Data-driven Link Quality Prediction Using Link Features

Tao Liu and Alberto E. Cerpa, University of California, Merced

As an integral part of reliable communication in wireless networks, effective link estimation is essential for routing protocols. However, due to the dynamic nature of wireless channels, accurate link quality estimation remains a challenging task. In this paper, we 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. The data collection step involves gathering link quality data, and based on our analysis of the data, we propose a set of guidelines for the amount of data to be collected in our experimental scenarios. The modeling step includes offline prediction model training and selection. We present three prediction models that utilize different machine learning methods, namely, naive Bayes classifier, logistic regression and artificial neural networks. Our models take a combination of PRR and the physical layer information, i.e., Received Signal Strength Indicator (RSSI), Signal to Noise Ratio (SNR) and Link Quality Indicator (LQI) as input, and output the success probability of delivering the next packet. From our analysis and experiments, we find that logistic regression works well among the three models with small computational cost. Finally, the third step involves the implementation of 4C, a receiver-initiated online link quality prediction module that computes the short temporal link quality. We conducted extensive experiments in the Motelab and our local indoor testbeds, as well as an outdoor deployment. Our results with single and multiple senders experiments show that with 4C, CTP improves the average cost of delivering a packet by 20% to 30%. In some cases, the improvement is larger than 45%.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture & Design—*Wireless communication*

General Terms: Design, Measurement, Performance

Additional Key Words and Phrases: Link quality estimation, Link quality prediction

ACM Reference Format:
Liu, T. and Cerpa, A., 2013, Data-driven Link Quality Prediction Using Link Features. *ACM Trans. Sen. Netw.* x, y, Article A (February 2014), 35 pages.
http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Power consumption is one of the main concerns when using battery-powered WSNs. Compared with the sensing components and the processor unit, the radio is often one of the most power hungry components in a wireless sensor [Pottie and Kaiser 2000]. Thus, reducing the total number of radio transmissions per packet is one of the main goals of network protocols for WSNs. For many sensor networks applications and deployments [Polastre et al. 2004; Xu et al. 2004; Werner et al.], 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

A prior version of this manuscript titled "Foresee (4C): Wireless Link Prediction using Link Features" has appeared in the Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN 2011).

http://dx.doi.org/10.1145/0000000.0000000

and the forwarding nodes such that the path cost of sending a packet to the root is minimal. In this regard, accurate link quality estimation is vital to achieve optimal routing topology. However, due to the dynamic nature of wireless channels, accurate link quality estimation remains a challenging task. Most of the current link estimation metrics are cost based, which means they compute the cost of delivering a packet through a link based on the packet reception ratio (PRR). For example, CTP [Gnawali et al. 2009], the main collection protocol in TinyOS [Levis et al. 2005], uses ETX [De Couto et al. 2003] to create a routing gradient.

However, PRR based metrics have two problems. First, since calculating PRR requires several packets, PRR based metrics are insensitive to link quality variations on a *per packet* level. A common practice [Woo et al. 2003; Fonseca et al. 2007] is the PRR based link estimators only update the link quality estimation with a predefined packet window, 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 [Cerpa et al. 2005; Alizai et al. 2009], 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 [Levis et al. 2004] or adaptive beacon policy [Gnawali et al. 2009] from the routing protocol, the link quality estimation only updates after several packet receptions. In this paper, we tackle this problem by trying to predict the expected link quality on a *per packet* basis.

In addition to PRR, physical layer information is another direct indicator of link quality. The CC2420 radio chip, a widely used off-the-shelf low power radio chip, can provide the Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) for received packets. Moreover, environment noise level can also be detected by CC2420, so the Signal to Noise Ratio (SNR) is also available. These parameters from physical layer (PHY) are directly related to the wireless channel quality when a packet is received, so there is usually a close correlation between the PHY parameters and the link quality. In particular, there are routing protocols that use LQI as link quality metric [MultihopLQI, TinyOS 1.x 2013; Tolle and Culler 2005]. However, due to the short temporal dynamics of wireless channel and differences in hardware calibration [Cerpa and Estrin 2003; Zhou et al. 2004], it is hard to find a well defined correlation between the PHY parameters and PRR over different links and even different networks. As a result, state of art routing protocols only utilize PRR based link estimation metrics such as ETX.

In this paper, we propose to use a machine learning approach to predict the short temporal quality of a link with both physical layer information and PRR. Our 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. 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. Our data-driven approach consists of three steps. The first step involves link quality data collection, and based on the analysis of the data we propose a set of guidelines for the amount of data to

be collected in our experimental scenarios. The second step includes offline prediction model training and selection. We develop three prediction models that utilize different machine learning methods, namely, naive Bayes classifier, logistic regression and artificial neural networks. From our analysis, we find that the logistic regression works well among the three models with small computational cost. Finally, the third step involves the implementation of 4C, a receiver-initiated online link quality prediction module that computes the short temporal link quality. We conducted extensive experiments in the Motelab testbed [Werner-Allen et al. 2005], our local indoor testbed and a temporary outdoor deployment. Our results show that with 4C, CTP improves the average cost of delivering a packet by 20% to 30%. In some cases, the improvement reaches 46%. Empirical evaluation also suggests that 4C is affected by the external interference when data packets in the network are sparse. Experimental results shows that under the presence of external interference, 4C can still reduce the delivery cost by around 30% on average with short inter-packet intervals, but with longer intervals such as 60 seconds, the cost reduction of 4C drops from 17% to 7% on average. Overall, we show that 4C can help routing protocols identify short term high quality links with small overhead, reducing the number of hops as well as improving the overall transmission costs. The main contributions of this work are:

— Analysis and evaluation of the use of both PRR and the physical layer information for link quality estimation. 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.
— Development and evaluation of 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.
— Design and experimental evaluation of 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.

The rest of the paper is organized as the following. Section 2 summarizes the related research, and Section 3 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 describes the implementation details of 4C, and Section 5 presents the experimental results of the CTP with 4C. Section 6 discusses the advantages and the limitations of our approach, and finally in Section 7 we conclude.

2. RELATED WORK

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.

2.1. PRR based Link Estimation

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.* [De Couto et al. 2003] 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 PRR based metric can achieve better routing performance than the shortest hop count. Further comparisons by Draves *et al.* [Draves et al. 2004] 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.* [Woo et al. 2003] 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.* [Fonseca et al. 2007] 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 [Kim and Noble 2001], 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) [Renner et al. 2011], 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, Cerpa *et al.* [Cerpa et al. 2005] 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. Liu *et al.* [Liu et al. 2009] implemented RNP in TinyOS and compared its performance with 4Bit experimentally. Their results show that RNP performs similarly with 4Bit with a tendency of using long links. In this paper, we use 4Bit as a baseline for the performance evaluation (Section 5) due to the much wider range of acceptance of 4Bit.

2.2. Physical Layer Parameters as Link Quality Metric

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 [Texas Instruments 2013] also implement 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 [Zuniga and Krishnamachari 2004]. However, experimental work with early platforms [Zhao and Govindan 2003] 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.* [Lai et al. 2003] 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.* [Son et al. 2006] 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.* [Senel et al. 2007] 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 [MultihopLQI, TinyOS 1.x 2013; Fonseca et al. 2007; Gomez et al. 2010] to adopt LQI as the quality metric of choice. However, Srinivasan and Levis [Srinivasan and Levis 2006] 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.* [Baccour et al. 2010] 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 [Fonseca et al. 2007] is another hybrid link estimator as it utilizes the white bit from the physical layer. Rondinone *et al.* [Rondinone et al. 2008] also suggested a reliable and efficient link quality indicator by combining PRR and normalized RSSI. Boano *et al.* [Boano et al. 2010] 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 [Gomez et al. 2010] proposed by Gomez *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.* [Baccour et al. 2012], along with a taxonomy of existing link quality estimation techniques and their performance analysis.

The 4C link estimator we propose takes similar hybrid method, but it mainly differs from the above link estimators in terms of modeling approach. Specifically, 4C uses numerical methods to build models driven by empirical data collected from the deploy-

ment site to find short-term correlation patterns between physical parameters and link quality, whereas other link estimators either employ heuristics based on experience, or, in the F-LQE and LETX cases, use only linear combination of PHY parameters as well as link layer information.

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 [Gnawali et al. 2009] 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. 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] [Zhao and Govindan 2003; Woo et al. 2003; Cerpa et al.]. Moreover, short time estimation on link quality often shows a bursty pattern on packets reception, which implies that packet loss is correlated [Srinivasan et al. ; Alizai et al. 2009].

To quantitatively study the characteristics of the burst link behavior, Srinivasan *et al.* [Srinivasan et al.] 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.* [Alizai et al. 2009] 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 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.

Our design is fundamentally different from the opportune transmission and STLE as we try to capture bursts of high quality periods using models trained with specific data traffic patterns instead of using heuristics. Our receiver-driven link estimator is similar to the design of STLE, 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

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

control and data packets and the expected probability of success output by our prediction model.

2.4. Link Quality Prediction

Applying data-driven methods on link quality prediction has been less studied. K. Farkas *et al.* [Farkas et al. 2006] 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 [Farkas et al. 2008]. 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.* [Wang et al. 2007] 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.

TALENT [Liu and Cerpa 2012] proposed by *Liu and Cerpa* adopts the similar prediction approach to specifically predict the short term link quality variations with the help of online learning techniques. Although 4C and TALENT bear many similarities, TALENT mainly differs from this work in terms of the prediction target as well as data requirement. TALENT predicts the link quality in near future whereas the prediction target of 4C is the reception of the *next* packet with the inter-packet interval specified in the training stage. In addition, 4C requires prior packet trace collection for the model training, whereas TALENT eliminates the need of data collection by employing online learning techniques.

3. MODELING

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) [Woo et al. 2003]. 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 procedure of model training. The modeling results are presented in the end.

3.1. 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 3.4. In short, the TMotes send 30-byte long packets with 0.1 seconds interval, and we record the packet reception as well as the physical layer parameters of the received packets.

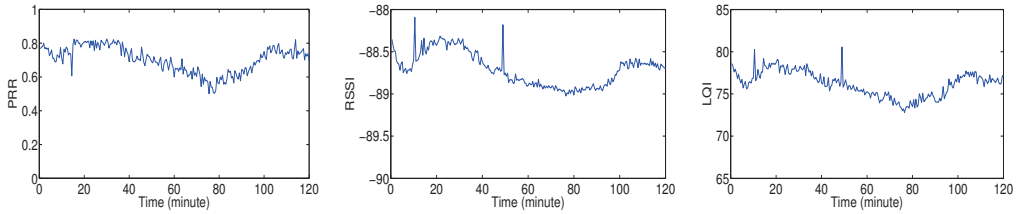
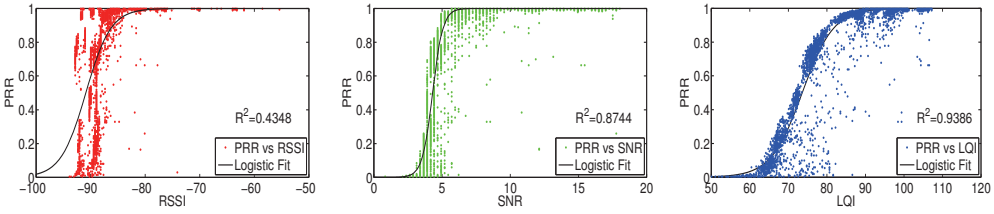


Fig. 1. Packet Reception Ratio (PRR), Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) variation of an intermediate link over a period of 2 hours.



(a) RSSI (b) SNR (c) LQI

Fig. 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.

Fig. 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 period. The similar variation patterns suggest that we can leverage the PHY information to predict link quality. Indeed, there are routing protocols that operate based on LQI [MultihopLQI, TinyOS 1.x 2013; Tolle and Culler 2005]. 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 [Polastre et al. 2005; Srinivasan et al. 2010] 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 [Srinivasan and Levis 2006; Gomez et al. 2010]. The difficulties of using PHY parameters as link quality metrics are summarized in [Baccour et al. 2012], and therefore widely used routing protocols like CTP [Gnawali et al. 2009] 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, several previous studies [Lai et al. 2003; Son et al. 2006; Srinivasan et al. 2010] 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 3.4), and plot PRR with respect to the corresponding PHY parameters (RSSI/SNR/LQI) in Fig. 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 note the goodness of fit [Casella and Berger 2001] (R^2) in each plot.

A simple visual inspection of Fig. 2(b) and 2(c) shows that both LQI and SNR have a significant correlation with PRR. On the other hand, Fig. 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 [Lai et al. 2003] and [Son et al. 2006] 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 [Srinivasan and Levis 2006]. 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.

3.2. Problem Definition

The model we want to create takes 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 [Texas Instruments 2013] 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 $[-100, 0]$. 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 [Woo et al. 2003] 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., $PRR_{new} = (\text{number of received packets}) \div (\text{window size})$. 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.

3.3. 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 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 [Mitchell 1997]. 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 [Bishop 2006]. It is an extensively used method in machine learning, and can be easily implemented 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 [Hagan et al. 1996] 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.

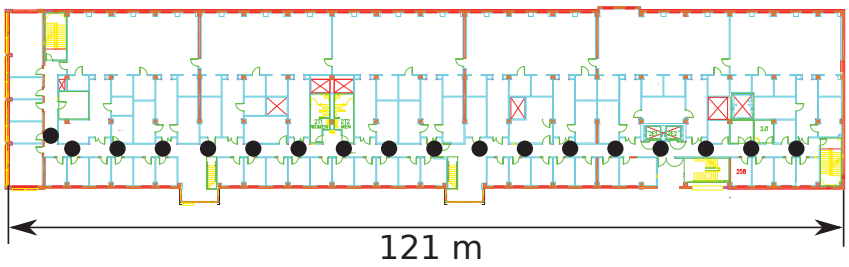


Fig. 3. Local testbed with 54 Tmotes Sky motes placed along an corridor of a typical office building. The motes are divided into 18 groups denoted by the black dots. The distance between each node group ranges from 6 to 7 meters, except for the node group in the far left which sits around a corner at the end the corridor. The nodes are numbered from 0 to 53, starting from the first group on the left to the last group on the right.

Table I. Modeling Parameters

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)

3.4. Data Collection

In order to train the model, we collected packet traces from two testbeds: a local wireless sensor network testbed and the Motelab [Werner-Allen et al. 2005], a sensor network testbed composed of 180 Tmote Sky motes. The local testbed is comprised of 54 Tmotes, installed on the ceiling of a corridor in a typical office building. Fig. 3 shows the placement of these nodes.

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 [Son et al. 2006; Reis et al. 2006; Kashyap et al. 2007], 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 ($PRR > 90\%$).

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 3.7, a dataset of this scale is *more than enough* to train prediction models with satisfying accuracy.

3.5. Modeling Parameters

We varied several parameters in the training process to explore the optimal training parameter collection. Table I 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 of historical packets needed to make a prediction, whereas I decides the periodicity of the prediction. The choice of them can greatly affect the prediction quality and the feasibility of implementing the model on resource constrained sensor nodes. For the input feature, we tried combinations 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. Note that although we only consider link diversity in the reception rate dimension, future work will explore this issue more deeply in a multi-dimensional space, where each dimension quantifies a different characteristic of the link, e.g., standard deviation and skewness.

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.

3.6. 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 3.2, 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 [Fonseca et al. 2007], 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 [Mitchell 1997], whereas LR models learn the regression coefficients from the training data using maximum likelihood estimation [Bishop 2006]. 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 [Hagan et al. 1996] 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 3.8, 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.

3.7. 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 5. 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.

3.7.1. 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 I. That is, the training data size is fixed to $L = 20$ links, and each

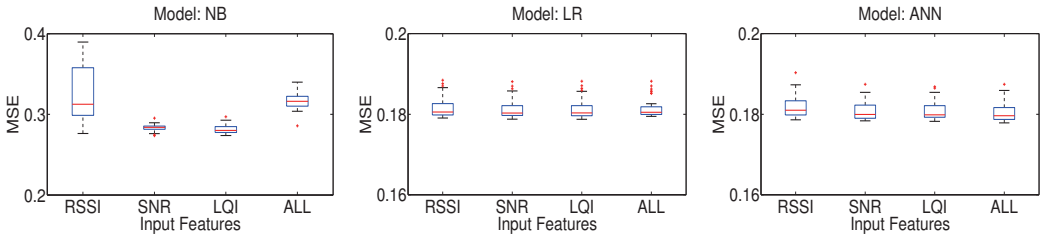


Fig. 4. Prediction errors for different input features. $L = 20$, $P = 36000$. Note the error ranges are different.

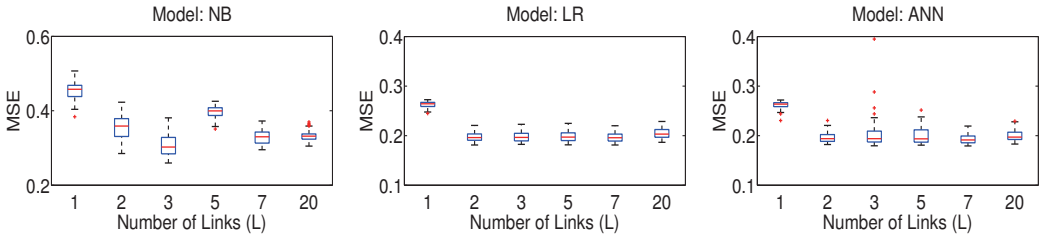


Fig. 5. Prediction errors for $L = \{1, 2, 3, 5, 7, 20\}$, $P = 36000$.

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 3.6. The modeling error (in terms of MSE) of the three models is aggregated and plotted in Fig. 4 respectively.

Fig. 4 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. 5, 6 and 7 are plotted in the similar manner.

From Fig. 4, 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 3.1 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 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.

3.7.2. 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. 5 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 trend of decreas-

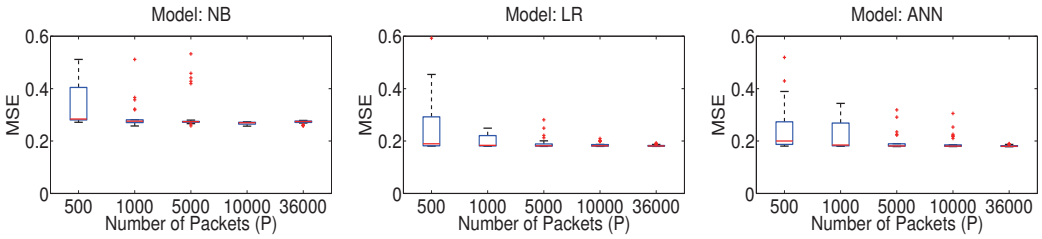


Fig. 6. Prediction errors for $L = 5$, $P = \{500, 1000, 5000, 10000, 36000\}$.

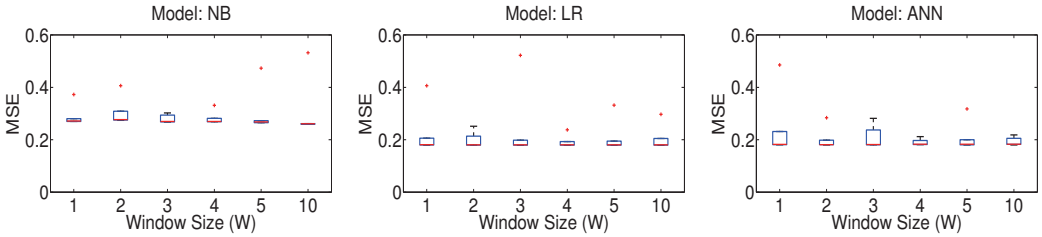


Fig. 7. Prediction errors for $W = \{1, 2, 3, 4, 5, 10\}$. Trained with $L = 5, P = 1000$.

ing 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 3.5.

3.7.3. 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. 6, we observe that for high values of P (36000 for example), the 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.

3.7.4. 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. 7 shows that 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 $W = 1$.

3.7.5. 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. 8 shows the prediction errors as the I (aperiodic) increases from 100 milliseconds to 1 minute,

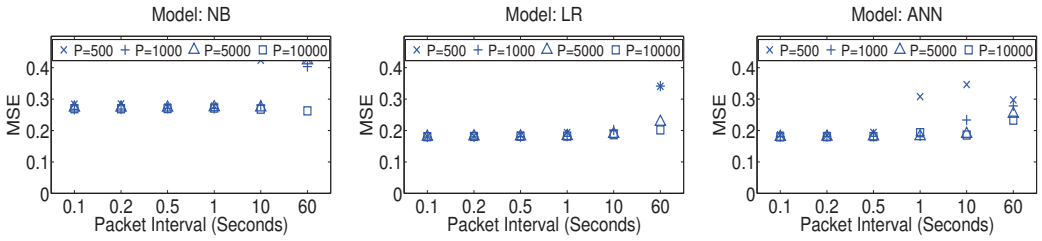


Fig. 8. Prediction errors of $I = \{0.1, 0.2, 0.5, 1, 10, 60\}$ seconds. Trained with $L = 5, W = 1$.

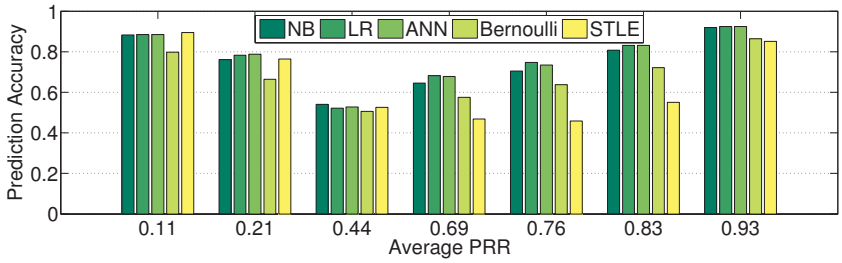
with the MSE of models trained with different P plotted together. Note that Fig. 8 presents the results of the individual models directly due to the small number of unique parameter sets (5 only), whereas Fig. 4, 6, 5 and 7 present the statistics of the modeling results with boxplots due to the large number of parameter sets.

Fig. 8 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.

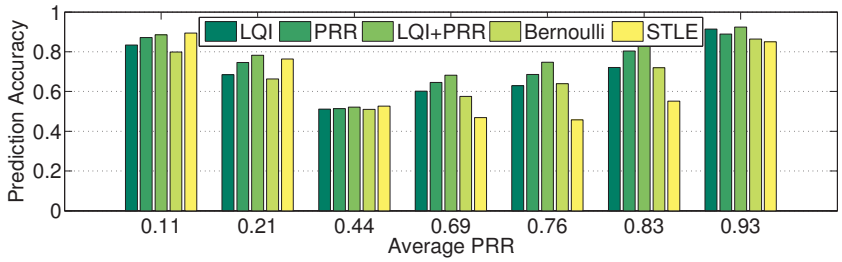
Note that although shorter Is give better results, in practice a model trained with short I may not perform well when there is a mismatch between the I and real packet sending intervals. For example, a model trained with 1 second I assumes that the average data rate is 1 packet/second, and predicts the 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. A simple solution to this problem is to train multiple models with different I values and do best matching based on the data rate conditions seen in the field. We explore the same I mismatch issue in Section 5.3, and leave the full evaluation of the solution for future work.

3.8. 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 [Alizai et al. 2009] and an informed Bernoulli process. STLE is a short temporal link estimator that is based on the heuristic that three consecutive packet receptions signify high link quality in near future and one packet loss signifies 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.



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



(b) LR model with different input features

Fig. 9. Prediction accuracy for links of varying PRR of models trained with $L = 5$, $P = 5000$, $W = 1$ and $I = 10$ seconds.

The prediction accuracy is computed as the ratio of the correctly predicted packets to the total number of packets of the link. Fig. 9(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. 9(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 [Cerpa et al.] 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. 9 also includes the predication accuracy of STLE. In both Fig. 9(a) and 9(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 no 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.

3.9. Summary

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 these statements experimentally in Section 5.

4. 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.

4.1. 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 as a temporary parent as detailed in Section 4.3. 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 denounces 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) [Alizai et al. 2009]. 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

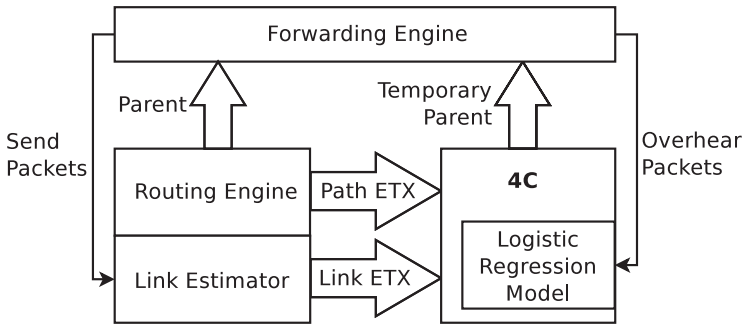


Fig. 10. Overall Design of 4C.

a wide range of traffic patterns by utilizing appropriate models. For example, experimental results in Section 5 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. 4C Design

Fig. 10 presents the overall design of the 4C. In general, 4C 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 4.3.

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.

4.3. Temporary Parent Announcement

A question we need to explore is when should a node announces 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

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

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

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

which transforms to

$$C_P + C_{S\rightarrow P} > C_T + C_{S\rightarrow T} + \frac{2}{n} \times C_{T\rightarrow S}. \quad (1)$$

Formula (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 (1) can be easily calculated on the intercept node with the path cost information from the CTP routing engine and 4Bit: $C_P + C_{S\rightarrow P}$ is the path cost of the sender node, which is available in the CTP packet header. C_T is maintained by CTP, $C_{T\rightarrow S}$ is maintained by the link estimator, and $C_{S\rightarrow T}$ is the link quality prediction that is constantly measured by 4C. Therefore, based on the calculation of

Formula (1), the intercept node can decide when to announce and denounce to be the temporary parent of the sender node.

4.4. Implementation Details

4.4.1. 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 agile 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.* [Amin et al. 1997] 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, optimizations 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.

4.4.2. 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 5.

4.4.3. 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 changes in path cost 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.

4.4.4. 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

Table II. Current draw of TMote Sky under various operation conditions.

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

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.

4.4.5. 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 acknowledgement. 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.

On 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 loss due to the asymmetric link. This is treated as a sender parent change event (addressed in Section 4.4.3), 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.

4.4.6. 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 4.3. 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 (from 28416 to 32540 bytes) with the addition of 4C, whereas the RAM requirement increased by 698 bytes (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 4.3, 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 [Moteiv Corporation 2013]. Table II lists the current consumption measurements as well as the time needed for the prediction and packet transmission operations. As shown in Table II, for a Tmote with 3 V input, 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 acknowledgement require

Fig. 11. The 21-node outdoor testbed.

about 5.25 milliseconds, the energy spent on transmitting a single packet is around $3 \times 20 \times 5.25 = 315\mu J$, which would be able to support about $315 \div 3 \div 2 = 52.5$ 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 orders of magnitude larger than the minimal requirement discussed here.

5. EXPERIMENTAL RESULTS

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.

5.1. 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 [Alizai et al. 2009] and 4Bit [Fonseca et al. 2007] under the similar experimental settings used by the authors of STLE. In the second stage, we further 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 can provide us 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 11. The outdoor testbed consists of 21 TMotes, deployed near a parking lot where the 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.

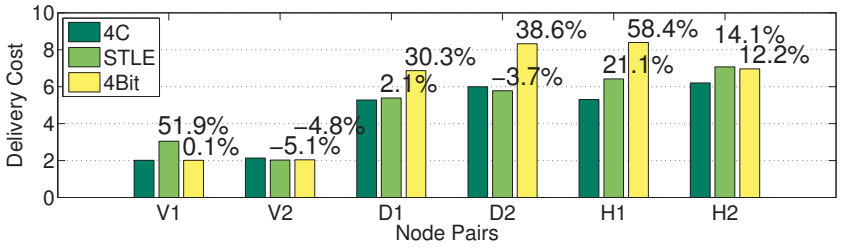


Fig. 12. Cost comparison of 4C, STLE and 4Bit. The percentages on the top indicate the reduction rates of 4C over STLE and/or 4Bit. Labels in the x-axis represent the node pairs listed in Table III, and the percentages on the top indicate the reduction rates of 4C over STLE and/or 4Bit.

Table III. Node pairs in the Motelab experiments.

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

5.2. Single Sender Experiment Results

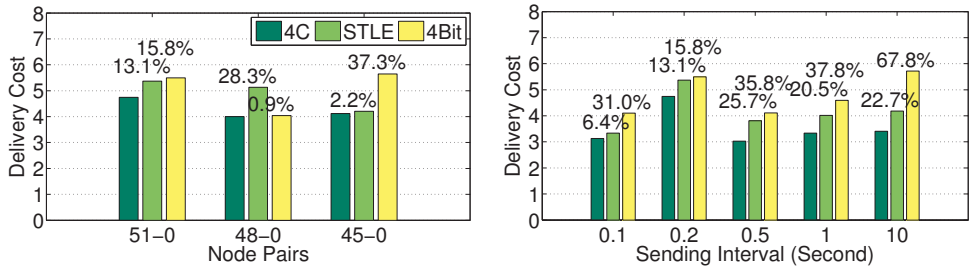
Due to the similarity of STLE and 4C design, it is reasonable to compare the performance of these two link estimators. To provide a plausible comparison, we implemented the STLE based on the design of Alizai *et al.* [Alizai et al. 2009] to the best of our ability. [2].We followed the experimental settings in [Alizai et al. 2009] 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.

5.2.1. 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 [Alizai et al. 2009], 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 III. For each node pair, we let the source node send packets with $SI = 200$ milliseconds for 15 minutes using 4C, STLE and 4Bit respectively. 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.

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



(a) Local – Different Nodes (b) Local – Different Intervals

Fig. 13. Cost comparison of 4C, STLE and 4Bit in the local testbed.

Fig. 12 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.

5.2.2. 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. 13.

Fig. 13(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. 13(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 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.

5.3. Multiple Sender Experiment Results

5.3.1. 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 hours. 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, and leave latency and throughput for future 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 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 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. 3), the inter-node interference was almost unavoidable. We avoid the heavy network congestion cases as neither 4C nor CTP [Gnawali et al. 2009] 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 the 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 the most routing gain) should be considered. This is exactly why 4C adopts a data-driven prediction model: to better evaluate the

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

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

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.

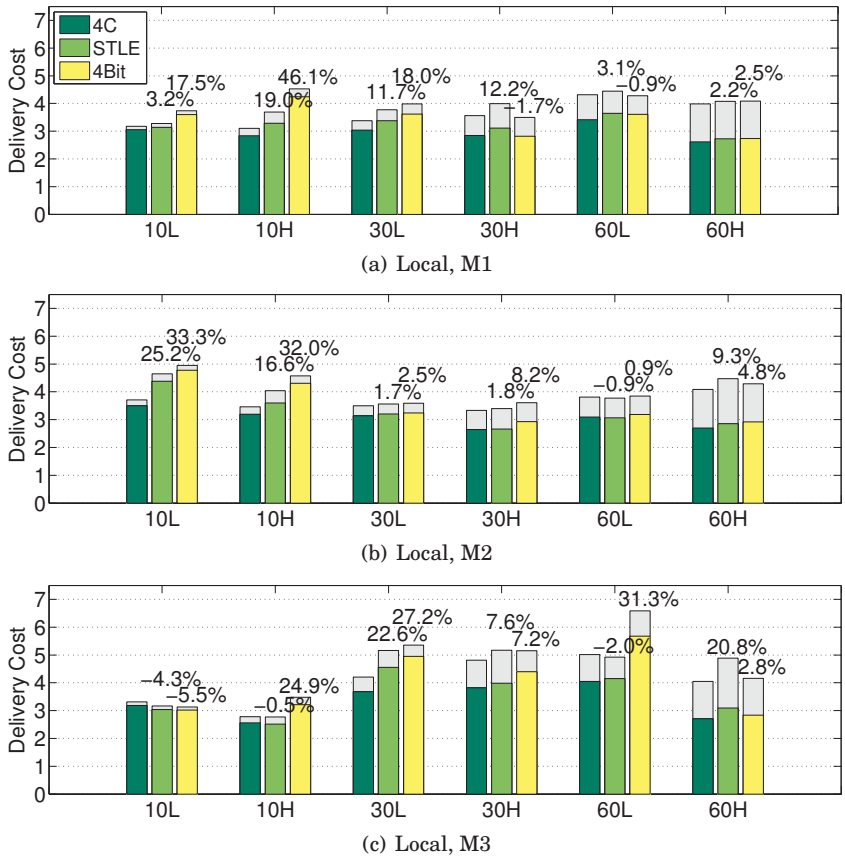
To evaluate the performance of models trained with different parameters, we use three LR models based on the modeling results discussed in Section 3.7. As seen in Fig. 8, LR models trained with $L = 5$, $P = 1000$ are enough to provide good prediction results when $I = 10$ seconds, whereas models trained with $L = 5$, $P = 5000$ perform well when $I = 60$ seconds. So, 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. The modeling parameters are listed in Table IV. All three models were tested in the local testbed experiments whereas model M1 and M2 were used in the outdoor testbed experiments. Next, we discuss the experiment results in terms of delivery cost, path length and beacon overhead.

5.3.2. Delivery Cost. Delivery cost refers to the number of send attempts needed to deliver a packet to the sink. Fig. 14 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.

From Fig. 14(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. 14(b), indicating that the performance of the model is agnostic to the periodicity of I used in the model training. In Fig. 14(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 V. The p-value 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 V, 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



(a) Local, M1

(b) Local, M2

(c) Local, M3

Fig. 14. 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.

Table V. 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.

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

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.

5.3.3. 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. 15 present a path length comparison in the local testbed experiments. Similar to Fig. 14, Fig. 15 presents the path length of CTP with 4C, STLE and 4Bit in different network sizes and densities, with each figure representing a different model

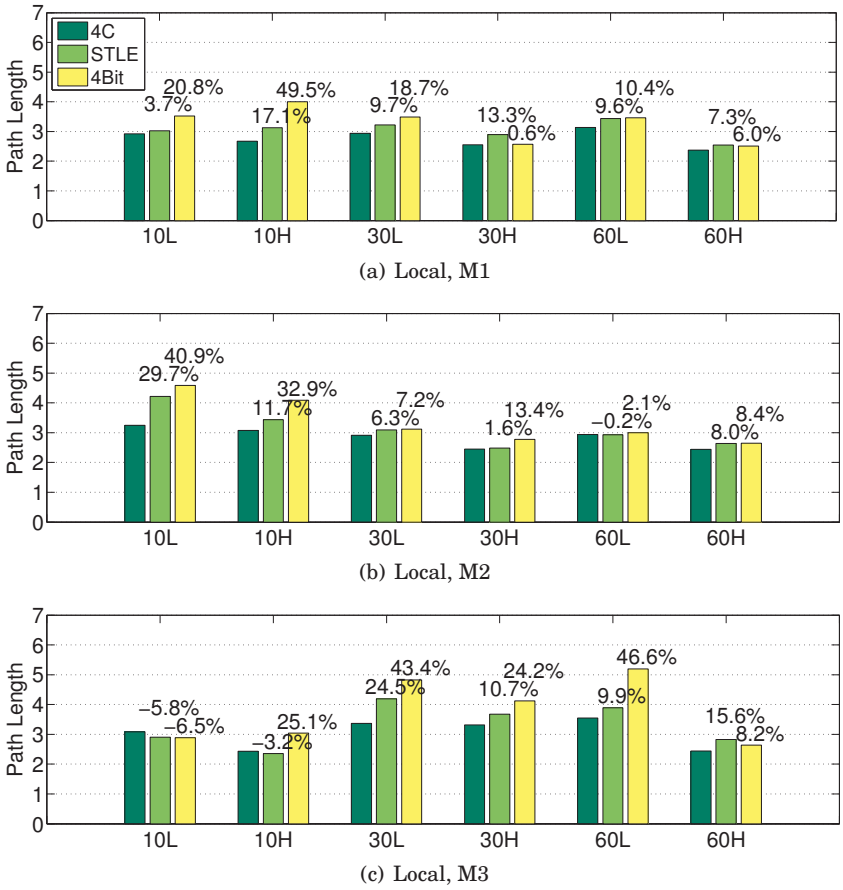
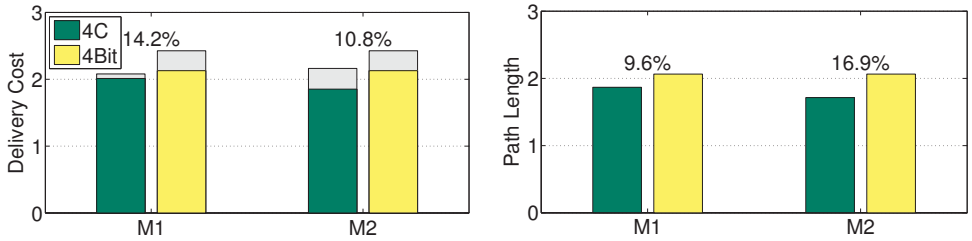


Fig. 15. Path length comparison of multiple sender experiments run on the local testbed.

listed in Table IV. 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. 14). In Fig. 15(a) and Fig. 15(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. 15 is slightly higher than the cost reduction rate shown in Fig. 14. 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. 14.



(a) Cost (b) Path Length

Fig. 16. Cost and path length of 4C running on outdoor testbed. Two models with periodic/aperiodic I are used ($W = 1$, average $I = 10$ seconds).

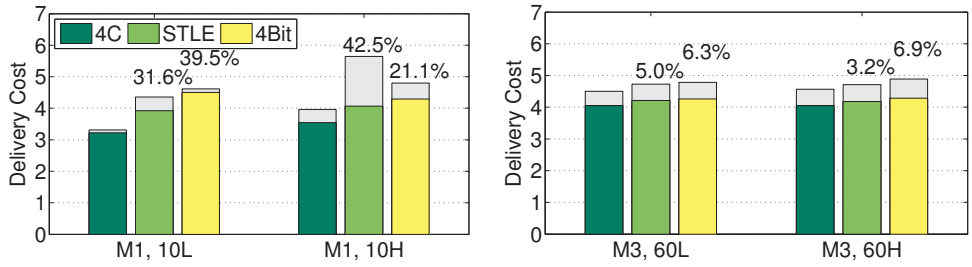
The closeness between the delivery cost and the path length results essentially shows that the cost of sending a packet over each link is close to 1 in all the experiments, therefore the end-to-end delivery cost is simply the number of hops in the forwarding path. However, the fact that CTP with 4C is 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.

5.3.4. 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 routable network structure. Compared with 4Bit, the main overhead of 4C as well as STLE is the temporary parent notification packets. Fig. 14 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.

5.3.5. Outdoor Testbed Results. We also repeated the multiple sender experiments in the outdoor testbed (see Fig. 11) 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 IV. The sending interval SI is set to 10 seconds, and the network density is fixed to 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. 16 shows the delivery cost as well as the path length of CTP using 4C and 4Bit. In Fig. 16(a) we can observe that 4C shows more than 10% improvement over 4Bit in terms of delivery cost, and Fig. 16(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.

5.3.6. 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 is under constant interference from the 802.11 (WiFi) communications in



(a) M1 with 10 seconds SI (b) M3 with 60 seconds SI

Fig. 17. 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.

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. 17 presents the results in terms of end-to-end delivery cost and beacon overhead. Similar to Fig. 14, 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. 17(a) and 17(b) show that the end-to-end delivery cost under external interference is generally larger than without it (see Fig. 14(a) and 14(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. 17(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. 14(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. 17(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. 14(c)), the average cost reduction drops from about 17% to 7%. Combined with the 10 seconds sending interval results presented in Fig. 17(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. Never-

theless, compared with 4Bit, 4C is able to reduce about 7% of the average delivery cost even when the sending interval is 60 seconds.

6. DISCUSSION

6.1. Advantages of a Data-Driven Approach

According to the 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 [Rappaport 2001]. For example, the link layer models proposed in [Zuniga and Krishnamachari 2004] 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 [Zhao and Govindan 2003; Lai et al. 2003]. Moreover, research [Cerpa and Estrin 2003; Zhou et al. 2004; Son et al. 2006] shows that the actual correlation between PRR and PHY parameters may be different due to hardware specific variations. As shown by Son *et al.* [Son et al. 2006], 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 the 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 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.

6.2. 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. The performance improvement when modeling parameters and actual conditions matched, implies that there are some repeatable patterns in packet reception. 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 5.

6.3. 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 5, 4C performs on par with 4Bit when the actual traffic pattern does not match the modeling parameter, indicat-

ing 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 4C such that the prediction model can evolve with the changing network conditions. We leave task of examining the 4C performance with other network dynamics, such as varying radio power and packet size, to future work.

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 paper, 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 networks just as well as in sparse networks with only marginal or unstable links, if not better. Figure 14(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).

7. CONCLUSIONS

In this work 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%. Future research directions involve incorporating time information to the existing input vector to help the model adapt better to varying traffic patterns and the use of online learning algorithms for link quality prediction under dynamic conditions.

8. ACKNOWLEDGEMENTS

Special thanks to Ankur Kamthe for providing data traces for the Motelab testbed and valuable discussions on 4C design, Varick Erickson for helping with the outdoor testbed. This material is based upon work partially supported by the National Science Foundation under grants #CNS-1254192 and #CNS-0923586, and the Center for Infor-

mation Technology Research in the Interest of Society under grants #CITRIS-SPF-81 and #CITRIS-SPF-165.

REFERENCES

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

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

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

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

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer. 380–382 pages.

C. A. Boano, Zúñiga, M. A., T. Voigt, A. Willig, and K. Römer. 2010. 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*. 1–7.

George Casella and Roger L. Berger. 2001. *Statistical Inference*. Duxbury Press. 556 pages.

Alberto Cerpa and Deborah Estrin. 2003. *SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments*. Technical Report 0021. UCLA.

Alberto Cerpa, Jennifer Wong, Louane Kuang, Miodrag Potkonjak, and Deborah Estrin. Statistical Model of Lossy Links in Wireless Sensor Networks. In *IPSN '05*. http://andes.ucmerced.edu/papers/Cerpa05a.pdf

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

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

Richard Draves, Jitendra Padhye, and Brian Zill. 2004. Comparison of Routing Metrics for Static Multi-Hop Wireless Networks. In *SIGCOMM '04*. ACM, 133–144.

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

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

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

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

Carles Gomez, Antoni Boix, and Josep Paradells. 2010. 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, Article 6 (Feb. 2010), 20 pages.

Martin T. Hagan, Howard B. Demuth, and Mark H. Beale. 1996. *Neural Network Design*. CT: Thomson Learning.

Anand Kashyap, Samrat Ganguly, and Samir R. Das. 2007. 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)*. ACM, New York, NY, USA, 242–253.

Minkyong Kim and Brian Noble. 2001. Mobile network estimation. In *Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom '01)*. ACM, New York, NY, USA, 298–309.

Dhananjay Lai, A. Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu, and A. Keshavarzian. 2003. Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In *GLOBECOM '03*.

P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. 2005. TinyOS: An Operating System for Sensor Networks. In *Ambient Intelligence*. Springer Berlin Heidelberg, 115–148.

Philip Levis, Neil Patel, David Culler, and Scott Shenker. 2004. Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st conference on Sym-*

posium on Networked Systems Design and Implementation - Volume 1 (NSDI'04). USENIX Association, Berkeley, CA, USA, 2–2.

Tao Liu and Alberto Cerpa. 2012. TALENT: Temporal Adaptive Link Estimator with No Training. In *SenSys '12*. ACM, 253–266.

Tao Liu, Ankur Kamthe, Lun Jiang, and Alberto Cerpa. 2009. Performance Evaluation of Link Quality Estimation Metrics for Static Multihop Wireless Sensor Networks. In *SECON '09*. IEEE.

Tom M. Mitchell. 1997. *Machine Learning*. McGraw Hill Higher Ed. 180–212 pages.

Moteiv Corporation. 2013. TMote Sky Datasheet. Downloaded from http://www.snm.ethz.ch/Projects/TmoteSky. (2013).

MultihopLQI, TinyOS 1.x. 2013. http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI. (2013).

J. Polastre, R. Szewczyk, and D. Culler. 2005. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005.* 364–369.

Joseph Polastre, Robert Szewczyk, Alan Mainwaring, David Culler, and John Anderson. 2004. Analysis of wireless sensor networks for habitat monitoring. Kluwer Academic Publishers, Norwell, MA, USA, 399–423.

G. J. Pottie and W. J. Kaiser. 2000. Wireless integrated network sensors. *Commun. ACM* 43, 5 (2000), 51–58.

Theodore Rappaport. 2001. *Wireless Communications: Principles and Practice*. Prentice Hall PTR. 104–106 pages.

Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. 2006. Measurement-based models of delivery and interference in static wireless networks. *SIGCOMM Comput. Commun. Rev.* 36, 4 (Aug. 2006), 51–62.

Christian Renner, Sebastian Ernst, Christoph Weyer, and Volker Turau. 2011. Prediction Accuracy of Link-Quality Estimators. In *Proceedings of the 8th European Conference on Wireless Sensor Networks (EWSN'11)*.

Michele Rondinone, Junaid Ansari, Janne Riihijärvi, and Petri Mähönen. 2008. 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)*. ACM, New York, NY, USA, 6–10.

M. Senel, K. Chintalapudi, Dhananjay Lal, A. Keshavarzian, and E. J. Coyle. 2007. A Kalman Filter Based Link Quality Estimation Scheme for Wireless Sensor Networks. In *GLOBECOM '07*. IEEE, 875–880.

Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. 2006. Experimental study of concurrent transmission in wireless sensor networks. In *SenSys '06*. ACM, 237–250.

Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. 2010. An empirical study of low-power wireless. *ACM Trans. Sen. Netw.* 6, 2, Article 16 (March 2010), 49 pages.

Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. The β-factor: measuring wireless link burstiness. In *SenSys '08*. ACM.

Kannan Srinivasan and Philip Levis. 2006. RSSI is Under Appreciated. In *EmNets '06*.

Texas Instruments. 2013. ChipCon CC2420. http://www.ti.com/product/cc2420. (2013).

Gilman Tolle and David Culler. 2005. Design of an application-cooperative management system for wireless sensor networks. In *EWSN '05*.

Yong Wang, Margaret Martonosi, and Li-Shiuan Peh. 2007. Predicting link quality using supervised learning in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review* 11, 3 (2007), 71–83.

Geoff Werner, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06*.

Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. 2005. MoteLab: a wireless sensor network testbed. In *IPSN '05*. IEEE, 68.

Alec Woo, Terence Tong, and David Culler. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03*. ACM.

Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. 2004. A wireless sensor network For structural monitoring. In *SenSys '04*. ACM, 13–24.

Jerry Zhao and Ramesh Govindan. 2003. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03*. ACM, 1–13.

Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. 2004. Impact of radio irregularity on wireless sensor networks. In *MobiSys '04*. ACM, 125–138.

Marco Zuniga and Bhaskar Krishnamachari. 2004. Analyzing the transitional region in low power wireless links. In *SECON '04*. IEEE, 517–526.