Loss Differentiation: Moving onto High-Speed Wireless LANs

Ruwaifa Anwar *, Kamran Nishat *, Mohsin Ali, Zahaib Akhtar, Haseeb Niaz, Ihsan Ayyub Qazi
Computer Science Department, LUMS, Pakistan
Email:{14100158,mkamran,14100148,zahaib.akhtar,haseeb.niaz,ihsan.qazi}@lums.edu.pk

***Abstract*— A fundamental problem in 802.11 wireless networks is to accurately determine the cause of packet losses. This becomes increasingly important as wireless data rates scale to Gbps, where lack of loss differentiation leads to higher loss in throughput. Recent and upcoming high-speed WLAN standards, such as 802.11n and 802.11ac, use frame aggregation and block acknowledgements for achieving efficient communication. This paper presents BLMon, a framework for loss differentiation, that uses loss patterns within aggregate frames and aggregate frame retries to achieve accurate and low overhead loss differentiation. Towards this end, we carry out a detailed measurement study on a real testbed to ascertain the differences in loss patterns due to noise, collisions, and hidden nodes. We then devise metrics to quantitatively capture these differences. Finally, we design BLMon, which collectively uses these metrics to infer the cause of loss without requiring any out-of-band communication, protocol changes, or customized hardware support. BLMon can be readily deployed on commodity devices using only driver-level changes at the sender-side. We implement BLMon in the ath9k driver and using real testbed experiments, show that it can provide up to 5× improvement in throughput.**

I. Introduction

Packet losses in 802.11 wireless networks can occur due to noisy channel conditions (e.g., signal attenuation, fading), collisions or hidden nodes[1]. A fundamental problem in these networks is to accurately infer the cause of a packet loss as it determines the corresponding action to be taken at the link-layer. For example, in case of collisions, the exponential backoff algorithm should be invoked, under noisy channel conditions, the bit rate adaptation algorithm should be used, and under hidden nodes, RTS/CTS may be enabled [1], [2].

The inability of a wireless sender to differentiate between losses in real-time causes unnecessary exponential backoffs (e.g., in case of noise-related losses), lowering of bit rate (e.g., due to collisions), and increase in collision probability in case of hidden nodes[2], all of which degrade throughput [1], [3]. With increases in physical layer (PHY) data rates, from hundreds of Mbps in 802.11n [4] to over Gbps in 802.11ac [5], this degradation becomes more significant as unnecessary exponential backoffs and the use of non-optimal bit rates lead to higher loss of throughput [6].

*Co-primary authors

[1]We use the term *collision* to refer to an event which occurs when nodes, that *can* carrier sense each other, have overlapping transmissions at a receiver. Losses due to *hidden nodes* occur when two or more senders, that *cannot* carrier sense each other, have overlapping transmissions at a receiver.

[2]Lowering of bit rate in response to packet losses due to hidden nodes increases the transmission duration which conflicts with the exponential backoff algorithm used by 802.11 to avoid a collision in the next retry.

This paper presents BLMon (Burst Loss Monitor), a low overhead loss differentiation scheme for high-speed wireless local area networks (WLANs) such as 802.11n and 802.11ac. BLMon achieves accurate loss differentiation without requiring any protocol changes, customized hardware support, or out-of-band communication. It can be readily deployed in commodity devices as it only requires driver-level changes at the sender-side. BLMon achieves these properties by leveraging *frame aggregation* and *block acknowledgements* (blockACK); key features used by 802.11n and emerging standards, such as 802.11ac, for achieving high MAC-layer efficiency [6]. With frame aggregation, multiple frames (or MPDUs) are packed into an aggregate frame (called A-MPDU) and sent as a single transmission. The success or failure of each frame is indicated in the blockACK returned by the receiver. BLMon leverages the differences in frame loss patterns within A-MPDUs and A-MPDU retries (ARET) to determine the cause of a packet loss.

In particular, we find that frame losses due to noise tend to be scattered within A-MPDUs whereas collisions and hidden nodes lead to bursty losses. In addition, while most A-MPDUs experience small number of retries (usually less than two) due to noise and collisions, large number of retries are observed in case of hidden nodes. A-MPDUs retries are small in case of noise because the PHY header/preamble is sent at a basic rate which is lower than data rates, thus making A-MPDU retries less likely even when the packet error rate (PER) for data packets is very high [4]. In case of collisions, the backoff algorithm of 802.11, which de-synchronizes transmissions using randomization, reduces the probability of successive collisions. With hidden nodes, most A-MPDUs experience large number of successive collisions for two reasons: (a) The absence of carrier sense allows an overlapping transmission to arrive at any time during an ongoing transmission. (b) An A-MPDU in 802.11n can carry up to 64 frames (as opposed to one in 802.11a/b/g). The longer duration of a transmission, coupled with physical layer capture, leads to large number of A-MPDU retries. We systematically validate these observations by carrying out an extensive measurement study on a real 802.11n testbed containing heterogeneous quality links.

To quantitatively capture the differences in frame loss patterns and A-MPDU retries, we propose novel metrics, that are computed upon the receipt of a blockACK. BLMon uses the joint distribution of these metrics to infer the cause of a loss. While our evaluation focuses on 802.11n, the proposed scheme is applicable to any standard that supports frame aggregation

978-14799-3360-0/14/$31.00 ©2014 IEEE

and blockACKs.

Several schemes have been proposed in the past for wireless loss differentiation [1], [2], [7], [8], [9], [10], [11]. While most of these schemes target 802.11a/b/g networks, they can be used with 802.11n/ac. SoftRate [1] uses per-bit confidences from the physical layer to determine interference-free BER but requires customized hardware support. COLLIE [2] uses bit-error patterns in receptions to isolate collisions from a weak signal. However, to do this, a COLLIE receiver relays the entire frame to the sender for pattern analysis, incurring significant overhead. In [7] and [8], authors use fragmentation or RTS/CTS for loss differentiation but it is well-known that using them can result in significant overhead, especially at high data rates [1], [11]. In addition, [7] also uses PIFS; a non-standard inter-frame spacing for 802.11 DCF. In RRAA [9], RTS/CTS is adaptively enabled to differentiate between losses. However, it also incurs the overhead of RTS/CTS. MiRA [11] and [10] use frame loss patterns for differentiation but do not distinguish between losses due to collisions and hidden nodes. Thus, prior schemes either require customized hardware support, protocol changes, incur large overhead, or differentiate between only two kinds of losses.

BLMon overcomes all these limitations. It distinguishes between losses due to noise, collisions, and hidden nodes by using information already available at the sender without requiring customized hardware support or protocol changes. It incurs low overhead as it does not require any additional frame transmissions or the use of RTS/CTS or fragmentation.

Altogether, this paper makes the following contributions:

- **Detailed measurement study**: We conduct extensive experiments on a real 802.11n testbed to identify patterns in frame losses within A-MPDUs and A-MPDU retries and show that they can be leveraged for differentiating between losses due to noise, collisions, and hidden nodes. The measurement study considers diverse settings including heterogeneous quality links, range of bit rates, different number of spatial streams, etc.
- **Design of metrics**: We propose metrics to quantitatively capture the differences in frame loss patterns and analyze their effectiveness.
- **Design and analysis of BLMon**: We propose BLMon, a low overhead scheme, that collectively uses the proposed metrics to perform accurate loss differentiation under diverse scenarios.
- **Driver-based implementation and evaluation**: We implement BLMon (∼80 lines of C code) in ath9k [12]; an open source driver for Atheros IEEE 802.11n based chipsets, and conduct extensive evaluation of BLMon across a range of scenarios including varying number of clients, mobility, application sending rates, and hidden nodes. Our results show that BLMon improves throughput by up to 5× with UDP and 1.9× with TCP.

The rest of this paper is organized as follows. We discuss background and our experimental platform in Section II. The measurement study is presented in Section III. Metrics and BLMon are presented in Sections IV and V, respectively, followed by evaluation in Section VI. We discuss related work in Section VII followed by some discussion in Section VIII.[2] We offer concluding remarks in Section IX.

II. Background and Experimental Platform

In this section, we discuss some of the key features supported by 802.11n for improving performance over earlier standards. These features are shared by 802.11ac [5]; an emerging WLAN standard. We also describe our experimental platform in this section.

Multiple-Input Multiple-Output (MIMO): MIMO employs multiple transmit and receive antennas to achieve spatial diversity and spatial multiplexing. With spatial diversity, a single stream is sent over all antennas. It leverages the independent fading over multiple antenna links to enhance signal diversity. With spatial multiplexing, each antenna transmits independent and separately encoded, spatial stream [4].

Frame Aggregation: It amortizes the access overhead by combining several MAC Protocol Data Units (MPDUs) into an A-MPDU frame, which is sent as a single PHY Protocol Data Unit (PPDU).

Block Acknowledgements (blockACK): With this mechanism, multiple MPDUs in an A-MPDU can be individually acknowledged in a single blockACK frame. The blockACK contains a 64-bit bitmap, where each bit represents the status (success or failure) of an MPDU.

A. Experimental Platform

We conduct our experiments using Ubiquiti Litestation as the AP based on Atheros AR 9160 2.4/5 GHz MAC/BB MIMO chipset. The clients are Core 2 Duo desktops that use PCI based TP-Link 802.11n cards having the Atheros AR5416 chipset. Both chipsets allow data rates of up to 130 Mbps and 300 Mbps for 20 MHz and 40 MHz channels, respectively. Frame aggregation and blockACKs are available and have been used in all experiments. Both AP and clients are of 3×3 antenna configuration and have 2 spatial streams. The AP has OpenWRT (Attitude Adjustment 12.09) installed on it whereas the clients run Ubuntu 12.04 but both are compiled to use Linux kernel 3.3.7. All nodes use the ath9k driver.

To sync wireless nodes, we use the Precision Time Protocol (PTP) [13]. The ath9k driver reports the number of MPDUs in an A-MPDU, number of MPDUs received with errors, number of retransmissions experienced by an A-MPDU, and whether a given A-MPDU experienced *xretries*. *xretries* happen when the number of A-MPDU retries exceed the maximum retransmissions allowed (which is 4 by default in ath9k). The driver uses a maximum A-MPDU length of 32 MPDUs and accumulates 4 ms worth of data before beginning an A-MPDU transmission.

III. Analysis of Wireless Losses

In this section, we conduct a detailed measurement study to analyze MPDU loss patterns within A-MPDUs and A-MPDU retries. Our results show that losses due to noise and collisions can be discerned based on the burstiness in MPDU losses whereas collisions and hidden nodes can be differentiated through A-MPDU retries and xretries.

Testbed Setup: We conduct our experiments on an indoor testbed in a campus setting. Figure 4 shows the floor plan

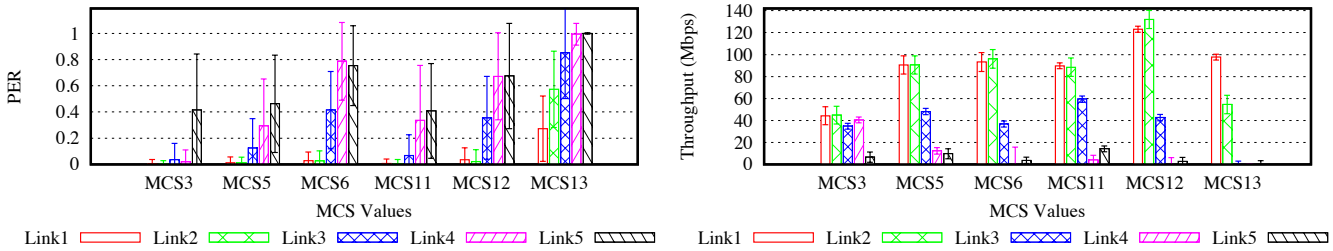


Fig. 1: PER and average throughput on testbed links at different MCS values.

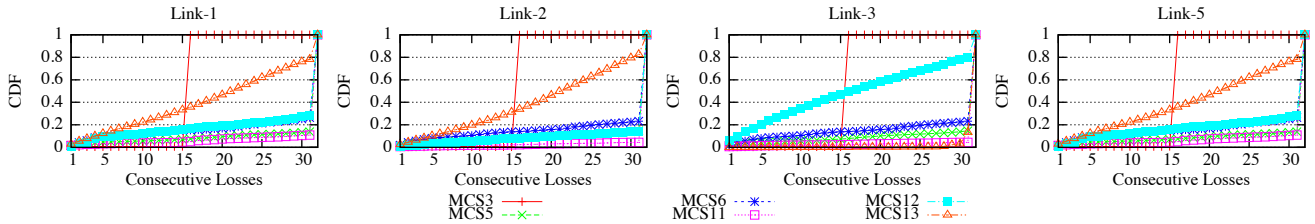


Fig. 2: CDF of consecutive MPDU losses within A-MPDUs across different testbed links and MCS values under noise.

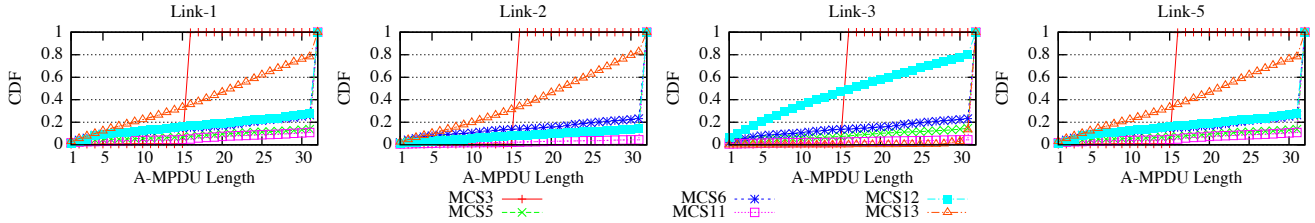


Fig. 3: CDF of A-MPDU lengths on different testbed links and MCS values under noise.

Link	Link-1	Link-2	Link-3	Link-4	Link-5
PER(min,max)%	(0.01,28)	(0.02,58)	(3,85)	(2,98)	(41,99)

TABLE I: Minimum and maximum PERs observed at testbed links.

of the building we use for running our experiments. Spots 0 to 6 represent different locations where the wireless nodes are placed. In several experiments, we fix the receiver at spot 0 and place senders at spots 1-5. When the sender is at spot i, we refer to the sender/receiver pair as Link-i, where $i \in \{1, 2, .., 5\}$. For generality of results, the testbed is organized to contain a diverse mix of links including line-of-sight (e.g., Link-1) and non-line-of-sight links (e.g., Links 2-5), links with different distances (e.g., Link-3 and Link-5), and links that experience different fading and shadowing effects (e.g., Link-1 and Link-4). We use location 6 only for setting up experiments involving hidden nodes. While carrying out experiments, we ensure that no external interference source affects our experiments. All the reported experimental results were conducted with channel bonding (i.e., with 40 MHz channels) as done in [11][3].

Figure 1(a) shows the packet error rate (PER) on all links in our testbed for different MCS (Modulation and Coding Scheme) index values (see Table II). Note that these links cover a broad range of PERs that can occur in wireless networks ranging from 0.01% to 99% as shown in Table I.

[3]We also conducted experiments without channel bonding but did not observe any significant performance difference. We omit those results due to space limitations.

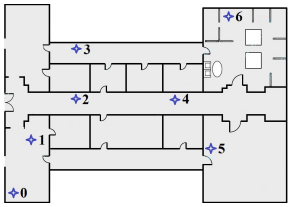


Fig. 4: Floor plan

MCS index	Bitrate (Mbps)	Streams	Modulation
2	40.5	1	QPSK
3	54	1	16-QAM
5	108	1	64-QAM
6	121.5	1	64-QAM
11	108	2	16-QAM
12	162	2	16-QAM
13	216	2	64-QAM

TABLE II: 802.11n rates

There are two important observations about these links: (a) When using the diversity-oriented, single-stream mode (corresponding to MCS values 3, 5, and 6 shown in Table II) or the spatial multiplexing driven, double-stream mode (corresponding to MCS values 11, 12, and 13), link PER increases as the MCS value increases. This happens because increasing the MCS value within each mode corresponds to an increase in the bit rate. (b) However, this monotonic increase in PER does not hold *across* modes e.g., the PER on Link-3 is much higher at MCS 6 than on MCS 12. Generally, links experience higher PERs in the spatial multiplexing mode as independent streams are sent on each antenna. This is consistent with the findings reported in [11]. Figure 1(b) shows the corresponding saturation throughput at various MCS values. Observe that at high MCS values, Link-3 and Link-5 achieve low throughput.

A. Noise Loss Analysis

We now analyze frame losses within A-MPDUs and A-MPDU retries on testbed links at various MCS values. We fix the receiver at location 0 (see Figure 4) and place transmitters at locations 1-5, one at a time, to collect our traces. Since

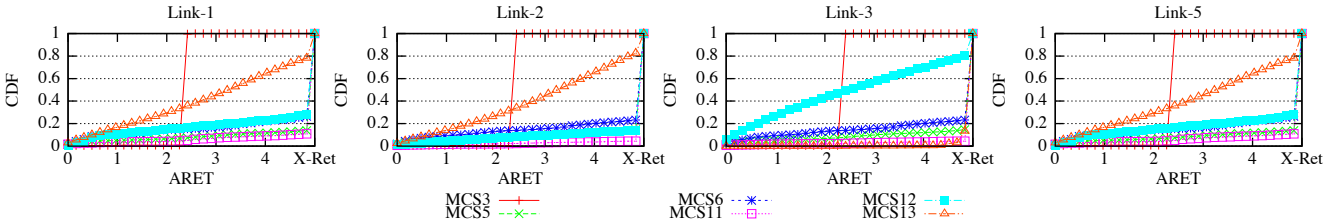
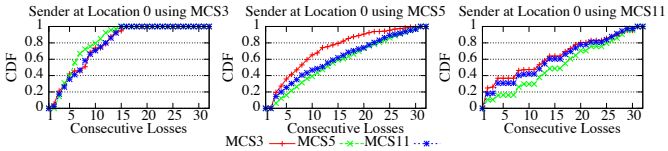


Fig. 5: CDF of A-MPDU retries (ARET) across different testbed links and MCS values under noise.



(a) (b) (c)

Fig. 6: CDF of consecutive losses for sender-A under collisions when paired with another sender using either MCS3, MCS5 or MCS11.

Link-4 and Link-5 had similar characteristics, we omit former due to space limitations.

1) Consecutive Losses: Figure 2 shows the cumulative distribution function (CDF) of the maximum number of consecutive losses within an A-MPDU across testbed links and Figure 3 shows the corresponding CDF of A-MPDU length. Observe that on Link-1 and Link-2, ∼80% and ∼70% of A-MPDUs experience less than 3 consecutive losses, respectively, across *all* MCS values. On Link-3 and Link-5, ∼70% and ∼60% of A-MPDUs experience less than or equal to 4 and 5 consecutive losses, respectively, for all MCS values except MCS 13. Note that on Link-3 and Link-5, the PER is 85% and 98%, respectively at MCS 13, which causes almost all frames within an A-MPDU to get lost. Across all links and MCS values, most A-MPDUs experience up to 4 consecutive losses as long as the PER is less than 80%.

2) A-MPDU Retries: Figure 5 shows the CDF of A-MPDU retries on testbed links at various MCS values. Observe that on Link-1 and Link-2, at least 93% of A-MPDUs do not experience any retries. On Link-3 and Link-5, ∼75% of A-MPDUs do not experience any retries, and ∼16% A-MPDUs are retried exactly once except at MCS 13. This happens because A-MPDUs are retried *only* when either the PHY header or the blockACK gets corrupted and each of these are more robust to channel errors compared to MPDUs. This is because the PHY header and the blockACKs are sent at basic rates which are lower than data rates [4]. Moreover, a higher basic rate is used for transmitting the PHY header when a higher corresponding data rate is used. Note that on Link-5, with MCS 13, ∼25% of A-MPDUs are retried at least 3 times. This happens because now even the basic rate is not robust to channel errors. Note that across all links and MCS values, zero or negligible xretries occurred.

These results show that with noise, the maximum number of consecutive losses remains small as long as the link PER is less than 80%. Most A-MPDUs experience either no retries or at most one retry except when the PER exceeds 70%. In addition, they experience either zero or negligible xretries.

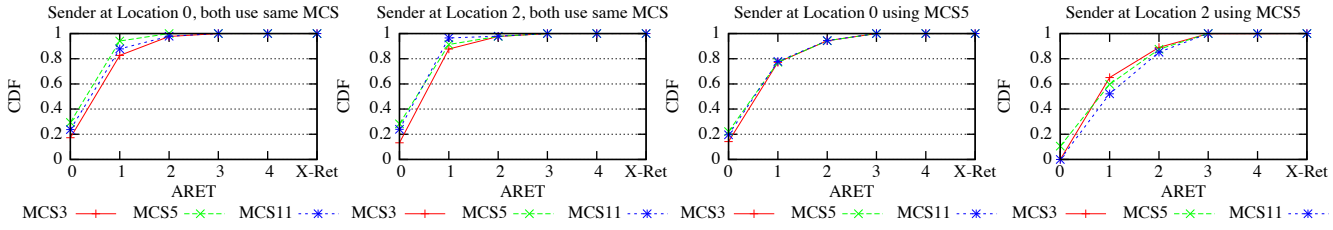
B. Collision Analysis

Using two senders that can carrier sense each other, we now perform collision experiments when they transmit to a common receiver. Note that in this case, a collision occurs when the backoff contention window of each sender becomes zero at the same time. We choose locations 0 and 2 for the senders and location 1 for the common receiver as shown in Figure 4. To establish ground truth about collisions, we record the sending time, t_s, of each A-MPDU (before it is sent to the hardware for transmission) and the time of receipt, t_r, of the corresponding blockACK at each sender. We refer to $[t_s,t_r]$ as the *transmission window* of an A-MPDU. A collision is inferred when there is an overlap in the transmission windows of two senders and the retry count for at least one of the senders is greater than zero. Note that the senders are time synced using PTP to ensure a clock skew of less than 10 μs.

The Atheros-based wireless cards we use implement the Message-in-Message (MIM) functionality [14]. With MIM, if enough synchronization bits within the preamble are received correctly, the receiver locks on that packet and only attempts to decode a later overlapping packet if the packet is more than 10dB stronger than the first transmission [15]. This leads to two capture thresholds based on the timing of the signal.

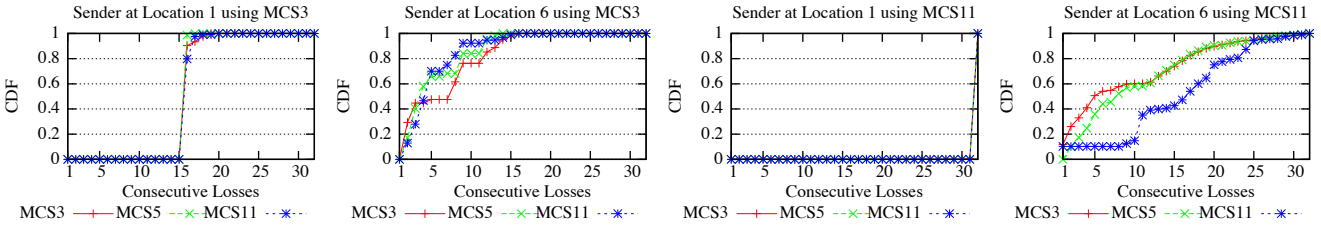
When an A-MPDU collision occurs, a receiver observes one of the following: (1) an A-MPDU is received with zero or more frames in error and (2) an A-MPDU is not received. Scenario 1 occurs when a stronger signal from one transmitter dominates the other resulting in the correct reception of its signal due to PHY capture. The sender which loses in PHY capture now acts as an interferer to the winning sender's transmission, which may lead to the corruption of MPDUs for the winning sender. The second scenario occurs in two cases: (a) when the relative signal strength for both the transmissions is approximately same, thus causing the corruption of the PHY header/preamble and (b) when a receiver is already locked onto a weak signal and then a stronger signal arrives. Due to PHY capture, the weaker signal may be dropped leading to A-MPDU loss.

1) Consecutive Losses: We first identify scenarios in which a sender's signal wins in PHY capture. Figure 6 shows the CDF of maximum consecutive losses for both the senders. The sender at location 0 (called sender-A) experiences bursty losses. For example, at least 65% of A-MPDUs experience at least 5 consecutive losses. Note that this observation holds even when senders use different bitrates as shown in Figures 6(a), 6(b), and 6(c) Moreover, this burstiness in losses increases when sender-A uses a higher bitrate as it require higher SNR for correct reception. This is due to the fact that sender-A's A-MPDUs have a higher signal strength, consequently,



(a) CWmin=15, CWmax=1023 (b) CWmin=15, CWmax=1023 (c) CWmin=1, CWmax=8 (d) CWmin=1, CWmax=8

Fig. 7: CDF of A-MPDU retries for transmitters at location 0 and location 2. In Figure (a) and (b), senders use the same bitrate and the default values for CWmin and CWmax. In Figure (c) and (d), senders-A uses MCS5 whereas sender-B uses different bitrates. In addition, both the senders use smaller values for CWmin and CWmax.



(a) (b) (c) (d)

Fig. 8: CDF of consecutive losses of senders at location 1 and 6 when they form hidden nodes. (a) Sender at location 1 uses MCS3 whereas the other sender uses different bitrates and (b) Sender at location 6 uses MCS3 whereas the other sender uses different bitrates. Both senders were observed to have an A-MPDU length of 16 frames for more than 99% of A-MPDUs.

sender-B (i.e., sender at location 2) act as an interferer when their packets overlap in time. Note that sender-B experiences corruption of *all* packets under collisions due to its lower signal strength.

2) A-MPDU Retries: Figure 7 shows the CDF of the number of A-MPDU retries (Note that we only consider A-MPDUs that experience at least one MPDU corruption in these plots). Observe that for both the senders ∼88% of the A-MPDUs are retried at most once, whereas, no A-MPDUs are retried more than 4 times. This happens because the 802.11 DCF doubles the contention window size on each successive A-MPDU loss and since senders pick a random backoff value, this helps in considerably reducing the probability of successive PHY header collisions. Note that these results hold when senders use different bit rates.

Smaller Contention Windows: To emulate a scenario comprising of large number of senders, we now reduce the minimum contention window size (CWmin) and the maximum contention window size (CWmax) to 1 and 8, respectively, in order to increase the probability of collisions. Figures 7(c) and 7(d) shows the CDF of A-MPDU retries. Observe that retries increase especially for sender-B which loses in capture. In particular, ∼90% of A-MPDUs experience less than 2 retries and a small fraction experiences 3 retries. However, the number of xretries experienced are still negligible.

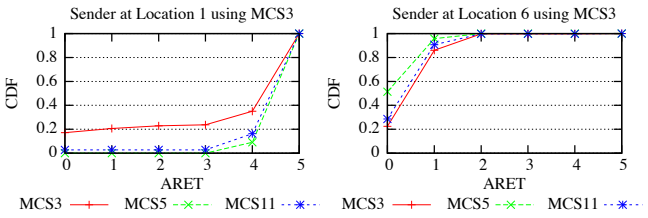
These results show that under collisions, senders either experience PHY header/preamble corruption or bursty losses (under PHY capture). In addition, majority of A-MPDUs are retried at most 2 times and xretries tend to be negligible.

C. Hidden Node Analysis

To obtain loss patterns in the presence of hidden nodes, we use two senders that communicate with a common receiver such that they are unable to carrier sense each other. We place one sender at location 1 (sender-1) and the other at location 6 (sender-6). The receiver is placed at location 4 (see Figure 4). We establish ground truth by ensuring that packet losses due to noise are negligible. We then determine loss patterns through blockACK information.

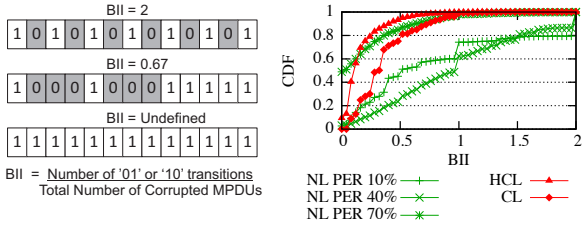
1) Consecutive Losses: Figure 8 shows the CDF of maximum consecutive losses experienced by the two senders when they use MCS 3 and MCS 11 and the other sender uses MCS 3, MCS 5, and MCS 11, one at a time. Observe that under hidden nodes, senders either experience complete A-MPDU losses or bursty losses. In this scenario, the capture effect has a much more significant impact on performance compared to regular collisions. Since a concurrent transmission may begin at any instant, the sender which has a relatively weaker signal strength at the receiver is likely to suffer greatly. Figure 8 shows that ∼99% of A-MPDUs (that had at least one MPDU loss) experienced complete MPDU losses, whereas sender-6 experiences bursty losses (e.g., more than 50% of A-MPDUs experienced at least 7 consecutive losses with MCS3 as shown in Figure 8(b)). Also observe that as the bit rate of sender-6 increases, its losses become more bursty as higher bit rates require higher SNR to decode packets.

2) A-MPDU Retries: We observe that a significant number of A-MPDUs under hidden nodes experience large number of retries and xretries (see Figure 9). This happens for the following reasons: (a) since nodes are unable to carrier sense, their transmissions frequently overlap, (b) retries are exacerbated by frame aggregation because senders can transmit up to 64 MPDUs in a single A-MPDU with 802.11n, which increases the transmission duration and thus the likelihood of successive collisions, and (c) due to capture (specifically, due to MIM), a receiver will start decoding a stronger signal that arrives at any time during an ongoing transmission.



(a) (b)

Fig. 9: CDF of A-MPDU retries for (a) sender-1 using MCS3 with varying bitrates of the other sender and (b) sender-6 using MCS3 under the hidden node scenario. Note that the results at other MCS values were similar, which we omit due to space limitations.



(a) (b)

Fig. 10: CDF of BII.

Figure 9 shows the CDF of A-MPDU retries experienced by both the senders. Observe that the losing sender (sender-1) gives up retrying in as many as ∼95% of cases with MCS 3 i.e., xretries occur very frequently. The results at other MCS values were similar, which we omit due to space limitations. With ath9k, no A-MPDU is retried more than 4 times by default. For the winning sender, ∼95% A-MPDUs are retried less than 4 times. This is due to the stronger signal of sender-6, which causes it to consistently win in capture.

These results show that under hidden nodes, the sender that loses in capture experiences complete losses and large number of xretries whereas the winning sender, similar to the collision case, observes bursty losses.

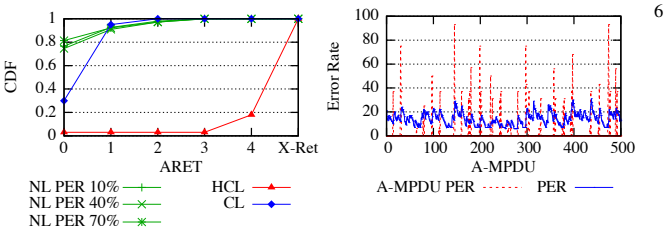
IV. Capturing Loss Patterns

In the last section, results showed losses due to noise and collisions can be discerned by leveraging the differences in MPDU loss patterns. In addition, losses due to collisions and hidden nodes can be differentiated through A-MPDU retries and xretries. In order to capture these patterns quantitatively, we now define some metrics.

A. Metrics

To quantify the differences in loss patterns due to noise and collisions, we need a metric which assigns different values to patterns depending on the degree of *isolation* of MPDU losses within them, thus providing a measure of the burstiness in losses. This motivates the design of the following metrics.

Burst Isolation Index (BII): It is the ratio of the number of "01" and "10" transitions in the bitmap of the received blockACK to the total number of unsuccessful MPDUs, where a 1 and 0 indicate successful and unsuccessful MPDUs as given in the bitmap vector of a blockACK, respectively (see Figure 10(a)). BII indicates how *isolated* the losses are. The maximum value of BII is two (when the entire blockACK bitmap is composed of alternating 1s and 0s where the sequence starts and ends with a 1) and the minimum value is zero (when the bitmap is all 0s i.e., all MPDUs are lost). The more isolated losses we have in a burst, the greater the value is (> 1). The intuition behind the metric is to give more weight to isolated losses (such as the "101" transition).



(a) (b)

Fig. 11: A-MPDU retries and comparison of PER and $\overline{\text{PER}}$.

Number of A-MPDU Retries (ARET): This is the number of times an A-MPDU is retransmitted. When the maximum retry limit is exceeded, it is counted as a one xretry. In ath9k, the default retry limit is 4 times.

History based PER (H-PER): The H-PER metric takes into account the short-term history of PER to isolate noise from collisions. With H-PER, a sender maintains an exponentially weighted moving average $\overline{\text{PER}}$ of per-A-MPDU PER (APER) while considering only those A-MPDUs that are concluded to have been affected by noise. When the APER exceeds $\overline{\text{PER}}$ beyond a certain threshold, a collision is inferred.

B. Evaluation of Metrics

We now evaluate these metrics for differentiating between losses. Figure 10(b) shows the CDF of BII for noise, collisions, and hidden nodes. Observe that as long as the PER is less than 60%-70%, BII is greater than 0.5 for most A-MPDUs with noise, indicating isolated losses. On the other hand, BII is less than 0.5 for at least 81% of A-MPDUs in case of collisions and hidden nodes, which indicates the bursty nature of losses. Thus BII can be leveraged for differentiating between noise and collisions (i.e., for both regular collisions and hidden nodes) Figure 11(a) shows the CDF A-MPDU retries. Observe that in case of noise and collisions, most A-MPDUs are retried once. However, in case of hidden nodes, more than 98% of A-MPDUs are retried at least 4 times. Thus, A-MPDU retries can be used for differentiating between hidden nodes and other kinds of losses. Figure 11(b) shows the $\overline{\text{PER}}$ and APER. Observe that collisions significantly increase the APER much beyond the $\overline{\text{PER}}$, as indicated by the H-PER metric.

V. Burst Loss Monitor (BLMon)

BLMon uses BII, ARET (including xretries), and H-PER to infer the cause of packet losses. When a blockACK is received, BLMon computes each of these metrics. If $\overline{\text{PER}}$ (due to noise only) is less than θ_{per} and two A-MPDUs experienced xretries in the last w blockACKs seen by the sender, BLMon attributes the loss to hidden nodes. This is based on the observation that xretries are negligible under noise (when PER is less than a certain value) and collisions. If the above condition does not hold, BLMon uses BII and H-PER to differentiate between noise and other sources of loss and employs ARET

Algorithm 1 BLMon Algorithm

Input: blockACK bitmap vector
Output: Loss inference $\in \{noise, collision, hidden\}$
1: **if** $\overline{\text{PER}} < \theta_{per}$ and xretries $= 1$ twice in the last w blockACKs **then** $hidden$
2: **end if**
3: **if** $\overline{\text{PER}} < \theta_{per}$ and BII $< \theta_{bii-col}$ and APER $\geq$ min($\overline{\text{PER}}+k$, 100%) **then**
4: **if** ARET $> \theta_{aret}$ and BII $< \theta_{bii-hid}$ **then** $hidden$
5: **else** $collision$
6: **end if**
7: **else** $noise$
8: **end if**

Metric	DA/FP	NL+CL	NL+HCL
H-PER	DA	92.2	99.9
	FP	46.2	29
BII	DA	86.5	87.9
	FP	27.2	0.7
ARET	DA	-	79.1
	FP	-	14.0
MiRA-LD [11]	DA	53.6	-
	FP	0.2	-

TABLE III: DA and FP for CL/HCL under the proposed metrics and MiRA-LD. Link PER (due to noise) was 10%.

to differentiate between collisions and hidden nodes as shown in Algorithm 1. Different BII thresholds are used in case of collisions and hidden nodes because losses tend to be more bursty in latter scenarios as shown by our results. Note that we disable BLMon when $\overline{\text{PER}}$ exceeds θ_{per}, hence attributing all losses to noise, which is the default behavior in ath9k.

A. Detection Accuracy and False Positive Rate

We now analyze BLMon's performance in terms of the *detection accuracy* (DA) and *false positive rate* (FP). DA is the percentage of A-MPDUs that are correctly detected to have experienced a certain kind of a loss whereas FP is the percentage of A-MPDUs concluded to have experienced a certain kind of a loss when in reality they did not.

We perform comparison with individual metrics as well as with MiRA-LD[4] [11]. In MiRA-LD, a collision is inferred if the retry count is at least 1 and the APER is less than 10%. To improve detection accuracy, MiRA-LD infers a collision if the above condition is met twice in a short timespan. First, we run each of these metrics/schemes on traces containing collision-induced losses and then on traces containing hidden node losses. Both types of traces also have losses due to noise. Figure 12(a) shows the impact of BII thresholds on DA and FP, signifying the tradeoff that exists between DA and FP. Based on the insights from the measurement study, we set θ_{per}=70%, $\theta_{bii-col}$=0.5, $\theta_{bii-hid}$=0.2, and w=5.

Table III shows the performance of individual metrics in comparison to MiRA-LD on a link with a PER of 10%. Observe that H-PER achieves a DA of 92.2% but results in a high FP in case collision losses (CL) and hidden collision losses (HCL). This happens because even at a PER of 10%, spikes in APER (due to noise) causes H-PER to frequently classify NL as CL. BII reduces FP in case of both CL and HCL. This happens because BII explicitly considers the burstiness in losses within A-MPDUs. Thus, BII is better able to discern between losses when APER is high but frame losses are well spread out. We use ARET to primarily detect HCL. It has a DA of 79.1% under HCL along with a FP of 14%. Note that BII and H-PER both outperform MiRA-LD in terms of DA for CL. This happens because MiRA-LD assumes APER to be less than 10% for collision inference. However, under collisions, senders experience APER $>$ 10% quite often, thus degrading the DA of MiRA-LD. This causes a sender to frequently lower its bit rate in case of CL, thus degrading performance (see Section VI). Note that since MiRA-LD relies on at least one retry for collision inference, a sender winning in capture cannot detect a collision.

[4]By MiRA-LD, we mean the heuristic employed in [11] to differentiate noise losses from collisions. MiRA, itself, is a rate adaptation protocol.

Link PER (%)	DA/FP	NL+CL	NL+HCL
0.1	DA	96.2	73.6
	FP	1.6	0
10	DA	85.1	73.7
	FP	26.2	0.1
30	DA	87.8	74.1
	FP	26.9	0.2
55	DA	87.3	75.8
	FP	31.5	0.2
77	DA	93.2	70.4
	FP	45.6	23

TABLE IV: DA and FP for CL/HCL with BLMon.

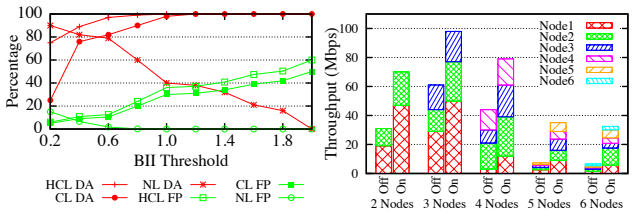
Table IV shows results for BLMon across a range of link PERs. First, observe that BLMon reduces FP in all cases compared to individual metrics. Compared to MiRA-LD, it significantly improves the DA. This is due to the use of multiple metrics employed by BLMon. Observe that FP with BLMon increases gracefully as link PER is increased. This happens because at high PERs, NL becomes more bursty and starts resembling collision losses. On the other hand, FP for HCL does not increase significantly (except when PER is 77%) since xretries still serve as a good indicator for detecting HCL. Our evaluation results in Section VI show that the parameter values we use provide significant throughput improvements across diverse traffic scenarios. BLMon can be further improved by optimizing the weights assigned to each of the metrics. However, we leave this for future work.

B. BLMon Implementation and Complexity

We implement BLMon in the ath9k driver (about 80 lines of C code). $\overline{\text{PER}}$ is already maintained by ath9k and is updated when a blockACK is received. Ath9k, by default, incorporates a retry in the computation of APER (assuming that all MPDUs are corrupted by noise) as APER $= \frac{nFrames \times retries + nBad}{(retries+1) \times nFrames}$, where $nFrames$ is the number of MPDUs in the transmitted A-MPDU, $retries$ is the number of retries, and $nBad$ is the number of MPDUs received in error. However, since a retry may be due to a collision and does not necessarily imply that all MPDUs were corrupted due to noise, we modify APER to only consider losses reported in the blockACKs as APER $= \frac{nBad}{nFrames}$. ARET is also reported by the driver by default. Hence, BLMon only adds the computation of BII, which requires one scan of the 64-bit blackACK bitmap and is thus a $O(1)$ operation as the bitmap size is always fixed.

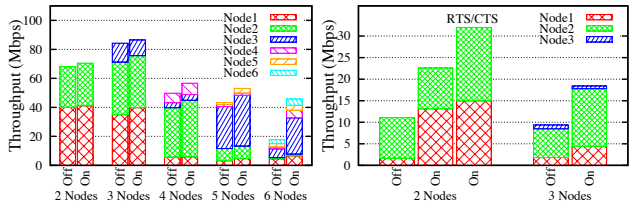
VI. Evaluation

We evaluate the performance of BLMon using testbed experiments under diverse network scenarios including varying



(a) (b)

Fig. 12: (a) Impact of BII threshold on DA and FP (b) Throughput comparison of concurrent senders with and without BLMon when CWmin=15 and CWmax=1023 (default settings)



(a) (b)

Fig. 13: (a) Throughput comparison of concurrent senders with and without BLMon (a) when CWmin=1, CWmax=8 (b) when nodes are hidden from each other with CWmin=15 and CWmax=1023.

number of clients, mobility, application sending rates, and hidden nodes. We use both UDP and TCP in our experiments. *iperf* [16] is used for generating traffic. All the experiments are run for 120 secs and the results averaged over five runs.

For throughput performance analysis, we implement a link adaptation protocol that uses BLMon to invoke rate adaptation only in case of noise-related losses and enable RTS/CTS in case of hidden nodes. In particular, if frame losses are due to a collision or hidden nodes, the protocol does not update the link PER and thus the rate adaptation algorithm is not called. In case of hidden nodes, we also enable the RTS/CTS mechanism to reduce the probability of collisions. In all experiments, we only report the impact of rate adaptation unless stated otherwise. BLMon can be used with any rate adaptation algorithm e.g., RRAA [9] and SampleRate [17]. However, we highlight the benefits of BLMon by using the Atheros MIMO RA [18]; the default rate adaptation algorithm used in ath9k. We have augmented Atheros MIMO RA with BLMon to make it aware of collisions and hidden nodes.

A. Static Clients

We now compare the performance of Atheros MIMO RA with and without BLMon when all clients are static. Our results show that Atheros MIMO RA with BLMon consistently outperforms Atheros MIMO RA without BLMon in all scenarios that we consider.

UDP with Collisions: Figure 12(b) shows the throughput of each sender as a function of the number of senders with and without BLMon. Senders are placed so that they have heterogeneous PERs to the common receiver. Observe that BLMon improves throughput ranging from 1.05× to 2.6× for two and six senders, respectively. This happens because the increase in the number of senders increases the collision probability and thus provides more opportunities for loss differentiation. BLMon reliably detects collisions and prevents a sender from lowering its bit rate. Without BLMon, a sender reduces its bit rate on collisions, thus degrading throughput. To emulate a scenario with larger number of senders, as is common in enterprise and University networks, we reduce CWmin and CWmax to 1 and 8, respectively. Figure 13(a) shows that BLMon improves throughput by up to 5×. Note that we tried several random locations and obtained similar results.

UDP with Hidden Nodes: Figure 13(b) shows the throughput under two and three nodes. Observe that BLMon improves throughput by 2.4× when there are two hidden nodes and only rate adaptation is adjusted. This happens because in the presence of hidden collisions, senders reduce their bit rate when BLMon is not used, which increases collisions and degrades throughput. However, with BLMon, senders are able to use a higher bit rate because it does not update PER on hidden losses, which improves throughput. Note that BLMon provides a 3.1× improvement in throughput when RTS/CTS is also enabled in addition to rate adaptation.

When there are three nodes, one sender has two hidden nodes, each of which can carrier sense each other. During this experiment, we set our AP on Channel 11. We sniff one other node operating on Channel 9 and the other on Channel 10. Both these channels are partially overlapping with Channel 11. Observe that BLMon still provides significant throughput gains (1.8×) even though the improvement is small compared to the two node scenario due to uncontrolled interference.

TCP Clients: We also conduct experiments with TCP flows. Figure 15 shows the performance with TCP sources when used in conjunction with BLMon. Observe that BLMon improves TCP throughput by 1.9×. The throughput with TCP is less compared to UDP because the congestion control mechanism of TCP decreases the sending rate in the face of congestion. This reduces the collision probability, which in turn leads to smaller improvement in throughput. We also conducted experiment with 30 KB TCP flows and found that BLMon provides similar benefits when the offered load is high. At low loads, collisions become fewer, which limits opportunities for loss differentiation and thus the benefit.

Varying Application Sending Rates: Figure 14 shows the throughput and A-MPDU length as a function of application sending rates. Observe that as the sending rate decreases the A-MPDU lengths also decrease. In addition, due to low offered load, the collision probability also reduces. This in turn leads to small or negligible improvement in throughput.

Comparison with MiRA-LD: We implemented MiRA-LD in ath9k and performed comparison with BLMon. Figure 15(b) shows the throughput under MiRA and BLMon in case of collisions and hidden losses. Observe that compared to MiRA, BLMon achieves >20 Mbps of throughput in case of collisions and more than doubles it under HCL.

B. Mobile Clients

In order to assess the responsiveness of BLMon in inferring the cause of loss under fast changing channel conditions, we carry a client and walk from spot 1 to 4 and return at approximately a constant speed of 1 meter/s. Figure 15 shows the throughput of the static and mobile clients in this scenario.

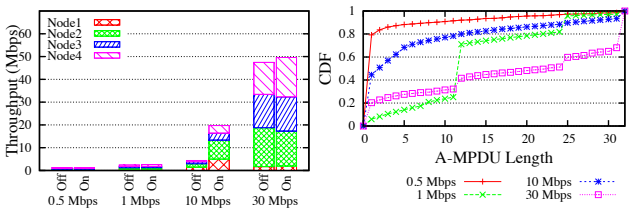


Fig. 14: Performance as a function of application sending rates.

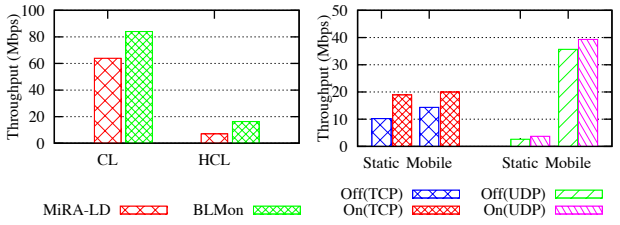


Fig. 15: (a) comparison of BLMon with MiRA-LD under collision and hidden node scenarios in static settings and (b) BLMon performance under static/mobile scenarios with TCP and UDP. In all scenarios, we used two clients wit CWmin=1 and CWmax=8.

Observe that BLMon improves throughput by 1.4× with TCP and 2.3× with UDP.

VII. Related Work

Several schemes have been proposed in the past for loss differentiation [1], [2], [7], [8], [9], [10], [11]. Besides schemes that use PHY layer confidences [1], error patterns in PHY symbols [2], fragmentation or RTS/CTS [7], [8], [9], loss differentiation can also be achieved by estimating the SNR on each packet reception and mapping it to the expected BER using known SNR-BER relationships. However, prior studies have shown that BER at a given SNR can vary by many orders of magnitude between environments [19]. Consequently these protocols must be carefully trained for each operating environment. ZigZag decoding [20] allows nodes to recover from collisions due to hidden nodes by exploiting interference-free stretches in the collided packets. Like Softrate, ZigZag decoding also requires customized hardware support.

VIII. Discussion and Future Work

Impact of A-MPDU length on BLMon. The accuracy of loss differentiation with BLMon depends on the A-MPDU length. A-MPDU length can vary for two reasons: (a) When the application sending rate is small compared to the PHY bit rate and (b) When PER is high, causing the blockACK window to limit the length. Results for (a) were shown in Section VI. In case of (b), results from our measurement study showed that even at high PERs, the average A-MPDU length is fairly high (e.g., the average length was ∼15 when PER was ∼78%).

Impact of PHY Capture. One could ask whether the results reported in the measurement study would be applicable if the wireless cards do not use the MIM functionality. First, Atheros chipsets, that implement the MIM functionality, are widely used [14]. Second, without MIM functionality, we do not expect the insights to significantly change. The reason is that in chipsets (e.g., Prism 2.5 [19]) that do not implement MIM, capture only occurs when the stronger signal arrives first or the stronger signal trails the weaker by less than the synchronization bits of the preamble. This will likely increase A-MPDU retries under collision and hidden nodes. However, we leave a detailed evaluation of such cards for future work.

IX. Conclusion

We design and implement BLMon, a loss differentiation framework for high-speed WLANs such as 802.11n and 802.11ac. Using frame aggregation and blockACKs, BLMon accurately differentiates losses while incurring minimal overhead. BLMon is readily deployable as it only requires driver-level modifications at the sender-side. It does not require any changes in the 802.11 protocols, customized hardware support, or out-of-band communication. We implemented BLMon in the ath9k driver. Using extensive testbed experiments across diverse traffic scenarios we show that BLMon can provide up to 5× improvement in throughput.

Acknowledgements: We thank Ahmed Mehfooz, Haris Choudhary, and Bilal Zaidi for their help with the experiments. We thank Adrian Chadd for his help with ath9k. We also thank Zafar Ayyub Qazi and Fahad Dogar for their feedback on earlier drafts of the paper. This work was supported by the LUMS Faculty Startup Grant and the DRC grant by the Computer Science Department at LUMS.

References

[1] M. Vutukuru, H. Balakrishnan, and K. Jamieson, "Cross-layer wireless bit rate adaptation," in *ACM SIGCOMM 2009*.
[2] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee, "Diagnosing wireless packet losses in 802.11: Separating collision from weak signal," in *IEEE INFOCOM 2008*.
[3] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *IEEE INFOCOM 2003*.
[4] "IEEE Standard for local and metropolitan area networks part 11; amendment 5: Enhancements for higher throughput," 2009, 802.11n.
[5] "802.11ac: The Fifth Generation of Wi-Fi," Cisco Technical White Paper. http://www.cisco.com/en/US/prod/collateral/wireless/ps5678/ps11983/white_paper_c11-713103.pdf,.
[6] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, J. Zhang, and Y. Zhang, "Fine-grained channel access in wireless lan," in *ACM SIGCOMM 2010*.
[7] G. Domenico, D. Malone, D. Leith, and K. Papagiannaki, "Measuring transmission opportunities in 802.11 links," *IEEE/ACM ToN 2010*.
[8] M. Khan and D. Veitch, "Isolating physical per for smart rate selection in 802.11," in *IEEE INFOCOM 2009*.
[9] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *ACM Mobicom 2006*.
[10] M. Cho, H. Jung, S. Oh, T. Kwon, and Y. Choi, "Distinguishing collisions from low signal strength in static 802.11n wireless lans," in *ACM CoNEXT Student Workshop 2011*.
[11] I. Pefkianakis, Y. Hu, S. H. Wong, H. Yang, and S. Lu, "Mimo rate adaptation in 802.11n wireless networks," in *ACM Mobicom 2010*.
[12] Ath9k. http://linuxwireless.org/en/users/Drivers/ath9k/.
[13] Precision Time Protocol (PTP). http://www.ieee1588.com/.
[14] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi, "An experimental study on the capture effect in 802.11a networks," in *WinTECH 2007*.
[15] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala, "Sniffing out the correct physical layer capture model in 802.11b," in *IEEE ICNP 2004*.
[16] "iperf," http://sourceforge.net/projects/iperf/.
[17] J. Bicket, "Bit-rate selection in wireless networks," Master's thesis, Dept. of Electrical Engineering and Computer Science, MIT, Feb 2005.
[18] Minstrel. http://madwifi-project.org/browser/madwifi/trunk/ath_rate/minstrel/minstrel.txt.
[19] J. Camp and E. Knightly, "Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation," in *ACM Mobicom 2008*.
[20] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *ACM SIGCOMM 2008*.