TALENT: Temporal Adaptive Link Estimator with No Training

Tao Liu and Alberto E. Cerpa
Electrical Engineering and Computer Science
University of California – Merced
{tliu,acerpa}@andes.ucmerced.edu

Abstract

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 paper, 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 4B, the state of the art link quality estimator, as well as improve 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

Keywords

Link quality estimation, Link quality prediction

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 level 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 [26, 35, 32], 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 [27], 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 energy consumption of communications as accurate link quality information is vital to achieve optimal routing topologies.

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 [12], 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 [24]. However, due to the short temporal dynamics of wireless channels and differences in hardware calibration [9], 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, WSNs routing protocols often utilize PRR based link estimation metrics such as ETX [13] instead of using PHY parameters directly. For example, CTP [18], the default collection protocol in TinyOS [31], uses ETX to measure link quality and create routing topologies.

An intrinsic problem of PRR based metrics is that they do not perform well when monitoring intermediate links that show often 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 [2] that have low average PRR in long term (PRR between 10% and 90%), but show continuous high quality in

SenSys '12, November 6–9, 2012, Toronto, Canada

short periods. Prior work [11] 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 paper, 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. Due to the dynamic nature of the wireless channel, such prediction will be valid for only a short period before the link quality changes. Nevertheless, with the knowledge of expected link quality in the immediate future, the routing protocol may be able to select efficient data forwarding path promptly during a burst transmission of data packets, and ultimately increase delivery efficiency and reduce communication costs.

An essential requirement of such 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 a estimator to improve its performance for a certain level of dynamics, this parameter set will not work in all the cases as we deploy in different environments or even as the temporal dynamics change in the same location.

Another important feature of the estimator is *plug-and-play*. Ideally, a link estimator should work on any network without pre-deployment efforts to tune the prediction model. Prior studies have shown that model based predictors such as 4C [21] significantly outperform link estimators such as STLE [2] and 4B [17], 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 training data or prior knowledge from the target deployment.

Based on the above requirements, we propose Temporal Adaptive Link Estimator with No 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 four fold. First, we show that by using online learning techniques, our prediction model can adapt to a wide range of network dynamics without prior training data and with fast convergence time. To our knowledge, this is the first attempt to introduce online learning techniques to adapt network link estimation parameters under environmental and network dynamics. Second, we designed and implemented TALENT in TinyOS and integrated it into CTP for a reference implementation. Third, we integrated TALENT with LPL [25], 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 improvment of packet delivery efficiency, with an average of 95.3% improvement over 4B [17] in many different scenarios, as well as an improvement in end-to-end delivery rate.

2 Related Work

Link quality estimation is a critical component for wireless communications in WSNs. Woo *et al.* [34] argued that PRR based metrics such as ETX [13] 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.* [17] proposed 4B, a hybrid link estimator that combines information from the physical, data-link and network layers. 4B uses the parameters from PHY layer as an indication of high quality wireless channel and ETX as link quality metric.

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 [37]. However, Zhao *et al.* [36] 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.* [19] confirmed that the expected packet success rate (PSR) can be approximated by SNR with a sigmoid function, but Son *et al.* [29] also showed that due to difference in radio and transmission power, there is a significant variation of about 6 dB in the threshold of the sigmoid function. Senel *et al.* [28] proposed a SNR based estimator that uses a Kalman Filter to processed SNR and estimates the PSR with a pre-calibrated SNR- PSR function. TALENT mainly differs from the above metrics in terms of modeling approach as detailed in the following sections.

It is well established that most of the link quality variations are observed in the intermediate quality links as pointed out by numerous studies [36, 34, 10]. Srinivasan *et al.* [30] 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.* [2] proposed a bursty routing extension (BRE) to CTP that utilizes a short term link esti-

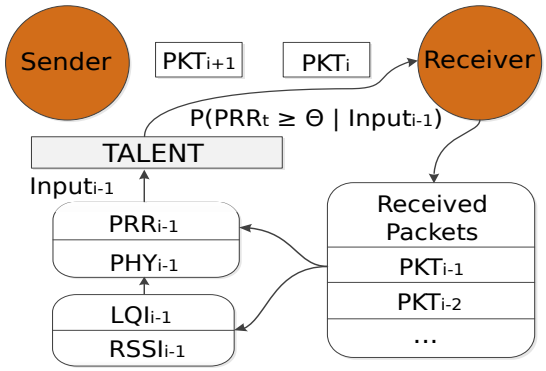


Figure 1. TALENT attempts to infer the probability that future PRR exceeds a certain threshold θ.

mator (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 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.* [16] proposed XCoPred, a pattern matching approach based on SNR to predict link quality variations, but their work was focused on 802.11 ad-hoc network. Liu and Cerpa [21] 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 a training data set. 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.

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:

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

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:

$$PKT_i = [PRR_i, PHY_i],\ PHY_i \subset (RSSI_i, SNR_i, LQI_i) \quad (2)$$

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 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:

$$P(PRR_t \geq \theta | Input_i) \quad (3)$$

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 a inter-packet interval of 0.1 seconds.We chose a small t and inter-packet interval because the temporal variation of the wireless channel is correlated at intervals smaller than 1 second [1, 30], and our model tries to take advantage of this behavior. We used the same parameters in our experiments in order to best approximate the real network conditions.

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

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 [22] to train a logistic regression classifier (LR) such that the model can adapt to the changing network conditions.We choose to use LR models because prior work [21] has shown that the link quality can be accurately predicted by LR models. We considered many online learning frameworks, such as weight majority, winnow and SGD, among others [5], and we settled for SGD mainly due

to its performance and simplicity to implement it under stringent computational and energy constraints.

Assume $X =< X_1 \ldots X_n >$ represents the input vector and Y is the binary variable denoting the high temporal link quality $PRR_t > \theta$, the logistic regression classifier can be expressed as:

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

and

$$P(Y=0|X) = \frac{\exp(-f(X)}{1+\exp(-f(X))} \quad (5)$$

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:

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

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 non-zero for any given Y^l.

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

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

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

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

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:

$$\beta_i \leftarrow \beta_i + \lambda \Delta\beta_i^l \quad (9)$$

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

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

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 [6]. Stochastic learning is also useful when the function being modeled is changing over time, a quite common scenario in networking where the data traffic patterns and the wireless channel quality variations are both non-deterministic. Also, stochastic learning often results in better solutions because the noise in the updates can cause the weights jumping into multiple, possibly deeper local minimum, whereas batch training will only converge to one minimum [20].

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 [14], online gradient descent proceeds rather slowly during the final convergence phase [7]. 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 [3] 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 the following equation:

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

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

$$\beta_i \leftarrow \beta_i + \lambda_i^l \Delta\beta_i^l. \quad (12)$$

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:

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

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 equation (12).

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

Another common extension of SGD algorithm is the use of momentum term [22]. With the momentum term, the weight update of lth sample becomes:

$$\beta_i \leftarrow \beta_i + \lambda^l \Delta\beta_i^l + m\Delta\beta_i^{l-1} \quad (14)$$

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.

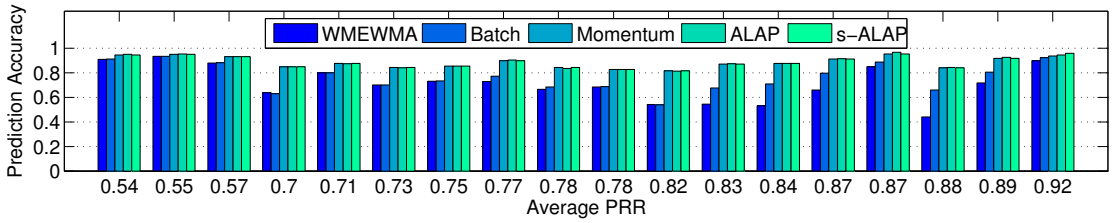


Figure 2. The prediction accuracy of LR models using three online learning algorithms, as well as a batch trained LR model and the WMEWMA estimator.

3.4 Online Learning Algorithm Evaluation

In order to select the best algorithm for predicting short term link quality when the network dynamics are non-negligible, we need packet traces of intermediate links to evaluate the candidate algorithms. The packet traces were collected from a local wireless testbed comprised of 54 TMote Sky motes. To collect the traces, one node was programmed to send 30-byte long packets with an inter-packet interval of 0.1 seconds for 1 hour, and all the other nodes in the network record the sequence number, RSSI and LQI of the received packets respectively.The wireless channel used was channel 26, and the sending node always used full power (RF power level = 0). After the sender node stops, the packet traces recorded from intermediate links are selected in the evaluation. The process was repeated 10 times, each time with a different sending node. The data was collected during three week days at different times of the day, including day and night times. In the end, we collected extensive packet traces with more than 490 thousand packets from 18 intermediate links with average PRR ranging from 0.54 to 0.89.

We then apply SGD with momentum, ALAP and s-ALAP to these empirical packet traces to evaluate their performance in terms of prediction accuracy, i.e., the ratio of the correctly predicted high quality periods to the total number of predictions made. In addition to these online algorithms, we also apply a WMEWMA estimator described in [34] as well as a batched trained logistics regression model (Batch) to compare their performance. Please note that the batch trained LR model was intentionally *over-fitted* to the specific link to maximize the accuracy of the batch trained model.

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 by the WMEWMA estimator, which can be expressed as:

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

where $\alpha = 0.9$ and the window size of ETX_{new} calculation is 5. These parameters are based on the default values used by 4B in TinyOS 2. In other words, for each input vector, the PRR is always the last WMEWMA's PRR estimate 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. As pointed out by prior studies [21], input of this size is enough for LR based models. We also experimented with RSSI and SNR as input, and our results indicate that including other PHY parameters do not significantly affect the prediction accuracy, which also confirms the results presented in [21].Therefore, we use PRR and LQI in our evaluation and omit the results of RSSI/SNR in the input vector for brevity.

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.

Figure 2 shows the prediction accuracy of the 5 link estimators with respect to the link PRR. It is clear that 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 batch trained model is better than WMEWMA, but its prediction accuracy is still consistently worse than any of the three online learning algorithms. This result implies that the best model for link quality prediction gradually changes over time due to frequent network dynamics: even if the LR model converges at a global optimal, the non-stationary wireless environments could soon make the static model obsolete. The online learning algorithms are ideal for tracking such non-stationary environments. In fact, all three methods achieved similar prediction accuracy with s-ALAP slightly better than the other two. Moreover, comparing with momentum, s-ALAP can select the learning rate adaptively, and consequently eliminates the need of selecting a global learning rate and momentum term. Therefore, we choose s-ALAP as the learning algorithm for TALENT.

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 the four categories: True Positives (TP), which means both the model output and the actual PRR is high; True Negatives (TN), meaning the output and the target PRR is both low; False Positives (FP), indicating the output are high whereas the actual target PRR is low; and False Negatives (FN), which means the output is low whereas the actual target is high. Then, the prediction accuracy is calculated as the ratio of TPs and TNs over all the four classes. The four categories are marked with different colors in Figure 3, and the prediction accuracy of the link are labeled on the top of each graph respectively. From the figure, it is obvious that WMEWMA performs the worst compared with the other two. Also the prediction accuracy of the batch trained LR model is lower

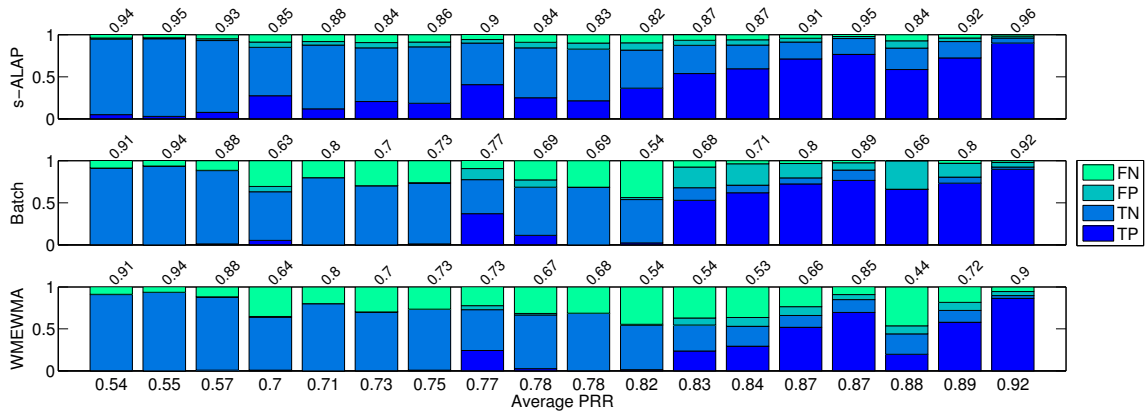


Figure 3. Prediction accuracy of three prediction methods applied to 18 intermediate links. The numbers on top of each graph shows the prediction accuracy, and the labels on the bottom indicate the link PRR. The detailed results (TP, TN, FP, FN) are marked with different colors for each link.

than using s-ALAP, suggesting that having a rigid model can not adjust to the changing network conditions. The size of the area for the TP cases (dark blue, bottom) indicates *the percentage of time an intermediate quality link could be used to forward packets with low losses*. TALENT is capable of detecting high quality periods for intermediate links and provides more viable paths for the routing protocol than WMEWMA. Note that WMEWMA is designed to estimate the average link quality, whereas the other estimators are designed and trained to predict the probability of the temporal link quality being high during time t in the future (in our case, $PRR_t > 0.9$). When the average PRR of the link is close but below the 0.9 threshold, WMEWMA tends to make many FN mistakes because it converges to the average link PRR that is below the threshold. Hence, WMEWMA fails to predict short intervals of time when the short term PRR is high, a very common event for a link with average PRR in the high 0.8 range. When the average link PRR is very different from the threshold (e.g., in the cases of links in the 0.5 range), 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.

3.5 Convergence Speed

We now investigate SGD with s-ALAP in terms of convergence speed. In the case of adapting to link dynamics, we want the model to adapt to the new distribution with as few new samples as possible, i.e., update to the new weights with only a few packets. Figure 4 shows how the prediction error evolves during the first 10 seconds when s-ALAP is applied to a bursty link with 54% PRR. The crosses in the figure denote the packet reception (0 means lost, 1 means received), and the solid line represents the prediction error, i.e., the absolute difference between the actual packet reception (1/0) and the model output. As seen in Figure 4, 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 $\pm$ 4 packets to reduce prediction error to below 5%.

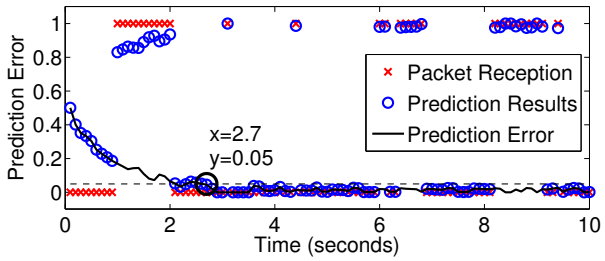


Figure 4. Prediction error with respect to time.

3.6 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 [21], which also uses LR based prediction model that requires a priori off-line training but does not adapt to network dynamics well. With these encouraging results, we show in the following section how we implement the LR based model with s-ALAP in TALENT to supplement the existing link estimators in TinyOS.

4 System Design

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 5, the predictor takes the LQI of the overheard packet, 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 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 [21] and STLE [2] 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. 4C and TALENT both use an LR model 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 re-collection for new training data if network conditions change. There is also no need to send the newly updated parameters to reprogram the nodes in this case. Moreover, 4C attempts to predict the success probability of the next packet, whereas TALENT predicts the probability of high quality periods.

4.1 Temporary Parent Announcement

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

In our scheme, we take the receiver-initiated approach

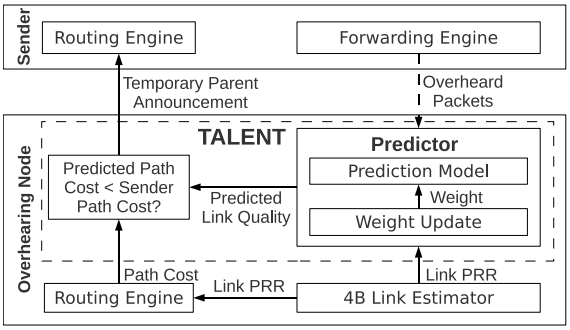


Figure 5. TALENT Overall Design

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

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

The link cost $C_{S\to O}$ is estimated by the underlying link estimator (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 $S \to O$ in a high quality period, and overrides the $C_{S\to O}$ with the minimum value of 1. In this case, if the new path cost of using O is smaller than the path cost of using P, then S should use O as a temporary parent. In other words, if the predictor indicates the link $S \to O$ is in a high quality period, and the following formula holds:

$$C_{S\to O}+1 < C_{S\to P}+C_P \tag{16}$$

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.

4.2 Predictor Implementation

The predictor is responsible for computing the prediction and performing the weight update for the model. The prediction calculation is done by the logistic regression model, which takes the LQI of overheard packets and the estimated PRR (from 4B) of the link between the sender and the overhearing node as the input. We employ a linear approximation proposed by H. Amin *et al.* [4] to accelerate the sigmoid function calculation required by the model. The measured execution time in TMote is 0.5 ± 0.004 ms.

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 $\lambda(t)$

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

The predictor also performs weight updates. Once a prediction is calculated, the predictor records the prediction output as well as the inputs, and then starts to measure the instantaneous PRR after the prediction. Because all the packets are embedded with a monotonically increasing sequence number, the exact number of packets sent 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. Once the instantaneous PRR is available, it is then used in the weight update along with the corresponding predictor input and output. The algorithm of the weight update is listed in Algorithm 1, which implements the Equation (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 tries to avoid multiplications and divisions as much as possible by replacing them with bit shift operations. Also, it only checks integer overflow when necessary, i.e., only check those operations that might involve large numbers. The measured execution time of a single weight update is 2.31 $\pm$ 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 (see Section 4.3)

Running the s-ALAP algorithm consumes extra energy. According to TMote Sky datasheet, the nominal current consumption of the MCU and radio on (TX or RX) is around 19 mA, whereas the current consumption of having MCU on and radio off is 1.8 mA. Assuming sending a 30-byte packet and receiving its ACK requires 5 milliseconds, the energy spent to transmit a single packet would be able to support more than 50 milliseconds of computation time. Given the 2.31 milliseconds execution time of the s-ALAP algorithm (less than 20 times the energy of a single packet), and that this operation is performed once every 10 packets in our implementation, we see that TALENT will save energy as long as it can save at least one packet every 200 packets sent.

4.3 Integration to Existing Network Stack

Overall, TALENT is implemented as an extension to the existing link estimator. As shown in Figure 5, 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 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 can not notify the sender even if it realizes the quality drop as the notification packet may get lost, and the sender will attempt to retransmit its packets until the route update mechanism of CTP kicks in and changes the parent node. To prevent this situation, we set a TP loss threshold that limits the maximum number of consecutive packet losses when a TP is set. In our implementation, the sender will switch back to the old parent after 5 consecutive packets losses instead of relying on the slow CTP route update. For all the switch back to old parent events seen in our experiments, only 12% of the cases were due to the loss threshold, whereas 54% and 34% of the cases were due to denounce TP notifications and CTP parent changes respectively.

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

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

4.4 Low Power Listening

For energy constrained sensor networks, Low Power Listening [25] (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 [23], 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 non-overhearing node will have to wake up and receive the packet being transmitted when channel activities are detected. It is the upper network layer's job to decide whether the packet is addressed to the node itself. In this sense, the energy overhead of overhearing nodes is only caused by the processing of overheard packets compared with non-overhearing nodes, and the radio energy consumption is independent of using overhearing. Because of the above reasons, receiver-initiated approaches such as TALENT will work with LPL in BoX- MAC normally without incurring any significant overhead on the energy usage. We believe that TALENT could still work even if the MAC protocol does not directly support overhearing with LPL as discussed in Section 6.

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

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 [17], STLE [2] and 4C [21]. 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.

5.1 Experimental Setup

We conducted extensive experiments in three wireless testbeds: a local testbed, the Harvard Motelab [33] testbed, and the Indriya [15] 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 conducted extensive single sender experiments in the three testbeds with similar experimental settings used by the STLE authors. Second, 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 -25 dBm to increase the network size, and for the Motelab and Indriya testbeds the power level is set to 0 dBm for better connectivity. We perform single sender experiments with little external interference on channel 26, as well as variable rate experiments with interference from 802.11 radios on channel 11. We run more than 80 experiments in all testbeds combined, with each experiment sending 6,000 packets for a total of 480,000 packets sent.

5.2 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 increase 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 burst traffic pattern to examine the differences in path selection and the end-to-end delivery cost.

Our evaluation employs a simple network consisting of five nodes in a linear topology: a sender *S*, three forwarders

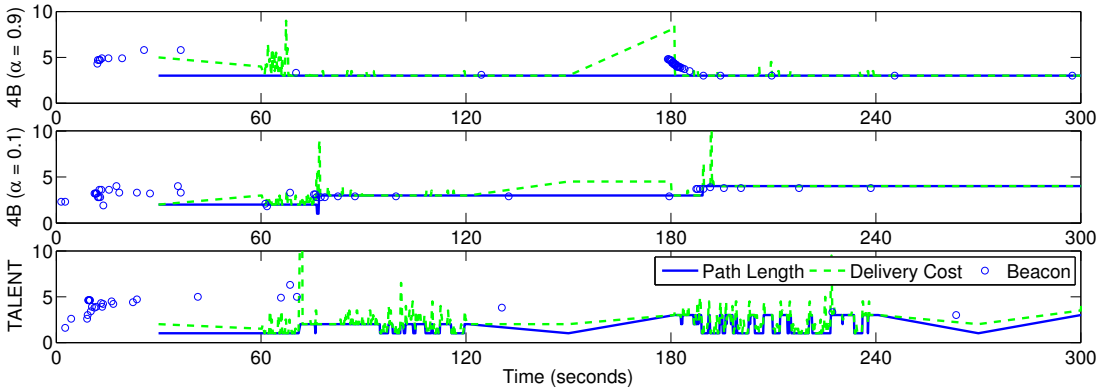


Figure 6. The path length and the delivery cost of 4B (stable and reactive) and TALENT during a 5 minutes experiment.

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

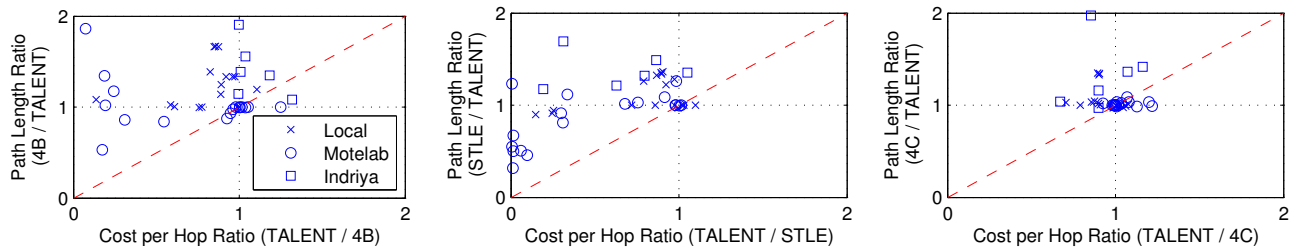
In this network, the sender sends packets in a burst pattern: only one packet is sent every 30 seconds in the first minute, and in the second minute the sender sends 10 packets per second with a 0.1 seconds inter-packet interval. This pattern repeats in the remaining three minutes till the experiment ends at the end of fifth minute. The experiment is first conducted with CTP and stable 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 6 respectively. In each plot, the solid line shows how the path length evolves over the course experiment, the dashed line indicates the corresponding end-to-end delivery cost, and the circles represent the beacons received by the sender S. For each circle, the x axis notes when the beacon is received and the y axis is the estimated delivery cost of the beacon sender.

Judging from the path length showed in the top plot in Figure 6, it is clear that with stable 4B, CTP took the longer path (4 hops) from the beginning and was never stray away from it over the course of the experiment. The merit of this path is that the links are of high quality and stable, thus almost all the send attempts were successful and very little number of retransmissions were required except for a few seconds after 60 seconds. This is reflected by the mostly smooth delivery cost in the plot. Note that due to the stable WMEWMA estimator, the several seconds of high losses are not enough to make the stable 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 6, 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, 34% smaller than stable 4B and 50% smaller than reactive 4B.

Why 4B does not switch to the shorter paths if they are available? The beacon distributions presented in Figure 6 offer an explanation. The beacon packet contains the link quality and delivery cost estimation from the beacon sender, and the recipient node of the beacon can compute the estimated delivery cost assuming the beacon sender as its parent. According to the CTP adaptive beaconing policy, all nodes send beacon packet frequently to establish the initial link quality estimations and select parents at the beginning of the experiment, but the inter-beacon interval grows exponentially as a stable route is established. This can be observed in all the three plots in Figure 6: 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



(a) TALENT vs 4B (b) TALENT vs STLE (c) TALENT vs 4C

Figure 7. 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.

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 for several times, CTP was able to switch back to the shorter path as soon as TALENT indicates a high quality period is available on the shorter path. This observation is based on a simple linear network, but it is also applicable to larger networks. In a dense network, CTP may have more links to choose from and may find alternative routes with small end-to-end costs. However, a dense network also means that the number of potential temporary parents is large, and this allows TALENT to find shorter routes as well. The relative savings of TALENT vs 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, 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.

5.3 Path Length vs. Delivery Cost

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

To study the trade off between a longer path with stable high quality links and a shorter path with unstable intermediate quality in larger networks, we extend our evaluation by comparing the behavior of TALENT with respect to 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. Each experiment was repeated three times with the same network settings, i.e., packet length, radio power level and node configuration in the network. The 4C modeling parameters were assigned based on LR model trained on training sets collected at each testbed, and this 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 7 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 4B in one experiment, and vice-versa.

As shown in Figure 7, the majority of the experiment results were marked above the $x = y$ line, indicating that the overall delivery cost of CTP with TALENT is better than the other estimators. For the local testbed and the Indriya testbed experiments, a large group of the results exhibits small or even negative cost increment while using shorter paths, implying that using TALENT can reduce the path length while maintaining the delivery cost. There are also cases where the path length of TALENT and the other estimators are the same but the cost of TALENT is much smaller (see Figure 7(a) and 7(b)). This is due to a poorly connected network and/or sudden link quality changes that causes 4B and STLE to send excessive retransmissions before switching to another path.

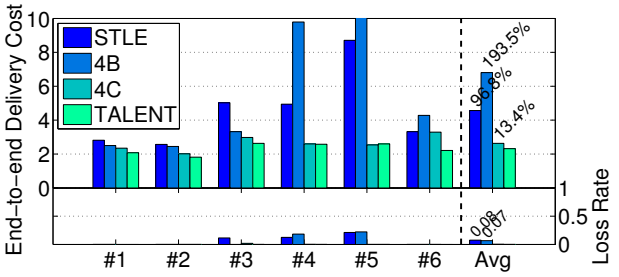


Figure 8. End-to-end delivery cost and loss rate of local testbed. Each bar represents the average results of 3 experiments with the same network settings.

In this case, TALENT enables fast route updates with such network dynamics by taking advantage of the temporary parent mechanism: the overhearing node can notify the sender about the alternative routes even if the ETX estimation in the sender side is lagging behind the link quality changes.

For the Motelab experiments, this behavior is more obvious. Originally, the Motelab testbed had 180 nodes, but the number of nodes has been reduced to 47 at the time we conducted our experiments. Due to the sparsely connected network, the number of possible routes are limited and the use of intermediate links is almost unavoidable in some cases to avoid network partitions. From Figure 7(a) and 7(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 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 re-training that is required in a real setting, is not included in the evaluation.

5.4 End-to-End Delivery Cost and Rate

We further present the performance of TALENT in terms of end- to-end delivery cost and loss rate in this section. Figures 8, 9 and 10 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. Figure 8 has different total number of nodes

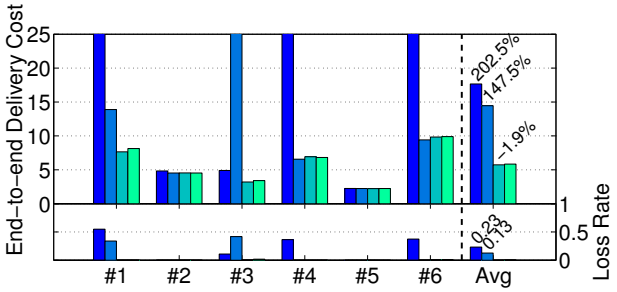


Figure 9. End-to-end delivery cost and loss rate of Motelab testbed experiments. Motelab testbed is sparsely connected, so the number of good paths is limited, which leads to similar cost when the network is stable and heavy cost increments when the path is disturbed.

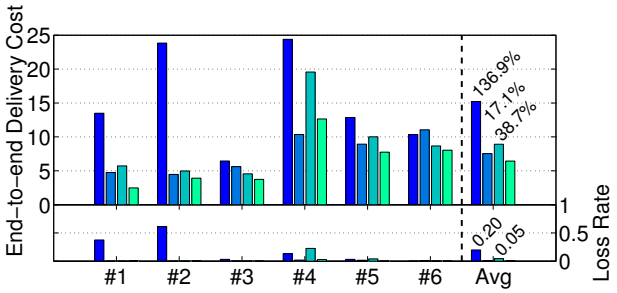


Figure 10. End-to-end delivery cost and loss rate of Indriya testbed.

for each experiment, with 5, 6, 11, 25, 42, and 57 respectively. Figure 9 uses 47 nodes, and Figure 10 uses 127 nodes for all experiments. 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 8 experiments 4 and 5, and from Figure 9 experiments 1 and 3 we see that the delivery cost of 4B is very high. This behavior can be explained by the problems explained in Section 5.2, i.e. slow beaconing activity on alternative paths and slow EWMA convergence. Trace analysis indicates that in these experiments, the number of available forwarding path was limited. If the old forwarding path was broken due to link failure, the sender could not find an alternative parent and had to wait for the beacon from neighboring nodes to update the link quality. This leads to many retransmissions and eventually packet losses, whereas TALENT maintains a connected network due to the receiver-initiated approach. For example, in Figure 8 Exp #5, the 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 achieved near 0% loss rate because the overhearing nodes can become temporary parents as soon as the sender's old parent is no longer reachable.

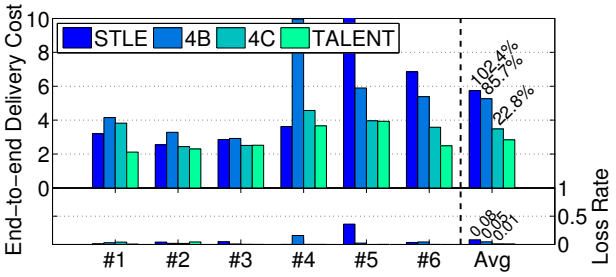


Figure 11. End-to-end delivery cost and loss rate of variable sending rate and single sender.

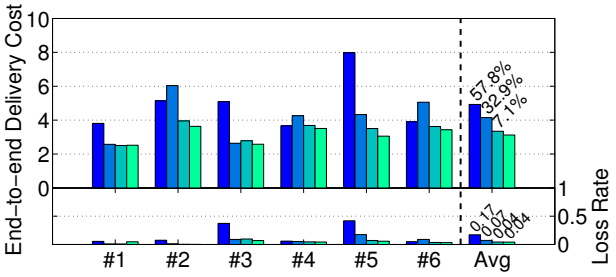


Figure 12. End-to-end delivery cost and loss rate of variable sending rate and multiple senders.

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

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

5.5 Variable Rate and Multiple Senders

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

The variable rate experiments used the same network settings in local single sender experiments, the only difference is the inter-packet interval is randomly selected in $[50, 150]$ ms range instead of using a fixed interval of 100 ms. As shown in Figure 11, the end-to-end the delivery cost and loss rate results are similar to the fixed interval experiments results observed in Figure 8, indicating the variable rate does not affect the performance of TALENT significantly.

To test the performance of TALENT in congested networks, we also conducted several experiments with multiple nodes sending simultaneously with packet interval ranging from 50 to 100 ms. Please note that although having multiple senders is common in WSNs, it is rare that these senders send packets at a high data rate at the same time. We consider this experiment setting as the worst case scenario where the network is congested and affected by external interference. Figure 12 shows that while TALENT still outperforms STLE and 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 created more data forwarding paths compared with single sender experiments, and hence helped the 4B link estimator to evaluate more links with higher rate. We leave further exploration of using TALENT in highly congested networks for future work.

6 Discussion

Integration with other MAC Protocols: LPL used in this work is realized in the BoX-MAC [23], the default MAC protocol in TinyOS 2. As discussed in Section 4.4, 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 [8] perform address checks before starting to receive the data packets, which limits the use of overhearing operation. In this case, using overhearing may negatively impact the performance of LPL. However, as pointed out by Moss and Levis [23], 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. We leave this evaluation for future work.

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

This limitation can be resolved 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 long value, and burst mode, in which TALENT is enabled and LPL incorporates short wakeup intervals. By doing local buffering and sending packets in bursts, applications allow TALENT to select the instantaneous low cost paths, trading off increased latency for significantly larger delivery efficiency and smaller delivery costs. We leave the design of such protocol to future work.

7 Conclusion

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

8 Acknowledgment

Special thanks to Jimei Yang and Nic Schraudolph for valuable discussions on online learning, the anonymous reviewers for their insightful feedback, and Jakob Eriksson for shepherding this paper. This material is based upon work partially supported by the National Science Foundation under grant #0923586, and the Center for Information Technology Research in the Interest of Society under grant #07427.

9 References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. *SIGCOMM Comput. Commun. Rev.*, 34(4):121–132, 2004.

[2] M. H. Alizai, O. Landsiedel, J. Ágila Bitsch Link, S. Götz, and K. Wehrle. Bursty traffic over bursty links. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pages 71–84. ACM, 2009.

[3] L. B. Almeida, T. Langlois, J. D. Amaral, and A. Plakhov. *Parameter adaptation in stochastic optimization*, pages 111–134. Cambridge University Press, New York, NY, USA, 1998.

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

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

[6] L. Bottou and Y. LeCun. Large scale online learning. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[7] L. Bottou and N. Murata. Stochastic approximations and efficient learning. In *The Handbook of Brain Theory and Neural Networks, Second edition,*. The MIT Press, Cambridge, MA, 2002.

[8] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *4th ACM Conference on Embedded Networked Sensor (SenSys '06)*, pages 307–320. ACM, 2006.

[9] A. Cerpa and D. Estrin. SCALE: A tool for simple connectivity assessment in lossy environments. Technical Report 0021, UCLA, 2003.

[10] A. Cerpa, J. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pages 81–88. ACM, 2005.

[11] A. Cerpa, J. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: Modeling and implications on multi-hop routing. In *6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)*, pages 414–425. ACM, 2005.

[12] ChipCon CC2420. http://www.ti.com/product/cc2420.

[13] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *9th International Conference on Mobile Computing and Networking (MobiCom '03)*, pages 134–146. ACM, 2003.

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

[15] M. Doddavenkatappa, M. C. Chan, and A. A. L. Indriya: A low-cost, 3D wireless sensor network testbed. In *7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM '11)*, 2011.

[16] K. Farkas, T. Hossmann, F. Legendre, B. Plattner, and S. K. Das. Link quality prediction in mesh networks. *Comput. Commun.*, 31(8):1497–1512, 2008.

[17] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four-bit wireless link estimation. In *6th Workshop on Hot Topics in Networks (HotNets VI)*. ACM, 2007.

[18] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pages 1–14. ACM, 2009.

[19] D. Lai, A. Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu, and A. Keshavarzian. Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In *IEEE Global Telecommunications Conference, 2003*, pages 446–452. IEEE.

[20] Y. Le Cun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag, 1998.

[21] T. Liu and A. Cerpa. Foresee (4C): Wireless link prediction using link features. In *10th International Symposium on Information Processing in Sensor Networks (IPSN '11)*, pages 294–305, Apr. 2011.

[22] T. M. Mitchell. *Machine Learning*. McGraw Hill Higher Ed., 1997.

[23] D. Moss and P. Levis. BoX-MACs: Exploiting physical and link layer boundaries in low-power networking. Technical Report SING-08-00, Stanford University, 2008.

[24] MultihopLQI. http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI.

[25] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *2nd ACM Conference on Embedded Networked Sensor (SenSys '04)*, pages 95–107. ACM, 2004.

[26] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, and J. Anderson. Analysis of wireless sensor networks for habitat monitoring. In *Wireless Sensor Networks*, pages 399–423. Springer US, 2004.

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

[28] M. Senel, K. Chintalapudi, D. Lal, A. Keshavarzian, and E. J. Coyle. A kalman filter based link quality estimation scheme for wireless sensor networks. In *IEEE Global Communications Conference, 2007*, pages 875–880. IEEE, 2007.

[29] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *4th ACM Conference on Embedded Networked Sensor Systems (SenSys '06)*, pages 237–250. ACM, 2006.

[30] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. The β-factor: measuring wireless link burstiness. In *6th ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*, pages 29–42. ACM, 2008.

[31] TinyOS 2. http://www.tinyos.net, accessed feb. 2012.

[32] G. Werner, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*, pages 381–396.

[33] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. In *4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, page 68. IEEE, 2005.

[34] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *1st ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 14–27. ACM, 2003.

[35] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *2nd ACM Conference on Embedded Networked Sensor Systems (SenSys '04)*, pages 13–24. ACM, 2004.

[36] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *1st ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 1–13. ACM, 2003.

[37] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *1st IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON '04)*, pages 517–526.