Joint Rate and Channel Width Adaptation for 802.11 MIMO Wireless Networks

Lara Deek†, Eduard Garcia-Villegas‡, Elizabeth Belding†, Sung-Ju Lee§, Kevin Almeroth†
UC Santa Barbara†, UPC-BarcelonaTECH‡, Narus Inc.§
laradeek@cs.ucsb.edu, eduardg@entel.upc.edu, ebelding@cs.ucsb.edu, sjlee@narus.com, almeroth@cs.ucsb.edu

***Abstract*—The emergence of MIMO antennas and channel bonding in 802.11n wireless networks has resulted in a huge leap in capacity compared with legacy 802.11 systems. This leap, however, adds complexity to selecting the right transmission rate. Not only does the appropriate data rate need to be selected, but also the MIMO transmission technique (e.g., Spatial Diversity or Spatial Multiplexing), the number of streams, and the channel width. Incorporating these features into a rate adaptation (RA) solution requires a new set of rules to accurately evaluate channel conditions and select the appropriate transmission setting with minimal overhead. To address these challenges, we propose ARAMIS (Agile Rate Adaptation for MIMO Systems), a standard-compliant, closed-loop RA solution that jointly adapts rate and bandwidth. ARAMIS adapts transmission rates on a per-packet basis; we believe it is the first 802.11n RA algorithm that simultaneously adapts rate and channel width. We have implemented ARAMIS on Atheros-based devices and deployed it on our 15-node testbed. Our experiments show that ARAMIS accurately adapts to a wide variety of channel conditions with negligible overhead. Furthermore, ARAMIS outperforms existing RA algorithms in 802.11n environments with up to a 10 fold increase in throughput.**

I. Introduction

Rate adaptation (RA) selects the best physical bitrate based on time-varying channel qualities. With the emergence of the IEEE 802.11n standard, WiFi technologies have witnessed a significant increase in sophistication and complexity that require novel approaches to RA. RA in 802.11 networks not only needs to choose the operating rate, but also the channel width and MIMO mode. Using MIMO, a solution can send a single stream using *spatial diversity* or multiple simultaneous streams to increase the transmission rate using *spatial multiplexing*.

There are two main approaches to RA: open-loop and closed-loop. In open-loop RA, the transmitter estimates the best rate of the link to the receiver by building on some set of parameters or metrics measured at the transmitter [1]. A closed-loop RA is one in which the receiver's insight into the channel conditions contributes to determining the transmit rate.

As networks become more complex, the use of open-loop RA techniques becomes increasingly inaccurate. An RA solution now has to account for many variables that a transmitter alone cannot accurately capture. In legacy clients, RA mechanisms have to choose among four PHY rates in 802.11b and eight rates in 802.11a/g, whereas 802.11n allows at least 64 combinations from 32 rates and 2 channel widths. By allowing the receiver to contribute to the RA process, we gain an accurate understanding of environment conditions, and the transmitter can more efficiently select the appropriate transmission rate for the link [2].

Perhaps the best RA solution for MIMO environments is to use 802.11n's Channel State Information (CSI) feedback from the receiver to compute the transmission rate [2]. However, complete CSI information is costly to obtain and store [3] and is therefore supported by very few 802.11n devices. Existing RA solutions adopt a practical approach and use a credit-based system [4] or rate sampling [5], [6], [7]. Instead of adapting the rate based on understanding the impact of environment conditions on 802.11n features, these solutions rely on a routine to converge to the best rate, which can be costly or misdirected. Therefore, there is a clear need to build RA solutions over a new, practical link metric that accurately characterizes links in MIMO environments.

In a closed-loop RA model, the receiver's insight into channel conditions is used to compute the transmission rate. A feedback mechanism should therefore be incorporated into the design. In fact, the 802.11n standard supports an explicit feedback system in *MCS Request* and *MCS Feedback* [8]. By exploiting this standard-compliant feedback mechanism, accurate receiver-based RA solutions can be designed for 802.11n MIMO environments.

The state of the art for RA in 802.11n calls for a standard-compliant, closed-loop solution that accurately exploits the new features in 802.11 MIMO environments. The RA solution therefore must adopt a link metric that accurately characterizes MIMO link performance. We propose such an RA solution, called ARAMIS (Agile Rate Adaptation for MIMO Systems).

ARAMIS is a closed-loop, per-packet RA solution that simultaneously adapts both rate and channel width. In previous work, we showed the importance of adapting bandwidth in 802.11n with RA to maximize performance [9], and we believe that ARAMIS is in fact the first RA solution to do so. ARAMIS incorporates a measurement-based, 802.11n link predictor in its design. Given the current channel conditions, the link predictor estimates Packet Reception Rate (PRR) for the given link at all supported rate and bandwidth combinations. Using this information, ARAMIS then selects the best operating point, and sends the feedback to the transmitter using a standard-compliant mechanism. To characterize MIMO link performance and capture channel conditions, we develop a new, practical link metric called *diffSNR*, which we then use in ARAMIS. *diffSNR* provides a good balance between implementability and accuracy as input to the link predictor. Through testing in a variety of environments, we show our

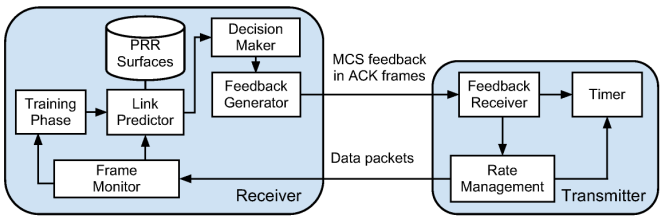


Fig. 1. Block diagram of ARAMIS.

link predictor estimates link quality with an accuracy of at least 95.5%.

We implemented ARAMIS and evaluated it on a 15 node testbed. We compare ARAMIS to leading RA solutions for 802.11n, namely Ath9k [7], Minstrel HT [6], and RAMAS [4]. We evaluate the solutions under various scenarios, including interference, mobility, and hidden nodes. We demonstrate that ARAMIS is robust, consistently performs well and outperforms existing solutions, with up to a 2.87 fold increase in throughput compared to its best competitor, RAMAS, and a 10 fold increase compared to Ath9k.

II. Overview of ARAMIS

ARAMIS is a closed-loop RA solution for 802.11n MIMO environments. In the design of ARAMIS, we identify three important, high-level components. The first component is a link metric that can be used to accurately characterize MIMO link performance. An existing MIMO link metric, called *effectiveSNR*, is one option [2]. However, *effectiveSNR* is built using CSI, and CSI is supported by few devices due to its costliness [3]. We therefore choose to develop a new, practical, MIMO link metric for systems where CSI is not available. We call this new metric *diffSNR*, and it provides a good balance between implementability and accuracy.

The second component is a mechanism that can accurately predict the Packet Reception Rate (PRR) of a link for any MCS and bandwidth combination, which we refer to as the *link predictor*. To predict PRR, the *link predictor* uses PRR performance models from the adopted link metric. The *link predictor* and link metric together form the backbone of the third and main design component, the *rate selector*. Based on current channel conditions which are determined using the link metric, and the corresponding PRR values computed using the *link predictor*, the *rate selector* finds the best operating rate and bandwidth with high accuracy. Since ARAMIS is a closed-loop RA solution, the *rate selector* also needs to implement a standard-compliant feedback mechanism.

Fig. 1 depicts the specific components in the design of ARAMIS and the corresponding communication flow between a transmitter and receiver pair. The primary functionality of ARAMIS is implemented at the receiver.

The first interface into ARAMIS's *rate selector* is the *Frame Monitor*. The *Frame Monitor* maintains updated information on channel conditions by measuring the link metric from existing data traffic. Current channel status information is used as input to the *Link Predictor*, which estimates the PRR of the link for all supported MCS and bandwidth combinations. The *Link Predictor* is measurement-based, and therefore must first access data storage to obtain the measured PRR values. To improve the accuracy of predictions, the *Link Predictor* goes through a *Training Phase* that corrects errors in predicted PRR values in real-time. The *Decision Maker* then takes the PRR predictions from the *Link Predictor*, and based on some performance model, selects the best operating point. Using a standard-compliant mechanism, the *Feedback Generator* encapsulates information on the best possible rate in ACK frames to be sent to the transmitter. However, in the case when the feedback is not received, for example due to lack of active traffic, a backup *Timer* is implemented at the transmitter to reset the operating rate.

In the following sections, we describe the main features of ARAMIS, namely the adopted link metric, the *link predictor*, and the *rate selector*.

III. 802.11n MIMO Link Metrics

We are first motivated by the need for a new metric by identifying the limitations of a commonly used and accessible link metric, RSSI (Received Signal Strength Indicator), in predicting link quality in 802.11n MIMO environments. We measure link quality or performance in terms of Packet Error Rate (PER, where: PER $= 1 -$ PRR). We then present *diffSNR* and examine how it can be used together with RSSI to accurately reflect performance.

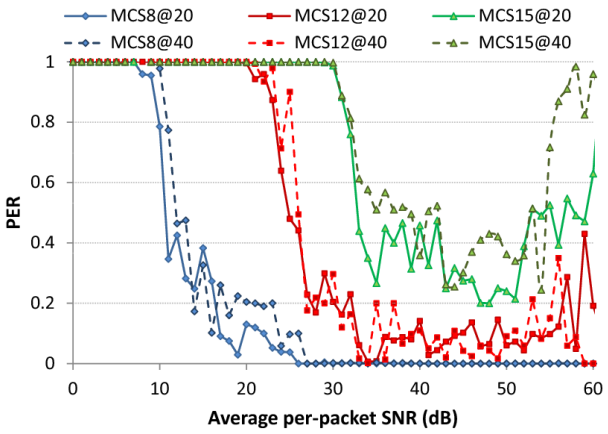
For legibility, we present a subset of our results that best reflects and is representative of patterns in the behavior of RSSI and *diffSNR*. We conduct all experiments for both 20MHz and 40MHz channels, and we discuss our observations for three MCS indices that cover robust (MCS 8), intermediate (MCS 12), and aggressive (MCS 15) PHY rates.

A. The Limitation of RSSI

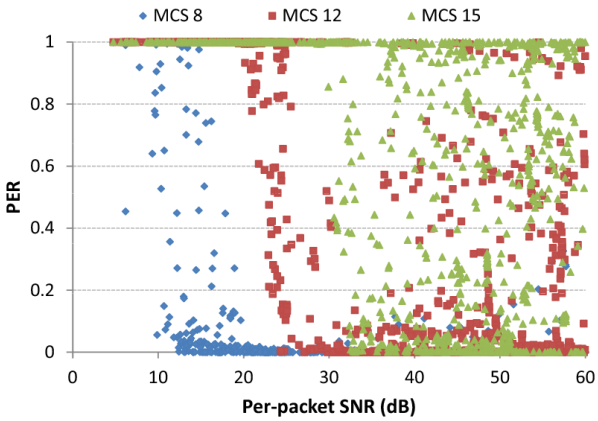
RSSI, defined as the signal to noise ratio (SNR) in dB, has traditionally been used to represent the quality of a link [10]. The existing models that map RSSI to performance show that a link's PER is 1.0 for sufficiently low RSSI and then steeply drops to 0.0 as RSSI increases beyond a threshold value [11].

Fig. 2(a) plots PER vs SNR averaged over 50 links in our testbed for two simultaneous streams. As RSSI increases, we expect PER to drop since the receiver can better decode the received signal. The plot, however, shows that this is not necessarily the case, particularly for aggressive modulation schemes with spatial multiplexing (MCS 12 and 15). PER does not converge to 0 for high SNR values and surprisingly, performance degrades for SNR $>$ 55dB.

There are two explanations to this behavior. High SNR values are achieved when the output power is high and/or when the propagation losses are low due to the close proximity of the transmitter/receiver pair in the absence of obstacles. The combination of OFDM and high order amplitude modulations (such as 64-QAM used in MCS 13 to 15) is prone to high peak-to-average ratios: high peaks cause the power amplifiers to move toward saturation [12], exhibiting non-linear behavior that produces inter-modulation distortion. The other explanation is that the presence of a dominant path between a transmitter/receiver pair, such as when the nodes are in direct



(a) Average over testbed links.



(b) Per-link measurements (20MHz channel).

Fig. 2. PER vs average per-packet SNR for two transmit streams.

line-of-sight, increases the Rician K-factor and the channel becomes increasingly correlated in space. This hampers the utilization of spatial multiplexing [13].

Fig. 2(a) also shows that SNR is a poor indicator of link quality for different channel widths. For the same SNR, a 40MHz channel suffers a higher PER. Wider transmissions are more likely to suffer from frequency selective fading, which causes SNR variations across the OFDM subcarriers, and PER is dominated by the lower SNR carriers. A 40MHz channel, therefore, not only requires a stronger transmission power to achieve the same SNR [14] but also a higher SNR to provide the same PER.

Fig. 2(b) plots per-link PER vs SNR for all testbed links. It shows the impact of RSSI on PER is unpredictable across different links. This result is consistent with previous work [15], [2]. The transition region between a good quality (PER ≈ 0) and a lossy (PER ≈ 1) link is wide, which means there is a wide range of SNR values where the performance of a link is uncertain. For MCS 8, the transition region is 10dB. For more aggressive PHY rates (e.g. MCS ≥ 12), it is as wide as 35dB.

B. Differential SNR (diffSNR)

It is clear that RSSI alone does not accurately capture the factors that cause the variability in 802.11 channels. Frequency selectivity due to multipath is one major factor whose effects are only captured using OFDM per-subcarrier SNR information [2]. Antenna correlation, or spatial selectivity, is another factor [16]. Both factors, however, require costly CSI which is supported by only very few devices [3].[1] For devices that do not support CSI, we develop a practical metric, called *diffSNR*, by using the channel metrics available to us in all commodity MIMO devices. We now show how we can use *diffSNR* to accurately reflect channel quality in 802.11n networks.

Multipath propagation in wireless environments produces constructive and destructive interference at the receiving antennas [17]. The resulting signal combination varies at different locations, a concept referred to as spatial selectivity. MIMO systems take advantage of these multipath phenomena to improve performance.

When received signals combine destructively in a process called *selective fading*, SNR can degrade and will reliably indicate a lossy link. Since per-packet SNR is the linear sum of all per-antenna measurements, if only a portion of the antennas experience fading, the reported SNR may be high even though the link could be lossy. Reported SNR does not reflect the extent of selective fading. We therefore argue that knowledge of the SNR combined with the per-antenna SNR provides us with some added insight, which can be used to predict the link performance with greater accuracy. We henceforth define the difference between the best and the worst SNR at any of the receiver's antennas as *diffSNR*.

In Fig. 3, we take a closer look at the real-time evolution of RSSI and *diffSNR* for a given link. A peak in *diffSNR* can occur when RSSI increases and one or more antennas receive constructive interference. However, we observe that with a higher certainty of 80%, *diffSNR* peaks are caused by some of the antennas suffering from fading.

Our experiments show that *diffSNR* does not depend significantly on the transmitter's output power, the MCS used, or the channel width. In fact, *diffSNR* shows a clear dependency on the environment (factors such as rich scattering, dynamic/static positioning, line-of-sight, and obstacles). We find that a more dynamic environment (during office hours) is reflected in a wider dispersion of the measured *diffSNR*, while a static scenario (night-time) exhibits fewer variations.

Given the predictable behavior of *diffSNR* and its correlation to RSSI, we next examine the implications of the (SNR, *diffSNR*) relationship and how it can be used to determine link quality or performance in terms of PER.

C. diffSNR and Packet Error Rate (PER)

For links with similar RSSI, we find that *diffSNR* can be used to characterize their performance differences. We illustrate this behavior in Fig. 4 using three representative links. We evaluate their PER vs SNR relationships using spatial multiplexing (MCS 12 and 15) and a 40MHz channel.

[1]Channel State Information (CSI) describes the current channel conditions with fine granularity. It consists of the attenuation and phase shift for each spatial stream to every receive antenna, for every OFDM subcarrier (52 subcarriers for 802.11n). Measuring a complete and timely CSI for all possible MIMO channel configurations incurs excessive sampling overhead [3].

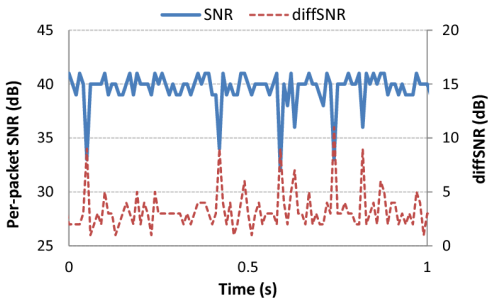


Fig. 3. Real-time evolution of per-packet SNR and *diffSNR*.

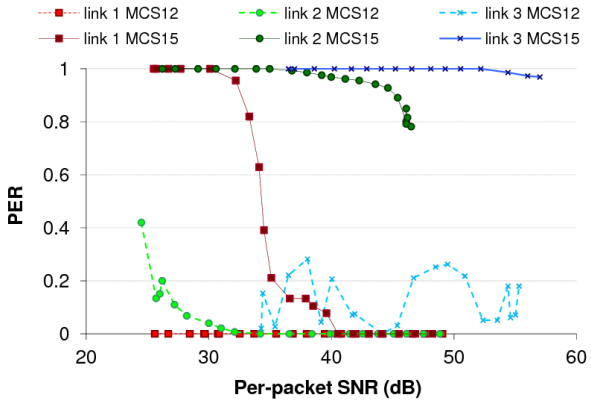


Fig. 4. PER vs per-packet SNR for three links (40MHz channel).

Link 1 successfully transmits packets (PER $<$ 0.02) using MCS 12; for MCS 15, there is a clear transition around 34dB SNR. Link 2 has similar RSSI values and exhibits a clear transition for MCS 12 (SNR $>$ 25dB), yet remains lossy at MCS 15 until SNR $>$ 44dB. The difference between Link 1 and 2 can be explained with *diffSNR*: for Link 1, we measure an average *diffSNR* of 1.82dB, with a standard deviation of 0.30, while for Link 2, the average *diffSNR* is 9.46dB, with a standard deviation of 0.37. Link 3 displays the worst performance, showing an average *diffSNR* of 13.41dB. This link does not exhibit a clear transition for MCS 12 and never works for MCS 15. We can explain this behavior with the dispersion of its measured *diffSNR* values with a standard deviation of 0.97.

Our analysis reveals the dependency of performance on RSSI and *diffSNR* together. Since robust modulations are less affected by fading, variations in *diffSNR* will be more clearly reflected on the performance of aggressive modulations. Similarly, *diffSNR* variations will have little impact on links with high SNR, but this impact will increase as SNR decreases. We believe this dependency is intriguing and leave further analysis of potential correlations to spatial or frequency selectivity to future work. Fig. 5 is a representative graph that effectively exemplifies this dependency.

Fig. 5 plots the measured PRR as a function of average per-packet SNR and *diffSNR* for a given MCS and bandwidth combination. We note that the dataset in Fig. 5 combines values obtained from real measurements with interpolated points. Fig. 5 shows that the PRR(SNR, *diffSNR*) relationship yields well-behaved surfaces that allow us to predict the PRR of a link for a given MCS and bandwidth. For example, with an SNR of 32dB, a link with *diffSNR* below 5dB performs well,

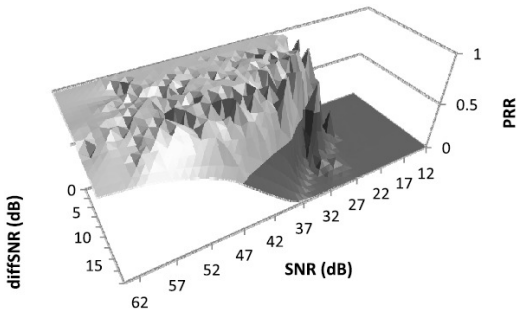


Fig. 5. PRR(SNR,*diffSNR*) surface for MCS 7 and 20MHz channel.

however with a *diffSNR* above 10dB, the PRR of the link is almost 0. Next, we describe how we utilize these surfaces in the design of our link predictor.

IV. A Measurement-Based Link Predictor

A link predictor accurately estimates the PRR of a link for all MCS and bandwidth combinations. We now describe the methodology we use to build such a predictor, and demonstrate how it accurately predicts PRR. In case of errors, we introduce a low-overhead training mechanism to improve accuracy.

A. Methodology

We design our predictor as the synthesis of the measurement-based PRR(SNR, *diffSNR*) surfaces for all MCS and bandwidth combinations. SNR and *diffSNR* measurements, along with the operating MCS and bandwidth of a link, are the input parameters the predictor uses to identify the corresponding expected PER for that link. Our testbed provides us with SNR data for the control channel and when channel bonding, the extended channel. Predictions for 20MHz links can be made from measurements under 40MHz links, but not vice versa [9]. Therefore, our predictor builds separate PRR surfaces for both 40MHz and 20MHz channels for each MCS.

To gather sufficient data to build PRR(SNR, *diffSNR*) surfaces for each MCS and bandwidth combination, we measure PRR(SNR, *diffSNR*) over all 50 testbed links while varying the transmit power from 0dBm to the maximum allowed power. The remaining values that are not measured are interpolated.

Intuitively, PRR depends on the packet length. Hence, packet length should be accounted for to predict PRR. The PRR for any given packet size can be roughly estimated from the PRR we measure for 1,000 byte packet size using the equation: $P_x = \left(\sqrt[L_{1000}]{P_{1000}}\right)^{L_x}$, where P_x is the PRR for a packet of length x bytes, and L_x is the length in bits of a frame carrying an x byte packet. For different packet sizes, our measurements show that the transition regions for links from high quality to lossy do not exhibit a noticeable difference.

B. Prediction Accuracy

The link predictor is a matrix, with dimensions defined by the number of supported MCS, bandwidths, and the range of expected SNR and *diffSNR* values. To evaluate our predictor, we build two $6 \times 70 \times 20$ matrices for MCS 0, 4, 7, 8, 12 and 15, with SNR values from 0 to 69dB and *diffSNR* values from 0

to 19dB, with 1dB precision.[2] Our complete predictor consists of two $16 \times 70 \times 20$ matrices, and is used in future sections. The reduced matrix consists of 40% of measured values (the remaining 60% are interpolated). We show that interpolation has no significant impact on the prediction accuracy.

We evaluate the accuracy of our measurement-based link predictor by comparing the predicted PER values against the measured values for two different groups of transmitter/receiver pairs. The first group consists of nodes located in the same environment where the data for the predictor was collected. The second group consists of a set of laptops placed in two different off-campus small office/home office environments as well as in an outdoor environment, on a rooftop free of obstacles with direct LoS between nodes placed 20m apart. We include this second group to evaluate the utility and accuracy of the proposed predictor in unfamiliar and dissimilar environments.

For the first group of nodes located in a familiar environment, the average absolute error in PER predictions, computed as the difference between the measured PER and the predicted PER, is only 4.8%. This error rate increases for high order modulations (up to 11% for MCS 15) since these modulations show a higher degree of uncertainty. Although the absolute error may be relatively high for some MCS indices, we reliably predict link feasibility with a 96.1% accuracy.[3] As for the second group of nodes in new environments, the average absolute error in PER predictions is 12% and the accuracy in feasibility predictions is 88.1%. These results show the importance of a calibration or training mechanism. To increase the prediction accuracy, we include the error of previous measurements in the new PER predictions such that:

$$PER_k^{m,B} = PER(m, B, SNR_k, diffSNR_k) + E_{k-1}^{m,B} \quad (1)$$

where $PER_k^{m,B}$ is the predicted PER for MCS m and bandwidth B; SNR_k, and $diffSNR_k$ are the currently measured RSSI and *diffSNR* values; and $PER(w, x, y, z)$ returns a PER value from the predictor using the input parameters. Finally, $E_{k-1}^{m,B}$ is the error in previous predictions for the same MCS and bandwidth, where $0 \leq E_{k-1}^{m,B} \leq 1$. We track the error by computing $E_k^{m,B}$ as an exponential moving average with α = 0.9. Our α is large to give more weight to recent error samples, since the mean error in PER predictions is close to 0.

We re-evaluate our results in the new environments using Equation 1. The average absolute error in PER predictions now falls below 5.8%, and link feasibility predictions improve to a 95.5% accuracy rate.

Not surprisingly, the performance of aggressive modulations is more difficult to predict, and this is translated into lower feasibility prediction rates and higher PER prediction errors. Also, PER prediction errors increase when spatial multiplexing is used (4.6% average absolute error for one stream vs 6.9% for two streams). Finally, the PER of a 20MHz channel can be predicted with slightly greater accuracy than a 40MHz channel (96.0% accuracy in link feasibility predictions for 20MHz channels vs 95.1% for 40MHz).

[2]The distribution of *diffSNR* in all tested environments lie below 19dB.

[3]We consider a link feasible for a given MCS if PER < 0.5.

Algorithm 1 ARAMIS(SNR, *diffSNR*)

Output: 1) MCS m; 2) Channel width B;

```
if newPacket = true then
    (SNR_avg, diffSNR_avg) ← update-moving-average(SNR, diffSNR)
    // If there is a change in channel conditions
    if exception(SNR, diffSNR) = true then
        (m,B) ← decision-maker() ← link-predictor(SNR_avg,diffSNR_avg)
    end if
end if
```

Recall that we interpolate to fill the gaps in a PER(SNR, *diffSNR*) surface. We observe that regardless of whether the predictions come from interpolated or measured values, the predictor accuracy remains the same. For the measurements conducted in unfamiliar environments, we predict 71% of the indoor links from measured values while the remaining 29% are interpolated. For outdoor links, the proportion is 61/39%. For the familiar environment, interpolated values are not used.

We have built a mechanism capable of accurately predicting PRR for all MCS and bandwidth combinations for a given link (SNR, *diffSNR*). This accuracy enables us to include our predictor in a rate selection mechanism.

V. Rate Selector

The rate selector is the final and main design component of ARAMIS. An effective rate selector in a closed-loop, 802.11 RA model identifies changes in environment conditions and responds with the appropriate rate using a standard-compliant feedback method. To achieve these goals, we now describe how we combine our knowledge of our link metric, in this case *diffSNR*, and the link predictor in the design of an effective rate selector. We use the terminology illustrated in Fig. 1.

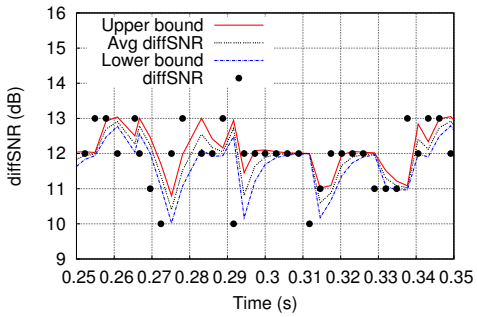
A. Frame Monitor

The first step of a rate selector is to identify changes in channel conditions. This step is necessary to determine when an alternative rate might be appropriate. We have verified the accuracy of (SNR, *diffSNR*) in predicting link quality. We now describe how we monitor the behavior of per-packet (SNR, *diffSNR*) in real-time, using existing active traffic, to identify changes in channel conditions.

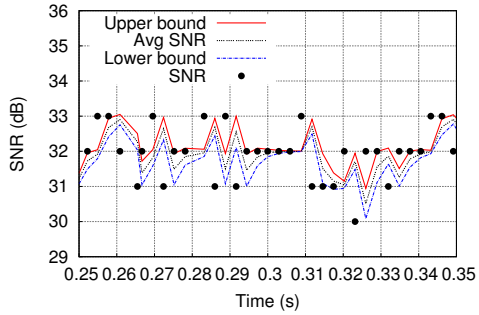
Fig. 3 depicts the evolution in per-packet (SNR, *diffSNR*) over time for a given link. Over a short period of time, (SNR, *diffSNR*) can fluctuate rapidly. To identify when changes in (SNR, *diffSNR*) could reflect a change in channel conditions, we apply an exponentially weighted moving average approach. ARAMIS stores (SNR, *diffSNR*) for every packet received and computes their moving average (SNR_{avg}, $diffSNR_{avg}$). We maintain moving averages not only for the average (SNR, *diffSNR*) values, but also for their standard deviation (SNR_{sd}, $diffSNR_{sd}$). ARAMIS initiates lookups to the link predictor if the current (SNR, *diffSNR*) lies outside of the range specified by $SNR_{avg} \pm SNR_{sd}$. The same conditions apply for *diffSNR*.

B. Decision Maker

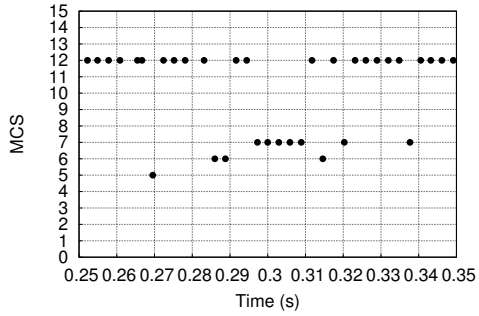
The rate selector uses a link's current channel conditions, reflected through the link metric, as input arguments to the link predictor, in this case using (SNR_{avg}, $diffSNR_{avg}$). The



(a) Variation of *diffSNR* with time.



(b) Variation of SNR with time.



(c) Selected rate based on above, correlated *diffSNR*, SNR values.

Fig. 6. Depiction of ARAMIS measurements.

predictor provides accurate PRR estimates for all supported MCS and bandwidths for that link. The role of the *Decision Maker* is to use this information to select the MCS and bandwidth configuration that yields the highest throughput. One model would be to select the configuration with the highest expected throughput. The computation of the expected throughput, however, requires a foreknowledge of the packet size implemented at the transmitter [9], which is not available at the receiver. Furthermore, this approach adds significant overhead to the computation of the appropriate rate.

We adopt a simple yet effective approach. Our model selects the MCS and bandwidth combination with the highest PHY bitrate from a reduced set of combinations whose predicted PRR is above a threshold. By adjusting this threshold, ARAMIS has the flexibility to adapt to environments with varying error tolerances.

Fig. 6 demonstrates the behavior of ARAMIS in real-time, as described in Algorithm 1. In Fig. 6(a) and (b), we plot the instantaneous values, moving averages, and upper and lower bounds for our link metrics, both SNR and *diffSNR*. Fig. 6(c) depicts how ARAMIS changes MCS on a per-packet basis based on the correlated (SNR, *diffSNR*) values,

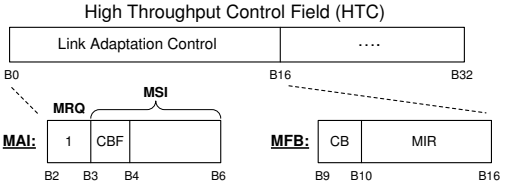


Fig. 7. 802.11n compliant MCS feedback system.

TABLE I. HTC SUBFIELDS THAT SUPPORT RECEIVER-BASED RA.

MRQ	MCS feedback request
MSI	MRQ sequence identifier
MFB	MCS feedback
CBF*	AP Channel bonding friendly
MIR*	Client MCS index request
CW*	Client channel width request

*: Bits allocated to support channel width feedback

where ARAMIS selects the MCS with the highest bitrate from those MCS that achieve a PRR above a given threshold for the current channel conditions. The corresponding bandwidth graph is not shown as ARAMIS always opts for a 40MHz channel in this run.

C. Training Phase

To improve the accuracy of predicted PRR values for all MCS and bandwidth combinations, a training mechanism is performed on-the-fly using the statistics of received frames. ARAMIS measures PRR by dividing the number of received frames with the Retry flag set to 0, by the total number of frames sent, the latter computed using frame sequence numbers. If aggregation is enabled, more precise PRR estimation could be provided by inspecting the bitmap field present in the Block ACK. ARAMIS then uses this measured PRR to update $E_k^{m,B}$ values in Equation 1.

D. Feedback Generator

So far, we have discussed how ARAMIS identifies an appropriate rate given the current channel conditions. This rate, however, should be sent as feedback to the transmitter using a standard-compliant mechanism. To fully exploit variations in a MIMO channel, the 802.11n standard supports MCS feedback (MFB) in link adaptation [8]. MFB is a subfield of the *HT Control field* (HTC). HTC is a 4B optional field added to control packets (such as ACKs and Block ACKs).

Fig. 7 shows the HTC field with its corresponding link adaptation control field, where the subfields are described in Table I. We propose utilizing the unused fields and creating subfields that control bandwidth feedback. These added subfields allow ARAMIS to operate in conjunction with a channel management solution, where the CBF field set by the AP defines the supported bandwidth in the given WLAN. For example, if CBF is set to 1, the client can request to operate on both a 20MHz and 40MHz channel, which it specifies in the CW subfield, and if CBF is set to 0, the client only operates on a 20MHz channel. It is worth noting that the emerging 802.11ac standard supports such a client-based bandwidth adaptation mechanism, given the maximum supported bandwidth at the AP.

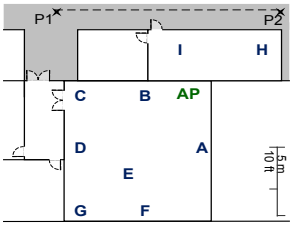


Fig. 8. Floorplan of our testbed environment.

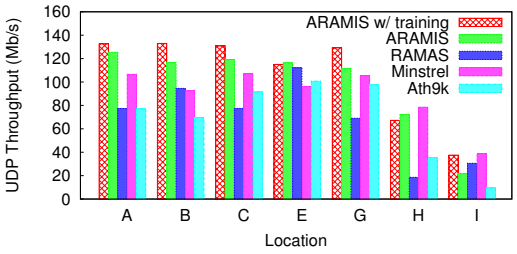


Fig. 9. Algorithm performance in an interference-free environment.

E. Timer

A transmitter stops receiving feedback when the ARAMIS receiver does not receive transmitter frames. This can happen for two reasons. First, channel conditions at any given time could change drastically such that the PRR for the PHY configuration in use suddenly drops to 0. Second, the transmitter may not have traffic to send. In both cases, the communication could be set at the wrong configuration with outdated information, since the transmitter is not receiving feedback to identify the appropriate MCS and bandwidth. This can lead to performance degradation. To mitigate this problem, we use a timer at the transmitter, whereby if feedback packets are not received before the timer expires, the MCS is set back to a reliable rate, MCS 8, then MCS 0 after a consecutive timeout, at the same bandwidth. Our results show that ARAMIS's per-packet rate adaptation is able to rapidly recover from this MCS reset.

VI. Performance Evaluation

We now evaluate ARAMIS in a testbed implementation under various scenarios, including interference, mobility, and hidden nodes. The goal of our evaluation is to demonstrate the efficacy of ARAMIS in accurately responding to channel conditions compared to other popular and leading 802.11n RA solutions. We measure performance in terms of achieved throughput. We demonstrate that ARAMIS is robust, consistently performs well and outperforms existing solutions.

A. Testbed Environment

The evaluation environment is built over our platform that consists of 15 laptops deployed in both an open office and semi-open office environment. Each laptop is equipped with an 802.11n 2×3 MIMO PC card with an Atheros AR5416/AR5133 2.4/5GHz chipset. The AR5416 baseband and MAC processor allow MCS indices 0 to 15. Each laptop runs the Linux 2.6.32 kernel, where the device driver is based on the Atheros Ath9k that supports 802.11n [7]. We run our experiments on the 5GHz frequency range and verify the lack of background traffic with a spectrum analyzer.

We compare the performance of ARAMIS to that of two widely used open source 802.11n RA solutions, Ath9k [7] and Minstrel HT [6], and RAMAS [4], which was recently shown to be one of the best performing 802.11n RA solutions. We run RAMAS using the implementation made available by its authors. RAMAS is a credit-based system that divides the features of 802.11n RA into two groups: a modulation group and a group that includes the number of streams and bandwidth. Each group follows a different credit system and is adapted independently of the other. Minstrel HT and Ath9k both use random sampling to find the best MCS. Minstrel HT, however, includes MCS with different bandwidths in its sampling group. Ath9k does not have a mechanism for enabling channel bonding, and to ensure a fair comparison, we set Ath9k's bandwidth to 40MHz to allow it to exploit higher data rates. Ath9k switches to a 20MHz channel when the PER is high. Other schemes select channel width based on their algorithm, and independently of the rate.

We evaluate the RA algorithms in a wide variety of scenarios, including interference and mobility. We fix transmit power to $11dBm$ and enable packet aggregation. We measure UDP throughput and PER, and average the results over 5 runs. The floorplan of our semi-open office, experimental environment is shown in Fig. 8, where the letters represent node locations.

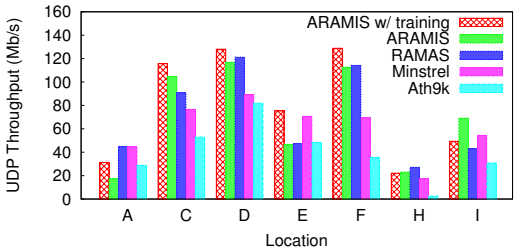
In our implementation of ARAMIS, we faced restrictions where the available chipset code does not support enabling an HTC field for 802.11n feedback. We mitigate this issue by implementing netlink sockets and transmitting packets with the HTC field over the wire from the receiver to the transmitter driver code. The overhead of user-space-kernel communications, though minimal, often lead to delayed rate feedback receptions which trigger timeouts that mimic ACK packet losses. Moreover, the devices do not provide open access to the hardware generated Block-ACK at the receiver. This leads to inaccurate PER measurements, which reduces the precision of the ARAMIS training mechanism, explained in Section V-C, and the accuracy of measured $E_k^{m,B}$ samples.

B. Testbed Results

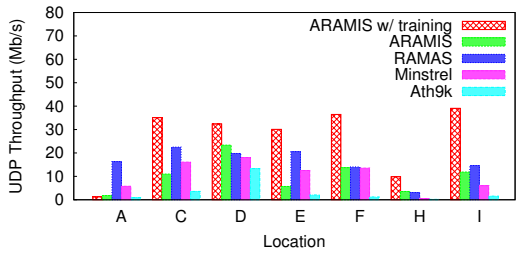
Fig. 9 and 10 show that ARAMIS consistently outperforms other algorithms in all test cases, with up to a 2 fold throughput increase in interference-free environments, a 10 fold increase in interference conditions, and a 25% increase in mobile environments.

Interference-free: To assess how well each algorithm handles random channel loss, for example due to shadowing or multipath, Fig. 9 shows the performance in an interference-free environment at seven different locations. Even without the training mechanism, ARAMIS outperforms other algorithms with throughput gains of up to 26% over Minstrel HT, 124% over Ath9k, and 287% over RAMAS. Note that our results for RAMAS are somewhat different from those reported [4], since they were obtained in different scenarios. RAMAS was previously evaluated only on the 2.4GHz frequency range, which significantly limits the performance benefits of 802.11n features [18], [5].

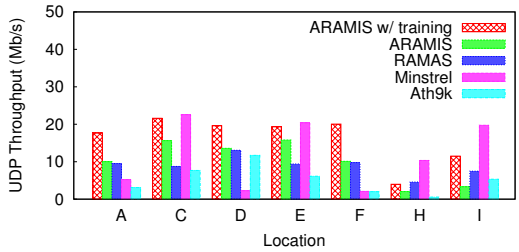
RAMAS leads to an average PER of 11% and a maximum of 20%. The credit scheme it uses to adapt the number



(a) Adjacent 40MHz interferer.



(b) Adjacent 20MHz interferer.



(c) Channel sharing with a 20MHz interferer.

Fig. 10. Algorithm performance under interference conditions.

of streams is conservative, while the scheme to adapt the modulation and coding is aggressive. This mismatch causes RAMAS to often operate at sub-optimal rates with high modulations and single stream (e.g., MCS 7), which leads to high PER and reduced performance. Ath9k and Minstrel HT's random sampling incurs high overhead that results in poor performance. Ath9k also assumes PER monotonically increases with rate, which causes it to often ignore suitable high rates.

ARAMIS relies on our link predictor for rate selection and hence does not require random sampling. Its link prediction accuracy and ability to adapt MCS and bandwidth on a per-packet basis maximize opportunities to exploit more aggressive rates without sacrificing PER. We observe an average PER between 4 and 6%. ARAMIS is therefore suitable for low error tolerance applications, such as online gaming and bulk file transfers.

Interference: We now assess how the algorithms perform under interference from signal leakage, hidden nodes, and channel sharing.

Signal leakage is produced by transmissions on adjacent channels and can result in collisions similar to the hidden node problem. We evaluate how the algorithms react to interference due to leakage with varying interferer bandwidth, as we discovered that the impact of leakage varies according to channel width [9]. Fig. 10(a) presents results with an interfering link that operates on an adjacent 40MHz channel. Fig. 10(b) presents results for an adjacent 20MHz interferer.

Ath9k and Minstrel HT respond frequently and rapidly to interference by reducing the rate. Reducing the rate exacerbates the impact of leakage; frame transmission time increases and so do the opportunities for collisions. Similarly, RAMAS responds to channel disturbances by first reducing the number of streams, thus reducing the transmission rate.

With signal leakage, the reported SNR may be low and collisions could be interpreted as wireless losses. ARAMIS's PRR predictions hence may not match the measured values from the training mechanism. When the prediction error $E_k^{m,B}$ exceeds a given threshold, which we set to 0.2 based on our experiments, ARAMIS interprets that there is a collision problem and limits the influence of the training mechanism; it sets $E_k^{m,B}$ to the maximum allowed value, thus maintains transmissions at suitable high rates. For an adjacent 40MHz interferer shown in Fig. 10(a), we improve the throughput by up to 60% over RAMAS, 85% over Minstrel HT, and 782% over Ath9k. For an adjacent 20MHz interferer shown in Fig. 10(b), the improvement is 220% over RAMAS, 412% over Minstrel HT, and 1908% over Ath9k. We observe greater performance improvements with an adjacent 20MHz interferer, since it is the more harmful configuration [9], and ARAMIS mitigates this interference.

We also investigate the channel sharing scenario with an interferer on a 20MHz channel. This scenario has been shown to create worse fairness issues than a 40MHz co-channel interferer whereby the slower 20MHz channel occupies the medium for longer periods of time [9]. In Fig. 10(c), we evaluate how well the algorithms perform under such conditions.

The presence of co-channel interference slightly increases collision probability, and thus $E_k^{m,B}$ increases, but remains under its maximum allowed value. As a result, the probability of using high rates is slightly reduced and Minstrel HT matches ARAMIS's performance in some locations since those collisions seldom affect Minstrel's random probing mechanism.

The timely detection and adaptation to the channel conditions give ARAMIS an advantage over other algorithms, and this advantage is also evident in channel sharing conditions. At all locations, ARAMIS maintains the high order rates, thus exploiting its available channel time. ARAMIS improves the throughput by up to 76% over RAMAS, 251% over Minstrel HT, and 366% over Ath9k.

Mobility: We create a mobility scenario to evaluate the responsiveness of ARAMIS to rapidly changing channel conditions. With a static AP placed at Location I, we move the client on a trolley through the adjacent corridor from the indicated P_1 to P_2 at an approximate speed of 5km/h. ARAMIS achieves throughput of 80.27Mb/s and improves the throughput by 25% over RAMAS, 7% over Minstrel HT, and 15% over Ath9k. The small differences in throughput in this case may be due to the fact that ARAMIS is close to the capacity of this channel, which is low.

Note that our ARAMIS implementation had to overcome significant limitations due to hardware restrictions. These limitations reduce the potential performance benefits of ARAMIS. Hence, we believe that the ARAMIS performance we observe from our experiment is a lower bound.

VII. Related Work

Wireless Link Metrics: A significant body of work has proposed methods to characterize link performance. RSSI, which is the most accessible link metric, has traditionally been used to identify a link's maximum expected throughput. Recent studies [11], [2], [15] have shown that RSSI is an unreliable metric to accurately predict performance. The utilization of *effective SNR* [2] is proposed, where the metric is generated using CSI feedback to accurately reflect link conditions in OFDM environments. However, complete CSI information could be costly to obtain and store [3] and is therefore not supported by all 802.11n devices.

Rate Adaptation: Rate adaptation has been one of the most popular research topics in WLANs [10], [1], [19] and new algorithms for 802.11n networks have been proposed [2], [5], [20], [21]. Although solutions for legacy clients have been effective, they fall short when applied in 802.11n OFDM-MIMO settings [5]. Existing 802.11n solutions require either costly CSI [22], [2] or some form of a guided search (e.g., by probing candidate rates) to determine the best operating rate [5], which is inefficient when the search space is large. Other algorithms for MIMO environments do not consider other 802.11n features, such as channel bonding [23], or consider alternative energy efficiency goals [21].

VIII. Conclusion and Future Work

The 802.11n standard has been touted as a new revolution in Wi-Fi technology, in part because of the number of new mechanisms that enable a multifold increase in transmission speeds relative to 802.11a/b/g. What is clear, however, is that while 802.11n has the theoretical ability to attain wireless data rates as high as a few hundred Mbps, it is only through intelligent and adaptive transmission strategies that such throughputs have a hope of being achieved. Among the most crucial questions for accessing the medium is the mechanism to select an appropriate data rate and bandwidth combination for transmission that is correctly responsive to changes in signal quality.

We have introduced ARAMIS, a closed-loop RA solution that jointly adapts rate and bandwidth. Through implementation of our solution, we have demonstrated that ARAMIS obtains impressive performance gains over leading 802.11n rate adaptation contenders, including up to a 10 fold increase in throughput. We believe that ARAMIS is a critical component of a fully adaptive, intelligent 802.11n management system that dynamically optimizes 802.11n performance in response to changing channel conditions commonly present in operational wireless networks. Our solution design can also be applied in the context of the emerging 802.11ac standard, where MCS and channel width selection are faced with further challenges.

IX. Acknowledgments

This work is partially supported by the National Science Foundation under Grant No. 1032981. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work is also partially supported by the Spanish Government through project TEC2009-11453 and Programa Nacional de Movilidad de Recursos Humanos del Plan Nacional de I-D+i 2008-2011.

References

[1] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *ACM MobiCom*, Sept. 2006.

[2] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," in *ACM SigComm*, Aug. 2010.

[3] R. Crepaldi, J. Lee, R. Etkin, S.-J. Lee, and R. H. Kravets, "CSI-SF: Estimating wireless channel state using CSI sampling and fusion," in *IEEE Infocom*, Mar. 2012.

[4] D. Nguyen and J. Garcia-Luna-Aceves, "A practical approach to rate adaptation for multi-antenna systems," in *IEEE ICNP*, Oct. 2011.

[5] I. Pefkianakis, Y. Hu, S. H. Wong, H. Yang, and S. Lu, "MIMO rate adaptation in 802.11n wireless networks," in *ACM MobiCom*, Sept. 2010.

[6] Minstrel HT Linux Wireless, "http://linuxwireless.org/en/developers".

[7] M. Wong, J. M. Gilbert, and C. H. Barratt, *Wireless LAN using RSSI and BER parameters for transmission rate adaptation*, US patent 7,369,510, 2008.

[8] "IEEE 802.11n-2009 Amendment 5: Enhancements for Higher Throughput," IEEE-SA, Oct. 2009.

[9] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, and K. Almeroth, "The impact of channel bonding on 802.11n network management," in *ACM CoNEXT*, Dec. 2011.

[10] J. Camp and E. Knightly, "Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation," in *ACM MobiCom*, Sept. 2008.

[11] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *ACM SigComm*, Aug. 2004.

[12] A. Amini, D. Lu, and C. Edelman, "Effects of high peak-to-average ratio on 5GHz WLAN power amplifier," in *DesignCon*, Feb. 2002.

[13] H. Yang, "A road to future broadband wireless access: MIMO-OFDM-based air interface," *IEEE Communications Magazine*, vol. 43, no. 1, pp. 53–60, Jan. 2005.

[14] M. Y. Arslan, K. Pelechrinis, I. Broustis, S. V. Krishnamurthy, S. Addepalli, and K. Papagiannaki, "Auto-configuration of 802.11n WLANs," in *ACM CoNext*, Nov. 2010.

[15] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-based models of delivery and interference in static wireless networks," in *ACM SigComm*, Sept. 2006.

[16] D.-S. Shiu, G. Foschini, M. Gans, and J. Kahn, "Fading correlation and its effect on the capacity of multielement antenna systems," *IEEE Transactions on Communications*, vol. 48, no. 3, pp. 502–513, Mar. 2000.

[17] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[18] V. Shrivastava, S. Rayanchu, J. Yoonj, and S. Banerjee, "802.11n under the microscope," in *ACM IMC*, Oct. 2008.

[19] M. Vutukuru, H. Balakrishnan, and K. Jamieson, "Cross-layer wireless bit rate adaptation," in *ACM SigComm*, Aug. 2009.

[20] W. H. Xi, A. Munro, and M. Barton, "Link adaptation algorithm for the IEEE 802.11n MIMO system," in *Networking LNCS*, 2008.

[21] C.-Y. Li, C. Peng, S. Lu, and X. Wang, "Energy-based rate adaptation for 802.11n," in *ACM Mobicom*, Aug. 2012.

[22] F. Peng, J. Zhang, and W. E. Ryan, "Adaptive modulation and coding for IEEE 802.11n," in *IEEE WCNC*, Mar. 11–15, 2007.

[23] W. Kim, O. Khan, K. Truong, S.-H. Choi, R. Grant, H. Wright, K. Mandke, R. Daniels, R. Heath, and S. Nettles, "An experimental evaluation of rate adaptation for multi-antenna systems," in *IEEE Infocom*, Apr. 2009.