A Link Availability Predictor for Wireless Sensor Networks

Haruki Oh
Stanford University
hoh@cs.stanford.edu

ABSTRACT

We present a machine learning based approach to link availability prediction to be used in low power wireless sensor network routing. Accurately predicting availability of intermediate quality links will enable cost-efficient routing without sacrificing reliability and stability. Our predictor performs just slightly better than a simple predictor.

1. INTRODUCTION

In wireless sensor networks, there are unstable links that offer better routing progress than stable, high-quality links. However, most link quality estimation and routing techniques have focused on identifying and utilizing consistently high quality links for packet forwarding. Typical routing protocols for wireless sensor networks, such as Collection Tree Protocol (CTP)[5], select links as suggested by their link estimator, which chooses only high quality links to achieve better connectivity and reliable communication[4]. However, this approach ignores links with lower connectivity, but it might reach further into the network, potentially more closer to the end node. Moreover, in a sparse network with low density of nodes, there may be no high quality link available, and nodes must deal with unstable connectivity.

Studies have shown that these intermediate quality links are highly bursty in their connectivity, meaning that the connectivity frequently change between stable and unstable periods. Accurately predicting the periods when the link is stable enables wireless nodes to utilize the link without affecting the reliability and stability of the network, and achieves more efficient routing to the end node.

Previous work have focused on estimating the link quality using some metrics. However, none focuses on predicting explicitly whether the link is stable or unstable at a given time. The main contributions of this work are (1) it predicts whether the link is stable or not, thus not requiring any change in the routing protocol, and (2) rather than defining a single metric, it uses machine learning techniques to identify patterns in successful transmissions.

The remainder of the paper is organized as follows: Section 2 discusses related work in taking advantages of intermediate quality links. Section 3 explains the design of this machine learning based system and how the features are selected and generated. Section 4 discusses the evaluation of the system, and Section 5 concludes the discussion and outlines future work.

2. BACKGROUND AND RELATED WORK

This section describes some existing ways of taking advantages of the links with intermediate quality.

2.1 Statistics of Packet Loss

While the majority of link estimators assume that individual packet delivery and loss events are independent, and that they follow Bernoulli distribution [11], studies have shown that for shorter time scales, group of packet delivery may be correlated [3, 9]. Furthermore, in the work by Becher et al. [1], the authors concluded that any link, regardless of quality, becomes temporarily reliable after 3-5 consecutive packets are received over that link. These results suggest that statistical learning techniques could be used to predict when the link becomes reliable.

2.2 Opportune Transmission

Srinivasan et al.[10] introduced the metric, β-factor that characterizes the bustiness of a link using Conditional Packet Delivery Function that describes the probability of packet delivery after n consecutive deliveries. Using β-factor, the authors introduced opportune transmission where sending node will pause for a period of time after a failure. This will increase the apparent packet reception rate, reducing the transmission cost at the expense of throughput and latency.

2.3 Opportunistic Routing

Opportunistic routing[2] uses coordination among intermediate quality links to increase throughput in sparse network where links that available are all intermediate quality. Because it requires sending packets in batch and coordination among intermediate nodes, it has a relatively high overhead cost and communication. Our machine learning based link availability predictor is designed so that sender node can choose an available and stable link to send, and will not require overhead coordination among nodes.

3. DESIGN OVERVIEW

The main task of LAP is to predict whether the next packet will be successfully transmitted to the destination or not. Since we do not know about the packet, such as signal strength, until the node actually sends out, we must use previous history to predict whether the next transmission will be successful. In this section we describe how to generate training data (and test data) from packet traces.

3.1 Packet Traces

The packet trace obtained from Stanford Information Networks Group contains two files. One file contains sequence number, sender mote ID, destination mote ID, Received Signal Strength Indicator (RSSI), Chip Correlation Index (CCI), and timestamp of each successful transmissions. The other file contains sequence number (which is unrelated to the one in the other file), mote ID, noise observed, and timestamp. Because noise is an interesting feature in predicting packet delivery, we first merge the two file so that the first file contains, in addition to packet delivery trace, the noise observed.

Because there is no one-to-one correspondence in timestamps of the two files, there may not be an observation of noise level when the packet was sent. To fill in the missing noise observations, we used locally weighted linear regression to estimate the noise at that time. Since noise level observation was done frequently (as frequent as packet transmission), we believe this approach gives very good estimate of the noise level when the packet was sent.

3.2 Estimating RSSI of Lost Packets

Because packet trace contains only records of successful transmissions (it does not contain any information about lost packets), we cannot run supervised learning directly. Initially we tried using density estimation with mixture of Gaussians. However this approach resulted in extremely poor accuracy even after numerous attempts in finding the best number of Gaussians as well as the best threshold for labeling.

In order to use supervised learning techniques, we must have information about lost packets. Because wireless nodes tried to send packets at a constant frequency, we already know what time the transmission of the lost packet was attempted, as well as the noise level at that time. Rusak et al. [8] introduced an accurate way of guessing the signal power of lost packets. Using a function that maps between signal-to-noise ratio and packet loss rate[7], we can calculate the distribution of lost packets at a certain signal strength.

During pre-processing, using data of packets that were delivered, we create a distribution of signal strength of the packets that were lost. For each packet that was lost, we use the time it was (supposed to be) attempted to estimate the signal strength and the RSSI. We have the following relation between signal strength and RSSI:

$$signal\_strength = 10\log_{10}(10^{\frac{RSSI}{10}} + p \times 10^{\frac{Noise}{10}})$$

The parameter P is the phase difference between the signal and the noise. (We found that the signal and noise are most likely out-of-phase.) After guessing the signal strength from probabilistic distribution, we use the same formula to calculate the RSSI.

3.3 Feature Selection and Generation

With data for lost packets, we can generate the features for supervised learning training. Because we are looking at history to predict the future, we have a parameter N, to be determined during training, that describes how long into the past we consider. Since previous work[10, 1] used histories of packet delivery, packet delivery history on that link, which is an array of booleans, are included in the feature set. Because some links are always better than others, (some links have near 100% connectivity; some have near 0%), mote ID describing the destination node was also chosen as a feature. Clearly signal strength and noise level affects packet reception, so they are included in the feature set as well.

In addition to the above, change in the above parameters might help. For example, if the noise level was clearly rising, it might indicate that the packet reception rate is dropping and the next packet might be lost. To this extent, we also chose to include the change in reception rate, signal strength, and noise level, as the slope of the linear regression.

We have also tried the CCI(Chip Correlation Indicator, which is related to the frequency of data corruption) and its history but this only lowered the accuracy. The β-factor[10] did not change the accuracy so it is not included in the final feature set. The features for other links significantly increased the training time, and due to time constraint, it is not included.

For training and classification, we used SVMLight[6] with Gaussian Kernel.

4. EVALUATION

For evaluation, we used packet traces from Intel Mirage/MicaZ testbed, on Channel 11. In this testbed, each node sends a unicast packet to another node in a round-robin fashion, every 15 seconds. A sequence number is generated for each attempt to send a packet to a particular node. In total, each node attempted to send 800 packets to another node. We chose the first 700 as the training example and the last 100 as the test set. Since there is time-correlation involved, we cannot just randomly split the dataset.

The parameter N, describing how much past data to look, was determined to be 5. N=10 did not improve the performance, and N=3 or 15 started to decrease the accuracy.

4.1 Simple Predictor

Since we did not have access to any real wireless sensornet testbed, we built a "simple" predictor to compare the performance against. A "simple" predictor will always label "yes" if packet reception rate (PRR) of that link is greater than 50%, and always label "no" otherwise. This gives the theoretical minimum accuracy of 50% for the simple predictor.

4.2 Results

Table 1 lists the results for sending packets from node 9, results from other nodes were similar. Some nodes, such as node 0, was excluded from the table because it did not exist in the network. PRR is the packet reception rate at that node; Simple is the prediction accuracy for the simple predictor; and SVM is the prediction accuracy for SVM predictor.

Overall SVM did about equally well as a simple predictor, doing slightly better in the intermediate quality nodes. (which are actually the important ones to make predictions for). SVM did not perform well when the PRR is close to 100%.

Table 1: Results for Node 9

Destination	PRR	Simple	SVM
2	0.97	0.97	0.95
3	0.2	0.8	0.81
4	0.97	0.97	0.95
5	0.97	0.97	0.95
6	0.97	0.97	0.94
7	0.97	0.97	0.95
10	0.96	0.96	0.95
11	0.87	0.87	0.87
12	0.73	0.73	0.74
13	0.99	0.99	0.96
15	0.27	0.73	0.75
16	0.9	0.9	0.89
17	1	1	0.96
18	0.47	0.53	0.59
19	0.01	0.99	0.91
20	0.84	0.84	0.84
21	0.97	0.97	0.94
24	0.8	0.8	0.8
30	0.38	0.62	0.64

This result suggest a two different predictor could be used together. Since for high quality links, a "simple" predictor will do well, and for intermediate quality links, SVM seems to do slightly better most of the time.

5. CONCLUSIONS AND FUTURE WORK

We have demonstrated that link availability prediction using machine learning techniques is reasonably accurate, with rooms for improvement. Data from other links may be a good feature to try as there was not enough time to run such a long training run multiple times.

Since we were only able to test accuracy of predicting whether the link is stable or not, it is essential to measure the end-end path effect and performance of this predictor on a real wireless network testbed.

6. ACKNOWLEDGMENTS

I would like to thank Prof. Philip Levis and Kannan Srinivasan for their advice on this project.

7. REFERENCES

[1] A. Becher, O. Landsiedal, G. Kunz, and K. Wehrle. Towards short-term link quality estimation. *Hot Emnets 2008*.

[2] S. Biswas and R. Morris. Exor: Opportunistic multi-hop routing for wireless networks. *SIGCOMM 2005*.

[3] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. *MobiHoc'05*.

[4] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four bit wireless link estimation. *HotNets VI (2007)*.

[5] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. *SenSys'09*, 2009.

[6] T. Joachims. Making large-scale svm learning practical. In B. SchŽlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[7] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. *IPSN 2007*.

[8] T. Rusak and P. Levis. Investigating a physically-based signal power model for robust wireless link simulation. *MSWiM (2008)*.

[9] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. Understanding the causes of packet delivery success and failure in dense wireless sensor networks. *SenSys'06*.

[10] K. Srinivasan, M. Kazandijeva, S. Agarwal, and P. Levis. The β-factor: Measuring wireless link burstiness. *SenSys'08*.

[11] A. Woo and D. Culler. Evaluation of efficient link reliability estimators for low-power wireless networks. Technical Report UCB/CSD-03-1270, UC Berkeley, 2003.