Reliable, Low Overhead Link Quality Estimation for 802.11 Wireless Mesh Networks

Lochan Verma, Seongkwan Kim, Sunghyun Choi
School of Electrical Engineering & INMC
Seoul National University
Seoul, 151-744, Korea
{lochan,skim}@mwnl.snu.ac.kr, schoi@snu.ac.kr

Sung-Ju Lee
Media Communications & Networking Lab
Hewlett-Packard Laboratories
Palo Alto, CA 94304
sjlee@hp.com

***Abstract*—We propose *QUEST* (QUality ESTimation), a new method that accurately estimates IEEE 802.11 wireless link quality with no in-band signaling overhead. Existing link quality estimation methods either are based on hello exchanges by fixing or varying transmission rates or rely on the history (e.g., delivery ratio) of previously sent data packets in a per-rate/-neighbor manner. QUEST on the other hand, is based on a delivery ratio vs. SNR (Signal to Noise Ratio) relation, called *profile*, that is managed offline. QUEST estimates the target link quality in terms of delivery ratio by performing profile lookup for any incoming messages including broadcast hello, beacon, data packets, etc. Therefore, it does not depend on a designated protocol to obtain the delivery ratio. Instead, in QUEST, the per-rate/-neighbor management of link quality is achieved by profile lookup. We perform testbed experiments to achieve the profile and also unravel two major bugs in MadWifi driver, widely employed by many researchers to build an 802.11-based system. Utilizing the large database of transmitter and receiver traces with an indigenously developed tool, we study the impact of altering the averaging time period on the profile for different transmission rates.**

I. Introduction

Wireless mesh networks aim to provide reliable high throughput network connectivity to wireless users. Recent research has focused on increasing the end-to-end throughput performance of wireless mesh backhauls, where routing is one of the most important factors in achieving higher performance in multi-hop wireless links.

An effective route establishment heavily relies on an elaborate design of link metric that represents wireless link quality. Therefore, the precise knowledge of link quality to calculate the routing metric of interest has become a prime aspect in wireless mesh networking. Link quality aware metrics proposed in [1], [2] have been shown to perform better than the traditional hop count-based metric in the wireless mesh settings [3]. Such mesh routing metrics can be derived using the measured packet delivery ratio on a target wireless link.

Two popular assessment methods in obtaining the delivery ratio are *broadcast* and *unicast* probing. Both of these schemes have there own pros and cons. The latter scores high on accuracy, while the former is preferred for the low overhead. Considering a scalable mesh network, the overhead incurred in link quality measurements can tarnish the performance of the network.

This work was supported in part by Seoul R&BD Program (10544).

We propose a new link quality assessment method, called *QUEST* (QUality ESTimation) in IEEE 802.11-based mesh networks. QUEST targets to estimate the delivery ratio on a mesh link without having any in-band signaling overhead, such as periodic exchange of hello messages. Moreover, unlike existing link quality estimation methods that either are based on hello exchanges by fixing or varying transmission rates or rely on the history (e.g., delivery ratio) of previously sent data packets in a per-rate/-neighbor manner, QUEST is based on a delivery ratio vs. SNR (Signal to Noise Ratio) mapping table called *profile* that can be managed offline. QUEST estimates the quality of a target link as the delivery ratio by performing profile lookup for any incoming messages including broadcast hello, beacon, data packets, etc. Therefore, no designated protocol to achieve the delivery ratio information, with the form of (delivery ratio, target neighbor, transmission rate)-tuple, is required in QUEST.

The usage of channel profile has been explored in [4]–[6] for interference estimation and link throughput prediction. They consider only the lowest rate. In CHARM (Channel-aware Rate Adaptation) [7] , the nodes estimate the delivery ratio of a packet across different rates using the exchanged data packets. However, it requires introducing a new element in beacon, probe request and probe response structures.

QUEST is a generic method, being applicable to any 802.11-based networks with wireless LAN/ad-hoc/mesh settings. We have built and evaluated QUEST in an open source 802.11 driver, MadWifi [8]. The implementations in [4]–[6] use earlier MadWifi releases with a fixed noise floor as -95 dBm whereas QUEST leverages the noise information exposed by the MadWifi (version 0.9.3.3). Although the usage of QUEST is independent of underlying PHY (Physical layer) technologies (802.11a/b/g), QUEST has been implemented on the 802.11a PHY as we found that there are two challenging issues in building QUEST in the driver. Moreover, considering the current trend that 802.11b/g PHYs are mainly used for client access, we believe that the use of 802.11a PHY as a backhaul channel could be a rational choice.

The contribution of this paper consists of the followings:

1) Building a wireless link profile: we design and evaluate a low overhead, high accuracy link quality estimation method based on a predetermined delivery ratio table, called profile. Such an offline table-based approach has not been developed previously for the wireless mesh network.

2) Accuracy study of the profile: through our indigenously developed tool, we study the effect of altering the averaging time period on profile for different transmission rates. Based on this study, the accuracy of profile can be evaluated.

3) Debugging in MadWifi: we expose some unexplored issues in MadWifi driver [8], and present our own solutions to tackle such bugs. In fact, the driver problems significantly abate the performance of MadWifi-based networks.

The rest of the paper is organized as follows. In Section II, we propose QUEST. The experimental setup information for profile generation is presented in Section III. Section IV discusses the challenges in the profile generation. The paper concludes with Section V.

II. QUEST

We present the details of QUEST, our link quality estimation method for wireless mesh networks. We describe the motivation of QUEST design by reviewing the existing methods and present the operation of QUEST.

A. Overhead for Link Quality Estimation

Most routing metrics in wireless mesh networks are based on measured packet delivery ratio. We briefly review two such metrics, namely, ETX (Expected Transmission Count) [1] and ETT (Expected Transmission Time) [2]. ETX is the expected number of data transmissions required to successfully send a packet over a link, including retransmissions. The derivation of ETX requires the measurement of forward and reverse packet delivery ratios (i.e., d_f and d_r) on a link, and it is given by

$$\mathrm{ETX} = \frac{1}{d_f \times d_r}.$$

ETT, which is designed to reflect multiple transmission rate capabilities onto the routing metric, is based on ETX as follows.

$$\mathrm{ETT} = \mathrm{ETX} \times \frac{S}{R},$$

where S denotes the size of the data packet and R is the raw data transmission rate of the link. In other words, the accuracy of ETT should also rely on the measured delivery ratio.

Two popular assessment methods to obtain the delivery ratio are *broadcast* and *unicast link quality probings*. $broadcast_{FR}$ (broadcast with Fixed Rate) measures the delivery ratio using a fixed, lowest transmission rate. Therefore, it does not need to send a probe packet, which is typically a *hello* message, to all one-hop neighbors with different transmission rates. However, considering the multiple transmission rates feature in the current 802.11 devices for data communications, relying on only the lowest transmission rate for the link quality measurement is undesirable. $broadcast_{MR}$ (broadcast with Multiple Rates), an improved form of $broadcast_{FR}$, performs periodic hello message broadcast at different transmission rates. On the other hand, unicast method scores high on accuracy as it utilizes all the available transmission rates for the delivery ratio measurement. Considering a large mesh network, however, the overhead incurred in quality measurement can degrade the performance of the network due to its per-neighbor/-rate probings.

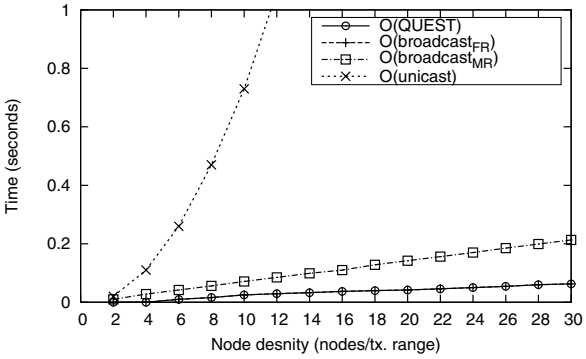


Fig. 1. Time overhead with different methods for link quality estimation and dissemination to neighbors.

We analyze the overhead of the existing delivery ratio measurement methods. Eqs. (1)–(3) give the time duration in microseconds to estimate the delivery ratio with respect to a node:

$$O(broadcast_{FR}) = \left[T_{difs} + \overline{T_b} + \frac{\ell_{data}}{r^1}\right] |\mathbb{N}|, \quad (1)$$

$$O(broadcast_{MR}) = \left[(T_{difs} + \overline{T_b})\,|\mathbb{R}| + \sum_{i=1}^{|\mathbb{R}|} \frac{\ell_{data}}{r^i}\right] |\mathbb{N}|, \quad (2)$$

$$O(unicast) = \left[(T_{difs} + \overline{T_b} + T_{sifs} + \tfrac{\ell_{ack}}{r^1})\,|\mathbb{R}| + \textstyle\sum_{i=1}^{|\mathbb{R}|} \frac{\ell_{data}}{r^i}\right] |\mathbb{N}|^2, \quad (3)$$

where T_{difs}, T_{sifs}, and $\overline{T_b}$ represent DIFS (Distributed Inter Frame Space), SIFS (Short Inter Frame Space), and the average backoff time, respectively. $\overline{T_b} = \frac{CWmin}{2} \cdot SlotTime$, where $CWmin$ and $SlotTime$ represent the minimum contention window and the backoff slot time, respectively. ℓ_{data} and ℓ_{ack} denote the length of a data and an ACK frame, respectively, which are fixed at 12000 and 112 bits in our analysis. $\mathbb{N}$ and $\mathbb{R}$ are the sets of neighbor nodes in the vicinity of the target node and transmission rates, respectively, while $|\cdot|$ represents the cardinality of the set. r^i represents the transmission rate with index i. For example, r^1 is the lowest transmission rate, e.g., 6 Mbps in the case of the 802.11a PHY. Note that the lowest transmission rate, i.e., 6 Mbps, is used for ACK transmission in the above analysis.

The overhead for $broadcast_{FR}$ is the minimum since periodic hello message exchanges at only the lowest rate are performed. $broadcast_{MR}$ performs periodic hello message broadcasts at different transmission rates, and thus has an

inflated overhead cost. As seen in Fig. 1, unicast packet probing method incurs the maximum overhead due to per-neighbor/-rate probing.

QUEST requires no in-band signaling, and estimates the delivery ratio by performing profile lookup for any received management, control or data packet. Ideally, it should have *zero* time duration overhead. However, in a wireless mesh network, a node needs to disseminate the link quality estimate for all links with other nodes in its transmission range, in order to assist the routing protocols in route establishment decisions. Accordingly, QUEST has the same overhead as broadcast probing, but it provides the link quality estimate for all transmission rates as compared with the broadcast probing, which provides the link estimate for only the lowest rate.

B. QUEST Algorithm

Algorithm 1 QUEST algorithm.

1: $RX_{SNR} \Leftarrow$ SNR of the received frame
2: $dr^i \Leftarrow$ Delivery ratio for rate i
3: **if** frame received **then**
4: do profile lookup with RX_{SNR} to attain dr^i for all i
5: **end if**

Algorithm 1 estimates the link quality without incurring any in-band signaling overhead. Profile, which is calculated offline, forms the backbone of QUEST, and each node in the mesh network is equipped with it. QUEST estimates the target link quality (i.e., the delivery ratio) by performing the profile look up for any incoming packet.

III. Experimental Setup for Profile Generation

Profile refers to a delivery ratio vs. SNR mapping table for all transmission rates developed through experimentation. We focus on 802.11a PHY module and materialize profile through elaborate calibrations using two NICs (Network Interface Cards) for different transmission rates in the 802.11a rate set. In fact, the profile is dependent on vendor and chipset. We use Atheros/MadWifi NIC/driver pair as MadWifi is an open source driver available for Atheros chipsets. We use two IBM X40 laptops running Ubuntu linux (kernel version 2.6) [9], equipped with Cisco aironet 350 802.11 a/b/g PCMCIA cards [10], which are based on Atheros chipset.

A. Driver

The employed card driver is MadWifi (version 0.9.3.3). We choose this version because it is the latest when this paper is written, and it provides live noise floor calibrations as opposed to a constant value of -95 dBm in older versions. In order to support the calibration of currently prevailing noise level on a given channel, this version uses a function, *ath_hal_process_noisefloor* unused in earlier releases of MadWifi. Atheros NICs measure RSSI (Received Signal Strength Indicator) as a gain over noise floor in dB. Thus, the reported RSSI is equivalent to SNR (Signal to Noise Ratio). All experiments are carried for 802.11a, operating in the ad hoc mode at channel 157 with the center frequency 5.785 GHz. Since we have only two nodes, and this is a non-used channel at our experimentation site, interference from other stations is assumed to be absent.

TABLE I
Total number of packets received during entire experimentation with broadcast packets for different transmission rates

Transmission Rate	Total Packets
6 Mbps	106,876
12 Mbps	74,759
18 Mbps	94,337
24 Mbps	81,430
36 Mbps	90,145
48 Mbps	88,091
54 Mbps	87,097

During our calibrations, the noise floor actually varied between -88 dBm and -93 dBm. We do not develop a profile for the transmission rate of 9 Mbps as it is well known to have a similar performance as 12 Mbps [11].

At the receiver node, there are two possible causes of reception errors:

1. An error occurs in the PLCP (Physical Layer Convergence Procedure) header. This is a PHY error and occurs when the PLCP preamble synchronization or PLCP header reception is unsuccessful. The driver does not report statistics such as RSSI in this case.
2. The CRC-32 (Cyclic Redundancy Check) in the FCS (Frame Check Sequence) of the MPDU (MAC Protocol Data Unit) fails. This is a MAC error and packets affected by this are dropped by NIC.

Only the second case can be detected by the receiver, as in the first case no information is reported.

We customize the driver to gather statistics from packets received correctly or with MAC error. For each received packet, we maintain the attributes including RSSI, timestamp, and sequence number. At the transmitter, the driver registers such information as retry count, timestamp, and sequence number for each transmitted packet. Table I is a reflection of the magnitude of our database from our receiver trace.

B. Traffic Generator Tool

We use Iperf (version 1.7) [12] network traffic generator tool. In order to end an active traffic session, Iperf inherently generates a handshake signal between the transmitter and the receiver. This is done to close the sockets opened for traffic stream transfer. The transmitter will retry 10 times to establish handshake in the case of a handshake establishment failure. This operation causes some unwanted packet transfer between the nodes and the time duration for active traffic session becomes unpredictable.

Iperf is modified to meet our needs. We use UDP packet size of 1001 bytes and generate CBR traffic at 1 Mbps. Odd packet size of 1001 bytes is employed to avoid spurious packets from causing ambiguity. Our database of transmitter and receiver traces maintains the timestamp for each packet. When we

experimented with a higher traffic rate, astonishingly identical timestamp values were recorded for many successive transmissions. The reason is related to the granularity of the reported timestamp. With MadWifi, the granularity of timestamp is $1\ ms$ and $1\ \mu s$ at the transmitter and the receiver, respectively. Therefore, we adopted the traffic rate of 1 Mbps to avoid identical timestamp reports. Our experimentation has been done using both broadcast and unicast packet transmissions. Each experiment run lasts for 200 seconds, and unless mentioned otherwise all calibrations are based on 5 experimental runs, i.e., around 17 minutes of experimentation for each transmission rate. Experimentation is carried out in evenings with multiple receiver node locations and transmission power control at the transmitter node.

C. Post Processing Tool

We developed the post processing tool to correlate the large database of the receiver and the transmitter traces. The major motivation to develop this tool is to determine the size of time window, called *Swin* (Sampling window), over which the delivery ratio and the corresponding average SNR should be calculated. To understand the operation of this tool, X packets are transmitted within an Swin interval. X is determined using the transmitter trace and the knowledge of traffic generation rate. Let Y be the total number of packets registered in the database. The delivery ratio and the average SNR are then computed for the Swin interval using Eqs. (4) and (5), and Y/X (delivery ratio, average SNR) pairs are obtained:

$$\text{Delivery Ratio} = \frac{rx\_no\_fault}{tx\_total}, \tag{4}$$

and

$$\text{Average SNR} = \frac{\sum_{i=1}^{rx\_total} SNR(i)}{rx\_total}, \tag{5}$$

where the following parameters are obtained for the given Swin interval in consideration.

- $SNR(i)$: the SNR measured during the i-th frame reception;
- rx_no_fault: the total number of correctly received frames;
- rx_mac_fault: the total number of frames received with MAC error;
- tx_total: the total number of transmitted MPDUs;
- rx_total $(= rx\_no\_fault + rx\_mac\_fault)$: the total number of received MPDUs.

Post processing tool provides us flexibility to generate different profiles from the same database of the receiver and the transmitter traces by adjusting the Swin size. As another benefit, this tool determines PHY errors by tracking the missing sequence number in the receiver trace and correlating it with the transmitter trace.

When using unicast packets, the measured delivery ratio is representative of the value at MAC SAP (Service Access Point). However, if broadcast packets are used, it can be associated with any layer of the ISO model since there are no retransmissions.

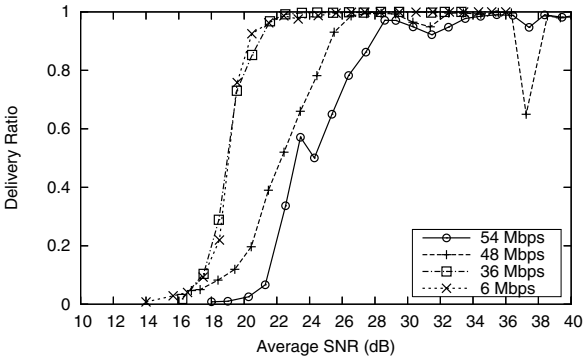


Fig. 2. Profile using 1 dB bucket along x-axis for different transmission rates obtained with unicast packets for Swin 0.1 second.

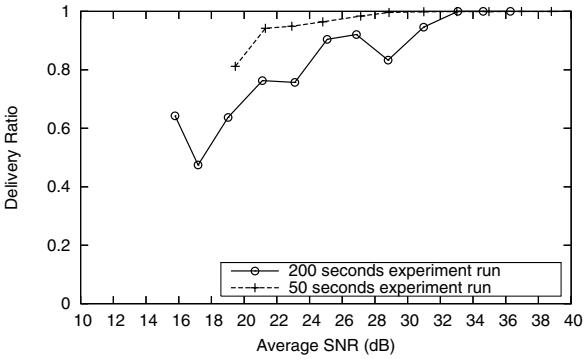


Fig. 3. Profile for 6 Mbps using 1 dB bucket along x-axis obtained with broadcast packets varying experiment time keeping Swin 0.1 second.

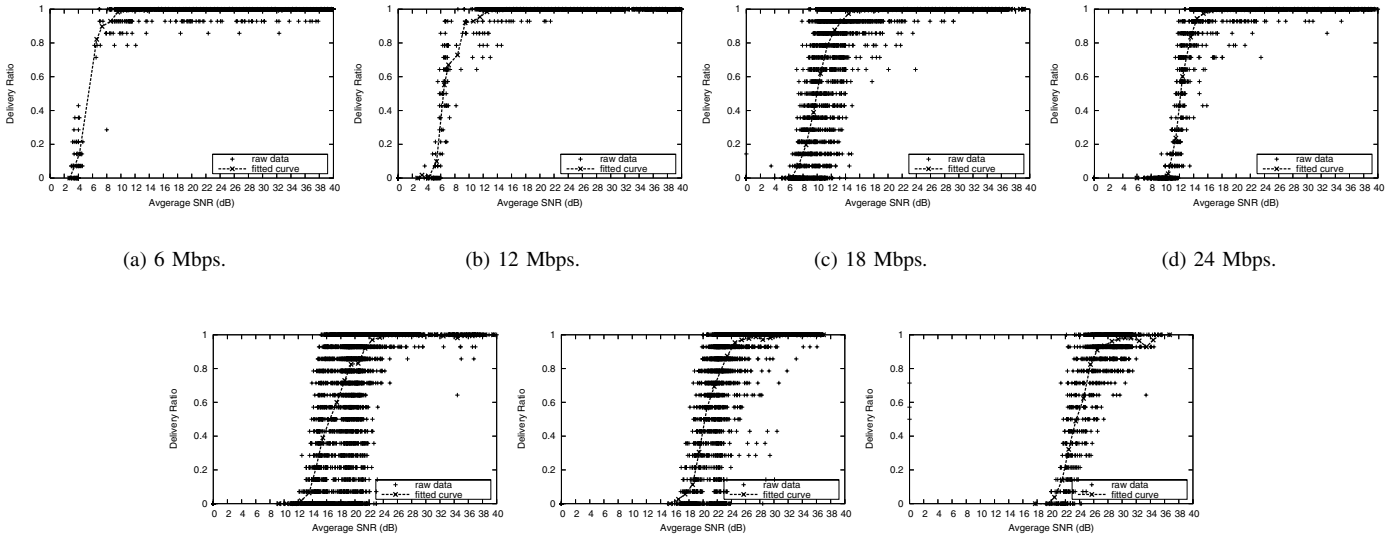
IV. Challenges for Profile Generation

In Fig. 2, the curves are obtained via linear interpolation using 1 dB bucket along the x-axis. The results are a surprise to us because the delivery ratio curve for 6 Mbps is nearly equivalent to that for 36 Mbps. This result was highly unreasonable and intrigued us to retry the entire experimentation by employing broadcast packets.

Fig. 3 depicts the delivery ratio vs. SNR relation obtained for 6 Mbps using broadcast packets. 6 Mbps is a robust rate and is expected to have high delivery ratio even for low SNR range. However, with varying experimentation time duration, unsatisfactory curves are materialized. Interestingly, smaller experimentation duration generated similar delivery ratio vs. SNR curve to that seen in Fig. 2 for 6 Mbps. With increased experiment run time, the curve shows unpredictable trend.

Fig. 3 proved that the problem did not lay in using either broadcast or unicast packets for making the profile. A close evaluation of Figs. 2 and 3 reveals that:

- Delivery ratio is abnormally reduced for 6 Mbps in both cases when using unicast and broadcast packets for measurement.
- The receive sensitivity being followed is not in accordance with the Cisco specifications as shown in Table II.



(a) 6 Mbps. (b) 12 Mbps. (c) 18 Mbps. (d) 24 Mbps.

(e) 36 Mbps. (f) 48 Mbps. (g) 54 Mbps.

Fig. 4. Scatter plot and fitted curves using 1 dB bucket along x-axis with broadcast packets and Swin 0.1 second.

TABLE II
TYPICAL RECEIVE SENSITIVITY FOR 802.11A WITH CISCO AIRONET 350 802.11 A/B/G CARDS

Transmission Rate	RSSI
6 Mbps	-87 dBm
9 Mbps	-87 dBm
12 Mbps	-87 dBm
18 Mbps	-87 dBm
24 Mbps	-84 dBm
36 Mbps	-79 dBm
48 Mbps	-74 dBm
54 Mbps	-72 dBm

The answer to the above questions is discussed below.

A. Diversity Issue

MadWifi device driver provides transmit and receive diversity control. When running multicast/broadcast traffic, after certain time period of operation ($20-30$ seconds) every second packet is lost at the receiver. Having such a deterministic loss pattern cannot be attributed to channel characteristics. This clearly suggests faulty handling of multicast/broadcast packets in the driver. Looking at Fig. 2, we conclude that the unicast packet reception does not suffer from such a flaw. The effect of this fault becomes more severe as the experimentation time increases as shown in Fig. 3. During the initial $20-30$ seconds of the experiment, rx_no_fault is not influenced by this fault. However, after such an initial time, this irregularity reduces rx_no_fault leading to inaccurate results by Eqs. 4 and 5.

The default antenna selection allows user to select the antenna port for transmissions. However, multicast/broadcast packets use a simple round-robin antenna port selection process, and hence, they do not respect the default transmit antenna setting causing half of the packets to be dropped although they are reported as transmitted. Enabling the transmit diversity did not resolve the issue, and we concluded that there is a deep rooted bug in the driver. For our purpose, we simply switched off both transmit and receive diversities as a fix to the problem.

B. Receive Sensitivity Issue

From Figs. 2 and 3, we can infer that both the unicast and the broadcast packets received at a node suffer from an unreasonably low sensitive behavior of NIC for low transmission rates. This concern has its roots in the HAL (Hardware Abstraction Layer) of the MadWifi driver. HAL, which acts as a wrapper around the hardware registers is proprietary of Atheros and is distributed in binary form.

An arriving 802.11 wireless signal is mixed with ubiquitous interference. The received power may be increased although the signal of interest is weak and distorted. This causes *false detects*, i.e, erroneously categorizing an interfering signal as a valid signal. There is an algorithm named ANI (Ambient Noise Immunity) that resides in HAL. The purpose of ANI is to reduce the false detects by maintaining certain interference immunity parameters based on empirical measurements. 'Noise' in ANI does not stand for thermal noise, but represents the ubiquitous interference.

When operating in non-station mode with OFDM PHY, ANI operates in a faulty manner resulting in reduced receive sensitivity for different transmission rates. We developed a solution to tackle ANI in the driver even though HAL is not transparent. HAL provides two powerful APIs (Application Program Interfaces), namely, *ath_hal_getcapability()* and *ath_hal_setcapability()*. Through *ath_hal_getcapability()*, we can determine whether the HAL version being used supports some specific capability, and *ath_hal_setcapability()* allows us

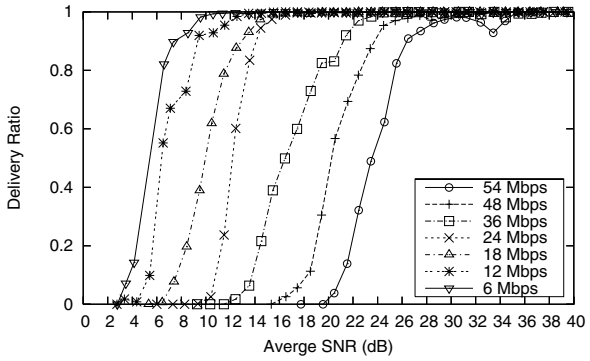


Fig. 5. Profile using broadcast packets with 1 dB bucket along the x-axis for different transmission rates and Swin 0.1 second.

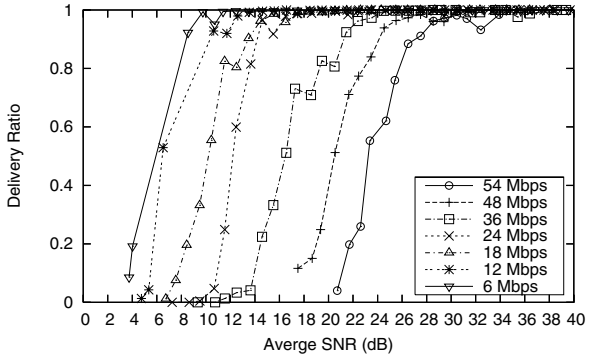


Fig. 6. Profile using broadcast packets with 1 dB bucket along the x-axis for different transmission rates with Swin 1 second obtained through post processing.

to enable or disable a capability. Of our interest is the HAL capability, called *HAL_CAP_INTMIT*. This capability deals with the interference mitigation being performed by ANI. Our implementation switches off this capability when operating in ad-hoc mode with OFDM PHY.

Figs. 4 and 5 show the scatter plots and the corresponding interpolated curves for different transmission rates drawn using the broadcast packets. The curves are more intuitive and prove that our fixes for the above mentioned problems work well. In Fig. 4, each raw data point in the scatter plot represents delivery ratio value over Swin size of 0.1 second, which corresponds to 14 broadcast packet transmissions from a node. The total number of received packets vary from [0,14] for Swin 0.1 second. Thus there can be 15 different delivery ratio values for a particular average SNR value.

Fig. 6 represents the interpolated curves with Swin of 1 second, which corresponds to 125 broadcast packet transmissions. Note that the scatter plots and profile curves can be easily obtained by post processing tool for other Swin values. The profile obtained with Swin interval of 0.1 second is more dependable and accurate than its counter part obtained with Swin of 1 second. If the Swin interval is comparable to *coherence time* (time duration over which channel characteristics do not show much variation), the average SNR of packets received in this interval is nearly similar to SNR of individual packets, leading to accurate delivery ratio vs. SNR relation determintation for this interval. On the other hand, a larger Swin interval causes inaccurate delivery ratio vs. SNR relation determination as the average SNR of packets received over that interval can be very different from the SNR of individual packets.

V. Conclusion and Future Work

We presented a new method called QUEST that accurately estimates IEEE 802.11 wireless link quality with no in-band signaling overhead. We also performed the analysis for the transmission time overhead incurred by both QUEST and other existing schemes, for the quality estimation of channel. QUEST depends on the delivery ratio vs. SNR relation, called the profile. The profile, in this paper, is developed through extensive experimentation with 802.11a PHY module, which it can be done with any other PHY. During the course of experimentation, we unearthed bugs in the popularly employed MadWifi driver, which can significantly degrade the performance of mesh networks. For future work, we plan to compare the network performance with different routing metrics like ETX [1] and ETT [2] when operating with broadcast/unicast probe methods and QUEST. We are in progress of extending the use of QUEST in transmission rate adaptation, thus achieving a true cross layer rate adaptation.

References

[1] D. S. J. De Couto, D. Auayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Networks," in *Proc. ACM MobiCom'03*, San Diego, CA, USA, Sep. 2003, pp. 134–146.

[2] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *Proc. ACM MobiCom'04*, Philadelphia, PA, USA, Sep. 2004, pp. 114–128.

[3] ——, "Comparison of Routing Metrics for Static Multi-Hop Wireless Networks," in *Proc. ACM SIGCOMM'04*, Portland, OR, USA, Sep. 2004, pp. 133–144.

[4] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-Based Models of Delivery and Interference in Static Wireless Networks," in *Proc. ACM SIGCOMM'06*, Pisa, Italy, Sep. 2006, pp. 51–62.

[5] L. Qiu, Y. Zhang, F. Wang, M. K. Han, and R. Mahajan, "A General Model of Wireless Interference," in *Proc. ACM MobiCom'07*, Montreal, Quebec, Canada, Sep. 2007, pp. 171–182.

[6] A. Kashyap, S. Ganguly, and S. R.Das, "A Measurement Based Approach to Modelling Link Capacity in 802.11-Based Wireless Networks," in *Proc. ACM MobiCom'07*, Montreal, Quebec, Canada, Sep. 2007, pp. 242–253.

[7] G. Judd, X. Wang, and P. Steenkiste, "Efficient Channel-aware Rate Adaptation in Dynamic Enviroments," in *ACM Mobisys'08*, Breckenridge, Colorado, U.S.A, Jun. 2008.

[8] MadWiFi: Multiband Atheros Driver for WiFi. http://madwifi.org/.

[9] Ubuntu: A community developed, linux-based operating system. http://www.ubuntu.com/.

[10] Cisco Systems. http://www.cisco.com/en/US/products/hw/wireless/.

[11] D. Qiao and S. Choi, "Goodput Enhancement of IEEE 802.11a Wireless LAN via Link Adaptation," in *Proc. IEEE ICC'01*, Helsinki, Finland, Jun. 2001, pp. 1995–2000.

[12] Iperf: The TCP/UDP Bandwidth Measurement Tool. http://dast.nlanr.net/Projects/Iperf/.