layers and not at the link layer.

4Bit allows CTP to choose better neighbors when the neighbor table is full. As a routing protocol, CTP tends to choose nodes with high quality path to the sink. This results in a very stable routing topology for the entire network (see Table 3.2), comprising of directed links to the sink. However, paths from multiple nodes to the sink will intrinsically lead to congestion at some shared links. This may cause degradation in link quality and hence, lowers the path quality and end-to-end delivery rate, especially in high density networks.

On the one hand, RNP exhibits slightly better performance in the absence of blacklisting with better path quality, higher delivery rates and less overhead, refer to Figures 3.7. This is because RNP could be more aggressive in picking longer links that would be filtered out, improving performance. On the other hand, 4Bit performs much worse without blacklisting, especially in high network density configuration. This counter-intuitive result can be explained by the design of the link estimator. Regardless of the metric, the link estimator maintains records for its 10 best neighbors by link quality. With blacklisting, a particular threshold is defined, and the link estimator rejects any new neighbor if the neighbor table is full and all the entries in it are larger than this blacklisting threshold, preventing the evaluation of potentially better quality neighbors. Without blacklisting, this threshold does not exist, and the link estimator simply replaces the lowest quality neighbor in its table, allowing the evaluation of *all* potential neighbors. This explains why ETX and RNP perform slightly better. In the case of 4Bit, if a new neighbor passes the blacklist threshold, the link estimator consults the routing protocol to evaluate the neighbor. If this neighbor has a high quality path to the

sink, the routing protocol instructs the link estimator to include this node in the neighbor table. In this case, 4Bit relies on blacklisting to filter out neighbors with low link quality. With blacklisting absent, the neighbor table will be polluted by neighbors with high quality path to the sink but low link quality to the node itself. In high density networks, the number of such neighbors will be big enough to affect the routing choices made in 4Bit, resulting in poor delivery rate as seen in Figure 3.7(c).

Finally, as brief summary, the above findings prove that RNP can reduce the packet delivery cost by utilizing long links with high quality, however it is difficult to identify these high quality, long links in various dynamic network environments with any single link quality metric. In the reminder of this chapter, we turn our attention to another critical aspect of the low-power sensor networks, i.e., the energy usage of radio communication in duty-cycled networks.

3.5 Summary of Link Qualition Metrics Comparison

This chapter examines the challenges of radio communication in low-power WSNs in two aspects: link quality estimation and energy usage in duty-cycled networks. Based on the empirical evaluation discussed in Section 3.3, we find that ETX and 4Bit exhibit a strong preference in choosing paths with near perfect links, whereas RNP tends to take advantage of long links and chooses shorter path length in comparison to ETX and 4Bit. This behavior of RNP results in better packet delivery cost in low network density scenarios than the other link estimators, however in the high network density scenarios, the same tendency causes RNP selecting paths with slightly lower quality and higher cost. These results highlight a major issue in terms of link quality estimation: the state-of-art link estimators can not accurately identify *when* an intermediate quality link is in a high quality

period, which prevents the routing protocol to utilize these long links with intermediate quality to form optimal routing topology. Chapter 4 and 5 discuss issue regarding link quality estimation of in depth and propose data-driven solutions for both long-term and short-term link estimation in various dynamic network environments.

On the other hand, Section 6.1 focus on another issue in duty-cycled networks, i.e., the energy wasted on idle waiting due to the asynchronous wakeup schedule. As seen in Figure 6.2, the majority of the energy usage is spent on idle transmitting or listening to wait for the receiver to wake up. Therefore, it is necessary to design a synchronous data forwarding protocol in order to minimize the idle transmission. Chapter 6 addresses this problem with a complete design and evaluation of such protocol.

CHAPTER 4

Data-Driven Link Quality Prediction

As pointed out in the previous chapter, most of the current link estimation metrics are based on link reliability, which means they compute the cost of delivering a packet through a link based on the packet reception ratio (PRR). For example, CTP [GFJ09], the main collection protocol in TinyOS [LMP04], uses ETX [DAB03] to create a routing gradient.

However, the performance results presented in Section 3.3 reveals the problems of PRR based metrics. First, since calculating PRR requires several packets, PRR based metrics are insensitive to link quality variations on a *per packet* level. A common practice [WTC03, FGJ07] is that the PRR based link estimators only update the link quality estimation after a predefined time interval, and therefore can not capture link quality variations on a per packet basis. Even if the link estimators are configured to be reactive and agile, PRR based metrics still can not capture the per packet link quality variations due to the windowed PRR update. As pointed out in prior work [CWP05, ALB09], these per packet level link quality variations are often observed in intermediate quality links, and by taking advantage of these long links of intermediate quality, the underlying routing protocol can reduce the number of hops in the path, and ultimately, reduce the number of transmissions for delivering a packet. Nevertheless, identifying *when* an intermediate link is in a high quality period is relatively hard for PRR based metrics due to the long data packet intervals in many WSN applications and the convergence

time of PRR itself. Second, PRR based metrics assume that the current link quality remains the same as the last estimation, but this assumption of stable link quality is often invalid due to the notoriously frequent variations of wireless links. In short, due to the above two reasons, PRR based metrics typically respond slowly to link quality changes, and even with the help of Trickle [LMP04] or an adaptive beacon policy [GFJ09] from the routing protocol, the link quality estimation may still updates too slowly.

In this chapter, we tackle these problems with 4C, a data-driven, modeling based link estimator that predicts the expected link quality on a *per packet* basis. 4C employs a machine learning approach to predict the expected quality of a link with both physical layer information and PRR. The 4C prediction models take the PHY parameters of the last received packets and PRR of the link as input, and predict the probability of receiving the next packet. In the following sections, we show that these parameters can reveal the current state of the wireless channel so that the models can perform a more accurate quality estimation than PRR.

The rest of this chapter is organized as the following. Section 4.1 details the exploratory analysis for the physical layer parameters, the data collection, the modeling process, including the model construction, model training, and an analysis of model selection for an actual implementation in resource constrained nodes. Section 4.2 describes the implementation details of 4C, and Section 4.3 presents the experimental results of the CTP with 4C. Section 4.4 discusses the advantages and the limitations of 4C, and finally in Section 4.5 we summarize the findings and move to the next chapter.

Contribution:

The contribution of this chapter is three fold. First, we analyzed and evaluated the use of both PRR and the physical layer information for link quality estima-

tion. We supplement the PRR based estimator with physical layer information to improve the estimation for intermediate quality links while maintaining accurate estimation for stable links.

Then, we developed the data-driven prediction models to be used for online link quality prediction. We show that these models, with the appropriate set of parameters, can be implemented in resource constrained nodes with very limited computation capabilities and small overhead.

Third, we designed 4C, a receiver-initiated online prediction module that informs the routing protocol about the short temporal high quality links, enabling the routing protocol to select temporary, low-cost routes in addition to the stable routes. Experimental evaluation showed that 4C improves the average cost of delivering a packet by 20% to 30%. In some cases, the improvement reaches 46%.

4.1 Modeling Approach

We propose to use machine learning methods to build models that predict the link quality with information from both physical layer and link layer. To predict the quality of the wireless links, we use a combination of the PHY parameters (RSSI, SNR and LQI) and the PRR filtered by a window mean estimator with exponentially weighted moving average (WMEWMA) [WTC03]. The intuition behind our approach is quite simple. We supplement the WMEWMA estimator, accurate mainly for high and low quality links that are stable in nature, with physical layer information to improve the estimation for the intermediate quality links which are highly unstable and show the most variation.

We first motivate our research with an exploratory analysis. Then, we formally define the problem trying to solve, the modeling methods, and the proce-

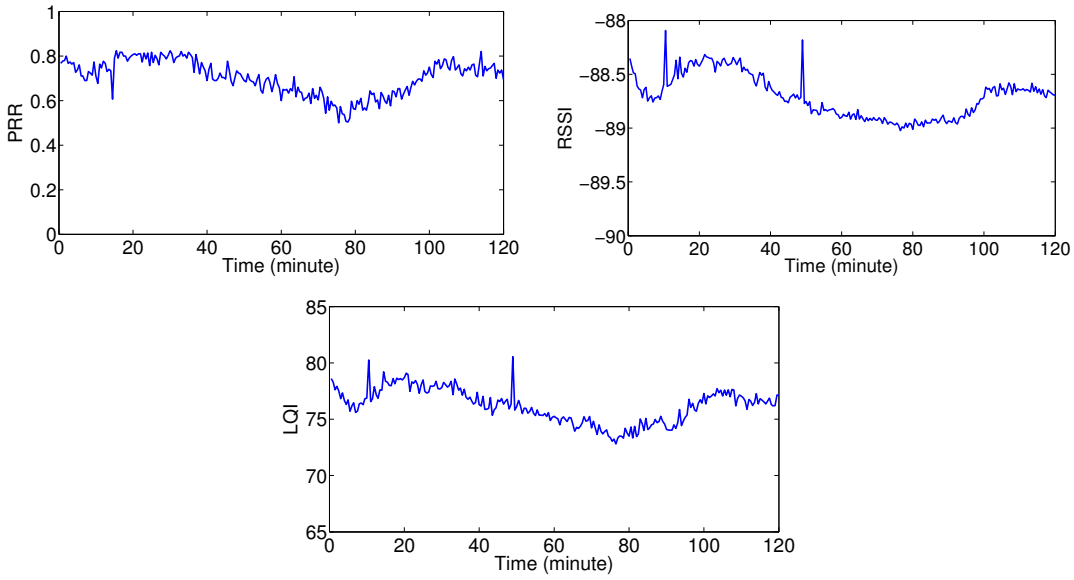


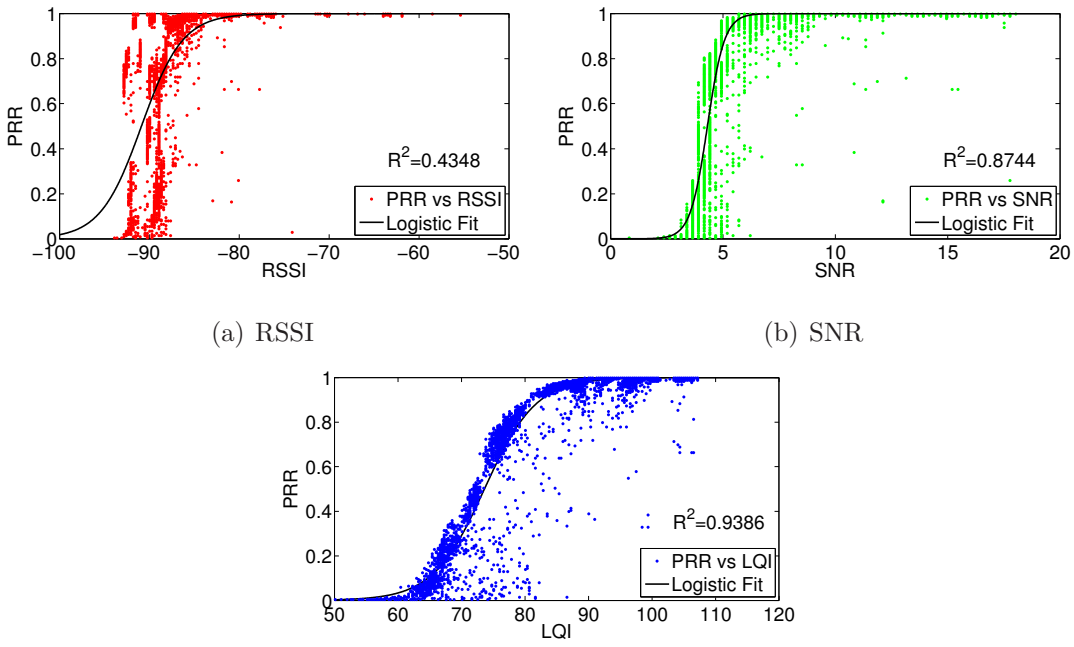
Figure 4.1: Packet Reception Ratio (PRR), Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) variation of an intermediate link over a period of 2 hours.

dure of model training. The modeling results are presented in the end.

4.1.1 Exploratory Data Analysis

PHY information is a direct measurement of the wireless channel quality when a packet is received, so we should expect some level of correlation between PHY information and link quality. To validate the correlation, we perform extensive packet trace collection in a local wireless sensor network testbed comprised of 54 TMote Sky motes as explained in Section 4.1.4. In short, the TMotes sends 30-byte long packets at 0.1 seconds interval, and we record the packet reception as well as the physical layer parameters of the received packets.

Fig. 4.1 shows the PRR variation of a intermediate link from the local testbed over 120 minutes, as well as the corresponding RSSI and LQI variations in that



(a) RSSI

(b) SNR

(c) LQI

Figure 4.2: Packet Reception Ratio (PRR) as a function of Received Signal Strength Indicator (RSSI), Signal to Noise Ratio (SNR) and Link Quality Indicator (LQI) for 160 hours of data for 72 links.

period. The similar variation patterns suggest that we can leverage the PHY information to predict PRR. Indeed, there are routing protocols that operate based on LQI [Mul, TC05]. However, due to the short temporal dynamics of wireless channel and differences in hardware calibration, it is hard to find well defined correlations between PHY information and PRR over different links and even different networks. For example, empirical studies [PSC05, SDT10] have found that although LQI is a better link quality indicator than RSSI, a single LQI reading is not sufficient for accurate link quality estimations due to the high level of variance [SL06, GBP10]. The difficulties of using PHY parameters as link quality metrics are summarized in [BKM12], and therefore widely used routing protocols like CTP [GFJ09] often utilize PRR based link estimators.

One question we would like to explore is to what extent we can correlate PHY parameters and PRR, such that we could use them as input in the prediction process of expected PRR. As mentioned in Section 2.3, several previous studies [LMH03,SKH06,SDT10] has pointed out that PRR can be approximated by PHY parameters with sigmoid curve, but the curves obtained in different links and/or networks have similar shape but shift different amount. To empirically verify the previous findings and to explore correlation between PRR and the PHY parameters, we aggregate all the packet traces from 68 links collected in the local testbed (data collection process explained in Section 4.1.4), and plot PRR with respect to the corresponding PHY parameters (RSSI/SNR/LQI) in Fig. 4.2. In this figure, the y-axis of each data point represents the PRR calculated with a window of 1 second, whereas the x-axis indicates the mean value of the corresponding PHY parameters from the packets received during the same 1 second window. To better visualize the correlation, we use logistic regression to fit the sigmoid curves to the PRR-PHY data points, and noted the goodness of fit [CB01] (R^2) in each plot.

A simple visual inspection of Fig. 4.2(b) and 4.2(c) shows that both LQI and SNR have a significant correlation with PRR. On the other hand, Fig. 4.2(a) shows that RSSI has values spread over a wider range of PRR values for RSSI values in the -94 and -85 dBm range. The solid line shows the logistic fit for each of the curves. As expected, both LQI and SNR present very high R^2 values of 0.9386 and 0.8744 respectively. RSSI has a much lower R^2 of 0.4348.

Based on these results, there are a couple of observations we can make. First, our results agree with previous findings in [LMH03] and [SKH06] for SNR. However, our results extend the findings to LQI, and also show it is the PHY parameter that has the best goodness of fit. Second, our results for RSSI differ from

those found in [SL06]. When collecting a larger number of experimental traces, we find that the LQI has a smaller variance than RSSI. We also noted that not *all* the links with -87 dBm RSSI values have PRR larger than 85%. In our data traces we get extreme cases of intermediate links with RSSI values larger than -87 dBm up to -74 dBm. Finally, based on our data analysis, it is clear that we should take advantage of PHY information to determine the expected packet reception. The following sections show how to use *both* PHY information and PRR to our advantage.

4.1.2 Problem Definition

The model we want to create takes RSSI, SNR or LQI and reception status from W packets as input to predict the reception probability of the next packet. In other words, the input to our model is a vector that is constructed from the the historical information of W packets. An input vector ($Input_i$) is expressed as follows:

$$Input_i = [PKT_{i-1}, PKT_{i-2}, \ldots, PKT_{i-W}]$$

and the output is the reception probability of the i_{th} packet:

$$P(Reception_i|Input_i).$$

The packet vector PKT_i is comprised of packet reception ratio and a subset of available physical layer information PHY_i corresponding to a packet. It is written as:

$$PKT_i = [PRR_i, PHY_i], \quad PHY_i \subset (RSSI, SNR, LQI)_i$$

All the values in a packet vector are discrete. PRR_i is the WMEWMA output and has a range between $[0, 1]$. The physical parameters (RSSI, SNR and LQI) have different ranges ($[-55, 45]$,$[0, 50]$ and $[40, 110]$ respectively), so we scale

them down linearly to the unit range $[0, 1]$, such that the physical parameter vector PHY_i is within the unit range. Note that the raw RSSI values from the CC2420 [CC2] radio chip is the actual RSSI value plus an offset of 45, therefore the RSSI range here is $[-55, 45]$, which corresponds to the actual RSSI range of $[-100, 0]$dBm. However, this range difference does not affect the input values as they were all scaled down to the unit range. with this notation, we can represent a lost packet as:

$$PKT_i = [PRR_i, 0]$$

where, $PHY_i = 0$ since there is no physical parameter available for lost packets.

As mentioned previously, PRR_i refers to the most recent WMEWMA output when the ith packet was received. WMEWMA proposed by [WTC03] is a widely used link estimation technique and is used in 4Bit. Essentially, the WMEWMA estimator calculates the most recent PRR with a small window size, and then smooths the new PRR value using an EWMA filer. It can be expressed as:

$$PRR_i = \alpha \times PRR_{i-1} + (1 - \alpha) \times PRR_{new}$$

where α is the smoothing factor of the EWMA filter, and the PRR_{new} is the current link PRR calculated based on packet reception of a certain window size, i.e., PRR_{new} = (number of received packets) $\div$ (window size). 4Bit uses a stable smoothing factor $\alpha = 0.9$, and the window size of PRR_{new} calculation for data packets is 5. To best approximate the actual output of 4Bit, we use the same parameters in the modeling process. In other words, the PRR_i in the model input is calculated by a WMEWMA estimator with $\alpha = 0.9$ and is updated every 5 packets.

In real world sensor network applications, the data traffic can be periodic (e.g. temperature monitoring) or aperiodic (e.g. event detection). We account for this behavior by using input vectors composed of data packets with fixed or random

inter-packet interval (I) times. With a fixed I, the input vector is composed of periodic packets separated by the same time interval. For random I, we use a Bernoulli process to select packets such that the time intervals between two consecutive packets follow a binomial distribution whose mean equals to I. These two data composition methods enable our link prediction scheme to deal with varying periodicity of data as seen in real applications. We train models using different average I values and data composition methods to test the prediction performance under changing periodicity.

4.1.3 Prediction Methods

To ensure the overhead of the prediction model will not interfere with the normal operations of the sensor nodes, the modeling method should satisfy the following requirements to be considered practical for sensor networks.

- – *Small Training Data:* The model should not need significant deployment efforts for gathering training data for extended periods of time. Otherwise, the overhead of gathering data to train the model alone might outweigh the benefits gained by using the model.
- – *Light Weight Online Prediction:* While training the model offline can be computationally costly, the implementation of the online link prediction scheme using the trained model should have low computational complexity and small memory requirements.

Based on these guidelines, we tried the following three modeling methods.

Naive Bayes classifier (NB) NB is a simple probabilistic classifier based on Bayesian theorem with the conditional independence assumptions: each

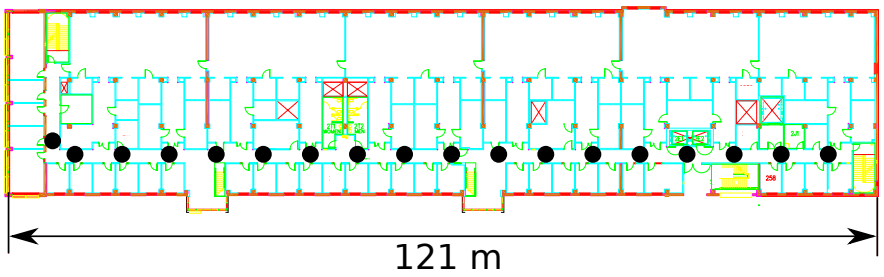


Figure 4.3: Local testbed with 54 Tmotes Sky motes placed along an corridor of a typical office building. Each dot indicate a group of 3 nodes.

feature in a given class is conditionally independent of every other feature. Although the independence assumption is quite strong and is often not applicable, NB works quite well in some complex real-world situations such as text classification [Mit97]. Due to its simplicity, we consider NB advantageous in terms of computation speed and use it as a baseline of our comparison.

Logistic Regression classifier (LR) LR is a generalized linear model that predicts a discrete outcome from a set of input variables [Bis06]. It is an extensively used method in machine learning, and is easy to implement in sensor nodes.

Artificial Neural Networks (ANN) ANN is a non-linear modeling technique used for finding complex patterns in the underlying data. For modeling, we used a standard two-layer feedforward network [HDB96] with one hidden layer of 5 neurons and a output layer of a single neuron. Both the hidden layer and the output layer use sigmoid as the transfer function. The small number of hidden units reduces the computational complexity for faster online prediction on a sensor node.

4.1.4 Data Collection

In order to train the model, we collected packet traces from two testbeds: a local wireless sensor network testbed and the Motelab [WSW05], a sensor network testbed composed of 180 Tmote Sky motes. The local testbed is an expansion of the sensor network testbed described in Section 3.2.2. It comprised of 54 Tmotes installed on the ceiling of a corridor in a typical office building. Fig. 4.3 shows the new placement of these nodes. The motes are divided into 18 groups (denoted by the black dots), and the distance the distance between each node group ranges from 6 to 7 meters except for the node group in the far left which sits around a corner at the end the corridor. The nodes are numbered from 0 to 53, starting from the first group on the left to the last group on the right.

We implemented a collection program to record the physical layer information of every received packet of a wireless link. During one data collection experiment, a sender node continuously transmits packets to a receiver node with 100 milliseconds inter-packet interval (Tx-power=0dB, channel 26). Upon packet reception, the receiver node records the sequence number, RSSI and LQI of the received packets. In addition, the receiver measures the noise floor level by sampling the environmental noise 15 times with 1 millisecond interval after every reception. Following examples set by prior work [SKH06, RMR06, KGD07], we average the measurements to calculate the accurate noise floor level, which enables us to compute the SNR. We ran the data collection program on 68 sender-receiver pairs in different time slots to avoid inter-node interference. In total, we recorded information for approximately 5.4 million packets over 160 hours of data collection. Each of the 68 packet traces contains records for 80,000 packets. Among the 68 links, there are 12 low quality links ($PRR < 10\%$), 14 intermediate links ($10\% < PRR < 90\%$), and 42 high quality links ($PRR > 90\%$).

Parameters	Values
Input Feature	$PRR + \{RSSI, SNR, LQI\}$
Number of Links (L)	$20, 7, 5, 3, 2, 1$
Number of Packets (P)	$36000, 10000, 5000, 1000, 500$
Window Size (W)	$1, 2, 3, 4, 5, 10$
Packet Interval (I)	$0.1, 0.2, 0.5, 1, 10, 60$ (seconds)

Table 4.1: Modeling Parameters

As for the Motelab testbed, we collected packet traces from 10 links for one hour. Different from the local testbed data, only RSSI and LQI were collected, and the inter-packet interval is set to 62.5 milliseconds (64 packets per second). All the links are of intermediate quality and exhibit large temporal quality variations during the course of one hour data collection. The average PRR of these 10 links ranges from 0.13 to 0.84. As we show in Section 4.1.7, a dataset collected from 10 links for one hour is *more than enough* to train prediction models with satisfying accuracy.

4.1.5 Modeling Parameters

We varied several parameters in the training process to explore the optimal training parameter collection. Table 4.1 shows the different parameter combinations with regards to the input data during the training of our models.

We experimented with different input feature vectors as well as various window size (W) and inter-packet interval (I) values during the training process. W denotes the amount the historical packets needed to make a prediction, whereas I decides the periodicity of the prediction. As discussed in Section 4.1.7, Figure 4.8 and 4.9, the choice of them can greatly affect the prediction quality and

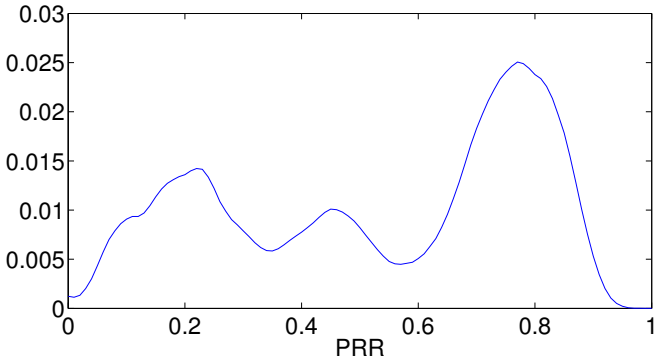


Figure 4.4: PRR distribution of the intermediate quality links.

the feasibility of implementing the model on resource constrained sensor nodes. For the input feature, we tried combination of PRR with all the physical information ($PKT = [PRR, RSSI, SNR, LQI]$) as well as PRR with each one of the parameters ($PKT = [PRR, RSSI/SNR/LQI]$).

The input vectors are composed of data combined from a number of links (L) with a number of packets per link (P). Ideally, the training data should cover links with different qualities such that the resulting model can cope with a large spectrum of link quality variation. To ensure maximum link diversity in terms of PRR, we maximize the difference between the L links in the reception rate such that the average reception rates are evenly distributed from 0% to 100%. Hence, a larger L implies better link diversity in the training set. For example, when $L = 5$, the PRR of these 5 links are 0.11, 0.23, 0.44, 0.76 and 0.93, whereas when $L = 7$, two links with PRR of 0.35 and 0.61 are added to increase the variance of PRR.

In addition to long term link quality diversity, it is also important to include links with diverse link quality variation patterns as the goal of the prediction model is to predict the next packet reception. Figure 4.4 shows the distribution function of PRR computed for the 14 intermediate quality links with a time

window of 10 seconds. As seen in this figure, PRR values of these 14 links distribute over the full PRR range between 0 to 1, with several spikes close to PRR values of 0.2, 0.4 and 0.8. This uneven distribution of PRR indicates a rich diversity in terms of link quality variation, which is the target of the prediction model.

The number of packets used from each link (P) is another modeling parameter when training our models. From a practical point of view, for constructing the model using the proposed approach in a different environment, the users need to collect certain minimum amount (in terms of link diversity and length) of traces to replicate conditions from the target environment. We parameterize these constraints (L and P) and explore the associated trade-off in the training process. To find a balance between the training data size and prediction accuracy, we vary the training dataset and compare the accuracy of the resulting model. We tried several combinations of link selection, ranging from using all the available links to selecting only one. The evaluation results presented in Section 4.1.7, Figure 4.7 and 4.6 indicate that a dataset collected from only a few links (5-7 links) for several minutes (2-10 minutes) is enough to train the model with satisfactory accuracy.

4.1.6 Training Procedure

Once the parameters are set, we use the following steps to train the models.

1. **Packet Selection:** Select L links from the collected data. From each link, select P packets according to the I. As described in Section 4.1.2, the time intervals between packets can be either fixed or random based on the periodicity of I.

2. **PRR Computation:** Compute the PRR by applying the WMEWMA on the selected packets. To mimic the ETX calculation of 4Bit [FGJ07], we set the window size of the WMEWMA filter to 5 packets and α to 0.9.

3. **Input Vector Construction:** Based on the input features, we take the PHY parameters of W packets and combine them with the most recent PRR value computed in the previous step to construct an input vector. We repeat this step until all selected packets are used. The target vector is also constructed during the process by checking the reception status of the next packet of each input vector: we mark the target (desired output) of an input vector as 1 if the next packet is received, and 0 otherwise.

4. **Model Training:** Once the input vectors and the target are constructed, we randomly select 60% of the total inputs as the training data and use the remaining 40% as the testing data. In the training step, the models using NB learn the conditional probabilistic distribution of the packet reception based on the input from the training data [Mit97], whereas LR models learn the regression coefficients from the training data using maximum likelihood estimation [Bis06]. Both NB and LR models finish training when all the training data is considered. Based on our experience, training NB and LR models only takes less than a minute with more than one million samples using a desktop computer with 2.4GHz Intel processor. In the ANN case, the model continuously updating the weights and biases values of the network to optimize network performance, which is defined as the mean square error (MSE) between the network outputs and the target outputs. The standard backpropagation algorithm [HDB96] is used to update the weights and biases, and the stopping criteria is the error gradient less than 1e-5, i.e., the training will stop when the gradient of MSE is less than 10^{-5}.

5. **Testing:** After the models are trained, we then apply the trained models to the testing data. The prediction outputs are compared with the corresponding target values to assess the performance as described in the following sections. Since NB and LR are classifiers, the outputs of NB and LR based models are binary values 0 and 1, representing the future packet reception probability of 0% and 100%. On the other hand, ANN based models output real values between 0 and 1, representing the actual reception probability between 0% and 100%. Please note that when determining the prediction accuracy in Section 4.1.8, the results of ANN are also converted into binary values by using a threshold of 0.5, i.e., if the predicted reception probability is greater than or equals to 0.5, the prediction is considered as the packet is received, otherwise it is considered as the packet is lost.

We use these five steps to create, train and test all three models with the same parameter set in MATLAB. To avoid excessive training, we first run the procedure with fixed input features to narrow down the reasonable input data size (L and P). We then fix the input data size and run the training procedure for different combinations of input features, W and I. Due to the simplicity of the NB and LR models, training for each model needs less than one minute on a regular PC, whereas the training of the more complex ANN based models can take up to several hours. In the end, we repeat the procedure for more than 50 times using data from the local testbed and the Motelab testbed, resulting in more than 150 models trained with different parameter sets for each testbed. Although the number of models is large, their performance trend is relatively clear as discussed in the following sections.

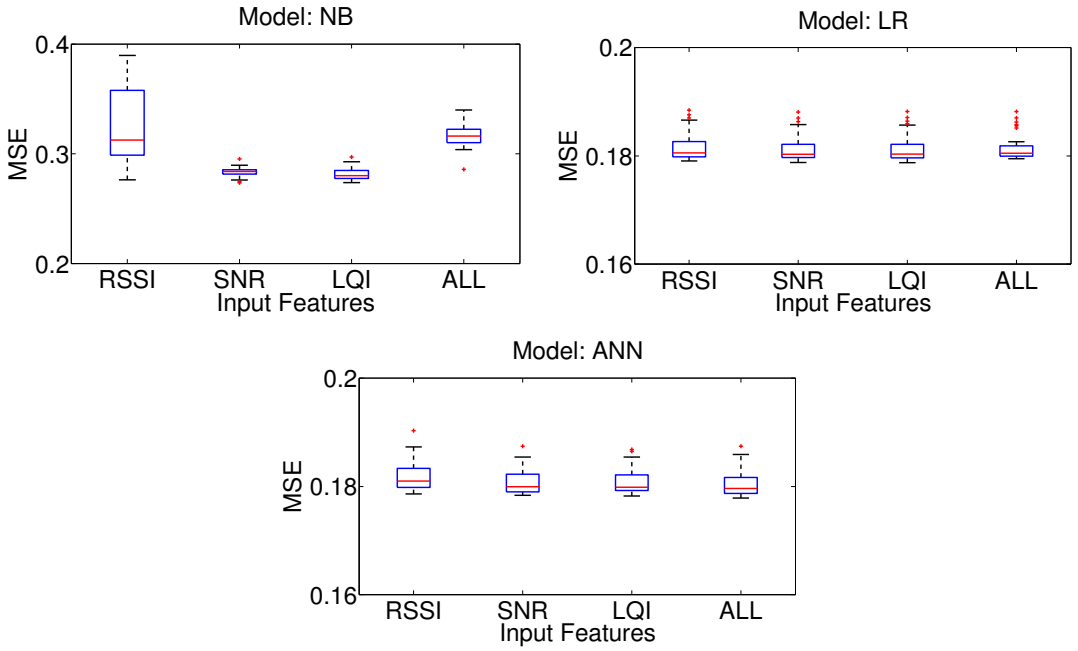


Figure 4.5: Prediction errors for different input features. $L = 20$, $P = 36000$. Note the error ranges are different.

4.1.7 Modeling Results

In this section we discuss the performance of our three models when evaluated on the testing data. We plot the variation in the mean square error (MSE) as a function of input features, L, P, W and I. Based on the results, we propose the data requirement of training (L and P) as well as the model selection guideline (W, I and the input features) for the experimental evaluation in Section 4.3. Ideally, we would like to have the error as low as possible when L, P and W are small and I is large. Note that although we trained models for the local testbed and the Motelab testbed respectively, their performance results are very similar and the overall trend is the same. As such, we only present the local testbed results for brevity.

4.1.7.1 Input Features

We first compare the prediction error of the three models with respect to four different input features, namely, PRR with RSSI, SNR and LQI only, and PRR with all three PHY parameters. For each input feature, we use all the available training data and evaluate the prediction performance with all the intended W and I values listed in Table 4.1. That is, the training data size is fixed to $L = 20$ links, and each link contains $P = 36000$ packets. Then, we train and test the three models with various W and I using the training procedure described in section 4.1.6. The modeling error (in terms of MSE) of the three models is aggregated and plotted in Fig. 4.5 respectively.

Fig. 4.5 presents the aggregate prediction error of all the trained models in three box plots. Each box plot corresponds to a modeling method as indicated in the plot title, and each box in a plot shows the median value of the probability distribution of the aggregate MSE, as well as the 5%, 25%, 75% and 95% percentiles of the models trained with input feature labeled in the X axis. The red dots outside some of the boxes are outliers, i.e., the data points that are outside the range between $Q_1 - 1.5(Q_3 - Q_1)$ and $Q3 + 1.5(Q3 - Q1)$, where Q_1 and Q_3 refer to the first quartile (25%) and the third quartile (75%) respectively. Fig. 4.6, 4.7 and 4.8 are plotted in the similar manner.

From Fig. 4.5, it is obvious that for LR and ANN model, the prediction performance is stable regardless of the PHY parameters used in the input feature. For NB model, using the PRR and RSSI as the input feature yields the highest error whereas using PRR and SNR/LQI results in lower errors. This result confirms the observation we made in section 4.1.1 that the correlation between PRR and RSSI is weaker than PRR with SNR or LQI. Note that even in the case of PRR and all the PHY parameters, the prediction error is higher than just using PRR

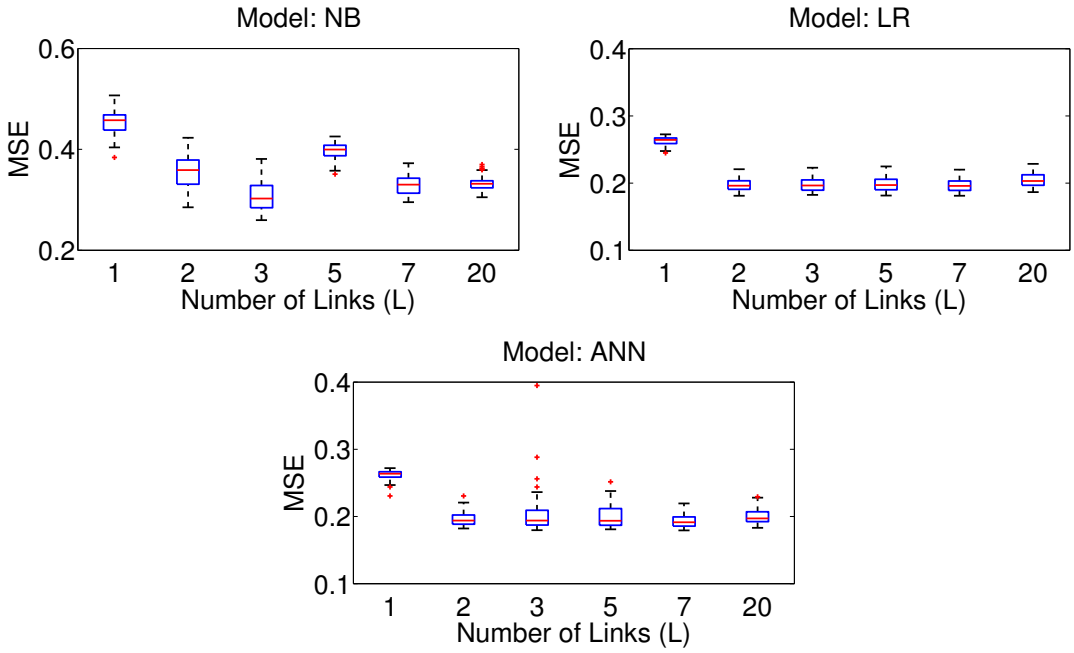


Figure 4.6: Prediction errors for $L = \{1, 2, 3, 5, 7, 20\}$, $P = 36000$.

and SNR/LQI as input features, suggesting that the inclusion of RSSI skews the closer correlation found between the PRR and SNR/LQI.

In short, our results show that the choice of PHY parameters does not affect the prediction result much except for the NB models. In the remainder of this section we show the modeling results of using $[PRR, LQI]$ as the input feature since the LQI value can be easily obtained in the CC2420 platform.

4.1.7.2 Number of Links (L)

The next step is to find the training data size requirement for the modeling. Ideally, we want to find the set of P and L that is large enough to train the model, but is also as small as possible to minimize the deployment impacts. Fig. 4.6 shows the variation in the prediction error of the three models with a fixed $P = 36000$ but different L. Across all three models, we see a common

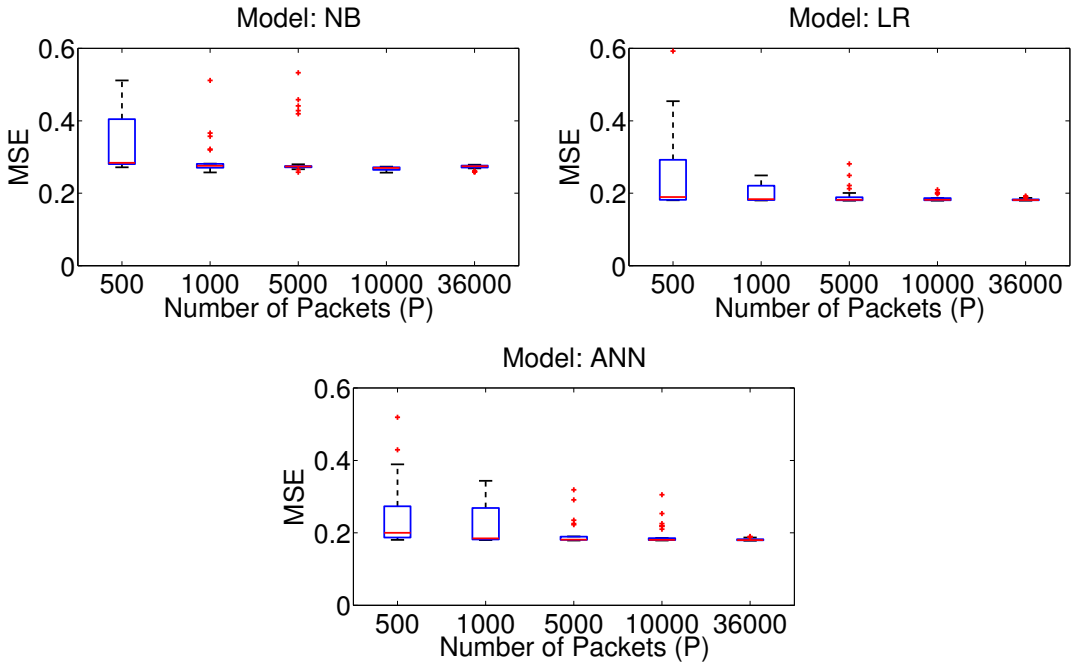


Figure 4.7: Prediction errors for $L = 5$, $P = \{500, 1000, 5000, 10000, 36000\}$.

trend of decreasing prediction error as L increases. For the NB models, the performance when using fewer links is much worse than for $L = 20$. However, for the LR and ANN models, the prediction error is small (≈ 0.2) if $L \geq 2$. This shows that the performance for the LR and ANN models trained with only two or more links is comparable to the ones trained with many more links. Hence, for modeling proposes, only a few links with intermediate PRR are required to model the variations in a large variety of links. Note that the underlying assumption behind this conclusion is that these links should cover a wide range of PRR to ensure link diversity as discussed in Section 4.1.5.

4.1.7.3 Number of Packets per Link (P)

Next, we plot the variation in the prediction error of the models as a function of P. From Fig. 4.7, we observe that for high values of P (36000 for example), the

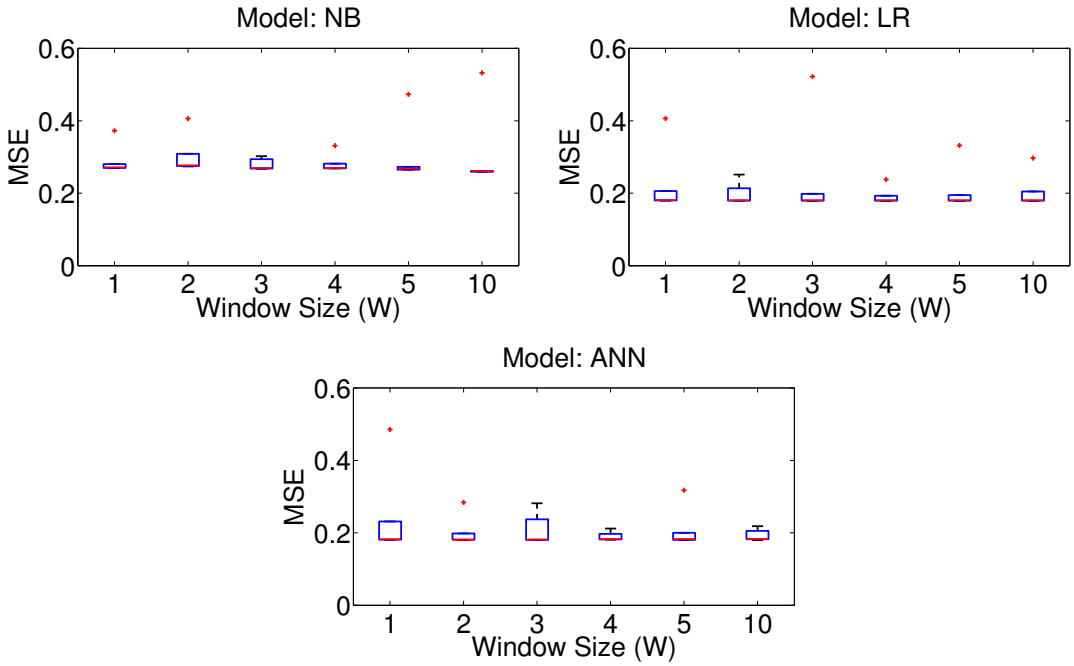


Figure 4.8: Prediction errors for $W = \{1, 2, 3, 4, 5, 10\}$. Trained with $L = 5, P = 1000$.

prediction error of all three models is almost the same with P values as low as 5000. In fact, the prediction error increases significantly only after P is dropped below 1000. This shows that we only need around 1000 packets per link to train the prediction models.

4.1.7.4 Window Size (W)

Now that the desired training data size is clear, we then explore the parameters that directly define the model inputs, namely, W and I. W corresponds to the amount of historical information required by the model to predict the reception probability of the next packet. Intuitively, large W means more information will be made available to the model, so it should improve the prediction performance at the cost of more buffering and processing needs. However, Fig. 4.8 shows that

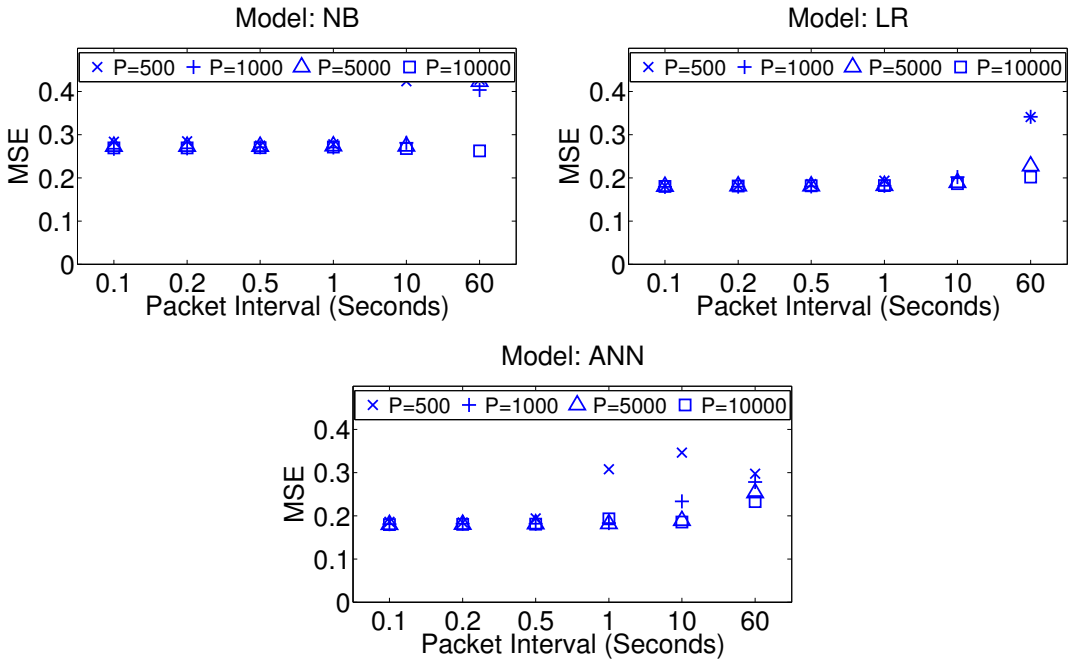


Figure 4.9: Prediction errors of $I = \{0.1, 0.2, 0.5, 1, 10, 60\}$ seconds. Trained with $L = 5$, $W = 1$.

the prediction error does not decrease very rapidly as we increase W from 1 to 10, which implies that only the most recent packet is important for the prediction. Therefore, the LR and ANN models should work reasonably well with a small window size such as $W = 1$.

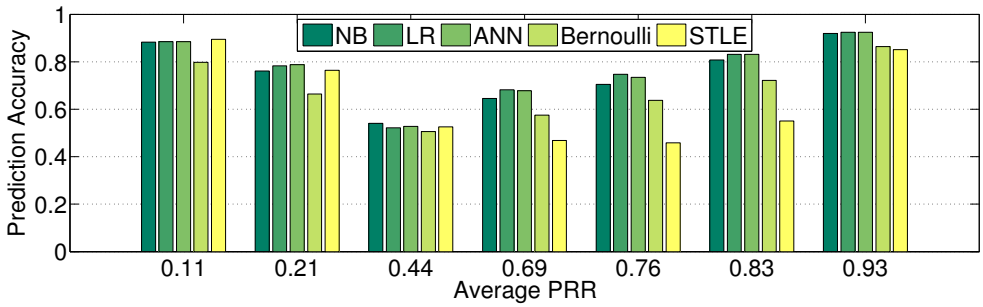
4.1.7.5 Inter-Packet Interval (I)

Intuitively, the longer the I, the older is the packet reception information used by the model. Therefore, the prediction error should be worse because intermediate links may experience significant temporal dynamics. Fig. 4.9 shows the prediction errors as the I (aperiodic) increases from 100 milliseconds to 1 minute, with the MSE of models trained with different P plotted together. Note that Fig. 4.9 presents the results of the individual models directly due to the small number

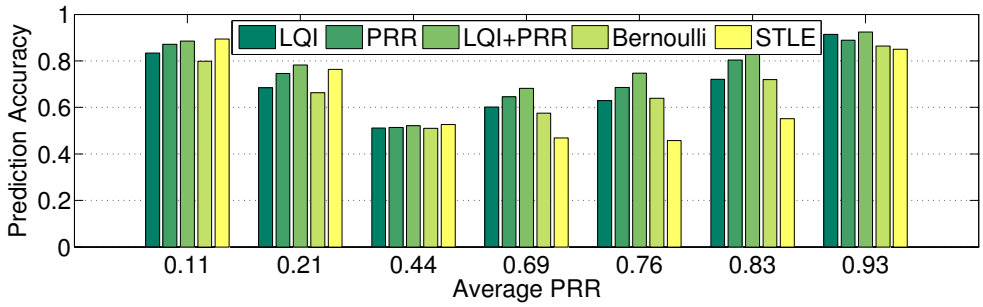
of unique parameter sets (5 only), whereas Fig. 4.5, 4.7, 4.6 and 4.8 present the statistics of the modeling results with boxplots due to the large number of parameter sets.

Fig. 4.9 shows that prediction error does not degrade much until $I = 1$ minute, implying that our models are not sensitive to $I < 1$ minute when $L = 5, P = 1000$. Moreover, the MSE of models with different P shows that using higher P can reduce the prediction error. In our case, models trained with $P = 5000$ is enough to provide similar accuracy compared to models with $P = 36000$. Periodic I gives similar results and are omitted here. Given the 0.1 seconds packet interval in the training data, $P = 1000$ corresponds to around 1.7 minutes of data collection, whereas packet traces with $P = 5000$ require about 8.3 minutes of collection. Therefore, we consider data collection between 2 to 10 minutes is sufficient for training the models.

Note that although shorter Is give better results, in practice a model trained with short I may not perform well when there is a mismatch between the I and real packet sending intervals. For example, a model trained with 1 second I assumes that the average data rate is 1 packet/second, and predicts the success probability of the next packet whenever a new packet is received. However, if the actual packet interval is 10 seconds, the prediction based on the 1 second interval assumption will expire for 9 seconds, and may not represent the success probability of the next packet. Therefore, a model performs the best when the actual packet interval matches the I value. We explore the same I mismatch issue in Section 4.3.3, and leave the full evaluation of the solution to Chapter 5.



(a) Input feature = $\{PRR, LQI\}$, different models



(b) LR model with different input features

Figure 4.10: Prediction accuracy for links of varying PRR of models trained with $L = 5$, $P = 5000$, $W = 1$ and $I = 10$ seconds.

4.1.8 Performance Gain with Prediction

The above evaluation is based on the MSE of the trained models applied to the testing dataset. To better evaluate the accuracy of the prediction model on individual links, we also compare the prediction performance of the models on a per-link basis with STLE [ALB09] and an informed Bernoulli process. STLE is a short temporal link estimator that based on the heuristic that three consecutive packet receptions signify high link quality in near future and one packet loss signify low link quality periods. The Bernoulli process is an informed estimator based on the full knowledge of the link PRR ahead of time. We set the success probability of the Bernoulli process to be equal to the PRR, so the 1/0 trail generated by the Bernoulli process can be used to predict packet reception based on the overall link quality. We apply our models, as well as STLE and the Bernoulli process on several empirical intermediate quality links and compare the prediction accuracy of different modeling methods and varying input features.

The prediction accuracy is computed as the ratio of the correctly predicted packets to the total number of packets of the link. Fig. 4.10(a) shows the performance of the NB, LR, ANN models, STLE and the Bernoulli process for wireless links of varying PRR. We see that for $I = 10$ seconds our prediction models consistently outperform the Bernoulli process. This result illustrates that our modeling approach can better adapt to link quality variations than the Bernoulli process. It also shows that LR based model can provide very good prediction accuracy at low computational costs. Fig. 4.10(b) shows the prediction accuracy of LR models with different input features. In almost all cases, we see that the prediction models perform better than the Bernoulli process. Moreover, we see that the best prediction result is achieved by using *both* PRR and LQI in our input feature, followed by PRR and LQI only cases. We see that PRR does a

better job than LQI, except for very good links on which the LQI based predictor performs better. The intuition behind this result is that although PRR is a good estimate for links with stable quality, it is too stable to account for the rapid quality variations. Additionally, LQI changes too drastically on wireless channel variations, making it unstable for long term estimations (unless the link is consistently good). By using the combination of LQI and PRR as input, the LR model can supplement the PRR with LQI, and therefore performs better in estimating link quality for intermediate links.

Note that in some cases, the accuracy of Bernoulli process is even worse than the average PRR of the link. In essence, the packet losses/successes are often correlated [CWK] and therefore the underlying packet reception distribution is not a Bernoulli process. For example, suppose a 5-packet trace of the form 11110 is collected with $PRR = 80\%$. A Bernoulli process with $p = 0.8$ will predict this sequence with prediction accuracy at least of 80% in only a 6 cases, namely 11110, 11111, 01110, 10110, 11010 and 11100. In all other 26 cases, the prediction accuracy will be $< 60\%$, so the expected prediction accuracy (weighed over the likelihood of each sequence) will be less than the PRR value. This simple example shows the difficulties a link estimator faces: even if it captures the PRR correctly, the correct prediction is still not guaranteed.

Fig. 4.10 also includes the predication accuracy of STLE. In both Fig. 4.10(a) and 4.10(b), we see that STLE performs on par with the prediction models for links with low and high quality ($PRR < 0.5$ or $PRR > 0.9$), but for links with PRR between 0.5 and 0.9 range, STLE is even worse than the Bernoulli trail. From these results, one can infer that for the low and high quality links, the heuristic of STLE is generally effective as these links are stable: most packets are received in high quality links, therefore STLE can correctly predict next packet

reception, whereas in the low quality link cases, the majority of STLE output will be packet loss since most of the packets are lost. However, due to the rapid and frequent quality variations of the links with PRR in 0.5 and 0.9 range, this heuristic is not longer valid and often misled by the short consecutive packet loss/receptions. In this case, even an informed Bernoulli process performs better as it can show the average quality of the links. These results reflect the limitations of the underlying heuristic of STLE: although it works well for some links, it can not adapt to the varying characteristics of all the links in the network. On the other hand, the data-driven approach enables the prediction model specifically trained for the deployment network, therefore provide better link quality prediction compared with STLE.

4.1.9 Summary of Modeling Results

These results are quite significant. They essentially show that a user that wants to train a model just needs to gather *several minutes (2-10 minutes) worth of data from only a few links (5-7 links)* to reach an MSE that is similar to models trained with much higher number of links, and with significantly longer packet traces. Moreover, the trained model only needs *one historical packet* for the prediction. We evaluate the statement experimentally in Section 4.3.

4.2 Estimator Design

In this section we present the design of the 4C link estimator. We first show how we integrate the prediction based link estimation with the existing link estimator, then discuss the main challenges and details of the model implementation.

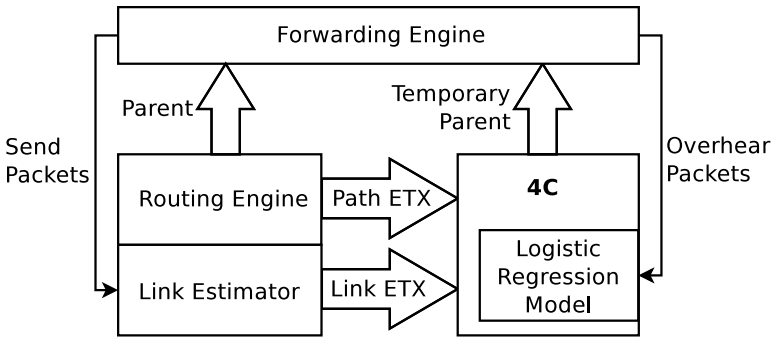


Figure 4.11: Overall Design of 4C.

4.2.1 Overview

We propose a receiver-initiated link estimator called 4C to implement the prediction model for link quality estimation. 4C works in parallel with an existing link estimator, using information from overheard packets to predict the link quality of neighboring nodes. If 4C finds that it can provide a better path cost than the parent node of a sender, it will send beacon packets to the sender to announce itself a temporary parent as detailed in Section 4.2.3. After reception of this beacon, the sender will switch its next hop from the parent node designated by the routing protocol to the temporary parent. Thereafter, the sender will send future packets to the temporary parent until the number of consecutive lost packets exceeds a threshold, or the temporary parent denounce itself. In this case, the sender node will switch back to the old parent node.

4C shares a similar receiver-initiated approach with the Short Term Link Estimator (STLE) [ALB09]. In essence, both STLE and 4C attempt to reduce the total number of transmissions per packet by using temporary routes on the basis of a stable network topology established by the routing protocol such as CTP. However, there are two main differences between STLE and 4C. First, they apply to different traffic patterns. 4C is focused on providing a more informed

link estimation whereas the main goal of STLE is to detect short term reliability of intermediate links using a simple heuristic procedure: 3 consecutive packet receptions means the link is usable, and a loss means the links is no longer usable. On the other hand, 4C can fit to a wide range of traffic patterns by utilizing appropriate models. For example, experimental results in Section 4.3 show that with the logistic regression model, 4C works the best when tested using similar traffic patterns used for training. On the other hand, STLE is most suited for bursty link discovery, therefore its heuristic-based approach applies specifically to only a bursty traffic pattern. Second, STLE is mainly a qualitative measurement whereas 4C is quantitative in nature. STLE can identify whether an intermediate link reliable or not, but it can not specify how reliable is the intermediate link. In contrast, 4C is designed to give a quantitative estimation of the link quality.

4.2.2 4C Design

Fig. 4.11 presents the overall design of the 4C. In general, 4C couples with CTP and operates in the intercept (overhearing) nodes and interacts with all three core components in CTP: the forwarding engine, which handles data packet sending and forwarding, the routing engine, which is in charge of choosing the next hop (parent) based on the link estimation as well as processing network-level information such as congestion detection, and the link estimator that is responsible for estimating the quality of the links to single-hop neighbors.

4C works in two stages: link prediction and path evaluation. In link prediction, 4C uses the data packet overheard by the forwarding engine and ETX from the link estimator to estimate the link quality of neighboring nodes. When the forwarding engine overhears a packet from a node, it passes the packet to

4C. 4C records the packet information, i.e., sequence number, packet sender and PHY parameters in its neighbor table. Then, 4C queries the link estimator for the link ETX of the sender, and takes the reciprocal of the returned ETX to get the estimated PRR. Finally, the underlying prediction model takes the estimated PRR and the PHY parameters as the input and outputs a reception probability for the next packet.

In the path evaluation stage, 4C uses the predicted reception probability to evaluate the path cost for the sender assuming the intercept node is the sender's parent. This is done by adding the path cost of the intercept node itself with the reciprocal of the predicted reception probability. If the calculated path cost is smaller than the actual path cost of the sender minus the overhead of notifying the sender, 4C sends notification packets to the sender, announcing itself to be the temporary parent. The overhead of sending the notifications is calculated based on the link quality between temporary parent and the sender, therefore the additional overhead introduced by the announcement packets will not offset the potential gain of using the temporary parent. The temporary parent announcement process is discussed in Section 4.2.3.

On the sender side, after receiving the announcement beacon, 4C notifies the forwarding engine about the temporary parent. The sender then starts forwarding its traffic to the temporary parent. The temporary parent continues to be the sender's next hop until one of the following three events happens: temporary parent denounces its parent status, the routing engine assigns a new parent, or the number of consecutive packet losses exceeds a threshold. If any of the above three cases occurs, the sender will switch back to the parent designated by the routing engine.

An important design decision is which prediction method 4C should use. To

implement a prediction model on conventional sensor network hardware, we need to select a model that is the most suitable for online link prediction. While the NB based model is the fastest, it has the worst performance out of the three approaches. The LR and ANN based models are even in performance, but the ANN model has computational complexity several orders of magnitude higher than the LR model. Hence, we decide to favor the LR model for implementation on sensor nodes.

4.2.3 Temporary Parent Announcement

A question we need to explore is when should a node announce to be a temporary parent such that the overhead of announcement beacons will not offset the cost gain. To investigate the problem, let's assume the following scenario.

Since a routing gradient has already been established by CTP, each node should have an associated path cost, which denotes the number of transmissions needed to deliver one packet to the root node. We define the path cost of a sender node S as C_S, and the path cost of the sender's parent, P, as C_P. Similarly, we define the cost of sending a packet on the link $S \rightarrow P$ as $C_{S \rightarrow P}$.

Let's assume the packets sent by S are overheard by node T. T decides to be the temporary parent of S, so it needs to send beacons to notify S. In order to guarantee the reception of the notification, T needs to send $C_{T \rightarrow S}$ beacons to the sender on average. Moreover, another $C_{T \rightarrow S}$ beacons are needed when T decides to renounce its temporary parent status. Together, we note the cost of sending notification beacons as $C_{beacon} = 2 \times C_{T \rightarrow S}$.

Suppose the temporary parent T forwards n data packets for S during the

process. The potential cost gain will be

$$GAIN = n \times ((C_P + C_{S \to P}) - (C_T + C_{S \to T})) - C_{beacon},$$

where C_T is path cost the node T. The gain should be greater than 0, so we have

$$n \times ((C_P + C_{S \to P}) - (C_T + C_{S \to T})) - C_{beacon} > 0,$$

which transforms to

$$C_P + C_{S \to P} > C_T + C_{S \to T} + \frac{2}{n} \times C_{T \to S}. \tag{4.1}$$

Formula (4.1) can be viewed as the criteria for intercept nodes to announce the status of temporary parents. If it is satisfied, an announcement of temporary parent will be beneficial even counting the beacon overhead. An important parameter here is n, the number of packets that will be forwarded by the temporary parent. In our implementation, we take a conservative stance and set $n = 1$. In this case, as long as the sender sends more than one packet to the temporary parent, the overall cost will be further reduced. In addition, Formula (4.1) can be easily calculated on the intercept node with the path cost information from the CTP routing engine and 4Bit: $C_P + C_{S \to P}$ is the path cost of the sender node, which is available in the CTP packet header. C_T is maintained by CTP, $C_{T \to S}$ is maintained by the link estimator, and $C_{S \to T}$ is the link quality prediction that is constantly measured by 4C. Therefore, based on the calculation of Formula (4.1), the intercept node can decide when to announce and denounce temporary parent to the sender node.

4.2.4 Implementation Details

4.2.4.1 Prediction Speed

A critical issue is the computation speed of the prediction module. A logistic regression classifier, if implemented naively, requires a vector multiplication and sigmoid function to compute the predicted reception probability for the next packet. In the resource constrained sensor nodes, it is not feasible to do this naively in terms of computational cost. Furthermore, 4C is a part of the network stack, which normally demands fast response to network events. As such, prediction speed is one of the main concerns of the model implementation. 4C uses PLAN, a pairwise linear approximation proposed by H. Amin *et al.* [ACH97] to implement the sigmoid function. PLAN approximates the sigmoid function with 5 line segments and requires only one bit shift and addition operation to compute a sigmoid. In the 4C implementation, the input of the model has the fixed size as the window size W is set to 1, therefore the computation time required by the prediction model is also a constant. In addition, optimization are made to avoid the use of floating point operations. Our implementation running on a Tmote Sky mote requires 0.5 ± 0.004 milliseconds to compute a prediction, well within the time constraints of the networking processing stack in TinyOS. As a comparison, the time needed to send a 30-byte packet and receive its acknowledgement frame is around 5 milliseconds, 10 times longer than the computation time.

4.2.4.2 Extendibility

In addition to speed, another issue is the extendibility of the 4C, in the sense that it should be able to adopt more sophisticated prediction models without major modifications. By design, the prediction model in 4C is implemented with

a generic interface, so that the LR model can be replaced by other models as long as the input is the same. The LR prediction module accepts the coefficients from a LR model as its parameter, so it can be easily extended to use different LR models by changing the coefficients. In our experiments, we used several LR models trained with different parameters as discussed in Section 4.3.

4.2.4.3 Stability

There are also stability issues to consider. A practical problem occurs when CTP selects a new parent due to link quality variations of the neighboring nodes, the path cost change need some time to propagate. In this routing information propagation stage, 4C should not announce any temporary parent because the routing gradient of the network will likely be changed. To avoid this situation, we added some hysteresis to the process. We use a counter to suppress temporary parent announcements: if a parent change from the routing engine is detected, it will stop forwarding packets to the temporary parent immediately, and set the suppression counter to a preset value H. The counter value will decrease by 1 after each packet is sent to the new parent until it reaches 0. While the counter is not 0, 4C will not send any temporary parent announcement. In other words, 4C will not operate after a parent change until H packets are sent to the new parent. We set the H value to 3 in the evaluation.

4.2.4.4 Link Failure

A subtle problem occurs when the link quality between the sender node and the temporary parent suddenly drops. In this case, the temporary parent, even if it realizes the quality drop, can not notify the sender as the notification packet may get lost, and therefore the sender will try to retransmit the packets until the

normal route update mechanism of CTP kicks in and changes the parent node. To avoid this excessive retransmission to a temporary parent, we set an additional threshold to the maximum number of packet losses when forwarding packets to a temporary parent. This was implemented to avoid a broken link situation. If the link from a sender to its temporary parent is broken, the temporary parent will not be able to denounce itself. In this case, if the forwarding engine is using a temporary parent, it will switch back to the old parent after 5 packets are lost.

4.2.4.5 Link Asymmetry

Another issue related with link failure is how to handle asymmetric links, i.e., the link quality from the sender to the overhearing node is high, but the reverse quality is low. This will cause the loss of notification packets, and consequently, disagreement between the sender node and the assumed temporary node.

In our design, the overhearing node will send the temporary parent notification packets only once and does not require any acknowledge. In the sender side, if a the notification packet is lost, the sender will just continue to send packets to the original parent. This is an implicit reverse link quality check to avoid link asymmetry cases. The loss of the notification packet is an indication of low reverse link quality, so the sender should not switch to the temporary parent even if the forward link quality is high.

In the temporary parent side, the overhearing node will consider itself as the temporary parent of the sender after the notification packet is sent. If the notification is indeed lost and the sender continues to send packets to the original parent, the assumed temporary parent node should overhear the packets from the sender, and consequently discover the notification lost due to the asymmetric link. This is treated as a sender parent change event (addressed in Section 4.2.4.3),

MCU	Radio	Current Consumption (mA)	Operation Time
Idle	Off	< 1	0.5 ± 0.004 ms
Busy	Off	2	Per Prediction
Idle	RX	20 ± 1	5.25 ± 1.125 ms
Idle	TX	19 ± 1	Per 30-byte Packet

Table 4.2: Current draw of TMote Sky under various operation conditions.

and the temporary parent (now just an overhearing node) will suppress the future temporary parent notifications with a suppression counter H. This is to avoid excessive notifications over an asymmetric link and to improve routing stability.

4.2.4.6 Memory and Computation Overhead

The memory overhead of 4C is mainly due to the implementation of the receiver-initiated approach, i.e., the temporary parent announcement mechanism discussed in Section 4.2.3. The prediction model, on the other hand, only occupies minimal amount of memory as it only needs to store the coefficients of the LR model. In our current implementation, the ROM size increased by 4124 bytes (14.6%, from 28416 to 32540 bytes) with the addition of 4C, whereas the RAM requirement increased by 698 bytes (14.8%, from 4019 to 4717 bytes). Given the 48 kB flash and 10 kB of RAM in the TMote Sky mote, the added memory footprint should not be a big concern.

In terms of computation overhead, the extra energy consumption of 4C is mainly due to the prediction model and the temporary parent announcement. As discussed in Section 4.2.3, a node sends the temporary parent announcement only when the predicted cost gain is greater than the overhead of sending the notification packets. Therefore, we only address the additional energy consumption

of running the prediction model.

To analyze the energy consumption overhead introduced by the prediction, we measured the current draw of the TMote Sky when the prediction is running as well as when the radio is in use. Our measurements are consistent with the TMote Sky datasheet [Mot]. Table 4.2 lists the current consumption measurements as well as the time needed for the prediction and packet transmission operations. As shown in Table 4.2, the current consumption of the MCU running the prediction model at the full speed is 2 mA, whereas the radio typically consumes 20 mA when receiving and 19 mA when sending packets. Since transmitting a 30-byte packet and receiving its acknowledgment require about 5.25 milliseconds, the energy spent on transmitting a single packet would be able to support about 50 milliseconds of MCU computation time. Given the 0.5 milliseconds execution time of the prediction model, we see that the energy overhead of 4C's prediction model is only 1% of the energy of sending a single packet. Combined with the conservative temporary announcement mechanism, having 4C is beneficial as long as the prediction model can save one packet every 100 predictions computed, and the temporary parent announcement mechanism will guarantee cost reduction as long as at least one packet is forwarded by the temporary parent. As shown in Section 5, the savings provided by 4C are order of magnitude larger than the minimal requirement discussed here.

4.3 Experimental Evaluation

This section presents an empirical evaluation of the 4C link estimator based on the results from extensive experiments done in three testbeds: the local testbed, the Motelab testbed and a temporary outdoor testbed.

4.3.1 Experimental Setup

Our experimental evaluation can be divided in two stages. In the first stage, we run single sender experiments to compare the performance of 4C, STLE [ALB09] and 4Bit [FGJ07] under the similar experimental settings used by the authors of STLE. In the second stage, we extend the evaluation of 4C under more realistic settings with multiple sender experiments, in which all nodes in the network send packets to the root node periodically. Because the multiple sender experiments better emulate typical WSN traffic pattern than the single sender experiments, we believe the multiple sender experiments provide us a more thorough evaluation in terms of delivery cost. The detailed experimental settings and results are discussed in the following sections.

We conducted experiments on the Motelab testbed, our local indoor 54 node testbed as well as an outdoor testbed, as shown in Fig 4.12. The outdoor testbed consists of 21 TMotes, deployed near a parking lot where human activity is minimum. For all local testbed and outdoor experiments, we set the radio output power level of CC2420 to -25 dBm, packet length to 30 bytes and use wireless channel 26 to avoid 802.11 interference. For the Motelab experiments, the parameters are the same except for the radio output power is set to max (0 dBm) for better connectivity.

4.3.2 Single Sender Experiment Results

Due to the similarity of STLE and 4C design, it is reasonable to compare the performance of these two link estimators. To provide a plausible comparison, we implemented the STLE based on the design of Alizai *et al.* [ALB09] to the best

Figure 4.12: The 21-node outdoor testbed.

of our ability [1].We followed the experimental settings in [ALB09] and conducted a series of single sender experiments in both Motelab and the local testbed.

In single sender experiments, only the sender node sends packets to the root node with a fixed interval using CTP. In each experiment, we keep the same sender/root pair but employ different link estimators to compare the communication costs of CTP with 4C, STLE and 4Bit under the same network condition. We vary the node pairs and packet sending interval (SI) to study how these link estimators perform under varying conditions. Please note that SI is different from the inter-packet interval (I) used in model training: SI represents the data rate in the application layer, whereas I is a parameter used in the training.

We use the end-to-end delivery cost as the performance parameter, which refers to the number of communications needed to deliver a packet to the sink. It is the sum of the send attempts including retransmissions, at each hop along the path. Another performance parameter is the end-to-end delivery rate, but it is close to 100% in all the experiments and therefore omitted in the evaluation.

[1]We tried getting the STLE code a few times from Feb. to Aug. 2010, but the code was never provided by the authors.

Label	Node Pair	Configuration
V1, V2	$183 \rightarrow 50$, $137 \rightarrow 9$	Vertical
D1, D2	$183 \rightarrow 9$, $137 \rightarrow 50$	Diagonal
H1, H2	$9 \rightarrow 50$, $183 \rightarrow 137$	Horizontal

Table 4.3: Node pairs in the Motelab experiments.

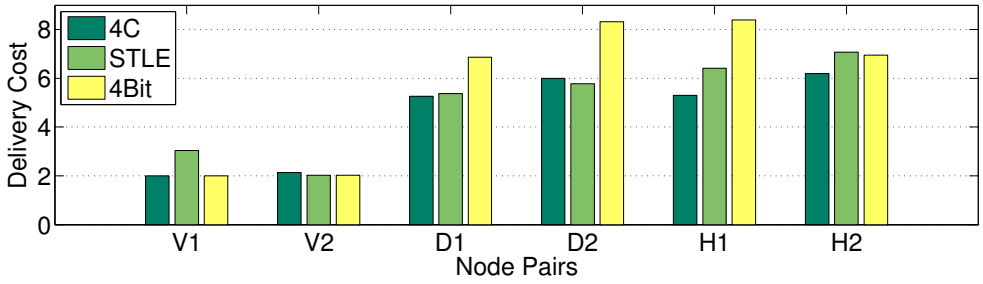
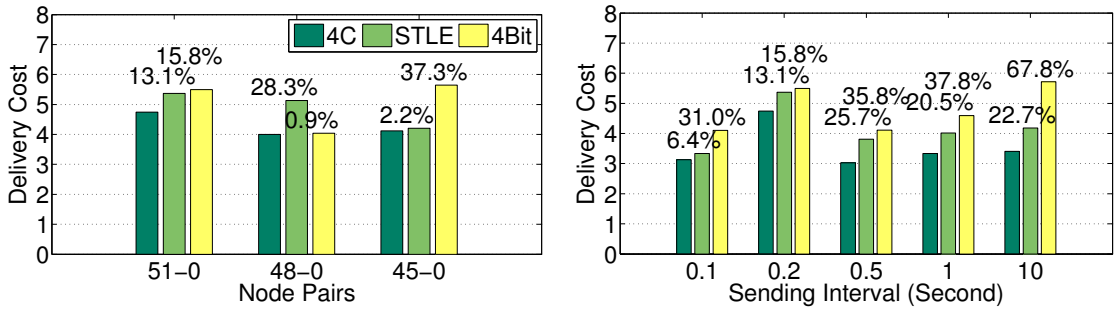


Figure 4.13: Cost comparison of 4C, STLE and 4Bit. the percentages on the top indicate the reduction rates of 4C over STLE and/or 4Bit. Labels in the x-axis represent the node pairs listed in Table 4.3, and the percentages on the top indicate the reduction rates of 4C over STLE and/or 4Bit.

4.3.2.1 Motelab Experiments

To emulate the network environment of the original STLE experiments as close as possible, we used three of the node configurations selected in [ALB09], namely, vertical, diagonal and horizontal. Vertical configuration means the source and destination are on different floors and on the same end; diagonal configuration means the source and destination are on different floors but on the opposite ends; and in the horizontal configurations the source and destination are on the same floor and on the opposite ends. The actual node pairs used are listed in Table 4.3. For each node pair, we let the source node send packets with $SI = 200$ milliseconds for 15 minutes using 4C, STLE and 4Bit respectively.



(a) Local – Different Nodes

(b) Local – Different Intervals

Figure 4.14: Cost comparison of 4C, STLE and 4Bit in the local testbed.

4C uses an LR model trained with data collected from the Motelab using the following parameters: input features = $[PRR, LQI]$, $L = 5$, $P = 1000$, $W = 1$ and fixed $I = 200$ milliseconds.

Fig. 4.13 shows the average delivery costs of the Motelab experiments. In the vertical configurations, the costs of all three link estimators are similar due to the short distance between the source and destination in terms of routing. However, STLE may have a larger cost as it can be seen in the V1 case. In the diagonal cases, STLE and 4C perform similarly whereas both of them outperform 4Bit. Finally, in the horizontal cases, usually the most common case of nodes on the same horizontal plane for many applications, 4C clearly outperforms both STLE and 4Bit whereas STLE may provide better performance compared with 4Bit in some cases (H1), or comparable in some other cases (H2). It is clear that overall, the original CTP with 4Bit has higher delivery costs compared with STLE and 4C. The results show in the most common cases for real deployments (horizontal), 4C can perform significantly better than STLE. In the vertical and diagonal cases, 4C's performance is at least comparable to STLE.

4.3.2.2 Local Testbed Experiments

We repeat the single sender experiments with three node pairs in the local testbed to verify the results. Also, to test the effect of packet sending interval in terms of delivery cost, we vary the SI from 100 milliseconds to 10 seconds. In all these experiments, 4C always employs the prediction model trained with a I that matches the actual SI. The results are presented in Fig. 4.14.

Fig. 4.14(a) presents the cost comparison between CTP with 4C, STLE and 4Bit link estimator run on different node pairs. Due to the topological constraints of the testbed, all the node pairs are of horizontal configuration, but the distance between the sender and the root node is different: link $51 \rightarrow 0$ has the longest distance, $48 \rightarrow 0$ is the second longest link and $45 \rightarrow 0$ is the shortest link. The result is similar to the horizontal experiment results from Motelab: either 4Bit or STLE have the highest cost, whereas 4C has the lowest cost in all the experiments.

Furthermore, Fig. 4.14(b) shows that with varying data rates, 4C outperforms STLE and 4Bit in all our experiments. It is interesting to note that in the experiments with short packet intervals ($SI \leq 1$ second), the delivery cost reduction of 4C versus 4Bit is generally lower than the experiments with the long intervals ($SI = 10$ seconds). It is because the path cost is updated more frequently with the shorter SI: with more frequent data packets transmissions, the ETX of the links are estimated in a faster pace, and therefore the cost estimation provide by 4Bit is more accurate. However, in the long SI cases, the ETX update is much slower and can only reflect the quality of the stable links, in other words, short links with high quality. As to the 4C case, the link quality is updated in a per-packet basis, therefore the estimation is more agile, which, in turn, enables the routing protocol to choose the long, temporary high quality links. STLE shows

Model	L	P	W	I
M1	5	1000	1	10 seconds, periodic
M2	5	1000	1	10 seconds, aperiodic
M3	5	5000	1	60 seconds, aperiodic

Table 4.4: Modeling parameters of the LR models used in the multiple sender experiments.

similar advantages but less efficient than 4C.

These results lead us to believe that overall, 4C can harness the potential of intermediate links better than STLE does under different network environments and varying traffic rates. Even in the worse performance cases, the delivery cost of 4C is still comparable with that of STLE or 4Bit.

4.3.3 Multiple Sender Experiment Results

4.3.3.1 Multiple Sender Experiment Settings

We continue our evaluation with multiple sender experiments, which try to emulate the traffic pattern of a typical data collection application where *all* the nodes in the network send packets to a single root node. Similar to the single sender experiments, each multiple sender experiment uses CTP with three different link estimators (4C, STLE and 4Bit) in three one-hour runs respectively, so the total time of an experiment was 3 hour. Furthermore, for each unique network settings, we repeat the experiment three times to minimize the effects of temporary network irregularities. Therefore, each experimental result under different network conditions is actually from 9 hours of experiments. These experiments covered day and night times, and there are no significant differences between day and night in terms of performance. Due to the relatively low data rate in many

WSNs applications, we focus on communication costs and reliability in this work.

We vary network density and packet sending interval (SI) in these experiments. Network density refers to the number of nodes we include in each experiment. In the local testbed experiments, we used two densities: high density, which includes all the 54 nodes in the testbed, and low density, which includes 18 nodes in a line topology. For SI, we use 10, 30 and 60 seconds in the local testbed, and 10 seconds in the outdoor testbed. The sending intervals are longer than what we used in the one sender experiments because i) traffic rates in these intervals are more aligned with some real WSN applications, and ii) longer intervals can reduce network congestion given the larger number of senders. Moreover, to avoid correlated interference, the actual packet sending interval is randomly chosen from $[\frac{1}{2}SI, \frac{3}{2}SI]$.

The combinations of different network densities and sending intervals creates a rich set of network conditions that covers a variety of realistic scenarios, including dense networks with high traffic rate and inter-node interference. For example, in the experiments done with 10 seconds interval and high network density, all the 54 nodes in the testbed sent 1 packet every 10 seconds, resulting in an average traffic rate of 5 packets per second for the nodes that are close to the root. Considering the densely deployed network (See Fig. 4.3), the inter-node interference was almost unavoidable. We avoid the heavy network congestion cases as neither 4C nor CTP [GFJ09] are design to handle heavy network congestion. Nevertheless, due to the reliable retransmission mechanism of CTP and the randomized sending interval, all the multiple sender experiments achieved near 100% end-to-end delivery rate. Therefore, we consider the delivery cost as the main performance metric in this work.

Please note that having 100% end-to-end delivery rate does not mean that

the links used in the forwarding path are prefect. The retransmission scheme of CTP will keep retransmit the lost packets that has not been ACKed for up to 30 times, providing reliable end-to-end packet delivery unless the network is partitioned or is under heavy congestion. In our evaluation, we found that the links used in packet forwarding have various qualities instead of near 100%.

Moreover, from routing point of view, finding good quality links does not guarantee an optimal forwarding path. The more challenging task is to find and use the long, intermediate quality links in addition to high quality links to provide the most efficient path, i.e., the forwarding path with the least end-to-end delivery cost. In this sense, most of the high quality links are irrelevant to the routing decision as only a few of them (i.e., the links that provide most routing gain) should be considered. This is exactly why 4C adopts a data-driven prediction model: to better evaluate the delivery cost of the links with intermediate quality, such that the routing protocol can utilize the long, intermediate quality links more aggressively in finding the optimal path.

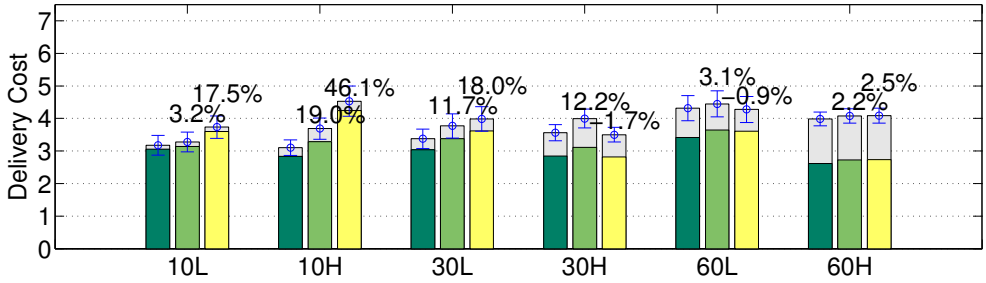
To evaluate the performance of models trained with different parameters, we use LR models based on the modeling results discussed in Section 4.1.7. As seen in Fig. 4.9, LR models trained with $L = 5$, $P = 1000$ are enough to provide good prediction results when $I = 10$ seconds, whereas models trained with $L = 5$, $P = 5000$ perform well when $I = 60$ seconds. Therefore, we include three models (M1, M2 and M3) in the evaluation as listed in Table 4.4. The first two models we chose (M1 and M2) are trained with input features = $[PRR, LQI]$, $L = 5$, $P = 1000$, $W = 1$, periodic and aperiodic $I = 10$ seconds. The third one (M3) has the same parameters as M2 except for $P = 5000$ and the aperiodic $I = 60$ seconds. All three models were tested in the local testbed experiments whereas model M1 and M2 were used in the outdoor testbed experiments.

Fig. 4.15 compares the delivery costs of the local testbed experiments. In each figure, the labels in the x axis note the experiment conditions: the number 10, 30 or 60 represents the SI, and the letter L or H represents the network density (18 and 54 nodes respectively). The gray boxes on the top represent the beacon overhead, i.e., the number of beacons sent per data packet delivered. The percentages on the top of the bars are the reduction ratio of 4C compared to STLE and 4Bit results. Next, we discuss the results in terms of delivery cost, path length and beacon overhead.

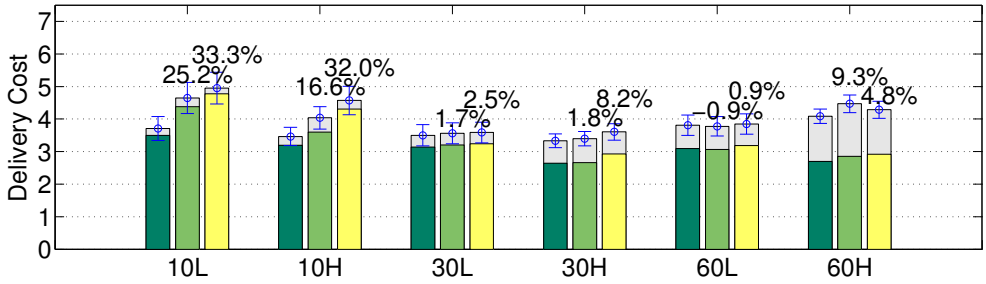
4.3.3.2 Delivery Cost

Delivery cost refers to the number of send attempts needed to deliver a packet to the sink. From Fig. 4.15(a), a significant cost reduction of 4C over 4Bit (more than 46%) can be observed on the experiments with 10 seconds SI. Since the LR model (M1) used in 4C is also trained with $I = 10$ seconds, this reduction gain implies that the 4C is indeed selecting better routes compared with 4Bit when the modeling parameters match the actual network conditions. The cost reduction is less significant as the SI increases, but the cost of 4C is still comparable with STLE and 4Bit even in the worst case. On other hand, STLE generally shows lower cost than 4Bit, but 4C is able to outperform STLE in almost all cases. Similar gain can be observed in Fig. 4.15(b), indicating that the performance of the model is agnostic to the periodicity of I used in the model training. In Fig. 4.15(c), which shows the results of 4C using model trained with $I = 60$ seconds, we see that cost gain is more pronounced in high SI scenarios, i.e., $SI = 30$ and 60, than $SI = 10$ seconds cases.

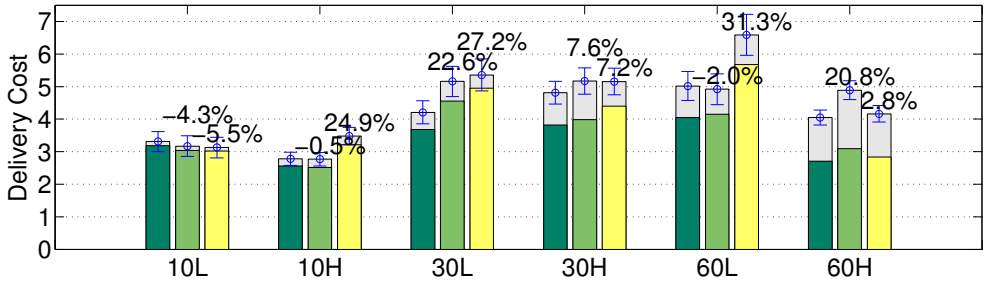
To verify the statistical significance of these results, we run t-tests on the 4C and 4Bit results and present the resulting p-values in Table 4.5. The p-value



(a) Local, M1



(b) Local, M2



(c) Local, M3

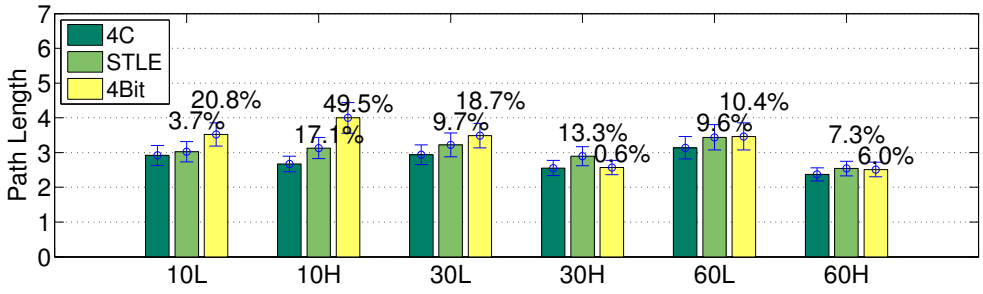
Figure 4.15: Cost comparison of multiple sender experiments run on the local testbed. Y axis denotes the average delivery cost (lower is better). Labels in the x axis in figure (a), (b) and (c) note the experiment conditions: the number 10, 30 or 60 is the sending interval, and the letter L, or H represents the network density. The gray boxes on top of the bars represent the beacon overhead, and the percentages on the top indicate the reduction rates of 4C over STLE and/or 4Bit.

Model	10L	10H	30L	30H	60L	60H
M1	**0.0375**	**0.0029**	**0.0441**	0.4160	0.6781	0.7280
M2	**0.0149**	**0.0230**	0.6428	0.7317	0.8843	0.7287
M3	0.7335	**0.0465**	**0.0283**	0.2716	**0.0105**	0.1091

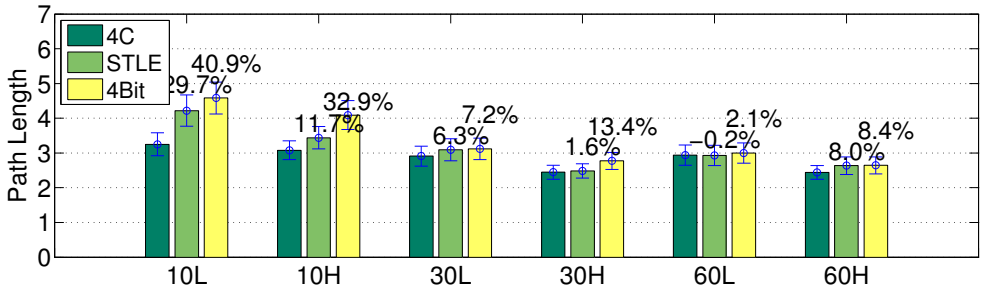
Table 4.5: P-values from t-tests run on the delivery cost results of 4C and 4Bit in multiple sender experiments. A p-value smaller than 0.05 indicates the cost difference between 4C and 4Bit are significant with 95% confidence.

represents the confidence that null-hypothesis is true, i.e., there is no significant difference between the 4C and 4Bit results. Therefore, if the p-value is smaller than 0.05, we can justify that the results of 4C are significantly different from 4Bit with 95% confidence. As seen in Table 4.5, the p-values for 10 seconds SI experiments with model M1 and M2 are all smaller than 0.05, indicating that the cost reduction is significant in these cases. Also, in the cases of M3, the cost differences are significant in three experiments with SI equals to 10, 30 and 60 seconds respectively, showing that M3 trained with $I = 60$ seconds performs better when the SI is also high.

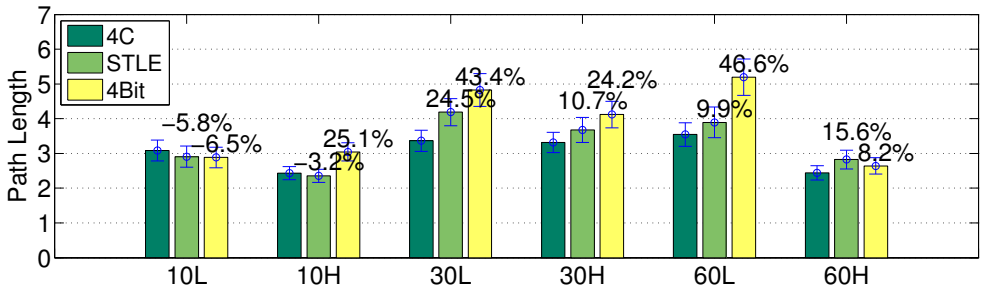
These results indicate that the modeling parameters of the model indeed affect the prediction performance: the models trained with $I = 10$ seconds can do well under the same network traffic rate, but when the traffic pattern and I are misaligned, the model can not effectively predict the packet reception probability anymore. However, even in these cases the performance of 4C is comparable to 4Bit, the default link estimator of CTP.



(a) Local, M1



(b) Local, M2



(c) Local, M3

Figure 4.16: Path length comparison of multiple sender experiments run on the local testbed.

4.3.3.3 Path Length

Path length is the number of hops along a path. It measures routing depth of nodes in the network, and is related to end-to-end latency and energy usage. Fig. 4.16 present a path length comparison in the local testbed experiments. Similar to Fig. 4.15, Fig. 4.16 presents the path length of CTP with 4C, STLE and 4Bit in different network sizes and densities, with each figure representing a different model listed in Table 4.4. The labels on the X axis denote the network configurations, and the percentages on the top indicate the path length reduction of 4C with respect to STLE and 4Bit correspondingly.

It is obvious that the path length distribution of CTP with 4C, STLE and 4Bit follows the trend observed in the delivery cost results very closely (see Fig. 4.15). In Fig. 4.16(a) and Fig. 4.16(a), the path length of CTP with 4C is significantly shorter than STLE and 4Bit when the modeling parameter I matches with the actual data sending interval SI (10 seconds), but the path length difference becomes less significant as the data rate SI moves away from the modeling parameter I. Similarly, the path length of 4C using model M3 (longer I) is much shorter than STLE and 4Bit in the 30 and 60 seconds SI cases, whereas in the 10 seconds SI cases the path length of all three link estimators are close to each other.

Also note that the path length reduction rate shown in Fig. 4.16 is slightly higher than the cost reduction rate shown in Fig. 4.15. This indicates that the path quality of 4C is slightly worse that the path selected by CTP with 4Bit, which leads to slight increase of packet retransmission. Nevertheless, the shorter path length offsets the increased retransmission cost, resulting the delivery cost reduction observed in Fig. 4.15.

The closeness between the delivery cost and the path length results essentially shows that the cost of sending a packet over each link is close to 1 in all the

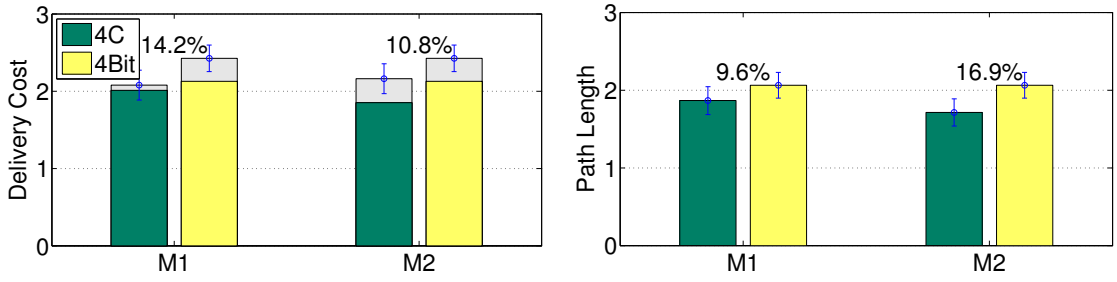
experiments, therefore the end-to-end delivery cost is simply the number of hops in the forwarding path. However, the fact that CTP with 4C is able to find these high quality paths with fewer number of hops proves that the prediction models can indeed exploit the high quality periods of long, intermediate links with historical packet information as old as 10 to 60 seconds.

4.3.3.4 Beacon Overhead

Beacon overhead is the average number of beacons sent by the routing protocol while it delivers one packet to the sink. The number of beacons represents the overhead of the routing protocol for maintaining a connected network structure. Compared with 4Bit, the main overhead of 4C as well as STLE is the temporary parent notification packets. Fig. 4.15 show the beacon overheads of 4C, STLE and 4Bit as the gray box on top of bars corresponding to each link estimator. It can be observed that the beacon overhead is similar in almost all experiments, which indicates that 4C does not incur significant overhead compared to CTP with 4Bit. Also, the beacon overhead experiments with long SI is higher than experiments with shorter SI, indicating that the beacon overhead is independent of the traffic rate.

4.3.3.5 Outdoor Testbed Results

We also repeated the multiple sender experiments in the outdoor testbed (see Fig. 4.12) to verify the local test results. The outdoor deployment was temporary and these outdoor experiments were only done for verification purposes. Therefore, we use models M1 and M2 in the outdoor experiments, which are trained with 5 links from the local indoor testbed data as specified in Table 4.4. The sending interval SI is set to 10 seconds, and the network density is fixed to



(a) Cost (b) Path Length

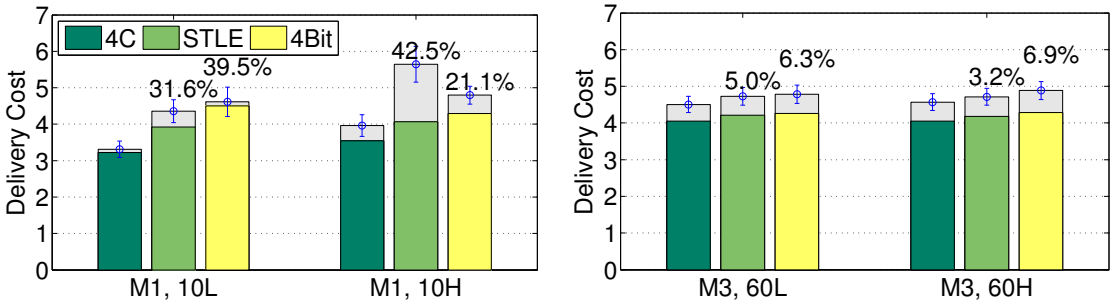
Figure 4.17: Cost and path length of 4C running on outdoor testbed. Two models with periodic/aperiodic Is are used ($W = 1$, average $I = 10$ seconds).

21 nodes. All other parameters are the same as the local testbed experiments. Also, at the time of the outdoor experiments, the STLE implementation was not available yet, so we only evaluate the performance of 4C and 4Bit in the outdoor environment.

Fig. 4.17 shows the delivery cost as well as the path length of CTP using 4C and 4Bit. In Fig. 4.17(a) we can observe that 4C shows more than 10% improvement over 4Bit in terms of delivery cost, and Fig. 4.17(b) shows similar path length reduction. In short, our outdoor experiment results confirmed the observation in the local testbed: 4C link estimator can reduce the end-to-end delivery cost significantly when the modeling parameter matches the actual network traffic.

4.3.3.6 Impact of External Interference

Interference from external sources, such as WiFi communication, Bluetooth and small appliances, is a common problem to wireless communications. To evaluate the impact of external interference to the performance of 4C, we run further multiple sender experiments in the local testbed using radio channel 12, which



(a) M1 with 10 seconds SI

(b) M3 with 60 seconds SI

Figure 4.18: End-to-end delivery cost of the multiple sender experiments run in the local testbed under external interference. Labels in the x-axis denotes the experiment conditions: M1 and M3 represent the prediction model, the number 10 and 60 is the sending intervals and the letter L, or H represents the network density. The gray boxes on top of the bars represent the beacon overhead, and the percentages on the top indicate the reduction rates of 4C over STLE and/or 4Bit.

is under constant interference from the 802.11 (WiFi) communications in the building. The new experiments use four of the experimental settings described in Section 5.3.1, namely, model M1 with 10 seconds sending interval and two different network densities (low and high), as well as model M3 with 60 seconds sending interval and low/high network densities. Fig. 4.18 presents the results in terms of end-to-end delivery cost and beacon overhead. Similar to Fig. 4.15, the height of the bars represents the end-to-end delivery cost, and the gray box on top of each bar represents the overhead of beacon packets. The experiment conditions are noted in the x-axis: M1 and M3 represents the LR prediction model used by 4C, the number 10 and 60 represents the sending interval, and finally the letter L and H denotes the low/high network densities (10 and 54 nodes respectively).

Fig. 4.18(a) and 4.18(b) show that the end-to-end delivery cost under external interference is generally larger than without it (see Fig. 4.15(a) and 4.15(c). This is to be expected, as the presence of external interference will likely cause packet loss and link quality degradation. When analyzing the relative cost reduction between 4C and 4Bit, in the case of 10 seconds sending interval with model M1 in a low density network (see Fig. 4.18(a)), the delivery cost reduction of 4C over 4Bit increases from 17.5% to 39.5% compared to the external interference-free experiments with the same settings (see Fig. 4.15(a)). However, in the 10 seconds sending interval and a high network density case, the cost reduction drops from 46.1% to 21.1% under the external interference from WiFi. On average over all the densities, with the M1 model settings, the relative cost reduction of 4C over 4Bit under interference is almost the same as the average cost reduction without interference (30.3% and 31.8% respectively), which implies that overall, the external interference only slightly affects the performance of 4C when the sending interval is 10 seconds.

The impact of interference is more significant when the sending interval is longer (60 seconds). As shown in Fig. 4.18(b), the cost reduction of 4C over 4Bit with 60 seconds sending interval and model M3 in low/high density network is 6.3% and 6.9% respectively. Compared with 31.3% and 2.8% cost reduction in the same experiment settings without interference (see Fig. 4.15(c)), the average cost reduction drops from about 17% to 7%. Combined with the 10 seconds sending interval results presented in Fig. 4.18(a), these results show that the performance of 4C starts to degrade significantly under the external interference as the packet sending interval increases from 10 seconds to 60 seconds, indicating that the prediction model of 4C can not accurately predict the link quality under interference with sending intervals larger than 10 seconds. Nevertheless, compared with 4Bit, 4C is able to reduce about 7% of the average delivery cost even when the sending interval is 60 seconds.

4.4 Discussion

4.4.1 Advantages of a Data-Driven Approach

According to communication theory, the PRR-SNR correlation can be derived from the frame size, the coding scheme and the bit error rate defined by the modulation format [Rap01]. For example, the link layer models proposed in [ZK04] provide deterministic functions to calculate PRR with SNR for a variety of modulation formats and coding schemes. Although these models are useful in network simulation, this idealized approach can not provide accurate PRR estimation for intermediate quality links which show large PRR variance [ZG03, LMH03]. Moreover, research [CE03, ZHK04, SKH06] shows that the actual correlation between PRR and PHY parameters may be different due to hardware specific variations.

As shown by Son *et al.* [SKH06], even at the same measured signal strength at the receiver, the signals from different sources may have different levels of distortion, in turn affecting the packet reception differently. Therefore, in order to find the right function to calculate PRR based on SNR, a user will need to collect PRR/SNR data across all possible node pairs in a network, and the function coefficients for each node pair may be different. This level of complexity will render this approach unfeasible for any network with more than dozens of nodes.

We account for this problem with a data driven approach. With a sufficiently large training dataset collected from actual links in the network, the machine learning algorithms can find the optimal function by minimizing the error between the output and the actual packet reception. From a modeling perspective, this approach can be viewed as a way to find the best overall correlation between PRR and PHY parameters given a training dataset, whereas the communication theory approach is a way to find the best correlation for a node pair. Therefore, the trained models can represent the optimal correlation over the underlying network in the deployment site as a whole. In addition, our model combines the PRR and a variety of PHY parameters to estimate the link quality instead of using only SNR. Another difference is that the 4C employs a receiver-initiated design, which means the quality estimation happens on the receiver side.

4.4.2 Training Data Requirements

A major concern of a data driven approach is the data requirement. In the previous sections we show that our modeling approach can indeed capture the underlying PRR distribution with small amount of data. Therefore, a sufficiently large training set is necessary to capture the underlying PRR distribution. Both the simulation and experimental results show that a training set consisting of

packet reception data collected from a few links for several minutes is sufficient to train the prediction model. If sufficient training data is available, our model can find correlations that *exceed the hundred of milliseconds*, and extend to *many tens of seconds* as shown in our results in Section 4.3.

4.4.3 Limitations

The evaluation results show that the performance of prediction models is similar with state-of-art link estimator when there are non-negligible network dynamics, e.g. changing packet sending interval. As discussed in Section 4.3, 4C performs on par with 4Bit when the actual traffic pattern does not match the modeling parameter, indicating that the prediction is inaccurate in this case. This is due to the fact that 4C only incorporates one prediction model, and consequently, can not adapt to varying network conditions from the original training set very well. However, we showed that the training can be done with a small amount of data, and 4C could potentially use multiple models trained with freshly collected data for different network conditions. Furthermore, it is possible to apply online learning algorithms in prediction model such that the prediction model can evolve with the changing network conditions, which is discussed in next chapter.

The performance of 4C is also limited by the link quality diversity of the network. 4C improves the overall transmission cost by identifying temporal high quality links with agile link quality prediction, and therefore the improvement is limited by the number of intermediate quality links in the network. The correlation between the link quality and the spatial distribution of the nodes is beyond the scope of this work, but based on our experience, it is not hard to find intermediate quality links in the local indoor testbed as well as in the Motelab testbed. As described in Section 3.4, the packet traces collection process in the

local testbed yielded 68 links with 12 low quality links, 14 intermediate links and 42 high quality links. For the Motelab testbed, we selected 10 links with PRR ranging from 0.13 to 0.84 out of 84 links we collected. Therefore, we do not consider finding intermediate links a big problem in this work.

Please note that in densely deployed networks where the high quality links are abundant, the number of intermediate links is also large due to the large number of nodes within the communication range of each other. By leveraging these long, intermediate links with high routing gain, 4C can reduce the transmission cost in dense network just as well in sparse networks with only marginal or unstable links, if not better. Figure 4.15(a) shows one such example: the cost reduction of 4C compared with 4Bit is actually larger in the high density network (labeled as 10 H) than in the low density network (labeled as 10L).

4.5 Summary

In this chapter we showed the usefulness of link quality prediction based on different machine learning methods, such as, naive Bayes classifier, logistic regression and artificial neural networks. Our models take a combination of PRR and PHY information as input, and output the reception probability of the next packet. We showed that users need very little data (5-7 links for a couple of minutes) in order to train the models in the environments tested. Our analysis showed that logistic regression works well among the three models with the additional advantage of having the small computational cost. Using this knowledge, we implemented 4C, a novel link quality estimator in TinyOS. We conducted extensive experiments in the Motelab and our local indoor testbeds, as well as an outdoor deployment. Our results show improvements in the order of 20% to 30% compared with 4Bit and STLE estimators in single and multiple sender experiments, with some cases

improving performance by more than 45%.

One limitation of 4C is that it does not adapt very well to the varying network conditions. Although 4C can be trained to fit to a deployment network with specific parameters such as inter-packet intervals, the trained prediction model is static and can not cope with non-negligible network dynamics. To overcome the limitation, we examine the possibility of incorporating online learning algorithms for link quality prediction under dynamic conditions in the next chapter.

CHAPTER 5

Short Temporal Link Quality Prediction with Online Learning

Previous chapter discussed the design of 4C, a prediction based link estimator trained with specific network conditions of the deployment network, and showed that 4C significantly improve the routing performance of the networks with inter-packet intervals in the order of several seconds. In this chapter, we further examine the possibility of employing online learning techniques to create prediction models that adapt to the changing network conditions.

An essential requirement of such prediction-based estimator is *adaptivity*. When the network exhibits large dynamics, the link estimator should be able to adjust itself to cope with changes. While it might be possible to find the correct set of parameters in an estimator to improve its performance for a certain level of dynamics, this parameter set will not work in all the cases as we deploy in different environments or even as the temporal dynamics change in the same location.

Another important feature of the estimator is *plug-and-play*. Ideally, a link estimator should work on any network without pre-deployment efforts to tune the prediction model. 4C can significantly outperform link estimators such as STLE [ALB09] and 4Bit [FGJ07], but the main disadvantage of 4C is the need to collect link data at the target deployment site for training the link prediction

model. Although the required training dataset is small, collecting it still requires additional effort and might not be feasible for all deployments. Furthermore, as wireless conditions change from the time we collect the training data, the same set of model parameters may cause performance degradation. Therefore, it is important to have an estimator that needs no off-line training data or prior knowledge from the target deployment.

Based on the above requirements, we propose Temporal Adaptive Link Estimator with No off-line Training (TALENT), an adaptive prediction based link estimator that focuses on estimating temporal link quality variations. TALENT utilizes online learning algorithms to adapt to different network conditions *without* any user intervention and no *a priori* training and is designed to be a *plug-and-play* estimator for any environment and level of dynamics. Due to the dynamic nature of the wireless channel, such prediction will be valid for only a short period before the link quality changes. Nevertheless, with the knowledge of expected link quality in the immediate future, the routing protocol may be able to select efficient data forwarding path promptly during a burst transmission of data packets, and ultimately increase delivery efficiency and reduce communication costs.

Contribution

The contributions of TALENT are four fold. First, we show that by using online learning techniques, our prediction model can adapt to a wide range of network dynamics without prior training data and with fast convergence time. To our knowledge, this is the first attempt to introduce online learning techniques to adapt network link estimation parameters under environmental and network dynamics. Second, we designed and implemented TALENT in TinyOS and integrated it into CTP for a reference implementation. Third, we integrated TALENT with LPL [PHC04], a low-power listening protocol for efficient communication

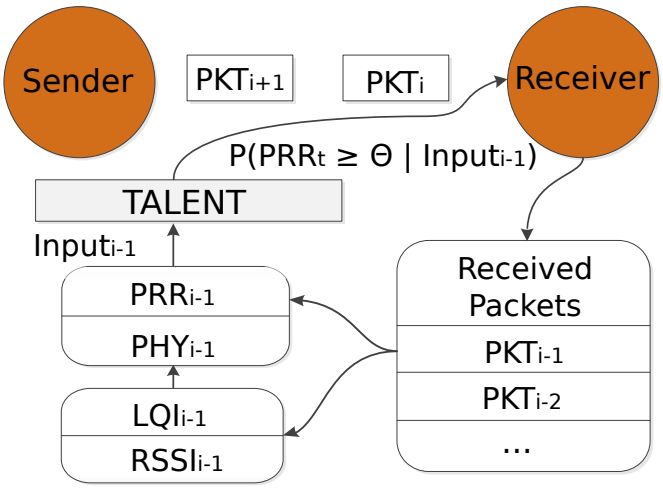


Figure 5.1: TALENT attempts to infer the probability that future PRR exceeds a certain threshold θ.

and actual energy savings when using duty-cycled radios. Finally, we show that by utilizing TALENT, the routing protocol could use intermediate links more efficiently and achieve lower communication costs when sending data packets in bursts. From our extensive experimental evaluation, we show significant improvement of packet delivery efficiency, with an average of 95.3% improvement over 4Bit [FGJ07] in many different scenarios, as well as an improvement in end-to-end delivery rate. Furthermore, we applied the prediction approach of TALENT to empirical packet traces from 802.11 networks and confirmed that TALENT outperforms ETX based link estimators significantly even in 802.11 networks with much higher data rate. These results suggest the potential application of TALENT in much wider range of wireless networks.

5.1 Modeling

Similar to 4C, we propose to build models to predict the future link quality with information from both the physical layer and the link layer. The intuition is the same: by using a combination of PRR from the link layer and PHY parameters from physical layer as input, the proposed model could supplement the PRR, accurate for long term link estimation, with PHY parameters to improve the short temporal quality estimation for the intermediate links, which are highly unstable and exhibit the most variations. The main differences between TALENT and 4C are 1) 4C only predicts the reception of the next packet with given inter-packet interval, whereas TALENT predicts whether the link quality will be high in the near future, and 2) TALENT utilizes online learning algorithms such that the models can adapt their parameters to the network dynamics without the overhead of data collection and training, whereas 4C uses static model trained with data collected from the deployment site. The follow sections address the modeling aspect of TALENT in detail.

5.1.1 Problem Definition

TALENT inherit many of the modeling aspects from 4C. As described in Section 4.1 of the previous chapter, the input of our model is the historical information available from W packets comprised of PRR information and PHY parameters. Different from 4C, the output of the model is now the probability of the temporal link quality being better than the threshold θ during the time t in the future:

$$P(PRR_t \geq \theta | Input_i)$$

The calculation of the instantaneous link quality PRR_t is subject to the time t and the number of packets sent during that time. For our analysis, we set θ to 0.9, the time t to 1 second, and the number of packets sent during t is fixed to 10 with a inter-packet interval (I) of 0.1 seconds. We chose a small t and short inter-packet interval because the temporal variation of the wireless channel is correlated at intervals smaller than 1 second [ABB04, SKA, RL09], and our model tries to take advantage of this behavior. This phenomena also is confirmed in section 5.1.5 which shows that the performance of the proposed model degrades rapidly as the inter-packet interval increases from 0.1 to 1 second. We use the same parameters in our experiments in order to best approximate the real network conditions.

A major difference between TALENT and 4C is the model output. 4C tries to predict the success probability of the next packet, whereas in TALENT, the model output is the probability of future link quality higher than a threshold over a short period of time. From the routing prospective, the evaluation of link quality over a period of time can help a routing protocol decide on the predicted quality of a path in a more powerful way than a prediction of an individual packet. Another advantage is that by predicting average link quality over a larger time scale than the reception of a single packet, we smooth the random noise that might affect the individual packet reception. Therefore, we consider the output of TALENT superior to 4C for practical routing purposes.

5.1.2 Modeling Method

The complex dynamics of wireless networks cannot be captured by a single rigid model. For example, the correlation between PRR and RSSI will likely change if a noise source is added to the network. In this case, the model needs to be adaptive to the changes and follow the dynamics by choosing a new set of model

parameters.

We propose to use a stochastic gradient descent (SGD) online learning algorithm [Mit97] to train a logistic regression classifier (LR) such that the model can adapt to the changing network conditions. We choose to use LR models because 4C has shown that the link quality can be accurately predicted by LR models. In Section 4.1 we compared the link quality prediction performance of three classification algorithms, namely, Naive Bayesian classifier (NB), logistic regression (LR) and Artificial Neural Network (ANN). Through extensive evaluation on empirical packet traces, we found that that ANN and LR perform similarly, whereas NB's performance is inferior to these two. Therefore, we selected LR over ANN due to less computation complexity and the similar performance. For the detailed performance analysis please refer to Section 4.1. For the online learning algorithm, we considered many many online learning frameworks, such as weight majority, winnow and SGD, among others [Ber12], and we settled for SGD mainly due to its performance and simplicity to implement it under stringent computational and energy constraints.

The idea of SGD is simple: based on the error gradient of each prediction result and its corresponding target, SGD updates the parameters of the LR model, and therefore learns the link characteristics online. Furthermore, we employ learning rate adaption algorithms such that the model can dynamically adjust the learning speed based on the error gradient. With the learning rate adaption algorithm, the model is able to accelerate the learning when the error is large so that it can quickly adapt to the underlying link quality variations, and slow down to a stable state when the error is small.

Formally speaking, assume $X =< X_1 \dots X_n >$ represents the input vector *Input* discussed in the previous section, and Y is the binary variable denoting

the high temporal link quality $PRR_t > \theta$, the logistic regression classifier can be expressed as:

$$P(Y = 1|X) = \frac{1}{1 + \exp(-f(X))}$$

and

$$P(Y = 0|X) = \frac{\exp(-f(X)}{1 + \exp(-f(X))}$$

where $f(X) = \beta_0 + \sum_{i=1}^{n} \beta_i X_i$, and β is a vector of the weight parameters to be estimated.

The input X is the model input defined in the previous section, which consists of the PHY parameters and PRR associated to W historical packets. Given a training set of N samples, $\{(X^1, Y^1) \ldots, (X^N, Y^N)\}$, we train the logistic regression classifier by maximizing the log of the conditional likelihood, which is the sum of the log likelihood for each training example:

$$\begin{aligned} l(\beta) &= \sum_{l=1}^{N} Y^l \log P(Y = 1|X^l, \beta) \\ &\quad +(1 - Y^l) \log(P(Y^l = 0|X^l, \beta)) \end{aligned}$$

Note that due to the fact that Y can take only values of 0 and 1, only one of the two terms in the expression will be non-zero for any given Y^l.

To maximize the log likelihood, we use the gradient, which is the partial derivative of the log conditional likelihood. The ith component of the gradient vector is:

$$\frac{\partial}{\partial \beta_i} l(\beta) = \sum_{l=1}^{N} (Y^l - \hat{P}(Y^l = 1|X_i^l, \beta)) X_i^l$$

where $\hat{P}(Y^l = 1|X_i^l, \beta)$ is the logistic regression prediction using equations (5.1.2) and (5.1.2) and the weights β.

A common approach to learn the weights is batch training, which updates the weights β on the basis of the gradient accumulated over the entire predefined

training set:

$$\beta_i \leftarrow \beta_i + \lambda \sum_{l=1}^{N} (Y^l - \hat{P}(Y^l = 1|X_i^l, \beta))X_i^l$$

where λ is the learning rate which determines the step size.

Different from the batch training that optimizes the cost function defined on all the training samples, SGD is an online algorithm that operates by repetitively drawing a fresh random sample and adjusting the weights on the basis of this single sample only. It performs weight updates on the basis of the gradient of a single sample X^l, Y^l:

$$\beta_i \leftarrow \beta_i + \lambda \Delta\beta_i^l$$

where $\Delta\beta_i^l$ is the gradient of the lth sample:

$$\Delta\beta_i^l = (Y^l - \hat{P}(Y^l = 1|X_i^l, \beta))X_i^l$$

Using an online learning algorithm to model the link quality variations has several advantages. From a networking aspect, each packet is a new sample, thus the training dataset continues to grow indefinitely. In terms of computation speed, the stochastic learning algorithms will likely outperform the batch learning algorithms that operate over a training set [BL04]. Stochastic learning is also useful when the function being modeled is changing over time, a quite common scenario in networking where the data traffic patterns and the wireless channel quality variations are both non-deterministic. Also, stochastic learning often results in better solutions because the noise in the updates can cause the weights jumping into multiple, possibly deeper local minimum, whereas batch training will only converge to one minimum [LBO98].

5.1.3 Learning Rate Adaptation

An important parameter of SGD is the learning rate λ. The rate affects the learning speed and how fast the gradient descent converges. Different from batch gradient descent, which has a linear convergence speed [DS83], online gradient descent proceeds rather slowly during the final convergence phase [BM02]. The noisy gradient estimate causes the parameter vector to fluctuate around the optimum in a bowl whose size depends on the actual learning rates. Ideally, we want a learning algorithm that converges quickly when the network is stable, and updates its parameters promptly once the prediction error increases due to network dynamics.

We tried two adaptive learning rate algorithms, ALAP and s-ALAP [ALA98]. ALAP is a normalized step size adaptation method with the main idea of changing the global learning rate λ to time-varying local learning rates $< \lambda_1 \dots \lambda_n >$ that adapt by gradient descent, while simultaneously adapting the weights. At time t, we would like to change the learning rate (before changing the weight) such that the error at the next time step is reduced. For the lth sample, ALAP performs the learning rate update with the following equation:

$$\lambda_i \leftarrow \max(0.5, 1 + q\Delta\beta_i^l \Delta\beta_i^{l-1})\lambda_i$$

where q is a meta learning rate which controls the step size of learning rate update. The weight is updated with the new local learning rate:

$$\beta_i \leftarrow \beta_i + \lambda_i^l \Delta\beta_i^l. \tag{5.1}$$

s-ALAP is a variation of ALAP with smoothed gradient descent by using an exponential trace of past gradients, which uses the following learning rate update rule:

$$\lambda_i \leftarrow \max\left(0.5, 1 + q\Delta\beta_i^l \frac{\Delta\beta_i^{l-1}}{(\Delta\beta_i^l)^2}\right)\lambda_i \tag{5.2}$$

where $\overline{(\Delta\beta_i^l)^2}$ is an exponential moving average of the square of $\Delta\beta_i^l$. The weight update rule is same as equation (5.1).

The only global parameter for both ALAP and s-ALAP is the meta learning rate q, which determines the step size of learning rate update. According to empirical experience, we set q to 0.8 in our evaluation.

Another common extension of SGD algorithm is the use of momentum term [Mit97]. With the momentum term, the weight update of lth sample becomes:

$$\beta_i \leftarrow \beta_i + \lambda^l \Delta\beta_i^l + m\Delta\beta_i^{l-1}$$

where $0 < m < 1$ is a new global momentum parameter which must be determined by trial and error. Momentum simply adds a fraction m of the previous weight update to the current one. When the gradient keeps pointing in the same direction, this increases the size of the steps taken towards the minimum and speeds up the learning process. On the other hand, when the gradient keeps changing direction, momentum will smooth out the variations. In our evaluation, we compared the performance of the two learning rate adaptation algorithms, ALAP and s-ALAP as well as the momentum to select the best candidate for the link quality predictor.

5.1.4 Online Learning Algorithm Evaluation

In this section, we evaluate the performance of the proposed online learning algorithm and compare with other link estimations techniques to understand the potential gains.

5.1.4.1 Evaluation Settings

In order to select the best algorithm for predicting short term links quality when the network dynamics are non-negligible, we need packet traces of intermediate links to evaluate the candidate algorithms. The packet traces were collected from a local wireless testbed comprised of 54 TMote Sky motes. To collect the traces, one node was programmed to send 30-byte long packets with an inter-packet interval of 0.1 seconds for 1 hour, and all the other nodes in the network record the sequence number, RSSI and LQI of the received packets respectively. The wireless channel used was channel 26, and the sending node always used full power (RF power level = 0). After the sender node stops, the packet traces recorded from intermediate links are selected in the evaluation. The process was repeated 10 times, each time with a different sending node. The data was collected during three week days at different times of the day, including day and night times. In the end, we collected extensive packet traces with more than 490 thousand packets from 18 intermediate links with average PRR ranging from 0.54 to 0.92.

We then apply SGD with momentum, ALAP and s-ALAP to these empirical packet traces to evaluate their performance in terms of prediction accuracy, i.e., the ratio of the correctly predicted high quality periods to the total number of predictions made. In order to study the performance gain of these online learning models with respect to other existing quality estimation schemes, we also apply three state-of-art link estimation schemes to the same packet traces, namely, the WMEWMA estimator described in [WTC03], the Short Term Link Estimator (STLE) proposed in [ALB09], as well as a batched trained logistics regression model (Batch).

The WMEWMA estimator is used in 4Bit, the default link estimator in TinyOS. Essentially, the output of the WMEWMA estimator is the ETX value

smoothed by an EWMA filer, which can be expressed as:

$$ETX_i = \alpha * ETX_{i-1} + (1 - \alpha) * ETX_{new}$$

where α is the smoothing factor of the EWMA filter, and the ETX_{new} is the current link ETX calculated based on packet reception of a certain window size:

$$ETX_{new} = \frac{\text{window size}}{\text{number of received packets}}$$

In 4Bit, the smoothing factor $\alpha = 0.9$ and the window size of ETX_{new} calculation for data packets is 5, and we use the same parameters to best approximate the actual output of 4Bit. In order to compare the performance of WMEWMA and the prediction models, we consider the output of WMEWMA as a prediction of future link quality: if the output ETX value correctly indicates the link quality in the next 1 second is higher than 0.9, the output is marked as correct, otherwise it is marked as false. This procedure converts the output of WMEWMA in the range of $[1, \infty]$ into a binary value, so that we can compute the prediction accuracy of WMEWMA similar to a prediction model.

While one could tune the WMEWMA parameters such that it can better match the level of dynamics seen by any particular data trace, please note that any fixed set of parameters will not adapt to the changing conditions since one parameter set does not fit all conditions. Furthermore, the update process would require user intervention, further data gathering and reprogramming the parameters. This is precisely what we want to avoid in our case, and one of the strengths of using a dynamically adaptive online learning algorithm.

STLE is a short term link estimator designed to predict the whether a link is in a high quality state. It is based on the heuristic that if there are three consecutive successful packet receptions, the link is in a high quality period, and if there is one packet loss, the link is no longer in high quality. Here, we implement

the STLE scheme and use the output as the prediction of future link quality in the next 1 second.

We also use a batch trained logistics regression model in the evaluation to study the performance of the same prediction model without the online learning algorithm. Please note that the batch trained LR model was intentionally *over-fitted* to the specific link to maximize the accuracy of the batch trained model, i.e., the LR model is first trained with all the packet reception information from each individual link, and then the trained model is applied to the same link for link quality prediction. This intentional over-fitting is to guarantee the best performance that can be achieved by the batch trained LR model as the model is trained and tested against the same data.

As to the prediction models, the input of the prediction models is comprised of PRR and LQI values from historical packets ($Input_i = [PRR_i, LQI_i]$). PRR is computed from the latest WMEWMA output:

$$PRR_i = \frac{1}{ETX_i}$$

As mentioned before, the parameters of WMEWMA are based on the default values used by 4Bit in TinyOS 2. In other words, for each input vector, PRR_i is always the last WMEWMA estimation and is updated every 5 packets received, whereas the LQI is updated for every packet received. The model computes the prediction for each input vector and runs the learning algorithms (ALAP/s-ALAP) for every packet received.

There are three parameters that affect the input of the model: the window size W, the inter-packet interval I, and the physical parameter used in the input. As pointed out by in Section 4.1, a window size of 1 ($W = 1$) is enough for the LR based models, and including which physical parameters does not significantly

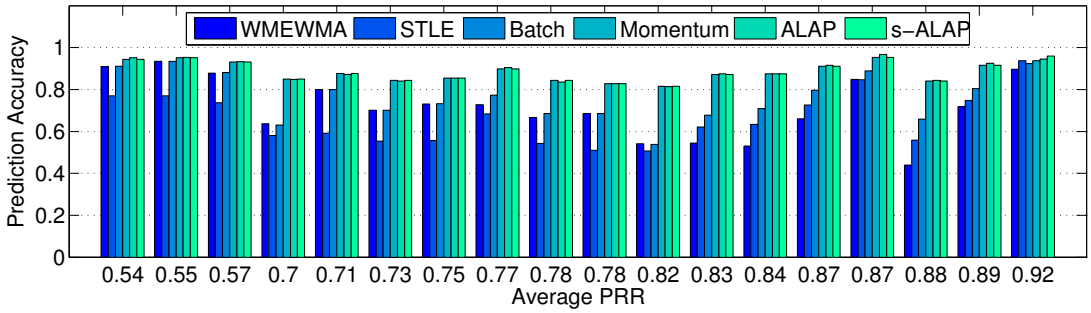


Figure 5.2: The prediction accuracy of LR models using three online learning algorithms, as well as a batch trained LR model, STLE and the WMEWMA estimator.

affect the prediction accuracy. In this section, we only present the prediction performance of the prediction models using PRR and LQI with $W = 1$, and leave the evaluation of the impact of using different parameter sets in Section 5.1.5.

5.1.4.2 Overall Prediction Accuracy

Fig. 5.2 shows the prediction accuracy of the 6 link estimators with respect to the link PRR. The x-axis denotes the average PRR of each link, whereas the bars represents the prediction accuracy of the link estimators respectively. From this figure, it is clear that for the links with average PRR in 0.8 range, the WMEWMA estimator performs the worst among other estimators, which verifies that the WMEWMA estimator could not accurately estimate the link quality variations in the short term. The prediction accuracy of STLE is generally better than WMEWMA for the links with PRR higher than 0.8, but it performs worse than WMEWMA for the links with PRR ranging between 0.5 to 0.8.

The batch trained model performs similarly to WMEWMA for the links with lower PRR but consistently better than both WMEWMA and STLE for the links

with PRR above 0.82, but its prediction accuracy is still consistently worse than any of the three online learning algorithms. This result implies that the best model for link quality prediction gradually changes over time due to frequent network dynamics: even if the LR model converges at a global optimal, the non-stationary wireless environments could soon make the static model obsolete. The online learning algorithms are ideal for tracking such non-stationary environments. In fact, all three methods achieved similar prediction accuracy with s-ALAP slightly better than the other two. Moreover, comparing with momentum, s-ALAP can select the learning rate adaptively, and consequently eliminates the need of selecting a global learning rate and momentum term. Therefore, we choose s-ALAP as the learning algorithm for TALENT. In the following section, we will look further in the results and try to understand the cause of the performance difference.

5.1.4.3 Detailed Results – WMEWMA and Batch Trained LR Model

To further analyze the potential gain of using s-ALAP, we plot the detailed prediction results of s-ALAP, batch trained LR model and the WMEWMA estimator in Fig. 5.3. In this figure, each prediction is classified into one of the four categories: True Positives (TP), which means both the model output and the actual PRR is high; True Negatives (TN), meaning the output and the target PRR is both low; False Positives (FP), indicating the output are high whereas the actual target PRR is low; and False Negatives (FN), which means the output is low whereas the actual target is high. Then, the prediction accuracy is calculated as the ratio of TPs and TNs over all the four classes. The four categories are marked with different colors in Fig. 5.3, and the prediction accuracy of the link are labeled on the top of each graph respectively.

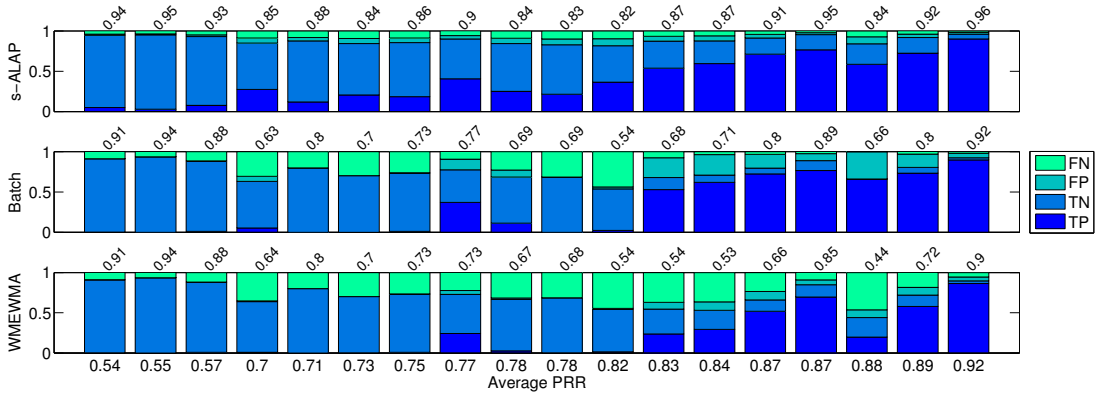
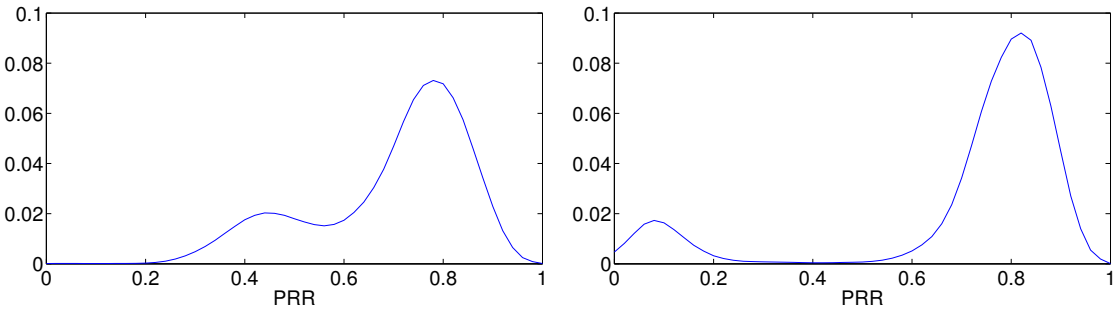


Figure 5.3: Prediction accuracy of three prediction methods applied to 19 intermediate links. The numbers on top of each graph shows the prediction accuracy, and the labels on the bottom indicate the link PRR. The detailed results (TP, TN, FP, FN) are marked with different colors for each link.

From the figure, it is obvious that WMEWMA performs the worse than s-ALAP due to the large FN (light green, top of each bar). Specifically, for links with 0.7 to 0.9 of PRR, WMEWMA shows significantly more FNs than s-ALAP, indicating that a WMEWMA based link estimator will likely overlook the short temporal high quality periods in these intermediate links, whereas a link quality predictor using s-ALAP will have much higher probability to capture such high quality periods. Similarly, in the case of the batch trained LR model, the prediction accuracy of the statically trained LR model is lower than using s-ALAP, suggesting that having a rigid model can not adjust to the changing network conditions. The size of the area for the TP cases (dark blue, bottom) indicates *the percentage of time an intermediate quality link could be used to forward packets with low losses*. TALENT is capable of detecting high quality periods for intermediate links and provides more viable paths for the routing protocol than WMEWMA and the batched trained model.



(a) Average PRR = 0.7 (b) Average PRR = 0.84

Figure 5.4: PRR distribution of two intermediate links.

In order to further justify the advantage of using s-ALAP in intermediate links, we plot the distribution of short term PRR computed with 1 second window of two links in Figure 5.4. Although these two links have different long term quality (average PRR equals to 0.7 and 0.84), both of them show two distinctive peaks centered in different PRRs. This bimodal distribution of short term link quality typically presents in bursty links [SKA,SDT10], where short burst of high quality periods are interleaved with periods of low quality. By leveraging the online learning techniques, TALENT can adapt quickly to the link quality changes and identify the future high quality periods better than the other schemes.

Note that WMEWMA is designed to estimate the average link quality, whereas the model based estimators are designed and trained to predict the probability of the temporal link quality being high during time t in the future (in our case, $PRR_t > 0.9$). When the average PRR of the link is close but below the 0.9 threshold, WMEWMA tends to make many FN mistakes because it converges to the average link PRR that is below the threshold. Hence, WMEWMA fails to predict short intervals of time when the short term PRR is high, a very common event for a link with average PRR in the high 0.8 range. When the average link PRR is very different from the threshold (e.g., in the cases of links in the 0.5 range),

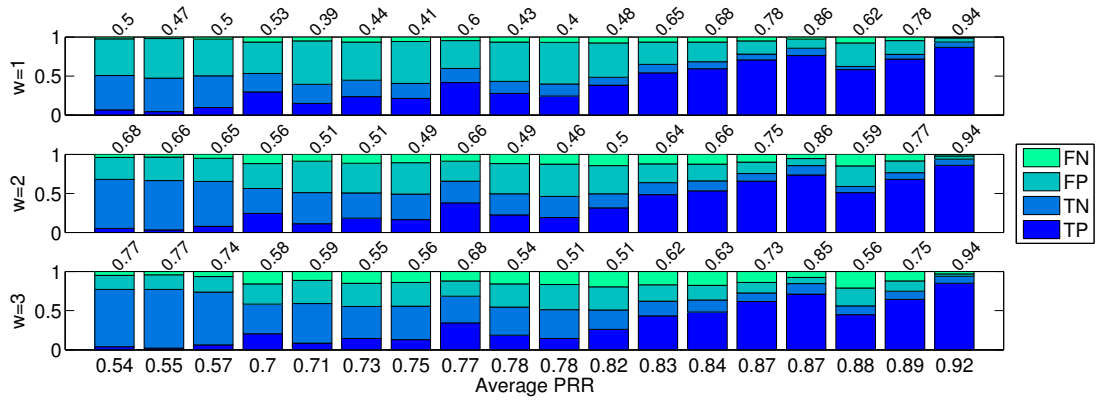


Figure 5.5: Prediction accuracy of STLE with different window applied to the intermediate links. The labels on the bottom indicate the link PRR, and the detailed results are marked with different color. The labels on the left of each plot represent the window size, i.e., how many consecutive packets are needed before STLE considers a link high quality.

or above the threshold (e.g. the 0.92 link) WMEWMA tends to perform much better. This is because the likelihood of a link in the mid 0.5 range to have short bursts of very high quality above the 0.9 threshold is significantly smaller, and therefore WMEWMA tends to correctly predict a failure. This behavior can be seen clearly in Fig. 5.3: when the link PRR is close to 0.5, WMEWMA achieves high prediction accuracy only because of the high number of TNs. When the average PRR of the link is above the threshold (like the 0.92 link), then all estimators work fine. Thus, it is clear that WMEWMA is not suitable for estimating temporal high quality periods for intermediate links.

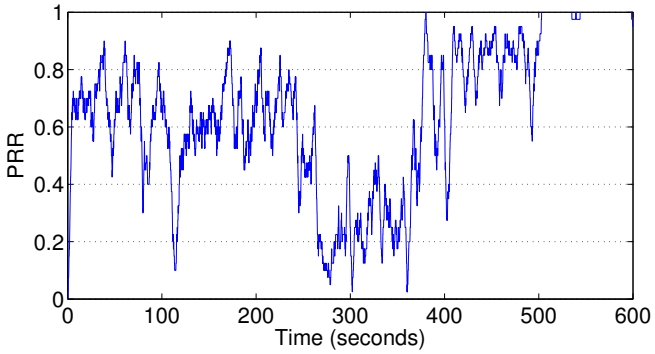
5.1.4.4 Detailed Results – STLE

STLE is a short term link estimator that considers a link high quality (future PRR in one second higher than 0.9) if there are three consecutive packet reception. In

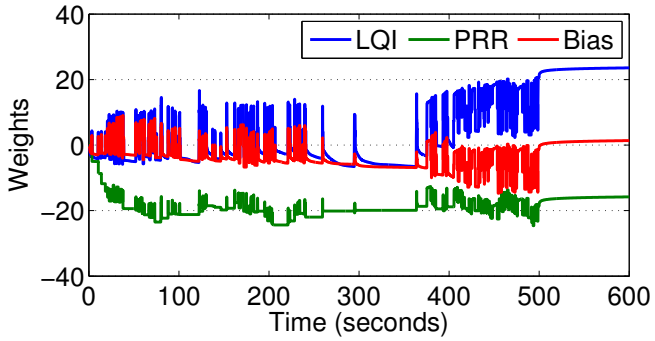
other word, the underlying assumption behind STLE is that the short term link quality can be predicted by counting the number of packets received without loss. If the number of consecutive reception exceed a certain window (three in this case), the link is regarded as in high quality period. However, as shown in Fig. 5.3, STLE performs worse than the online models, and sometimes even worse than WMEWMA. To understand the performance results and to evaluate the effectiveness of STLE assumption, we apply STLE with varying window sizes and plot the detailed results in Fig. 5.5.

Similar to the previous figure, Fig. 5.5 categorizes the results into four categories: FN, FP, TN and TP. The labels in the x-axis represent the link PRR, whereas the labels on the left of each plot denote the window size of STLE. For example, when $w = 1$, STLE will mark a link of high quality as long as the previous packet is successfully received, whereas $w = 3$ (default value in [ALB09]) means that STLE considers the link quality high when three packets are received consecutively. The labels on top of each plot represent the prediction accuracy.

From Fig. 5.5, we see that regardless of the window size, a major portion of the error comes from FP, which means the STLE output often falsely indicates short term high link quality periods. This result suggests that the assumption of STLE is too optimistic: the instantaneous PRR in the next 1 second is often smaller than 0.9 even if the previous three packets were all received. In addition, comparing the results of $w = 1$ (top plot) and $w = 3$ (bottom plot), it can be observed that for links with PRR ranging from 0.5 to 0.8, STLE with a large window size $w = 3$ performs better than using a small window size $w = 1$, whereas the opposite is true for the links with PRR higher than 0.8. The performance difference on the different links highlights a weakness of using a fixed set of parameter in link estimation: even if the parameter is optimally selected for a particular set of links,



(a) PRR Variations



(b) Weights

Figure 5.6: Short term PRR variations of a link with 0.7 average PRR and the corresponding weights of the LR model over time.

it might not work well for other links due to the difference in the link behaviors. On the other hand, the prediction model can use the online learning techniques to adapt to the dynamics of each individual link, and therefore performs better than statically trained models for links with varying qualities.

5.1.4.5 Detailed Results – Weight Update

We further analyze the detailed weight update behavior of the LR model with s-ALAP in order to understand how each parameter in the model input contributes to the PRR prediction. In Figure 5.6, we show the correlation between RPP

variations with respect to the corresponding weight changes of the LR model over time for a typical intermediate link. Figure 5.6(a) presents the short term PRR variations (computed with 1 second interval) of a link with overall PRR of 0.7. The link presented here shows frequent PRR variations during the first 250 seconds, and then the link quality drops below 0.4 in general for about 100 seconds. Around time $T = 400$ seconds, PRR of the link increases again and finally stabilizes to almost 1, indicating a near perfect quality. On the other hand, we apply the online learning LR model with s-ALAP to the same link and plot the weights of model over the same time duration in Figure 5.6(b). The three lines in Figure 5.6(b) are the weights correspond to the three parameters included in the model input, namely, the PHY parameter (LQI in this case), PRR calculated by the WMEWMA estimator, as well as constant bias term with a value of 1.

Comparing Figure 5.6(a) and Figure 5.6(b), we see that during the first 250 seconds, the model updates its weights frequently, indicating that it is actively trying to adapt to the varying link quality. At about time $T = 250$ seconds, the model converges to a stable set of weights as the PRR of the link drops below the 0.9 threshold consistently. Then, as the quality of the link rises again at around $T = 400$ seconds, the model observes more high quality periods and starts to update the weights to reflect the PRR changes. Finally, after $T = 500$ seconds, the link stabilizes to a near perfect quality, and consequently the model also converges to a stable state. The close correlation between the link quality variations and the frequent weight changes suggests that the LR model with online learning actively adapts to the dynamics of the underlying link, resulting in an adaptive and agile estimation of short term link quality.

5.1.5 Parameter Selection

This section compares performance of the prediction models with different sets of parameters to find the the optimal parameter set for TALENT. Based on the problem definition in 5.1.1, there are three tunable modeling parameters: window size W, representing the number of the packets used in a single input, the interval-packet interval I, representing the time interval between the packets, and the physical parameters, indicating which parameter from the physical layer should be used in the input. The following sections will discuss impact of these parameters individually. In addition, we also explore the choice of the decision threshold, i.e., the threshold applied to the continuous output of the LR model to determine the actual classification result (0 or 1), using ROC curves [Bra97].

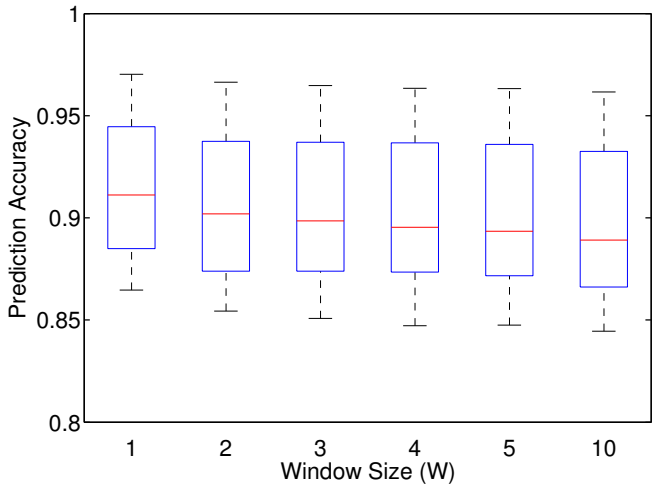


Figure 5.7: Prediction accuracy of models using SGD with s-ALAP as a function of the window size W. X-axis represents the window size, whereas each box plots the distribution of the predication results of the model using W labeled in the x-axis. The inter-packet interval is 0.1 seconds.

5.1.5.1 Window Size

The window size W is the number of packets in each input, which corresponds to the amount of historical information required by the model to predict the future link quality. Fig. 5.7 plots the aggregated prediction accuracy of applying s-ALAP with different W in a box plot. The x-axis denotes W, and each box in the figure shows the median value of the probability distribution of the predication accuracy (red line in the middle), as well as the 5%, 25%, 75% and 95% percentiles of the prediction accuracy of using W labeled in the x-axis. Other box plots in this section are plotted in the similar fashion.

Intuitively, large W means more information will be made available to the model, so it should improve the prediction performance at the cost of more buffering and processing needs. However, Fig. 5.7 shows that the prediction accuracy actually decreases slowly as we increase W from 1 to 10, implying that only the most recent packet is important for the prediction.

This counter-intuitive result highlights the rapid link quality variations in the low power wireless communication. Due to the short coherence time of the wireless channel [Rap01], the historical channel quality may become soon uncorrelated with current quality, and consequently, the packet information can not reflect the link quality at the current moment. In other words, there is no point to look at a long history, when such history becomes soon of no use. Moreover, the slow degradation of the prediction accuracy suggests that incorporating packet information from long past in the model input has an adverse impact on the prediction accuracy: since the past information is almost irreverent to the current link quality, it only adds to the noise in the model input. Therefore, we use window size of 1 ($W = 1$) in our evaluation and implementation of TALENT.

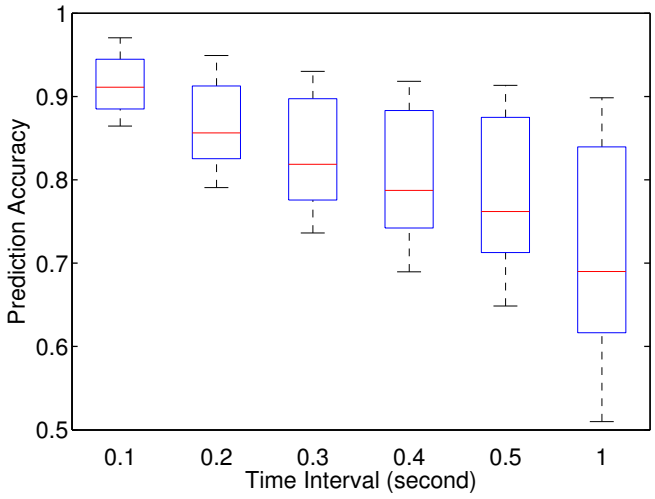


Figure 5.8: Prediction accuracy of models using SGD with s-ALAP as a function of the inter-packet interval I. The window size W is 1.

5.1.5.2 Inter-packet Interval

The inter-packet interval I represents the time interval of the packets used in the input. The longer I, the older is the packet reception information used by the model. Therefore, the prediction accuracy should degrade as I increases due to the short coherence time of the wireless channel discussed in the previous section. Fig. 5.8 illustrates the distribution of the prediction accuracy of SGD with s-ALAP with respect to different inter-packet interval. The window size W is 1 in this figure.

As predicted, Fig. 5.8 shows that the median prediction accuracy drops from above 0.9 to below 0.7 as I increases from 0.1 seconds to 1 seconds, suggesting that the links experience significant temporal dynamics. Please note that although shorter Is give better results, in practice the network traffic is controlled by the upper layer applications and might not show short inter-packet intervals. In this case, the prediction result will not reflect the short term link quality accurately.

This limits the applicability of the prediction model to high data rate traffic only as the prediction is only accurate enough when the packet information is from recent past. Based on the results shown in Fig 5.8, we consider the prediction result is only valid for 1 second.

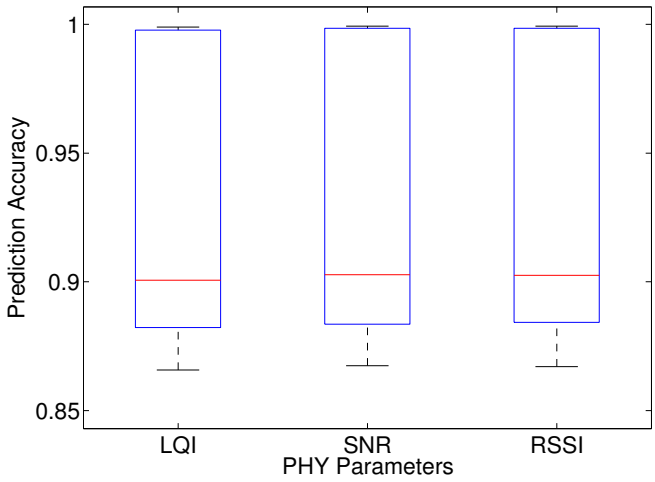
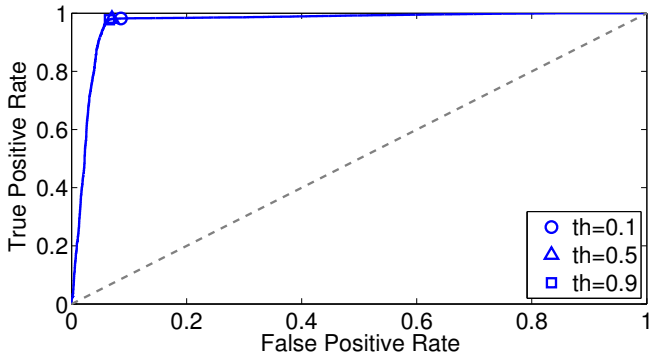


Figure 5.9: Prediction accuracy of SGD/s-ALAP models using varying physical parameters in the input. The window size $W = 1$, and the inter-packet interval $I = 0.1$ seconds.

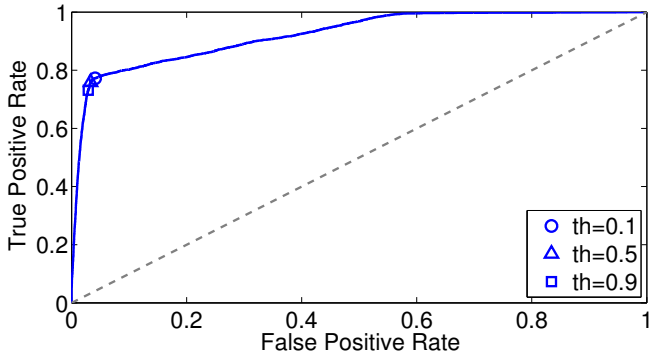
5.1.5.3 Physical Parameters

The choice of physical parameters, namely, RSSI, SNR and LQI, is directly related to the input of the model. To evaluate the effect of using different physical parameters in the input, we apply SGD with s-ALAP on the empirical packet traces and compare the the prediction accuracy of using each parameter. The results are aggregated and plotted in Fig. 5.9.

Fig. 5.9 presents the aggregate prediction accuracy of s-ALAP with different physical parameter in a box plot. The window size W is 1 and the inter-packet interval is 0.1 seconds. From this figure, it is obvious that which physical pa-



(a) Link with 0.7 PRR, $AUC = 0.9223$



(b) Link with 0.57 PRR, $AUC = 0.9685$

Figure 5.10: ROC curves of TALENT with varying decision thresholds.

rameter to use does not really affect the prediction accuracy, which confirms our previous findings in Section 4.1 and [LC11]. In the remainder of this chapter, we show the modeling results of using $[PRR, LQI]$ as the input feature since the LQI value can be easily obtained in the CC2420 platform.

5.1.5.4 Decision Threshold

Another important parameter for the LR classifier is the decision threshold. In general, binary classification models that produce continuous output (e.g., estimation of class membership probability of a given input) need to use a predefined decision threshold to predict the actual classification: if the estimated probability

is higher than the threshold, the classification result will be positive (1), whereas if the estimated probability is lower than the threshold, the classification result will be negative (0). In our case, TALENT also needs to select a decision threshold as the output of the underlying LR model is continuous. The analysis of the optimal decision threshold is often visualized by the Receiver Operating Characteristic (ROC) curves [Bra97].

ROC curve is a plot of the true positive rate against the false positive rate of a classifier at various threshold settings. The true positive rate represents the sensitivity of the classifier, i.e., proportion of actual positives correctly identified, whereas the false positive rate is one minus the true negative rate, i.e., the proportion of correctly identified negatives. In the case of TALENT, the LR classifier should achieve a high true positive rate (e.g., correctly predict high quality periods) while maintain a low false positive rate. In Figure 5.10, we plot the ROC curves for two links with PRR equals to 0.7 and 0.57 respectively.

In both Figure 5.10(b) and 5.10(b), we can see that area under the ROC curve (AUC) are higher than 0.9, which indicates a high overall accuracy of the online prediction model. We also note the points where the decision threshold equals to 0.1, 0.5 and 0.9 in the both figures. In Figure 5.10(a), we see that the TALENT models with different decision thresholds perform similarly and all locate in the turning point of the POC curve. This observation indicates that any threshold within 0.1 to 0.9 range will yield a good prediction model with high sensitivity (true positive rate) and low false positive rate. Figure 5.10(b) shows similar results. Overall, we select 0.5 as the decision threshold for the LR models in this chapter.

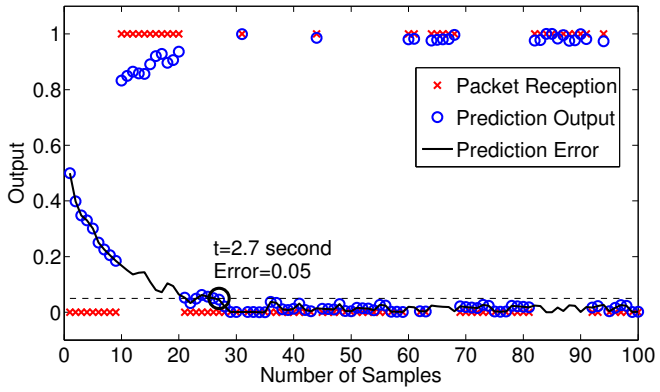


Figure 5.11: Prediction error with respect to time.

5.1.6 Convergence Speed

We now investigate SGD with s-ALAP in terms of convergence speed. In the case of adapting to link dynamics, we want the model to adapt to the new distribution with as few new samples as possible, i.e., update to the new weights with only a few packets. Fig. 5.11 shows how the prediction error evolves during the first 10 seconds when s-ALAP is applied to a bursty link with 54% PRR. The crosses in the figure denote the packet reception (0 means lost, 1 means received), and the solid line represents the prediction error, i.e., the absolute difference between the actual packet reception (1/0) and the model output. As seen in Fig. 5.11, in the first 2 seconds right after initializing the node, the prediction error starts from 50% and quickly declines as the s-ALAP algorithm updates its weight for the link. By the time $t = 2.1$ seconds, the prediction error already drops below 5%. After 2.7 seconds, the error stabilizes and never exceeds the 5% mark. Given the 0.1 seconds inter-packet interval, the SGD model with s-ALAP took about 20 – 30 samples (2 to 3 seconds) to converge. This number is quite consistent for all the 18 packet traces: on average, SGD with s-ALAP needs 25 ± 4 packets to reduce prediction error to below 5%.

5.1.7 Summary

To summarize, we propose to use machine learning methods to build models that can predict the short temporal high link quality periods for intermediate links. Through analysis with empirical data traces, we showed that LR based models using PRR and LQI values can predict the instantaneous PRR in the near future significantly better than WMEWMA for intermediate links. Moreover, by using the SGD online learning algorithm and the s-ALAP learning rate adaption method, the model is able to adapt to individual links and network dynamics in around 2 to 3 seconds without prior data collection or training. This adaptive behavior is a major advantage over 4C, which also uses LR based prediction model that requires a priori off-line training but does not adapt to network dynamics well.

With these encouraging results, we show in the following section how we implement the LR based model with s-ALAP in TALENT to supplement the existing link estimators in TinyOS.

5.2 System Design

In this section, we first present the overall design of the TALENT, and then describe the implementation of the predictor in detail, as well as the integration of TALENT to the existing network stack.

5.2.1 Overview

TALENT is implemented as an extension module of CTP, the default collection protocol in TinyOS. Our design is receiver-initiated: the prediction model in TALENT works in an overhearing node and notifies the sender if a better path is

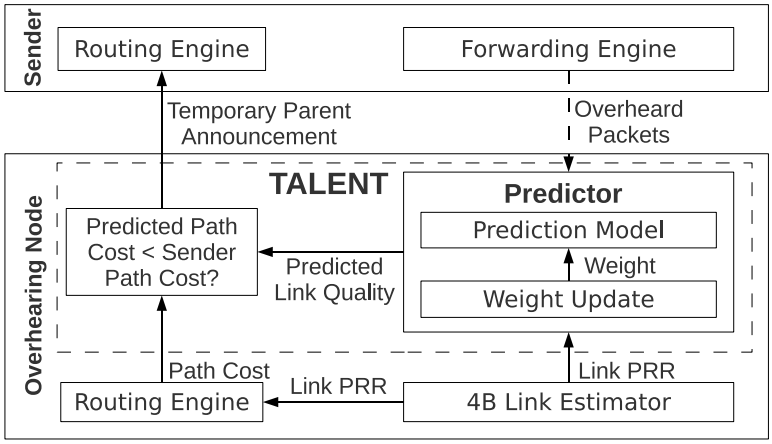


Figure 5.12: Overall Design of TALENT

available. As illustrated in Fig. 5.12, the predictor takes the LQI of the overheard packet, combines it with the link PRR given by the 4Bit link estimator to predict the future link quality. The predicted link quality is then added to the node's routing cost to compute the expected path cost if the packet was to be forwarded by the overhearing node. If the expected cost is smaller than the cost of current forwarding path, the overhearing node notifies the sender about the availability of the new path, announcing itself as a new temporary parent (TP). On receiving the TP notification, the sender uses the TP as the next hop until the TP notifies the sender again to denounce its TP status. The sender reverts back to use the original next hop when the TP denouncement packet is received, or when a number of packets are lost consecutively.

The receiver-initiated design is similar to 4C and STLE [ALB09] since all of them allow the overhearing nodes to become temporary parents. However, there are multiple differences between TALENT and these two link estimators.

The main difference between TALENT and STLE is the prediction model. STLE is based on the heuristic that three consecutive packet receptions signify

a high link quality period, whereas TALENT utilizes machine learning methods to model the link characteristics without assuming prior heuristics. Moreover, due to the adaptive online learning algorithm, TALENT will be able to adapt to network conditions when the STLE heuristic does not apply.

In the case of 4C and TALENT, although both schemes are aimed for reducing delivery cost by using LR models for link quality prediction, but one major improvement of TALENT over 4C is the adaptive online algorithm: TALENT can adapt to network dynamics quickly without any overhead of data collection and model training, whereas 4C needs off-line training to tune the model parameters. This advantage is *significant* from a practical point of view. There is no need for a priori data collection (with the associated costs) for training, nor re-collection for new training data if network conditions change. There is also no need to send the newly updated parameters to reprogram the nodes in this case. Moreover, 4C attempts to predict the success probability of the next packet, whereas TALENT predicts the probability of high quality periods.

5.2.2 Temporary Parent Announcement

In CTP, each node is assigned with a routing gradient that represents the number of transmissions needed to deliver a packet from this node to the root node. The root node normally acts as a base-station and has a gradient of 0. A non-root node selects its next hop, or parent based on the path cost, which is calculated by adding the gradient of a neighbor node and the link ETX, i.e., the number of transmissions needed to send a packet from the node to the neighbor. The node selects the neighbor node (among other neighbor nodes) with the least path cost as its parent and will send its packet to the parent only. This scheme is sender-initiated as it is the sender who picks its parent.

In our scheme, we take the receiver-initiated approach where the nodes on the receiver side compute the path cost for the sender and notify the sender if better paths are available. An overhearing node can snoop on the traffic of a neighboring sender, and the predictor in TALENT will try to predict the link quality between the sender and the overhearing node. By adding the predicted link quality to the routing gradient of the overhearing node itself, the overhearing node can compute the path cost if the sender were to route its packets to it. Then, if the predicted path cost is smaller than the sender's current parent, the overhearing node sends a notification to the sender to be a potential temporary parent.

Specifically, assume a sender S sends packets to its parent P with a gradient of C_P while an overhearing node O is snooping on the channel, whose routing gradient is C_O. The path cost of forwarding through P is $C_{S\to P} + C_P$, where $C_{S\to P}$ is the link cost between S and P. Similarly, the path cost of using O as the forwarder is $C_{S\to O} + C_O$. The sender S selected P as its parent, therefore $C_{S\to O} + C_O > C_{S\to P} + C_P$.

The link cost $C_{S\to O}$ is estimated by the underlying link estimator (4Bit) by exchanging beacon packets. In parallel, the predictor in TALENT continuously predicts the link quality with the PHY parameters from the overheard data packets from S. If the prediction output is greater than 0.5, TALENT considers the link $S \to O$ in a high quality period, and overrides the $C_{S\to O}$ with the minimum value of 1. In this case, if the new path cost of using O is smaller than the path cost of using P, then S should use O as a temporary parent. In other words, if the predictor indicates the link $S \to O$ is in a high quality period, and the following formula holds:

$$C_{S\to O} + 1 < C_{S\to P} + C_P$$

then the overhearing node O will send a notification to S to announce itself as

the temporary parent. On the other hand, if this formula does not hold due to change of the prediction output or the routing gradients, O will again send a notification to S to denounce the temporary parent status.

Algorithm 1: s-ALAP Weight Update Rule

Require: Input $x_j(t) = [PRR(t), LQI(t)]$, output $y(t)$, instantaneous PRR $PRR_{Inst}(t)$ and meta learning rate q

Ensure: Updated weights $W(t)$ and learning rate $\lambda(t)$

if $PRR_{Inst}(t) > 0.9$ **then**
  $target(t) \leftarrow 1$
else
  $target(t) \leftarrow 0$
end if
for $j = 1 : D$ **do**
  $gradient_j(t) \leftarrow (target(t) - y(t))x_j(t)$
  $\Delta W_j(t-1) \leftarrow \Delta W_j(t)$
  $\Delta W_j(t) \leftarrow W_j(t) gradient_j(t)$
  $\overline{\Delta W_j^2}(t) \leftarrow 0.8\overline{\Delta W_j^2}(t) + 0.2\Delta W_j^2(t)$
  $\lambda_j(t) \leftarrow \lambda_j(t) \max\left(0.5, 1 + q\Delta W_j(t)\frac{\Delta W_j(t-1)}{\overline{\Delta W_j^2}(t)}\right)$
  $W_j(t) \leftarrow W_j(t) + \lambda_j()\Delta W_j(t)$
end for
return $W(t)$, $\lambda(t)$

5.2.3 Predictor Implementation

The predictor component (see Fig. 5.12) implements the prediction model and performs weight update for the model. Overall, the LR based prediction model

computes a new link quality prediction for every overheard/received data packet, and overhearing nodes use the predicted link quality to calculate the routing cost in the temporary parent announcement process discussed in the previous section. The predictor also updates the weight of the prediction model for every 10 packets overheard/received from the same sender.

More specifically, the prediction model functions as an extension to the existing link estimator 4Bit. Whenever a new data packet is overheard or received, the predictor takes the LQI value of the packet, combined with the current estimated PRR of the link between the sender and the recipient node to create an input for the LR based prediction model. The PRR is estimated by 4Bit, which employs WMEWMA with the same parameters used in the performance comparison presented in Section 3. The prediction model outputs the predicted link quality from the sender to the recipient node for the next 1 second, which is then used to calculate the routing cost for the temporary parent announcement. In order to reduce the computation time for the prediction, we employ a linear approximation proposed by H. Amin et al. [ACH97] to accelerate the sigmoid function calculation required by the model. The measured execution time of prediction computation in TMote is 0.5 ± 0.004 ms.

The predictor also performs weight updates for the prediction model. Once a prediction is calculated, the predictor records the prediction output as well as the inputs, and then starts to measure the instantaneous PRR as the target of this prediction. Because all the packets are embedded with a monotonically increasing sequence number, the exact number of packets sent by a sender can be inferred by counting the gap between the sequence numbers of received packets. The instantaneous PRR is then calculated by dividing the number of packets overheard with the packet gap. In our implementation, we calculate the instan-

taneous PRR once the accumulated packet gap equals to or is greater than 10 such that the instantaneous PRR can reflect the real-time link quality after previous prediction. Once the instantaneous PRR is available, it is then used in the weight update along with the corresponding predictor input and output. The algorithm of the weight update is listed in Algorithm 1, which implements the Equation (5.1) and (5.2).

The implementation of the s-ALAP algorithm takes several measures to minimize the execution time. First, it uses integer numbers instead of floating points by scaling decimal values up to avoid floating point calculation. Second, the implementation avoids multiplications and divisions as much as possible by replacing them with bit shift operations. Also, it only checks integer overflow when necessary, i.e., only check those operations that might involve large numbers. The measured execution time of a single weight update is 2.31 ± 0.19 ms in TMote. Considering the usual packet interval of sensor networks is at least in the order of 100 ms, this execution time should not hamper the normal operation of the node. More importantly, we perform the s-ALAP weight update only once every 10 packets as discussed in the next section.

5.2.4 Integration to Existing Network Stack

Overall, TALENT is implemented as an extension to the existing link estimator. As shown in Fig. 5.12, TALENT communicates with almost all the components of the network stack, including the 4Bit link estimator, the routing engine and the forwarding engine. We carefully designed TALENT such that it does not interfere the normal operation of other components; furthermore, we made some modifications to 4Bit to take full advantage of the overheard packets.

The first problem of TALENT and CTP integration is routing stability. CTP

establishes a routing gradient using the path cost. When a node changes its parent, CTP updates the routing gradient with beacon packets. During the routing gradient update, TALENT should not send any TP notification as the path cost itself is not stabilized. Therefore, we added a counter to suppress the TP announcements when a parent change is detected. Moreover, to avoid two or more overhearing nodes trying to become TP of the same sender, the same counter is used to prevent such racing conditions.

A common problem is how to deal with broken links. If the link quality between the sender and the TP suddenly drops, the TP can not notify the sender even if it realizes the quality drop as the notification packet may get lost, and the sender will attempt to retransmit its packets until the route update mechanism of CTP kicks in and changes the parent node. To prevent this situation, we set a TP loss threshold that limits the maximum number of consecutive packet losses when a TP is set. In our implementation, the sender will switch back to the old parent after 5 consecutive packets losses instead of relying on the slow CTP route update. For all the switch back to old parent events seen in our experiments, only 12% of the cases were due to the loss threshold, whereas 54% and 34% of the cases were due to denounce TP notifications and CTP parent changes respectively.

An important design decision is when and how to preform weight updates due to the short effective period of the prediction. Conceptually, after the prediction model is updated, the prediction output is only valid for 1 second before network dynamics render the prediction inaccurate. In our design, TALENT performs weight updates once every 10 packets and uses a timer to keep track of the most current update. Any prediction output is marked invalid if the prediction is made after 1 second of the previous weight update. The intuition here is that when the traffic rate is high (e.g., inter-packet interval is 0.1 seconds), packets arrive

at a fast rate so that the prediction model can be updated frequently and the output will be mostly valid, whereas in the low traffic rate cases, the timer will prevent the use of out-of-date predictions as the model will be updated less often. When a temporary parent has obsolete predictions, we take an opportunistic approach that allows the sender to continue sending packets to the temporary parent without notifying the expiration of the prediction. Due the the presence of the TP loss threshold, the sender realizes of any potential link quality degradation and switches back to the old parent quickly.

Another subtle issue is how to perform the weight update on big packet losses. Large packet losses leave a big gap in packet reception, which translates to a long trace of lost packets that requires multiple weight updates. To avoid unnecessary weight updates and computational stress on the mote, we limit the number of weight updates caused by large packet gaps to 5, such that the weight update operations do not hamper normal operations of the mote.

5.2.5 Low Power Listening

For energy constrained sensor networks, Low Power Listening [PHC04] (LPL) is an important component that conserves energy by duty-cycling the radio. LPL periodically wakes up the radio to perform clear channel assessment (CCA) and turns off the radio if there is no activity detected. If there is activity in the channel, LPL keeps the radio on so that the MAC protocol can receive the potential packet. Once the packet is received, the MAC protocol will signal the receive event to upper layers, and LPL will put the radio back to sleep after a short wait period.

TALENT is implemented on top of BoX-MAC [ML08], the default MAC protocol of CC2420 radio in TinyOS which supports overhearing. In BoX-MAC,

Operation	Execution Time (ms)
Prediction Calculation	0.5 ± 0.004
Weight Update	2.31 ± 0.19
Packet Transmission (30 bytes)	5.25 ± 1.125

Table 5.1: Typical execution time of TALENT.

LPL wakes up the radio purely based on the periodical CCAs, and therefore overhearing-based operations are perfectly functional with LPL. An overhearing node can wake up for channel activities to snoop for packets just like it were to receive the packets. Furthermore, since the CCA in BoX-MAC does not perform any address check, even a non-overhearing node will have to wake up and receive the packet being transmitted when channel activities are detected. It is the upper network layer's job to decide whether the packet is addressed to the node itself. In this sense, the energy overhead of overhearing nodes is only caused by the processing of overheard packets compared with non-overhearing nodes, and the radio energy consumption is independent of using overhearing. Because of the above reasons, receiver-initiated approaches such as TALENT will work with LPL in BoX- MAC normally without incurring any significant overhead on the energy usage. We believe that TALENT could still work even if the MAC protocol does not directly support overhearing with LPL as discussed in Section 5.4.

The wake up interval is the most important parameter as it controls the frequency of the CCA operation, and therefore decides the duty-cycle rate of the radio. A short interval may increase the duty-cycle rate unnecessarily, whereas setting the interval too long may cause packet losses due to queue overflow on the sender nodes. In our experiments, we set the wake up interval to 100 ms to meet the relatively high data rate.

5.2.6 Memory and Computation Overhead

The memory overhead of TALENT is mainly due to the implementation of the receiver-initiated approach, i.e., the temporary parent announcement mechanism discussed in Section 4.2, as well as the the coefficients (weights and learning rates) of the LR prediction model. In our implementation, the ROM size increased by 5269 bytes (from 28416 to 33685 bytes) with the addition of TALENT, whereas the RAM requirement increased by 861 bytes (from 4019 to 4880 bytes). Given the 48 kB flash and 10 kB of RAM in the TMote Sky mote, the added memory footprint should not be a big concern.

In terms of computation overhead, the extra energy consumption of TALENT is mainly due to the prediction model and the temporary parent announcement. As discussed in Section 4.2, a node sends the temporary parent announcement only when the predicted cost gain is greater than the overhead of sending the notification packets. Therefore, here we focus on the additional energy consumption of running the prediction model.

To analyze the energy consumption overhead introduced by the prediction model, we measured the current draw and execution time of the prediction model running on TMote Sky as well as the current draw and packet transmission time of using the radio. The current draw was measured by a multimeter connected to a TMote and a 3 Volt power source in series, while we programmed the TMote running on different conditions, i.e., with the radio on/off and with/without the prediction model continuously running. Table 4.2 lists the current consumption measurements, and Table 5.1 presents the time needed for the prediction and packet transmission operations. Our measurements are consistent with the TMote Sky datasheet [Mot].

Table 4.2 shows that the current consumption of the MCU running the pre-

diction model at the full speed is 2 mA, whereas the radio typically consumes 20 mA when receiving and 19 mA when sending packets. Since transmitting a 30-byte packet and receiving its acknowledgment require about 5.25 milliseconds, the energy spent on transmitting a single packet would be able to support about 50 milliseconds of MCU computation time. As described in Section 4.4, the prediction model in TALENT executes two operations: calculate a new prediction for each data packet receive/overheard, and update the weights every 10 packets received/overheard from a sender node. Given the 0.5 milliseconds execution time for the prediction calculation and 2.31 milliseconds for weight update listed in Table 5.1, TALENT introduces a computation overhead of 7.31 milliseconds for every 10 packets received/overheard, or 0.731 milliseconds per packets on average. Compared with overhead of transmitting a packet, the energy overhead of the TALENT prediction model is only 1.5% of the energy of sending a single packet. Combined with the conservative temporary announcement mechanism, having TALENT is beneficial as long as the prediction model can save one packet every 67 predictions computed, and the temporary parent announcement mechanism will guarantee cost reduction as long as at least one packet is forwarded by the temporary parent. As shown in Section 5.3, the savings provided by TALENT are order of magnitude larger than the minimal requirement discussed here.

5.3 Experimental Evaluation

We evaluate the performance of TALENT in terms of end-to-end delivery cost, loss rate and path length. The delivery cost refers to the total number of transmissions needed to deliver a packet to the root, the loss rate is the percentage of packets sent but never received at the root, and the path length refers to the number of hops in a delivery path. The performance of TALENT is compared against

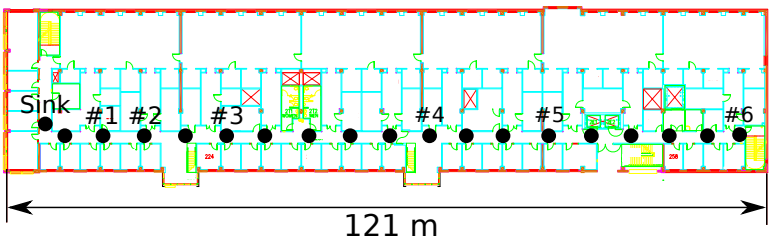


Figure 5.13: Local testbed with 57 Tmote Sky motes placed along an corridor of a typical office building. The motes are divided into 19 groups denoted by the black dots. The distance between each node group ranges from 6 to 7 meters, except for the node group in the far left which sits around a corner at the end the corridor. The root node is denoted as Sink and the sender node in each experiment are denoted as #1 to #6. The number of nodes included in each experiment is 5, 6, 11, 25, 42, and 57 respectively.

three other state-of-the-art link estimators, namely, 4Bit [FGJ07], STLE [ALB09] and 4C. To take full advantage of the snoop interface, we updated the networking stack so it can use the overheard packets to update the ETX estimation. This modification is applied to all receiver-initiated estimators, namely, TALENT, 4C and STLE so there is fair ground for performance comparison.

In the reminder of this section, we first present a detailed analysis about the behavior of link estimators in a typical network setting to motivate the use of short term link estimator. Then we expand our evaluation to multiple experiments in three different testbeds to verify the performance gain of using TALENT. In addition, we also stress test TALENT with variable data rates in congested networks.

Exp. #	Node Pair	Configuration
1	184 → 19	Diagonal
4	140 → 50	Diagonal
2	50 → 19	Horizontal
6	184 → 140	Horizontal
3	140 → 19	Vertical
5	184 → 50	Vertical

Table 5.2: Node pairs in the Motelab experiments.

Exp. #	Node Pair	Configuration
1	112 → 31	Diagonal
2	107 → 19	Diagonal
3	112 → 107	Horizontal
4	31 → 19	Horizontal
5	112 → 19	Vertical
6	107 → 31	Vertical

Table 5.3: Node pairs in the Indriya experiments.

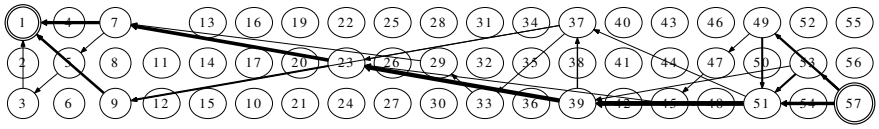


Figure 5.14: A connection map of an experiment with 57 nodes in the local testbed. The width of the lines indicates the percentage of the total data packets being transmitted through the corresponding link. There are 26 unique paths in total and the paths with sparse packet transmissions are not shown.

5.3.1 Experimental Setup

We conducted extensive experiments in three wireless testbeds: a local testbed, the Harvard Motelab [WSW05] testbed, and the Indriya [DCL11] testbed. The local testbed is comprised of 54 Tmotes placed along the corridor of a typical office building. The Motelab testbed is a sensor network testbed composed of 180 Tmotes deployed on three floors. Unfortunately, only 47 nodes were available at the time of our experiments due to node failures. The Indriya testbed consists of 127 TelosB motes deployed across three floors of the School of Computing of the National University of Singapore.

Since TALENT tries to predict the short temporal link quality, our experiments are focused on bursty traffic. We tried different scenarios to test TALENT under different conditions. First, we performed in-depth analysis on an experiment done in the local testbed with 5 nodes to justify the use of short term link quality estimators under bursty traffic. Then, we conducted extensive single sender experiments in the three testbeds with similar experimental settings used by the STLE authors. Third, we tested TALENT under variable sending rates to see the impact on its prediction ability and performance. Finally, we tried multiple sender experiments to stress test the performance of TALENT in con-

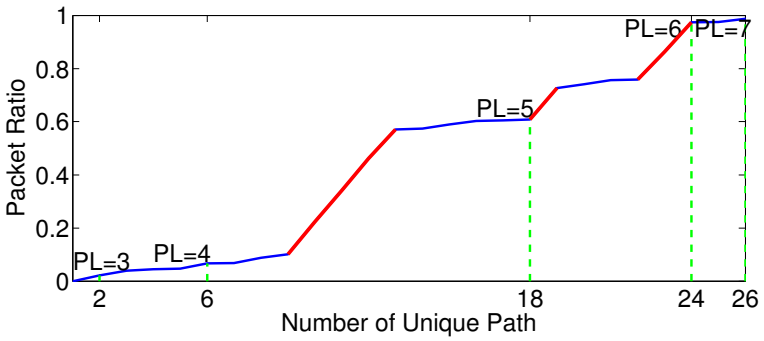


Figure 5.15: The cumulative distribution of the ratio of the packets transmitted through each path with respect to the number of unique paths in the 57 node experiments. 26 unique paths are observed in this experiment, with the path length ranging from 3 to 7 as denoted in the figure. The distribution increases gradually, indicating that there is no single dominate data forwarding path in the local testbed.

gested networks by letting multiple nodes send data packets at high traffic rates simultaneously.

In all experiments, we have LPL active, the sender(s) send 30-byte long data packets with a sending interval of 100 ms to mimic burst data transmissions. When testing variable sending rate, we add 50 ms randomization to the nominal sending rate. For all the local testbed experiments, we set the radio output power level to -25 dBm to increase the network size, and for the Motelab and Indriya testbeds the power level is set to 0 dBm for better connectivity. We perform single sender experiments with little external interference on channel 26, as well as variable rate experiments with interference from 802.11 radios on channel 11. We run more than 80 experiments in all testbeds combined, with each experiment sending 6,000 packets for a total of 480,000 packets sent.

5.3.1.1 Testbed Settings

In order to evaluate the performance of TALENT in a diverse set of network environments, we perform our experiments in the three testbeds with many different settings. For the local testbed experiments, we vary the total number of nodes in the network in order to create different network density for each experiment. Fig. 5.13 illustrates the location of the sender nodes in each experiment, with 5, 6, 11, 25, 42, and 57 nodes included in the respective experiments.

Please note that although the local testbed is deployed along a corridor of a typical office building, the network topology seen in our experiments are not completely linear. Fig. 5.14 illustrates a map of the major data forwarding paths seen in a 57-node experiment. The directional lines indicate the links used to forward the data packets, and the width of the lines indicates the percentage of the total data packets being transmitted through the corresponding link. From this map we can clearly see a large portion of the packets traveled through the path $57 \rightarrow 51 \rightarrow 39 \rightarrow 23 \rightarrow 7 \rightarrow 1$, but there are actually 26 unique forwarding paths observed in total. This observation is further confirmed by Fig. 5.15, which presents the packet distribution with respect to the number of unique forwarding paths. As seen in Fig. 5.15, the major percentage of the total packets were forwarded through a few frequently used paths (marked as red), but a significant proportion of the total packets took other paths with different length ranging from 3 to 7. In general, the cumulative distribution of the packets travel through each unique path increases gradually, indicating that there is no single dominate data forwarding path in the local testbed.

In the Motelab and Indriya experiments, we fixed the total number of nodes in the network, but varied the sender/sink node pair to create a variety of network environments. Following the example node configurations proposed in [ALB09],

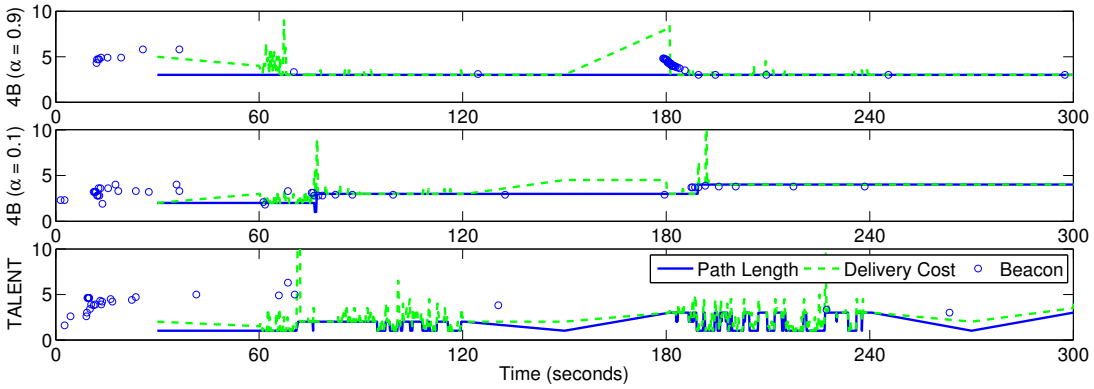


Figure 5.16: The path length and the delivery cost of 4Bit (stable and reactive) and TALENT during a 5 minutes experiment. The beacon packets are notes as dots

we used three types of sender/sink pairs to cover a rich set of geographically different network configurations, namely, vertical, diagonal and horizontal. Vertical configuration means the sender and sink node are on different floors and on the same end; diagonal configuration means the sender and sink are on different floors but on the opposite ends; and in the horizontal configurations the sender and sink are on the same floor and on the opposite ends. For all the Motelab experiments, the total number of nodes used is 47, and Table 5.2 lists the actual node pairs. For the Indriya experiments, the number of nodes is 127, and the the node pairs used are listed in Table 5.3.

5.3.2 Link Estimation with Bursty Traffic

We perform some preliminary analysis to motivate the use of short term link quality estimators under bursty traffic. We consider three link estimator settings: 4Bit with default WMEWMA parameter $\alpha = 0.9$, 4Bit with $\alpha = 0.1$ and TALENT. Intuitively, 4Bit with $\alpha = 0.9$ means the ETX calculation will give

more weight to the historical link quality, making 4Bit insensitive to sparse link quality changes. On the other hand, 4Bit with $\alpha = 0.1$ assigns more weight to the current link quality, and hence increase the reactiveness of 4Bit. In the reminder of this section, we refer to 4Bit with $\alpha = 0.9$ as "stable 4Bit" and 4Bit with $\alpha = 0.1$ as "reactive 4Bit". These two different flavors of 4Bit are compared with TALENT experimentally under a burst traffic pattern to examine the differences in path selection and the end-to-end delivery cost.

Our evaluation employs a simple network consisting of five nodes in a linear topology: a sender S, three forwarders $F1$, $F2$ and $F3$, and a root node R. The links between immediate neighbors, such as $S \rightarrow F1$ and $F1 \rightarrow F2$, are high quality and stable, whereas other links, such as the links between $S \rightarrow F2$ and $S \rightarrow R$, are of various quality with temporal variations. Therefore, there are five possible paths to deliver packets from S to R: a long path using only good links ($S \rightarrow F1 \rightarrow F2 \rightarrow F3 \rightarrow R$), or short paths using the intermediate links, such as $S \rightarrow F1 \rightarrow F2 \rightarrow R$ or $S \rightarrow R$. Short paths have less hops than the longer paths and are preferable to the routing protocol, but the low quality links may offset the advantage due to packet losses. The choice of least costly path is truly determined by the link estimator.

In this network, the sender sends packets in a burst pattern: only one packet is sent every 30 seconds in the first minute, and in the second minute the sender sends 10 packets per second with a 0.1 seconds inter-packet interval. This pattern repeats in the remaining three minutes till the experiment ends at the end of fifth minute. The experiment is first conducted with CTP and stable 4Bit ($\alpha = 0.9$), and then repeated using reactive 4Bit ($\alpha = 0.1$) and TALENT back to back to ensure minimal environmental changes. The behaviors of the three estimators are illustrated in Fig. 5.16 respectively. In each plot, the solid line shows how

the path length evolves over the course experiment, the dashed line indicates the corresponding end-to-end delivery cost, and the circles represent the beacons received by the sender S. For each circle, the x axis notes when the beacon is received and the y axis is the estimated delivery cost of the beacon sender.

Judging from the path length showed in the top plot in Fig. 5.16, it is clear that with stable 4Bit, CTP took the longer path (4 hops) from the beginning and was never stray away from it over the course of the experiment. The merit of this path is that the links are of high quality and stable, thus almost all the send attempts were successful and very little number of retransmissions were required except for a few seconds after 60 seconds. This is reflected by the mostly smooth delivery cost in the plot. Note that due to the stable WMEWMA estimator, the several seconds of high losses are not enough to make the stable 4Bit change its path. In other words, stable 4Bit selected a path with a cost per hop almost equal to 1 and never changed even with packet losses.

Different behavior can be observed from reactive 4Bit in the middle plot of the figure. CTP started off by using a short path (3 hops), but when the data rate was increased to 10 packets per second after time $t = 60$ seconds, reactive 4Bit soon realized that the selected path quality is not perfect due to the high losses and switched to a longer path immediately at around 80 seconds. The time of the switch is truly dependent on the timing of the losses in the experiment, as another switch occurred at around 190 seconds, again due to high losses. In summary, the reactive 4Bit is sensitive to packet losses and changes to longer paths with stable, high quality links almost immediately after experiencing losses.

The situation is quite different from CTP using TALENT. As seen in the bottom plot in Fig. 5.16, the path length is 1 at the beginning, indicating that the shortest path was selected. The cost of delivering a packet fluctuated when the

data rate increased to 10 packets per second after 60 seconds, confirming that the $S \rightarrow R$ link is not stable and has intermediate quality. At around $t = 75$ second, CTP switched to a longer path due to quality degradation on the shortest path. However, TALENT enabled CTP to quickly switch back and fourth between the shortest path and longer path once the instantaneous link quality of $S \rightarrow R$ is high enough. This switching behavior can be clearly observed between 90 and 120 seconds, as well as from 180 seconds to 240 seconds in the experiment. Despite the cost fluctuation associated with the shortest path, the average delivery cost of CTP with TALENT is significantly smaller than both stable 4Bit and reactive 4Bit. In this experiment, the average delivery cost of stable 4Bit is 3.12, reactive 4Bit is 3.47, whereas TALENT is 2.32, 34% smaller than stable 4Bit and 50% smaller than reactive 4Bit.

Why 4Bit does not switch to the shorter paths if they are available? The beacon distributions presented in Fig. 5.16 offer an explanation. The beacon packet contains the link quality and delivery cost estimation from the beacon sender, and the recipient node of the beacon can compute the estimated delivery cost assuming the beacon sender as its parent. According to the CTP adaptive beaconing policy, all nodes send beacon packet frequently to establish the initial link quality estimations and select parents at the beginning of the experiment, but the inter-beacon interval grows exponentially as a stable route is established. This can be observed in all the three plots in Fig. 5.16: the beacon packets received by the sender are clustered within the first 30 seconds of the experiment, whereas only several beacon packets were received in the remaining time. Once CTP switched from a short path to a longer path due to link quality degradation, the ETX estimation of the old parent only reflects the bad link quality that caused the switch. For CTP to use the shorter path again, the previous bad quality estimation must be updated to reflect the current path quality. Unfortunately,

the sparse beacon packets could not adjust the estimation due to the EWMA filter fast enough even for the reactive 4Bit. In this case, the skewed ETX estimation will prevent CTP from utilizing the shorter path over the course of the experiment. On the other hand, TALENT constantly updates its link quality estimation by using the overheard data packets and follows closely with the actual link quality. Even though the ETX estimation was skewed by the lossy periods of the link for several times, CTP was able to switch back to the shorter path as soon as TALENT indicates a high quality period is available on the shorter path. This observation is based on a simple linear network, but it is also applicable to larger networks. In a dense network, CTP may have more links to choose from and may find alternative routes with small end-to-end costs. However, a dense network also means that the number of potential temporary parents is large, and this allows TALENT to find shorter routes as well. The relative savings of TALENT vs 4Bit across different network densities remain relatively constant as shown in the following section.

This result highlights the caveat of using only cost based estimators. For a cost based estimator such as 4Bit, the ETX of a link is evaluated based on reception of beacon packets provided that the link is not part of the forwarding path. Meanwhile, CTP adapts the adaptive beacon policy, which increases the beacon packet interval exponentially when the route is stable. The problem arises when CTP finds a stable route, the ETX of this link will be updated less often due to the increased beacon interval. Consequently, the ETX estimation of an intermediate link could be easily skewed by short temporal quality degradation, and it will take a long time for the ETX estimation to converge to the average link quality. The combination of all these factors effectively prevents CTP with 4Bit from utilizing intermediate links even if they exhibit frequent high quality periods. Moreover, with reactive 4Bit, CTP switches to longer paths at the first

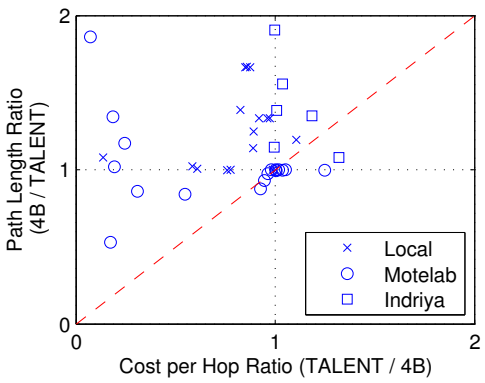
hint of packet losses, making it even less efficient than stable 4Bit.

While in this section we have presented some small quantitative and qualitative analysis and explanations of the different link estimators' behaviors, it is not sufficient to understand their performance under a wider range of network conditions. In the following sections, we present some more detail performance analysis and experiments for different levels of network dynamics and environments.

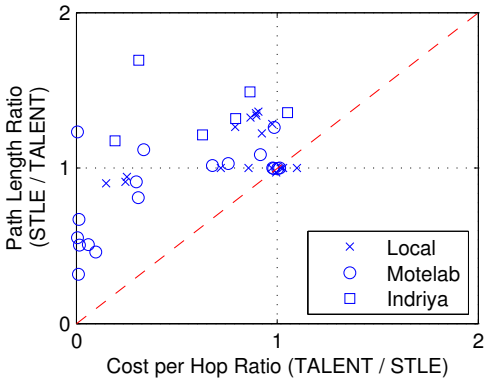
5.3.3 Path Length vs. Delivery Cost

The above experiment shows a simple but illustrative example of why CTP with 4Bit may prefer a longer and more stable path, while in many cases in practice a shorter path with a dynamic intermediate link might be better (less costly). We argue that the intermediate links are underutilized with cost based link estimators such as 4Bit, and using TALENT would enable the routing protocol to actively select the shorter paths with more intermediate links. Although the cost per hop may be higher for intermediate quality links compared with high quality links, the end-to-end delivery cost is reduced ultimately due to the lower number of hops.

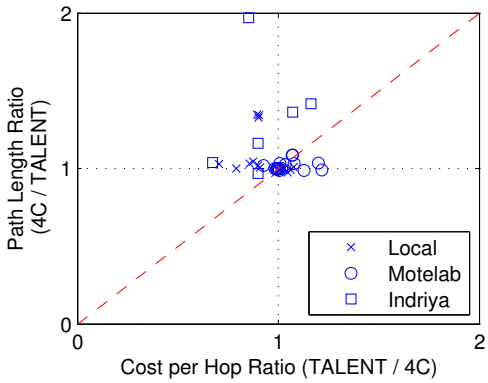
To study the trade off between a longer path with stable high quality links and a shorter path with unstable intermediate quality in larger networks, we extend our evaluation by comparing the behavior of TALENT with respect to 4Bit, STLE and 4C in extensive single sender experiments conducted in three wireless testbeds: the local, Motelab and Indriya testbeds. The experimental conditions were described in Section 5.3.1in detail. Each experiment was repeated three times with the same network settings, i.e., packet length, radio power level and node configuration in the network. The 4C modeling parameters were assigned based on LR model trained on training sets collected at each testbed, and this



(a) TALENT vs 4Bit



(b) TALENT vs STLE



(c) TALENT vs 4C

Figure 5.17: The average delivery cost per hop ratio of TALENT and (4Bit/STLE/4C) versus the path length ratio (4Bit/STLE/4C) and TALENT in all the single sender experiments. The marks above the line $x - y = 0$ indicate better overall costs of TALENT over the other link estimators.

training cost is not included.

To make our point clearer, we break down the end-to-end delivery cost into the hop count times the cost per hop. Fig. 5.17 shows the average delivery cost *per hop* ratio of TALENT and 4Bit/STLE/4C versus the path length ratio of 4Bit/STLE/4C and TALENT for the experiments conducted in the three testbeds. In general, if the rate of cost per hop increase is smaller than the rate of path length decrease, the overall cost for delivering packets is reduced. For example, if $\frac{CostPerHop(TALENT)}{CostPerHop(4Bit)} < \frac{PathLength(4Bit)}{PathLength(TALENT)}$, TALENT achieves lower cost than 4Bit. Therefore, any point plotted above the line $x = y$ indicates TALENT has lower delivery cost 4Bit in one experiment, and vice-versa.

As shown in Fig. 5.17, the majority of the experiment results were marked above the $x = y$ line, indicating that the overall delivery cost of CTP with TALENT is better than the other estimators. For the local testbed and the Indriya testbed experiments, a large group of the results exhibits small or even negative cost increment while using shorter paths, implying that using TALENT can reduce the path length while maintaining the delivery cost. There are also cases where the path length of TALENT and the other estimators are the same but the cost of TALENT is much smaller (see Fig. 5.17(a) and 5.17(b)). This is due to a poorly connected network and/or sudden link quality changes that causes 4Bit and STLE to send excessive retransmissions before switching to another path. In this case, TALENT enables fast route updates with such network dynamics by taking advantage of the temporary parent mechanism: the overhearing node can notify the sender about the alternative routes even if the ETX estimation in the sender side is lagging behind the link quality changes.

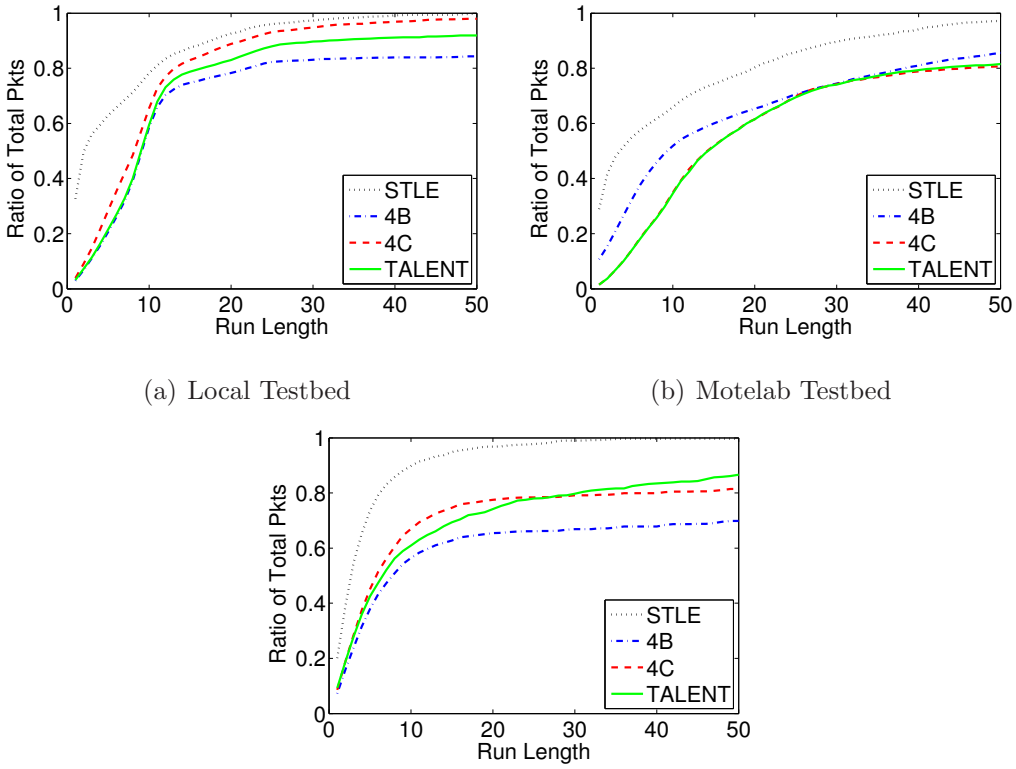
For the Motelab experiments, this behavior is more obvious. Originally, the Motelab testbed had 180 nodes, but the number of nodes has been reduced to 47

at the time we conducted our experiments. Due to the sparsely connected network, the number of possible routes are limited and the use of intermediate links is almost unavoidable in some cases to avoid network partitions. From Fig. 5.17(a) and 5.17(c), it can be observed that many of the Motelab experiments for 4Bit and 4C are clustered around (1,1), indicating that both TALENT and 4Bit/4C took similar paths. However, several experiments show drastic cost reduction of TALENT over 4Bit and STLE particularly when the path length ratio does not show particular trend. This is because the crucial links in the forwarding path were suddenly broken, in which case the 4Bit and STLE estimators could not find an alternative path fast enough. As a consequence of the network partition, the data packets were accumulated in the forwarding nodes and eventually dropped before a new route is established, causing low end-to-end delivery rate in addition to high delivery cost. On the other hand, TALENT can recover quickly from such dynamics due to the fast adapting predictor and the receiver-initiated approach of temporary parent selection.

While the overall performance of TALENT is still better than 4C on average, the improvement is smaller when compared with the previous two estimators. This can be seen by the smaller distance from all the points towards the identity line. It should be noted that 4C was extensively trained using a priori collected training data for each testbed. This additional training cost, together with the propagation of the updated parameters in the case of re-training that is required in a real setting, is not included in the evaluation.

5.3.4 Run Length Analysis

To further understand the characteristics of the wireless links in the experiments, we extend our analysis to the run length of the links that were used in the data



(a) Local Testbed

(b) Motelab Testbed

(c) Indriya Testbed

Figure 5.18: CDF of the ratio of the packets sent through a particular run length with respect to the total number of packets sent in the data forwarding path. The run length is labeled in the x-axis and limited to $[1, 50]$ range. The results from different link estimators are marked with different line styles.

forwarding path. If we consider the packet reception of a link as a binary string comprised of 0s and 1s whereas 0 represents packet loss and 1 represents packet successfully received and acknowledged, the run length of the link can be defined as the number of consecutive 1s in the binary string. Therefore, the distribution of the run length of a link can represent the stability and the overall quality of the link: large number of short run length suggests that frequent packet losses occur in the link and the overall quality is low, whereas long run length implies that sparse packet losses and high link quality. In our case, multiple links are selected by the routing protocol to form the data forwarding path during each of the experiments, so the aggregated run length distribution of all the links used in the forwarding path will give us some useful insights on the overall link quality variations of the routing path. Fig. 5.18 presents such run length distributions for all the experiments run in three testbeds.

Fig. 5.18 contains three sub-figures, and each sub-figure plots the CDF of the ratio of the packets sent through a particular run length (labeled in the x-axis) with respect to the total number of packets sent in all the experiments done in the respective wireless sensor testbed. Results from experiments done with the four link estimators, namely, STLE, 4Bit, 4C and TALENT, are plotted with different styles and colors to show the link selection preferences of these link estimators respectively. We limit the maximum run length shown in each figure to 50 because including larger run lengths will skew the figure and hide the important details in the [1, 50] run length range. Plus, only a small fraction of the total packets are sent with run length larger than 50. Therefore, we consider the CDF of the ratio of packets sent through run lengths between [1, 50] range is illustrative enough to show the characteristics of the links selected by routing protocol with the four different link estimators.

Fig. 5.18(a) shows a clear distinction between the four link estimators. In the case of STLE, more than 80% of the packets are sent with run length smaller than 10, suggesting that STLE strongly prefers the links with frequent packet losses. This behavior of STLE is consistent for all the experiments done in the other two testbeds, as shown in Fig. 5.18(b) and 5.18(c). Furthermore, combined with Fig. 5.17(b), we see that STLE tends to aggressively select long, intermediate quality links to form short routing paths. However, these long links cause excessive packet losses and offset the advantage of having a short routing path, ending up with a higher overall cost.

4Bit shows an opposite trend of STLE. In the local testbed and the Indriya testbed, 4Bit sent high ratio of packets with long run lengths, which is reflected by the low curves in Fig. 5.18(a) and 5.18(c). This behavior shows that 4Bit tends to use high quality links in the forwarding path. Again, combined with Fig. 5.17(a), it can be inferred that the general strategy of 4Bit is to use high quality links to construct long but reliable paths. Nevertheless, as shown in the previous section and the later sections, this consecutive strategy is not optimal compared with 4C and TALENT.

On the other hand, due to the sparsely connected network in the Motelab testbed and limited number of possible paths, 4Bit is forced to use more intermediate links in the Motelab experiments. As shown in Fig. 5.18(b), 4Bit sent large number of packets with short run lengths, causing a higher packet ratio than 4C and TALENT in the range between 1 and 20. As discussed in the previous section, this high packet ratio in the short run length range suggests that 4Bit tried to send packets through links with degraded quality, whereas 4C and TALENT is able to switch to alternative routes fast than 4Bit, resulting in lower overall costs.

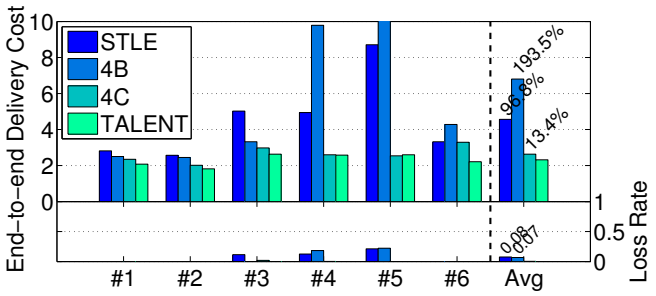


Figure 5.19: End-to-end delivery cost and loss rate of local testbed. The experimental settings are described in Fig. 5.13. Each bar represents the average results of 3 experiments with the same network settings.

As to 4C and TALENT, their run length distributions are very similar in all the experiments and always lay between 4Bit and STLE except for the Motelab experiments. This signifies the effectiveness of the underlying prediction models of 4C and TALENT: the prediction output of high quality periods enables them to select potential high quality links more accurately than STLE, and form shorter paths than 4Bit. By finding the middle point between the aggressive STLE and conservative 4Bit, 4C and TALENT show better end-to-end cost than both STLE and 4Bit as presented in the next section.

5.3.5 End-to-End Delivery Cost and Loss Rate

In this section, we further present the performance of TALENT in terms of end-to-end delivery cost and loss rate, as well as the network settings for each experiment. As mentioned in Section 5.3.1, the results presented here are from extensive single sender experiments conducted in three wireless testbeds: the local, Motelab and Indriya testbeds.

Figs. 5.19, 5.20 and 5.21 show the average end-to-end delivery cost per packet sent on the top and the end-to-end loss rate at the bottom of each figure. The

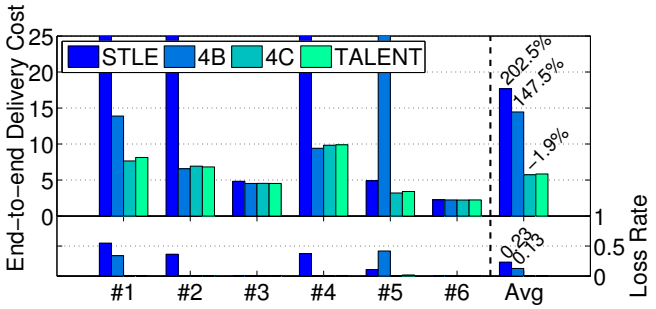


Figure 5.20: End-to-end delivery cost and loss rate of Motelab testbed experiments. Motelab testbed is sparsely connected, so the number of good paths is limited, which leads to similar cost when the network is stable and heavy cost increments when the path is disturbed.

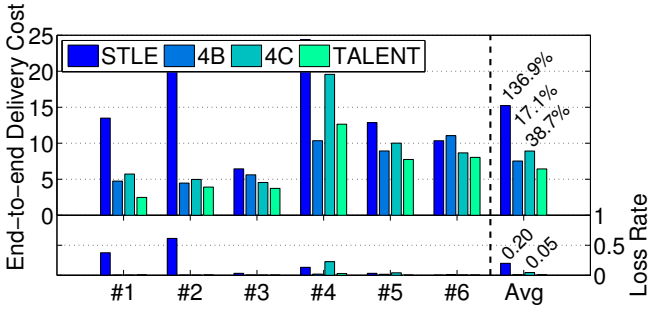


Figure 5.21: End-to-end delivery cost and loss rate of Indriya testbed.

numbers on the x axis mark the experiment number, representing different network settings. Each column represents the results of three experiments conducted in the same network under the same conditions using the same sender.

Each experiment uses STLE, 4Bit, 4C and TALENT for 10 minutes back to back. The last column in each figure shows the average of all the experiments in each testbed. Overall, we see that TALENT provides the overall best packet delivery cost, with average improvements over all testbeds of 18% over 4C, 145% over STLE and 119% over 4Bit. Moreover, TALENT reduces the end-to-end loss rate on average over all our experiments by 1.5% over 4C, 17% over STLE, and

6.7% over 4Bit.

From Fig. 5.19 experiments 4 and 5, and from Fig. 5.20 experiments 1 and 3 we see that the delivery cost of 4Bit is very high. This behavior can be explained by the problems explained in Section 5.3.2, i.e. slow beaconing activity on alternative paths and slow EWMA convergence. Trace analysis indicates that in these experiments, the number of available forwarding path was limited. If the old forwarding path was broken due to link failure, the sender could not find an alternative parent and had to wait for the beacon from neighboring nodes to update the link quality. This leads to many retransmissions and eventually packet losses, whereas TALENT maintains a connected network due to the receiver-initiated approach. For example, in Figure 8 Exp #5, the 4Bit's end-to-end loss rate is close to 25%, which corresponds to a high delivery cost due to the excessive send attempts to a parent that's no longer available. When we run TALENT under the same conditions in the same network, it achieved near 0% loss rate because the overhearing nodes can become temporary parents as soon as the sender's old parent is no longer reachable.

The performance of STLE was all over the place. In all the different environments tested, sometimes it achieved reasonable results, but other times it led to a significant increase in delivery cost and loss rate. STLE had the highest rate of parent changes of all the estimators tested, which leads to a lot of control packet overhead. Further, the heuristic used to decide parent changes (3 consecutive successes to switch to a temporal parent, and 1 loss to go back to the previous path), may lead to wrong routing decisions and bad paths are chosen. STLE ended up with the worst performance in terms of end-to-end delivery rate of all the schemes tested.

It can be observed that TALENT is still better than 4C on average but not by

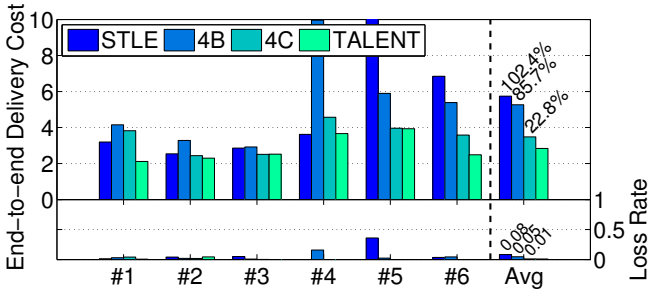


Figure 5.22: End-to-end delivery cost and loss rate of variable sending rate and single sender.

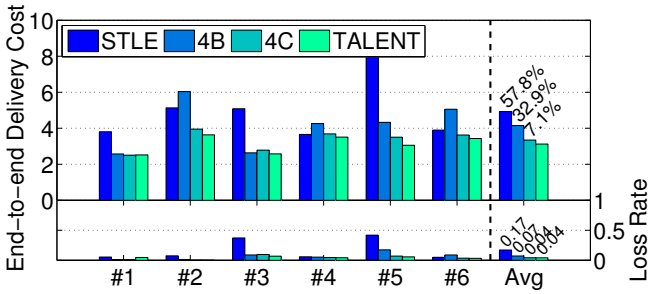


Figure 5.23: End-to-end delivery cost and loss rate of variable sending rate and multiple senders.

a wide margin compared to STLE or 4Bit. This is not very surprising given that both 4C and TALENT employ LR based prediction models, but again, the cost of model training required by 4C is not included in the results, whereas TALENT does not require prior training due to the use of an online learning algorithm.

5.3.6 Variable Rate and Multiple Senders

To evaluate TALENT in more realistic environments, we conducted more experiments in the local testbed with variable sending interval as well as multiple senders. We also introduce additional interference by using wireless channel 11, which is often shared by 802.11 traffic.

The variable rate experiments used the same network settings in local single sender experiments, the only difference is the inter-packet interval is randomly selected in $[50, 150]$ ms range instead of using a fixed interval of 100 ms. As shown in Fig. 5.22, the end-to-end the delivery cost and loss rate results are similar to the fixed interval experiments results observed in Fig. 5.19, indicating the variable rate does not affect the performance of TALENT significantly.

To test the performance of TALENT in congested networks, we also conducted several experiments with multiple nodes sending simultaneously with packet interval ranging from 50 to 100 ms. Please note that although having multiple senders is common in WSNs, it is rare that these senders send packets at a high data rate at the same time. We consider this experiment setting as the worst case scenario where the network is congested and affected by external interference.

Fig. 5.23 shows that while TALENT still outperforms STLE and and 4Bit by 57.8% and 32.9% in terms of delivery cost, the cost reduction is smaller than with only one sender. Trace analysis shows that multiple senders created more data forwarding paths compared with single sender experiments, and hence helped the 4Bit link estimator to evaluate more links with higher rate.

5.4 Discussion

5.4.1 Performance in 802.11 Networks

The results presented in Section 5.1 and 5.3 are encouraging as they clearly indicate the potentials of the online prediction approach in low power wireless networks based on IEEE 802.15.4 standards. Moreover, our approach of using online learning prediction models to estimate the short term wireless link quality with both link layer and physical layer information applies to wireless networks

in general and is not restricted to low power wireless networks. In order to analyze the potential application of the prediction model in high power, high data rate networks, we evaluate the prediction approach with empirical packet traces collected from 802.11 networks.

We used two packet traces available in the CRAWDAD [CRA08] wireless network data repository. The first dataset is from Rutgers University noise dataset [KGS07], which is collected from an indoor wireless network testbed comprises 128 IEEE 802.11a/b/g radio interfaces attached to 64 static nodes arranged on an 8 by 8 grid. The dataset includes more than 500 packet traces, each trace contains received signal strength indicator (RSSI) for each correctly received frame at receiver nodes with certain levels of noise injected on the testbed, whereas the transmitter sends one beacon packet per 100 milliseconds. The testbed injects additive white Gaussian noise interference at center frequencies of 250KHz to 6GHz using an Agilent E4438C ESG vector signal generator.

The other dataset is from the indoor 802.11 signal strength measurements [BMA09] conducted by the System Research Lab from the University of Colorado at Boulder (UCB). This dataset provides a comprehensive set of RSSI readings from within an indoor office building. It captures RSSI behavior when 802.11 frames are transmitted using a stock omni-directional antenna with the transmit power set to 16 dBm. The omni-directional RSSI measurements are collected from roughly 180 distinct physical locations throughout a large office building. The transmitter sends 500 packets from each of the 180 physical positions, and the measurement packets are recorded by 5 passive monitors, which are commodity Linux machines with 802.11 cards. Each RSSI measurement is labeled with the transmitter's physical location.

In order to evaluate the performance of TALENT in 802.11 networks, we ap-

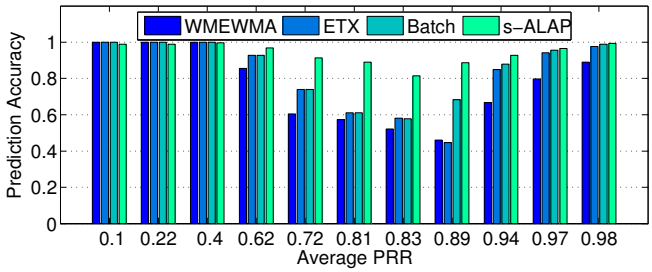


Figure 5.24: Prediction accuracy of WMEWMA, ETX, batched trained LR model and LR model with s-ALAP applied to Rutgers dataset. Only 11 out of 112 total links are shown here to represent links with diverse qualities.

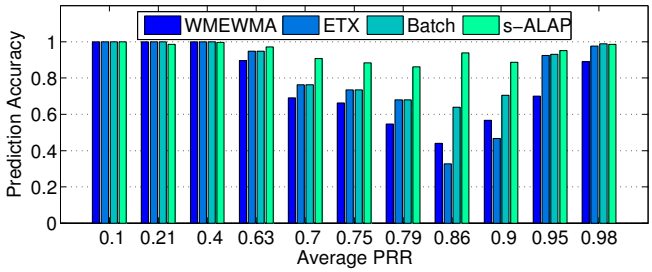
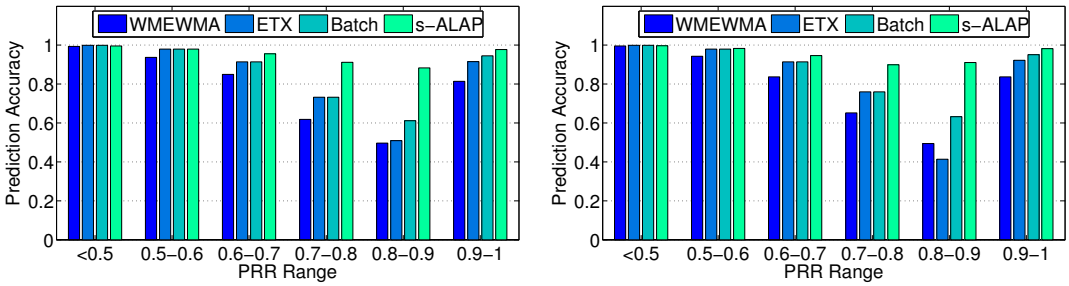


Figure 5.25: Prediction accuracy of WMEWMA, ETX, batched trained LR model and LR model with s-ALAP applied to University of Colorado at Boulder dataset. Only 11 out of 130 total links are shown here to represent links with diverse qualities.

ply four link estimators to the two datasets, namely, 1) WMEWMA with a strong smoothing factor $\alpha = 0.9$ and ETX calculation window of 5, which represents the long term ETX based link estimation used in 4Bit, 2) ETX computed with a window of 5 without any smoothing for reactive ETX estimations, 3) batch trained LR model fitted to each individual link, which represents the best accuracy of the batch trained LR model can achieve (i.e., over-fitted), and 4) the online learning LR model with s-ALAP used in TALENT. The only modification made to the TALENT model is to include RSSI in the input instead of LQI.



(a) Rutgers, 112 links total

(b) UCB, 130 links total

Figure 5.26: Average prediction accuracy of WMEWMA, ETX, batched trained LR model and LR model with s-ALAP, categorized with respect to PRR range.

Fig. 5.24 and 5.25 present some of the prediction accuracy results of these four link estimators applied in Rutgers University dataset and the dataset from UCB respectively. The x-axis in these figures denotes the average PRR of each link, whereas the bars represents the prediction accuracy of the link estimators respectively. For better readability, we only include 11 links with diverse PRRs ranging from 0.1 to 0.98 in both figures, but the total number of links in these two datasets are much larger (113 in the Rutgers dataset, 130 in the UCB dataset). The overall prediction accuracy results are presented in Fig. 5.26(a) and Fig. 5.26(b) respectively.

In general, Fig. 5.24 and 5.25 show that the online learning LR model used in TALENT performs significantly better than the other link quality estimators for the links with average PRR between 0.7 and 0.95, confirming the findings presented Fig. 5.2. More specifically, the ETX estimation outperforms the WMEWMA for the links with PRR higher than 0.8, but it is worse than WMEWMA for the links with PRR ranging between 0.6 to 0.8. This result implies that neither the reactive ETX estimation nor the smoothed ETX estimation from WMEWMA can fit to a wide range of links with varying quality as a sin-

gle rigid smoothing factor can not capture the underlying dynamics of different links. The batch trained LR model performs better than reactive ETX for the links with with PRR above 0.8 and is on par with WMEWMA for the links with lower PRR. The online learning model in TALENT (s-ALAP) outperforms all three other link estimators especially for the links with PRR ranging between 0.7 to 0.95, indicating that TALENT can potentially better predict the link quality even in 802.11 networks. This trend can be also be observed in the overall prediction accuracy results from Fig. 5.26. In the Rutgers results presented in Fig. 5.26(a), s-ALAP model used by TALENT consistently outperforms the other link estimators especially for the links with PRR between 0.7 to 0.9 range. In particular, s-ALAP achieves 88% prediction accuracy for links with PRR between 0.7 and 0.8, 38% higher than WMEWMA results for the same links. The results from the UCB dataset presented in Fig. 5.26(b) are consistent with the Rutgers results.

These results confirm that the prediction approach of TALENT indeed can be used in other wireless networks. Although in this work, TALENT is used to improve the routing cost and reduce the energy consumption for low-power wireless sensor networks, it also applies to ad-hoc networks for better route selection, higher throughput and lower latency. With a predictive link estimator similar to TALENT, the transmitting nodes in ad-hoc networks can better identify the short temporal link quality in the near future and select the node with the best routing cost to forward the data packets, forming a better routing topology in terms of throughput and/or end-to-end latency. However, the exact design and potential applications of such predictive model, especially in multi-rate 802.11 networks are beyond the scope of this work.

5.4.2 Impact of Low Power Listening

Based on the results presented in Section 5.3, TALENT works well with LPL enabled. However, if LPL is misconfigured, it could potentially reduce the performance of TALENT and CTP.

The most important LPL parameter is the *wakeup interval*, which determines how long should radio sleep between receive checks. If this interval is too long, i.e., longer than the inter-packet interval, packet loss may occur due to queue overflow on the sender node. It is normally not a problem for low data rate applications, but for applications with burst traffic pattern, the wakeup interval needs to be carefully selected. In our evaluation, we set the wakeup interval to 100 ms such that the nodes can wake up frequently enough to receive or snoop packets in the wireless channel.

Another parameter is the *delay after receive*, which controls the time for the radio to stay on after receiving a packet. This parameter is useful in bursty traffic as it can prevent the node from entering sleep mode unnecessarily when packets are sent back to back with small intervals, and consequently reduce the overhead of radio sleep/wakeup operations.

Fine tuning these parameters is out of the scope of this chapter. Nevertheless, an application that generates bursty traffic could potentially set the LPL parameters to facilitate the burst data transfer using TALENT as discussed in the next chapter.

5.4.3 Integration with other MAC Protocols

TALENT relays on the assumption that the underlying MAC protocol supports overhearing, i.e., snooping on packets that are not addressed to the node. In

this work, TALENT is implemented on top of LPL, which is realized in the BoX-MAC [ML08], the default MAC protocol in TinyOS 2. As discussed in Section 5.2.5, a nice feature of BoX-MAC is that it provides the overhearing interface without incurring much overhead in terms of power consumption because the energy-based receive check (CCA) does not perform any address check. Note that some MAC protocols such as X-MAC [BYA06] perform address checks before starting to receive the data packets, which limits the use of overhearing operation. In this case, using overhearing may negatively impact the performance of LPL. However, as pointed out by Moss and Levis [ML08], on CC2420-based platforms, BoX-MAC consumes up to 40 – 50% less energy than X-MAC under reasonable workload. Therefore, we consider that our evaluation with BoX-MAC is sufficient. Furthermore, even if the underlying MAC protocol does not support overhearing with LPL, we believe that the PHY parameters could still be estimated when the MAC performs receive checks.

5.4.4 Limitations of TALENT

The main limitation of TALENT is that it only works in high data rate scenarios. Due to the short coherence time of the wireless channel and quick dynamics of the link quality, historical packets from several seconds ago may not represent the currently channel quality anymore and do not correlate with the current packet receptions. Consequently, TALENT only works well under high data rate when the last packet transmission happened recently. Using TALENT in low data rate applications will not harm the routing performance, but it will not provide much gain in terms of delivery cost.

This limitation can be overcome by utilizing TALENT only when a batch of packets needs to be sent. We leave the decision to the application/network

level as the higher level protocols will have more control of when and how many packets to send. Ideally, TALENT-aware routing protocols should have two operation modes: low data rate mode, in which the TALENT is disabled and the LPL wakeup intervals are set to a large value, and burst mode, in which TALENT is enabled and LPL incorporates short wakeup intervals. By doing local buffering and sending packets in bursts, applications allow TALENT to select the instantaneous low cost paths, trading off increased latency for significantly larger delivery efficiency and smaller delivery costs. Chapter 6 provides a reference design of such routing protocol.

5.5 Summary

In this chapter, we present TALENT, a self-learning, plug-and-play estimator to predict the quality of a wireless link in the near future using a combination of packet and physical level quality indicators. One of the main advantages of TALENT is the use of online learning techniques that are able to adapt to the wireless dynamics without the need for data collection and model re-training. When using TALENT together with CTP, our experimental results show that on many different environments TALENT increases the delivery efficiency more than 1.95 times in comparison to state-of-the-art link quality estimators. TALENT and 4C presented in the previous chapter provide agile link estimation for both short term (1 second into the future) and long term (1 minute) link quality prediction, and offer reference implementation in TinyOS. In order to better utilize TALENT in typically low data-rate, duty-cycled sensor networks, we temporarily move away from the topics of link estimation and focus on efficient data forwarding protocols in duty-cycled sensor networks in the next chapter. The goal is to study how to combine TALENT with the data forwarding protocol to provide

a holistic solution to the energy efficient data forwarding in low-power sensor networks.

CHAPTER 6

Synchronous Anypath Forwarding in Duty-Cycled Networks

As discussed in Section 2.1, a common technique to reduce energy usage in low-power WSNs is to duty-cycle the radio, such that the radio is turned on only when there is data to send. For example, the default TinyOS MAC protocol, X-MAC [ML08], uses a low power listening scheme called LPL to duty-cycle the radio. X-MAC takes an asynchronous approach that does not involve any time synchronization. Instead, each packet is transmitted repeatedly until the receiver wakes up and acknowledges the packet. This asynchronous approach requires no central duty-cycle control or any time synchronization scheme. However, in practice, the sender nodes may need to spend excessive amount of energy by repeatedly sending data just to wait for the receiver to wake up. For example, if the receiver wakes up every 500 milliseconds, the sender needs to repeat sending the same packet for 250 milliseconds on average, enough for a sensor to node to send a 50-byte long packet more than 20 times. The same situation applies to other receiver-initiated MAC protocols such as RI-MAC [SGJ08] and A-MAC [DDC10], which let the sender keep the radio on until the receiver wakes up and polls the wireless channel. In this case, the energy is spent on idle listening instead of blindly sending. Thus, in general, the asynchronous duty-cycle techniques often spend excessive amount of energy in idle sending/receiving.

The uncertainty in packet transmission schedule poses an even bigger problem for routing protocols in WSNs such as CTP [GFJ09]. In general, routing algorithms try to send data packets as soon as they receive them from the upper layers and/or other child nodes when forwarding packets. While this behavior tends to minimize network latency, it has also several drawbacks:

1. Long intermediate quality links (i.e. links that cover longer distances in the transmission region) can provide potential gains by reducing the end to end forwarding costs, provided that these links are in a high quality period when the transmission occurs. However, as demonstrated in Chapter 5, the high quality periods are not persistent and highly variable due to the dynamic nature of the intermediate links. With sparse and unpredictable traffic patterns, the packets from upper layers seldom arrive during the high quality periods, making it hard for the routing protocol to take advantage of the intermediate quality links even though short term link estimators such as TALENT can identify when such high quality periods would occur. As a result, routing protocols such as CTP mostly utilize shorter but stable links to from a reliable but sub-optimal routing structure.

2. The sender's wake up schedule is essentially defined by the application and/or the timing of data arrival (in case of data forwarding). Hence, the networking stack has little information about the expected transmission schedule and this complicates prediction of future data transmissions.

3. The resulting communication pattern mainly consists of forwarding individual data packets, which makes it hard for the networking protocols to take advantage of techniques for efficient bulk data transfer such as opportunistic forwarding [BM05].

On the other hand, a more predictable transmission schedule has many advantages since it can: (a) minimize idle sending time, (b) allow multiple nodes to wakeup only when packet(s) are ready, and (c) enable opportunistic routing in low duty-cycle networks.

Our approach to address the above problems is to design a cross layer data forwarding scheme named Synchronized Anypath Forwarding (SAF). The main idea of SAF is simple: we try to force the traffic pattern to be bursty (high data rate in the short term) and predicable (to facilitate scheduling) by buffering the packets from upper layers and scheduling the duty-cycles of neighboring nodes in a synchronized fashion. By shaping the traffic to a more regular and predicable pattern, SAF eliminates the need of idle sending/receiving. More importantly, with bursty traffic patterns and a synchronized wake-up schedule, we can apply the principles of opportunistic routing and utilize the intermediate links to forward packets more efficiently. The price to pay for this is increased end to end latency in the delivery of packets.

Contribution and Novelty

The main contribution of this chapter is the design and the implementation of SAF, a novel opportunistic anycast routing protocol. SAF actively regulates the traffic and takes the advantage of the predicable and bursty traffic pattern to forward data in an energy efficient manner. By forcing a sender-driven adaptive wakeup schedule, SAF allows synchronization between senders and multiple forwarding nodes and adapts the wakeup value based on the dynamic data traffic rate measured by the sender. Also, we implemented SAF in TinyOS 2.x and show that SAF reduces the total duty-cycle by 71% (up to 90% for individual nodes), and data transmission duty-cycle by 98% (up to 99% for individual nodes, while increasing overall end to end latency by 24 seconds on average. Compared

with ORW [LGD12], another opportunistic routing scheme, SAF reduces total duty-cycle by 48%, and data transmission duty-cycle by 97%.

The novelty of SAF comes from the unique combination of several previously proposed techniques which provides an efficient cross-layer data-forwarding scheme for duty-cycled networks. For example, traffic shaping by buffering data packets is used in previously proposed protocols either implicitly [ED04] or explicitly [GH07], and the synchronous wakeup/sleep schedule has been proposed in many studies on low duty-cycle protocols [YHE02, ED04, LYH05, YSH06]. However, the focus of SAF is to utilize the traffic shaping and synchronous wakeup schedule to create a bursty traffic pattern such that the routing component in the network layer can leverage the common wakeup schedule to transmit the buffered packets to multiple nodes in an efficient manner, which is achieved by using batch packet transmission with bitmap ACK and opportunistic routing. Again, bitmap ACK has been proposed before [SGD09], and opportunistic routing has been extensively studied in the literature [BM05,SIV09,LGD12], but SAF is the first to combine batch packet transmission with opportunistic routing in low data-rate, low duty-cycle scenarios. Also, with the help of the previously proposed TALENT, SAF can potentially optimize the routing structure even further. Therefore, although the individual techniques used in SAF are not new, we consider SAF is novel as it proposes a new design angle on how to integrate these existing techniques together to provide an efficient data-forwarding protocol for duty-cycled networks.

6.1 Energy Usage in Duty-Cycled Sensor Networks

As discussed in Section 2.1, duty-cycled sensor networks present more challenges for radio communications in addition to the vagaries of WSNs. In order to thor-

oughly understand the energy usage of radio communication in duty-cycled sensor networks, we conduct comparative analysis of power consumption for several combinations of state-of-art routing/MAC protocols, namely, CTP working with X-MAC [ML08] and A-MAC [DDC10], as well as ORW [LGD12], an integrated cross layer routing/MAC scheme. Our analysis dissects the energy usage for *single hop* packet delivery into three categories: *check*, the energy used in checking for radio activities in the wireless channel, *wait*, the energy spent by the radio while the sender is waiting for the receiver to wake up, and *send*, the energy used to deliver the packet to destination. The analysis focuses on the energy usage of different MAC/routing protocols in the most common scenario, i.e. a single link with one sender and one receiver, in order to understand the fundamental components involved in the overall power consumption.

6.1.1 X-MAC

Figure 6.1 illustrates the basic operation of X-MAC. In X-MAC, a node turns on the radio with a fixed wakeup interval T_w to check for any radio activity in the wireless channel, indicated by the Clear Channel Assessment (CCA) in the figure. To send a unicast packet, the sender attempts to send the packet to the receiver repeatedly until the receiver also wakes up, successfully receives the packet, and sends an ACK packet back to the sender. Then, the sender returns to the regular wake up schedule after receiving the ACK, whereas the receiver will keep the radio on for a short delay to account for potential consecutive packets. As shown in Figure 6.1, the sender S has to repeat the packet (denoted as D) twice before it is acknowledged by the receiver S due to channel dynamics/errors. The wakeup schedule of X-MAC is asynchronous, which means the sender and the receiver will wake up at different times, and they are not aware of the wakeup schedule

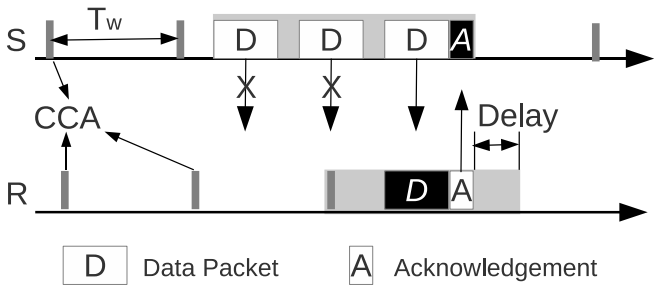


Figure 6.1: Packet flow of X-MAC. The gray area indicates period when the node turns its radio on.

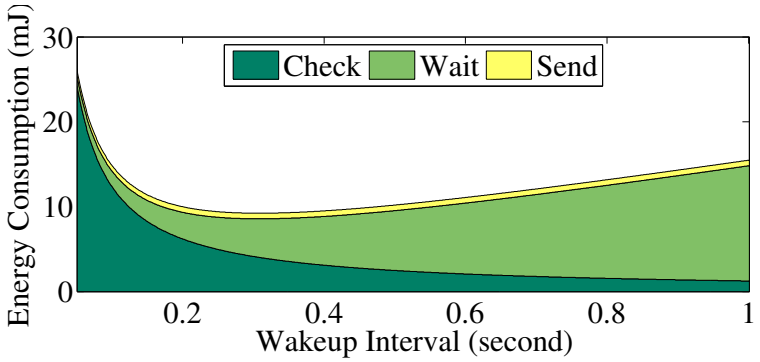


Figure 6.2: Energy breakdown of X-MAC with respect to wakeup interval T_w. The inter-packet interval $T_{ipi} = 2$ seconds.

of the other nodes.

Energy Usage Breakdown: Based on this simple scenario, and assuming no packet losses to simplify the analysis, the energy consumption of a sender node delivering a packet in X-MAC can be represented as follows:

$$E_{sender} = E_{sleep} + E_{check} + E_{wait} + E_{send} \tag{6.1}$$

Assuming the sender is a TMote, a commonly used sensor node with CC2420 low-power radio chip, we can derive the expected energy consumption using the

Symbol	Meaning	Value
P_{tx}	Power when transmitting	52.2mW
P_{rx}	Power when receiving	56.4mW
P_{chk}	Power when checking	56.4mW
P_{sleep}	Power when sleeping	3μW
L_{data}	Data Packet Length	50 bytes
t_{chk}	Time to check the channel	11ms
t_{pkt}	Avg. time to transmit a packet	5ms
t_{ack}	Avg. time to transmit an ACK	3ms
t_{dly}	Time wait for more packets	20ms
T_w	Wakeup interval	Varying
T_{ipi}	Inter-packet interval	Varying

Table 6.1: Radio parameters of a TMote Sky mote running TinyOS 2.

parameters listed in Table 6.1. From this table, we notice that the power consumption in sleep mode is several orders of magnitude less than the power consumption when the radio is on, therefore the E_{sleep} can be safely omitted from Equation (6.1).

Assuming the sender sends packets with a certain inter-packet interval T_{ipi} and wakes up to check for radio activity every($T_{ipi} > T_w$), the number of channel checks performed during this T_{ipi} is approximately T_{ipi}/T_w. Therefore, the total energy of channel checking during T_{ipi} is:

$$E_{check} \approx P_{cca}\frac{T_{ipi}}{T_w}t_{chk} \tag{6.2}$$

In a duty-cycled network, the sender also needs to wait for the receiver to wake up. In the X-MAC case, the sender has to keep repeating the data packet long enough such that the receiver detects radio activity when it wakes up to check the

channel. Due to the asynchronous wake up schedule, the waiting time could be between $[0, T_w]$, but on average, the mean waiting time is $t_{wait} = T_w/2$. Therefore, the average energy spent on waiting is:

$$E_{wait} = P_{tx} T_w / 2 \tag{6.3}$$

The energy used to deliver a single packet only consists of sending the packet and receiving the ACK. Thus, we have E_{send} as a fixed value:

$$E_{send} = P_{tx} t_{pkt} + P_{rx} t_{ack} \tag{6.4}$$

Substituting Equations (6.2)-(6.4) into (6.1), we have the expected energy consumption of the sender given the inter-packet interval T_{ipi}:

$$E_{sender} = P_{chk} \frac{T_{ipi}}{T_w} t_{chk} + P_{tx} T_w / 2 + P_{tx} t_{pkt} + P_{rx} t_{ack} \tag{6.5}$$

Similarly, we can derive the energy consumption of the receiver node:

$$\begin{aligned} E_{receiver} &= E_{check} + E_{wait} + E_{receive} \\ &= P_{chk} \frac{T_{ipi}}{T_w} t_{chk} + P_{rx} t_{dly} + P_{rx} t_{pkt} + P_{tx} t_{ack} \end{aligned} \tag{6.6}$$

Here E_{wait} refers to the delay at the end of packet reception. Combining the energy consumption of both the sender and the receiver, we obtain the combined channel checking energy consumption as:

$$\hat{E}_{check} = 2 \times \frac{T_{ipi}}{T_w} P_{chk} t_{chk} \tag{6.7}$$

The total energy spent on idle sending/waiting as:

$$\hat{E}_{wait} = \frac{1}{2} P_{tx} T_w + P_{rx} t_{dly} \tag{6.8}$$

And finally the energy for delivering one packet is:

$$\hat{E}_{send} = P_{rx} t_{snd} + P_{tx} t_{snd} \tag{6.9}$$

where $t_{snd} = t_{pkt} + t_{ack}$, representing the total time to deliver a packet. The total energy consumption is the sum of all three terms:

$$\hat{E} = \hat{E}_{check} + \hat{E}_{wait} + \hat{E}_{send}. \tag{6.10}$$

To understand how much each term weights in the total energy usage, we plot the detailed energy usage as a function of the wakeup interval T_w for a TMote Sky node [Mot] in Figure 6.2. The radio parameters are listed in Table 6.1. It can be observed that, with a small T_w, the majority of the energy is spent on channel checking. However, as T_w increases, the channel checking energy is quickly declined whereas the energy used in waiting is increasing. For example, when $T_w = 0.2$ seconds, E_{check} occupies 51% of total energy usage, whereas E_{wait} occupies 43% of total energy. Combined together, they are responsible for more than 95% of total energy usage for delivering a packet. When $T_w = 0.5$ seconds, E_{wait} weights more than 80% of the total energy usage, whereas E_{check} corresponds to 15% of energy usage. In both of the cases, the energy used for actual send the packet to the receiver (E_{send}) only accounts for less than 5% of total energy usage. This behavior reveals an important characteristic of asynchronous duty cycled MAC protocols such as X-MAC: in the most of the cases, a large portion of the energy usage is spent on idle transmitting to wait for the receiver waking up.

6.1.2 A-MAC

A-MAC [DDC10] is a receiver-initiated asynchronous MAC protocol. As illustrated in Figure 6.3, a sender using A-MAC does not inject packets into the wireless channel immediately. When there is a data packet to be sent, the sender simply turns on the radio and listens to the wireless channel, waiting for a probe packet from the receiver. On the receiver side, the receiver node wakes up and

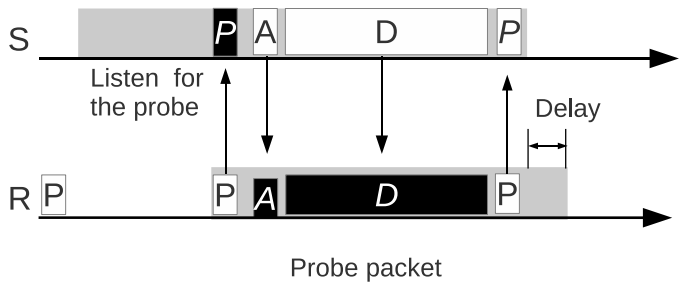


Figure 6.3: Packet flow of A-MAC. The sender listens to the channel until it receives a probe from the receiver, and after the probe is acked, the data packet is sent.

polls the channel using a probe packet periodically. Upon receiving the probe packet, the sender acknowledges the pending data packets and starts sending the data packet after a short random delay. After the data packet is received, the receiver again sends a probe packet, acknowledges the data packet and polls the channel again for any potential senders. Compared with the sender-initiated X-MAC, A-MAC initiates data transmission only after the receiver polls the channel.

Energy Usage Breakdown: The energy usage of A-MAC in this single link scenario is similar to X-MAC as both of them are asynchronous and check channel for pending packets periodically. Therefore, the energy usage of A-MAC can be divided into three components similar to X-MAC, namely, E_{check}, E_{wait} and E_{send}.

A-MAC uses small probe packets to periodically poll the channel. Assuming the time to send a probe is similar to an ACK packet, the energy spent on channel checking can be described as:

$$\hat{E}_{check} = 2 \times \frac{T_{ipi}}{T_w} P_{tx} t_{ack} \tag{6.11}$$

where T_{ipi} is the inter-packet interval, and T_w is the wakeup interval, i.e., the time interval between channel polling.

In A-MAC, the sender needs to wait for the probe packet in an idle listening state, and the receiver also keeps the radio on for a short delay after the data packet is received. Therefore, the total energy spent waiting is:

$$\hat{E}_{wait} = \frac{1}{2} P_{rx} T_w + P_{rx} t_{dly} \tag{6.12}$$

which is very similar to X-MAC except that power consumption of the sender is T_{rx} instead of T_{tx}. Also, the energy for delivering one packet is the same as X-MAC, i.e.:

$$\hat{E}_{send} = P_{rx} t_{snd} + P_{tx} t_{snd} \tag{6.13}$$

The overall energy consumption of A-MAC is the sum of Equation (6.11), (6.12) and (6.12), which is very similar to the energy consumption of X-MAC described by Equation (6.7), (6.8) and (6.9). Therefore, A-MAC has the same weakness as X-MAC: the majority of the energy is spent on idle listening to account for asynchronous wake up.

6.1.3 ORW

ORW [LGD12] tries to address the energy efficiency problem of duty cycled wireless sensor networks with an opportunistic routing approach. The basic idea of ORW is illustrated in Figure 6.4. Assuming we have one sender S and two potential receivers R_1 and R_2, and S is trying to send a data packet to R_1 with X-MAC. Without ORW, only R_1 should respond to the S's packets, and the other receiver R_2 will not acknowledge the packet from S even if it is overheard by R_2. On the other hand, with ORW, all the neighboring nodes can participate in the data forwarding as long as they can provide efficient routes. In this case, whichever

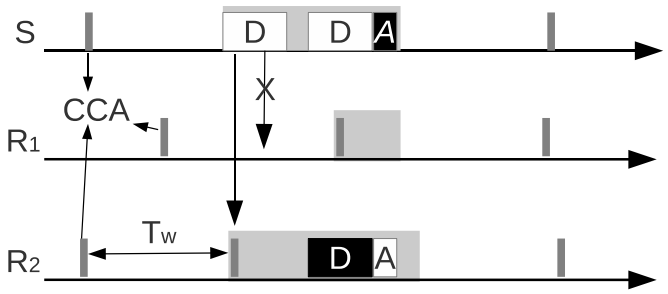


Figure 6.4: Packet flow of ORW with with two potential receivers. One of the receivers lost the sender's packet, but the other receiver can still acknowledge the packet, thus reducing the sender's waiting time.

node receives the packet first, it acknowledges it to S and forwards the packet. By doing so, ORW can reduce the time needed to wait for the receiver to wake up as there are multiple potential receivers. It also takes the advantages of multiple possible forwarding paths to reduce the number of failed sending attempts: if the intended forwarder lost a packet, the packet might be still picked up by other potential forwarders as depicted in Figure 6.4.

Energy Usage Breakdown: Compared with X-MAC, energy usage of ORW is the same in the single link case. However, in the case of multiple receivers, the sender spends less time in idle waiting on average as the sender only needs to wait for any of the receivers to wake up. Assuming N potential receivers, the average time on idle waiting should be N times less compared with a single receiver scenario. Therefore, the energy spent on idle waiting for the sender/receiver pair is defined as:

$$\hat{E}_{wait} = \frac{1}{2N} P_{tx} T_w + P_{rx} t_{dly} \tag{6.14}$$

The energy consumption on channel checking and actually sending the packet is

the same as Equation (6.7) and (6.9).

We can make two observations from the design of ORW. First, although the wait time and the failed send attempts are reduced by utilizing multiple forwarding paths, they are not eliminated as the potential forwarders still wake up asynchronously. Therefore, the energy spent on idle waiting is still a significant portion of the total energy usage. More importantly, because the potential forwarders wake up asynchronously, only a small subset of them (possibly only one if the neighborhood is limited) are able to receive the packet. Once the packet is ACKed by one of the forwarders, the sender does not send the packet again, and effectively excludes all the other nodes from forwarding the packet even if there are better routing paths available. These problems motivate us to design an opportunistic routing protocol that can coordinate all the potential forwarders *synchronously*.

6.2 Protocol Design

6.2.1 Overview

SAF consists of four main ideas: (a) hop by hop *packet buffering* that enables *traffic shaping* and permits efficient *batch transmissions*; (b) *sender-driven transmission schedule* that allows synchronization between the sender and multiple forwarding nodes; (c) *opportunistic anypath routing*, coordinating packet forwarding of multiple nodes by the use of bitmap acknowledgments; and (d) an *adaptive wakeup schedule* based on dynamic data traffic rate. (this last optimization is discussed in Section 6.4.3). In essence, SAF forces the data traffic pattern to be predicable by buffering the packets from upper layers and routing forwarding queues, and then it sends the buffered packets to multiple potential forwarders in

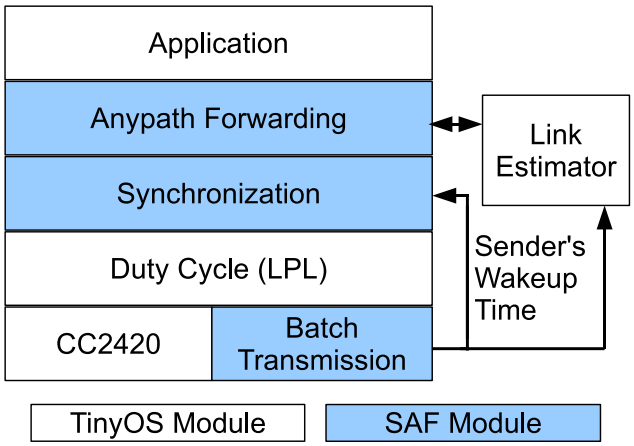


Figure 6.5: Overall SAF Design. After received/overheard data packets, SAF checks whether to participate the data forwarding and store the time information of the sender for synchronization.

a synchronized fashion. The potential forwarders use a time-based coordination scheme similar to ExOR to the select the best forwarders.

The overall structure of SAF is presented in Figure 6.5. As shown in this figure, we extend X-MAC to support batch packet transmission, and add a local synchronization layer on top of the MAC layer duty cycle module, i.e., LPL, to control the local send/receiver synchronization. The opportunistic anypath routing/forwarding implements the routing mechanism, responsible for establishing routing topology and forwarding packets to the sink. These three components are tightly coupled with each other and function as a unit. In the following sections, we discuss the design of all these components in more detail.

6.2.2 Packet Buffering and Batch Transmission

The Anypath Forwarding engine module is in charge of maintaining a packet buffer queue of packets to be transmitted (both from other nodes and the upper application layers). We implement the batch packet send/receive functions in the MAC layer to ensure fast packet transmission and reception. The batch packet transmission module provides an explicit batch send interface for both sender and receiver nodes. On the sender side, this interface is used by the opportunistic/anypath routing component to send a group of packets back-to-back without requesting explicit ACK for each packet, whereas on the receiver side, SAF allows all the nodes with communication range of the sender to receive the packets. A batch sequence number (BSN) is added to the data packet header denoting the number of pending packet in the group, i.e., the first packet has a BSN equals to the size of the packet group minus one, and the BSN of the last packet equals to 0. On the receiver side, the batch transmission suppresses the automatic ACKs and simply notifies the upper layer about the received packet batch. The task of acknowledging the packets is left to the anypath forwarding module.

6.2.3 Transmission Schedule Synchronization

We implement the local sender transmission schedule synchronization as a module that sits between the network layer (anypath routing) and the MAC layer. The task of the synchronization module is to record the next scheduled sender's transmission time based on the time information embedded in the packet, and to wake up the node at the next scheduled time. On the sender side, the sender sets a transmission timer after each packet transmission to trigger a buffer check at a preset transmission interval T_{ti}. The buffer check initiates the batch packet

transmission if the number of packets in the buffer is larger than or equals to the packet group size G. Therefore, the sender guarantees that the next batch transmission will occur at multiples of T_{ti} after the last transmission finished. On the receiver side, once a packet in a batch transmission is received, the end time of the ongoing batch transmission can be calculated based on the the packet BSN as the time to transmit a packet is constant:

$$T_{end} = T_{current} + BSN * t_{pkt} + T_{bitmap} \tag{6.15}$$

where T_{bitmap} is a fixed delay of 30 milliseconds for the bitmap ACKs as discussed in Section 6.2.4. By using the end-of-transmission time and the fixed send interval, the receiver nodes can infer when the sender might start the next batch transmission, and therefore synchronize with respect to the sender. The synchronization module maintains a transmission schedule table that keeps the next wakeup time for the senders whose packets been received or overheard recently, and updates the table whenever new packets are received.

Based on the transmission schedule table, the module wakes up the radio to receive data packets from potential senders. To ensure the receiver can wake up slightly earlier before the sender, a guard time is added to the receiver's schedule. This is to account for the possible time synchronization errors caused by the clock drifts. In the current implementation, the guard time is 10 milliseconds, which is sufficient based on our experience for the traffic conditions tested.

The synchronization module runs in parallel with the LPL, the duty cycling component of X-MAC, therefore the nodes duty cycle the radio with the fixed wakeup interval T_w, and check the channel for radio activities periodically. Please note that even through SAF should be able to synchronize to most of the transmissions with the transmission schedule table, the periodic channel checking is still necessary for catching broadcast packets that are sent at random times, as

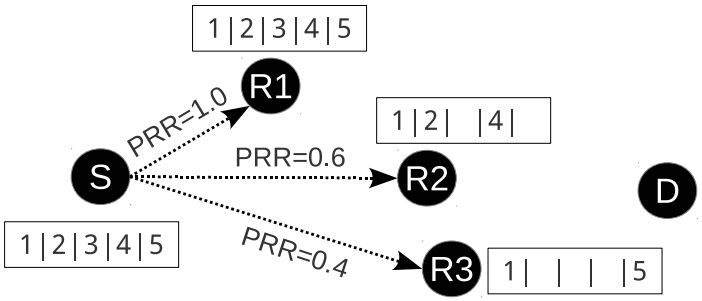


Figure 6.6: Opportunistic routing in SAF uses all nodes as forwarders and route packets through different paths.

well as to discover new or out-of-sync nodes. We present some experimental results with variable adaptive wakeup intervals in Section 6.4.3. Also note that at the end of each batch packet transmissions, we reset the wakeup schedule such that all the participating nodes – sender and receivers – have a synchronized LPL wakeup schedule. This helps the early transmission described in the next section.

6.2.4 Opportunistic Anypath Routing

This is the core component of SAF, which implements opportunistic routing and is responsible for packet forwarding. It reuses the routing module of CTP, but replaces the packet transmission logic of CTP with an anypath routing scheme similar to ExOR [BM05], i.e., allow all the nodes within the broadcast domain to participate packet forwarding operations. As illustrated in Figure 6.6, the sender sends a group of packets in batch transmission, whereas multiple potential forwarders receive the packets and self-coordinate to forward the group of packets collaboratively. By considering multiple path via different forwarders, anypath routing leverages the spatial diversity of the wireless links to improve reliability

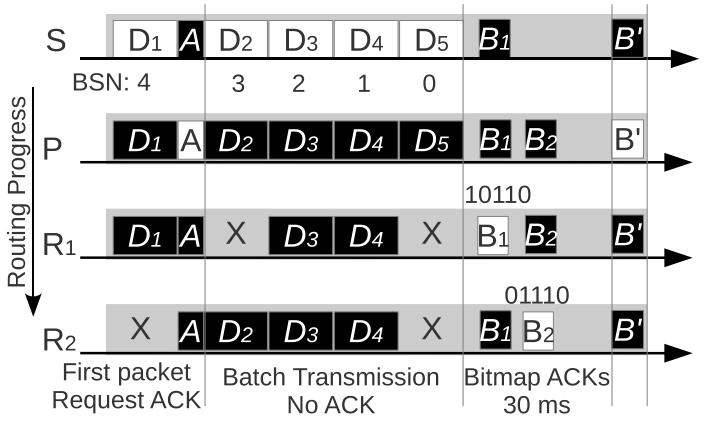


Figure 6.7: Batch packet transmission using anypath routing. The overhearing nodes R_1 and R_2 notify the parent node P about their corresponding packet receptions in a 30 ms period. P decides which node should forward what packets in the end and broadcasts the discussion.

and to reduce packet delivery cost.

The details of the opportunistic anypath routing are illustrated in Figure 6.7. There are four nodes included in this figure, namely, a sender S and three synchronized receiver nodes, P, R_1 and R_2. Node P is the "parent" of S, which means that P is the next hop of S, designated by the traditional CTP tree routing algorithm . Although the other two nodes, R_1 and R_2 have better paths to the sink than P, the low link quality from S to R_1 and R_2 prevents them from being selected as the parent. In traditional routing protocols such as CTP, S ignores R_1 and R_2 and only sends its packets to P. However, SAF operates differently as described below.

Sender Side Operation: In SAF, the sender node buffers the packets to be routed coming from other nodes and from the node's application upper layer until the number of packets in the buffer is larger than or equal to a preset packet group

size G. Once the buffer size exceeds G, the sender tries to send the packets in the buffer as soon as the synchronization module turns on the radio at the next transmission interval. As shown in Figure 6.7, the sender S sends the first packet to its parent P and requests automatic acknowledgment from P as a normal unicast packet. Requesting automatic ACK for the first packet guarantees that at least the parent node is listening to the sender. In the case shown in Figure 6.7, all three receivers wake up synchronously when S starts sending, therefore P acknowledges the first packet (denoted as D_1) immediately. Knowing that P has already waked up, S initiates a batch transmission that sends the remaining packets (D_2 to D_5) back to back without requesting acknowledgments. Once the batch transmission is done, the sender enters a listening state for the receivers to select the final forwarders.

Selection of Potential Forwarders: While the sender is sending, the receiver nodes first compare their routing gradient with the sender's parent. If a receiver node has a lower routing cost than the sender's parent, the node can be a potential forwarder. Otherwise, the node duty-cycle the radio and quit from this packet forwarding cycle. Another pint is that the potential forwarders must have good connection to the parent node as the parent is the coordinator of the forwarders in the later stage. Forwarders without good connection to the coordinator will cause duplicate packets as discussed in Section 6.3.2.

Selection of Final Forwarders: The potential forwarders listen to the sender's packets till the end of the batch transmission (BSN decreases to 0). Then, the receiver nodes need to select the final forwarders as multiple nodes may have received the same set of packets. Conceptually, the forwarders should be able to increase routing efficiency and guarantee the packet delivery at the same time. To achieve this, at least one of the receiver nodes need to be aware of the routing

gradient of other nodes, as well as the packets each node received. SAF uses the parent node P as a coordinator to aggregate the packet reception information of all receiver nodes. The task of the coordinator is to establish a one-to-one packet to forwarder mapping such that each packet in a group is delegated to one and only one forwarder.

Specifically, after the batch transmission, the receiver nodes broadcast bitmap ACKs containing their corresponding packet reception bitmap and the routing gradient in a 30 milliseconds period. To avoid ACK collisions, the bitmap ACK is sent after a random delay between 0 to 25 milliseconds. Moreover, to quickly filter out the nodes with high routing costs, we set the following backoff rules. If a receiver node overhears another node's bitmap ACK, it checks: 1) if the other node has better routing gradient than itself, and 2) if the other node has all the packets received by itself. If both of the conditions are true, the node will quit from this packet forwarding cycle and duty-cycle the radio as the other node is a better forwarder candidate.

During this 30 milliseconds period, the parent node aggregates the reception bitmaps from all the receiver nodes and updates an one to one packet-to-forwarder mapping such that the node with the best routing gradient can forward all the packets it received, and the second best node is responsible for forwarding packets missed by the first forwarder, and so on. At the end of the 30 milliseconds, the parent node broadcasts a coordination packet which pushes the packet-to-forwarder mapping to the sender and all the receiver nodes. If there are holes in the mapping, i.e., packets not assigned by any forwarders, these packets are lost to all the potential forwarders. SAF employs a lazy retransmission technique to retransmit the lost packet as described in the next section.

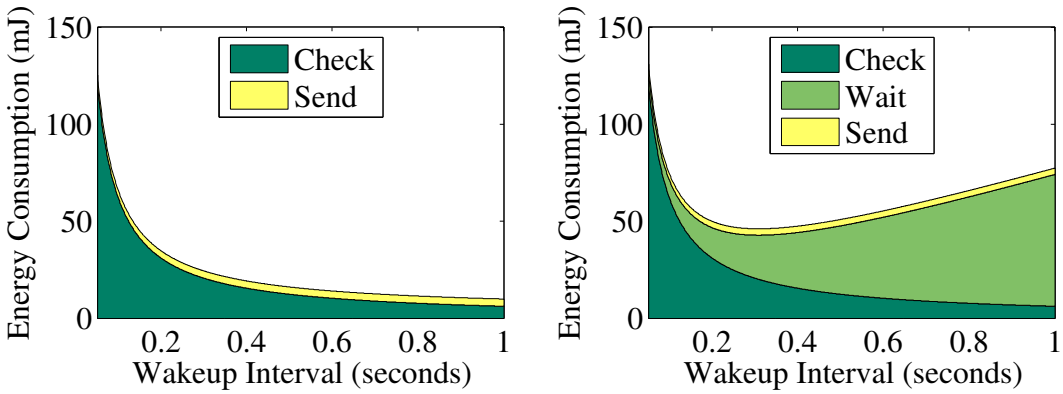
For example, in Figure 6.7, R_1 and R_2 only received a subset of the packet

group, whereas P received all the packets. Once the transmission is done, the parent node enters a listening state to wait for bitmap ACKs from potential overhearing nodes. In our example, R_2 sends its bitmap ACK (denoted as B_2) of 01110 first, which indicates that it has only received packets D_2 to D_4. After a short delay, R_1 also sends a bitmap ACK (denoted as B_1) of 10110, indicating that it has received packets D_1, D_3 and D_3. Because the routing gradient of R_2 is better than both R_1 and P, R_2 is the forwarder of D_2, D_3 and D_4. R_1 is responsible for forwarding only D_1 as other received packets are now delegated to R_2. D_5 was missed by both R_1 and R_2, therefore P is responsible for it. At the end of 30 milliseconds, the parent node P broadcasts the forwarder coordination packet with the packet-to-forwarder mapping, which marks the end of this transmission cycle. The nodes turn off the radio and wait for the next transmission interval.

If one or more forwarders miss the coordination packet sent by P (e.g. due to the vagaries of the wireless link), there are two choices. Either the forwarder(s) does not forward any of the data packets received, or it forwards them all. The first option prevents duplicate transmissions, *but* it may compromise end-to-end reliability/efficiency if the forwarder was the only node receiving those packets. The second option may lead to duplicate transmissions if there is a better forwarder that also received a subset of the packets. In our implementation, we decided to use the latter option. The following section covers the details of SAF implementation.

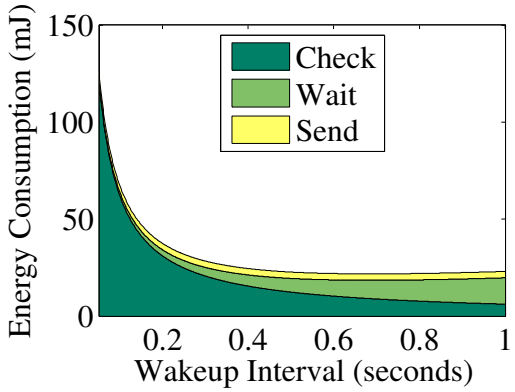
6.3 Energy Usage Analysis and Implementation Details

We implemented SAF in TinyOS based on the protocol design proposed above. In this section, we first analysis and compare the energy usage of SAF with respect to X-MAC in a single hop scenario, then cover the details of our SAF



(a) SAF

(b) X-MAC, Sparse Traffic



(c) X-MAC, Bursty Traffic

Figure 6.8: Energy consumption of SAF, X-MAC with sparse traffic and bursty traffic, side by side comparison.

implementation.

6.3.1 Energy Usage Analysis

Energy Usage Breakdown: The energy on channel checking in SAF is the same as X-MAC, since nodes still need to check for radio activities periodically to account for dynamics in the network. Energy spent on idle wait is now minimal due the synchronization of the sending schedules between the sender and

receivers. As to the energy on sending data packets, in this analysis we assume a simple case: the sender sends G packets in one batch transmission, and the packets are received by N receivers without retransmissions (issues about retransmission are addressed in Section 6.2.2). We also assume that the time to send the bitmap ACK is the same as t_{ack}. Under this assumptions, the sender's energy usage is:

$$E_{sender} = GP_{tx}t_{pkt} + NP_{rx}t_{ack}$$

The energy used by each receiver is:

$$E_{receiver} = GP_{rx}t_{pkt} + (N-1)P_{rx}t_{ack} + Ptxt_{ack}$$

Combining these two equations, the total energy usage of sending G packets to N receivers is:

$$\begin{aligned} E_{send} &= E_{sender} + E_{receiver} \\ &= (Gt_{pkt} + Nt_{ack})P_{tx} + (NGt_{pkt} + N^2t_{ack})P_{rx} \end{aligned} \tag{6.16}$$

Finally, the energy used in sending one packet is E_{send}/G:

$$\hat{E}_{send} = (t_{pkt} + \frac{N}{G}t_{ack})P_{tx} + (Nt_{pkt} + \frac{N^2}{G}t_{ack})P_{rx} \tag{6.17}$$

Comparison with X-MAC

Figure 6.8 presents an energy usage comparison between SAF and X-MAC in a single hop case of one sender transmitting a batch of 5 packets to two potential receivers. Assuming the average T_{ipi} from the upper layer is 2 seconds, SAF starts one batch transmission every 10 seconds to send the 5 buffered packets. The total energy usage during this 10 seconds can be calculated by the Equation (6.7) and (6.13). Figure 6.8(a) shows the SAF energy consumption in 10 seconds as a function of wakeup interval.

Now consider X-MAC under the same scenario. In 10 seconds, X-MAC also sends 5 packets, but the traffic pattern is sparse as X-MAC does not explicitly buffer the data packets. Therefore, the total energy consumption of X-MAC in the 10 seconds is 5 times the energy consumption in $T_{ipi} = 2$ seconds. Figure 6.8(b) plots the X-MAC energy consumption in 10 seconds with respect to wakeup interval.

In addition, we also consider the energy consumption of X-MAC under bursty traffic, i.e. X-MAC sends 5 packet back to back every 10 seconds. X-MAC only needs to wait for the receiver when sending the first packet: due to the delay after the first packet reception, the receiver will be awake when the following packets are sent. In this bursty traffic pattern, the energy consumption of the X-MAC is plotted in Figure 6.8(c) as a function of wakeup interval.

Comparing the three plots in Figure 6.8, we see that the energy consumption of SAF is lower than X-MAC in both of the sparse and bursty traffic cases as SAF minimizes the wait time by synchronizing the sender and receivers. Although SAF uses slightly more energy when sending packets than X-MAC, the difference in E_{send} is negligible compared with E_{wait}. The advantage of removing the wait time is more pronounced when the wakeup interval is large: X-MAC spends more energy on waiting as the wakeup interval increases, whereas the energy usage of SAF actually drops due to reduction in channel checking energy. Section 6.4.2 experimentally confirmed this analytical result.

6.3.2 SAF Implementation

Early Transmission: As pointed out in Section 6.2.3, the LPL wakeup schedule of a sender and its potential forwarders will be synchronized after each batch packet transmission as all the nodes participated in the batch transmission duty-

cycled at the same time. Therefore, it is safe to send at LPL wakeups as long as the parent node is not changed. Based on this observation, SAF adopts an early transmission mechanism, which enables a sender node to send the buffered packets provided that 1) the buffer size exceeds the packet group size G, and 2) the parent node designated by CTP does not change since last transmission. This early transmission mechanism allows nodes to forward groups of packets with the much shorter LPL wakeup interval compared with the send interval $Tsi = 10$ seconds, and consequently reduce the end-to-end latency as discussed in Section 6.4.

Lazy Retransmission: Packet loss is common in WSNs due to link dynamics, environmental variation and node failure. While anypath routing in SAF is generally more reliable due to the parallel use of multiple nodes and links than a unicast packet sent to a single next hop node, it is still possible that a packet is lost to all the receivers. Instead of retransmitting the lost packet immediately, SAF employs a lazy retransmission policy that postpones the packet retransmission to the next transmission interval. In other words, if a packet is not ACKed, the sender simply puts it back in the send queue and waits for the next transmission interval. By delaying the retransmission to the next interval, the sender does not need to constantly wake up just for retransmission. In addition, in the case of link quality related packet loss, delaying the retransmission effectively forces the node to skip the low link quality periods, and consequently reduce the probability of packet loss.

Duplicate Packets: One of the problems of the opportunistic routing is duplicate packets [CJK07]. For example, if two receiver nodes are not in the same communication range, it is possible that both of the nodes will forward the same packet as they do not receive each other's ACKs. It is less an issue in SAF because

the sender's parent are used as a coordinator to manage the forwarders. In the rare case of coordination packet loss due to asymmetric or unstable link, a potential forwarder will buffer the received packet at the end of the transmission and send them at the next transmission interval, causing excessive duplicate packets. To suppress duplicate packets, nodes keep a sent packet buffer which records the signature (source, sequence number and THL) of the packets sent before, as well as the overheard packets sent by other nodes. Before a node buffers a packet to forward, it checks the sent buffer for duplicates. If the packet to be sent matches the signatures in the sent buffer, the packet is a duplicate and will be dropped.

Memory Overhead: Due to the batch packet transmission behavior, SAF needs to allocate enough memory for the data packet buffer. The buffer has to be big enough to accommodate G packets before the next transmission, plus ample room for the incoming traffic. In our implementation, we use a buffer with twice the size of G. In our experiments, the G is set to 10, so the buffer allocated is for 20 packets. In terms of memory overhead, the compiled SAF program uses 3874 bytes of RAM, whereas the same program with CTP uses 2754 bytes of RAM. Consider the 10kB RAM of TMote, the memory overhead of 1120 bytes is acceptable.

Buffer Overrun: There might be situations when the application transmission rates are very high and a node buffer may be overrun by multiple senders transmitting batches of packets without giving an opportunity to the node to empty the buffer in the next transmission interval. In this case, packets are dropped, and end to end delivery rate is affected. Careful consideration of both the *transmission* and *wakeup intervals* values must be used if the application rates are expected to be very high. In addition, SAF early transmission setting (see above), helps ameliorate this problem.

Parameter	Value
T_{ipi} (second)	$2, 3, 4, 5, 6, 7, 8$
T_w (second)	$0.125, 0.25, 0.5, 1, 2, 4, 8$
T_{ti} (second)	10
G	10
Network Density	57 (high) and 19 (low) nodes
RF Power	-25dBm
Packet Length	50 bytes

Table 6.2: The parameters used in the experiments.

6.4 Experimental Evaluation

To evaluate the performance of SAF, we conducted extensive experiments in the indoor testbed consisting of 57 TMote Sky nodes (see Figure 5.13). The TMotes are arranged as 19 groups of three motes, thus form a narrow grid (19×3) covering a 120 meter long corridor in a typical office building.

We compare the performance of SAF against three data collection schemes designed for duty cycled WSNs, namely, ORW, CTP with A-MAC (CTP/A), and CTP with X-MAC (CTP/X), the latter being the default data collection network stack in TinyOS 2.x. To ensure a fair comparison, we run SAF, CTP/A, ORW, CTP/X back to back 3 times in each in our experiments for each set of parameters tested. Each graph shows the average value over all the runs. In each experiment, we set multiple sender nodes that send data packets with a randomized inter-packet interval of $T_{ipi} \pm \frac{1}{2}T_{ipi}$ for 60 minutes. All experiments are run under the same network conditions, and they are run during the day and at night. In all the experiments, we set the radio power level to -25 dBm to increase the network size, and set the radio channel to 11 to avoid interference

from WiFi communications. The packet length is set to 50 bytes.

The performance is evaluated in terms of average duty cycle per node, end-to-end latency and end-to-end packet delivery rate. Average duty cycle is a good approximation of the system energy consumption as the radio is usually the most power hungry component in sensor nodes. End-to-end latency represents the communication delay, i.e., the time from the source sending a packet to the final destination (the base station/sink) receiving it, whereas end-to-end delivery rate represents the communication reliability. In addition, we also use the average number of transmissions (TX) per packet to measure the end-to-end forwarding cost, i.e., the number of transmission attempts needed for delivering a packet to the sink. This metric indicates how good the communication protocol is in finding optimal data forwarding path. In general, having a smaller TX count is better as it means the the routing protocol can forward the packets from source to destination with fewer number of transmissions.

In the following sections, we empirically analyze the performance of SAF under varying conditions, such as different inter-packet intervals T_{ipi}, wakeup intervals T_w, packet group threshold G and the network density. Table 6.2 lists the parameters used in the experiments.

6.4.1 Impact of Inter-packet Interval

This section discusses the performance of SAF with respect to varying inter-packet intervals (T_{ipi}) and fixed wakeup intervals (T_w). In these experiments, T_{ipi} varies from 2 to 8 seconds, and the wakeup interval is set to 0.5 seconds. We first analyze the experimental results from experiments in the high density network with 57 nodes, and then compare with the results from the low density network with 19 nodes.

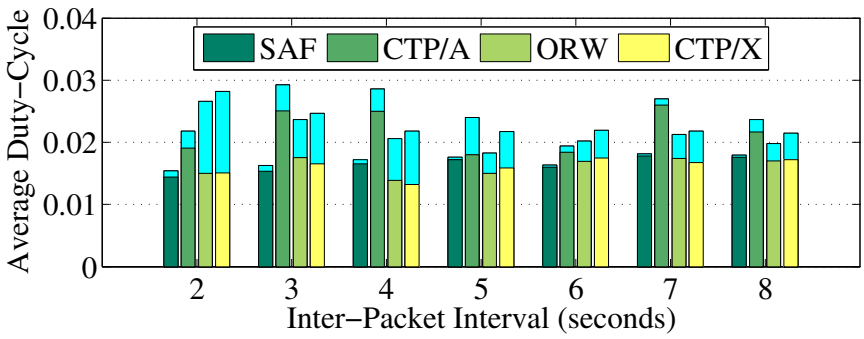


Figure 6.9: The average duty cycle of SAF, CTP/A, ORW and CTP/X with respect to different inter-packet intervals. Bars with different colors represent the duty cycle for checking the channel, whereas the box on top of each bar represents the duty cycle for data packets TX.

6.4.1.1 Performance in High Density Network

Average Duty Cycle Per Node

Duty cycle represents the ratio of the radio on/off time during the course of the experiments. Figure 6.9 presents the average duty cycle per node of the experiments conducted in the 57-node high density network with T_{ipi} ranging from 2 seconds to 8 seconds. In this figure, each group of bars represents the duty cycle of SAF, CTP/A, ORW and CTP/X respectively given the T_{ipi} in the x axis. Furthermore, each bar is divided into two parts: the top part with light color presents the duty cycle for data communication, i.e., the rate of radio on time for actually transmitting data packets, whereas the bottom part indicates the duty cycle spent on periodical channel checking. Combined together, these two parts present the overall duty cycle per node averaged over the entire network.

From Figure 6.9, it can be observed that the duty cycle for checking the wireless channel (bottom part of each bar) is similar for SAF, ORW and CTP/X, confirming that all three protocols perform channel check in the same fashion.

Compared with these three, CTP/A shows higher channel checking duty cycle in all the experiments due to the different approach (channel polling with probe packets). In general, CTP/A shows higher duty cycle than the other protocols in most of the cases, and the variations of duty cycle is also the highest.

Different from channel checking duty cycle, the data duty cycle (top part of each bar) is drastically different for the four protocols. In the case of SAF, the duty cycle used for data transmission is small and relatively stable, showing that SAF uses only a small fraction of total radio on time for the data transmission. On the other hand, CTP/X spends a significant portion of the total duty cycle to send data, resulting a much higher total duty cycle. For example, in the case of $T_{ipi} = 2$ seconds, the data duty cycle of CTP/X is 0.013, 13 times more than the 0.001 data duty cycle of SAF. Overall, the total duty cycle of CTP/X is 0.028, 87% more than the 0.016 total duty cycle of SAF. Also, the data duty cycle of CTP/X shows more variation compared with SAF. This significant reduction in terms of duty cycle is a result of the synchronous data forwarding in SAF. Due to the synchronized transmission schedule, the sender nodes using SAF can send their packets without waiting for the receivers to wake up, and consequently use the radio only for actual the data transmissions. In the case of CTP/X, the data duty cycle is much larger and shows more variations, which reflects the energy spent on idle sending as discussed in Section 6.3.1: in CTP/X, a sender needs to repeatedly send the same packet until the receiver wakes up and acknowledges the packet. This behavior introduces a period of idle sending ranging from 0 to T_w, resulting in a large and highly variable data duty cycle. Therefore, although both SAF and CTP/X show similar channel checking duty cycle, the drastically reduced data duty cycle allows SAF to consume less energy.

As to ORW, the duty cycle for channel checking is almost the same with

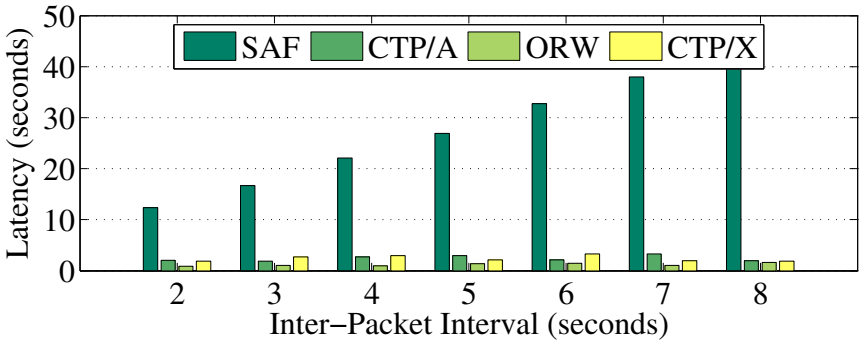


Figure 6.10: End-to-end latency of SAF, CTP/A, ORW and CTP/X with respect to inter-packet intervals. The latency of SAF is higher than the other protocols and is proportional to the inter-packet interval.

CTP/X and SAF due to the same low power listening scheme. The data duty cycle of ORW is less than CTP/X, but still much larger than SAF. This result shows that ORW is able to forward the packets faster than X-MAC as the ORW's anypath routing allows any receiver to forward the packet given that the receiver provides a viable route. However, due to the asynchronous wakeup schedule between the sender and the receiver nodes, ORW still needs to spend significant amount of time in idle sending to wait for the potential receivers. On the other hand, SAF tries to minimize the idle sending by synchronizing the sender and the potential receivers, therefore only spending minimal amount of time in idle sending.

End-to-End latency

End-to-end latency is the time needed for a packet to travel across the network from source to destination. In WSNs, it often represents the delay between the time an event is detected and the time the information of this event arrives the base station. Figure 6.10 shows the latency results of SAF, CTP with A-MAC, ORW and CTP with X-MAC.

From this figure, it is clear that the end-to-end latency of SAF is much larger than the other protocols and is almost proportional to the inter-packet interval. This large delay is introduced by the buffering and traffic shaping in SAF. As described in Section 6.2.2, the sender only sends the packets when the size of the buffer is equal or larger than the packet group size G, whereas the time needed for accumulating G packet depends on T_{ipi} and the value of G. As the result, for each packet in a group of G packets, SAF introduces a latency between 0 and $T_{ipi} \times G$ seconds. On average, the latency of SAF sending a group of G packets is $\frac{1}{2}T_{ipi} \times G$. This additional latency is the price to pay for a regulated traffic pattern, which is critical to SAF for managing a predicable wakeup/transmission schedule.

In our experiments, the packet group size G is set to 10, which means the sender nodes only start to send when 10 or more packets are in the buffer. Consequently, the sender nodes introduce an average delay of $5 \times T_{ipi}$ for each group of 10 packets, which is the main component of the end-to-end latency in the SAF experiments. For example, in the experiment with the average inter-packet interval $T_{ipi} = 2$ seconds, the end-to-end latency is 12.4 seconds, which is consistent with the above analysis.

The end-to-end latency of ORW, CTP/A and CTP/X are mostly below 3 seconds. Note that the latency of ORW is consistently lower than CTP/X, again indicating ORW spends less time on data forwarding due to the anypath routing. The latency of CTP/A is generally higher compared with ORW and CTP/X, and shows the most variations among these three protocols.

End-to-End Delivery Rate

The end-to-end delivery rate is the ratio of the number of packets received by the sink versus the total number of packets sent by the sender nodes. It indicates

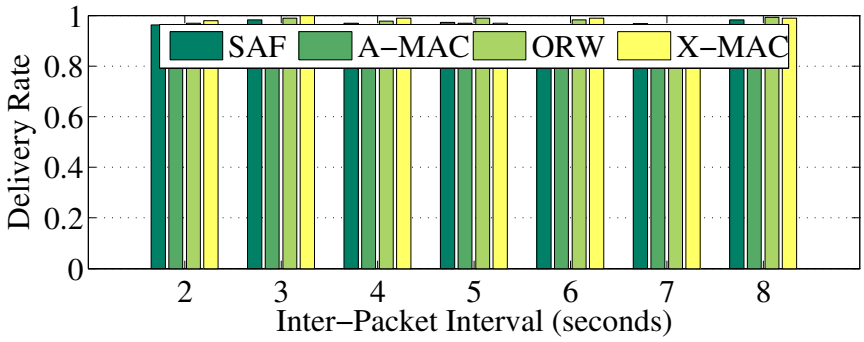


Figure 6.11: The end-to-end delivery rate of SAF, CTP/A, ORW and CTP/X with respect to inter-packet intervals.

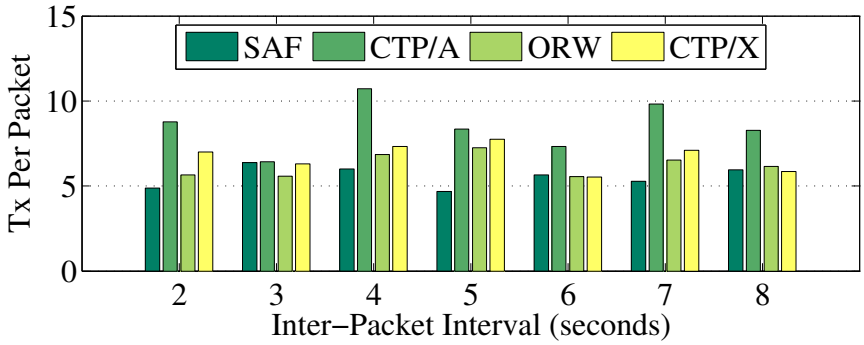


Figure 6.12: The average TX per packet of SAF, CTP/A, ORW and CTP/X with respect to inter-packet intervals.

the reliability of the data forwarding protocol. Figure 6.11 presents the end-to-end delivery rate of SAF, CTP/A, ORW and CTP/X. Based on the results, SAF, ORW and CTP/X achieve a delivery rate higher than 95% for all the experiments, whereas the delivery rate of CTP/A is close to 90%. Nevertheless, the high delivery rate indicates that all four protocols are highly reliable in terms of data forwarding.

Average Transmission Per Packet

Average transmission (TX) per packet refers to the number of radio transmission

needed to deliver a packet from source to sink. It is the sum of the hop by hop transmission attempts including retransmissions, at each hop along the path. Figure 6.12 illustrates the average TX per packet of SAF, CTP/A, ORW and CTP/X with respect to inter-packet interval.

First of all, we can see from Figure 6.12 that on average, SAF achieves lower end-to-end delivery cost compared with other three schemes. It is to be expected as the opportunistic anypath routing would enable multiple nodes with the low path cost to participate and forward the data packets. In addition, the low Tx per packet also benefits from the lazy retransmission mechanism. As described in Section 6.3.2, when a packet is not acknowledged by any potential forwarder, SAF simply put the packet back to the send queue and wait for the next wakeup, as opposed to continuous retransmission of lost packets in CTP and ORW. By employing the lazy retransmission, SAF effectively migrate the retransmission to the next wakeup interval, and therefore reduce the number of retransmissions by skipping the low link quality periods.

We can also notice that the TX per packet of ORW is slightly lower than CTP/X. Although ORW also employs anypath routing, the actual forwarding node is heavily affected by the asynchronous wakeup schedule of the potential forwarders. In other words, ORW forwards the packets through the forwarder that responds to the sender first, which is often not the node with the most routing gain. This again highlights benefit of using SAF: with synchronized batch packet transmission, all the potential forwarders are aware of the data traffic in the network, and by waking up synchronously, the node with the best routing gain will be able to respond to the sender whenever it transmits data.

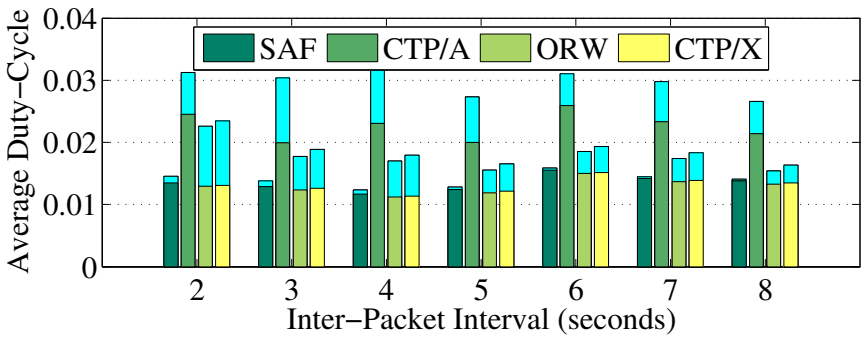


Figure 6.13: The average duty cycle of SAF, CTP/A, ORW and CTP/X with respect to different inter-packet intervals in the 19-node low density network.

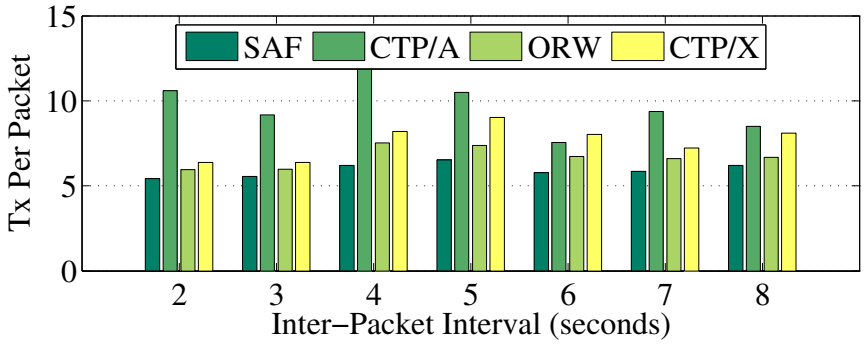


Figure 6.14: The average transmission per packet of SAF, CTP/A, ORW and CTP/X with respect to inter-packet intervals in the low density network.

6.4.1.2 Performance in Low Density Network

To study the impact of network density to the performance of SAF, we reduce the number of nodes in the network to 19 (one node in each group, see Figure 5.13) and repeat the experiments with varying inter-packet interval in this low density network. Similar to high density network experiments, the performance of the four protocols is evaluated in terms of average duty cycle, end-to-end latency, end-to-end delivery rate and average Tx count. The experimental results of average duty cycle and Tx per packet are presented in Figure 6.13 and 6.14 respectively.

The results of end-to-end latency and delivery rate are very similar to the high density network results (see Figure 6.10 and 6.11), and omitted for brevity.

Comparing the average duty cycle in low (Figure 6.13) and high (Figure 6.9) network density, we see that SAF is relatively unaffected by the different network density and show consistent duty cycle. In the case of ORW and CTP/X, it is worth noting that the data duty cycle of ORW is closer to that of CTP/X, indicating that the number of potential forwarders is limited in the low density network. Similar to the results from high density network, the overall duty cycle of CTP/A is higher than the other three protocols and shows the most variations.

Figure 6.14 shows the average TX per packet in the low density network. Compared with the results from high density network (see Figure 6.12), TX per packet of ORW CTP/X increases slightly as the number of good forwarding paths is now limited in the low density network. If the packet strays from these few good paths due to inaccurate link quality estimations and/or temporal dynamics in the path, additional transmissions might be required for retransmitting lost packets or resolving routing loops. On the other hand, TX per packet for SAF is relatively unaffected. This is because in SAF, nodes with good routing costs can always respond to the sender due to the synchronous wakeup schedule and the opportunistic multiple path routing, which leads an optimal routing topology regardless of the network density.

6.4.2 Impact of Wakeup Interval

This section focuses on the performance of SAF under varying wakeup intervals. We use a fixed inter-packet interval of 8 seconds, and conduct a series of experiments with the wakeup interval changing from 0.5 seconds to 8 seconds. The performance of SAF is evaluated by the four metrics set in the previous section,

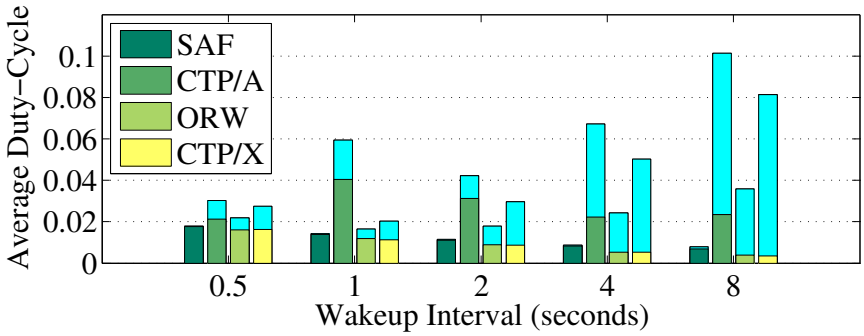


Figure 6.15: The average duty cycle of SAF, CTP/A, ORW and CTP/X wrt different wakeup intervals.

and compared with the performance of ORW, CTP/A and CTP/X.

Average Duty Cycle Per Node

Figure 6.15 shows the duty cycle of SAF, CTP/A, ORW and CTP/X in experiments done with different wakeup intervals, namely, 0.5, 1, 2, 4 and 8 seconds. Similar to Figure 6.9 and 6.9, the top part of each bar with light color presents the duty cycle for data communication, and the bottom part represents the duty cycle for checking the channel. Colors in the each bar group denote the respective protocol.

From Figure 6.15, it is clear that the duty cycle for checking the wireless channel decreases steadily as the wakeup interval increases for SAF, ORW and CTP/X. Furthermore, in the case of CTP/X, the increment of the data forwarding duty cycle also matches the increase of the wakeup interval, whereas the data duty cycle of ORW is consistently lower than CTP/X due to the anypath routing. These results confirmed our analysis in Section 6.3.1. In particular, when the wakeup interval is large, the data forwarding duty cycle becomes the dominating factor in the total duty cycle of ORW and CTP/X.

In the case of SAF, the duty cycle for channel checking is slightly larger than

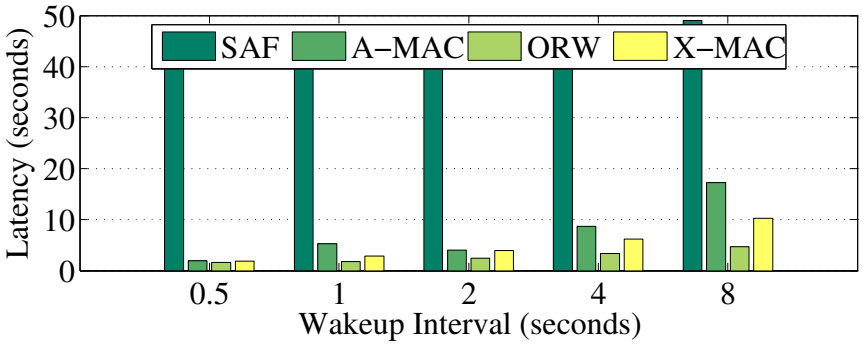


Figure 6.16: The end-to-end latency of SAF, CTP/A, ORW and CTP/X with respect to wakeup intervals.

ORW or CTP/X as SAF performs additional checks every transmission interval T_{ti}, but the data duty cycle of SAF is almost unaffected by the wakeup interval due to the synchronous wakeup schedule. In this case, the duty cycle reduction using SAF is substantial: for example, when $T_w = 8$ second, the total duty cycle of CTP is 0.081, 20.2 times more SAF, and data duty cycle of CTP/X is more than 250 times higher than SAF.

End-to-End Latency

Figure 6.16 presents the end-to-end latency of the four protocols. Due to SAF's buffering and traffic shaping, SAF's latency is generally higher than 40 seconds as SAF needs about 80 seconds to fill a group of 10 packets. On the other hand, due to the synchronized wakeup/transmission schedule, the delay of sending batch of packets through the network is minimal compared with the delay introduced by the packet grouping, except for experiment with wakeup interval of 8 seconds. In this particular experiments, several senders are out of sync with their forwarders due to clock drift and the long wakeup interval, which causes addition delay as seen in Figure 6.16, $T_w = 8$ seconds.

The latency of ORW and CTP/X increases steadily as the wakeup interval

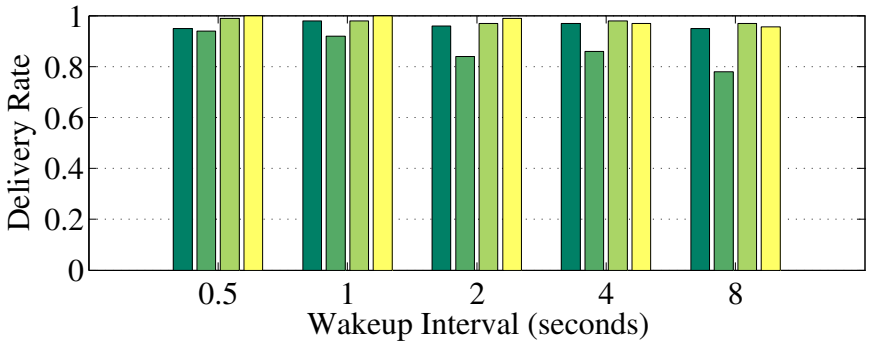


Figure 6.17: The end-to-end delivery rate of SAF, CTP/A, ORW and CTP/X.

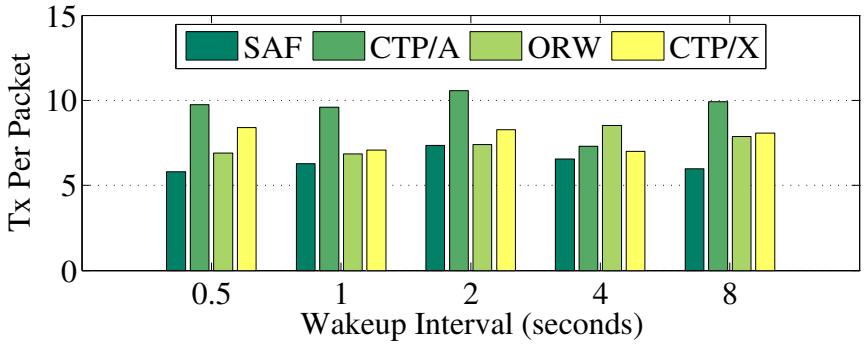


Figure 6.18: The average Tx per packet of SAF, CTP/A, ORW and CTP/X.

increase from 0.5 to 8 seconds. Again we see that the latency of ORW is consistently lower than CTP/X, showing the advantage of anypath routing in ORW. CTP/A's delay also increases with the wakeup interval, but it is generally much higher than ORW and CTP/X.

End-to-End Delivery Rate

Almost all the four protocols have a end-to-end delivery rate higher than 90%, with the exception of CTP/A as shown in Figure 6.17. CTP/A exhibits decreasing delivery rate as the wakeup interval increases, implying that CTP/A is less reliable with longer wakeup intervals.

Average Transmission Per Packet

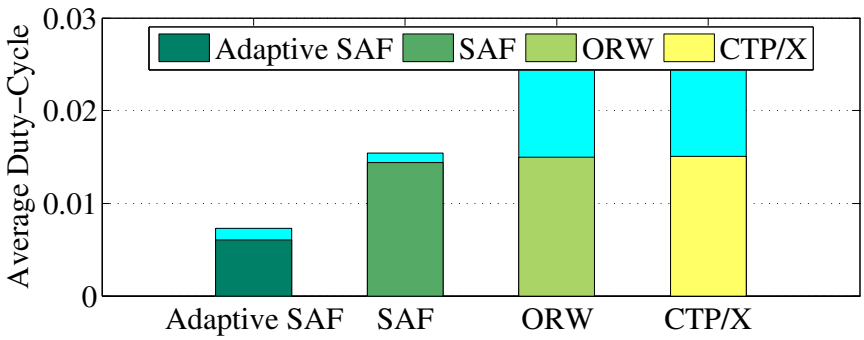


Figure 6.19: The average duty-cycle of Adaptive SAF, SAF, ORW and CTP/X.

Figure 6.18 shows the results of the average TX costs per packet for the four protocols. Similar to the TX per packet results from Figure 6.12 and 6.14, the average TX per packet of CTP/X is higher than SAF in all experiments, whereas ORW shows slightly lower TX per packet than CTP/X C. Also note that the transmission cost reduction is more pronounced when the wakeup interval is larger. For example, when $T_w = 8$ seconds, the Tx per packet of SAF is 35% less than CTP/X. This highlights the advantage of opportunistic anypath routing: when the wakeup interval is large, the link estimation is less accurate as the interval between packets is also large. The skewed link estimation might lead to inefficient forwarding paths for ORW and CTP, but it is less a problem for SAF as it explores multiple potential forwarders during the packet batch transmission. As a result, the multiple paths selected by SAF are generally better than the single path selected by CTP or ORW.

6.4.3 Adaptive Wakeup Interval

SAF tries to minimize the node duty cycle by using a synchronous transmission schedule, batch packet transmission and anypath routing. However, another significant part of the total energy consumption is the channel checking. This is

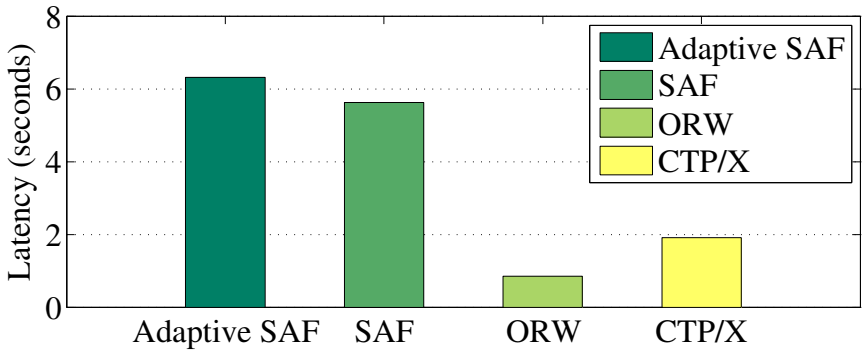


Figure 6.20: The end-to-end latency of Adaptive SAF, SAF, ORW and CTP/X.

controlled by the wakeup interval and it is not modified by SAF.

Ideally, a node should only wake up when there are data packets to send or receive. Therefore, one possibility to reduce the channel checking energy costs is to set the wakeup interval adaptively such that the node only turns the radio when there is radio activities and/or it has packets to send. However, this requires knowledge of the traffic pattern, i.e., when the next packet will be sent to it, as well as the wakeup times of its down stream nodes. Therefore, this adaptive wakeup scheme is hard to implement in an asynchronous MAC protocol, but it is possible to implement in SAF as the traffic shaping can regulate the traffic into a predictable pattern, whereas the locally synchronized transmission schedule can provide the necessary time synchronization information.

We designed and implemented a simple adaptive wakeup scheme for SAF. As described in Section 6.2.2, SAF tries to buffer packets from the application layer such that it can send them at regular intervals, therefore as long as a node is synchronized with its neighbor, it only needs to wakeup at the sender node's schedule to receive a potential packet, and to send/forward its own packet. In the original design, SAF only keeps a sender timer for receiving packets from the sender node, and sends the buffered batch of packets when the transmis-

sion timer fires, which ticks at the same time as the global wakeup interval T_w. Therefore, to implement an adaptive wakeup interval, a node just need to set the T_w based on the actual packet transmission interval T_{ti}. In our design, we use an EWMA filter to calculate the smoothed interval between the last two packet transmissions, and limit the possible wakeup intervals to a small set of values $SI = \{0.125, 0.25, 0.5, 1, 2, 4, 8\}$ seconds. The next T_w is the closest value in the SI set that is smaller than the smoothed interval. For example, if the smoothed T_w is between 0.5 to 1 seconds, the next T_w will be 0.5 seconds.

T_w also changes based on the buffer size at each wake up. Conceptually, if the buffer is close to full, T_w should be shorter to accommodate the high traffic rate, whereas when the buffer is almost empty, T_w should be increased to avoid unnecessary wakeups. In our design, if the buffer size is greater than $1.5 \times G$, SAF will set T_w to the next value in SI smaller than the current T_w unless T_w is already the smallest value (0.125 seconds). On the other hand, if the buffer size is less than $0.5 \times G$, we will set T_w to the next larger value in SI. Effectively, we half the previous T_w if the buffer is close to full, and vice versa.

We run a few experiments with SAF using the adaptive wakeup interval (hence called Adaptive SAF), SAF with fixed wakeup interval, ORW and CTP/X in the 57-node high density network conditions. The inter-packet interval is set to 2 seconds, and the initial wakeup interval, i.e., T_w at the beginning of the experiments, is set to 0.5 seconds. We use a different packet group size G of 2 as opposed to 10 used in Section 6.4. The results in terms of duty cycle, latency and Tx per packet are presented in Figures 6.19, 6.20 and 6.21.

Figure 6.19 shows that Adaptive SAF indeed reduces the duty cycle for channel checking by almost 50% compared with non-adaptive SAF, ORW and CTP/X. The data duty cycle of Adaptive SAF is slightly larger than non-adaptive SAF

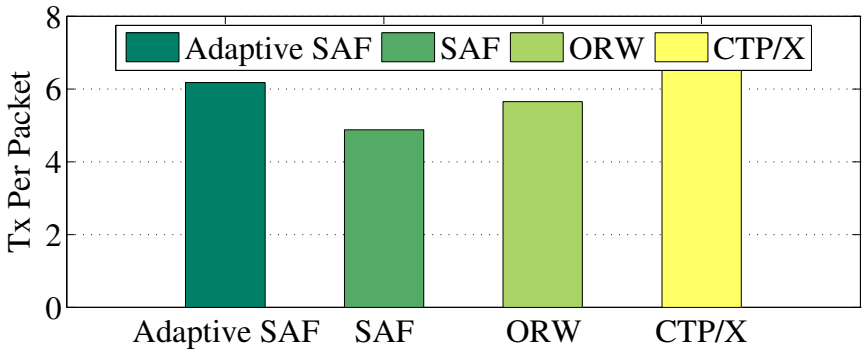


Figure 6.21: The average TX per packet Adaptive SAF, SAF, ORW and CTP/X.

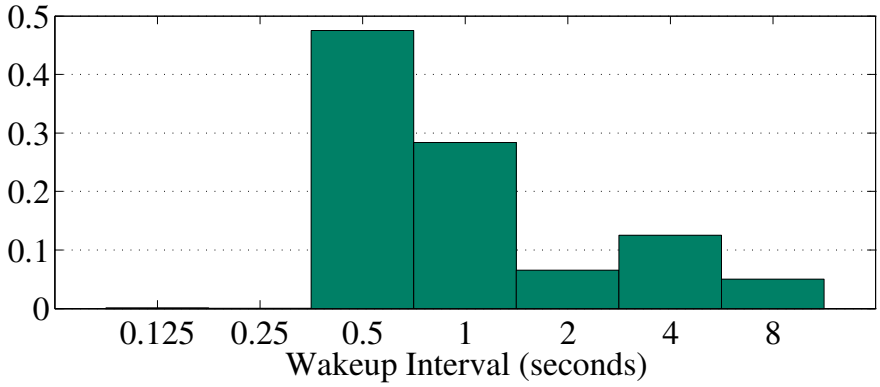


Figure 6.22: The distribution of wakeup intervals.

due to occasional synchronization problems, but it is still much smaller than ORW and CTP/X. Overall, the total duty cycle of Adaptive SAF is 0.007, 50% lower than non-adaptive SAF and 70% lower than CTP/X. In terms of end-to-end latency and Tx per packet, Adaptive SAF's performance is similar to non-adaptive, indicating adaptive SAF does not cause significant overhead.

To further analyze the wakeup interval of Adaptive SAF, we plot a histogram of the actual T_w seen over the course of a 30 minute experiment (see Figure 6.22. It can be seen that about 50% of the wakeup intervals are set to 0.5 seconds, which is the initial T_w, whereas more than half of the wakeup intervals are distributed

between 1 to 8 seconds, which explains the cause of the duty cycle reduction seen in Figure 6.19: with adaptive SAF, the aggregated wakeup interval is larger than the default 0.5 second wakeup interval, and therefore the nodes using Adaptive SAF wake up less often and spend less energy on unnecessary channel checking. It is also interesting to notice that there is a raise at $T_w = 4$ seconds, indicating that a substantial number of nodes send their packets at the rate of 4 seconds. Given the 2 seconds inter-packet interval and packet group size of 2, we can conclude that these are sender nodes and the nodes with only one sender at the edge of the network.

6.4.4 Summary

There are a couple of conclusions based on our extensive evaluation. First, SAF can drastically reduce the radio duty cycle used in data transmissions. In some cases, the reduction on data duty cycle is more than 13 times, and the overall duty cycle of the entire network is reduce by 87%. Second, SAF can also significantly reduce the number of transmissions needed for delivering a packet with the opportunistic anypath routing. Our experimental results show that the delivery cost can be reduced by more than 30%. Finally, in order to achieve these results, SAF sacrifices the latency for the reduction duty cycle and delivery cost. Due to its traffic shaping, SAF introduces a much larger delay compared with the send-when-available policy employed by the other routing protocols.

6.5 Discussion

6.5.1 Integration of SAF and 4C/TALENT

SAF is an ideal routing protocol for prediction based link estimators due to its traffic shaping mechanism and burst traffic pattern. As detailed in Section 6.2, sender nodes in SAF buffer the data packets to maintain a fixed transmission schedule, and share the transmission schedule with the receivers. Therefore, with prediction models trained specifically for the transmission schedule, 4C can potentially predict the link quality for the next scheduled batch packet transmission and inform SAF about the future link quality. With the predicted link quality information of next packet transmission, SAF can make informed decision on whether to send the buffered packets as scheduled to avoid packet losses over low quality periods.

Furthermore, short term link estimator such as TALENT is also useful during the batch packet transmission. As the packets are always transmitted in batch, the inter-packet interval within a packet batch is sufficiently small such that TALENT can accurately predict the temporal link quality variations during the transmission. Thus, the receiver nodes can potentially use the quality information to decide whether to quit from an on-going transmission due to low link quality. We leave the integration of SAF and 4C/TALENT to future work.

6.5.2 Limitations of Opportunistic Routing

The main disadvantage of opportunistic routing is that it selects the forwarders greedily. Although this approach is effective for selecting nodes with better routing cost and improving routing progress compared with traditional routing, opportunistic routing does not guarantee finding the optimal forwarding path cur-

rently available in the network. It is possible that an overhearing node decides to forward packets for the sender, but in fact the original sender's parent can provide a better path due to the availability of a new link. In this case, the delay in the link estimation will cause SAF to select a sub-optimal path. Nonetheless, we believe that our approach improves the traditional approach under various network conditions.

Another subtle point is that by using multiple forwarders and bitmap acknowledgment, the sender is unaware of packet reception status of the parent node, and therefore misses the opportunity for estimating the link quality to the parent using data packets. Without the updated link quality estimation, the sender node may send packets to the parent through low quality links, causing packet loss and excessive routing changes. One possibility to solve this problem is to let the parent node always notify the sender nodes about the link quality. We leave this improvement to future work.

6.5.3 Trade-off between Duty Cycle and Latency

SAF takes an unique angle in the design space to improve the radio duty cycle of WSNs: it tries to shape the uncertain traffic into predictable patterns, and takes advantages of the predictable traffic pattern to reduce the radio duty cycle by employing synchronized batch packet transmission and anypath routing. However, the traffic shaping comes with the cost of the end-to-end latency. By actively buffering the packets for batch transmissions, SAF effectively sacrifices the latency to improve the radio duty cycle. The high latency may limit the use of SAF in event detection applications that require low delay between the source and destination. However, SAF is perfectly suitable for applications that need periodic sensing such as environment monitoring and data gathering. These

applications do not emphasize on minimizing the latency, and SAF can be used to reduce energy usage as well as extend system lifetime. Based on the results presented in Section 6.4, the total energy consumption can be reduced by more than 70% with the cost of increased latency (from about 1 seconds to 5 seconds).

CHAPTER 7

Conclusion

This work addresses two main challenges in the radio communication protocol design for energy constrained sensor networks, namely, wireless link quality estimation under varying network conditions and efficient data forwarding in duty-cycled networks. We first offer an comprehensive study on the routing performance of existing link estimators in order to highlight the challenges in link quality estimation of energy-constrained sensor networks. We then propose 4C, a novel link estimator that applies link quality *prediction* along with link estimation. Our approach is data-driven and consists of three steps: data collection, offline modeling and online prediction. With the collected data from the deployment site, we train prediction models that take a combination of PRR and the physical layer information as input, and output the reception probability of the next packet. Analytical and empirical results show that logistic regression classifier can accurately predict the link quality with the additional advantage of having the small computational cost. We then further extend 4C to TALENT, a self-learning, plug-and-play estimator to predict the quality of a wireless link in the near future using a combination of packet and physical level quality indicators. When using TALENT together with CTP, experimental results show that on many different environments TALENT increases the delivery efficiency significantly in comparison to state-of-the-art link quality estimators.

The author then propose SAF, a data forwarding protocol in order to utilize

TALENT in duty-cycled networks. The SAF is a cross-layer routing protocl that applies opportunistic routing techniques and trades off latency for a significant increase in data transmission efficiency with low duty cycle. SAF synchronizes the sender and receiver nodes locally and therefore eliminates the need of idle sending/listening in typical duty cycle techniques such as Low Power Listening. Furthermore, it takes the advantage of the synchronous wake-up schedule to send the packets to multiple nodes, such that the recipient node(s) with the largest routing gain can forward the packet. This synchronous schedule also enables the utilization of the short term link estimators such TALENT.

There are several possibilities for future improvements. In terms of link estimation, as shown in Section 5.4, the prediction approach of TALENT can be potentially used in 802.11 networks. Although TALENT is designed to improve the routing cost and reduce the energy consumption for low-power wireless sensor networks, it also applies to ad-hoc networks for better route selection, higher throughput and lower latency. With a predictive link estimator similar to TALENT, the transmitting nodes in ad-hoc networks can better identify the short temporal link quality in the near future and select the node with the best routing cost to forward the data packets, forming a better routing topology in terms of throughput and/or end-to-end latency. Another possibility is to further improve SAF by integrating link quality predictor such as 4C and TALENT. As discussed in Section 6.5, SAF is ideal for utilizing the prediction based link estimators due to its traffic shaping mechanism and burst traffic pattern. By utilizing 4C to predict the future link quality for the next scheduled batch packet transmission, and using TALENT to quickly determine the short term link quality during batch packet transmissions, SAF can further reduce the radio communication cost by avoid packet losses over low quality periods. Refining these approaches will not only advance the efficiency of radio communication for wireless sensor networks,

but also benefits the wireless communication in general.

References

[ABB04] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. "Link-level measurements from an 802.11b mesh network." *SIGCOMM Comput. Commun. Rev.*, **34**(4):121–132, 2004.

[ACH97] H. Amin, K. M. Curtis, and B. R. Hayes-Gill. "Piecewise linear approximation applied to nonlinear function of a neural network." *IEE Proceedings - Circuits, Devices and Systems*, **144**(6):313–317, 1997.

[ALA98] Luís B. Almeida, Thibault Langlois, José D. Amaral, and Alexander Plakhov. *Parameter adaptation in stochastic optimization*, pp. 111–134. Cambridge University Press, New York, NY, USA, 1998.

[ALB09] Muhammad Hamad Alizai, Olaf Landsiedel, Jó Ágila Bitsch Link, Stefan Götz, and Klaus Wehrle. "Bursty Traffic Over Bursty Links." In *SenSys '09*, pp. 71–84. ACM, 2009.

[Ber12] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 4th edition, 2012.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BKM12] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zú niga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. "Radio link quality estimation in wireless sensor networks: A survey." *ACM Trans. Sen. Netw.*, **8**(4):34:1–34:33, September 2012.

[BKY10] Nouha Baccour, Anis Koubâ, Habib Youssef, Maissa Ben Jamâa, Denis do Rosário, Mário Alves, and Leandro Becker. "F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks." In *EWSN '10*, volume 5970 of *Lecture Notes in Computer Science*, pp. 240–255. Springer Berlin / Heidelberg, 2010.

[BL04] Léon Bottou and Yann LeCun. "Large Scale Online Learning." In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[BM02] Léon Bottou and Noboru Murata. "Stochastic Approximations and Efficient Learning." In *The Handbook of Brain Theory and Neural Networks, Second edition,*. The MIT Press, Cambridge, MA, 2002.

[BM05] Sanjit Biswas and Robert Morris. "ExOR: opportunistic multi-hop routing for wireless networks." *SIGCOMM Comput. Commun. Rev.*, **35**(4):133–144, 2005.

[BMA09] Kevin Bauer, Damon McCoy, Eric W. Anderson, Dirk Grunwald, and Douglas C. Sicker. "CRAWDAD data set cu/rssi (v. 2009-05-28)." Downloaded from http://crawdad.cs.dartmouth.edu/cu/rssi, May 2009.

[Bra97] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." *Pattern Recognition*, **30**(7):1145–1159, 1997.

[BYA06] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks." In *SenSys '06*, pp. 307–320. ACM, 2006.

[BZV10] C. A. Boano, Zúñiga, M. A., T. Voigt, A. Willig, and K. Römer. "The Triangle Metric: Fast Link Quality Estimation for Mobile Wireless Sensor Networks." In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pp. 1–7, August 2010.

[CB01] George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Press, 2001.

[CC2] CC2420. "ChipCon CC2420 2.4 GHz IEEE 802.15.4 ZigBee-ready RF Transceiver, http://www.ti.com/product/cc2420, accessed Feb. 2012.".

[CCD11] M. Ceriotti, M. Corra, L. D'Orazio, R. Doriguzzi, D. Facchin, S. T. Guna, G. P. Jesi, R. Lo Cigno, L. Mottola, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele. "Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels." In *IPSN '11*, pp. 187–198, 2011.

[CE03] Alberto Cerpa and Deborah Estrin. "SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments." Technical Report 0021, UCLA, 2003.

[CJK07] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. "Trading structure for randomness in wireless opportunistic routing." *SIGCOMM Comput. Commun. Rev.*, **37**(4):169–180, August 2007.

[CRA08] CRAWDAD. "CRAWDAD: Community Resource for Archiving Wireless Data At Dartmouth." http://crawdad.cs.dartmouth.edu, 2008.

[CWK] Alberto Cerpa, Jennifer Wong, Louane Kuang, Miodrag Potkonjak, and Deborah Estrin. "Statistical Model of Lossy Links in Wireless Sensor Networks." In *IPSN '05*.

[CWP05] Alberto Cerpa, Jennifer Wong, Miodrag Potkonjak, and Deborah Estrin. "Temporal Properties of Low Power Wireless Links: Modeling and Implications on Multi-Hop Routing." In *MobiHoc '05*, pp. 414–425. ACM, 2005.

[DAB03] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. "A High-Throughput Path Metric for Multi-Hop Wireless Routing." In *MobiCom '03*, pp. 134–146. ACM, 2003.

[Dav07] Kevin Klues David Moss, Jonathan Hui. "TinyOS TEP 105, Low Power Listening.", 2007.

[DCL11] Manjunath Doddavenkatappa, Mun Choon Chan, and Ananda A. L. "Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed." In *7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM '11)*, 2011.

[DDC10] Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan Mike Liang, and Andreas Terzis. "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless." In *SenSys '10*, pp. 1–14, New York, NY, USA, 2010. ACM.

[DGV11] H. Dubois-Ferrière, Matthias Grossglauser, and M. Vetterli. "Valuable Detours: Least-Cost Anypath Routing." *Networking, IEEE/ACM Transactions on*, **19**(2):333–346, 2011.

[DL03] Tijs van Dam and Koen Langendoen. "An adaptive energy-efficient MAC protocol for wireless sensor networks." In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pp. 171–180, New York, NY, USA, 2003. ACM.

[DPZ04] Richard Draves, Jitendra Padhye, and Brian Zill. "Comparison of routing metrics for static multi-hop wireless networks." *SIGCOMM Comput. Commun. Rev.*, **34**(4):133–144, August 2004.

[DS83] J. Dennis and R. Schnabel. *Numerical Methods For Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., 1983.

[DS05] David M. Doolin and Nicholas Sitar. "Wireless sensors for wildfire monitoring." In *Proceedings of SPIE Symposium on Smart Structures and Materials/ NDE 2005*, 2005.

[DYM] TYMO: TinyOS implementation of the DYMO protocol. "http://www.tinyos.net/tinyos-2.x/tos/lib/net/tymo/. Accessed Feb. 2013.".

[ED04] Amre El-Hoiydi and Jean-Dominique Decotignie. "WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks." In *Algorithmic Aspects of Wireless Sensor Networks*, volume 3121 of *Lecture Notes in Computer Science*, pp. 18–31. Springer Berlin Heidelberg, 2004.

[ESK11] Emre Ertin, Nathan Stohs, Santosh Kumar, Andrew Raij, Mustafa al'Absi, and Siddharth Shah. "AutoSense: unobtrusively wearable sensor suite for inferring the onset, causality, and consequences of stress in the field." In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pp. 274–287, New York, NY, USA, 2011. ACM.

[FGJ07] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Levis. "Four-Bit Wireless Link Estimation." In *HotNets VI*. ACM, 2007.

[FHL08] Károly Farkas, Theus Hossmann, Franck Legendre, Bernhard Plattner, and Sajal K. Das. "Link quality prediction in mesh networks." *Computer Communications*, **31**(8):1497–1512, 2008.

[FHR06] Károly Farkas, Theus Hossmann, Lukas Ruf, and Bernhard Plattner. "Pattern matching based link quality prediction in wireless mobile ad hoc networks." In *MSWiM '06*, pp. 239–246. ACM, 2006.

[GBP10] Carles Gomez, Antoni Boix, and Josep Paradells. "Impact of LQI-based routing metrics on the performance of a one-to-one routing protocol for IEEE 802.15.4 multihop networks." *EURASIP J. Wirel. Commun. Netw.*, **2010**:6:1–6:20, February 2010.

[GFJ09] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. "Collection tree protocol." In *SenSys '09*, pp. 1–14. ACM, 2009.

[GH07] Yu Gu and Tian He. "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links." In *Proceedings*

of the 5th international conference on Embedded networked sensor systems, SenSys '07, pp. 321–334, New York, NY, USA, 2007. ACM.

[GSA06] C. Gomez, P. Salvatella, O. Alonso, and J. Paradells. "Adapting AODV for IEEE 802.15.4 mesh sensor networks: theoretical discussion and performance evaluation in a real environment." In *World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006. International Symposium on a*, pp. 9 pp.–170, 2006.

[GYH04] O. Gnawali, M. Yarvis, J. Heidemann, and R. Govindan. "Interaction of retransmission, blacklisting, and routing metrics for reliability in sensor network routing." In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pp. 34–43, 2004.

[HDB96] Martin T. Hagan, Howard B. Demuth, and Mark H. Beale. *Neural Network Design*. CT: Thomson Learning, 1996.

[KGD07] Anand Kashyap, Samrat Ganguly, and Samir R. Das. "A measurement-based approach to modeling link capacity in 802.11-based wireless networks." In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, pp. 242–253, New York, NY, USA, 2007. ACM.

[KGS07] Sanjit Krishnan Kaul, Marco Gruteser, and Ivan Seskar. "CRAWDAD data set rutgers/noise (v. 2007-04-20)." Downloaded from http://crawdad.cs.dartmouth.edu/rutgers/noise, April 2007.

[KJD09] Ankur Kamthe, Lun Jiang, Matt Dudys, and Alberto Cerpa. "SCOPES: Smart Cameras Object Position Estimation System." In *In the Proceedings of the 6th European Conference on Wireless Sensor Networks(EWSN 2009)*, pp. 279–295, Cork, Ireland, February 2009. Springer-Verlag.

[KLK07] Branislav Kusy, Akos Ledeczi, and Xenofon Koutsoukos. "Tracking mobile nodes using RF Doppler shifts." In *Proceedings of the 5th international conference on Embedded networked sensor systems*, SenSys '07, pp. 29–42, New York, NY, USA, 2007. ACM.

[KN01] Minkyong Kim and Brian Noble. "Mobile network estimation." In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pp. 298–309, New York, NY, USA, 2001. ACM.

[LBO98] Yann Le Cun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. "Efficient Backprop." In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag, 1998.

[LC11] Tao Liu and Alberto Cerpa. "Foresee (4C): Wireless link prediction using link features." In *IPSN '11*, pp. 294–305, April 2011.

[LGD12] Olaf Landsiedel, Euhanna Ghadimi, Simon Duquennoy, and Mikael Johansson. "Low power, low delay: opportunistic routing meets duty cycling." In *IPSN '12*, pp. 185–196, NY, USA, 2012. ACM.

[LM07] I. Leontiadis and C. Mascolo. "GeOpps: Geographical Opportunistic Routing for Vehicular Networks." In *WoWMoM '07*, pp. 1–6, 2007.

[LMH03] Dhananjay Lai, Arati Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu, and A. Keshavarzian. "Measurement and characterization of link quality metrics in energy constrained wireless sensor networks." In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 1, pp. 446–452 Vol.1, 2003.

[LMP04] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler. "TinyOS: An operating system for sensor networks." In *in Ambient Intelligence*. Springer Verlag, 2004.

[LSZ04] Ting Liu, Christopher M. Sadler, Pei Zhang, and Margaret Martonosi. "Implementing software on resource-constrained mobile sensors: experiences with Impala and ZebraNet." In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pp. 256–269, New York, NY, USA, 2004. ACM.

[LYH05] Yuan Li, Wei Ye, and John Heidemann. "Energy and Latency Control in Low Duty Cycle MAC Protocols." In *Proceedings of the IEEE Wireless Communications and Networking Conference*, New Orleans, LA, USA, March 2005.

[MCP02] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. "Wireless sensor networks for habitat monitoring." In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pp. 88–97, New York, NY, USA, 2002. ACM.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw Hill Higher Ed., 1997.

[ML08] David Moss and Philip Levis. "BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking." Technical Report SING-08-00, Stanford University, 2008.

[MLT08] Razvan Musaloiu-E., Chieh-Jan Mike Liang, and Andreas Terzis. "Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks." In *IPSN '08*, pp. 421–432, Washington, DC, USA, 2008. IEEE Computer Society.

[Mot] Moteiv Corporation. "TMote Sky Datasheet, http://www.snm.ethz.ch/Projects/TmoteSky, accessed Jan. 2013.".

[Mul] MultihopLQI. "http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI. Accessed Feb. 2012.".

[NDL12] Shahriar Nirjon, Robert F. Dickerson, Qiang Li, Philip Asare, John A. Stankovic, Dezhi Hong, Ben Zhang, Xiaofan Jiang, Guobin Shen, and Feng Zhao. "MusicalHeart: a hearty way of listening to music." In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pp. 43–56, New York, NY, USA, 2012. ACM.

[PHC04] Joseph Polastre, Jason Hill, and David Culler. "Versatile low power media access for wireless sensor networks." In *SenSys '04*, pp. 95–107, New York, NY, USA, 2004. ACM.

[PK00] G. J. Pottie and W. J. Kaiser. "Wireless integrated network sensors." *Commun. ACM*, **43**(5):51–58, 2000.

[PSC05] J. Polastre, R. Szewczyk, and D. Culler. "Telos: enabling ultra-low power wireless research." In *Information Processing in Sensor Networks, 2005. IPSN 2005.*, pp. 364–369, April 2005.

[PSM04] Joseph Polastre, Robert Szewczyk, Alan Mainwaring, David Culler, and John Anderson. "Analysis of wireless sensor networks for habitat monitoring." pp. 399–423. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[PTD11] Bogdan Pavković, Fabrice Theoleyre, and Andrzej Duda. "Multipath opportunistic RPL routing over IEEE 802.15.4." In *MSWiM '11*, pp. 179–186, New York, NY, USA, 2011. ACM.

[Rap01] Theodore Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2001.

[RAR08] Michele Rondinone, Junaid Ansari, Janne Riihijärvi, and Petri Mähönen. "Designing a reliable and stable link quality metric for wireless sensor networks." In *Proceedings of the workshop on Real-world wireless sensor networks*, REALWSN '08, pp. 6–10, New York, NY, USA, 2008. ACM.

[REW11] Christian Renner, Sebastian Ernst, Christoph Weyer, and Volker Turau. "Prediction Accuracy of Link-Quality Estimators." In *Proceedings of the 8th European Conference on Wireless Sensor Networks (EWSN'11)*, February 2011.

[RL09] Tal Rusak and Philip Levis. "Burstiness and scaling in the structure of low-power wireless links." *SIGMOBILE Mob. Comput. Commun. Rev.*, **13**:60–64, June 2009.

[RMR06] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. "Measurement-based models of delivery and interference in static wireless networks." *SIGCOMM Comput. Commun. Rev.*, **36**(4):51–62, August 2006.

[RWA08] Injong Rhee, A. Warrier, M. Aia, Jeongki Min, and M. L. Sichitiu. "Z-MAC: A Hybrid MAC for Wireless Sensor Networks." *Networking, IEEE/ACM Transactions on*, **16**(3):511–524, 2008.

[SCL07] M. Senel, K. Chintalapudi, Dhananjay Lal, A. Keshavarzian, and E. J. Coyle. "A Kalman Filter Based Link Quality Estimation Scheme for Wireless Sensor Networks." In *GLOBECOM '07*, pp. 875–880. IEEE, 2007.

[SDT10] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. "An empirical study of low-power wireless." *ACM Trans. Sen. Netw.*, **6**(2):16:1–16:49, March 2010.

[SGD09] Yanjun Sun, Omer Gurewitz, Shu Du, Lei Tang, and David B. Johnson. "ADB: an efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks." In *In Sensys '09: Proceedings of the 7 th International Conference on Embedded Networked Sensor Systems*, pp. 43–56, 2009.

[SGJ08] Yanjun Sun, Omer Gurewitz, and David B. Johnson. "RI-MAC: A Receiver Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks." In *SenSys '08*, 2008.

[SIV09] Gunnar Schaefer, François Ingelrest, and Martin Vetterli. "Potentials of Opportunistic Routing in Energy-Constrained Wireless Sensor Networks." In *Proceedings of the 6th European Conference on Wireless Sensor Networks*, EWSN '09, pp. 118–133, Berlin, Heidelberg, 2009. Springer-Verlag.

[SKA] Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. "The β-factor: measuring wireless link burstiness." In *SenSys '08*. ACM.

[SKH06] Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. "Experimental study of concurrent transmission in wireless sensor networks." In *SenSys '06*, pp. 237–250. ACM, 2006.

[SL06] Kannan Srinivasan and Philip Levis. "RSSI is Under Appreciated." In *EmNets '06*, 2006.

[Sta05] Crossbow Stargate. "Crossbow Technology, Stargate Gateway Sensor.", 2005.

[TC05] Gilman Tolle and David Culler. "Design of an application-cooperative management system for wireless sensor networks." In *EWSN '05*, 2005.

[TSG11] Lei Tang, Yanjun Sun, O. Gurewitz, and D. B. Johnson. "PW-MAC: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks." In *INFOCOM '11*, pp. 1305–1313, 2011.

[WIS09] Intel WISP. "Intel Research, Intel WISP RFID Enabled Sensor.", 2009.

[WLJ06] Geoff Werner, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. "Fidelity and yield in a volcano monitoring sensor network." In *OSDI '06*, 2006.

[WMP07] Yong Wang, Margaret Martonosi, and Li-Shiuan Peh. "Predicting link quality using supervised learning in wireless sensor networks." *ACM SIGMOBILE Mobile Computing and Communications Review*, **11**(3):71–83, 2007.

[WSW05] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. "MoteLab: a wireless sensor network testbed." In *IPSN '05*, p. 68. IEEE, 2005.

[WTC03] Alec Woo, Terence Tong, and David Culler. "Taming the underlying challenges of reliable multihop routing in sensor networks." In *SenSys '03*, pp. 14–27. ACM, 2003.

[XRC04] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. "A wireless sensor network For structural monitoring." In *SenSys '04*, pp. 13–24. ACM, 2004.

[YHE02] Wei Ye, J. Heidemann, and D. Estrin. "An energy-efficient MAC protocol for wireless sensor networks." In *INFOCOM '02*, volume 3, pp. 1567–1576, 2002.

[YSH06] Wei Ye, Fabio Silva, and John Heidemann. "Ultra-low duty cycle MAC with scheduled channel polling." In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pp. 321–334, New York, NY, USA, 2006. ACM.

[ZG03] Jerry Zhao and Ramesh Govindan. "Understanding packet delivery performance in dense wireless sensor networks." In *SenSys '03*, pp. 1–13. ACM, 2003.

[ZHK04] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. "Impact of radio irregularity on wireless sensor networks." In *MobiSys '04*, pp. 125–138. ACM, 2004.

[ZK04] Marco Zuniga and Bhaskar Krishnamachari. "Analyzing the transitional region in low power wireless links." In *SECON '04*, pp. 517–526. IEEE, 2004.

[ZLY07] Kai Zeng, Wenjing Lou, Jie Yang, and Donald R. Brown,III. "On throughput efficiency of geographic opportunistic routing in multihop wireless networks." *Mob. Netw. Appl.*, **12**(5):347–357, December 2007.

[ZSC11] Chengjie Zhang, Affan Syed, Young Cho, and John Heidemann. "Steam-powered sensing." In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pp. 204–217, New York, NY, USA, 2011. ACM.