Computer Networks 83 (2015) 332–348

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A practical framework for 802.11 MIMO rate adaptation

Lara Deek[a,*], Eduard Garcia-Villegas[b], Elizabeth Belding[c], Sung-Ju Lee[d], Kevin Almeroth[c]

[a] *University of Illinois at Urbana-Champaign, United States*
[b] *UPC-BarcelonaTECH, Spain*
[c] *UC Santa Barbara, United States*
[d] *KAIST, South Korea*

ARTICLE INFO

Article history:
Received 1 December 2014
Received in revised form 26 February 2015
Accepted 23 March 2015
Available online 9 April 2015

Keywords:
IEEE 802.11n
Rate adaptation
Channel bonding
Measurement
Performance
Experimentation

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 optimizing transmission. 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, our contributions in this work are two-fold. First, we propose a practical link metric that accurately captures channel conditions in MIMO 802.11n environments, and we call this metric *diffSNR*. Using *diffSNR* captured from real testbed environments, we build performance models that accuractely predict link quality in 95.5% of test cases. Practicality and deployability are guaranteed with *diffSNR* as it can be measured on all off-the-shelf MIMO WiFi chipsets. Second, we propose ARAMIS (Agile Rate Adaptation for MIMO Systems), a standard-compliant, closed-loop RA solution that jointly adapts rate and bandwidth, and we utilize the *diffSNR*-based 802.11n performance models within ARAMIS's framework. ARAMIS adapts transmission rates on a per-packet basis; we believe it is the first closed-loop, 802.11 RA algorithm that simultaneously adapts rate and channel width. We have implemented ARAMIS with *diffSNR* 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.

1. 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* to improve signal strength, or multiple simultaneous streams using *spatial multiplexing* to increase the transmission rate.

Identifying a link metric that accurately characterizes and exploits 802.11n MIMO link performance is an

* Corresponding author.
E-mail addresses: laradeek@illinois.edu (L. Deek), eduardg@entel.upc.edu (E. Garcia-Villegas), ebelding@cs.ucsb.edu (E. Belding), sjlee@cs.kaist.ac.kr (S.-J. Lee), almeroth@cs.ucsb.edu (K. Almeroth).

http://dx.doi.org/10.1016/j.comnet.2015.03.015
1389-1286/© 2015 Elsevier B.V. All rights reserved.

important component of an effective RA solution. 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. However, complete CSI information is costly to obtain and store [1] and is therefore supported by very few 802.11n devices. Existing RA solutions adopt a practical approach and use a credit-based system [2] or rate sampling [3–5]. Instead of adapting the rate based on understanding the impact of environment conditions on 802.11n features, these solutions rely on certain heuristics 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.

To characterize MIMO link performance and capture channel conditions, particularly for the majority of systems where CSI is not available, our previous work developed a practical link metric called *diffSNR*, which provides a good balance between implementability and accuracy [6]. Similar to CSI-based metrics [7], *diffSNR* also provides the flexibility to predict performance for a given rate and channel width combination simultaneously. *diffSNR* is computed as the difference between the best and the worst SNR (Signal-to-Noise Ratio) at any of the receiver's antennas; it reveals insights on the nature of signal reception, i.e. whether signals combine constructively or destructively at the receiver's antennas. Our close analysis revealed the dependency of performance on *diffSNR*, and we exploit this relationship in the design of a measurement-driven link quality predictor. Through testing in a variety of environments, we showed that our *diffSNR*-based link predictor estimates link quality over all supported rate and bandwidth combinations with an accuracy of at least 95.5%.

A natural extension is to evaluate the application and impact of *diffSNR* on RA by implementing or incorporating *diffSNR* in the context of an effective 802.11n RA framework. There are two main approaches to RA one can adopt: an open-loop and a closed-loop approach. 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 [8]. A closed-loop RA is one in which the receiver's insight into the channel conditions contributes to determining the 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 (32 rates $\times$ 2 channel widths) and 802.11ac multiplies this number by four. 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 rate for the link [7].

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* [9]. 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. Therefore, the RA solution must adopt a link metric that accurately characterizes MIMO link performance. We propose such an RA solution, which we call ARAMIS (Agile Rate Adaptation for MIMO Systems) [6].

ARAMIS is a closed-loop, per-packet RA solution that simultaneously adapts both rate and channel width. In our previous work on channel bonding, we showed the importance of adapting bandwidth in 802.11n with RA to maximize performance [10,11]. 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. We use our proposed metric *diffSNR* to characterize MIMO link performance and capture channel conditions, as well as serve as input to the link predictor.

We implemented ARAMIS with *diffSNR* and evaluated it on a 15-node testbed [6]. We compared ARAMIS to leading RA solutions for 802.11n, namely Ath9k [5], Minstrel HT [4], and RAMAS [2]. We evaluated the solutions under various scenarios, including interference, mobility, and hidden nodes. We demonstrated that ARAMIS is robust, consistently performs well and outperforms existing solutions, with an average of 0.5-fold and up to a 2.87-fold increase in throughput compared to its best competitor, RAMAS, and an average of 3.85 and up to a 10-fold increase compared to Ath9k.

We further provide a more detailed evaluation of ARAMIS in two respects. First, we compare ARAMIS against an ideal solution. This comparison allows us to gauge how closely ARAMIS approximates a performance upper bound. To conduct this evaluation, we use a trace-driven simulation, where we implement both ARAMIS with *diffSNR* and an ideal solution. We find that ARAMIS closely approximates the ideal by taking advantage of per-packet processing. Per-packet RA enables quick and fine adjustments to varying channel conditions and allows the exploitation of narrow windows of higher bandwidth opportunities. Ideally, we would compare ARAMIS to an existing solution in literature that uses CSI to perform rate adaptation [7]. However, we are unable to implement CSI-based RA solutions, since our chipsets do not support CSI. We believe, however, that the ideal solution evaluated in the trace-driven simulations provides a more accurate upper bound, since it is implementation-independent.

Our second contribution to a detailed evaluation is of ARAMIS's link predictor that is built over *diffSNR*. This evaluation reveals that the link predictor exhibits relatively consistent behavior for different frame sizes and rates. Not surprisingly, the performance of aggressive modulations is more difficult to predict, and this is translated into higher Packet Reception Rate (PRR) prediction errors.

Furthermore, PRR prediction errors increase when spatial multiplexing is used. Notwithstanding, ARAMIS's link predictor maintains a reasonable level of accuracy, with an average absolute error in PRR predictions of 12%, and by adding a training mechanism, errors fall to 5.8%.

A final contribution of our work is a detailed evaluation of the performance insights SNR provides in 802.11n MIMO environments. Though the inaccuracies of SNR have been identified in prior work [7,12], we show that SNR measurements are still useful to provide a high-level assessment of the channel. Through our detailed evaluation of SNR behavior, we come to understand the impact of different 802.11n features, namely spatial multiplexing, spatial diversity, and channel bonding, on its performance.

This paper is organized as follows. We first evaluate the efficacy of SNR and our metric *diffSNR* in 802.11n MIMO environments in Section 2, where we also detail the implementation cost of CSI. We then present the application of *diffSNR* in the design of a link predictor in Section 3, and evaluate its prediction accuracy. Section 4 discusses our adopted ARAMIS rate selection algorithm and its components. We then evaluate ARAMIS with *diffSNR* and compare it to existing solutions under both a simulation and testbed environment in Section 5. Related areas of research are discussed in Section 6. Finally, we conclude in Section 7.

2. Metrics for MIMO links

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 dBm), and the cost of using full CSI (Channel State Information), when available. We then present *diffSNR* and examine how it can be used together with RSSI to accurately reflect the performance of an 802.11n MIMO link. We measure link quality or performance in terms of Packet Error Rate (PER, where: PER = 1 − PRR). We conduct all experiments for both 20 MHz and 40 MHz channels, and we discuss our observations for three MCS indices that cover robust (MCS 8), intermediate (MCS 12), and aggressive (MCS 15) PHY rates. For each MCS, we send 5,000 1kB UDP datagrams over 50 different links, selected to cover a wide variety of cases.[1] For legibility, we present a subset of our results that best represents the patterns in the behavior of RSSI and *diffSNR*.

2.1. The limitation of RSSI

RSSI, used to directly compute the Signal-to-Noise Ratio (SNR) in dB,[2] has traditionally been used to represent the quality of a link [13]. With knowledge of SNR, and assuming a channel with AWGN noise (additive white Gaussian noise), empirical curves or known theoretical formulas have been used to infer the bit error ratio (BER) for any given MCS. With the BER and the transmitted frame length, an upper bound for the packet error rate (PER) can then be estimated. 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 [14].

Fig. 1 plots PER vs SNR averaged over the 50 links in our testbed. Fig. 1(a) plots the values for one transmit stream and Fig. 1(b) for two streams. As RSSI increases, we expect PER to drop since the receiver can better decode the received signal. Fig. 1, however, shows that this is not necessarily the case. When we compare Figs. 1(a) and (b), we observe irregular behavior particularly for aggressive modulation schemes with spatial multiplexing (MCS 12 and 15). PER does not converge to 0 for high SNR and surprisingly in Fig. 1(b), performance degrades for $\text{SNR} > 55$ dB; that is, contrary to what seems to be an established dogma among many network administrators, higher transmitter power does not translate to better reception.

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. Therefore, one explanation is that high output power can be the source of constellation errors when using OFDM. The combination of OFDM and high order amplitude modulations (such as 64-QAM used in MCS 5 to 7 and 13 to 15) is prone to high peak-to-average ratios: high peaks cause the power amplifiers to move toward saturation [15], exhibiting non-linear behavior that produces inter-modulation distortion. However, the anomaly is observed for MCS 12 and 15, but not for MCS 7.

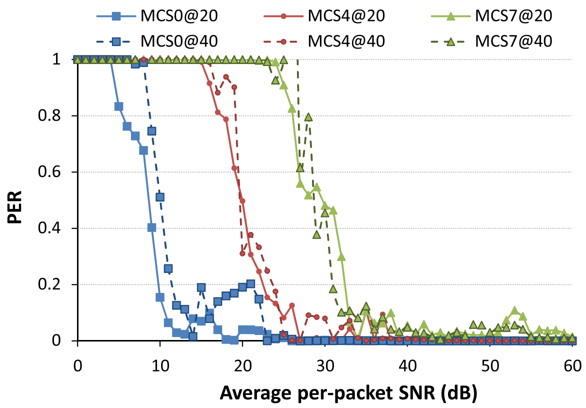
Therefore, the answer must be the presence of a dominant path between a transmitter/receiver pair, such as when the nodes are close to each other in direct line-of-sight, which results in a high Rician *K*-factor and the channel becomes increasingly correlated in space. This hampers the utilization of spatial multiplexing [16], since it requires that simultaneous streams follow independent paths with sufficiently different spatial signatures.

Fig. 1 also shows that SNR is a poor indicator of link quality for different channel widths. For the same SNR, a 40 MHz 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 40 MHz channel, therefore, not only requires a stronger transmission power to achieve the same SNR [17] but also a higher SNR to provide the same PER.

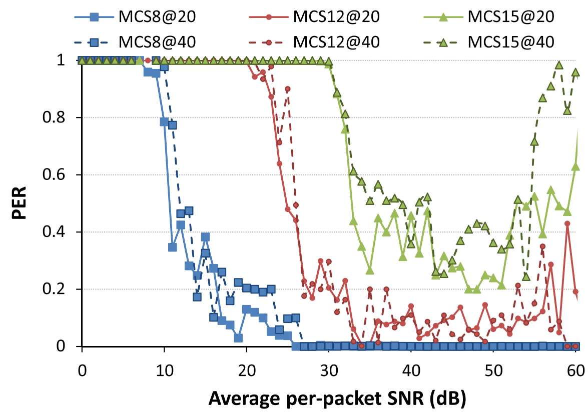
Despite those anomalies, Fig. 1 seems to show clear transitions between usable and unusable links for low MCS indices. However, if we observe the raw data used to obtain the average behavior depicted in Fig. 1, those transitions are less apparent. Figs. 2 and 3 plot per-link PER vs SNR for all testbed links and bandwidths. On the *Y*-axis, each point represents PER measured over 5,000 frame transmissions in one particular link. On the *X*-axis, each point is the average SNR of the received stream of packets. Fig. 2 shows results for a single stream, while Fig. 3 for two streams.

[1] See Section 5 for testbed details.

[2] 802.11n Atheros driver assumes a predefined noise floor and thus computes RSSI = SNR + NoiseFloor, where SNR is measured from the channel, and NoiseFloor is set to a default, predefined value of −95 dBm. We therefore use RSSI and SNR interchangeably, as the latter is a scaled version of the former.

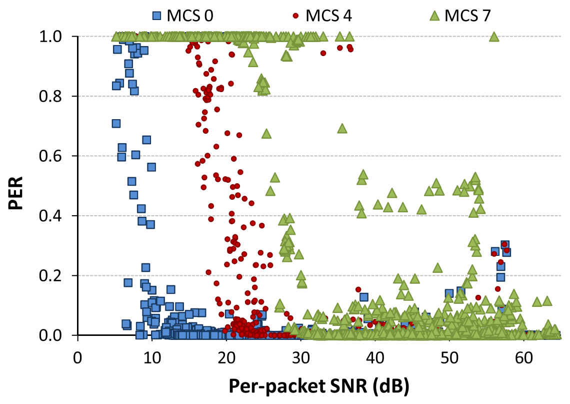


(a) One transmit stream.

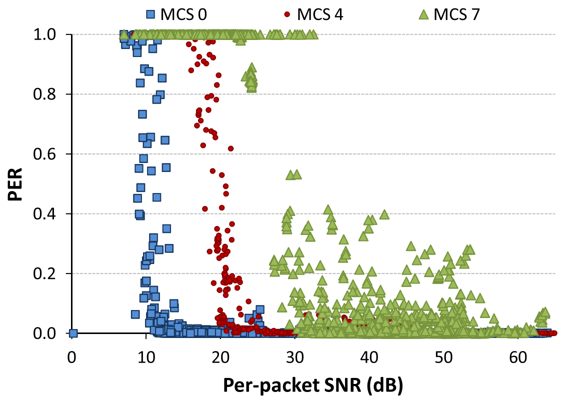


(b) Two transmit streams.

Fig. 1. Average PER and per-packet SNR over testbed links.



(a) 20MHz wide channel.



(b) 40MHz wide channel.

Fig. 2. Per-link PER vs SNR measurements for one stream.

Figs. 2 and 3 show that RSSI is a reliable metric when robust MCS modes are used that exploit spatial diversity. For example, the transition region for MCS 0 is only 3–4 dB wide; for a given link at MCS 0, if the measured SNR is below 6 dB, the link is infeasible (PER ≈ 1) and, if the SNR is above 10 dB, it is feasible (PER ≈ 0). However, for SNR values between 5 and 10 dB, the feasibility of a link is uncertain; some links yield an excellent performance with an SNR of 6 dB, while others are not feasible with a higher SNR of 10 dB. This uncertainty is amplified with spatial multiplexing and more aggressive modulations, where the transition region between a feasible and an infeasible link becomes wider. For example, when both spatial multiplexing and moderate or fast PHY rates are used (e.g. MCS $\geqslant 12$), the transition region could be as wide as 35 dB! In such cases, RSSI alone does not provide sufficient information to assess the feasibility of a link. This result is consistent with previous work [12,7].

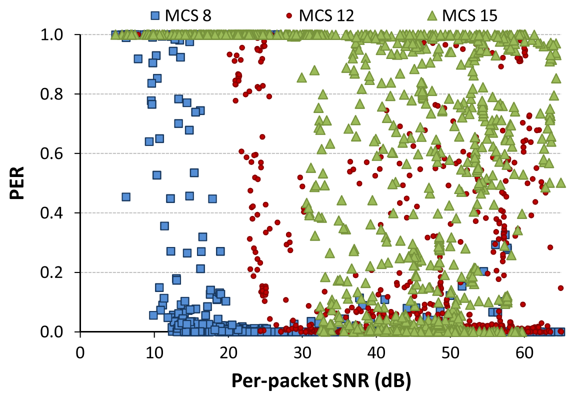
2.2. Channel State Information (CSI)

CSI describes the current channel conditions with fine granularity.[3] It consists of the attenuation and phase shift for each spatial stream to every receive antenna, for every OFDM subcarrier (56 subcarriers for a 20 MHz bandwidth and 114 for a 40 MHz bandwidth in 802.11n). Measuring a complete and timely CSI for all possible MIMO channel configurations requires excessive sampling overhead [1].

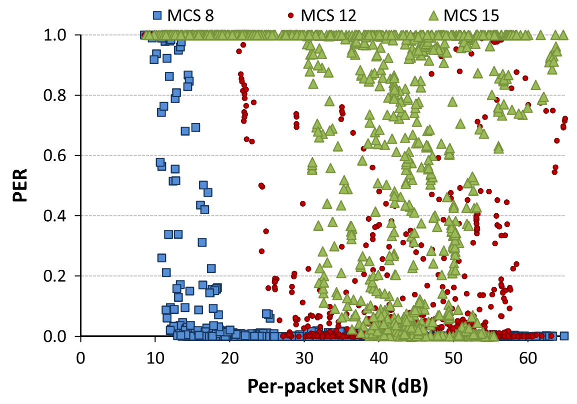
In some implementations, successful decoding of a data packet is required to compute CSI [7]. Additionally, for a $T \times R$ MIMO system of bandwidth W, a series of probe frames must be sent using T transmit antennas over a bandwidth W, and received over R receive antennas to obtain the complete $T \times R \times W$ CSI matrix. For example, a 3×3 MIMO system allows transmissions using one, two or three simultaneous data streams,[4] and thus the complete CSI requires probing all combinations of number of streams and transmit antennas. As a result, current CSI estimation approaches require seven probes (or samples) to obtain CSI for all possible configurations of a 3×3 MIMO system. Single-stream MCSs require three probes, to collect CSI for each individual transmit antenna. Similarly, two-stream MCSs require three probes, to collect CSI for each combination of two transmit antennas. Finally, three-stream MCSs require a single probe from a transmission from all three

[3] Here, we refer to the full CSI matrix and not the coarse grained CSI provided by Intel 802.11n chipsets [7].

[4] Where the maximum number of supported data streams in a $T \times R$ MIMO system is $min(T, R)$.



(a) 20MHz wide channel.



(b) 40MHz wide channel.

Fig. 3. Per-link PER vs SNR measurements for two streams.

transmit antennas. The number of required probes increases dramatically with more antennas and wider channel widths.

Communicating the computed CSI matrix in a feedback packet also consumes significant bandwidth overhead. The size in bytes of a feedback packet with complete, noncompressed CSI is 1.029 KB for a 20 MHz channel and 2.095 KB for a 40 MHz channel [9]. Based on channel coherence time [18], CSI at the transmitter needs to be updated at least once every 50 ms. CSI feedback, as a result, consumes 160.64 Kb/s to 335.2 Kb/s respectively. In a per-packet RA implementation, where CSI needs to be updated frequently, the bandwidth consumed by CSI feedback quickly becomes a significant overhead.

Complete CSI is clearly expensive to obtain and communicate, and therefore its applicability to a per-packet RA solution, particularly in dynamic environments where timely channel information is necessary, is limited. Our aim is, therefore, to identify an alternative MIMO link metric in the design of an agile rate adaptation mechanism.

2.3. 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 [7]. Antenna correlation, or spatial selectivity, is another factor [19]. Both factors, however, require costly CSI which is supported by only very few devices [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 [20]. 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*.

After analyzing real-time traces of RSSI and *diffSNR* in different scenarios, we observe that *diffSNR* does not depend significantly on either (i) the transmitter's output power, where *diffSNR* varies less than 6%, (ii) the MCS used, where *diffSNR* varies less than 2% as shown in Fig. 4, or (iii) the channel width. On the other hand, *diffSNR* shows a clear dependency on the environment: factors such as rich scattering, dynamic/static positioning, line-of-sight, and obstacles. Fig. 5 provides two paradigmatic examples of the real-time evolution of SNR and *diffSNR*. We find that a static scenario exhibits fewer variations, as shown in Fig. 5(a), while a more dynamic environment is reflected in a wider dispersion of the measured *diffSNR* which exhibits frequent peaks, as depicted in Fig. 5(b). A peak in *diffSNR* can occur when RSSI increases and a subset of the antennas receive constructive interference. However, we observe that high diffSNR peaks are often (80% of the time) caused by some of the antennas suffering from fading; that is, there is a negative correlation between RSSI and *diffSNR*. This behavior is also deduced in Fig. 5(b).

Moreover, for links with similar RSSI, we find that *diffSNR* can be used to characterize their performance differences. We illustrate this behavior in Fig. 6 using three representative links. We evaluate their PER vs SNR relationships using spatial multiplexing (MCS 12 and 15) and a 40 MHz channel. Link 1 successfully transmits packets ($PER < 0.02$) using MCS 12; for MCS 15, there is a clear transition around 34 dB SNR. Although Link 2 has similar RSSI values to Link 1, it clearly exhibits worse performance: for MCS 12, PER increases rapidly when $SNR < 30$ dB and MCS 15 remains lossy until $SNR > 44$ dB. The difference between Link 1 and 2 can be explained with *diffSNR*: for Link 1, we measure an average *diffSNR* of

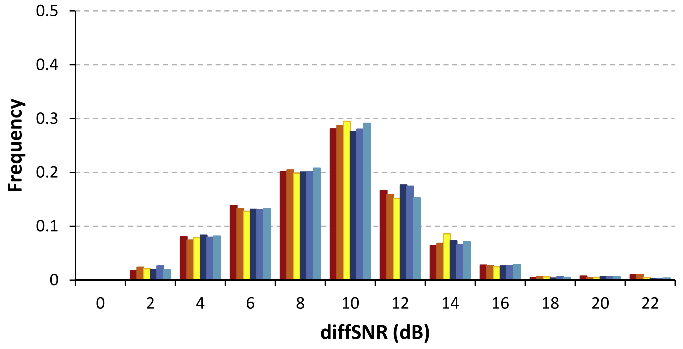
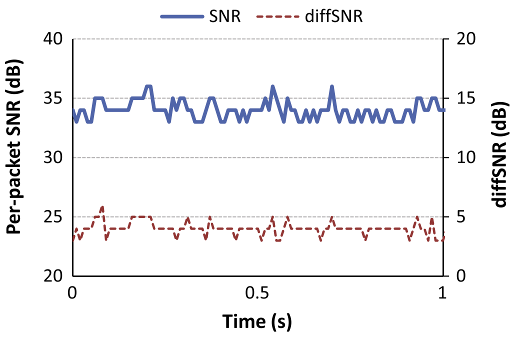
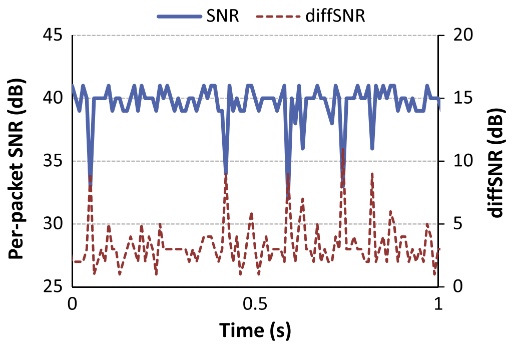


Fig. 4. Normalized frequency of diffSNR measurements. Leftmost bars (red tones) represent MCS 0, 4, and 7; rightmost bars (blue tones) represent MCS 8, 12, and 15. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) Static scenario (night run) and 20MHz channel.



(b) Dynamic scenario (during office hours) and 20MHz channel.

Fig. 5. Real-time evolution of per-packet SNR and *diffSNR*.

1.82 dB, with a standard deviation of 0.30, while for Link 2, the average *diffSNR* is 9.46 dB, with a standard deviation of 0.37. Link 3 displays the worst performance, showing an average *diffSNR* of 13.41 dB. 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.

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

3. Link predictions with *diffSNR*

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.

3.1. A measurement-based approach

Our analysis, summarized in Section 2.3, reveals the dependency of performance on RSSI and *diffSNR* together. That is, 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; we show two representative graphs in Figs. 7 and 8. Fig. 7 shows the measured PRR for MCS 7 (aggressive modulation, one stream) and Fig. 8 for MCS 12 (intermediate modulation, two streams) as a function of average per-packet SNR and *diffSNR*. To more clearly show the transition between the links with PRR > 0.5 and those with PRR < 0.5, we include representative 2D cross-cuts of the 3D plots. For example, as shown in the projected image in Fig. 7(b) with an SNR of 32 dB, a link with *diffSNR* below 5 dB performs well. However with a *diffSNR* above 10 dB, the PRR falls to almost 0.

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. For aggressive modulations, and particularly when multiple streams are used, predicting links with even moderate *diffSNR* values involves higher uncertainty. This uncertainty explains the presence of data points with PER > 0.5 inside the “feasible” region (cf. *diffSNR* > 8 dB in Fig. 8).

From our experiments, we gather sufficient data to plot the PRR(SNR, *diffSNR*) surfaces for each allowable MCS. The combination of these surfaces for all MCS values constitutes our measurement-based link predictor. Our proposed predictor is thus based on a three dimensional matrix, as depicted in Fig. 9. SNR and *diffSNR* measurements, along with the operating MCS and bandwidth of a link, constitute the matrix coordinates from which our measurement-

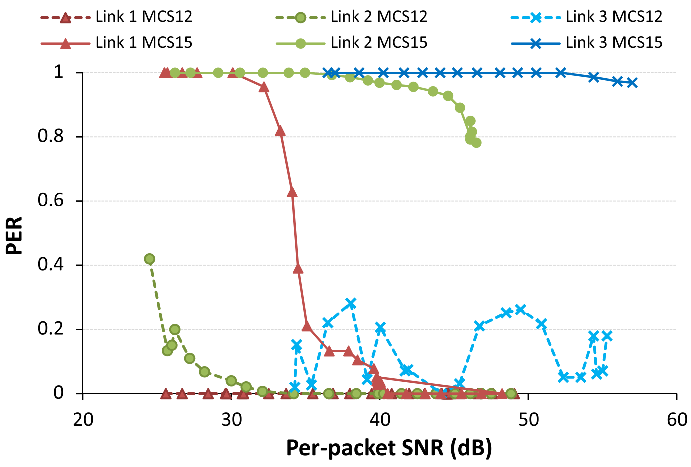


Fig. 6. PER vs per-packet SNR for three links (40 MHz channel).

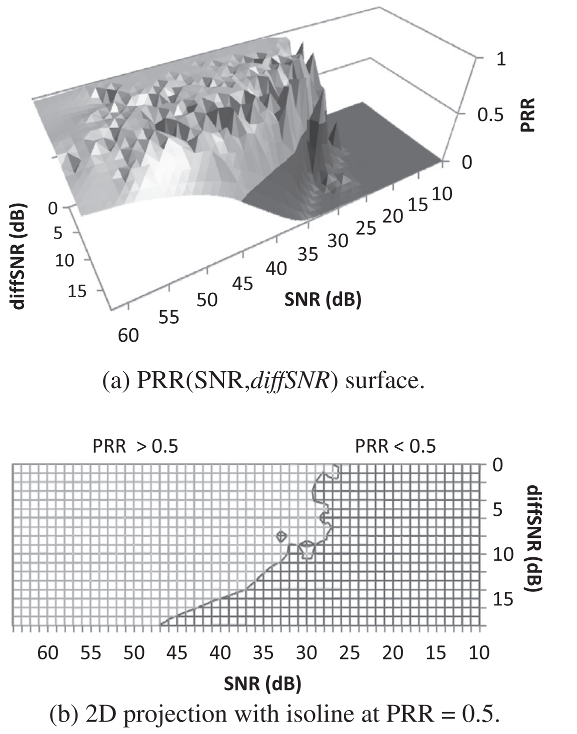


Fig. 7. PRR as a function of packet-SNR and *diffSNR* for MCS 7 and 20 MHz channel.

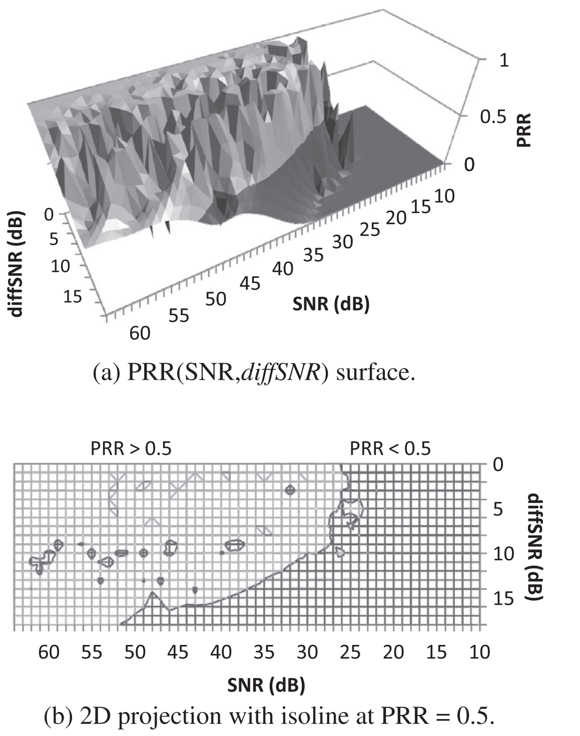


Fig. 8. PRR as a function of packet-SNR and *diffSNR* for MCS 12 and 20 MHz channel.

based link predictor identifies the corresponding expected PRR for that link. It is important to note that our testbed provides us with SNR data for the control (i.e. primary) 20 MHz channel and when channel bonding, the extended 40 MHz channel. Predictions for 20 MHz links can be made from measurements under 40 MHz links, but not vice versa [10,11]. Therefore, our predictor builds separate PRR surfaces for both 40 MHz and 20 MHz channels for each MCS.

To gather sufficient data to pre-compute and build the 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. As for the (SNR, *diffSNR*) data points that do not have measured values, we fill them by interpolation, using the nearest measured data points.

3.1.1. About frame size

Intuitively, PRR depends on the length of the frame. Hence, frame length should be accounted for to predict PRR. The PRR for any given frame size can be roughly

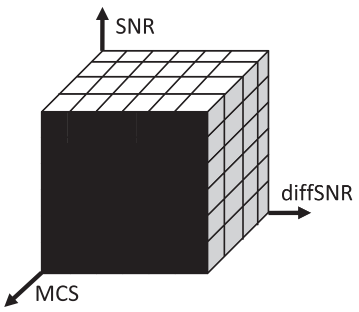


Fig. 9. The 3D matrix contains measured and interpolated PER values and is the structure of our proposed link predictor.

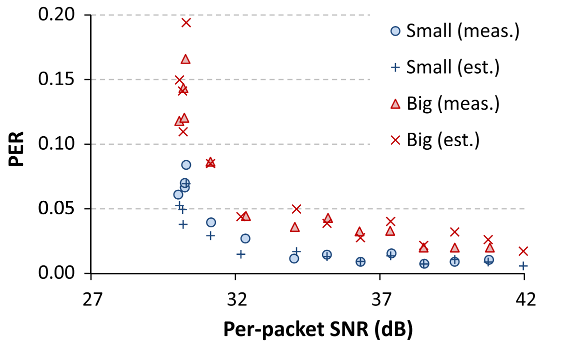


Fig. 10. Measured and estimated PER vs SNR for payload size of 300B (Small) and 1500B (Big).

estimated from the PRR we measure for frames carrying 1kB payload using the equation: $P_x = \left(\sqrt[L_{1000}]{P_{1000}}\right)^{L_x}$, where P_x is the PRR for a payload of x bytes, and L_x is the total length (in bits) of a frame carrying an x byte payload [21]. Our measurements in different scenarios consistently show that the transition regions for a link from high quality to lossy do not exhibit a noticeable difference when the payload size is changed. Fig. 10 plots both the estimated PER, following the P_x equation above, as well as the measured PER for payloads of 1500B (Big) and 300B (Small) for a given link using MCS 12. We observe that, as expected, larger frames show higher PER; however, the impact of frame size on the feasibility of a link is still negligible. This result indicates that transition regions do not depend on frame size, and thus we do not add frame size as an additional dimension.

3.2. Prediction accuracy

The link predictor is a pre-computed 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 69 dB and *diffSNR* values from 0 to 19 dB, with 1 dB precision.[5] 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, pre-computed link predictor by comparing the predicted PRR 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 20 m apart. We include this second group to evaluate the utility and accuracy of the proposed predictor in unfamiliar and dissimilar environments.

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

For the first group of nodes located in a familiar environment, the average absolute error in PRR predictions, computed as the difference between the measured PRR and the predicted PRR, is only 4.8%. This error ratio 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.[6] As for the second group of nodes in new environments, the average absolute error in PRR 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 PRR predictions such that:

$$PRR_k^{m,B} = PRR(m, B, SNR_k, diffSNR_k) + E_{k-1}^{m,B} \tag{1}$$

where $PRR_k^{m,B}$ is the predicted PRR for MCS m and bandwidth B; SNR_k, and $diffSNR_k$ are the currently measured RSSI and *diffSNR* values; and $PRR(w, x, y, z)$ returns a PRR 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 $0 \leqslant E_{k-1}^{m,B} \leqslant 1$. As detailed in Section 4.2.3, we compute the error by tracking the real PRR from received data frames (i.e. no extra signaling is required) and comparing it with the predicted PRR. The value $E_k^{m,B}$ is computed as an exponential moving average of the measured error samples with a configurable α. A large α allows the prediction mechanism to adapt faster to rapidly varying channels (e.g. with user mobility), while smaller α values improve the accuracy of the error estimation in more stable environments.

We re-evaluate our results in the new environments using Eq. (1). Fig. 11(a) computes the average absolute error in PRR predictions, for all tested MCS indices. On average, the error in our improved PRR predictions lies below 5.8%. Fig. 11(b) shows that link feasibility predictions improve to a 95.5% hit rate.

[6] Given the steep transitions shown in PRR surfaces, PRR values between 0.3 and 0.7 are good indicators of the limits of link feasibility. Henceforth, for the purpose of these tests, we consider a link feasible for a given MCS if PRR > 0.5.

Recall that we interpolate to fill the gaps in a PRR(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% come from interpolated values. For outdoor links, the proportion is 61 measured to 39% interpolated. For the familiar environment, interpolated values are not used. It is therefore not surprising that the outdoor environment has a greater number of predictions from interpolated values, as it exhibits different multipath characteristics from an indoor environment, where the PRR(SNR, *diffSNR*) surfaces were built.

A detailed evaluation of PRR predictions in these two new environments reveals the same patterns observed in the lab measurements. First, performance of aggressive modulations is more difficult to predict, and this is translated into lower feasibility prediction rates and higher PRR prediction errors. Second, PRR prediction errors increase when spatial multiplexing is used (4.6% average absolute error for one stream vs 6.9% for two). Finally, the PRR of a 20 MHz channel can be predicted with slightly greater accuracy than a 40 MHz channel (96.0% feasibility prediction hits for 20 MHz channels vs 95.1% hits for 40 MHz).

Spatial multiplexing, aggressive modulations, and wider (i.e. $>$20 MHz) channels are all features of new generation IEEE WLANs that achieve higher data rates at the risk of greater susceptibility to loss and changes in environment conditions. Accurately reflecting such detailed environment conditions however, such as the number of independent paths and frequency selectivity, is achieved using fine-grained CSI alone. Our link predictor, based on *diffSNR* forgoes the high cost of fine-grained CSI for ease of implementation, by using coarse-grained information on channel conditions, while maintaining a reasonable level of accuracy in predicting environment conditions. In the remainder of this paper, we prove the utility of *diffSNR* by implementing a novel close-loop rate adaptation mechanism that includes the link predictor described in this section.

4. ARAMIS

4.1. Overview

ARAMIS is a closed-loop RA solution for 802.11n MIMO environments. In the design of such a solution, we identify three important, high-level components. These three components form the critical foundation towards the implementation of ARAMIS. The first component is a link metric that can be used to accurately characterize MIMO link performance. We use our proposed, practical, MIMO link metric, *diffSNR*, which provides a good balance between implementability and accuracy, and is deployable on any and all off-the-shelf WiFi chipsets (cf. Section 2.3).

The second component is a mechanism that can accurately predict the PRR of a link for any MCS and bandwidth combination, which we refer to as the *link predictor* (cf. Section 3). 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 main and all-inclusive 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.

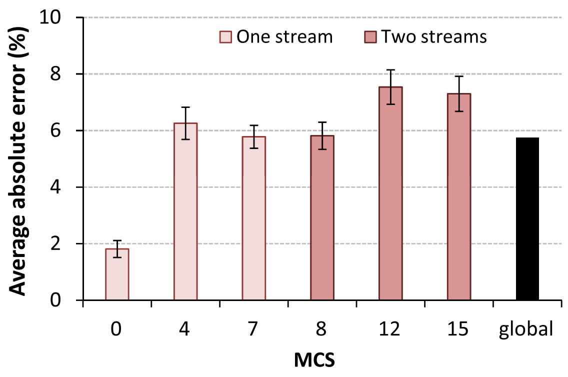
Building on these three, high-level components, Fig. 12 depicts the specific components in our implementation of ARAMIS. In other words, Fig. 12 presents the elements of ARAMIS's *rate selector*, and the corresponding communication flow between an 802.11 transmitter and receiver pair. Note that the primary functionality of ARAMIS is implemented at the receiver. We now follow with a description of each component, starting with the first interface into ARAMIS's functionality, which is the *Frame Monitor*, implemented at the receiver.

The *Frame Monitor* maintains updated information on channel conditions by measuring the link metric from existing data traffic. We use our proposed practical MIMO link metric, *diffSNR*. 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. As explained in Section 3, our measurement-based *Link Predictor* needs access to the set of pre-computed PRR surfaces. To improve the accuracy of predictions, the *Link Predictor* is assisted by a *Training Phase* that corrects errors in predicted PRR values in real-time by comparing recent predictions with current performance reported by the *Frame Monitor*. 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. Finally, the transmitter's *Feedback Receiver* forwards the selected MCS and bandwidth to a *Rate Management* entity, which configures the PHY accordingly. In the absence of feedback, a backup *Timer* is implemented at the transmitter to reset the operating rate.

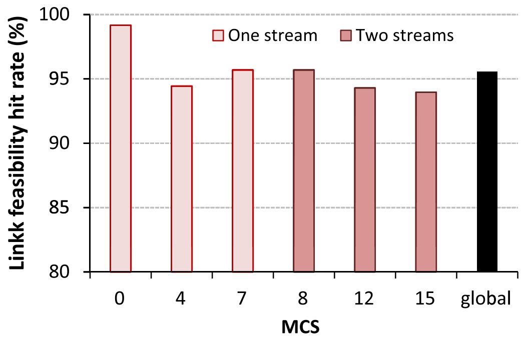
Following this outline, we now describe in detail the design components of our *rate selector*, which is used as a framework for the implementation of RA with *diffSNR* (including the *diffSNR*-based link predictor).

4.2. Rate selector

The *rate selector* is the final main and all-inclusive 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 (SNR, *diffSNR*), and the *Link Predictor* in the design of an effective rate selector. We use the terminology illustrated in Fig. 12.



(a) Performance of link PRR predictions (99% confidence intervals).



(b) Performance of link feasibility predictions.

Fig. 11. Evaluation of the link predictor in new environments.

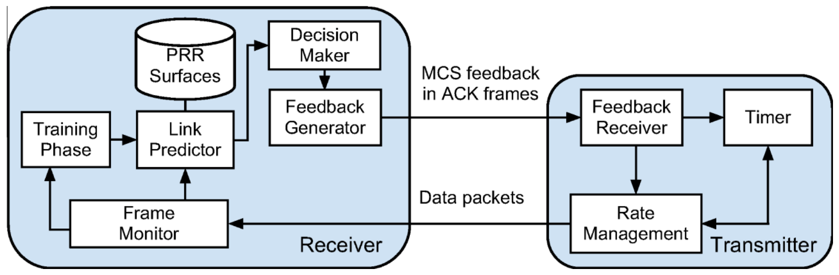


Fig. 12. Block diagram of ARAMIS and the communication flow between its components implemented at the receiver (left block) and transmitter (right block).

Algorithm 1. ARAMIS(SNR, *diffSNR*)

```
Output: (1) MCS $m$; (2) Channel width $B$;
1: if $newPacket$ = true then
2:    ($SNR_{avg}$, $diffSNR_{avg}$) ← update-moving-average(SNR, $diffSNR$)
3:    if exception(SNR, $diffSNR$) = true then
4:       ($m$,$B$) ← decision-maker() ← link-predictor($SNR_{avg}$, $diffSNR_{avg}$)
5:    end if
6: end if
```

4.2.1. 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. 5 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*.

4.2.2. Decision maker

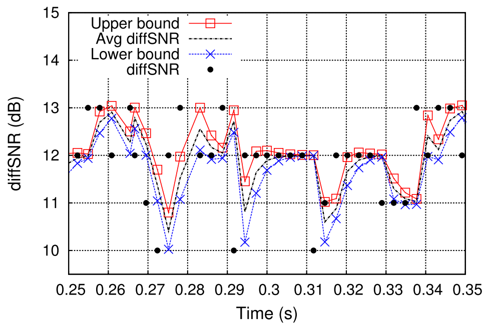
Our *rate selector* uses a link's current channel conditions, reflected through the link metric, as input arguments to the *Link Predictor*, in this case using (SNR_{avg}, $diffSNR_{avg}$). The *Link Predictor* determines accurate PRR estimates for all supported MCS and bandwidths for that link, as described in Section 3. 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 [10], 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 (increased to meet the requirements of reliability demanding applications, or relaxed to increase raw throughput).

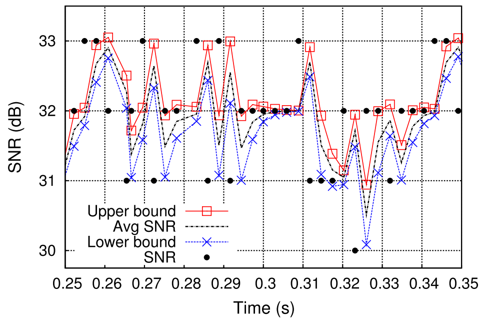
Fig. 13 demonstrates the behavior of ARAMIS in real-time, as described in Algorithm 1. In Fig. 13(a) and (b), we plot the instantaneous values, moving averages, and upper and lower bounds for our link metrics, both SNR and *diffSNR*. Fig. 13(c) depicts how ARAMIS changes MCS on a per-packet basis based on the correlated (SNR, *diffSNR*) values, 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 40 MHz channel for this link. Although not depicted, ARAMIS opted for a 20 MHz channel, particularly for the weakest links in our evaluation.

4.2.3. Training phase

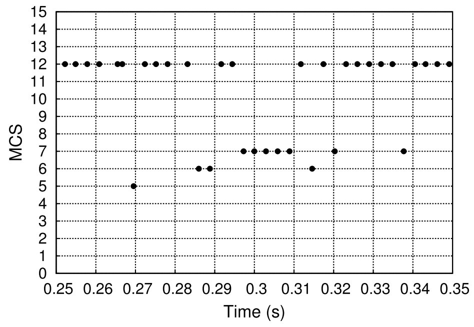
To improve the accuracy of predicted PRR values for all MCS and bandwidth combinations, a training mechanism is performed on-the-fly using the statistics of received frames, which does not incur any extra overhead. ARAMIS measures the link's actual PRR by dividing the number of received frames with the 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 Eq. (1). In our implementation, $E_k^{m,B}$ is computed as the moving



(a) Variation of *diffSNR* with time.



(b) Variation of SNR with time.



(c) Selected rate based on *diffSNR*, SNR values from previous packet measurements.

Fig. 13. Depiction of ARAMIS measurements and behavior. We show that ARAMIS responds to changes in (SNR, *diffSNR*) conditions by modifying the MCS and bandwidth when necessary. In this case, the link always chooses a 40 MHz channel.

average of error samples with $\alpha = 0.9$. Our α is large to give more weight to recent error samples, since the long-term mean error in PRR predictions is close to 0.

4.2.4. Feedback generator

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 [9]. 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).

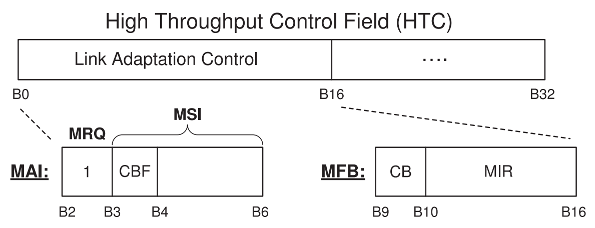


Fig. 14. 802.11n compliant MCS feedback system.

Table 1
HTC subfields that support receiver-based RA.

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

[a] Bits allocated to support channel width feedback.

Fig. 14 shows the HTC field with its corresponding link adaptation control field, where the subfields are described in Table 1. 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[11], where the CBF field set by the AP defines the supported bandwidth in the given WLAN. This allows ARAMIS to make informed channel width decisions using the insight from network layer conditions. For example, if CBF is set to 1 by a channel management approach, the client can request to operate on both a 20 MHz and 40 MHz channel, which it specifies in the CW subfield, and if CBF is set to 0, the client only operates on a 20 MHz 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.

4.2.5. 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.[7] Our results show that ARAMIS's per-packet rate adaptation is able to rapidly recover from this MCS reset.

[7] This timer presents a tradeoff: a small timer may hastily fall back to low rates in the presence of severe collisions, and a large timer may prevent ARAMIS from rapidly adapting to degradations in signal quality.

5. Performance evaluation

We evaluate ARAMIS, first using simulation based on packet traces from our experimental platform. We then implement ARAMIS on a real testbed and compare its performance to that of existing RA solutions under various network conditions. The goal of the trace-driven simulation is to evaluate the design choices for ARAMIS, since it gives us the flexibility to reproduce environment conditions while evaluating the performance of various design parameters. Furthermore, the post-processing of these traces allows us to simulate an optimal (and impractical) algorithm to use as a benchmark for our approach.

In our testbed implementation, we evaluate ARAMIS under various scenarios, including interference, mobility, and hidden nodes. Our goal is to demonstrate the efficacy of ARAMIS in accurately responding to channel conditions compared to other popular 802.11n RA solutions. We measure performance in terms of achieved throughput.

Testbed details: Both evaluation environments are built over our testbed 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 2.4/5 GHz chipset. The AR5416 baseband and MAC processor allow MCS indices 0 to 15. Each laptop runs the Atheros Ath9k device driver that supports 802.11n [5]. We run our experiments on the 5 GHz range and verify the lack of background traffic with a spectrum analyzer.

5.1. Trace-driven simulations

5.1.1. Simulation environment

The simulation utilizes packet traces that we collect over our testbed for various links. For each MCS, we send 5000 1kB UDP datagrams from the AP to its client at a constant inter-packet delay of 2.7 ms. We introduce this delay to avoid issues related to buffer overflow and conditions that restrict our ability to reproduce environment conditions. For the same reasons, we disable packet aggregation. The packet traces are stored at the client and consist of per-packet (SNR, *diffSNR*) values and inter-packet delays, as well as the computed PER, average SNR, and throughput for the entire transmission. We fix the transmit power to 11 dBm, which is the maximum common power level among all MCS. We conduct the above for both 20 MHz and 40 MHz channels.

We collect packet traces in interference-free environments as well as controlled interference conditions shown to affect 802.11n performance [10]. For the interference conditions, we introduce an interfering link that operates on either the same or an adjacent channel.

We use the above packet traces as input to our simulator. The simulator is built in custom C and Python. The simulator works by replaying per-packet transmissions using each packet's transmission characteristics, namely its MCS, channel width, delay to the next consecutive packet, and packet loss.

We implement ARAMIS and other solutions, namely *Best Fixed* and *Oracle* in our simulator. *Best Fixed* fixes the MCS that maximizes throughput for the entire simulation run, and serves as a performance baseline. *Best Fixed* differs from ARAMIS in that it does not perform per-packet RA, but rather per-transmission RA by selecting the best MCS that maximizes throughput for that transmission. We add *Best Fixed (stable)* to the set of alternative solutions, and it represents the MCS *Best Fixed* chooses under stable, interference-free conditions. Finally, Oracle pre-processes the entire dataset of traces for every MCS and bandwidth combination and is thus able to select, for each packet, the fastest MCS that guarantees successful reception. Oracle makes optimal per-packet RA decisions, and therefore it serves as an upper bound for performance. Each simulation is run for 200 s of simulation time.

5.1.2. Simulation results

Fig. 15 presents the simulation results. Fig. 15(a) depicts the UDP throughput under interference-free channel conditions for two types of links, where a weak link is unlikely to support high MCS due to weak SNR/*diffSNR*. Fig. 15(b) shows the result with channel interference. We include *Best Fixed (stable)* to Fig. 15(b) to show how the best MCS that is selected in interference-free conditions would perform in interference environments. The insight from these results is the importance of *per-packet* rate adaptation in the presence of interference as well as in stable environments, where changes in the channel occur on narrow timescales. *Best Fixed (stable)* performs poorly when interference is introduced. ARAMIS takes advantage of per-packet processing, thus allowing quick and fine adjustments to varying channel conditions. In the presence of interference, however, there are fewer opportunities to take advantage of per-packet RA. As a result, ARAMIS is shown to provide near-optimal performance, similar to that obtained by the best fixed MCS, which is another ideal solution that requires foreknowledge of interference conditions to select the appropriate MCS. With aggregation disabled, all three solutions show little performance differences since they are close to the maximum theoretical throughput. In the next section we show how this PHY adaptation is combined with aggregation, a link layer feature, to leverage the potential of IEEE 802.11n.

5.2. Testbed implementation

5.2.1. Testbed environment

We compare the performance of ARAMIS to that of two widely used open source 802.11n RA solutions, Ath9k [5] and Minstrel HT [4], and RAMAS [2], 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, which consists of the 802.11n-supported modulation types, and an *enhancement* group that includes the number of independent spatial streams and bandwidth. RAMAS implements two independent credit-based systems for upgrading and downgrading the features of each group, where each group has a different set of rules for accumulating credits. For example, if the flow between a transmitter/receiver pair accumulates a number of credits within a given time window that exceeds a set threshold (where the credit counter is incremented by one each time packet errors fall below a given threshold), RAMAS switches to a more aggressive modulation (and vice versa).

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 40 MHz to allow it to exploit higher data rates. Ath9k switches to a 20 MHz 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 11 dBm 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. 16, where the letters represent node locations. We note that this evaluation is based on a reduced set of 11 nodes. This reduced set is carefully chosen to include links with different characteristics (LoS, non-LoS, and a wide range of received signal strengths).

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

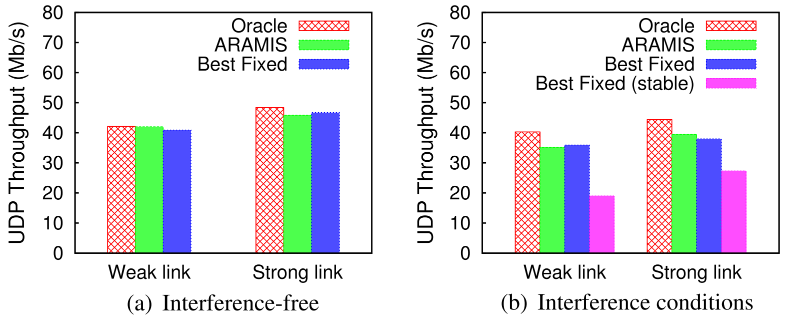


Fig. 15. Comparison of ARAMIS against other representative solutions.

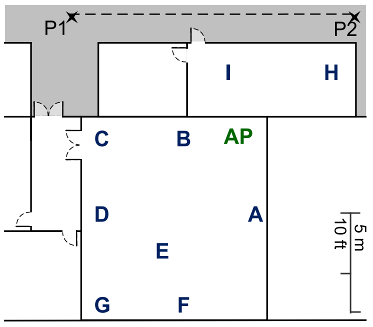


Fig. 16. Floorplan of our testbed environment.

by implementing netlink sockets and transmitting packets with the HTC field over the wire from the receiver to the transmitter driver code. As a result, we believe the performance in our evaluation is a lower bound. Although transmitting feedback over the wired ensures no delivery loss, note that if a packet is not successfully received (e.g. there is a collision or loss), feedback will not be generated over the wire, as no ACK will be sent over the air. If the packet is successfully received, the loss of an ACK is unlikely as shorter ACK frames are sent at low, reliable rates, while the feedback over the wire is always received with a larger delay. The overhead of user-space-kernel communications, though minimal, often lead to delayed rate feedback receptions that 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 4.2.3, and the accuracy of measured $E_k^{m,B}$ samples.

5.2.2. Testbed results

Figs. 17 and 18 show that ARAMIS consistently outperforms other algorithms in all test cases, with an average of 0.62-fold and up to a 2-fold throughput increase in interference-free environments, an average of 2-fold and up to 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. 17 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

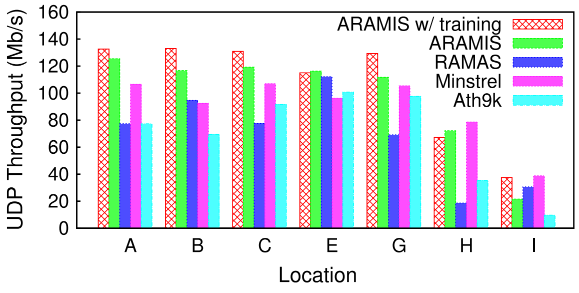
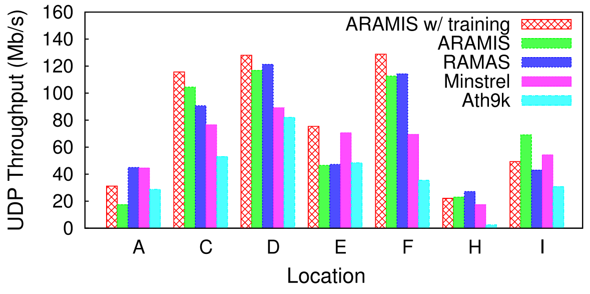
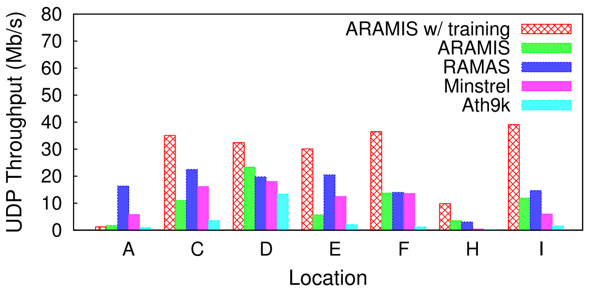


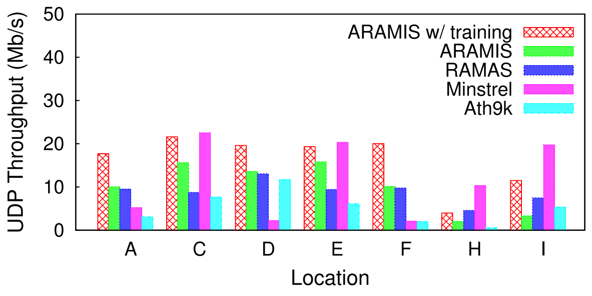
Fig. 17. Algorithm performance in an interference-free environment.



(a) Adjacent 40MHz interferer



(b) Adjacent 20MHz interferer



(c) Channel sharing with a 20MHz Interferer.

Fig. 18. Algorithm performance under interference conditions.

26% and an average of 16% over Minstrel HT, up to 124% and an average of 90% over Ath9k, and up to 287% and an average of 79% over RAMAS. Note that our results for RAMAS are somewhat different from those reported [2], since they were obtained in different scenarios. RAMAS was previously evaluated only on the 2.4 GHz range, which significantly limits the performance benefits of 802.11n features [22,3].

RAMAS leads to an average PER of 11% and a maximum of 20%. The credit scheme it uses to adapt the number 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 seek a very low PER region (between 2 and 5%) at the cost of often ignoring 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 [10]. Fig. 18(a) presents results with an interfering link that operates on an adjacent 40 MHz channel, while Fig. 18(b) for an adjacent 20 MHz 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 40 MHz interferer shown in Fig. 18(a), we improve the throughput by an average of 10% and up to 60% over RAMAS, an average of 25% and up to 85% over Minstrel HT, and an average of 192% and up to 782% over Ath9k. For an adjacent 20 MHz interferer shown in Fig. 18(b), the improvement is an average of 88% and up to 220% over RAMAS, an average of 400% and up to 412% over Minstrel HT, and an average of 1900% and up to 1908% over Ath9k. We observe greater performance improvements with an adjacent 20 MHz interferer, since it is the more harmful configuration [10], and ARAMIS mitigates this interference.

Although the cap on the error threshold mitigates the effect of interference on RA, it can also degrade performance, as seen from location A in Fig. 18(b): ARAMIS first selects a rate which it identifies is best under interference-free conditions, and then reacts to high losses by capping the error threshold to 0.2. We find that, in cases such as location A when a strong link suddenly becomes extremely lossy due to strong interference from channel leakage (or other non-802.11 interference), the error threshold forces ARAMIS to stay at higher rates than best, leading to performance losses.

We also investigate the channel sharing scenario with an interferer on a 20 MHz channel. This scenario has been shown to create worse fairness issues than a 40 MHz co-channel interferer whereby the slower 20 MHz channel occupies the medium for longer periods of time [10]. In Fig. 18(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. At locations H and I in Fig. 18(c), we notice that channel sharing coupled with poor channel conditions can hamper the performance of ARAMIS. Channel sharing limits the number of opportunities to transmit, and if channel conditions are already poor and most transmissions are lost, this phenomenon can trigger ARAMIS's expiration timer, leading to frequent fall-backs to slow MCS. This phenomenon motivates the need for a dynamic expiration timer based on channel conditions.

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 5 km/h. ARAMIS achieves throughput of 80.27 Mb/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.

6. 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 [14,7,12] have shown that RSSI is an unreliable metric to accurately predict performance. The utilization of *effective SNR* [7] 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 [1] 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 [13,8,23] and new algorithms for 802.11n networks have been proposed [7,3,24–26]. Although solutions for legacy clients have been effective, they fall short when applied in 802.11n OFDM-MIMO settings [3]. Existing 802.11n solutions require either costly CSI [27,7] or some form of a guided search (e.g., by probing candidate rates) to determine the best operating rate [3], 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 [28,26], or consider alternative energy efficiency goals [25].

7. 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.

Given the high costs of adopting CSI in 802.11n environments, we have introduced *diffSNR* as a practical and deployable link metric that can be used as a framework for effective RA on any off-the-shelf WiFi chipsets. We apply *diffSNR* within the framework of ARAMIS, our proposed closed-loop RA solution that jointly adapts rate and bandwidth. ARAMIS identifies and adopts a new method to quantify performance in 802.11n MIMO environments, and through trace-driven simulations, we have shown that ARAMIS with *diffSNR* achieves good accuracy and closely approximates an optimal solution.

Through further implementation of our solution, we have demonstrated that ARAMIS with *diffSNR* obtains impressive performance gains over leading 802.11n RA 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.

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 TEC2012-32531 and Programa Nacional de Movilidad de Recursos Humanos del Plan Nacional de I-D+i 2008–2011.

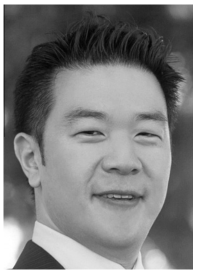
References

[1] R. Crepaldi, J. Lee, R. Etkin, S.-J. Lee, R.H. Kravets, CSI-SF: estimating wireless channel state using CSI sampling and fusion, in: IEEE Infocom, 2012.
[2] D. Nguyen, J. Garcia-Luna-Aceves, A practical approach to rate adaptation for multi-antenna systems, in: IEEE ICNP, 2011.
[3] I. Pefkianakis, Y. Hu, S.H. Wong, H. Yang, S. Lu, MIMO rate adaptation in 802.11n wireless networks, in: ACM MobiCom, 2010.
[4] Minstrel HT Linux Wireless, <http://linuxwireless.org/en/developers>.
[5] M. Wong, J.M. Gilbert, C.H. Barratt, Wireless LAN using RSSI and BER parameters for transmission rate adaptation, US patent 7,369,510, 2008.
[6] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, K. Almeroth, Joint rate and channel width adaptation for 802.11 MIMO wireless networks, in: IEEE Secon, 2013.
[7] D. Halperin, W. Hu, A. Sheth, D. Wetherall, Predictable 802.11 packet delivery from wireless channel measurements, in: ACM SigComm, 2010.
[8] S.H.Y. Wong, H. Yang, S. Lu, V. Bharghavan, Robust rate adaptation for 802.11 wireless networks, in: ACM MobiCom, 2006.
[9] IEEE 802.11n-2009 Amendment 5: Enhancements for Higher Throughput, IEEE-SA, October 2009.
[10] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, K. Almeroth, The impact of channel bonding on 802.11n network management, in: ACM CoNEXT, 2011.
[11] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, K. Almeroth, Intelligent channel bonding in 802.11n WLANs, IEEE Trans. Mobile Comput. 13 (6) (2013) 1242–1255.
[12] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, J. Zahorjan, Measurement-based models of delivery and interference in static wireless networks, in: ACM SigComm, 2006.
[13] J. Camp, E. Knightly, Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation, in: ACM MobiCom, 2008.
[14] D. Aguayo, J. Bicket, S. Biswas, G. Judd, R. Morris, Link-level measurements from an 802.11b mesh network, in: ACM SigComm, 2004.
[15] A. Amini, D. Lu, C. Edelman, Effects of high peak-to-average ratio on 5GHz WLAN power amplifier, in: DesignCon, 2002.
[16] H. Yang, A road to future broadband wireless access: MIMO-OFDM-based air interface, IEEE Commun. Mag. 43 (1) (2005) 53–60.
[17] M.Y. Arslan, K. Pelechrinis, I. Broustis, S.V. Krishnamurthy, S. Addepalli, K. Papagiannaki, Auto-configuration of 802.11n WLANs, in: ACM CoNext, 2010.
[18] E. Aryafar, N. Anand, T. Salonidis, E.W. Knightly, Design and experimental evaluation of multi-user beamforming in wireless LANs, in: ACM MobiCom, 2010.
[19] D.-S. Shiu, G. Foschini, M. Gans, J. Kahn, Fading correlation and its effect on the capacity of multielement antenna systems, IEEE Trans. Commun. 48 (3) (2000) 502–513.
[20] D. Tse, P. Viswanath, Fundamentals of Wireless Communication, Cambridge University Press, 2005.
[21] M. Pursley, D. Taipale, Error probabilities for spread-spectrum packet radio with convolutional codes and viterbi decoding, IEEE Trans. Commun. 35 (1) (1987) 1–12.
[22] V. Shrivastava, S. Rayanchu, J. Yoonj, S. Banerjee, 802.11n under the microscope, in: ACM IMC, 2008.
[23] M. Vutukuru, H. Balakrishnan, K. Jamieson, Cross-layer wireless bit rate adaptation, in: ACM SigComm, 2009.
[24] W.H. Xi, A. Munro, M. Barton, Link adaptation algorithm for the IEEE 802.11n MIMO system, in: Networking LNCS, 2008.
[25] C.-Y. Li, C. Peng, S. Lu, X. Wang, Energy-based rate adaptation for 802.11n, in: ACM Mobicom, 2012.
[26] R. Combes, A. Proutiere, D. Yun, J. Ok, Y. Yi, Optimal rate sampling in 802.11 systems, in: IEEE Infocom, 2014.
[27] F. Peng, J. Zhang, W.E. Ryan, Adaptive modulation and coding for IEEE 802.11n, in: IEEE WCNC, 2007.
[28] W. Kim, O. Khan, K. Truong, S.-H. Choi, R. Grant, H. Wright, K. Mandke, R. Daniels, R. Heath, S. Nettles, An experimental evaluation of rate adaptation for multi-antenna systems, in: IEEE Infocom, 2009.

Lara Deek is a postdoctorate researcher in the Electrical and Computer Engineering Department at the University of Illinois at Urbana-Champaign. She received her PhD in Computer Science under the guidance of Professors Elizabeth Belding and Kevin Almeroth at the University of California, Santa Barbara. Her undergraduate degree was a BE in Computer and Communications Engineering from the American University of Beirut. Her research is focused on designing resource-efficient wireless systems for emerging wireless networks. She is broadly interested in mobile and wireless network technologies, deployment, measurement, protocol design, and implementation.

Eduard Garcia-Villegas received his MSc and PhD from the Universitat Politècnica de Catalunya BarcelonaTech (UPC) in 2003 and 2010, respectively. He is an assistant professor at the same university and a member of the Wireless Networks Group (WNG). He also collaborates with the i2CAT Foundation and with the MOMENT Lab at UC Santa Barbara. His research interests include, but are not limited to IEEE 802.11 WLANs, radio resource management in wireless networks, wireless mesh networks, and future Internet architectures.

Elizabeth Belding is a Professor in the Department of Computer Science at the University of California, Santa Barbara. Her research focuses on mobile networking, specifically multimedia, monitoring, advanced service support, and solutions for developing and underdeveloped regions. She is the founder and director of the Mobility Management and Networking (MOMENT) Lab. She is the author of over 100 technical papers and has served on over 60 program committees for networking conferences. She is currently on the editorial board of the IEEE Pervasive Magazine and is an editor-at-large for IEEE Transactions on Networking. She is the recipient of an NSF CAREER Award, and a 2002 MIT Technology Review 100 award. She is an IEEE Fellow and an ACM Distinguished Scientist.



Sung-Ju Lee is an Associate Professor at the Computer Science Department of KAIST. Before joining KAIST in 2015, he spent 15 years in the Silicon Valley performing practical research for the networking industry, including at Hewlett-Packard Company and Narus, which is now part of Symantec. He received his PhD in Computer Science from University of California, Los Angeles (UCLA) in 2000. He has published over 100 technical papers in peer-reviewed journals and conferences. His papers are well-cited, with his publications receiving a total of over 9800 citations. He currently holds 29 US patents and 40-plus pending patents. He won the HP CEO Innovation Award in 2010. He is an IEEE Fellow and an ACM Distinguished Scientist.

Kevin Almeroth is currently a Professor in the Department of Computer Science at the University of California in Santa Barbara where his main research interests include computer networks and protocols, wireless networking, multicast communication, large-scale multimedia systems, and mobile applications. At UCSB, he is the former founding Associate Director of the Center for Information Technology and Society (CITS), a founding faculty member of the Media Arts and Technology (MAT) Program, Technology Management Program (TMP), and the Computer Engineering (CE) Program. In the research community, he has authored nearly 200 refereed papers and is heavily engaged in stewardship activities for a variety of research outlets including journal editorial boards, conference steering committees, new workshops, and the IETF. He is a Member of the ACM and a Fellow of the IEEE.