RAM: Rate Adaptation in Mobile Environments

Xi Chen, *Student Member*, *IEEE*,
Prateek Gangwal, *Student Member*, *IEEE*, and Daji Qiao, *Member*, *IEEE*

Abstract—Channel asymmetry and high fluctuation of channel conditions are two salient characteristics of wireless channels in mobile environments. Therefore, when using IEEE 802.11 devices in mobile environments, it is critical to have an effective rate adaptation scheme that can deal with these issues. In this paper, we propose a practical rate adaptation scheme called Rate Adaptation in Mobile environments (RAM) and implement it in the MadWifi device driver. RAM uses a receiver-based approach to handle channel asymmetry and a conservative SNR prediction algorithm to deal with high channel fluctuation. More importantly, RAM allows the receiver to convey the feedback information to the transmitter in a creative manner via ACK transmission rate variation, which does not require changes to the device firmware and hence is implementable at the device driver level. In addition, RAM adopts an effective scheme to guarantee that RAM-based and legacy IEEE 802.11 devices can interoperate with each other. The effectiveness of RAM is demonstrated via in-depth experimental evaluation in indoor static and mobile environments as well as outdoor vehicular environments.

Index Terms—IEEE 802.11 WLAN, rate adaptation, MadWifi.

1 Introduction

The increasing number of IEEE 802.11[1] devices have been used in various mobile applications. Since most resource management schemes for 802.11 devices are designed for static environments, 802.11-based systems may experience severe performance degradation in mobile environments, such as low throughput and high latency. This is due to the salient differences in wireless channel characteristics between static and mobile environments. For example, channel conditions in mobile environments usually exhibit more severe asymmetry and higher fluctuation than those in static environments.

In this paper, we study *rate adaptation* in mobile environments. Rate adaptation is one of the fundamental resource management issues for 802.11 devices. The goal is to maximize the throughput via exploiting the multiple transmission rates available for an 802.11 device and adjusting its transmission rate dynamically to the time-varying and location-dependent wireless channel condition. From the experiments, we find that most existing rate adaptation schemes cannot handle channel asymmetry or high fluctuation of channel conditions well, and hence may not be suitable for mobile environments. A few existing schemes may be able to deal with channel asymmetry but require changes to the CTS or ACK frame formats, which typically are hard coded in the device firmware. As a result, these schemes do not conform to the 802.11 standard and thus may not be implementable with commercial 802.11 devices, which limits their practical applications drastically.

1. For simplicity, we use 802.11 instead of IEEE 802.11 in this paper.

• The authors are with the Iowa State University, Coover Hall, Ames, IA 50011. E-mail: {leon6827, prateek, daji}@iastate.edu.

Manuscript received 26 June 2009; revised 7 Feb. 2011; accepted 12 Mar. 2011; published online 28 Apr. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-06-0255.
Digital Object Identifier no. 10.1109/TMC.2011.91.

We propose a practical rate adaptation scheme, called Rate Adaptation in Mobile environments (RAM), and demonstrate its effectiveness in both mobile and static environments via experiments and simulations. RAM has the following features:

- RAM is a practical rate adaptation scheme and we have implemented RAM in the MadWifi device driver [1].
- RAM is a receiver-based scheme and can deal with channel asymmetry well. Different from existing receiver-based rate adaptation schemes, RAM uses the variation of the ACK transmission rate to convey the feedback information implicitly. This means that RAM does not require changes to the CTS or ACK frame formats and, hence, can be implemented at the device driver level without modifying the device firmware. To the best of our knowledge, this is one of the first efforts in designing and implementing a receiver-based rate adaptation scheme that works with commercial 802.11 devices.
- RAM is an SNR-based scheme. To deal with high SNR fluctuation, RAM adopts a conservative SNR prediction algorithm to avoid overestimating future SNR values which may cause unnecessary frame losses and retransmissions.
- RAM uses an RTS window to regulate the usage of RTS frames to deal with hidden nodes. Comparing with RTS adaptation in existing rate adaptation schemes, RTS adaptation in RAM is designed based on a thorough examination of all possible transmission outcomes and the RTS window is updated in a timely manner.
- RAM adopts an effective scheme to guarantee that RAM-based and legacy 802.11 devices can interoperate with each other.

The rest of the paper is organized as follows: Section 2 discusses the related work on rate adaptation. Based on

TABLE 1
Classification of Existing Rate Adaptation Schemes

Schemes	Tx/Rx -based	Based on SNR	Window/ frame-based	Imple- mented	Deal with hidden
ARF/AARF	Tx	No	window	Yes	No
CARA-like	Tx	No	window	No	Yes
RRAA	Tx	No	window	Yes	Yes
SampleRate	Tx	No	window	Yes	No
CHARM	Tx	Yes	frame	Yes	No
Scheme in [9]	Tx	Yes	frame	No	No
SGRA	Tx	Yes	frame	Yes	No
ONOE	Tx	No	window	Yes	No
RBAR	Rx	Yes	frame	No	No
OAR	Rx	Yes	frame	No	No
RARA	Rx	Yes	frame	No	No
RAF	Rx	Yes	frame	No	Yes

experiments, a few observations about wireless channel conditions in mobile environments are presented in Section 3. Section 4 describes the design and implementation of the proposed RAM scheme. Experimental study and simulation-based performance evaluation are presented in Sections 5 and 6, respectively. The paper concludes in Section 7.

2 Related Work

Rate adaptation in static environments has been well studied in the past [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. As shown in Table 1, these rate adaptation schemes can be classified in the following ways: *transmitter-based* or *receiver-based*; *packet statistics-based* or *SNR-based*; *window-based* or *frame-based*.

Transmitter-based versus receiver-based. In *transmitter-based* schemes, the transmitter makes the rate selection decisions. By comparison, in *receiver-based* schemes, the receiver monitors the channel quality, makes the rate selection for the next frame transmission, and feeds the decision back to the transmitter.

Packet statistics-based versus SNR-based. Based on the information used to infer the channel condition, rate adaptation schemes can be classified as *packet statistics-based* and *SNR-based*. In packet statistics-based schemes, ARF [2], AARF [3] (also known as AMRR for its implementation in the MadWifi device driver), and CARA-like schemes [4], [5] use consecutive frame transmission failure and success counts as indicators of the channel quality. ONOE [11] and RRAA [6] calculate the frame loss ratio and RRAA compares it with certain thresholds to make rate updating decisions. SampleRate [7] chooses the rate with the shortest expected frame transmission time. In SNR-based schemes, CHARM [8], the scheme described in [9] and SGRA [10] are transmitter-based. They use the Receive Signal Strength Indictor (RSSI) values of the ACK frames received by the transmitter to infer the channel condition at the receiver side based on the assumption of a symmetric channel. In comparison, RBAR [12], OAR [13], and RARA [14] are receiver-based and they instead use the RSSI values of data frames received by the receiver. In RAF [15], the receiver determines the optimal transmission rate and frame size based on the measured interference level.

Window-based versus frame-based. Based on the rate updating period, rate adaptation schemes can be classified as *window-based* and *frame-based*. ARF, AARF, and CARA-like schemes use frame transmission failure and success counts and make rate adjustment when the number of frame transmission failures or successes is above a certain threshold. The window sizes for RRAA and SampleRate are 150 ms and 1 second, respectively, by default. Window-based schemes are reactive in nature as they rely on the past history to predict future channel conditions. Moreover, it is usually difficult to determine the optimal window size in dynamic environments when the channel condition varies often. In comparison, frame-based schemes adapt much faster to rapid variations of the channel condition that are often caused by fading and mobility.

In Table 1, we also list whether a rate adaptation scheme has already been implemented. To our knowledge, none of the receiver-based schemes has yet been implemented to work with commercial 802.11 devices. In fact, these receiver-based schemes likely are not implementable with commercial 802.11 devices for the following reasons. RBAR and OAR require modifications to the CTS (and possibly RTS) frame formats, which does not conform to the 802.11 standard, while the variation patterns of the ACK transmission rate proposed in RARA are not supported by commercial 802.11 devices and their device drivers.

In the presence of hidden nodes, it is difficult for the transmitter to differentiate channel-error-induced frame transmission failures from collision-induced ones, which may lead to pessimistic usage of the transmission rates. Adaptive usage of RTS/CTS has been recognized as an effective way to deal with hidden nodes, and it has been used in a few rate adaptation schemes such as CARA-like schemes and RRAA.

The scheme in [16] combines the sender- and receiver-based methods to adjust the rates of data as well as control frames. In [17], the authors propose a rate adaptation scheme for vehicular networks based on the context information such as distance and relative velocity. It is a history-based approach and requires repetitive training before usage. It is designed specifically for vehicles traveling along known routes. Therefore, this scheme may not be suitable for dynamic mobile environments where routes are not known a priori and the channel condition is unpredictable. In [18], the authors modify SampleRate for mobile environments by reducing the estimation window size.

3 Observations from Experiments

To design an effective rate adaptation scheme for mobile environments, it is critical to have a good understanding of the characteristics of wireless channels in mobile environments. To do so, we conduct experiments with two laptops equipped with CB9-GP-EXT 802.11a/b/g cards [19] in various indoor (static and mobile) and outdoor (vehicular) environments. Each laptop is loaded with the MadWifi device driver v0.9.4 [1] to measure and record the channel conditions. Details of the setup of the experiments can be found in Section 5.1. In this section, we present the observations and findings from the experiments.

3.1 Issues with Packet Statistics-Based Schemes

3.1.1 *Window-Based Rate Adaptation Schemes*

This type of schemes collects packet statistics within a time window (or a window of certain number of packets) and

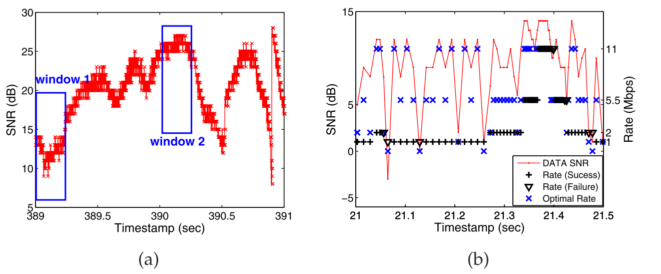


Fig. 1. Packet statistics-based rate adaption schemes are not suitable for mobile environments. (a) Issues with window-based schemes. (b) Issues with count-based schemes.

makes rate selection decisions at the end of the window. In mobile environments, since the channel condition fluctuates frequently, which will be discussed in Section 3.3, packet statistics collected at the current window may become obsolete when making rate selection decisions for future transmission attempts. Fig. 1a shows a trace of DATA SNR values in an experimental run for an outdoor vehicular scenario. It can be seen from the figure that it would be too pessimistic or optimistic to use the packet statistics collected from window 1 or window 2 to select rates for future packet transmissions. Another issue with window-based rate adaptation schemes lies in the selection of a proper window size. If the window size is too large, some of the collected information may become outdated at the end of the window, while if it is too small, the collected statistics may not be accurate enough.

3.1.2 *Count-Based Rate Adaptation Schemes*

ARF and CARA-like schemes use consecutive frame transmission failure or success counts to select the rate for the next transmission attempt. They increase the rate after 10 consecutive transmission successes and decrease the rate upon two consecutive failures. This type of approaches may not work effectively in mobile environments with high SNR fluctuations. As shown in Fig. 1b, the success count of 10 may be too conservative for rate increasing, and the failure count of two may be too pessimistic for rate decreasing, which may lead to potential underutilization of the channel.

3.2 Severe Channel Asymmetry

One interesting observation from our experiments is the severe channel asymmetry in practical scenarios. As shown in Figs. 2 and 3, ACK SNR values collected at the transmitter usually differ significantly from DATA SNR values collected at the receiver. The SNR difference is as high as 15 dB in some outdoor vehicular scenarios. Since channel symmetry is one of the key assumptions in several existing transmitter-based rate adaptation schemes such as CHARM and SGRA, these schemes may not be suitable for mobile environments. Instead, receiver-based approaches may be a better option.

3.3 High SNR Fluctuation

High SNR fluctuation is another important observation from our experiments. In some traces such as the one shown in Fig. 4a, the differences between consecutive SNR values are as large as 10 dB. From the experiments, we notice that high SNR fluctuation usually occurs when the environment suddenly changes, e.g., opening or closing a door, sudden acceleration of the vehicle, and vehicle making a turn. Fig. 4b plots the histogram of the SNR values shown in Fig. 4a and we can see that the distribution of SNR values is quite irregular.

SNR prediction algorithms used in existing rate adaptation schemes may not be able to handle the high SNR fluctuation properly. As shown in Figs. 4a and 4c, the Light Weighted Moving Average (LWMA) scheme used in

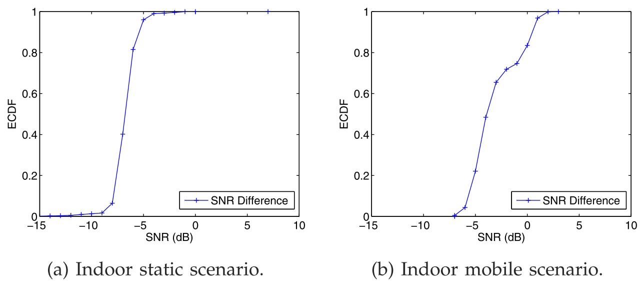


Fig. 2. Empirical Cumulative Distribution Function (ECDF) of the difference between DATA and ACK SNR values in indoor static and mobile scenarios.

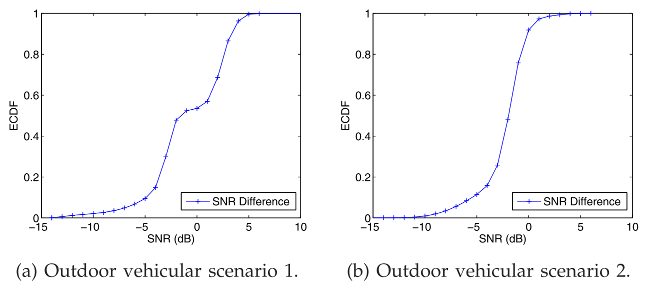
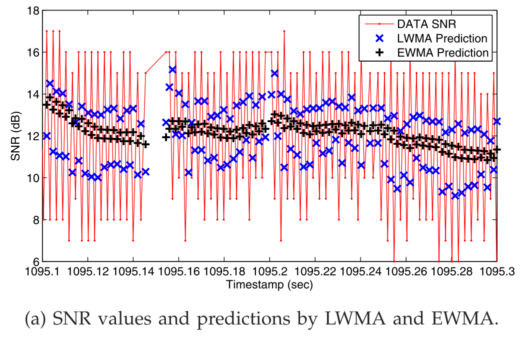


Fig. 3. ECDF of the difference between DATA and ACK SNR values in outdoor vehicular scenarios.



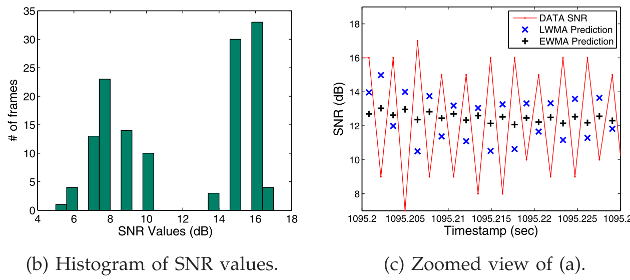


Fig. 4. High SNR fluctuation in mobile environments.

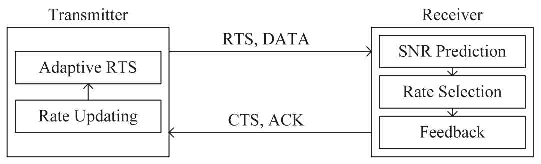


Fig. 5. Overall structure of RAM.

CHARM almost always uses the previous SNR value as the prediction for the next SNR value, which results in a large number of overestimations of SNR values and hence frame transmission failures. Similar problem exists for the simple Exponentially Weighted Moving Average (EWMA) scheme as well.

4 Design and Implementation of RAM

To deal with the issues discussed in the previous section, we propose a practical rate adaptation scheme, called RAM that can be implemented with commercial 802.11 devices. As shown in Fig. 5, RAM has the following components—at the receiver side: 1) SNR prediction, 2) rate selection based on SNR prediction, and 3) feedback of rate selection to the transmitter; and at the transmitter side: 1) rate updating and 2) adaptive usage of RTS/CTS. Interoperability between RAM-based and legacy 802.11 devices will be discussed at the end of the section.

4.1 Receiver: SNR Prediction

To deal with high SNR fluctuation and irregular SNR distribution, we propose a simple conservative SNR prediction algorithm as follows: it maintains the moving averages of the SNR values as well as the deviations to the average SNR value:

$$\begin{cases} S_{\text{avg}} = (1-\delta)\cdot S_{\text{avg}} + \delta \cdot S_{\text{curr}}, \\ DEV_{\text{avg}} = (1-\rho)\cdot DEV_{\text{avg}} + \rho \cdot |S_{\text{curr}} - S_{\text{avg}}|, \end{cases} \tag{1}$$

and predicts the SNR value for the next frame as

$$S_{\text{est}} = S_{\text{avg}} - \eta \cdot DEV_{\text{avg}}, \tag{2}$$

where S_{curr} is the SNR value reported by MadWifi upon each frame reception[2] and δ, ρ, η are design parameters. We set $\delta = \rho = 0.1$ and $\eta = 1$ in RAM, and the reason for choosing these values will be discussed in the next section. In comparison, both EWMA and LWMA predict the SNR value without considering the deviation of recent SNR values. They work as follows:

$$S_{\text{est}} = S_{\text{avg}} = (1-\delta)\cdot S_{\text{avg}} + \delta \cdot S_{\text{curr}}. \tag{3}$$

For EWMA, δ is a fixed smoothing factor. For LWMA, δ is adjusted during the runtime: $\delta = \frac{1}{1+f(T)}$ where T is the time interval between consecutive transmissions and $f(T)$ is a linearly decreasing function of T, starting at 1 and decreasing to 0 when T exceeds a decision time window (2 seconds).

2. Ideally, the SNR value of each received frame should be used as S_{curr} in the algorithm. Unfortunately, the current MadWifi does not support per-frame-based SNR measurement. While it measures the received signal level for each frame, it only updates the noise level upon each interrupt and usually multiple frames are served between interrupts [20]. Therefore, strictly speaking, the SNR value reported by MadWifi upon each frame reception is *not* the exact SNR value of the frame *but* an approximation to it. Nevertheless, even with such limitation of the current MadWifi, RAM still yields a noticeable performance improvement over existing rate adaptation schemes, which will be shown in later sections via both experimental and simulation results.

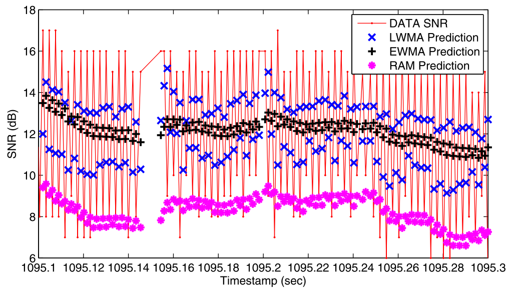


Fig. 6. An example of our SNR prediction algorithm.

By considering the deviation of recent SNR values when making the prediction, our algorithm can deal with high SNR fluctuation well. It is designed to predict future SNR values as accurately as possible without overestimating them. This can be seen from an example shown in Fig. 6 where the predictions by our algorithm follow the lower envelop of the SNR variation closely.

4.1.1 *Effects of RAM Parameters*

The values of three RAM parameters—η, δ, and ρ-are chosen based on the results of our trace-based ns-2 simulation. We import the SNR traces from experiments using a time stamp-based approach. Basically, according to the time stamp of a packet, we set its SNR value based on the collected traces. We collect SNR traces from four experimental scenarios: Walk-1, Walk-2, SlowDrive-1, and FastDrive-1. The details of these scenarios can be found in Section 5.1 and Table 5. For each scenario, we collect five different SNR traces. Other simulation setups are the same as those in Section 6.1.

We have simulated various combinations of η, δ, and ρ values, and selected results are shown in Fig. 7, where each point is averaged over five traces. We can clearly see that a small η does not perform well as it would lead to overestimation of the channel condition and hence more frame failures, retransmissions, and backoffs. On the other hand, a large η does not perform well either as it would lead to underestimation of the channel condition and hence waste of the good channel condition. For the two smoothing factors δ and ρ that are used in the moving average

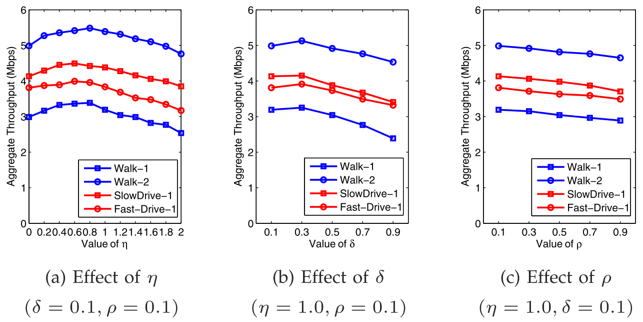


Fig. 7. Effects of RAM parameters.

predictions of S_{avg} and DEV_{avg}, respectively, a larger value would lead to a higher fluctuation in prediction, which makes S_{est} more fluctuated. Overall, simulation results show that RAM yields better performance when η is between 0.5 and 1, and δ and ρ are between 0.1 and 0.3; there is no significant performance difference when they vary in these ranges. For this reason, we set $\eta = 1.0$ and $\delta = \rho = 0.1$ in RAM.

4.2 Receiver: Rate Selection Based on SNR Prediction

To select the proper rate for the next frame transmission to maximize the throughput, RAM maintains a throughput-versus-(rate, SNR) table. For each ($\text{rate} = R$, $\text{SNR} = S$) pair in the table, we use $G(R, S)$ to denote the expected throughput when the frame is transmitted at rate R and its SNR at the receiver is S. The table is updated upon each data frame reception. In order to respond quickly in the presence of bursty strong interference in the network, our table updating scheme uses a moving average approach:

$$G_{\text{avg}}(R, S) = (1 - \gamma) \cdot G_{\text{avg}}(R, S) + \gamma \cdot G_{\text{inst}}(R, S), \tag{4}$$

where $G_{\text{inst}}(R, S)$ is the instant throughput for each distinct (R, S) pair observed between two consecutive data frame receptions. Then, based on the predicted SNR value (S_{est}), the receiver looks up the table and selects the rate for the next frame transmission as follows:

$$R^* = \arg\max_R G_{\text{avg}}(R, S_{\text{est}}). \tag{5}$$

We implement RAM in the MadWifi device driver, which employs the *multirate retry (MRR) mechanism* to transmit a data frame. In RAM, transmitter and receiver agree on the multirate retry mechanism and receiver may use this knowledge to update the throughput-versus-(rate, SNR) table. Before proceeding to the details about how the table is updated, we first give a brief introduction about the multirate retry mechanism.

4.2.1 Multirate Retry Mechanism

In MadWifi, whenever a frame is ready to send, MadWifi can specify up to four different rates ($r_1 > r_2 > r_3 > r_4$) along with their maximum retry counts ($c_i, i = 1, \ldots, 4$) for the frame and pass these information to the card firmware along with the frame. The frame is discarded after ($c_1 + c_2 + c_3 + c_4$) unsuccessful transmission attempts, i.e., c_i attempts at the rate of r_i. In MadWifi, the maximum total number of transmission attempts for a frame is 10, meaning that $(c_1 + c_2 + c_3 + c_4) \leq 10$. The card firmware reports the total number of transmission attempts to MadWifi after the frame has been transmitted successfully or discarded. In RAM, we set $c_1 = 4, c_2 = c_3 = c_4 = 2$, and r_{i+1} to be the next lower rate to r_i ($i = 1, 2, 3$). The reason for setting a larger retry count ($c_1 = 4$) for the first rate r_1 is because RAM adopts a conservative SNR prediction algorithm and hence r_1 usually has already been selected conservatively. Meanwhile, c_1 should not be set too large; this is to avoid unnecessary retransmissions caused by prediction errors. More discussions on the settings of retry counts may be found in [3] and [8].

4.2.2 Updating the Throughput-versus-(Rate, SNR) Table

When the receiver receives a data frame successfully, there are two possible outcomes for each of the unsuccessful transmission attempts (if any) prior to the successful reception of the frame: *frame was corrupted but header can be retrieved by the receiver*, or *frame was completely lost*. For the former case, the information about the unsuccessful transmission attempt such as the transmission rate and the SNR of the frame are reported to MadWifi by the card firmware, while for the latter case, this information is not available. Interestingly, our experiments show that the latter case rarely occurs in practice and we hence always assume the former case when updating the throughput-versus-(rate, SNR) table. Note that the SNR values reported by the card firmware are integer values, meaning that the number of entries in the table is a small finite number.

Suppose a data frame is received successfully during the kth attempt at rate R_k and SNR of S_k, where $1 \leq k \leq (c_1 + c_2 + c_3 + c_4)$. For each of the unsuccessful attempts, the receiver retrieves the rate and SNR information reported by the card firmware. Then, we have the following:

$$\begin{cases} L(R_j, S_j) = 0, & \text{for } 1 \leq j < k, \\ L(R_j, S_j) = data\_payload, & \text{for } j = k, \end{cases} \tag{6}$$

and

$$T(R_j, S_j) = txtime(R_j) + backoff(j), \quad \text{for } 1 \leq j \leq k, \tag{7}$$

where $L(R_j, S_j)$ and $T(R_j, S_j)$ are the amount of data received successfully at rate R_j and SNR of S_j, and the transmission time for such a frame, respectively. Here, *data_payload* is the data payload of the frame, $txtime(R_j)$ is the frame transmission duration at rate R_j, and $backoff(j)$ is the average backoff time prior to the jth transmission attempt, which is given by

$$backoff(j) = \min\left[\frac{(\text{CW}_{\min} + 1) \cdot 2^{j-1} - 1}{2}, \frac{\text{CW}_{\max}}{2}\right] \times \text{aSlotTime}. \tag{8}$$

Then, the instant throughput for each distinct (R, S) pair observed during k transmission attempts is calculated by

$$G_{\text{inst}}(R, S) = \frac{\sum_{(R,S)==(R_j,S_j)} L(R_j, S_j)}{\sum_{(R,S)==(R_j,S_j)} T(R_j, S_j)}, \tag{9}$$

and the corresponding $G_{\text{avg}}(R, S)$ is updated using (4).

4.3 Receiver: Feedback of Rate Selection to the Transmitter

The 802.11 standard [21] specifies that the ACK frame should be transmitted at the highest rate in the basic rate set that is less than or equal to the transmission rate of the data frame it is acknowledging. We call such ACK transmission rate the *default ACK rate*. For example, the 802.11g basic rate set is {1, 2, 5.5, 11, 6, 12, 24} Mbps. So if a data frame is transmitted at 18 Mbps, the default rate of the corresponding ACK frame is 12 Mbps. In practice, MadWifi allows two different transmission rates for ACK frames, as listed in Table 2 for Atheros chipset-based 802.11g cards. MadWifi

TABLE 2
In MadWifi: Two Rates Available for ACK Frames

data rate (Mbps)	1	2	5.5	11	6	9	12	18	24	36	48	54
low ACK rate	1	2	2	2	6	6	6	6	6	6	6	6
high ACK rate	1	2	5.5	11	6	6	12	12	24	24	24	24

can specify that an ACK frame is transmitted at a low rate or a high rate (the default rate) via setting different values of a special register [22].

RAM takes advantage of this MadWifi feature and conveys the feedback information implicitly via the ACK transmission rate variation.[3] Specifically, if the receiver wants to inform the transmitter to send the next frame at the same rate as the previous frame, or at the next higher rate, it transmits the ACK frame at the default high rate or the low rate, respectively. For example, if the receiver receives a data frame successfully at 36 Mbps, it can signal the transmitter to send the next frame at 36 or 48 Mbps by transmitting the ACK frame at 24 or 6 Mbps, respectively.

Note that for rates of 1, 2, 6, and 9 Mbps, there is only one option for the ACK transmission rate. In RAM, we disable the data transmission rates of 6 and 9 Mbps since it has been observed from experiments that the throughput performances of 6 and 9 Mbps are worse than that of 5.5 Mbps [8]. For rates of 1 or 2 Mbps, rate increasing decisions are made at the transmitter side, which will be explained in the next section, along with rate decreasing decisions which are always made at the transmitter side regardless of the current rate.

4.4 Transmitter: Updating the Transmission Rate

As described in Section 4.2.1, once r_1 is decided for a data frame, the multirate retry mechanism for the frame is decided. In RAM, we decide r_1 for the next data frame according to the transmission result of the *last* attempt (suppose at the rate of R_{last}) of the previous data frame:

1. *If it fails.* The transmitter sets r_1 to R_{last}.
2. *If it succeeds.* The transmitter may take the following actions depending on R_{last}:

   a. If $R_{\text{last}} > 2$ Mbps, the transmitter relies on the feedback from the receiver to set the rate for the next frame. Specifically, if the transmitter receives an ACK frame at the default high rate, $r_1 = R_{\text{last}}$; otherwise, r_1 is set to the next higher rate to R_{last}.

   b. If $R_{\text{last}} = 1$ or 2 Mbps, the transmitter makes the rate updating decision using the following heuristic. In RAM, the transmitter keeps track of the ACK SNR values. When the current ACK SNR is 5 dB higher than the previous one or when the number of consecutive frame transmission successes is larger than 4, r_1 is set to the next higher rate to R_{last}. In an extreme case when $R_{\text{last}} = 1$ Mbps and the current ACK SNR is 9 dB larger than the previous one, r_1 is increased to 5.5 Mbps directly. These thresholds are obtained from the experiments.

3. RAM-like rate adaptation solutions may be designed for other wireless communication devices as long as the device driver supports similar functions as MadWifi, such as dual ACK transmission rates.

TABLE 3
Data Transmission Attempts, Outcomes, and Corresponding Actions on Updating RTSWnd

Data Frame Transmission Attempt	Outcome of Transmission Attempt	Action on Updating RTSWnd
DATA with RTS	RTS Fail	RTSWnd = 3×RTSWnd
	RTS Succ, DATA Fail	RTSWnd = RTSWnd − 1
	RTS Succ, DATA Succ	
DATA without RTS	DATA Fail	RTSWnd = RTSWnd+1
	DATA Succ	RTSWnd = RTSWnd/2

By default, MadWifi uses the high ACK rate to calculate the NAV value for a data frame transmission. In RAM, since the receiver may transmit an ACK frame at the low rate to signal rate increasing for the next data frame, we modify the NAV calculation in MadWifi by using the low ACK rate instead. This can be done by modifying the value of a special register [22]. Since ACK frames are short, such modification does not affect the performance much, which will be discussed in Section 6.3. In addition, since the RAM receiver uses a moving average to update the SNR estimation and feeds back the rate selection decisions to the transmitter on a per-frame basis, the RAM transmitter is able to converge quickly (usually a few frames) to the proper transmission rate even when the interval between two consecutive frame transmissions is large.

4.5 Transmitter: Adaptive Usage of RTS/CTS

Adaptive usage of RTS/CTS has been recognized as an effective way to deal with hidden nodes in 802.11 networks and it has been used in several rate adaptation schemes such as CARA-like schemes and RRAA. We propose an advanced adaptive RTS scheme in RAM. Similar to the one used in RRAA, our adaptive RTS scheme uses an RTS window (with the size of RTSWnd) to regulate the usage of RTS frames. All data frames within the RTS window shall be transmitted with RTS/CTS support. Moreover, our scheme examines all possible transmission outcomes thoroughly and updates RTSWnd in a timely manner as follows.

Table 3 lists two ways of attempting a data frame transmission (i.e., with or without RTS/CTS), possible outcomes of each attempt, and the corresponding actions on updating RTSWnd. Initially, RTSWnd is set to zero to disable RTS usage. The basic heuristic behind our adaptive RTS scheme is that RTSWnd should increase more quickly if a shorter frame is lost (which implies a higher collision probability), and decrease more quickly if a longer frame succeeds (which implies a lower collision probability). Since an RTS frame is short, an RTS failure indicates that the collision problem may be severe. Hence, we multiply RTSWnd by 3. If an RTS transmission succeeds, we decrease RTSWnd slowly $(\text{RTSWnd} = \text{RTSWnd} - 1)$ regardless whether the subsequent data transmission succeeds or not. This is because an RTS/CTS exchange has already reserved the channel and reduces the probability of collision to the subsequent data transmission. On the other hand, without RTS/CTS support, a successful data transmission (usually with a long transmission duration) indicates a small chance of collision and hence we decrease RTSWnd by half. When a data transmission fails with no preceding

RTS, the cause of the failure is unclear. In this situation, we increase RTSWnd slowly ($\mathrm{RTSWnd} = \mathrm{RTSWnd} + 1$). We also set a maximum value of 32 for RTSWnd to guarantee stable performance.

Note that in MadWifi, it is impossible to control the RTS usage on a per-transmission-attempt basis. Therefore, when we implement RAM in MadWifi, the RTS usage is controlled on a per-frame basis. In other words, the transmitter updates RTSWnd when it receives the report from the card firmware after the frame has been transmitted successfully or discarded.

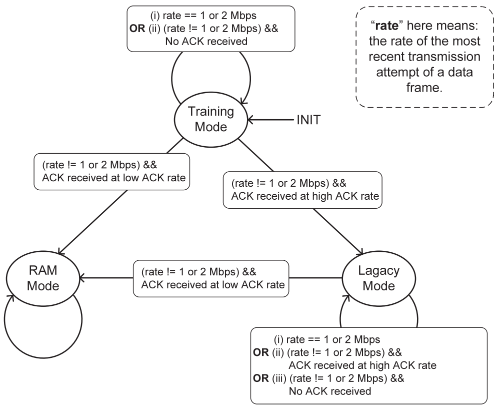
4.6 Interoperability between RAM-Based and Legacy 802.11 Devices

Given the fact that numerous legacy 802.11 devices have been deployed and will continue to be used in many real-world Wi-Fi applications, it is important to make sure that RAM-based and legacy 802.11 devices can interoperate with each other. We propose an effective scheme in RAM to achieve this goal. Before proceeding to the details of the scheme, we first introduce three modes that a RAM device may operate in: *Legacy mode*, *RAM mode*, and *Training mode*. Note that 1) a RAM device maintains a state transition diagram for each of its communicating partners, meaning that it could operate in different modes to communicate with different partners; and 2) a legacy device always operates in Legacy mode.

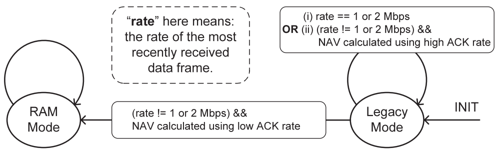
- *Legacy mode*. The transmitter in Legacy mode adopts a non-RAM transmitter-based rate adaptation scheme, e.g., SampleRate. Also, it calculates the NAV value using the default high ACK rate. On the other hand, the receiver in Legacy mode always replies ACK at the default high ACK rate.
- *RAM mode*. The transmitter in RAM mode adopts the mechanism mentioned in Section 4.4 to update the transmission rate, and calculates the NAV value using the low ACK rate. The receiver in RAM mode replies ACK at the high or low ACK rate according to the schemes described in Sections 4.1, 4.2, and 4.3.
- *Training mode*. Training mode is a transient operation mode and it will switch to RAM or Legacy mode eventually. The transmitter in Training mode adopts a non-RAM transmitter-based rate adaptation scheme, but calculates the NAV value using the low ACK rate.

The state transition diagram for RAM transmitters is shown in Fig. 8a. As shown in the figure, a RAM transmitter always starts at Training mode. In Training mode, the transmitter uses a transmitter-based rate adaptation scheme to update the rate. Meanwhile, it sets the NAV using the low ACK rate to announce that it supports RAM. Note that in Table 2, the high and low ACK rates are identical for the data rates of 1 and 2 Mbps. Therefore, the transmitter in Training mode will not switch to other modes if the data rate is 1 or 2 Mbps. Besides, if no ACK frame is received, the transmitter in Training mode also will not trigger a state transition.

In Training mode, when the transmitter receives an ACK frame successfully for a data frame sent at a rate other than 1 and 2 Mbps, a state transition is triggered as follows: if the


(i) rate == 1 or 2 Mbps
OR (ii) (rate != 1 or 2 Mbps) && No ACK received
"rate" here means: the rate of the most recent transmission attempt of a data frame.
Training Mode
INIT
(rate != 1 or 2 Mbps) && ACK received at low ACK rate
(rate != 1 or 2 Mbps) && ACK received at high ACK rate
RAM Mode
(rate != 1 or 2 Mbps) && ACK received at low ACK rate
Legacy Mode
(i) rate == 1 or 2 Mbps
OR (ii) (rate != 1 or 2 Mbps) && ACK received at high ACK rate
OR (iii) (rate != 1 or 2 Mbps) && No ACK received

(a) State transition diagram for RAM transmitters.


"rate" here means: the rate of the most recently received data frame.
(i) rate == 1 or 2 Mbps
OR (ii) (rate != 1 or 2 Mbps) && NAV calculated using high ACK rate
RAM Mode
(rate != 1 or 2 Mbps) && NAV calculated using low ACK rate
Legacy Mode
INIT

(b) State transition diagram for RAM receivers.

Fig. 8. State transition diagrams for RAM devices.

ACK frame is transmitted at the low ACK rate, the transmitter knows that its communicating partner must be a RAM device and thus switches to RAM mode immediately. On the other hand, if the ACK frame is transmitted at the default high ACK rate, the transmitter considers its communicating partner a legacy device and then switches to Legacy mode, during which it keeps on monitoring the status of future ACK frames. Any time it sees an ACK frame transmitted at the low ACK rate for a data frame sent at a rate other than 1 or 2 Mbps, the transmitter switches to RAM mode; otherwise, it remains in Legacy mode.

The state transition diagram for RAM receivers is shown in Fig. 8b. As shown in figure, a RAM receiver always starts at Legacy mode. If it receives a data frame with NAV value calculated using the low ACK rate and the data rate is not 1 or 2 Mbps, it realizes that its communicating partner must be a RAM device and then switches to RAM mode immediately. Otherwise, it remains in Legacy mode.

5 Experimental Study

In this section, we evaluate the effectiveness of RAM using experimental results. We implement all the RAM modules described in Section 4 in MadWifi. We call this complete version of RAM implementation RAM-FULL. For comparison purpose, we also implement another version of RAM, called RAM-BASIC, which does not include the adaptive RTS module.

5.1 Experimental Setup

The hardware and software configurations used in our experiments are listed in Table 4. All experiments are conducted using Dell Latitude D620 laptops equipped with CB9-GP-EXT 802.11a/b/g WLAN adaptors, which embed Atheros 5213 chipsets. We use off-the-shelf hardware instead of sophisticated equipments to conduct

TABLE 4
Configuration Parameters

Parameters	Values
Computer	Dell Latitude D620 Laptop
Operating system	Linux Kernel 2.6.24-16
WLAN adaptor	CB9-GP-EXT 802.11a/b/g
Device driver	MadWifi v0.9.4
802.11 PHY	802.11g
Transmit Power	16 dBm
CBR packet size	1470 octets
CBR rate	30 Mbps

TABLE 5
Description of Experimental Scenarios

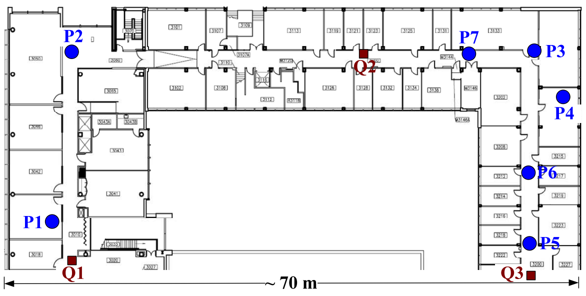
Scenarios	Descriptions
Static-1	STA1 at P2, STA2 at P1
Static-2	STA1 at P3, STA2 at P5
Static-3	STA1 at P3, STA2 at P2
Static-4	STA1 at P7, STA2 at P6
Walk-1	STA1 at P3, STA2: Q2→Q1→Q2
Walk-2	STA1 at P3, STA2: P3→Q3→P3
Walk-3	STA1 at P4, STA2: P6→Q2→P6
Walk-4	STA1 at P6, STA2: P3→Q2→P3
SlowDrive-1	STA1 is static, STA2 moves along the line up to 25 MPH
SlowDrive-2	STA1 is static, STA2 moves along the curve up to 25 MPH
FastDrive-1	STA1 is static, STA2 moves along the line up to 45 MPH
FastDrive-2	STA1 is static, STA2 moves along the curve up to 45 MPH

experiments as this makes our experimental results comparable to what users of ordinary 802.11 devices may expect in realistic scenarios. For outdoor vehicular experiments, we use 7 dBi omnidirectional external antennas, which are shown in Fig. 9b.

In our experiments, we use Iperf [23] as the UDP packet generator. Constant Bit Rate (CBR) traffic is generated at 30 Mbps with packet size of 1,470 octets. The results for each scenario are averaged over five experimental runs. In order to minimize potential unexpected performance variation caused by people's movement and interference from other 802.11 devices, indoor experiments are conducted at nighttime or weekends and outdoor experiments are conducted in the mornings during weekends.

We conduct experiments in both static and mobile scenarios. Indoor experiments (static and mobile) are performed on the third floor of Coover Hall (our department building), as shown in Fig. 9a, and outdoor vehicular experiments are performed in a parking lot near Jack Trice stadium, as shown in Fig. 9b. We mark several locations and moving trajectories on the figures, based on which we design 12 different experimental scenarios, as described in Table 5.

We compare the throughput performances of RAM-FULL and RAM-BASIC against the following state-of-the-art rate adaptation schemes: SampleRate [7], AMRR [3], ONOE [11], ARF [2], RRAA [6], and CHARM [8]. While SampleRate, AMRR, and ONOE have already been implemented in MadWifi, ARF, RRAA, and CHARM have not or their codes are not available. Therefore, we implement these three schemes in MadWifi from scratch. Moreover, we do not consider CARA [4] in the performance evaluation because per-transmission-attempt RTS probing and CCA detection proposed in CARA are infeasible in MadWifi.



(a) Venue for indoor experiments: 3rd floor of Coover Hall.

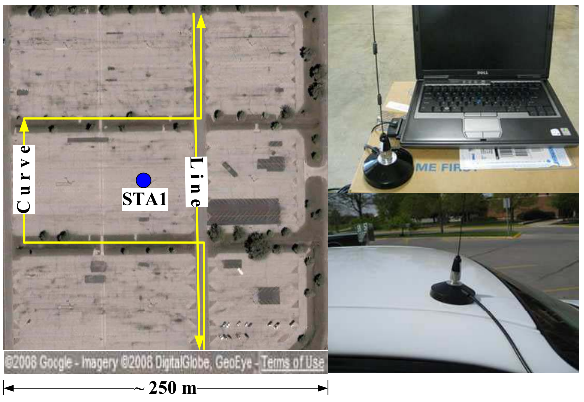

Curve
Line
STA1
©2008 Google - Imagery ©2008 DigitalGlobe, GeoEye - Terms of Use
~ 250 m
(b) Venue for outdoor experiments: a parking lot near Jack Trice stadium.

Fig. 9. Venues for our indoor and outdoor experiments.

5.2 Implementation Details

In this section, we describe the implementation details of testing schemes mentioned above.

5.2.1 *SampleRate, ONOE, and AMRR*

SampleRate, ONOE, and AMRR are three existing rate adaptation schemes available in MadWifi. In the following, we give a brief introduction on how they are implemented in MadWifi.

SampleRate tries to maximize the throughput by estimating per-frame transmission time for each rate and selecting the rate with the lowest expected per-frame transmission time. In MadWifi, SampleRate uses EWMA with a smoothing factor of 0.05 to update the average transmission time for each rate, and adjusts the rate after each second. As for the multirate retry mechanism, SampleRate attempts transmitting a data frame at the selected rate for seven times and then at 1 Mbps for three times, i.e., $c_1 = 7, c_2 = 3$, and $r_2 = 1$ Mbps.

ONOE is less sensitive to individual frame failures. It collects the number of successful frame transmissions (ϕ_{succ}) and the total number of transmission attempts (ϕ_{total}) during a 1-second period. If all frames need retry on average (i.e., $\phi_{\text{total}} > 2\ \phi_{\text{succ}}$), the credit for the current transmission rate is decremented by 1. On the other hand, if less than 10 percent of frames need retry on average (i.e., $\phi_{\text{total}} < 1.1\ \phi_{\text{succ}}$), the credit for the current transmission rate is incremented by 1. The rate is increased when the credit for the current transmission rate is larger than 10.

AMRR is the MadWifi version of AARF which is an adaptive variant of the well-known ARF scheme. AMRR uses the success and failure counts of frame transmission attempts when making the rate adaptation decisions.

TABLE 6
Parameter Values in Our RRAA Implementation

Rate (Mbps)	P_{ORI} (%)	P_{MTL} (%)	*ewnd*	Rate (Mbps)	P_{ORI} (%)	P_{MTL} (%)	*ewnd*
1	50.0	N/A	6	12	18.61	28.68	20
2	31.25	52.5	6	18	13.25	37.22	20
5.5	18.75	62.5	6	24	16.81	26.5	40
11	11.75	37.5	6	36	11.5	33.63	40
6	25.0	35.0	6	48	4.7	23.0	40
9	14.34	39.32	10	54	N/A	9.4	40

Compared with ARF, AMRR uses binary exponential backoff to adjust the success count threshold. As a result, when the channel condition is stable, unnecessary probes of the channel status at a higher rate may be avoided. However, in MadWifi, since the MRR mechanism is adopted, it is infeasible to collect the accurate success and failure counts because the transmission result of a frame is only reported to MadWifi after a number of transmission attempts for the frame and this number varies for different frames. For this reason, AMRR sets $c_1 = c_2 = c_3 = c_4 = 1$ for MRR and defines "success count" in a different way: if the number of attempts at rate r_2 is less than 10 percent of the number of attempts at rate r_1 in a time window of 500 ms, it is considered a success count of 1 at rate r_1. Similarly, if the number of attempts at rate r_2 is larger than one third of the number of attempts at rate r_1 in a time window of 500 ms, it is considered a "failure count" of 1 at rate r_1. This means that AMRR may only increase the rate after at least 5 seconds ($= 500 \text{ ms} \times 10$) and decrease the rate after at least 1 second ($= 500 \text{ ms} \times 2$).

5.2.2 Implementation of ARF and RRAA

We have implemented three other testing schemes in MadWifi: ARF, RRAA, and CHARM. We will discuss the implementation details of ARF and RRAA in this section and CHARM in the next section.

For ARF, we disable the MRR mechanism by setting $c_1 = 1$ and $c_2 = c_3 = c_4 = 0$. This way, we can collect the accurate success and failure counts of frame transmission attempts, which is different from AMRR. A 2-second timer is started when two consecutive transmissions fail. When either the timer expires or the number of successfully received acknowledgments reaches 10, the transmission rate is increased.

For RRAA, we monitor the loss ratio for the current rate during a short time window of 150 ms. At the end of the window, we compare the loss ratio with predefined thresholds to make the rate change decision. In RRAA, each rate is associated with three parameters: an estimation window size (*ewnd*, in terms of number of frames), a Maximum Tolerable Loss threshold (P_{MTL}), and an Opportunistic Rate Increase threshold (P_{ORI}). The authors of RRAA only specify the parameter values for 802.11a rates in [6]. In our implementation, by following the same approach as in [6], we set the parameters for 802.11g rates, which are listed in Table 6.

5.2.3 Implementation of CHARM

In [8], the authors of CHARM introduce the Rate SNR Threshold Estimation module without, however, giving the implementation details. CHARM maintains a table for each rate where every entry (called *bin*) of the table corresponds to an integer SNR value (in dB) and contains the numbers of successful and failed transmissions at this SNR level. Each rate is associated with an SNR threshold. Ideally, all frame transmissions at a certain rate should succeed (fail) if the SNR is higher (lower) than the SNR threshold of the rate.

In our implementation of the Rate SNR Threshold Estimation module, we define two variables, *thresh_inc_count* and *thresh_dec_count*, for each rate and use them to determine how to adjust the SNR threshold for the rate. Specifically, for each rate, we periodically (every second) examine all the bins in its table and do the following:

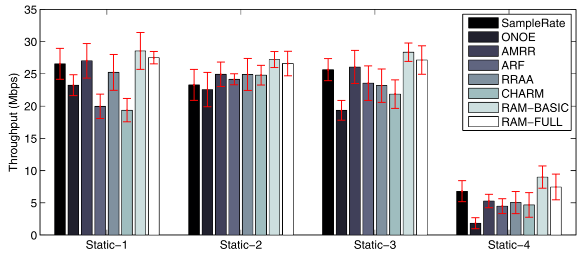
- For each SNR level that is lower than the current SNR threshold of the rate, if the frame success ratio indicates a good performance (larger than 80 percent in our implementation) of the rate at this SNR level, *thresh_dec_count* is incremented by 1. The rationale behind this is that when most transmissions at this rate succeed at an SNR level lower than its current SNR threshold, the SNR threshold should be decreased.
- For each SNR level that is higher than the current SNR threshold of the rate, if the frame success ratio indicates a bad performance (less than 20 percent in our implementation) of the rate at this SNR level, *thresh_inc_count* is incremented by 1. The rationale behind this is that when most transmissions at this rate fail at an SNR level higher than its current SNR threshold, the SNR threshold should be increased.
- If *thresh_inc_count* is larger or less than *thresh_dec_count*, the SNR threshold of the rate is increased or decreased by 1 dB.

5.2.4 Implementation of RAM

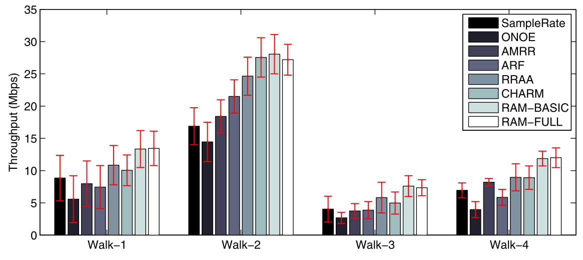
We will discuss the following two implementation details of RAM: 1) how the ACK rate variation is achieved in MadWifi; and 2) how the duration field value is set in MadWifi. In general, these are done by tuning special Atheros register values which are listed in [22].

For point 1, there is a special Atheros register called *AR5K_AR5212_STA_ID1* in which the upper 16 bits are used to store part of the device MAC address and the lower 16 bits are used for device specifications. Two bits (B6 and B7) of the device specifications are used for setting the ACK rate: if both are set to 1, the device will send ACK at the low ACK rate, while if both are set to 0, ACK will be sent at the default high ACK rate.

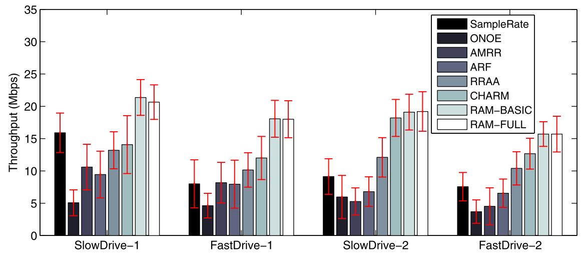
We achieve point 2 via tuning a special register called *AR5K_RATE_DUR*. MadWifi has two major modules, ATH (which is open source) and HAL (which is in binary form). ATH is the central component of MadWifi and it calls the *net80211* stack layer to exploit 802.11 functionalities and calls HAL to communicate with the hardware. We find that even if we change the value of the duration field of a frame in ATH, it will be overwritten in HAL after the frame is passed from ATH to HAL. In fact, HAL maintains a table of duration values for each transmission rate, which is stored in the *AR5K_RATE_DUR* register. In our implementation, we manually set the duration value for each rate in this register according to the proposed RAM scheme.



(a) indoor static scenarios.



(b) indoor mobile scenarios.



(c) outdoor vehicular scenarios.

Fig. 10. Comparison of throughput performances (each point is plotted with 90 percent confidence interval).

5.3 Experimental Results

We compare the throughput performances of testing schemes in indoor static, indoor mobile, and outdoor vehicular scenarios, and results are plotted in Fig. 10. In this section, we first give a few general observations on the results, and then discuss the results in each of the three scenarios in detail.

The first general observation is that RAM schemes clearly outperform other testing schemes in all experimental scenarios, indoor or outdoor, static or mobile, and the performance gain becomes more significant as the relative speed of two stations goes up. This is because RAM is receiver-based. By using the feedback from the receiver, the transmitter can select the proper transmission rate to match the current channel condition. In comparison, CHARM is transmitter-based and suffers from channel asymmetry. Moreover, RAM is frame-based, which can adapt much faster to rapid variations of the channel condition. In comparison, SampleRate, AMRR, ONOE, and RRAA are transmitter-based schemes and based on packet statistics. As a result, they usually are slow in adapting to the channel variation.

Another general observation is that ARF and ONOE perform the worst in most experimental scenarios. The reason is as follows: ARF waits for 10 consecutive successful transmission attempts before increasing the rate. Unfortunately, from our experiments, we find that channel fluctuation is common in practice, even in indoor static environments. In the presence of channel fluctuation, it is

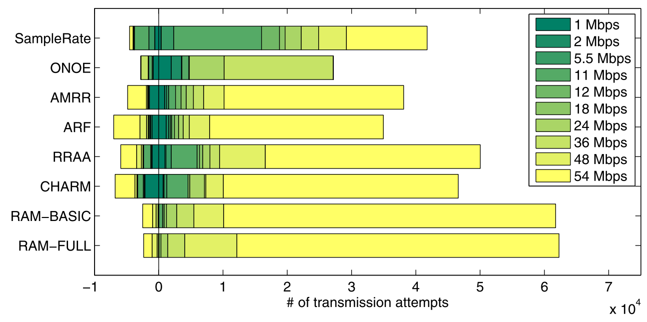


Fig. 11. Rate usage distribution for Walk-1. The number of successful or failed transmission attempts is shown as a positive or negative bar.

rare to have 10 consecutive successes. In comparison, AMRR—ARF's adaptive variant—adapts the success count threshold by using a binary exponential backoff starting with 10. As a result, AMRR performs better than ARF in scenarios when the channel condition is stable, e.g., indoor static scenarios. However, in mobile scenarios, the performance of AMRR is as poor as ARF.

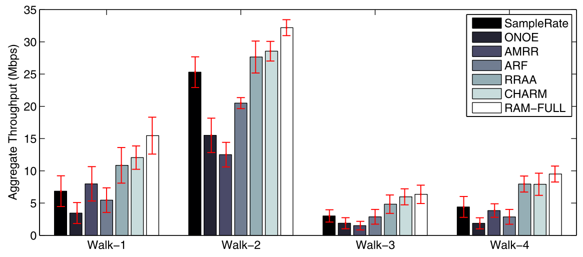
ONOE is a conservative rate adaptation scheme by design: it increases the transmission rate at most once during any 1-second period. Once it decides that a rate does not work well for the current channel condition, it will not attempt this rate again until at least 10 seconds later. From the experiments, we observe that ONOE takes quite long time to converge to the proper transmission rate. This also indicates that the selection of a proper initial transmission rate is critical for ONOE.

5.3.1 Indoor Static Scenarios

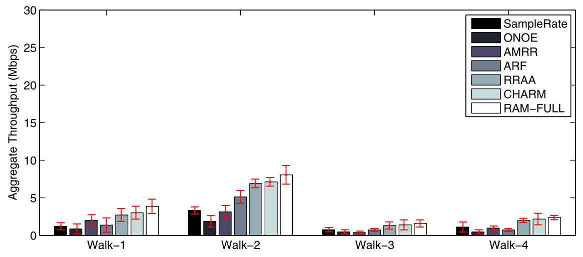
Experimental results for indoor static scenarios are plotted in Fig. 10a. It can be seen that both RAM-FULL and RAM-BASIC show comparable or better performances than other testing schemes in all four scenarios. RAM-FULL and RAM-BASIC yield similar performances. As will be shown in Sections 5.3.2 and 5.3.3, their performances are similar as long as there are no hidden nodes in the network. This is because without hidden nodes, the RTS usage is disabled for most of the time and hence RAM-FULL is almost equivalent to RAM-BASIC. For Static-1 scenario, we observe that the performance of CHARM is worse than all others. This is because CHARM is designed based on the assumption of symmetric channel conditions, which does not hold in Static-1 where one station is at P1 (inside an office) and the other station is at P2 (outside the office). From the SNR traces collected in Static-1, we notice that the channel conditions are highly asymmetric and the difference between DATA SNR and ACK SNR is as large as 11 dB.

5.3.2 Indoor Mobile Scenarios

Experimental results for indoor mobile scenarios are plotted in Fig. 10b. Similar to indoor static environments, RAM-FULL and RAM-BASIC yield higher throughput than others for all indoor mobile scenarios. Again, because RRAA adopts a time window of 150 ms (hence may not respond to channel variation quickly) and CHARM is unable to handle channel asymmetry (hence may cause unnecessary frame losses or underutilization of the good channel condition), both RRAA and CHARM perform worse than RAM.



(a) indoor mobile scenarios with duplex communication.



(b) indoor mobile scenarios with interfering communication.

Fig. 12. Comparison of throughput performances in more complicated indoor mobile scenarios (each point is plotted with 90 percent confidence interval).

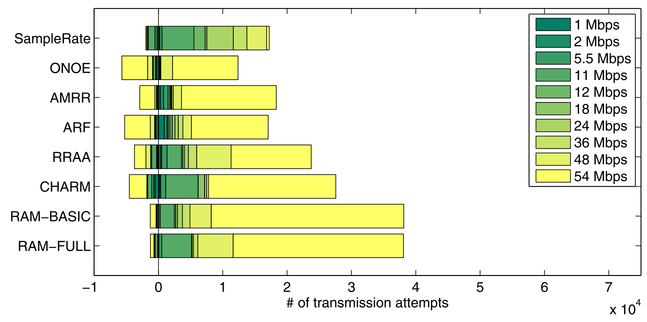


Fig. 13. Rate usage distribution for FastDrive-1. The number of successful or failed transmission attempts is shown as a positive or negative bar.

TABLE 7
Description of Experimental Scenarios for Interoperability between RAM-Based and Legacy 802.11 Devices

Scenarios	Communication Pair 1		Communication Pair 2	
	Tx (P6→Q2→P6)	Rx (at P4)	Tx (various locations)	Rx (at P4)
C1	Legacy	Legacy	Legacy	Legacy
C2	Legacy	RAM-FULL	Legacy	Legacy
C3	RAM-FULL	Legacy	Legacy	Legacy
C4	RAM-FULL	RAM-FULL	Legacy	Legacy

In order to have a good understanding on how and why RAM schemes outperform others, we investigate the Walk-1 scenario in more depth and study the cause of the observed throughput differences by plotting the rate usage distribution of each scheme in Fig. 11. The numbers of successful or failed transmission attempts at different transmission rates are shown as positive or negative bars of different colors. As shown in the figure, RAM schemes make effective usage of the available transmission rates: 1) a majority of the successfully transmitted frames are attempted at the highest rates of 48 or 54 Mbps; 2) a very few frames are transmitted at the lowest rates of 1 or 2 Mbps; and 3) the frame loss ratio is low (3.63 percent for RAM-FULL and 3.87 percent for RAM-BASIC). In comparison, both RRAA and CHARM suffer a much higher frame loss ratio at 10.56 and 12.70 percent, respectively. Moreover, we observe that SampleRate and ONOE transmit a large portion of the frames at low rates due to their conservative natures.

To evaluate the performance of RAM further, we have done a few additional experiments under more complicated indoor mobile scenarios: 1) *duplex communication*, where two stations transmit data to each other simultaneously; and 2) *interfering communication*, where two additional communication pairs are introduced as the interfering sources to operate on the same channel as the target communication pair that we evaluate. In scenario 2, one of the interfering pairs is colocated at the P4 position shown in Fig. 9a and communicate at high rates, while the other pair is placed at the P2 and P3 positions and communicates at low rates. Results are plotted in Fig. 12, where the superior performance of RAM-FULL over other schemes can be observed.

5.3.3 Outdoor Vehicular Scenarios

Fig. 10c shows the experimental results for outdoor vehicular scenarios and Fig. 13 plots the rate usage distribution of each testing scheme in the FastDrive-1 scenario. In general, the results are similar to those in indoor mobile environments. In addition to the similar observations discussed in the previous section, we have a few more observations as follows:

First, from the SNR traces collected (not included in the paper due to space limitation), we observe that the channel condition in outdoor vehicular environments (with faster station movement) fluctuates more frequently and at a larger scale than indoor mobile environments (with slower station movement).

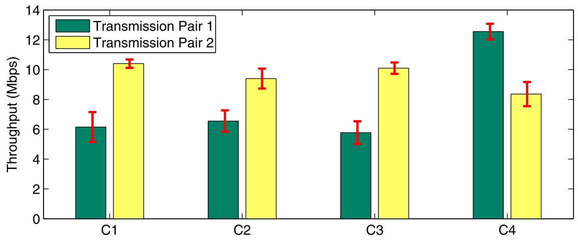
Second, in spite of the high fluctuation of channel conditions in outdoor vehicular environments, RAM continues to outperform all other schemes and the performance gain becomes more significant. As shown in Fig. 13, when using RAM-FULL in FastDrive-1, 81.27 percent of the frames are transmitted successfully at 48 or 54 Mbps while the frame loss ratio is only 3.15 percent.

Third, as shown in Fig. 13, both RRAA and CHARM suffer an even high frame loss ratio (13.55 and 14.08 percent, respectively, for FastDrive-1, in comparison to 10.56 and 12.70 percent for Walk-1) due to more dynamic channel conditions in outdoor vehicular environments.

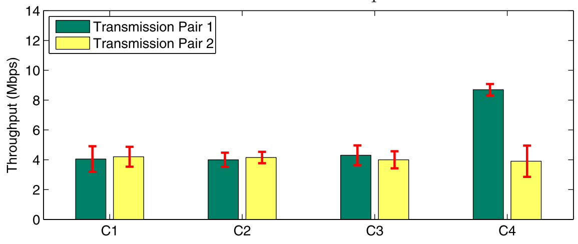
Lastly, comparing Fig. 13 with Fig. 11, we can see that ONOE behaves quite differently in FastDrive-1 and Walk-1 scenarios. In FastDrive-1, ONOE transmits frames at much higher rates for most of the time but the frame loss ratio is also higher. This is because we use different initial transmission rates for ONOE in FastDrive-1 and Walk-1 scenarios.

5.4 Interoperability between RAM-Based and Legacy 802.11 Devices

As discussed in Section 4.6, it is important that RAM-based devices can interoperate with legacy 802.11 devices. In this section, we verify the effectiveness of our proposed solution to interoperability using experimental results. We design four experimental scenarios which are listed in Table 7.



(a) Communication pair 1 is mobile while communication pair 2 is static. The transmitter of communication pair 2 is static at P3.



(b) Both communication pairs are mobile. The transmitter of communication pair 2 moves in the same trajectory of P6→Q2→P6 as the transmitter of communication pair 1.

Fig. 14. Experimental results for interoperability between RAM-based and legacy 802.11 devices (each point is plotted with 90 percent confidence interval).

There are two communication pairs in the experiments. Communication pair 2 uses legacy 802.11 devices running the SampleRate rate adaptation scheme, while communication pair 1 uses different combinations of RAM-FULL-based and legacy 802.11 devices in different scenarios.

In the first experiment, we keep the transmitter of communication pair 2 static at the P3 location shown in Fig. 9a. Results are shown in Fig. 14a. When communication pair 1 proceeds independently, it achieves a throughput of 15.2 Mbps when both stations use RAM-FULL. On the other hand, if communication pair 2 proceeds independently, it achieves a throughput of 25.6 Mbps. As we can see from Fig. 14a, the performance of each link decreases when both links are active and contend for the channel.

As discussed in Section 4.6, by using our proposed solution, receiver of communication pair 1 in scenario C2 and transmitter of communication pair 1 in C3 will eventually operate in Legacy mode. It means that C1, C2, and C3 shall have a similar throughout performance, which can clearly be observed in Fig. 14a. On the other hand, stations of communication pair 1 in C4 will operate in RAM mode eventually, which yields a much higher throughput than other scenarios. Moreover, we observe that the performance of communication pair 2 in C4 actually decreases a bit comparing to other scenarios. This is because RAM calculates the NAV value in a conservative manner, which may give more chance for RAM-based devices to contend for the channel when ACK is sent at the high rate. This issue could be addressed by introducing another timer and assigning it a value equal to the NAV value; RAM-based device will not attempt to access the channel until this timer expires, thus contending fairly with other devices in the network.

In the second experiment, the transmitter of communication pair 2 moves in the same trajectory as the transmitter of communication pair 1. Results are shown in Fig. 14b and similar observations can be made as in the first experiment.

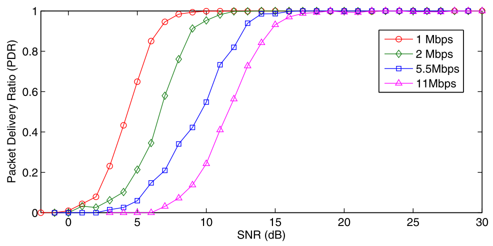


Fig. 15. The PDR versus SNR curves for 802.11b rates used in the simulation.

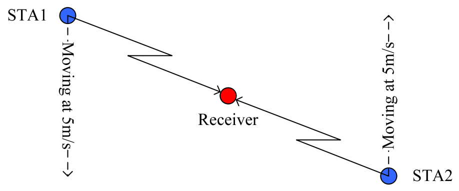


Fig. 16. The simulated hidden nodes scenario.

6 Simulation Study

In this section, we use the ns-2 simulator [24] to evaluate the effectiveness of RAM schemes in the presence of hidden nodes, as well as the effects of ACK rate variation on the system performance. The reasons for using simulation instead of experiment to study these issues are as follows: 1) in practice, it is difficult to set up a hidden nodes scenario; and 2) to study the effects of ACK rate variation, we need to compare RAM with an ideal rate adaptation scheme that can only be achieved in the simulation. In the following, we first introduce the simulation setup, followed by simulation results.

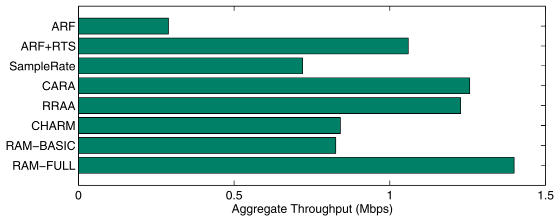
6.1 Simulation Setup

Instead of using the simple 0/1 packet delivery model given in the ns-2 simulator, we use the empirical Packet Delivery Rate (PDR) versus SNR curves in the simulations. These curves (shown in Fig. 15) are obtained from 10 experimental traces with each lasting for about 10 minutes. Moreover, for the clarity of presentation and explanation, we assume the 802.11b PHY in the simulations. Simulations for other 802.11 PHYs yield similar results.

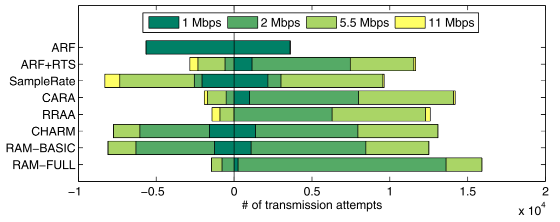
In the simulations, each transmitter transmits in the saturation mode, i.e., its data queue is never empty, and all data frames are transmitted without fragmentation. We use LLC/IP/UDP as the upper layer protocol suite, and the MAC-layer data payload length is 1,500 octets.

6.2 Simulation Results for Hidden Nodes Scenario

In the first part of the simulation, we investigate the performances of testing schemes in the presence of hidden nodes. The simulated hidden nodes scenario is shown in Fig. 16, where two transmitters are located at opposite sides of the receiver. Both transmitters start moving at 5 m/s at the same time and they are hidden to each other along most of the trajectory. We simulate a Ricean fading channel with a K-factor of 6 dB and assume a maximum speed of 10 m/s for movement in the environment.



(a) Throughput comparison.



(b) Comparison of rate usage distributions.

Fig. 17. Simulation results for the hidden nodes scenario.

We compare the throughput performances of RAM schemes against the following rate adaptation schemes: SampleRate, ARF, ARF + RTS (ARF with RTS enabled all the time), RRAA, CHARM, and CARA. Note that CARA is a rate adaptation scheme that is not implementable with commercial 802.11 devices and hence is not evaluated in our experimental study.

Throughputs of testing schemes are compared in Fig. 17a. As expected, schemes with RTS capabilities such as RAM-FULL, ARF + RTS, CARA, and RRAA can deal with hidden nodes well and yield higher throughput. Among them, RAM-FULL has the best performance. In comparison, schemes without RTS capabilities suffer significant performance degradation, including RAM-BASIC, CHARM, ARF, and SampleRate. ARF performs particularly bad because it cannot differentiate collision-induced losses from channel-error-induced losses and hence transmits at a very low rate. Note that CHARM yields comparable throughput as RAM-BASIC. This is because we use the Ricean fading model to simulate a perfect symmetric channel, and hence CHARM does not suffer from asymmetric channel conditions.

We plot the rate usage distributions of testing schemes in Fig. 17b. High frame loss rate caused by hidden nodes can be seen in the figure for RAM-BASIC, CHARM, ARF, and SampleRate. Among them, ARF has the highest frame loss ratio of 61 percent. In comparison, when RTS is used to deal with hidden nodes, frame loss ratio is reduced drastically for RAM-FULL, ARF + RTS, CARA, and RRAA. RAM-FULL has the largest number of successful transmission attempts and the smallest frame loss ratio, which explains its best throughput performance.

6.3 Effects of ACK Rate Variation on the System Performance

In the second part of the simulation, we study the effects of ACK rate variation used in RAM schemes on the system performance. In RAM schemes, since we set the duration field using the low ACK rate, when the receiver replies ACK at the high rate, the channel will be idle for a short period of time. To verify that this mechanism will not cause noticeable performance degradation, we simulate multiple Tx-Rx pairs transmitting simultaneously and stations are randomly placed within a circle with a radius of 40 m. Transmitters always transmit in the saturation mode.

TABLE 8
Throughput Comparison (in Mbps) with Multiple Simultaneous Tx-Rx Pairs

# Tx-Rx Pairs	1	2	4	8	16	32
Ideal-RAM	4.5828	4.0640	3.9815	3.8752	3.1668	2.1563
RAM-FULL	4.5822	3.9989	3.9555	3.7992	3.0887	2.0291

We compare RAM-FULL with a scheme called Ideal-RAM in terms of the system throughput. Ideal-RAM operates in the same way as RAM-FULL except that Ideal-RAM does not use the ACK rate variation to convey the feedback information; rather, the rate selections by the receiver are made available to the transmitter by modifying the ns-2 simulator. So, Ideal-RAM is only possible with the simulator but not implementable in practice. We vary the number of Tx-Rx pairs and simulation results (averaged over 20 simulation runs) are given in Table 8. We can see that even when the network is heavily loaded with 32 Tx-Rx pairs, the ACK rate variation used in RAM-FULL only results in a small 5.9 percent degradation of the system throughput.

7 Conclusions

From the experiments, we find that conditions of wireless channels in mobile environments exhibit severe asymmetry and high fluctuation. To deal with these issues, we propose a practical rate adaptation scheme, called RAM, for 802.11 devices in mobile environments. RAM is a receiver-based scheme and the main novelty of RAM lies in the usage of ACK rate variation to convey the feedback information from the receiver to the transmitter. Moreover, a RAM-based 802.11 device is able to identify whether its communicating partner is a RAM-based or legacy 802.11 device, based on which it operates in the corresponding mode. This way, the interoperability between RAM-based and legacy 802.11 devices can be guaranteed.

We have implemented RAM in the MadWifi device driver and extensive experimental results show that RAM is able to deal with channel symmetry and adapts quickly to the channel variation. It outperforms existing rate adaptation schemes in both static and mobile environments, particularly in outdoor vehicular scenarios.

Acknowledgments

The research reported in this paper was supported in part by the Information Infrastructure Institute (iCube) of Iowa State University and the US National Science Foundation under Grants CNS 0716744 and CNS 0831874.

References

[1] Multiband Atheros Driver for Wi-Fi, http://www.madwifi.org, 2011.

[2] A. Kamerman and L. Monteban, "WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band," *Bell Labs Technical J.*, vol. 2, no. 3, pp. 118-133, Aug. 1997.
[3] M. Lacage, M.H. Manshaei, and T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach," *Proc. ACM Int'l Symp. Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '04),* 2004.
[4] J. Kim, S. Kim, S. Choi, and D. Qiao, "CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs," *Proc. IEEE INFOCOM,* 2006.
[5] S. Kim, S. Choi, D. Qiao, and J. Kim, "Enhanced Rate Adaptation Schemes with Collision Awareness," *Proc. Int'l IFIP-TC6 Conf. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet (Networking '07),* 2007.
[6] S.H.Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust Rate Adaptation for 802.11 Wireless Networks," *Proc. ACM MobiCom,* 2006.
[7] J. Bicket, "Bit-Rate Selection in Wireless Networks," master's thesis, MIT, 2005.
[8] G. Judd, X. Wang, and P. Steenkiste, "Efficient Channel-Aware Rate Adaptation in Dynamic Environments," *Proc. ACM MobiSys,* 2008.
[9] J. del Prado Pavon and S. Choi, "Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement," *Proc. IEEE Int'l Conf. Comm. (ICC '03),* 2003.
[10] J. Zhang, K. Tan, J. Zhao, H. Wu, and Y. Zhang, "A Practical SNR-Guided Rate Adaptation," *Proc. IEEE INFOCOM,* 2008.
[11] Onoe Rate Control, http://www.madwifi.org/browser/trunk/ath_rate/onoe, 2011.
[12] G. Holland, N. Vaidya, and P. Bahl, "A Rate-Adaptive Mac Protocol for Multi-Hop Wireless Networks," *Proc. ACM MobiCom,* pp. 236-251, 2001.
[13] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic Media Access for Multirate Ad Hoc Networks," *Proc. ACM MobiCom,* 2002.
[14] H. Jung, K. Cho, Y. Seok, T. Kwo, and Y. Choi, "RARA: Rate Adaptation Using Rate-Adaptive Acknowledgment for IEEE 802.11 WLANs," *Proc. IEEE Consumer Comm. and Networking Conf. (CCNC '08),* pp. 62-66, 2008.
[15] C. Chen, H. Luo, E. Seo, N.H. Vaidya, and X. Wang, "Rate-Adaptive Framing for Interfered Wireless Networks," *Proc. IEEE INFOCOM,* 2007.
[16] Z. Li, A.K. Gupta, and S. Nandi, "A Full Auto Rate (FAR) Mac Protocol for Wireless Ad Hoc Networks," *IEE Proc.-Comm.,* vol. 152, no. 3, pp. 311-319, June 2005.
[17] P. Shankar, T. Nadeem, J. Rosca, and L. Iftode, "CARS: Context Aware Rate Selection for Vehicular Networks," *Proc. IEEE Int'l Conf. Network Protocols (ICNP '08),* 2008.
[18] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal, "Vehicular Opportunistic Communication under the Microscope," *Proc. ACM MobiSys,* 2007.
[19] Wistron CB9-GP-EXT 802.11a/b/g Cardbus Card, http://www.wneweb.com/products.htm, 2011.
[20] RSSI in MadWifi, http://madwifi-project.org/wiki/UserDocs/RSSI, 2011.
[21] *IEEE 802.11 Standard, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* IEEE, 2007.
[22] Atheros Registers, http://madwifi.org/wiki/DevDocs/AtherosRegisters, 2011.
[23] Iperf, http://dast.nlanr.net/projects/Iperf, 2011.
[24] Network Simulator - ns-2, http://www.isi.edu/nsnam/ns, 2011.

Xi Chen received the BS degree in electrical engineering and information science from the University of Science and Technology of China, Hefei, China, in 2006. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, Iowa State University, Ames. His research interests include protocol design and performance evaluation for 802.11-based wireless/mobile networks. He is a student member of the IEEE.

Prateek Gangwal received the BE degree in information technology from the National Institute of Technology (formerly REC), Durgapur, India, in 2006, and the MS degree in computer engineering from Iowa State University, Ames, in 2009. He is currently working for Thomson Reuters, Minnesota. His research interests include wireless networks, web application frameworks, and semantic web. He is a student member of the IEEE.

Daji Qiao received the PhD degree in electrical engineering systems from the University of Michigan, Ann Arbor, in February 2004. He is currently an associate professor in the Department of Electrical and Computer Engineering, Iowa State University, Ames. His current research interests include algorithm and protocol innovation and implementation for IEEE 802.11 wireless local area networks, system modeling and performance analysis of wireless sensor networks, and pervasive computing applications. He is a member of the IEEE and ACM.