Temporal Adaptive Link Quality Prediction with Online Learning

TAO LIU and ALBERTO E. CERPA, University of California, Merced

Link quality estimation is a fundamental component of the low-power wireless network protocols and is essential for routing protocols in Wireless Sensor Networks (WSNs). However, accurate link quality estimation remains a challenging task due to the notoriously dynamic and unpredictable wireless environment. In this article we argue that, in addition to the estimation of current link quality, prediction of the future link quality is more important for the routing protocol to establish low-cost delivery paths. We propose to apply machine learning methods to predict the link quality in the near future to facilitate the utilization of intermediate links with frequent quality changes. Moreover, we show that, by using online learning methods, our adaptive link estimator (TALENT) adapts to network dynamics better than statically trained models without the need of a priori data collection for training the model before deployment. We implemented TALENT in TinyOS with Low-Power Listening (LPL) and conducted extensive experiments in three testbeds. Our experimental results show that the addition of TALENT increases the delivery efficiency 1.95 times on average compared with a 4B, state-of-the-art link quality estimator, as well as improves the end-to-end delivery rate when tested on three different wireless testbeds.

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:
Tao Liu and Alberto E. Cerpa. 2014. Temporal adaptive link quality prediction with online learning. ACM Trans. Sensor Netw. 10, 3, Article 46 (April 2014), 41 pages.
DOI: http://dx.doi.org/10.1145/2594766

1. INTRODUCTION

Link quality estimation is a fundamental component of the network protocols in WSNs. Typically, a link estimator measures the quality of the wireless links continuously and provides the link quality information to the network protocols in the upper layer such that the routing protocol can establish efficient end-to-end delivery paths in an ad hoc manner. For many sensor networks applications and deployments [Polastre et al. 2004b; Xu et al. 2004; Werner et al. 2006], the routing protocol employs a multihop tree topology, in which the nodes in the network connect to the root node(s) through one or more hops, forming a tree-like structure. Due to the high power consumption of the radio components [Pottie and Kaiser 2000], one of the main goals of the network protocols is to reduce the total number of radio transmissions in packet delivery, especially for battery-powered WSNs. Efficient link quality estimation is important for reducing the

A prior version of this manuscript titled "TALENT: Temporal Adaptive Link Estimator with No Training" appears in *Proceedings of the 10^{th} ACM Conference on Embedded Networked Sensor Systems (SenSys12)*.
This material is based on work partially supported by the National Science Foundation under grant nos. CNS-1254192 and CNS-0923586, and the Center for Information Technology Research in the Internet Society under grant nos. CITRIS-SPF-81 and CITRIS-SPF-165.
Authors' addresses: T. Liu (corresponding author) and A. E. Cerpa, University of California, Merced, CA; email: tliu@andes.ucmerce.edu.

DOI: http://dx.doi.org/10.1145/2594766

energy consumption of communications, as accurate link quality information is vital to achieve optimal routing topology.

The physical-layer (PHY) information such as Received Signal Strength Indicator (RSSI) and Signal-to-Noise Ratio (SNR) is directly related to the quality of the wireless channel, and can be used as direct indicators of link quality. Also, the CC2420 [Texas Instruments 2013], a widely used off-the-shelf low-power radio chip, provides Link Quality Indicator (LQI) as another link quality metric from the physical layer. These PHY parameters reflect the wireless channel quality when a packet is received and can be used in link quality estimation. For example, there are routing protocols that use LQI as link quality metric [MultihopLQI, TinyOS 1.x 2013]. However, due to the short temporal dynamics of wireless channels 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 Packet Reception Rate (PRR) over different links and even different networks. As a result, WSN routing protocols often utilize PRR-based link estimation metrics such as ETX [De Couto et al. 2003] instead of using PHY parameters directly. For example, CTP [Gnawali et al. 2009], the default collection protocol in TinyOS [TinyOS 2.x 2013], uses ETX to measure link quality and create routing topology.

An intrinsic problem of PRR-based metrics is that they do not perform well when monitoring intermediate links that often show short temporal quality variations. Because the calculation of PRR requires several packets, PRR-based metrics such as ETX tend to capture the long-term link quality instead of short-term quality variations. As a result, routing protocols such as CTP tend to ignore the intermediate bursty links [Alizai et al. 2009] that have low average PRR in the long term (PRR between 10% and 90%), but show continuous high quality in short periods. Prior work [Cerpa et al. 2005b] has shown that intermediate bursty links usually cover longer distances than high-quality links, and routing protocols could take advantage of the high-quality periods of bursty links and use them when forwarding a packet. This strategy can reduce the number of hops in the path and ultimately reduce the number of transmissions for delivering a data packet. Nevertheless, it is relatively hard for ETX to identify *when* an intermediate link will be in a high-quality period due to the convergence time of ETX itself. Another problem is that 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. Thus, accurate quality estimation of intermediate links remains a difficult task due the convergence time of ETX and the dynamic nature of wireless channels.

We argue that, in order to leverage the intermediate-quality links, the network protocol needs not only a link quality estimator that measures the past link quality, but also a link quality predictor that predicts the link quality in the near future. In this article we propose to *predict* the *expected* quality of the link based on historical information of PRR as well as PHY parameters of recently received packets. The underlying intuition is that, by using a combination of PRR from the link layer and PHY parameters from physical layer as input, the proposed link quality predictor could supplement the PRR, accurate for long-term link estimation, with PHY parameters to improve the short temporal quality estimation for the intermediate links, which are highly unstable and exhibit the most variations. Due to the dynamic nature of the wireless channel, such prediction will be valid for only a short period before the link quality changes. Nevertheless, with the knowledge of expected link quality in the immediate future, the routing protocol may be able to select an efficient data forwarding path promptly during a burst transmission of data packets, and ultimately increase delivery efficiency and reduce communication costs.

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

Another important feature of the estimator is *plug-and-play*. Ideally, a link estimator should work on any network without predeployment efforts to tune the prediction model. Prior studies have shown that model-based predictors such as 4C [Liu and Cerpa 2011] significantly outperform link estimators such as STLE [Alizai et al. 2009] and 4B [Fonseca et al. 2007], but the main disadvantage of 4C is the need to collect link data at the target deployment site for training the link prediction model. Although the required training dataset is small, collecting it still requires additional effort and might not be feasible for all deployments. Furthermore, as wireless conditions change from the time we collect the training data, the same set of model parameters may cause performance degradation. Therefore, it is important to have an estimator that needs no offline training data or prior knowledge from the target deployment.

Based on the preceding requirements, we propose Temporal Adaptive Link Estimator with No offline Training (TALENT), an adaptive prediction-based link estimator that focuses on estimating temporal link quality variations. TALENT utilizes online learning algorithms to adapt to different network conditions without any user intervention and no a priori training and is designed to be a plug-and-play estimator for any environment and level of dynamics.

The contributions of our work are fourfold. First, we show that, by using online learning techniques, our prediction model can adapt to a wide range of network dynamics without prior training data and with fast convergence time. To our knowledge, this is the first attempt to introduce online learning techniques to adapt network link estimation parameters under environmental and network dynamics. Second, we designed and implemented TALENT in TinyOS and integrated it into CTP for a reference implementation. Third, we integrated TALENT with LPL [Polastre et al. 2004a], a low-power listening protocol for efficient communication and actual energy savings when using duty-cycled radios. Finally, we show that, by utilizing TALENT, the routing protocol could use intermediate links more efficiently and achieve lower communication costs when sending data packets in bursts. From our extensive experimental evaluation, we show significant improvement of packet delivery efficiency, with an average of 95.3% improvement over 4B [Fonseca et al. 2007] in many different scenarios, as well as an improvement in end-to-end delivery rate. Furthermore, we applied the prediction approach of TALENT to empirical packet traces from 802.11 networks and confirmed that TALENT outperforms ETX-based link estimators significantly even in 802.11 networks with much higher data rate. These results suggest the potential application of TALENT in a much wider range of wireless networks.

2. RELATED WORK

Link quality estimation is a critical component for wireless communications in WSNs. De Couto et al. [2003] proposed ETX, a widely used wireless link quality metric that uses the Packet Reception Rate (PRR) to estimate the expected transmission cost over a wireless link. Their study shows that this PRR-based metric can achieve better routing performance than the shortest hop count. Further comparisons by 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. [2003] argued that PRR-based metrics such as ETX [De Couto et al. 2003] are more suitable in cost-sensitive WSNs and showed that ETX with the windowed mean estimator with exponentially weighted moving average (WMEWMA) works better than other established estimation techniques. Following their design, Fonseca et al. [2007] proposed 4B, a hybrid link estimator that combines the information from the physical, data-link, and network layers into four bits, namely the *ack* bit from the link layer that indicates whether an acknowledgment is received for a sent packet, the *pin* bit and the *compare* bit from the network layer that interact with the underlying routing protocols to keep important neighbor nodes monitored, and the *white* bit from the physical layer that denotes the high-quality wireless channel by checking the rate of the decoding error in the received packets. Although the white bit uses a physical-layer parameter, 4B only considers it as a quick indication of the overall wireless channel quality. In essence, 4B inherits the WMEWMA design proposed by Woo et al. and uses ETX as its link quality metric.

Information from the physical layer (PHY) is also a candidate of the link quality metric, as the physical layer can provide immediate information on the wireless channel as well as the quality of received packets. Many studies have attempted to find correlations between the PHY parameters and PRR. In theory, PRR can be computed based on SNR and other radio parameters such as the modulation scheme, encoding scheme, frame, and preamble lengths [Zuniga and Krishnamachari 2004]. However, Zhao and Govindan [2003] showed that estimation for low- and intermediate-quality links using RSSI values only is difficult due to the short wireless channel coherence time. Lai et al. [2003] confirmed that the expected packet success rate (PSR) can be approximated by SNR with a sigmoid function, but Son et al. [2006] also showed that, due to the difference in radio and transmission power, there is a significant variation of about 6dB in the threshold of the sigmoid function. Senel et al. [2007] proposed an SNR-based estimator that uses a Kalman filter to process SNR and estimates the PSR with a precalibrated SNR-PSR function. TALENT mainly differs from the previous metrics in terms of modeling approach, as detailed in the following sections.

Another research direction is to create hybrid link estimators that employ the physical-layer parameters in addition to the PRR-based metrics. F-LQE, proposed by Baccour et al. [2010], is a fuzzy logic link quality estimator that utilizes linear membership functions to compute the quality estimation based on four characteristics: packet delivery (PRR), link asymmetry, stability, and SNR. The aforementioned 4Bit [Fonseca et al. 2007] can be also considered as a hybrid link estimator as it utilizes the white bit from the physical layer. The TALENT link estimator also uses a similar hybrid method, but it mainly differs from the aforesaid link estimators in terms of modeling approach. Specifically, TALENT uses numerical methods to build models driven by empirical data 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 case, use only a linear combination of PHY parameters as well as link-layer information.

It is well established that most of the link quality variations are observed in the intermediate-quality links as pointed out by numerous studies [Zhao and Govindan 2003; Woo et al. 2003; Cerpa et al. 2005a]. Srinivasan et al. [2008] showed that packet losses are correlated and the intermediate links often show bursty patterns on packet reception. Moreover, they proposed a β factor to quantitatively measure the burstiness of a link. Alizai et al. [2009] proposed a bursty routing extension (BRE) to CTP that utilizes a short-term link estimator (STLE) to detect short-term high-quality links. STLE takes a receiver-initiated design, which allows the overhearing nodes to notify the senders about the availability of the better path. The overhearing nodes detect the high-quality links based on the heuristic that any link becomes temporarily reliable after three consecutive packets are received over that link. Our design is fundamentally different from STLE as we try to predict high-quality periods in the near future using models trained with online learning techniques instead of using heuristics.

Using data-driven models to predict the wireless link quality has been relatively less studied in WSNs. Farkas et al. [2008] proposed XCoPred, a pattern matching approach based on SNR, to predict link quality variations, but their work was focused on an 802.11 ad hoc network. Liu and Cerpa [2011] presented 4C, a data-driven link quality estimator that tries to predict the success probability of the next packet using a logistic regression classifier. The classifier is trained with data collected a priori in the intended environment. TALENT shares many of its design decisions, but uses online learning techniques to avoid the need for an offline training dataset. Furthermore, the goal of TALENT is to estimate the quality of the link in the near future instead of the reception probability of a single packet.

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 preceding 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 offline, whereas XCoPred considers SNR only and uses cross-correlation without prior training. 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 explores many more parameters with different traffic patterns.

3. MODELING

We propose to build models to predict the future link quality with information from both the physical layer and the link layer. The intuition of using information from both layers is simple: by using a combination of PRR from the link layer and PHY parameters from physical layer as input, the proposed model could supplement the PRR, accurate for long-term link estimation, with PHY parameters to improve the short temporal quality estimation for the intermediate links, which are highly unstable and exhibit the most variations. More importantly, we propose to utilize online learning algorithms such that the models can adapt their parameters to the network dynamics without the overhead of data collection and training.

3.1. Problem Definition

To predict the high link quality in the near future based on the recent packet reception, we propose creating a model to solve the following problem: given W packets as input, determine the probability that the future reception rate on the link will be greater than a predefined threshold θ during a short period of time t. Figure 1 illustrates our approach. Formally, the input of the model is the historical information available from W packets:

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

The information available for a packet PKT_i is comprised of a subset of the PHY-layer parameters associated with the packet as well as the PRR when the packet is received:

$$PKT_i = [PRR_i, PHY_i],\ PHY_i \subset (RSSI_i, SNR_i, LQI_i). \tag{2}$$

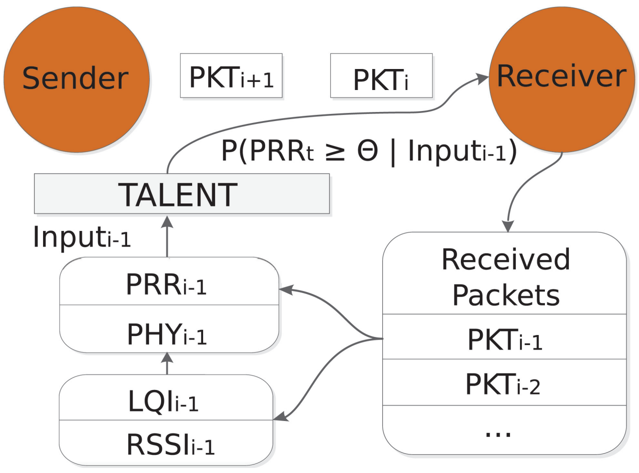


Fig. 1. TALENT attempts to infer the probability that future PRR exceeds a certain threshold θ.

All the values in a packet vector are discrete. PRR_i can be calculated from the WMEWMA output and has a range between [0, 1]. The physical parameters (RSSI, SNR, and LQI) are scaled down to the unit range [0, 1] also. With this notation, we can represent a lost packet as $PKT_i = [PRR_i, 0]$, where the $PHY_i = 0$ since there is no physical parameter available for the lost packet.

The output of the model is the probability of the temporal link quality being better than the threshold θ during the time t in the future:

$$P(PRR_t \geq \theta | Input_i). \tag{3}$$

The calculation of the instantaneous link quality PRR_t is subject to the time t and the number of packets sent during that time. For our analysis, we set θ to 0.9, the time t to 1 second, and the number of packets sent during t is fixed to 10 with an inter-packet interval (I) of 0.1 seconds. We chose a small t and short inter-packet interval because the temporal variation of the wireless channel is correlated at intervals smaller than 1 second [Aguayo et al. 2004; Srinivasan et al. 2008; Rusak and Levis 2009], and our model tries to take advantage of this behavior. This phenomena also is confirmed in Section 3.5 which shows that the performance of the proposed model degrades rapidly as the inter-packet interval increases from 0.1 to 1 second. We use the same parameters in our experiments in order to best approximate real network conditions.

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

3.2. Modeling Method

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

We propose to use a stochastic gradient descent (SGD) online learning algorithm [Mitchell 1997] to train a logistic regression classifier (LR) such that the model can adapt to changing network conditions. We choose to use LR models because prior work [Liu and Cerpa 2011] has shown that the link quality can be accurately predicted by LR models. In Liu and Cerpa [2011] we compared the link quality prediction performance of three classification algorithms, namely Naive Bayesian classifier (NB), logistic regression (LR), and Artificial Neural Network (ANN). Through extensive evaluation on empirical packet traces, we found that ANN and LR perform similarly, whereas NB's performance is inferior to these two. Therefore, we selected LR over ANN due to less computation complexity and the similar performance. For the detailed performance analysis, please refer to Liu and Cerpa [2011]. For the online learning algorithm, we considered many online learning frameworks, such as weight majority, winnow, and SGD, among others [Bertsekas 2012], and we settled for SGD mainly due to its performance and simplicity to implement it under stringent computational and energy constraints. The idea of SGD is simple: based on the error gradient of each prediction result and its corresponding target, SGD updates the parameters of the LR model, and therefore learns the link characteristics online. Furthermore, we employ learning rate adaption algorithms such that the model can dynamically adjust the learning speed based on the error gradient. With the learning rate adaption algorithm, the model is able to accelerate learning when the error is large so that it can quickly adapt to the underlying link quality variations, and slow down to a stable state when the error is small.

Formally speaking, assume $X =< X_1 \ldots X_n >$ represents the input vector *Input* discussed in the previous section, and Y is the binary variable denoting the high temporal link quality $PRR_t > \theta$, the logistic regression classifier can be expressed as

$$P(Y = 1 \mid X) = \frac{1}{1 + \exp(-f(X))} \tag{4}$$

and

$$P(Y = 0 \mid X) = \frac{\exp(-f(X))}{1 + \exp(-f(X))}, \tag{5}$$

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

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

$$\begin{aligned} l(\beta) \; = \; & \sum_{l=1}^{N} Y^l \log P(Y = 1 | X^l, \beta) \\ & + (1 - Y^l) \log(P(Y^l = 0 \mid X^l, \beta)). \end{aligned} \tag{6}$$

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

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

$$\frac{\partial}{\partial \beta_i} l(\beta) = \sum_{l=1}^{N} \left(Y^l - \hat{P}\left(Y^l = 1 | X_i^l, \beta\right)\right) X_i^l, \tag{7}$$

where $\hat{P}(Y^l = 1 | X_i^l, \beta)$ is the logistic regression prediction using Eqs. (4) and (5) and the weights β.

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

$$\beta_i \leftarrow \beta_i + \lambda \sum_{l=1}^{N} \left(Y^l - \hat{P}\left(Y^l = 1 | X_i^l, \beta\right)\right) X_i^l, \tag{8}$$

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

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

$$\beta_i \leftarrow \beta_i + \lambda \Delta \beta_i^l, \tag{9}$$

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

$$\Delta \beta_i^l = \left(Y^l - \hat{P}\left(Y^l = 1 | X_i^l, \beta\right)\right) X_i^l. \tag{10}$$

Using an online learning algorithm to model the link quality variations has several advantages. From a networking aspect, each packet is a new sample, thus the training dataset continues to grow indefinitely. In terms of computation speed, the stochastic learning algorithms will likely outperform the batch learning algorithms that operate over a training set [Bottou and LeCun 2004]. Stochastic learning is also useful when the function being modeled is changing over time, quite a common scenario in networking where the data traffic patterns and the wireless channel quality variations are both nondeterministic. Also, stochastic learning often results in better solutions because the noise in the updates can cause the weights jumping into multiple, possibly deeper local minimum, whereas batch training will only converge to one minimum [Le Cun et al. 1998].

3.3. Learning Rate Adaptation

An important parameter of SGD is the learning rate λ. The rate affects the learning speed and how fast the gradient descent converges. Different from batch gradient descent, which has a linear convergence speed [Dennis and Schnabel 1983], online gradient descent proceeds rather slowly during the final convergence phase [Bottou and Murata 2002]. The noisy gradient estimate causes the parameter vector to fluctuate around the optimum in a bowl whose size depends on the actual learning rates. Ideally, we want a learning algorithm that converges quickly when the network is stable, and updates its parameters promptly once the prediction error increases due to network dynamics.

We tried two adaptive learning rate algorithms, ALAP and s-ALAP [Almeida et al. 1998]. ALAP is a normalized step-size adaptation method with the main idea of changing the global learning rate λ to time-varying local learning rates $<\lambda_1 \ldots \lambda_n>$ that adapt by gradient descent while simultaneously adapting the weights. At time t, we would like to change the learning rate (before changing the weight) such that the error at the next time step is reduced. For the lth sample, ALAP performs the learning rate update with

$$\lambda_i \leftarrow \max\left(0.5,\, 1 + q\Delta\beta_i^l \Delta\beta_i^{l-1}\right)\lambda_i, \tag{11}$$

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

$$\beta_i \leftarrow \beta_i + \lambda_i^l \Delta\beta_i^l. \tag{12}$$

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

$$\lambda_i \leftarrow \max\left(0.5,\, 1 + q\Delta\beta_i^l \frac{\Delta\beta_i^{l-1}}{\overline{\left(\Delta\beta_i^l\right)^2}}\right)\lambda_i, \tag{13}$$

where $\overline{(\Delta\beta_i^l)^2}$ is an exponential moving average of the square of $\Delta\beta_i^l$. The weight update rule is same as Eq. (12).

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

Another common extension of the SGD algorithm is the use of a momentum term [Mitchell 1997]. With the momentum term, the weight update of lth sample becomes

$$\beta_i \leftarrow \beta_i + \lambda^l \Delta\beta_i^l + m\Delta\beta_i^{l-1}, \tag{14}$$

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

3.4. Online Learning Algorithm Evaluation

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

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

We then apply SGD with momentum, ALAP, and s-ALAP to these empirical packet traces to evaluate their performance in terms of prediction accuracy, that is, the ratio of the correctly predicted high-quality periods to the total number of predictions made. In order to study the performance gain of these online learning models with respect to other existing quality estimation schemes, we also apply three state-of-art link estimation schemes to the same packet traces, namely the WMEWMA estimator described in Woo et al. [2003], the Short-Term Link Estimator (STLE) proposed in Alizai et al. [2009], as well as a batched trained logistics regression model (Batch).

The WMEWMA estimator is used in 4B, the default link estimator in TinyOS. Essentially, the output of the WMEWMA estimator is the ETX value smoothed by an EWMA filer, which can be expressed as

$$ETX_i = \alpha * ETX_{i-1} + (1-\alpha) * ETX_{new}, \tag{15}$$

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

$$ETX_{new} = \frac{\text{window size}}{\text{number of received packets}}. \tag{16}$$

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

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

STLE is a short-term link estimator designed to predict whether a link is in a high-quality state. It is based on the heuristic that if there are three consecutive successful packet receptions, the link is in a high-quality period, and if there is one packet loss, the link is no longer in high quality. Here, we implement the STLE scheme and use the output as the prediction of future link quality in the next 1 second.

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

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

$$PRR_i = \frac{1}{ETX_i}. \tag{17}$$

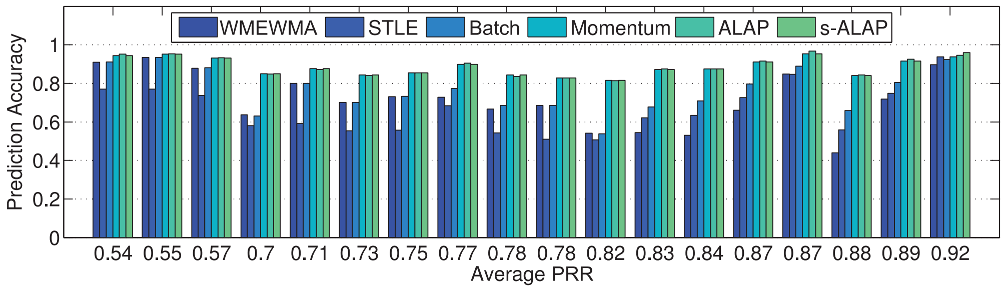


Fig. 2. The prediction accuracy of LR models using three online learning algorithms, as well as a batch-trained LR model, STLE, and the WMEWMA estimator.

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

There are three parameters that affect the input of the model: the window size W, the inter-packet interval I, and the physical parameter used in the input. As pointed out by previous work [Liu and Cerpa 2011], a window size of 1 ($W = 1$) is enough for the LR-based models, and including which physical parameters does not significantly affect the prediction accuracy. In this section, we only present the prediction performance of the prediction models using PRR and LQI with $W = 1$, and leave the evaluation of the impact of using different parameter sets to Section 3.5.

3.4.2. Overall Prediction Accuracy. Figure 2 shows the prediction accuracy of the 6 link estimators with respect to the link PRR. The x-axis denotes the average PRR of each link, whereas the bars represent the prediction accuracy of the link estimators respectively. From this figure, it is clear that for the links with average PRR in 0.8 range, the WMEWMA estimator performs the worst among other estimators, which verifies that the WMEWMA estimator could not accurately estimate the link quality variations in the short term. The prediction accuracy of STLE is generally better than WMEWMA for the links with PRR higher than 0.8, but it performs worse than WMEWMA for the links with PRR ranging between 0.5 to 0.8.

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

3.4.3. Detailed Results – WMEWMA and Batch-Trained LR Model. To further analyze the potential gain of using s-ALAP, we plot the detailed prediction results of s-ALAP, batch-trained LR model, and the WMEWMA estimator in Figure 3. In this figure, each prediction is classified into one of four categories: True Positives (TP), which means

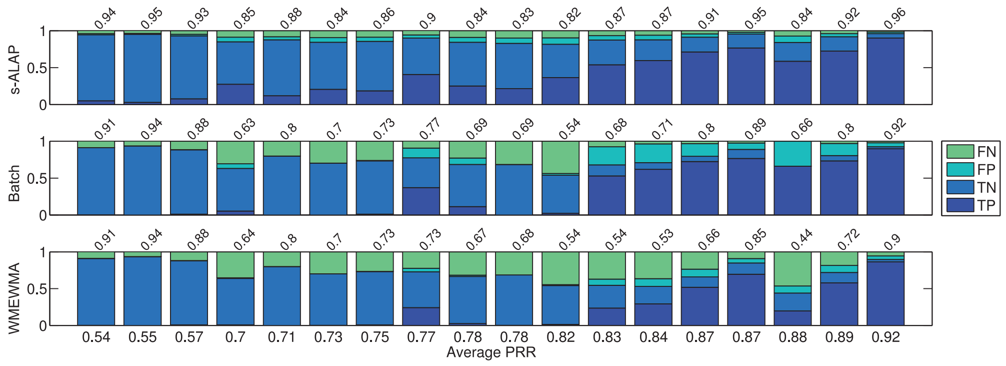


Fig. 3. Prediction accuracy of three prediction methods applied to 19 intermediate links. The numbers on top of each graph show the prediction accuracy, and the labels on the bottom indicate the link PRR. The detailed results (TP, TN, FP, FN) are marked with different colors for each link.

both the model output and the actual PRR are high; True Negatives (TN), meaning the output and the target PRR are both low; False Positives (FP), indicating the output is high whereas the actual target PRR is low; and False Negatives (FN), which means the output is low whereas the actual target is high. Then, the prediction accuracy is calculated as the ratio of TPs and TNs over all the four classes. The four categories are marked with different colors in Figure 3, and the prediction accuracy of the link is labeled on the top of each graph respectively.

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

In order to further justify the advantage of using s-ALAP in intermediate links, we plot the distribution of short-term PRR computed with 1-second window of two links in Figure 4. Although these two links have different long-term quality (average PRR equals 0.7 and 0.84), both of them show two distinctive peaks centered in different PRRs. This bimodal distribution of short-term link quality typically presents in bursty links [Srinivasan et al. 2008], where short bursts of high-quality periods are interleaved with periods of low quality. By leveraging the online learning techniques, TALENT can adapt quickly to link quality changes and identify future high-quality periods better than the other schemes.

Note that WMEWMA is designed to estimate the average link quality, whereas the model-based estimators are designed and trained to predict the probability of the temporal link quality being high during time t in the future (in our case, $PRR_t > 0.9$). When the average PRR of the link is close to but below the 0.9 threshold, WMEWMA tends to make many FN mistakes because it converges to the average link PRR that is below the threshold. Hence, WMEWMA fails to predict short intervals of time when the

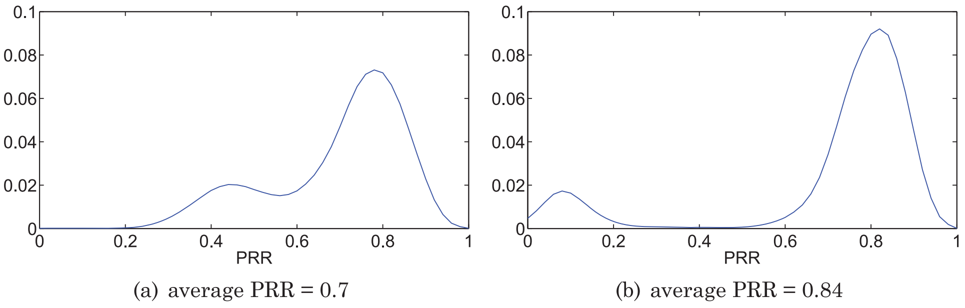


Fig. 4. PRR distribution of two intermediate links.

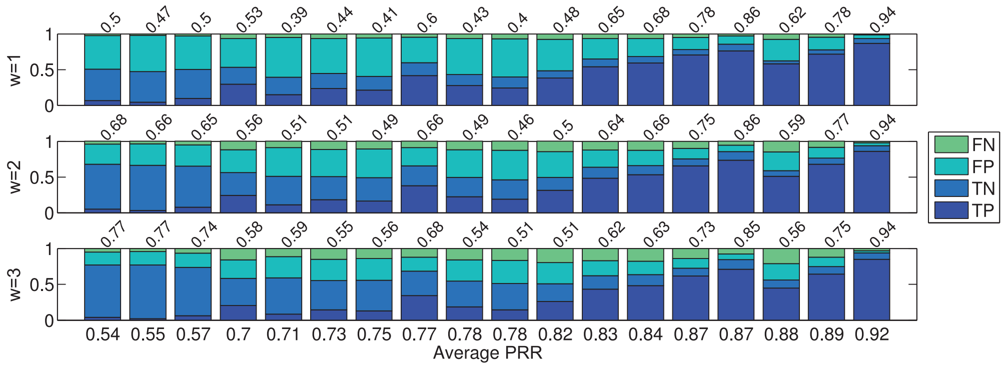
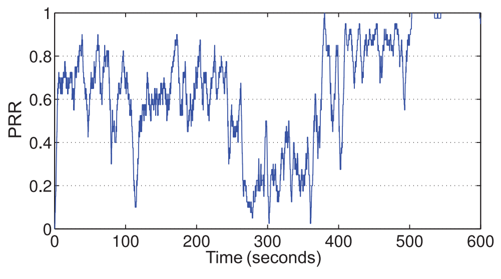


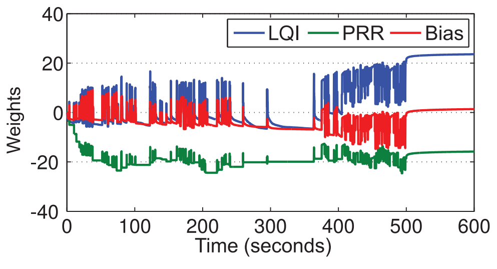
Fig. 5. Prediction accuracy of STLE with different windows applied to the intermediate links. The labels on the bottom indicate the link PRR, and the detailed results are marked with different color. The labels on the left of each plot represent the window size, that is, how many consecutive packets are needed before STLE considers a link high quality.

short-term PRR is high, a very common event for a link with average PRR in the high 0.8 range. When the average link PRR is very different from the threshold (e.g., in the cases of links in the 0.5 range), or above the threshold (e.g., the 0.92 link) WMEWMA tends to perform much better. This is because the likelihood of a link in the mid-0.5 range to have short bursts of very high quality above the 0.9 threshold is significantly smaller, and therefore WMEWMA tends to correctly predict a failure. This behavior can be seen clearly in Figure 3: when the link PRR is close to 0.5, WMEWMA achieves high prediction accuracy only because of the high number of TNs. When the average PRR of the link is above the threshold (like the 0.92 link), then all estimators work fine. Thus, it is clear that WMEWMA is not suitable for estimating temporal high-quality periods for intermediate links.

3.4.4. Detailed Results – STLE. STLE is a short-term link estimator that considers a link high quality (future PRR in one second higher than 0.9) if there are three consecutive packet receptions. In other words, the underlying assumption behind STLE is that the short-term link quality can be predicted by counting the number of packets received without loss. If the number of consecutive receptions exceeds a certain window (three in this case), the link is regarded as in high-quality period. However, as shown in Figure 3, STLE performs worse than the online models, and sometimes even worse than WMEWMA. To understand the performance results and to evaluate the effectiveness of the STLE assumption, we apply STLE with varying window sizes and plot the detailed results in Figure 5.



(a) PRR Variations



(b) Weights

Fig. 6. Short-term PRR variations of a link with 0.7 average PRR and the corresponding weights of the LR model over time.

Similar to the previous figure, Figure 5 categorizes the results into four categories: FN, FP, TN, and TP. The labels in the x-axis represent the link PRR, whereas the labels on the left of each plot denote the window size of STLE. For example, when $w = 1$, STLE will mark a link of high quality as long as the previous packet is successfully received, whereas $w = 3$ (default value in Alizai et al. [2009]) means that STLE considers the link quality high when three packets are received consecutively. The labels on top of each plot represent the prediction accuracy.

From Figure 5, we see that, regardless of the window size, a major portion of the error comes from FP, which means the STLE output often falsely indicates short-term high link quality periods. This result suggests that the assumption of STLE is too optimistic: the instantaneous PRR in the next 1 second is often smaller than 0.9 even if the previous three packets were all received. In addition, comparing the results of $w = 1$ (top plot) and $w = 3$ (bottom plot), it can be observed that for links with PRR ranging from 0.5 to 0.8, STLE with a large window size $w = 3$ performs better than using a small window size $w = 1$, whereas the opposite is true for those links with PRR higher than 0.8. The performance difference on the different links highlights a weakness of using a fixed set of parameters in link estimation: even if the parameter is optimally selected for a particular set of links, it might not work well for other links due to the difference in the link behaviors. On the other hand, the prediction model can use the online learning techniques to adapt to the dynamics of each individual link, and therefore performs better than statically trained models for links with varying qualities.

3.4.5. Detailed Results – Weight Update. We further analyze the detailed weight update behavior of the LR model with s-ALAP in order to understand how each parameter in the model input contributes to the PRR prediction. In Figure 6, we show the correlation between RPP variations with respect to the corresponding weight changes of the LR model over time for a typical intermediate link. Figure 6(a) presents the short-term PRR variations (computed with 1-second interval) of a link with overall PRR of 0.7. The link presented here shows frequent PRR variations during the first 250 seconds, and then the link quality drops below 0.4 in general for about 100 seconds. Around time $T = 400$ seconds, PRR of the link increases again and finally stabilizes to almost 1, indicating a near-perfect quality. On the other hand, we apply the online learning LR model with s-ALAP to the same link and plot the weights of the model over the same time duration in Figure 6(b). The three lines in Figure 6(b) are the weights corresponding to the three parameters included in the model input, namely, the PHY parameter (LQI in this case), PRR calculated by the WMEWMA estimator, as well as constant bias term with a value of 1.

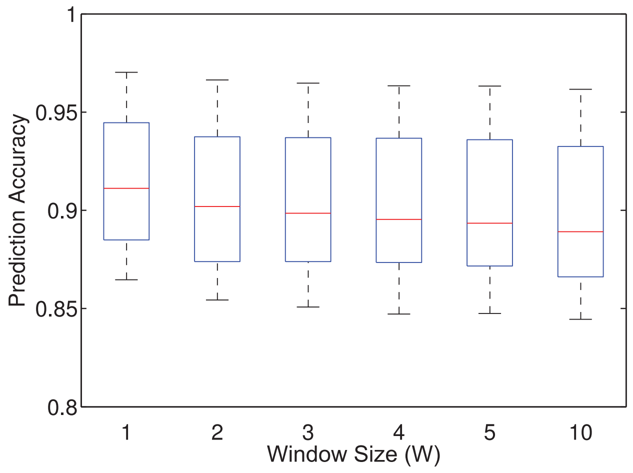


Fig. 7. Prediction accuracy of models using SGD with s-ALAP as a function of the window size W. The x-axis represents the window size, whereas each box plots the distribution of the predication results of the model using W labeled in the x-axis. The inter-packet interval is 0.1 seconds.

Comparing Figure 6(a) and Figure 6(b), we see that, during the first 250 seconds, the model updates its weights frequently, indicating that it is actively trying to adapt to the varying link quality. At about time $T = 250$ seconds, the model converges to a stable set of weights as the PRR of the link drops below the 0.9 threshold consistently. Then, as the quality of the link rises again at around $T = 400$ seconds, the model observes more high-quality periods and starts to update the weights to reflect the PRR changes. Finally, after $T = 500$ seconds, the link stabilizes to a near-perfect quality, and consequently the model also converges to a stable state. The close correlation between the link quality variations and the frequent weight changes suggests that the LR model with online learning actively adapts to the dynamics of the underlying link, resulting in an adaptive and agile estimation of short-term link quality.

3.5. Parameter Selection

This section compares performance of the prediction models with different sets of parameters to find the optimal parameter set for TALENT. Based on the problem definition in Section 3.1, there are three tunable modeling parameters: window size W, representing the number of packets used in a single input, the interval-packet interval I, representing the time interval between the packets, and the physical parameters, indicating which parameter from the physical layer should be used in the input. The following sections will discuss impact of these parameters individually. In addition, we also explore the choice of the decision threshold, namely the threshold applied to the continuous output of the LR model to determine the actual classification result (0 or 1), using ROC curves [Bradley 1997].

3.5.1. Window Size. The window size W is the number of packets in each input, which corresponds to the amount of historical information required by the model to predict the future link quality. Figure 7 plots the aggregated prediction accuracy of applying s-ALAP with different W in a box plot. The x-axis denotes W, and each box in the figure shows the median value of the probability distribution of the predication accuracy (red line in the middle), as well as the 5%, 25%, 75%, and 95% percentiles of the prediction accuracy of using W labeled in the x-axis. Other box plots in this section are plotted in similar fashion.

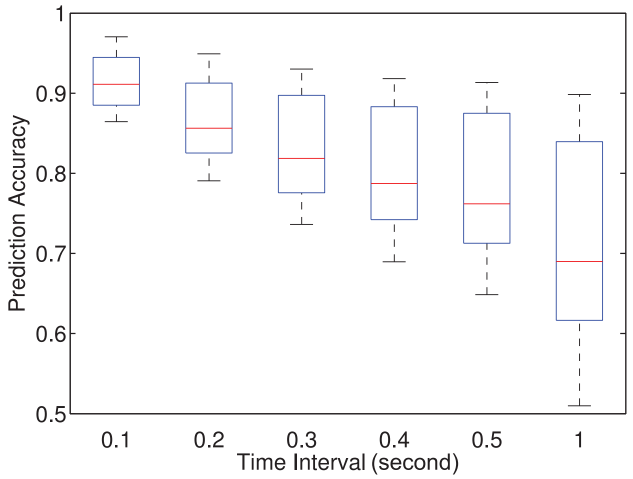


Fig. 8. Prediction accuracy of models using SGD with s-ALAP as a function of the inter-packet interval I. The window size W is 1.

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, Figure 7 shows that the prediction accuracy actually decreases slowly as we increase W from 1 to 10, implying that only the most recent packet is important for the prediction.

This counterintuitive result highlights the rapid link quality variations in low-power wireless communication. Due to the short coherence time of the wireless channel [Rappaport 2001], the historical channel quality may soon become uncorrelated with current quality, and consequently, the packet information cannot reflect the link quality at the current moment. In other words, there is no point to look at a long history, when such history becomes soon of no use. Moreover, the slow degradation of the prediction accuracy suggests that incorporating packet information from long past in the model input has an adverse impact on the prediction accuracy: since the past information is almost irrelevent to the current link quality, it only adds to the noise in the model input. Therefore, we use window size of 1 ($W = 1$) in our evaluation and implementation of TALENT.

3.5.2. Inter-Packet Interval. The inter-packet interval I represents the time interval of the packets used in the input. The longer I, the older is the packet reception information used by the model. Therefore, the prediction accuracy should degrade as I increases due to the short coherence time of the wireless channel discussed in the previous section. Figure 8 illustrates the distribution of the prediction accuracy of SGD with s-ALAP with respect to different inter-packet intervals. The window size W is 1 in this figure.

As predicted, Figure 8 shows that the median prediction accuracy drops from above 0.9 to below 0.7 as I increases from 0.1 seconds to 1 seconds, suggesting that the links experience significant temporal dynamics. Please note that, although shorter Is give better results, in practice the network traffic is controlled by the upper-layer applications and might not show short inter-packet intervals. In this case, the prediction result will not reflect the short-term link quality accurately. This limits the applicability of the prediction model to high data rate traffic only, as the prediction is only accurate enough when the packet information is from the recent past. Based on the results shown in Figure 8, we consider the prediction result is only valid for 1 second.

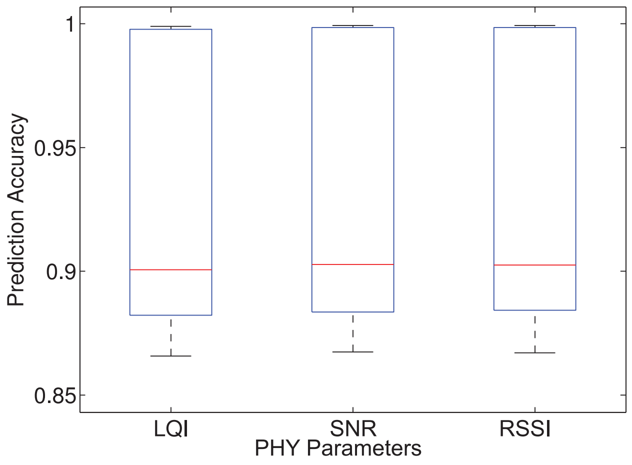


Fig. 9. Prediction accuracy of SGD/s-ALAP models using varying physical parameters in the input. The window size $W = 1$, and the inter-packet interval $I = 0.1$ seconds.

3.5.3. Physical Parameters. The choice of physical parameters, namely RSSI, SNR, and LQI, is directly related to the input of the model. To evaluate the effect of using different physical parameters in the input, we apply SGD with s-ALAP on the empirical packet traces and compare the prediction accuracy of using each parameter. The results are aggregated and plotted in Figure 9.

Figure 9 presents the aggregate prediction accuracy of s-ALAP with different physical parameters in a box plot. The window size W is 1 and the inter-packet interval is 0.1 seconds. From this figure, it is obvious that which physical parameter to use does not really affect the prediction accuracy, which confirms our previous findings in Liu and Cerpa [2011]. In the remainder of this article 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.

3.5.4. Decision Threshold. Another important parameter for the LR classifier is the decision threshold. In general, binary classification models that produce continuous output (e.g., estimation of class membership probability of a given input) need to use a predefined decision threshold to predict the actual classification: if the estimated probability is higher than the threshold, the classification result will be positive (1), whereas if the estimated probability is lower than the threshold, the classification result will be negative (0). In our case, TALENT also needs to select a decision threshold as the output of the underlying LR model is continuous. The analysis of the optimal decision threshold is often visualized by the Receiver Operating Characteristic (ROC) curves [Bradley 1997].

ROC curve is a plot of the true positive rate against the false positive rate of a classifier at various threshold settings. The true positive rate represents the sensitivity of the classifier, that is, proportion of actual positives correctly identified, whereas the false positive rate is one minus the true negative rate, that is, the proportion of correctly identified negatives. In the case of TALENT, the LR classifier should achieve a high true positive rate (e.g., correctly predict high-quality periods) while maintaining a low false positive rate. In Figure 10, we plot the ROC curves for two links with PRR equal to 0.7 and 0.57, respectively.

In both Figure 10(a) and 10(b), we can see that areas under the ROC curve (AUC) are higher than 0.9, which indicates a high overall accuracy of the online prediction

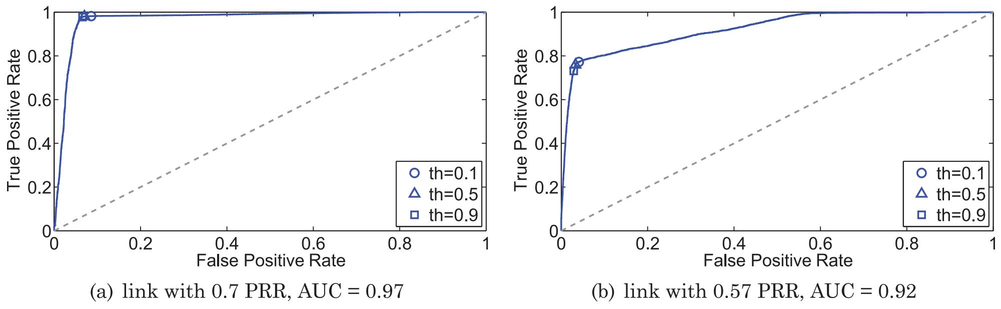


Fig. 10. ROC curves of TALENT with varying decision thresholds.

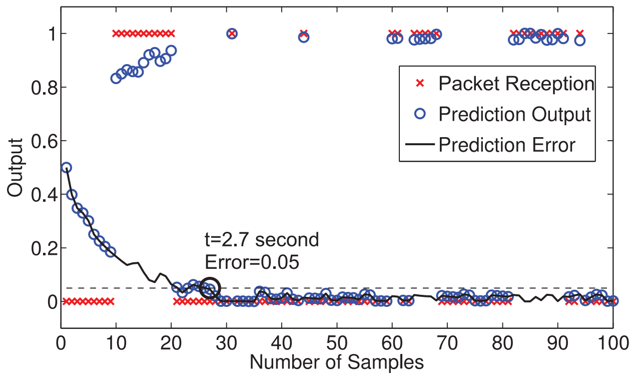


Fig. 11. Prediction error with respect to time.

model. We also note the points where the decision threshold equals to 0.1, 0.5, and 0.9 in the both figures. In Figure 10(a), we see that the TALENT models with different decision thresholds perform similarly and all locate in the turning point of the ROC curve. This observation indicates that any threshold within 0.1 to 0.9 range will yield a good prediction model with high sensitivity (true positive rate) and low false positive rate. Figure 10(b) shows similar results. Overall, we select 0.5 as the decision threshold for the LR models in this work.

3.6. Convergence Speed

We now investigate SGD with s-ALAP in terms of convergence speed. In the case of adapting to link dynamics, we want the model to adapt to the new distribution with as few new samples as possible, that is, update to the new weights with only a few packets. Figure 11 shows how the prediction error evolves during the first 10 seconds when s-ALAP is applied to a bursty link with 54% PRR. The crosses in the figure denote the packet reception (0 means lost, 1 means received), and the solid line represents the prediction error, namely the absolute difference between the actual packet reception (1/0) and the model output. As seen in Figure 11, in the first 2 seconds right after initializing the node, the prediction error starts from 50% and quickly declines as the s-ALAP algorithm updates its weight for the link. By the time $t = 2.1$ seconds, the prediction error already drops below 5%. After 2.7 seconds, the error stabilizes and never exceeds the 5% mark. Given the 0.1 seconds inter-packet interval, the SGD model with s-ALAP took about 20–30 samples (2 to 3 seconds) to converge. This number is quite consistent for all the 18 packet traces: on average, SGD with s-ALAP needs 25 ± 4 packets to reduce prediction error to below 5%.

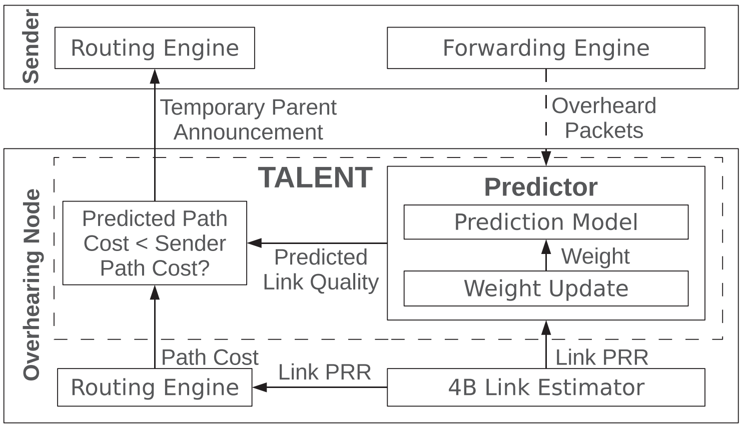


Fig. 12. TALENT overall design.

3.7. Summary

To summarize, we propose to use machine learning methods to build models that can predict the short temporal high link quality periods for intermediate links. Through analysis with empirical data traces, we showed that LR-based models using PRR and LQI values can predict the instantaneous PRR in the near future significantly better than WMEWMA for intermediate links. Moreover, by using the SGD online learning algorithm and the s-ALAP learning rate adaption method, the model is able to adapt to individual links and network dynamics in around 2 to 3 seconds without prior data collection or training. This adaptive behavior is a major advantage over 4C [Liu and Cerpa 2011], which also uses an LR-based prediction model that requires a priori offline training but does not adapt to network dynamics well.

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

4. SYSTEM DESIGN

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

4.1. Overview

TALENT is implemented as an extension module of CTP, the default collection protocol in TinyOS. Our design is receiver initiated: the prediction model in TALENT works in an overhearing node and notifies the sender if a better path is available. As illustrated in Figure 12, the predictor takes the LQI of the overheard packet and combines it with the link PRR given by the 4B link estimator to predict the future link quality. The predicted link quality is then added to the node's routing cost to compute the expected path cost if the packet was to be forwarded by the overhearing node. If the expected cost is smaller than the cost of the current forwarding path, the overhearing node notifies the sender about the availability of the new path, announcing itself as a new Temporary Parent (TP). On receiving the TP notification, the sender uses the TP as the next hop until the TP notifies the sender again to denounce its TP status. The sender reverts back to use the original next hop when the TP denouncement packet is received, or when a number of packets are lost consecutively.

The receiver-initiated design is similar to 4C [Liu and Cerpa 2011] and STLE [Alizai et al. 2009] since all of them allow the overhearing nodes to become temporary parents. However, there are multiple differences between TALENT and these two link estimators.

The main difference between TALENT and STLE is the prediction model. STLE is based on the heuristic that three consecutive packet receptions signify a high link quality period, whereas TALENT utilizes machine learning methods to model the link characteristics without assuming prior heuristics. Moreover, due to the adaptive online learning algorithm, TALENT will be able to adapt to network conditions when the STLE heuristic does not apply.

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

4.2. Temporary Parent Announcement

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

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

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

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

$$C_{S\to O} + 1 < C_{S\to P} + C_P, \tag{18}$$

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

ALGORITHM 1: s-ALAP Weight Update Rule

```
Input: Input $x_j(t) = [PRR(t), LQI(t)]$, output $y(t)$, instantaneous PRR $PRR_{Inst}(t)$ and meta
    learning rate $q$
Output: Updated weights $W(t)$ and learning rate $\lambda(t)$
 1: if $PRR_{Inst}(t) > 0.9$ then
 2:    $target(t) \leftarrow 1$
 3: else
 4:    $target(t) \leftarrow 0$
 5: end if
 6: for $j = 1 : D$ do
 7:    $gradient_j(t) \leftarrow (target(t) - y(t))x_j(t)$
 8:    $\Delta W_j(t-1) \leftarrow \Delta W_j(t)$
 9:    $\Delta W_j(t) \leftarrow W_j(t)gradient_j(t)$
10:    $\overline{\Delta W_j^2}(t) \leftarrow 0.8\overline{\Delta W_j^2}(t) + 0.2\Delta W_j^2(t)$
11:    $\lambda_j(t) \leftarrow \lambda_j(t)\max\left(0.5, 1 + q\Delta W_j(t)\frac{\Delta W_j(t-1)}{\overline{\Delta W_j^2}(t)}\right)$
12:    $W_j(t) \leftarrow W_j(t) + \lambda_j()\Delta W_j(t)$
13: end for
14: return $W(t)$, $\lambda(t)$
```

4.3. Predictor Implementation

The predictor component (see Figure 12) implements the prediction model and performs weight update for the model. Overall, the LR-based prediction model computes a new link quality prediction for every overheard/received data packet, and overhearing nodes use the predicted link quality to calculate the routing cost in the temporary parent announcement process discussed in the previous section. The predictor also updates the weight of the prediction model for every 10 packets overheard/received from the same sender.

More specifically, the prediction model functions as an extension to the existing link estimator 4B. Whenever a new data packet is overheard or received, the predictor takes the LQI value of the packet, combined with the current estimated PRR of the link between the sender and the recipient node, to create an input for the LR-based prediction model. The PRR is estimated by 4B, which employs WMEWMA with the same parameters used in the performance comparison presented in Section 3. The prediction model outputs the predicted link quality from the sender to the recipient node for the next 1 second, which is then used to calculate the routing cost for the temporary parent announcement. In order to reduce the computation time for the prediction, we employ a linear approximation proposed by Amin et al. [1997] to accelerate the sigmoid function calculation required by the model. The measured execution time of prediction computation in TMote is 0.5 ± 0.004 ms.

The predictor also performs weight updates for the prediction model. Once a prediction is calculated, the predictor records the prediction output as well as the inputs, and then starts to measure the instantaneous PRR as the target of this prediction. Because all the packets are embedded with a monotonically increasing sequence number, the exact number of packets sent by a sender can be inferred by counting the gap between the sequence numbers of received packets. The instantaneous PRR is then calculated by dividing the number of packets overheard with the packet gap. In our implementation, we calculate the instantaneous PRR once the accumulated packet gap equals to or is greater than 10 such that the instantaneous PRR can reflect the real-time link quality after previous prediction. Once the instantaneous PRR is available, it is then used in the weight update along with the corresponding predictor input and output. The algorithm of the weight update is listed in Algorithm 1, which implements Eqs. (12) and (13).

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

4.4. Integration to Existing Network Stack

Overall, TALENT is implemented as an extension to the existing link estimator. As shown in Figure 12, TALENT communicates with almost all the components of the network stack, including the 4B link estimator, the routing engine, and the forwarding engine. We carefully designed TALENT such that it does not interfere with the normal operation of other components; furthermore, we made some modifications to 4B to take full advantage of the overheard packets.

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

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

An important design decision is when and how to preform weight updates due to the short effective period of the prediction. Conceptually, after the prediction model is updated, the prediction output is only valid for 1 second before network dynamics render the prediction inaccurate. In our design, TALENT performs weight updates once every 10 packets and uses a timer to keep track of the most current update. Any prediction output is marked invalid if the prediction is made after 1 second of the previous weight update. The intuition here is that when the traffic rate is high (e.g., inter-packet interval is 0.1 seconds), packets arrive at a fast rate so that the prediction model can be updated frequently and the output will be mostly valid, whereas in the low traffic rate cases, the timer will prevent the use of out-of-date predictions as the model will be updated less often. When a temporary parent has obsolete predictions, we take an opportunistic approach that allows the sender to continue sending packets to the temporary parent without notifying the expiration of the prediction. Due the the presence of the TP loss threshold, the sender realizes any potential link quality degradation and switches back to the old parent quickly.

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

4.5. Low-Power Listening

For energy-constrained sensor networks, Low-Power Listening [Polastre et al. 2004a] (LPL) is an important component that conserves energy by duty-cycling the radio. LPL periodically wakes up the radio to perform clear channel assessment (CCA) and turns off the radio if there is no activity detected. If there is activity in the channel, LPL keeps the radio on so that the MAC protocol can receive the potential packet. Once the packet is received, the MAC protocol will signal the receive event to upper layers, and LPL will put the radio back to sleep after a short wait period.

TALENT is implemented on top of BoX-MAC [Moss and Levis 2008], the default MAC protocol of CC2420 radio in TinyOS which supports overhearing. In BoX-MAC, LPL wakes up the radio purely based on the periodical CCAs, and therefore overhearing-based operations are perfectly functional with LPL. An overhearing node can wake up for channel activities to snoop for packets just like it were to receive the packets. Furthermore, since the CCA in BoX-MAC does not perform any address check, even a nonoverhearing node will have to wake up and receive the packet being transmitted when channel activities are detected. It is the upper-network layer's job to decide whether the packet is addressed to the node itself. In this sense, the energy overhead of overhearing nodes is only caused by the processing of overheard packets compared with nonoverhearing nodes, and the radio energy consumption is independent of using overhearing. Because of the aforesaid reasons, receiver-initiated approaches such as TALENT will work with LPL in BoX- MAC normally without incurring any significant overhead on the energy usage. We believe that TALENT could still work even if the MAC protocol does not directly support overhearing with LPL, as discussed in Section 6.

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

4.6. Memory and Computation Overhead

The memory overhead of TALENT is mainly due to the implementation of the receiver-initiated approach, namely the temporary parent announcement mechanism discussed in Section 4.2, as well as the coefficients (weights and learning rates) of the LR prediction model. In our implementation, the ROM size increased by 5269 bytes (from 28416

Table I. Current Draw of TMote Sky under Various Operation Conditions

MCU	Radio	Current (mA)
Idle	Off	<1
Busy	Off	2
Idle	RX	20
Idle	TX	19

Table II. Typical Execution Time of TALENT

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

to 33685 bytes) with the addition of TALENT, whereas the RAM requirement increased by 861 bytes (from 4019 to 4880 bytes). Given the 48kB flash and 10kB of RAM in the TMote Sky mote, the added memory footprint should not be a big concern.

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

To analyze the energy consumption overhead introduced by the prediction model, we measured the current draw and execution time of the prediction model running on TMote Sky as well as the current draw and packet transmission time of using the radio. The current draw was measured by a multimeter connected to a TMote and a 3-Volt power source in series, while we programmed the TMote running on different conditions, that is, with the radio on/off and with/without the prediction model continuously running. Table I lists the current consumption measurements, and Table II presents the time needed for the prediction and packet transmission operations. Our measurements are consistent with the TMote Sky datasheet [Moteiv Corporation 2013].

Table I shows that the current consumption of the MCU running the prediction model at full speed is 2 mA, whereas the radio typically consumes 20 mA when receiving and 19 mA when sending packets. Since transmitting a 30-byte packet and receiving its acknowledgment require about 5.25 milliseconds, the energy spent on transmitting a single packet would be able to support about 50 milliseconds of MCU computation time. As described in Section 4.4, the prediction model in TALENT executes two operations: calculate a new prediction for each data packet received/overheard, and update the weights every 10 packets received/overheard from a sender node. Given the 0.5-milliseconds execution time for the prediction calculation and 2.31 milliseconds for weight update listed in Table II, TALENT introduces a computation overhead of 7.31 milliseconds for every 10 packets received/overheard, or 0.731 milliseconds per packets on average. Compared with overhead of transmitting a packet, the energy overhead of the TALENT prediction model is only 1.5% of the energy of sending a single packet. Combined with the conservative temporary announcement mechanism, having TALENT is beneficial as long as the prediction model can save one packet every 67 predictions computed, and the temporary parent announcement mechanism will guarantee cost reduction as long as at least one packet is forwarded by the temporary parent. As shown in Section 5, the savings provided by TALENT are an order of magnitude larger than the minimal requirement discussed here.

5. EXPERIMENTAL EVALUATION

We evaluate the performance of TALENT in terms of end-to-end delivery cost, loss rate, and path length. The delivery cost refers to the total number of transmissions needed to deliver a packet to the root, the loss rate is the percentage of packets sent but never received at the root, and the path length refers to the number of hops in a delivery path. The performance of TALENT is compared against three other state-of-the-art link estimators, namely 4B [Fonseca et al. 2007], STLE [Alizai et al. 2009], and 4C [Liu and Cerpa 2011]. To take full advantage of the snoop interface, we updated the networking stack so it can use the overheard packets to update the ETX estimation. This modification is applied to all receiver-initiated estimators, namely TALENT, 4C, and STLE, so there is fair ground for performance comparison.

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

5.1. Experimental Setup

We conducted extensive experiments in three wireless testbeds: a local testbed, the Harvard Motelab [Werner-Allen et al. 2005] testbed, and the Indriya [Doddavenkatappa et al. 2011] testbed. The local testbed is comprised of 54 TMotes placed along the corridor of a typical office building. The Motelab testbed is a sensor network testbed composed of 180 TMotes deployed on three floors. Unfortunately, only 47 nodes were available at the time of our experiments due to node failures. The Indriya testbed consists of 127 TelosB motes deployed across three floors of the School of Computing of the National University of Singapore.

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

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

5.1.1. Testbed Settings. In order to evaluate the performance of TALENT in a diverse set of network environments, we perform our experiments in the three testbeds with many different settings. For the local testbed experiments, we vary the total number of nodes in the network in order to create different network density for each experiment.

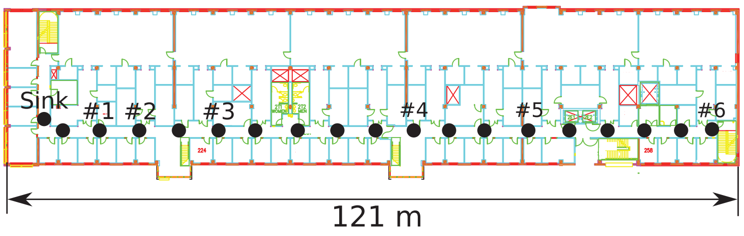


Fig. 13. Local testbed with 57 TMote Sky motes placed along a corridor of a typical office building. The motes are divided into 19 groups denoted by the black dots. The distance between each node group ranges from 6 to 7 meters, except for the node group in the far left which sits around a corner at the end of the corridor. The root node is denoted as Sink and the sender nodes in each experiment are denoted as no. 1 to no. 6. The number of nodes included in each experiment is 5, 6, 11, 25, 42, and 57, respectively.

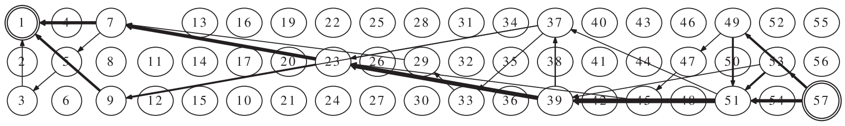


Fig. 14. A connection map of an experiment with 57 nodes in the local testbed. The width of the lines indicates the percentage of the total data packets being transmitted through the corresponding link. There are 26 unique paths in total and the paths with sparse packet transmissions are not shown.

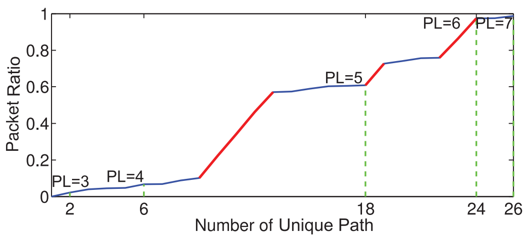


Fig. 15. The cumulative distribution of the ratio of the packets transmitted through each path with respect to the number of unique paths in the 57-node experiments. 26 unique paths are observed in this experiment, with the path length ranging from 3 to 7 as denoted in the figure. The distribution increases gradually, indicating that there is no single dominant data forwarding path in the local testbed.

Figure 13 illustrates the location of the sender nodes in each experiment, with 5, 6, 11, 25, 42, and 57 nodes included in the respective experiments.

Please note that, although the local testbed is deployed along a corridor of a typical office building, the network topology seen in our experiments is not completely linear. Figure 14 illustrates a map of the major data forwarding paths seen in a 57-node experiment. The directional lines indicate the links used to forward the data packets, and the width of the lines indicates the percentage of the total data packets being transmitted through the corresponding link. From this map we can clearly see a large portion of the packets traveled through the path 57 → 51 → 39 → 23 → 7 → 1, but there are actually 26 unique forwarding paths observed in total. This observation is further confirmed by Figure 15, which presents the packet distribution with respect to the number of unique forwarding paths. As seen in Figure 15, the major percentage of the total packets were forwarded through a few frequently used paths (marked as red), but a significant proportion of the total packets took other paths with different length ranging from 3 to 7. In general, the cumulative distribution of the packets travelling through each unique path increases gradually, indicating that there is no single dominant data forwarding path in the local testbed.

Table III. Node Pairs in the Motelab Experiments

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

Table IV. Node Pairs in the Indriya Experiments

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

In the Motelab and Indriya experiments, we fixed the total number of nodes in the network, but varied the sender/sink node pair to create a variety of network environments. Following the example node configurations proposed in Alizai et al. [2009], we used three types of sender/sink pairs to cover a rich set of geographically different network configurations, namely vertical, diagonal, and horizontal. Vertical configuration means the sender and sink node are on different floors and on the same end; diagonal configuration means the sender and sink are on different floors but on opposite ends; and in the horizontal configurations the sender and sink are on the same floor and on opposite ends. For all the Motelab experiments, the total number of nodes used is 47, and Table III lists the actual node pairs. For the Indriya experiments, the number of nodes is 127, and the node pairs used are listed in Table IV.

5.2. Link Estimation with Bursty Traffic

We perform some preliminary analysis to motivate the use of short-term link quality estimators under bursty traffic. We consider three link estimator settings: 4B with default WMEWMA parameter $\alpha = 0.9$, 4B with $\alpha = 0.1$, and TALENT. Intuitively, 4B with $\alpha = 0.9$ means the ETX calculation will give more weight to the historical link quality, making 4B insensitive to sparse link quality changes. On the other hand, 4B with $\alpha = 0.1$ assigns more weight to the current link quality, and hence increases the reactiveness of 4B. In the reminder of this section, we refer to 4B with $\alpha = 0.9$ as "stable 4B" and 4B with $\alpha = 0.1$ as "reactive 4B". These two different flavors of 4B are compared with TALENT experimentally under a bursty traffic pattern to examine the differences in path selection and the end-to-end delivery cost.

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

In this network, the sender sends packets in a bursty pattern: only one packet is sent every 30 seconds in the first minute, and in the second minute the sender sends 10 packets per second with a 0.1-seconds inter-packet interval. This pattern repeats in the remaining three minutes until the experiment ends at the end of fifth minute. The experiment is first conducted with CTP and stable 4B ($\alpha = 0.9$), and then repeated using reactive 4B ($\alpha = 0.1$) and TALENT back to back to ensure minimal environmental changes. The behaviors of the three estimators are illustrated in Figure 16. In each plot, the solid line shows how the path length evolves over the course of the experiment, the dashed line indicates the corresponding end-to-end delivery cost, and the circles

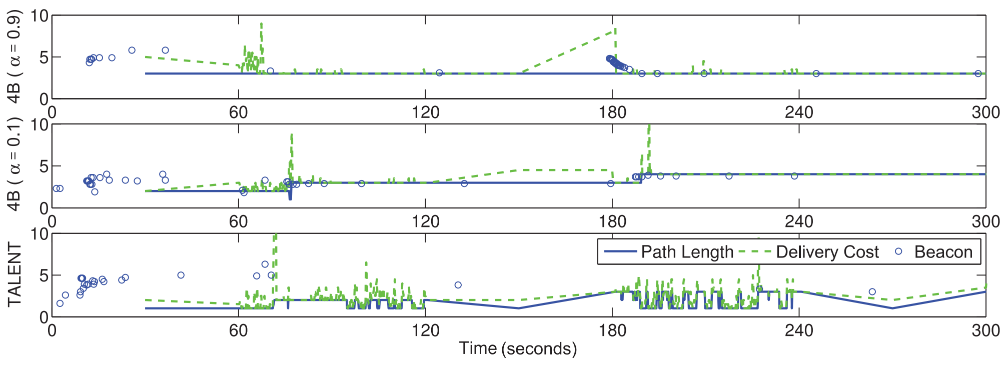


Fig. 16. The path length and the delivery cost of 4B (stable and reactive) and TALENT during a 5-minute experiment.

represent the beacons received by the sender S. For each circle, the x-axis notes when the beacon is received and the y-axis is the estimated delivery cost of the beacon sender.

Judging from the path length shown in the top plot in Figure 16, it is clear that with stable 4B, CTP took the longer path (4 hops) from the beginning and never strayed away from it over the course of the experiment. The merit of this path is that the links are of high quality and stable, thus almost all the send attempts were successful and very little number of retransmissions were required, except for a few seconds after 60 seconds. This is reflected by the mostly smooth delivery cost in the plot. Note that, due to the stable WMEWMA estimator, the several seconds of high losses are not enough to make the stable 4B change its path. In other words, stable 4B selected a path with a cost per hop almost equal to 1 and never changed even with packet losses.

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

The situation is quite different from CTP using TALENT. As seen in the bottom plot in Figure 16, the path length is 1 at the beginning, indicating that the shortest path was selected. The cost of delivering a packet fluctuated when the data rate increased to 10 packets per second after 60 seconds, confirming that the $S \rightarrow R$ link is not stable and has intermediate quality. At around $t = 75$ second, CTP switched to a longer path due to quality degradation on the shortest path. However, TALENT enabled CTP to quickly switch back and fourth between the shortest path and longer path once the instantaneous link quality of $S \rightarrow R$ is high enough. This switching behavior can be clearly observed between 90 and 120 seconds, as well as from 180 seconds to 240 seconds in the experiment. Despite the cost fluctuation associated with the shortest path, the average delivery cost of CTP with TALENT is significantly smaller than both stable 4B and reactive 4B. In this experiment, the average delivery cost of stable 4B is 3.12, reactive 4B is 3.47, whereas TALENT is 2.32, specifically 34% smaller than stable 4B and 50% smaller than reactive 4B.

Why does'nt 4B switch to the shorter paths if they are available? The beacon distributions presented in Figure 16 offer an explanation. The beacon packet contains the link quality and delivery cost estimation from the beacon sender, and the recipient node of the beacon can compute the estimated delivery cost assuming the beacon sender as its parent. According to the CTP adaptive beaconing policy, all nodes send beacon packets frequently to establish the initial link quality estimations and select parents at the beginning of the experiment, but the inter-beacon interval grows exponentially as a stable route is established. This can be observed in all the three plots in Figure 16: the beacon packets received by the sender are clustered within the first 30 seconds of the experiment, whereas only several beacon packets were received in the remaining time. Once CTP switched from a short path to a longer path due to link quality degradation, the ETX estimation of the old parent only reflects the bad link quality that caused the switch. For CTP to use the shorter path again, the previous bad-quality estimation must be updated to reflect the current path quality. Unfortunately, the sparse beacon packets could not adjust the estimation due to the EWMA filter fast enough, even for the reactive 4B. In this case, the skewed ETX estimation will prevent CTP from utilizing the shorter path over the course of the experiment. On the other hand, TALENT constantly updates its link quality estimation by using the overheard data packets and follows closely with the actual link quality. Even though the ETX estimation was skewed by the lossy periods of the link several times, CTP was able to switch back to the shorter path as soon as TALENT indicates a high-quality period is available on the shorter path. This observation is based on a simple linear network, but it is also applicable to larger networks. In a dense network, CTP may have more links to choose from and may find alternative routes with small end-to-end costs. However, a dense network also means that the number of potential temporary parents is large, and this allows TALENT to find shorter routes as well. The relative savings of TALENT versus 4B across different network densities remain relatively constant, as shown in the following section.

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

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

5.3. Path Length vs. Delivery Cost

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

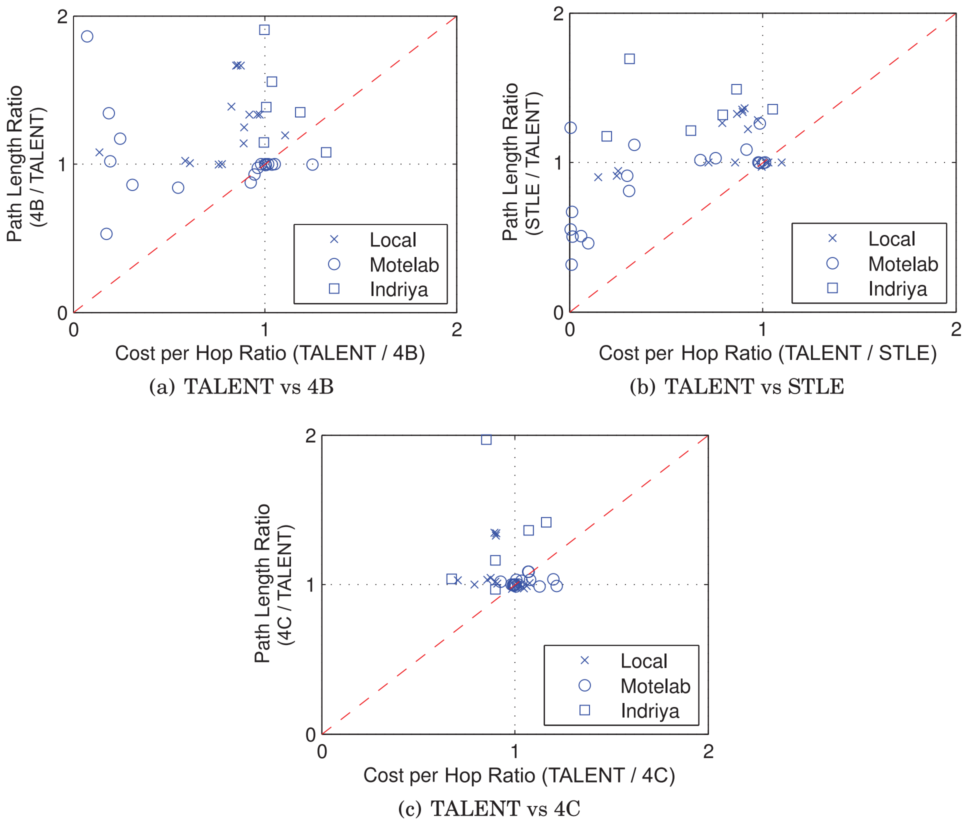


Fig. 17. The average delivery cost per hop ratio of TALENT and (4B/STLE/4C) versus the path length ratio (4B/STLE/4C) and TALENT in all the single-sender experiments. The marks above the line $x - y = 0$ indicate better overall costs of TALENT over the other link estimators.

intermediate-quality links compared with high-quality links, the end-to-end delivery cost is ultimately reduced due to the lower number of hops.

To study the trade-off between a longer path with stable high-quality links and a shorter path with unstable intermediate quality in larger networks, we extend our evaluation by comparing the behavior of TALENT with respect to 4B, STLE, and 4C in extensive single-sender experiments conducted in three wireless testbeds: the local, Motelab, and Indriya testbeds. The experimental conditions were described in Section 5.1 in detail. Each experiment was repeated three times with the same network settings, that is, packet length, radio-power level, and node configuration in the network. The 4C modeling parameters were assigned based on the LR model trained on training sets collected at each testbed, and this training cost is not included.

To make our point clearer, we break down the end-to-end delivery cost into the hop count times the cost per hop. Figure 17 shows the average delivery cost *per hop* ratio of TALENT and 4B/STLE/4C versus the path-length ratio of 4B/STLE/4C and TALENT for the experiments conducted in the three testbeds. In general, if the rate of cost per hop increase is smaller than the rate of path-length decrease, the overall cost for delivering packets is reduced. For example, if $\frac{CostPerHop(TALENT)}{CostPerHop(4B)} < \frac{PathLength(4B)}{PathLength(TALENT)}$, TALENT achieves lower cost than 4B. Therefore, any point plotted above the line $x = y$ indicates TALENT has lower delivery cost than 4B in one experiment, and vice versa.

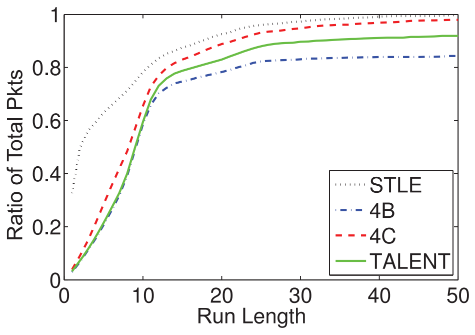
As shown in Figure 17, the majority of the experiment results were marked above the $x = y$ line, indicating that the overall delivery cost of CTP with TALENT is better than the other estimators. For the local testbed and the Indriya testbed experiments, a large group of the results exhibit small or even negative cost increment while using shorter paths, implying that using TALENT can reduce the path length while maintaining the delivery cost. There are also cases where the path length of TALENT and the other estimators are the same but the cost of TALENT is much smaller (see Figure 17(a) and 17(b)). This is due to a poorly connected network and/or sudden link quality changes that cause 4B and STLE to send excessive retransmissions before switching to another path. In this case, TALENT enables fast route updates with such network dynamics by taking advantage of the temporary parent mechanism: the overhearing node can notify the sender about alternative routes even if the ETX estimation on the sender side is lagging behind the link quality changes.

For the Motelab experiments, this behavior is more obvious. Originally, the Motelab testbed had 180 nodes, but the number of nodes had been reduced to 47 at the time we conducted our experiments. Due to the sparsely connected network, the number of possible routes are limited and the use of intermediate links is almost unavoidable in some cases to avoid network partitions. From Figure 17(a) and 17(c), it can be observed that many of the Motelab experiments for 4B and 4C are clustered around (1,1), indicating that both TALENT and 4B/4C took similar paths. However, several experiments show drastic cost reduction of TALENT over 4B and STLE, particularly when the path-length ratio does not show a particular trend. This is because the crucial links in the forwarding path were suddenly broken, in which case the 4B and STLE estimators could not find an alternative path fast enough. As a consequence of the network partition, the data packets were accumulated in the forwarding nodes and eventually dropped before a new route is established, causing low end-to-end delivery rate in addition to high delivery cost. On the other hand, TALENT can recover quickly from such dynamics due to the fast adapting predictor and the receiver-initiated approach of temporary parent selection.

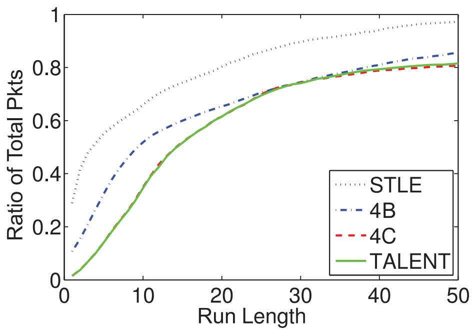
While the overall performance of TALENT is still better than 4C on average, the improvement is smaller when compared with the previous two estimators. This can be seen by the smaller distance from all the points towards the identity line. It should be noted that 4C was extensively trained using a priori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

5.4. Run-Length Analysis

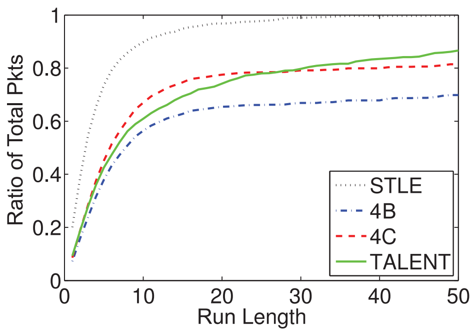
To further understand the characteristics of the wireless links in the experiments, we extend our analysis to the run length of the links that were used in the data forwarding path. If we consider the packet reception of a link as a binary string comprised of 0s and 1s whereas 0 represents packet loss and 1 represents packet successfully received and acknowledged, the run length of the link can be defined as the number of consecutive 1s in the binary string. Therefore, the distribution of the run length of a link can represent the stability and the overall quality of the link: a large number of short run length suggests that frequent packet losses occur in the link and the overall quality is low, whereas long run length implies sparse packet losses and high link quality. In our case, multiple links are selected by the routing protocol to form the data forwarding path during each of the experiments, so the aggregated run-length distribution of all the links used in the forwarding path will give us some useful insights on the overall link quality variations of the routing path. Figure 18 presents such run-length distributions for all the experiments run in the three testbeds.



(a) local testbed



(b) Motelab testbed



(c) Indriya testbed

Fig. 18. CDF of the ratio of packets sent through a particular run length with respect to the total number of packets sent in the data forwarding path. The run length is labeled in the x-axis and limited to [1, 50] range. The results from different link estimators are marked with different line styles.

Figure 18 contains three subfigures, and each subfigure plots the CDF of the ratio of packets sent through a particular run length (labeled in the x-axis) with respect to the total number of packets sent in all the experiments done in the respective wireless sensor testbed. Results from experiments done with the four link estimators, namely STLE, 4B, 4C, and TALENT, are plotted with different styles and colors to show the link selection preferences of these link estimators. We limit the maximum run length shown in each figure to 50 because including larger run lengths will skew the figure and hide the important details in the [1, 50] run-length range. Plus, only a small fraction of the total packets are sent with run length larger than 50. Therefore, we consider the CDF of the ratio of packets sent through run lengths between [1, 50] range is illustrative enough to show the characteristics of the links selected by a routing protocol with the four different link estimators.

Figure 18(a) shows a clear distinction between the four link estimators. In the case of STLE, more than 80% of the packets are sent with run length smaller than 10, suggesting that STLE strongly prefers links with frequent packet losses. This behavior of STLE is consistent for all the experiments done in the other two testbeds, as shown in Figure 18(b) and 18(c). Furthermore, combined with Figure 17(b), we see that STLE tends to aggressively select long, intermediate-quality links to form short routing paths. However, these long links cause excessive packet losses and offset the advantage of having a short routing path, ending up with a higher overall cost.

4B shows an opposite trend of STLE. In the local testbed and the Indriya testbed, 4B sent a high ratio of packets with long run lengths, which is reflected by the low curves

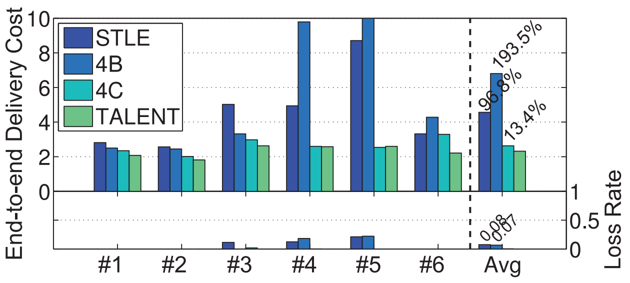


Fig. 19. End-to-end delivery cost and loss rate of local testbed. The experimental settings are described in Figure 13. Each bar represents the average results of 3 experiments with the same network settings.

in Figure 18(a) and 18(c). This behavior shows that 4B tends to use high-quality links in the forwarding path. Again, combined with Figure 17(a), it can be inferred that the general strategy of 4B is to use high-quality links to construct long but reliable paths. Nevertheless, as shown in the previous section and the later sections, this consecutive strategy is not optimal compared with 4C and TALENT.

On the other hand, due to the sparsely connected network in the Motelab testbed and limited number of possible paths, 4B is forced to use more intermediate links in the Motelab experiments. As shown in Figure 18(b), 4B sent a large number of packets with short run lengths, causing a higher packet ratio than 4C and TALENT in the range between 1 and 20. As discussed in the previous section, this high packet ratio in the short run-length range suggests that 4B tried to send packets through links with degraded quality, whereas 4C and TALENT are able to switch to alternative routes faster than 4B, resulting in lower overall costs.

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

5.5. End-to-End Delivery Cost and Loss Rate

In this section, we further present the performance of TALENT in terms of end-to-end delivery cost and loss rate, as well as the network settings for each experiment. As mentioned in Section 5.1, the results presented here are from extensive single-sender experiments conducted in three wireless testbeds: the local, Motelab, and Indriya testbeds.

Figures 19, 20, and 21 show the average end-to-end delivery cost per packet sent on the top and the end-to-end loss rate at the bottom of each figure. The numbers on the x-axis mark the experiment number, representing different network settings. Each column represents the results of three experiments conducted in the same network under the same conditions using the same sender.

Each experiment uses STLE, 4B, 4C, and TALENT for 10 minutes back to back. The last column in each figure shows the average of all the experiments in each testbed. Overall, we see that TALENT provides the overall best packet delivery cost, with average improvements over all testbeds of 18% over 4C, 145% over STLE, and 119% over 4B. Moreover, TALENT reduces the end-to-end loss rate on average over all our experiments by 1.5% over 4C, 17% over STLE, and 6.7% over 4B.

From Figure 19 experiments 4 and 5, and from Figure 20 experiments 1 and 3, we see that the delivery cost of 4B is very high. This behavior can be explained by the problems

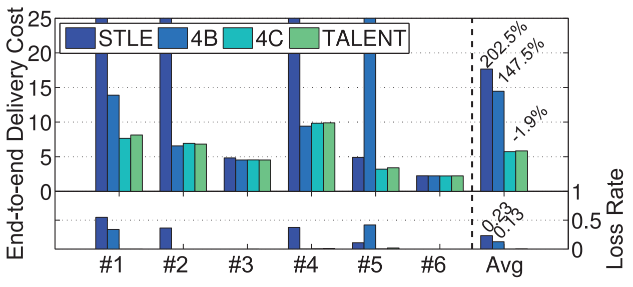


Fig. 20. End-to-end delivery cost and loss rate of Motelab testbed experiments. Motelab testbed is sparsely connected, so the number of good paths is limited, which leads to similar cost when the network is stable and heavy cost increments when the path is disturbed.

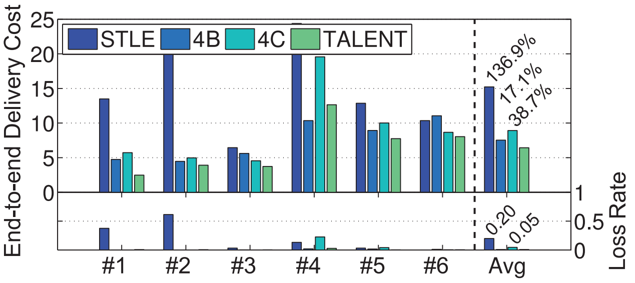


Fig. 21. End-to-end delivery cost and loss rate of Indriya testbed.

explained in Section 5.2, namely, slow beaconing activity on alternative paths and slow EWMA convergence. Trace analysis indicates that in these experiments, the number of available forwarding paths is limited. If the old forwarding path was broken due to link failure, the sender could not find an alternative parent and had to wait for the beacon from neighboring nodes to update the link quality. This leads to many retransmissions and eventually packet losses, whereas TALENT maintains a connected network due to the receiver-initiated approach. For example, in Figure 8 Exp no. 5, the 4B's end-to-end loss rate is close to 25%, which corresponds to a high delivery cost due to the excessive send attempts to a parent that's no longer available. When we run TALENT under the same conditions in the same network, it achieves near 0% loss rate because the overhearing nodes can become temporary parents as soon as the sender's old parent is no longer reachable.

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

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

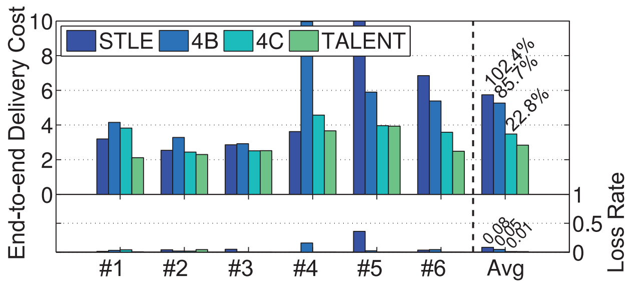


Fig. 22. End-to-end delivery cost and loss rate of variable sending rate and single sender.

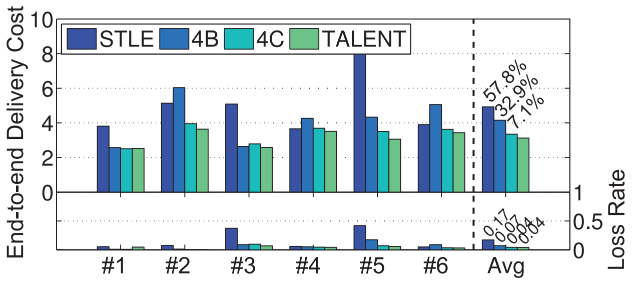


Fig. 23. End-to-end delivery cost and loss rate of variable sending rate and multiple senders.

5.6. Variable Rate and Multiple Senders

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

The variable-rate experiments used the same network settings in local single-sender experiments, the only difference being the inter-packet interval that is randomly selected in $[50, 150]$-ms range instead of using a fixed interval of 100 ms. As shown in Figure 22, the end-to-end delivery cost and loss rate results are similar to the fixed interval experiment results observed in Figure 19, indicating the variable rate does not affect the performance of TALENT significantly.

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

Figure 23 shows that while TALENT still outperforms STLE and 4B by 57.8% and 32.9% in terms of delivery cost, the cost reduction is smaller than with only one sender. Trace analysis shows that multiple senders create more data forwarding paths compared with single-sender experiments, and hence help the 4B link estimator to evaluate more links with higher rate.

6. DISCUSSION

6.1. Performance in 802.11 Networks

The results presented in Sections 3 and 5 are encouraging as they clearly indicate the potential of the online prediction approach in low-power wireless networks based on

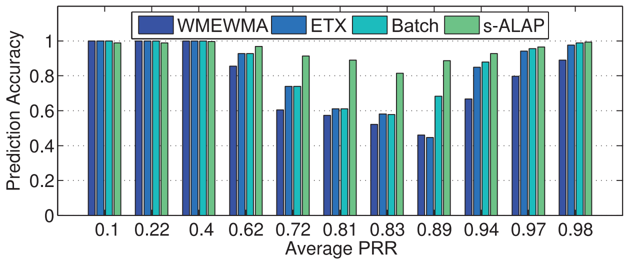


Fig. 24. Prediction accuracy of WMEWMA, ETX, batched-trained LR model, and LR model with s-ALAP applied to Rutgers dataset. Only 11 out of 112 total links are shown here to represent links with diverse qualities.

IEEE 802.15.4 standards. Moreover, our approach of using online learning prediction models to estimate the short-term wireless link quality with both link-layer and physical-layer information applies to wireless networks in general and is not restricted to low-power wireless networks. In order to analyze the potential application of the prediction model in high-power, high data rate networks, we evaluate the prediction approach with empirical packet traces collected from 802.11 networks.

We used two packet traces available in the CRAWDAD [2008] wireless network data repository. The first dataset is from Rutgers University noise dataset [Kaul et al. 2007], which is collected from an indoor wireless network testbed comprised of 128 IEEE 802.11a/b/g radio interfaces attached to 64 static nodes arranged on an 8-by-8 grid. The dataset includes more than 500 packet traces, each trace containing the received signal strength indicator (RSSI) for each correctly received frame at receiver nodes with certain levels of noise injected on the testbed, whereas the transmitter sends one beacon packet per 100 milliseconds. The testbed injects additive white Gaussian noise interference at center frequencies of 250 KHz to 6 GHz using an Agilent E4438C ESG vector signal generator.

The other dataset is from the indoor 802.11 signal strength measurements [Bauer et al. 2009] conducted by the System Research Lab from the University of Colorado at Boulder (UCB). This dataset provides a comprehensive set of RSSI readings from within an indoor office building. It captures RSSI behavior when 802.11 frames are transmitted using a stock omnidirectional antenna with the transmit power set to 16dBm. The omnidirectional RSSI measurements are collected from roughly 180 distinct physical locations throughout a large office building. The transmitter sends 500 packets from each of the 180 physical positions, and the measurement packets are recorded by 5 passive monitors, which are commodity Linux machines with 802.11 cards. Each RSSI measurement is labeled with the transmitter's physical location.

In order to evaluate the performance of TALENT in 802.11 networks, we apply four link estimators to the two datasets, namely: (1) WMEWMA with a strong smoothing factor $\alpha = 0.9$ and ETX calculation window of 5, which represents the long-term ETX-based link estimation used in 4B; (2) ETX computed with a window of 5 without any smoothing for reactive ETX estimations; (3) a batch-trained LR model fitted to each individual link, which represents the best accuracy the batch-trained LR model can achieve (i.e., overfitted); and (4) the online learning LR model with s-ALAP used in TALENT. The only modification made to the TALENT model is to include RSSI in the input instead of LQI.

Figures 24 and 25 present some of the prediction accuracy results of these four link estimators applied in Rutgers University dataset and the dataset from UCB, respectively. The x-axis in these figures denotes the average PRR of each link, whereas the bars represent the prediction accuracy of the link estimators. For better readability,

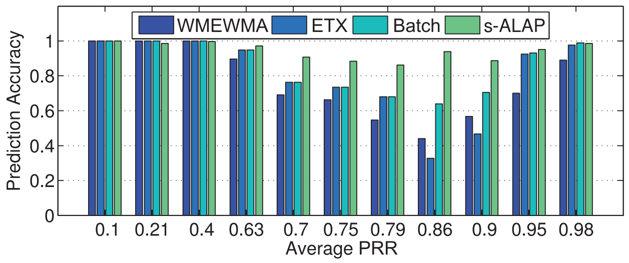
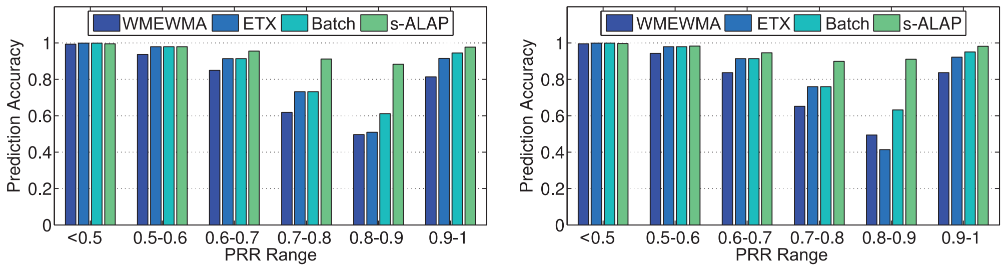


Fig. 25. Prediction accuracy of WMEWMA, ETX, batched-trained LR model, and LR model with s-ALAP applied to University of Colorado at Boulder dataset. Only 11 out of 130 total links are shown here to represent links with diverse qualities.



(a) Rutgers, 112 links total (b) UCB, 130 links total

Fig. 26. Average prediction accuracy of WMEWMA, ETX, batched-trained LR model, and LR model with s-ALAP, categorized with respect to PRR range.

we only include 11 links with diverse PRRs ranging from 0.1 to 0.98 in both figures, but the total number of links in these two datasets is much larger (113 in the Rutgers dataset, 130 in the UCB dataset). The overall prediction accuracy results are presented in Figure 26(a) and Figure 26(b), respectively.

In general, Figures 24 and 25 show that the online learning LR model used in TALENT performs significantly better than the other link quality estimators for those links with average PRR between 0.7 and 0.95, confirming the findings presented Figure 2. More specifically, the ETX estimation outperforms the WMEWMA for links with PRR higher than 0.8, but it is worse than WMEWMA for links with PRR ranging between 0.6 to 0.8. This result implies that neither the reactive ETX estimation, nor smoothed ETX estimation from WMEWMA can fit to a wide range of links with varying quality as a single rigid smoothing factor cannot capture the underlying dynamics of different links. The batch-trained LR model performs better than reactive ETX for links with PRR above 0.8 and is on par with WMEWMA for links with lower PRR. The online learning model in TALENT (s-ALAP) outperforms all three other link estimators, especially for links with PRR ranging between 0.7 to 0.95, indicating that TALENT can potentially better predict the link quality even in 802.11 networks. This trend can be also be observed in the overall prediction accuracy results from Figure 26. In the Rutgers results presented in Figure 26(a), the s-ALAP model used by TALENT consistently outperforms the other link estimators, especially for links with PRR between 0.7 to 0.9 range. In particular, s-ALAP achieves 88% prediction accuracy for links with PRR between 0.7 and 0.8, 38% higher than WMEWMA results for the same links. The results from the UCB dataset presented in Figure 26(b) are consistent with the Rutgers results.

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

6.2. Impact of Low-Power Listening

Based on the results presented in Section 5, TALENT works well with LPL enabled. However, if LPL is misconfigured, it could potentially reduce the performance of TALENT and CTP.

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

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

Fine-tuning these parameters is out of the scope of this article. Nevertheless, an application that generates bursty traffic could potentially set the LPL parameters to facilitate the bursty data transfer using TALENT.

6.3. Integration with Other MAC Protocols

TALENT relies on the assumption that the underlying MAC protocol supports overhearing, that is, snooping on packets that are not addressed to the node. In this work, TALENT is implemented on top of LPL, which is realized in BoX-MAC [Moss and Levis 2008], the default MAC protocol in TinyOS 2. As discussed in Section 4.5, a nice feature of BoX-MAC is that it provides the overhearing interface without incurring much overhead in terms of power consumption because the energy-based receive check (CCA) does not perform any address check. Note that some MAC protocols such as X-MAC [Buettner et al. 2006] perform address checks before starting to receive the data packets, which limits the use of the overhearing operation. In this case, using overhearing may negatively impact the performance of LPL. However, as pointed out by Moss and Levis [2008], on CC2420-based platforms, BoX-MAC consumes up to 40–50% less energy than X-MAC under reasonable workload. Therefore, we consider that our evaluation with BoX-MAC is sufficient. Furthermore, even if the underlying MAC protocol does not support overhearing with LPL, we believe that the PHY parameters could still be estimated when the MAC performs receive checks.

6.4. Limitations of TALENT

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

This limitation can be overcome by utilizing TALENT only when a batch of packets needs to be sent. We leave the decision to the application/network level as the higher-level protocols will have more control of when and how many packets to send. Ideally, TALENT-aware routing protocols should have two operation modes: low data rate mode, in which the TALENT is disabled and the LPL wakeup intervals are set to a large value, and bursty mode, in which TALENT is enabled and LPL incorporates short wakeup intervals. By doing local buffering and sending packets in bursts, applications allow TALENT to select the instantaneous low-cost paths, trading off increased latency for significantly larger delivery efficiency and smaller delivery costs.

7. CONCLUSION

Prior studies have shown that model-based predictors such as 4C significantly outperform link estimators such as STLE and 4B. However, the main disadvantage of 4C is the need to collect link data at the target deployment site for training the link prediction model. In this article we present TALENT, a self-learning, plug-and-play estimator to predict the quality of a wireless link in the near future using a combination of packet- and physical-level quality indicators. One of the main advantages of TALENT is the use of online learning techniques that are able to adapt to the wireless dynamics without the need for data collection and model retraining. When using TALENT together with CTP, our experimental results show that in many different environments TALENT increases the delivery efficiency more than 1.95 times in comparison to state-of-the-art link quality estimators.

ACKNOWLEDGMENTS

We would like to thank all the referees for their constructive comments and helpful suggestions. Also, special thanks to Jimei Yang and Nic Schraudolph for the valuable discussions on online learning.

REFERENCES

Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. 2004. Link-level measurements from an 802.11b mesh network. *SIGCOMM Comput. Comm. Rev.* 34, 4, 121–132.

Muhammad Hamad Alizai, Olaf Landsiedel, Jó Ágila Bitsch Link, Stefan Götz, and Klaus Wehrle. 2009. Bursty traffic over bursty links. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09).* ACM Press, New York, 71–84.

Luís B. Almeida, Thibault Langlois, José D. Amaral, and Alexander Plakhov. 1998. *Parameter Adaptation in Stochastic Optimization*. Cambridge University Press, New York, 11–134.

Hesham Amin, K. Memy Curtis, and Barrie R. Hayes-Gill. 1997. Piecewise linear approximation applied to nonlinear function of a neural network. *IEE Proc. Circ. Devices Syst.* 144, 6, 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 *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN'10).* Lecture Notes in Computer Science, vol. 5970, Springer, 240–255.

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

Dimitri P. Bertsekas. 2012. *Dynamic Programming and Optimal Control*. 4th Ed. Athena Scientific.

Léon Bottou and Yann LeCun. 2004. Large scale online learning. *Adv. Neural Inf. Process. Syst.* 16.

Léon Bottou and Noboru Murata. 2002. Stochastic approximations and efficient learning. In *The Handbook of Brain Theory and Neural Networks*, 2nd Ed., The MIT Press, Cambridge, MA.

Andrew P. Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.* 30, 7, 1145–1159.

Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. 2006. X-MAC: A short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor (SenSys'06).* ACM Press, New York, 307–320.

Alberto Cerpa and Deborah Estrin. 2003. SCALE: A tool for simple connectivity assessment in lossy environments. Tech. rep. 0021, UCLA.

Alberto Cerpa, Jennifer Wong, Louane Kuang, Miodrag Potkonjak, and Deborah Estrin. 2005a. Statistical model of lossy links in wireless sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05).* ACM Press, New York, 81–88.

Alberto Cerpa, Jennifer Wong, Miodrag Potkonjak, and Deborah Estrin. 2005b. Temporal properties of low power wireless links: Modeling and implications on multi-hop routing. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05).* ACM Press, New York, 414–425.

CRAWDAD. 2008. CRAWDAD: Community resource for archiving wireless data at dartmouth. http://crawdad.cs.dartmouth.edu.

Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. 2003. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th International Conference on Mobile Computing and Networking (MobiCom'03).* ACM Press, New York, 134–146.

John Dennis and Robert Schnabel. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall.

Manjunath Doddavenkatappa, Mun Choon Chan, and Akkihebbal L. Ananda. 2011. Indriya: A low-cost, 3d wireless sensor network testbed. In *Proceedings of the 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM'11).*

Richard Draves, Jitendra Padhye, and Brian Zill. 2004. Comparison of routing metrics for static multi-hop wireless networks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'04).* ACM Press, New York, 133–144.

Károly Farkas, Theus Hossmann, Franck Legendre, Bernhard Plattner, and Sajal K. Das. 2008. Link quality prediction in mesh networks. *Comput. Comm.* 31, 8, 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 *Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM'06).* ACM Press, New York, 239–246.

Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Levis. 2007. Four-bit wireless link estimation. In *Proceedings of the 6th Workshop on Hot Topics in Networks (HotNets'07).* ACM Press, New York.

Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. 2009. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09).* ACM Press, New York, 1–14.

Sanjit Krishnan Kaul, Marco Gruteser, and Ivan Seskar. 2007. CRAWDAD data set rutgers/noise (v. 2007-04-20). http://crawdad.cs.dartmouth.edu/rutgers/noise.

Dhananjay Lai, Arati Manjeshwar, F. Herrmann, Elif Uysal-Biyikoglu, and Abtin Keshavarzian. 2003. Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In *Proceedings of the IEEE Global Telecommunications Conference*. 446–452.

Yann Le Cun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 1998. Efficient backprop. In *Neural Networks: Tricks of the Trade*. Springer, 9–50.

Tao Liu and Alberto Cerpa. 2011. Foresee (4c): Wireless link prediction using link features. In *Proceedings of the 10th International Symposium on Information Processing in Sensor Networks (IPSN'11).* 294–305.

Tom M. Mitchell. 1997. *Machine Learning*. McGraw Hill.

David Moss and Philip Levis. 2008. BoX-MACs: Exploiting physical and link layer boundaries in low-power networking. Tech. rep. SING-08-00, Stanford University.

Moteiv Corporation. 2013. TMote sky datasheet. http://www.snm.ethz.ch/Projects/TmoteSky.

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

Joseph Polastre, Jason Hill, and David Culler. 2004a. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor (SenSys'04).* ACM Press, New York, 95–107.

Joseph Polastre, Robert Szewczyk, Alan Mainwaring, David Culler, and John Anderson. 2004b. Analysis of wireless sensor networks for habitat monitoring. In *Wireless Sensor Networks*, Springer, 399–423.

Gregory J. Pottie and William J. Kaiser. 2000. Wireless integrated network sensors. *Comm. ACM* 43, 5, 51–58.

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

Tal Rusak and Philip Levis. 2009. Burstiness and scaling in the structure of low-power wireless links. *SIGMOBILE Mobile Comput. Comm. Rev.* 13, 1, 60–64.

Murat Senel, Krishnakant Chintalapudi, Dhananjay Lal, Abtin Keshavarzian, and Edward J. Coyle. 2007. A kalman filter based link quality estimation scheme for wireless sensor networks. In *Proceedings of the IEEE Global Communications Conference.* 875–880.

Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. 2006. Experimental study of concurrent transmission in wireless sensor networks. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys'06).* ACM Press, New York, 237–250.

Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. 2008. The β factor: Measuring wireless link burstiness. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys'08).* ACM Press, New York, 29–42.

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

TinyOS. 2013. 2.X. http://www.tinyos.net.

Yong Wang, Margaret Martonosi, and Li-Shiuan Peh. 2007. Predicting link quality using supervised learning in wireless sensor networks. *ACM SIGMOBILE Mobile Comput. Comm. Rev.* 11, 3, 71–83.

Geoff Werner, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. 2006. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06).* 381–396.

Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. 2005. MoteLab: A wireless sensor network testbed. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05).* IEEE, 68.

Alec Woo, Terence Tong, and David Culler. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03).* ACM Press, New York, 14–27.

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 *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04).* ACM Press, New York, 13–24.

Jerry Zhao and Ramesh Govindan. 2003. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03).* ACM Press, New York, 1–13.

Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. 2004. Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Service (MobiSys'04).* ACM Press, New York, 125–138.

Marco Zuniga and Bhaskar Krishnamachari. 2004. Analyzing the transitional region in low power wireless links. In *Proceedings of the 1st IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON'04).* 517–526.

Received February 2012; revised June 2013; accepted September 2013