An Experimental Evaluation of Several Rate Adaptation Protocols

Soon-Hyeok Choi, Robert Grant, Wonsoo Kim, Hyrum K. Wright, Robert W. Heath Jr., Scott M. Nettles
Wireless Networking and Communications Group (WNCG)
Department of Electrical and Computer Engineering
The University of Texas at Austin
1 University Station C0803, Austin, TX 78712-0240
Email: {schoi, gdavid, wkim, hwright, rheath, nettles}@ece.utexas.edu

***Abstract*—There are a seemingly unbounded number of protocols and algorithms that aim to improve the function or performance of wireless computer networks, including many that are based on "cross-layer" techniques. Despite these numbers and despite the close coupling of many of these techniques to both the physical layer and the channel over which packets are transmitted, the vast majority of results are based only on analytical modelling and/or simulation. Even very basic techniques that require modification of stock MAC and physical layers have not been studied on actual hardware, particularly hardware that uses modern PHY technologies, such as orthogonal frequency division multiplexing (OFDM). Controlling the transmission rate of packets is such a technique and we present an experimental evaluation of two well known techniques: Auto Rate Fallback (ARF) and Receiver Based Auto Rate (RBAR). Our evaluation is based on the Hydra wireless networking testbed, which has a fully programmable MAC and PHY, based on the 802.11 DCF MAC and the 802.11a/g OFDM PHY. Our experiments are based on a variety of channels, both real and, for greater control and flexibility, emulated. These experiments allow us to evaluate the design and performance of the techniques in real world situations and give us insight into their interactions with the PHY and channel that would otherwise be hard to gain.**

Many physical layers (PHYs) used in wireless networking allow packets to be transmitted at a variety of rates by controlling modulation and coding. For example, the PHY of 802.11a can support rates of 6, 9, 12, 18, 24, 36, 48 and 54 Mbps. The rate that can be supported reliably depends on the quality of the wireless channel over which the packet is transmitted and choosing the highest reliable rate can significantly improve system performance.

A number of rate control algorithms have been proposed for 802.11 systems. Because the rate depends on the packet destination and the PHY is not aware of the destination of the packets it transmits, these algorithms are implemented as part of the link/media access control (MAC) layer. Some algorithms do not require information from the PHY, while others are "cross-layer" and depend on information from the PHY, usually channel state. Also, some algorithms require different packet formats than the standard 802.11 formats. We study one algorithm, Auto Rate Fallback (ARF) [1], that needs no information from the PHY and no modifications to the standard packet format. We study another algorithm, Receiver Based Auto Rate (RBAR) [2], that both requires information from the PHY and packet modifications.

In this paper, we study ARF and RBAR experimentally using the Hydra wireless network testbed [3]. The chief goal of Hydra is to provide flexible, but realistic, implementations of cross-layer wireless protocols in which there is a close coupling between the PHY, MAC, and other higher levels. Such systems are difficult to study with off the shelf hardware because in general it is not possible to modify the basic operation of the node. Even systems such as MadWifi [4], which allow researchers to make significant changes to the MAC and to interact closely with the PHY do not allow wide ranging changes to the PHY or tight coupling of changes to the PHY and the MAC. Hydra has a flexible software-based MAC and PHY and changes to both are under the control of the algorithm and protocol developers.

The rest of the paper is organized as follows. Section I presents ARF and RBAR and other related work. Section II overviews the Hydra testbed. Section III discusses the experimental setup and in particular the channels used. The heart of the paper is Section IV which presents a series of experiments that compare ARF and RBAR along a number of dimensions. Finally, Section V presents future work and concludes.

I. Rate Adaptation Algorithms

ARF is an example of a rate adaptation algorithm that is history based (others include [5], [4], [6]). ARF adjusts the rate by using an indirect indicator of the channel quality - the success or failure of a transmission. If a transmitter sends a packet and receives a corresponding ACK, that is clear evidence that the channel is good enough for the current rate and perhaps for one that is higher. Therefore, ARF increases the rate if it sees some number (10 for [1]) of consecutive successful transmissions. On the other hand, ARF treats a transmission failure as an indication that the channel cannot support the current rate. Therefore, it reduces the rate when it sees several (2 for [1]) transmission failures in a row.

ARF has the advantages that it requires no information from the PHY and no packet format changes. However, there are several problems with ARF that we will investigate. First, because it is history based, it can only adapt to changes in the channel that take place on time scales greater than a few packet

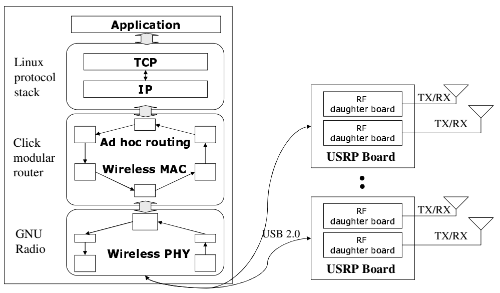


Fig. 1. Block diagram of a Hydra node.

exchange times. Second, even if the channel is constant, ARF occasionally raises the rate to "probe" to see if a higher rate is feasible, resulting in failed transmissions when the higher rate fails. A final issue we will not investigate is that ARF assumes that all failures are because the channel cannot support the current rate, even though the failure may have some other cause, such as a collision.

RBAR [2] is feedback based. RBAR was designed to address the problems with ARF by directly measuring the channel just before every data transmission using the request-to-send (RTS) and clear-to-send (CTS) handshake. When a receiver receives an RTS, the PHY estimates the channel quality. The MAC uses the channel quality estimate to choose a rate and sends the rate back on the CTS. Finally, the transmitter sends the data packet at the specified rate.

RBAR addresses the problems we identified for ARF. First, it adjusts the rate directly based on the channel quality. Holland [2] used the signal-to-noise ratio (SNR) to estimate the channel quality, achieving significant throughput gains over ARF. Second, by estimating the channel quality just before every data transmission, RBAR can adapt to channels that vary more slowly than a RTS/CTS handshake but faster than tens of packet times. This also avoids unsuccessful probes using higher rates. We will investigate several problems with RBAR. First, it requires the use of an RTS/CTS exchange, which if not otherwise needed increases overhead. Second, if the channel changes faster than an RTS/CTS/Data exchange RBAR may well estimate the rate incorrectly. One issue that we leave to future work is that for some PHY technologies and channels, SNR may not be the best metric for choosing the rate.

The goal of this paper is to experimentally explore basic issues that arise in real implementations of rate adaptation using RBAR and ARF as exemplars. Thus we touch only briefly on other related work. Opportunistic Auto Rate (OAR) [7] uses rate control algorithms such as ARF or RBAR to send multiple packets in a fixed amount of time, rather than simply shortening the transmission time of a single packet. Adaptive ARF (AARF) [5] is an extension of ARF that continuously changes the number of successes and failures that trigger rate changes. The authors show that AARF can be practically implemented using existing wireless devices such as MadWifi [4] and simulations show an improvement over ARF. Another approach was proposed in [8], [9], where rates are chosen based on channel status measured at the transmitter. The key assumption is channel reciprocity. The Hybrid rate control protocol [9] uses the signal strength indicator of the ACK to look up the next rate to use. This protocol also adjusts its lookup table to calibrate the performance of each rate. In [9], they performed experiments using a off-the-shelf wireless network interface and showed good results, although they ignored cases in which such channel state information becomes stale.

II. The Hydra Testbed

Our experiments are performed using the Hydra testbed. Hydra has been presented in some detail in [3] and an overview of a variety of PHY issues that arise when using Hydra (and when experimenting with PHYs, MACs, and cross-layer design) will appear in [10]. Here we present a brief overview of Hydra.

Figure 1 shows a block diagram of a Hydra node. The programmable RF front-end is the Universal Software Radio Peripheral (USRP) [11], which interfaces to the general purpose host through a USB 2.0 connection. The diagram shows several USRPs with multiple antenna, but here we use a single USRP with a single antenna. All other aspects of Hydra, the PHY, MAC, and higher layers, run on a general purpose processor (GPP) running Linux. The PHY is written in C++ using the GNU Radio framework [12]. It implements an orthogonal frequency division multiplexing (OFDM) transceiver that is based on 802.11a/g [13], [14]. The MAC is also written in C++ but using the Click programmable router framework [15]. It implements the 802.11 distributed coordination function (DCF) MAC [16]. Click also provides ad-hoc routing and interfaces to the standard Linux stack. Implementing the PHY and MAC in C++ using general frameworks greatly eases the task of creating working cross-layer prototypes.

The Hydra (and IEEE 802.11a) physical layer uses 52 subcarriers that are modulated using BPSK, QPSK, 16 QAM, and 64 QAM and uses rate $\frac{1}{2}$, $\frac{2}{3}$, and $\frac{3}{4}$ punctured convolutional codes. Hydra's data rates are limited due to the bandwidth constraints of the USB 2.0 interface between the PC and the RF front-end. Thus the prototype supports physical layer data rates of 0.6, 0.9, 1.2, 1.8, 2.4, 3.6, 4.8 and 5.4Mbps, exactly 10 times less than 802.11a.

Hydra's software implementation means that the receiver PHY takes a significant amount of time to process a packet and that time is dependent on packet length. Thus, Hydra uses spacing between packets that are longer relative to the packet transmission times than for the 802.11a standard. Our goal is to provide insights about systems with more typical, hardware-based PHYs, and so we have processed the throughput results presented in Section IV so that they reflect the interpacket spacing defined in the IEEE 802.11a standard [13].

III. Experimental Setup

Since the Hydra system is implemented modularly in software and can be run on laptops, we can easily reconfigure the system for different experiments. Many of the experiments described below are run over real wireless channels using the USRP RF frontend. However, it can be difficult to create a variety of channels, and achieving reproducibility in real channels is challenging. One advantage of Hydra's software implementation is that it is easy to create an emulator that processes the baseband output of the Hydra transmitter and then sends it to a Hydra receiver. We discuss the setup for both our real and emulated channel experiments in more detail in subsequent sections.

A. Real channels

Over a real wireless channel we use two GPP machines running the Hydra transceiver code. Each transceiver is connected to a USRP with daughter cards operating in the 400 MHz band. Though we have the ability to operate at 2.4 GHz by simply replacing the daughter cards, we run our experiments at 400 MHz because of the ubiquitous interference present in the 2.4 GHz ISM band.

Most of our experiments are based on a line-of-sight (LOS) channel between two nodes set up in a small lab space. The distance between the nodes is such that given the power constraints of the USRPs, we can explore a range of SNRs that allow us to adapt over the full set of rates. We vary the average SNR in these experiments by changing the transmission power.

Creating a non-LOS channel is more difficult, partly because at 400 MHz radio transmissions have significant penetrating power. Initially, we used a steel computer case, but it did not completely block the LOS component. We solved this problem by scavenging a 1/2 inch thick steel plate that is several feet on a side. This allowed us to create a non-LOS channel, but we still found that the range of channel variation and especially its time-dependence was limited. Having people walk around the environment added time variation, but is not a realistic general solution. Our current best solution involves an oscillating metal fan that we place in front of the steel plate so that how it reflects RF into the environment varies as it oscillates. We present some results from this channel in Subsection IV-F, but we continue to search for a better solution.

B. Emulated channels

The difficulty in creating and controlling real channels means that channel emulation is also important. Our experimental setup for an emulated channel consists of three GPP machines, two acting as Hydra nodes and one running the emulator code.

The channel emulator allows us to experiment with a variety of channel models and impairments. Thus far, we have implemented a gaussian model and a rayleigh fading model along with frequency and time diversity. In many cases, we first test our PHY code with a gaussian channel model to provide a theoretical performance bound for our system. This turns out to be a very powerful debugging tool. We also compare our emulated results to those obtained over real channels. Initial experiments of this kind showed a puzzling performance gap. However, other testing [10] revealed that the USRP has some frequency nonlinearities. Once we added these impairments to our emulator, we found a close match in our results. The results presented below include this impairment.

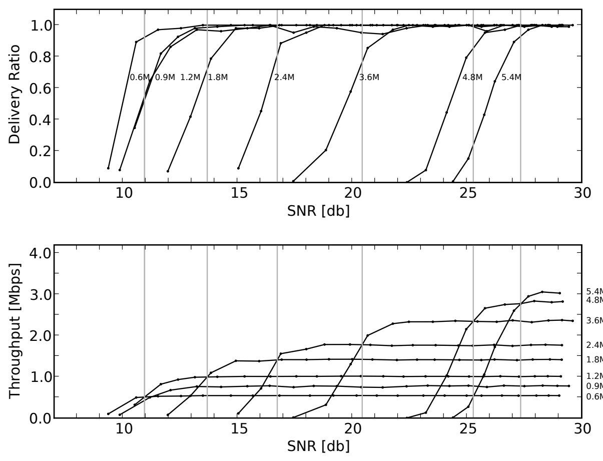


Fig. 2. Delivery Ratio and Throughput vs. SNR for fixed data rates. Dotted vertical lines indicate rate transition points for RBAR.

IV. Experiments

Our experimental results follow. Unless otherwise stated, all of these results are based on the LOS channel described in Section III and use 1370 byte packets, which produce 1500 byte packets (a maximum size ethernet frame) at the MAC layer. We use the abbreviation ARF-W for results based on ARF with a RTS/CTS exchange and ARF-WO for results based on ARF without such an exchange. The standard description of the delivery ratio and throughput graphs is found in Subsection IV-A.

A. Calibrating the RBAR transition points

Any RBAR implementation will need to make a choice about when to transition from one rate to another. As with Holland [2], we use SNR as the criteria for channel quality. We choose the transition points by experimental calibration using the fixed rates supported by Hydra as a function of SNR.

Figure 2 shows the experimental results we used to choose our transition points. Both graphs have SNR on the X-axis and show the results for all of the fixed rates supported by Hydra. The Y-axis of the top graph shows the packet delivery ratio and of the bottom graph shows the throughput achieved in Mbps. The vertical lines show the transition points we chose.

One design detail is what criteria to use for choosing the transitions. We choose to transition from one rate to another when doing so would result in higher throughput. Thus the transition points are located at the points where the throughput of a higher rate crosses that of the best performing lower rate. As shown by the vertical lines, the transitions we use are (in dB): 10.9, 13.7, 16.7, 20.4, 25.3, and 27.3. For our data, the 1.2 Mbps curve actually crosses the 0.6 Mbps curve at a lower SNR than the 0.9 Mbps curve does. This means that we never use the 0.9 Mbps rate. The 0.9 Mbps rate uses BPSK and a $\frac{3}{4}$ coding rate, while the 1.2 Mbps rate uses QPSK and an $\frac{1}{2}$ coding rate. As discussed in [17], the implication is the improvement from using $\frac{1}{2}$ coding out weighs the

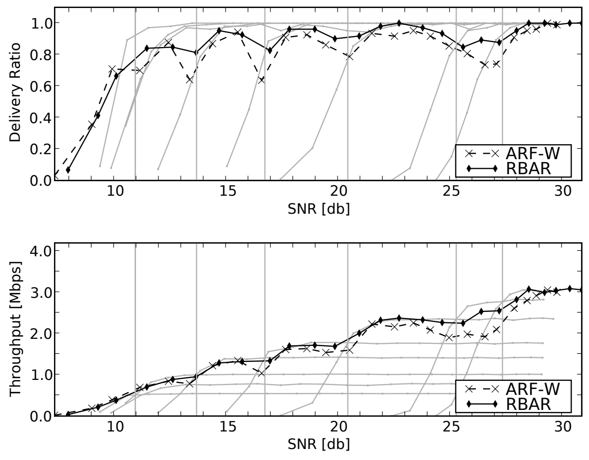


Fig. 3. Delivery Ratio and Throughput vs. SNR for RBAR and ARF-W. Fixed rate lines are shown lightly for reference.

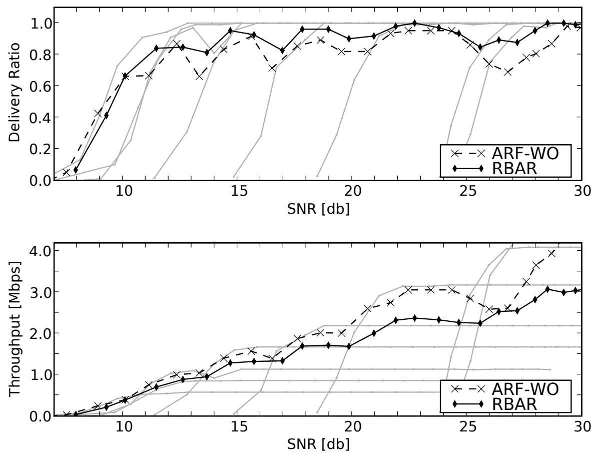


Fig. 4. Delivery Ratio and Throughput vs. SNR for RBAR and ARF-WO. Fixed rate lines are shown lightly for reference.

improvement from using BPSK. Looking at the delivery ratio graph we see the two rates are almost identical, implying that it is best to use the higher rate.

Our choice of throughput as the criteria represents a tradeoff in terms of reliability. Since throughput is the product of rate and the delivery ratio, choosing to transition to a higher rate when the throughput is equal implies that the delivery ratio will go down at the transition. We can see this on the delivery ratio graph because the vertical lines intersect the curves at points where the delivery ratio is still increasing. Obviously, it would never make sense to choose lower SNR transitions, since that would result in both worse reliability and worse throughput. However, one might wish to impose a reliability requirement as well and choose somewhat higher SNR transitions if it did not overly impact throughput.

B. RBAR and ARF with RTS/CTS

The goal of our first experiment is to compare the performance of RBAR and ARF when both incur the overhead of the RTS/CTS exchange. Figure 3 shows the experimental results comparing RBAR and ARF-W in the standard format. The fixed rate data is shown lightly for reference.

Considering the throughput first, we see that RBAR tracks the top of the curves formed by the fixed rate data relatively closely, indicating that RBAR is generally choosing the best rate possible. ARF-W also tracks the fixed rate data reasonably well, except in the region from about 24 dB to 28 dB. In general, RBAR achieves better throughput than ARF.

Considering the delivery ratios, we can account for most of the deviations from the ideal curve seen for the throughput. Looking at RBAR, we see that close to the transition points, the delivery ratio goes down as we expected given the criteria for choosing the transitions. Why RBAR seems less reliable around 26 dB is less clear, perhaps it is because two transitions are closely spaced, but this dip does account for the throughput dip also seen at this point. For ARF-W, we see greater variability. We expected that since the channel is not sampled directly the delivery ratio would be worse, but the sharp dips are somewhat unexpected. We note that they are quite correlated with the transition points we chose for RBAR. This suggests that at these SNRs, ARF is trying to “choose” between a lower rate and a higher rate and that this is resulting in higher packet loss when the channel fluctuates. We are attempting to verify this hypothesis.

C. RBAR and ARF-WO

The goal of this experiment is to study the impact of the RTS/CTS overhead on ARF. Figure 4 shows the experimental results comparing RBAR (using the same data as Figure 3) and ARF-WO in the standard format. The fixed rate data without RTS/CTS is shown for reference.

Considering throughput, ARF-WO is almost always better than RBAR indicating that, for this channel, RTS/CTS overhead dominates gains achieved by better channel adaptivity. (So as to keep the scale constant across the throughput graphs, we have truncated the curve at high SNR. ARF-WO actually achieves 4.2 Mbps at the maximum.) However, in general, ARF-WO often does not achieve the highest possible throughput. The delivery ratio results shows that, as expected, ARF-WO is almost identical to the previous results for ARF-W. Further we see that these deviations correlate well with the dips in throughput performance, as before.

D. Higher RBAR thresholds

The goal of this experiment is to study the impact of increasing the rate transition thresholds on RBAR. As discussed in Section IV-A, choosing which transition thresholds to use is a practical consideration of implementing RBAR on a real system and may impact both performance and reliability. We choose the thresholds for RBAR-cons(ervative) to be 1 dB higher than those chosen originally to see if we could improve reliability without impacting performance significantly.

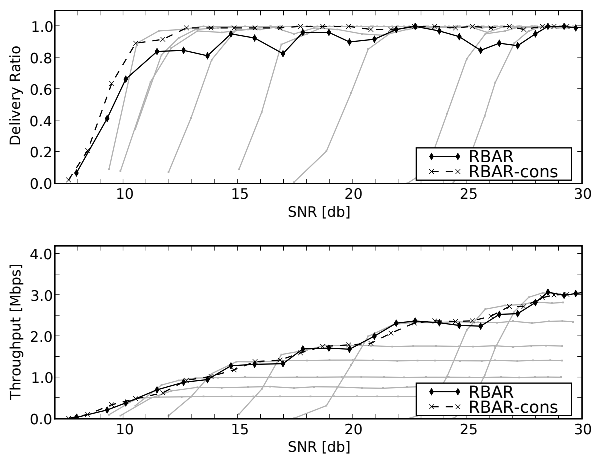


Fig. 5. Delivery Ratio and Throughput vs. SNR for RBAR and RBAR-cons. Fixed rate lines are shown lightly for reference.

Figure 5 shows the experimental results comparing RBAR (using the same data and rate transitions as Figure 3) and RBAR-cons. The throughput of RBAR-cons tracks that of RBAR, sometimes falling a little below, but sometimes falling a little above. The delivery ratio results are more dramatic. RBAR-cons shows significant improvements and for most of the SNR range is as reliable as the fixed rate examples. For higher level protocols that are sensitive to packet drops, such as TCP, this seems like a significant improvement.

E. Throughput as a function of packet size

In our previous experiments, we used 1370 B packets. Using large packets will minimize the impact of RTS/CTS overheads, helping RBAR. The goal of this experiment is to study how smaller packets might affect the tradeoffs between RBAR and ARF. These measurements have been made at a transmit power that guarantees both RBAR and ARF use the highest rate. This also helps ARF, since at the highest power ARF does not probe for a higher rate, thus eliminating that source of packet loss and throughput degradation.

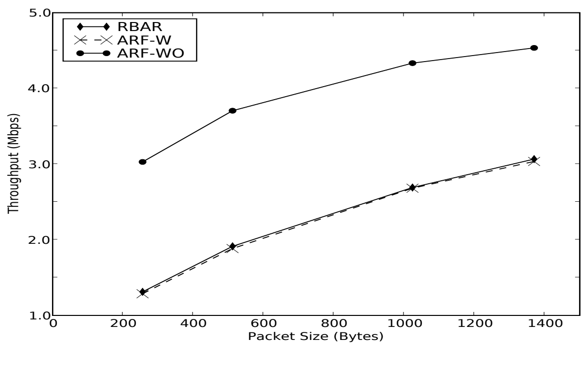


Fig. 6. Throughput for RBAR, ARF-W, and ARF-WO versus packet size.

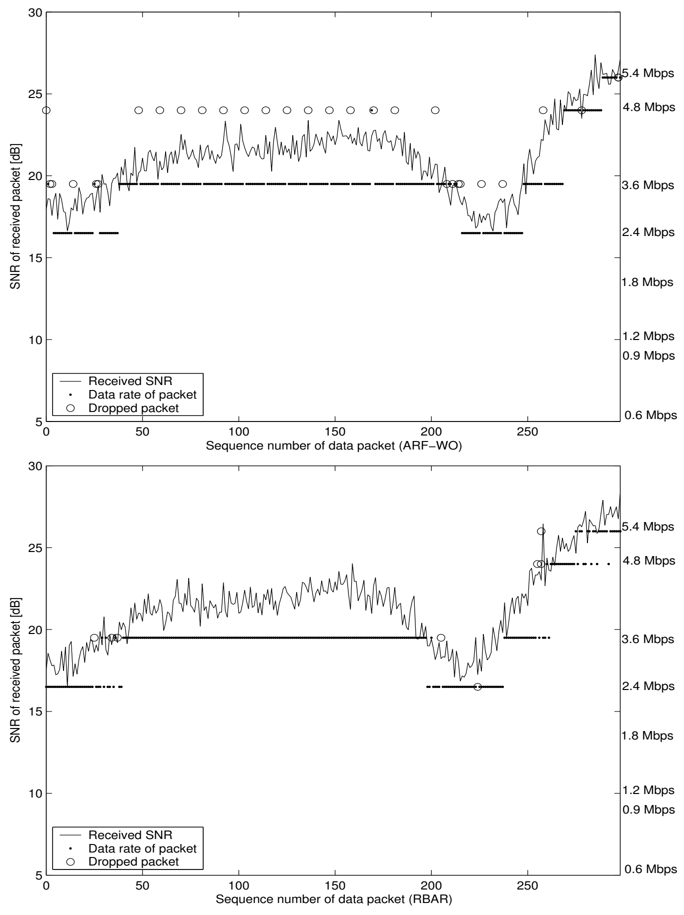


Fig. 7. MAC trace in a slow channel (Doppler=0.001)

Figure 6 shows the experimental results for RBAR, ARF-W, and ARF-WO. The X-axis is the size of the data packet in Bytes and the Y-axis is throughput in Mbps. For RBAR and ARF-W, the results are as expected. Since both will choose the highest rate and both have the same RTS/CTS overhead, we would expect the curves to match and they do so almost exactly. As expected they a significantly below ARF-WO, which also behaves as expected, with performance increasing as overhead goes down.

F. Time varying channels

The critical difference between ARF and RBAR is how they behave as the channel varies in time. Understanding this behavior is aided by using reproducible channels with controllable time variations. Thus this section focuses on measurements using our channel emulator. We begin by showing detailed traces of the behavior of both ARF and RBAR in a slow, faster, and fast channels at a fixed transmission power. We then observe both the throughput and delivery ratio as a function of SNR. We conclude with some measurements of a real time varying channel.

We implemented a time-varying channel by using Jake's Model [18], which is parameterized by the normalized doppler shift, a measure of the speed at which a channel changes. In the

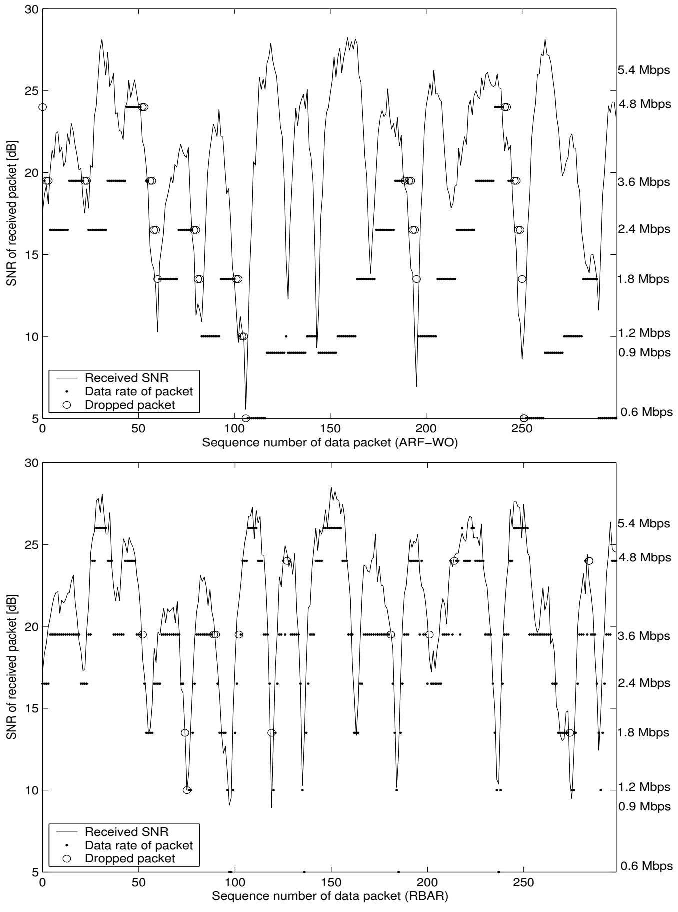


Fig. 8. MAC trace in a faster channel (Doppler=0.01)

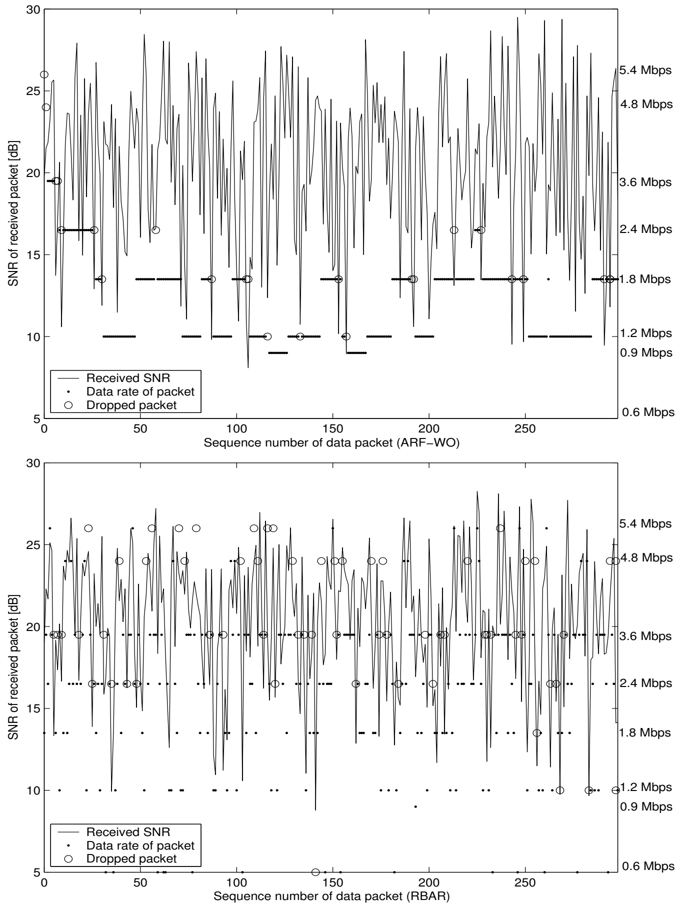


Fig. 9. MAC trace in a fast channel (Doppler=0.1)

channel model, the normalized doppler is defined as $f_D \cdot T_f$, where f_D and T_f denote doppler frequency and packet interval respectively. The doppler shift parameter ranges from 0.0 to 1.0, which we can set according to channel variation we want to model. We used 0.001 for our slow channel, 0.01 for our faster channel, and 0.1 for our fast channel.

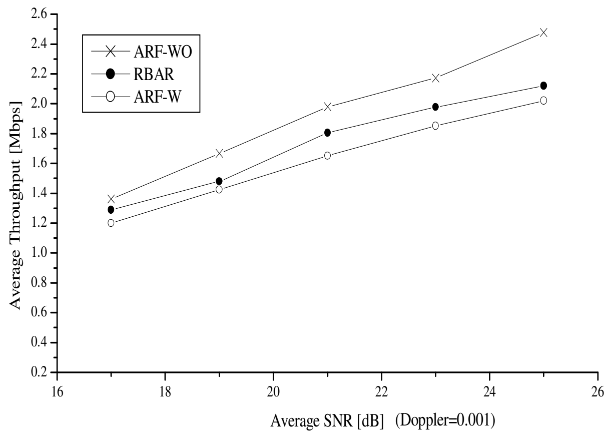
1) Detailed traces of MAC behavior: We begin by showing detailed traces of the behavior of the MAC for both RBAR and ARF in slow, faster, and fast channels. This allows us to confirm our understanding of how these algorithms work and to develop some intuitions about what detailed behavior to expect. For these MAC tracing plots, we set the TX power to 10.3 dBm which allowed us to see some rate variation in the results, and we sent 1024 Byte CBR traffic at a 0.3 msec interval. The traces each represent 300 packets.

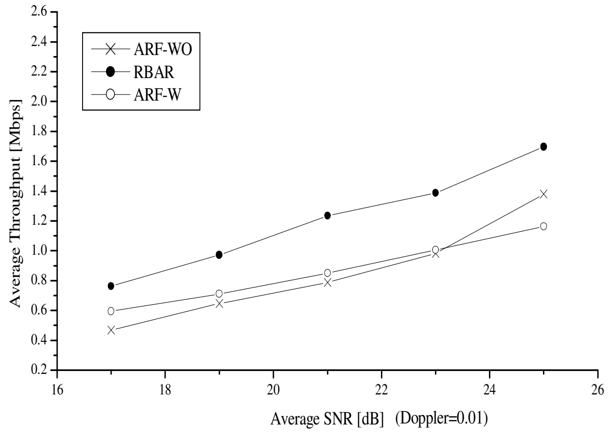
The top graph in Figure 7 shows a trace of the MAC behavior of ARF in the slowest channel (doppler 0.001). The X-axis is the sequence number of the packet. The left Y-axis shows the SNR of the received packet, while the right Y-axis shows the rate chosen for each packet. The line shows the received SNR for each packet and small dots show packets that were successfully received and larger open circle show packets that had errors.

This is the channel in which would expect ARF to track the channel the best. We see that it does track the channel well, but we also see that when it probes the channel every tenth packet is likely to fail, which is as expected.

The bottom graph of Figure 7 shows a trace of the MAC behavior of RBAR in the slowest channel (doppler 0.001) in the same format. We see that it tracks the channel as well as ARF, but without ARF's periodic failures. It does have an occasional failure when the channel becomes worse between sending the RTS and sending the data. From these results, we would expect that RBAR would outperform ARF in terms of packet delivery ratio and for throughput when ARF is used with RTS/CTS. However, ARF may outperform RBAR for throughput when it does not need RTS/CTS.

Figure 8 shows a trace of ARF on the top and of RBAR on the bottom both in a faster channel (doppler 0.01) in the same format. As expected, ARF can no longer track the channel well and it takes some time to "ramp up" during periods when the channel is good. However, we see that drops usually occur when the channel gets worse and such fades are well tracked. Looking closely it is possible to see the double drops that trigger rate reduction. Also as expected, RBAR tracks the channel well, with failures coming chiefly when the channel is falling, making it more likely that a data packet experiences a worse channel that its RTS. For this case, it seems likely that





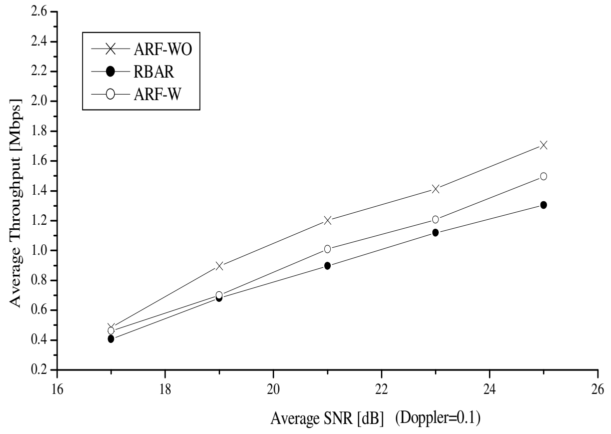


Fig. 10. Throughput in Time-Varying Channels (Slow to Fast)

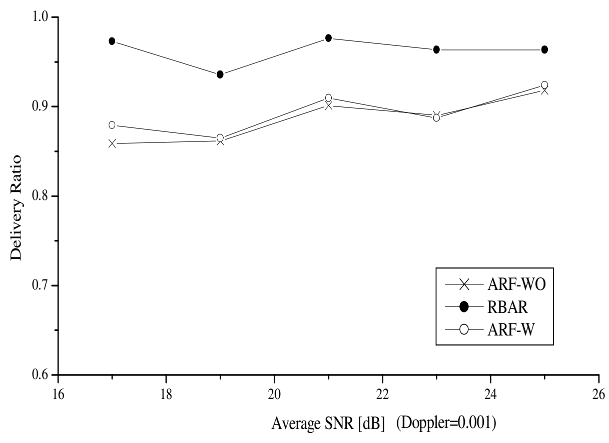
RBAR will outperform all versions of ARF for all metrics.

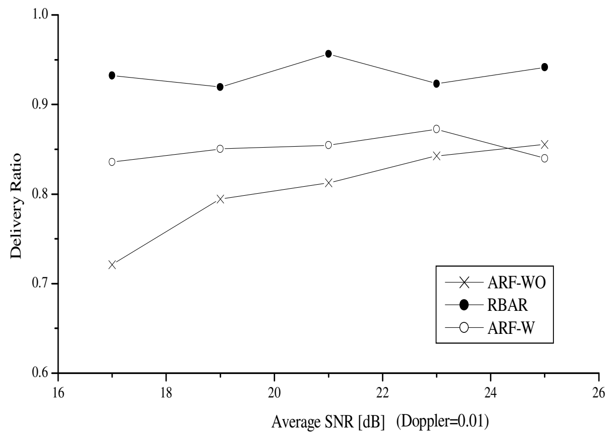
Finally, Figure 9 shows a trace of ARF on the top RBAR on the bottom both in a fast channel (doppler 0.1) in the same format. ARF tends to track the bottom of the envelope formed by the fast varying channel. In some case, probes succeed and for some period the rate increases, but in other cases they fail and the rate remains low. RBAR shows more complex behavior as it attempts to track the fast changing channel. When the channel goes up between the RTS and the data, the packet is successful, although perhaps at a lower rate than could have been achieve. On the other hand, when the channel goes down the data packet is sent at too high a rate and fails. Notice that the drops are more concentrated in the higher rates and the successful transmissions in the lower ones. We would expect ARF to perform significantly better in terms of delivery ratio, but it is hard to predict the results for throughput.

2) Throughput and delivery ratio as a function of average SNR: Now that we have some detailed intuition about how ARF and RBAR perform in our slow, faster, and fast channels, we examine their behavior as a function of SNR so as to confirm our intuitions and answer some of the questions left unresolved the detailed MAC traces above.

Figure 10 shows the results for throughput. For each graph, the X-axis is average SNR (in dB), the Y-axis is the throughput (in Mbps), and results are shown for RBAR and both variants of ARF. The top graph is the slow channel (Doppler 0.001). As expected, RBAR has somewhat better throughput than ARF-W due to failed probes, but worse performance than ARF-WO due to the added overhead of the RTS/CTS exchange. The middle graph is the faster channel (Doppler 0.01). As expected, RBAR has significantly better throughput than ARF-W, due to its significantly better ability to track the channel. Less predictably, for this channel, RBAR also performs better that ARF-WO, indicating that at least for some channels the ability to track the channel can outweigh the overhead of the RTS/CTS exchange. In fact, ARF-W has comparable performance to ARF-WO for some SNRs. Some insight can be gained by looking at the delivery ratio graph, where we see that ARF-W delivers more packets. What appears to be happening is that when sending RTSs, ARF-W resends any RTSs that are dropped, but without lower its rate. When an RTS gets through the channel has improved and ARF-W sends the packet. ARF-WO does not have the benefit of the RTS probe. Finally, the bottom graph is the fast channel (Doppler 0.1). From our MAC traces above, we knew that RBAR's greater failure rate might dominate its ability to sometimes transmit packets at a higher rate than ARF. This graph shows that this is indeed the case, with both versions of ARF outperforming RBAR.

Figure 11 shows the results for delivery ratio. For each graph, the X-axis is average SNR (in dB), the Y-axis is the delivery ratio, and results are shown for RBAR and both variants of ARF. The top graph is the slow channel (Doppler 0.001). As expected, RBAR is somewhat more reliable that both versions of ARF due to failed probes and because the channel is slow changing RBAR almost always chooses the correct rate. The middle graph is the faster channel (Doppler 0.01). As expected, RBAR is significantly more reliable than either version of ARF because it is able to track the channel more closely, although because of the possibility that the channel worsens before the data transmission, it is less reliable than in the slow channel. Finally, the bottom graph is the fast channel (Doppler 0.1). As expected from our MAC traces above, now RBAR does significantly worse than either version





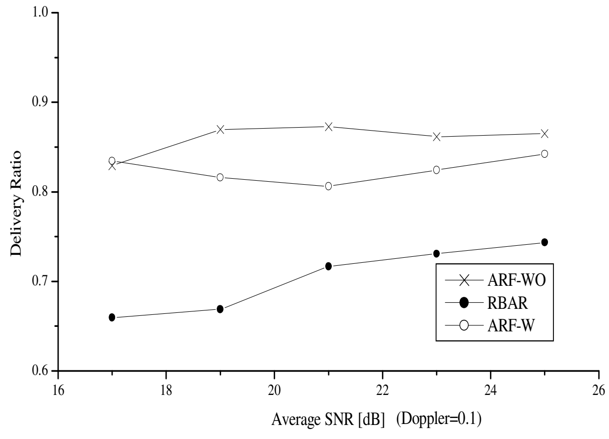
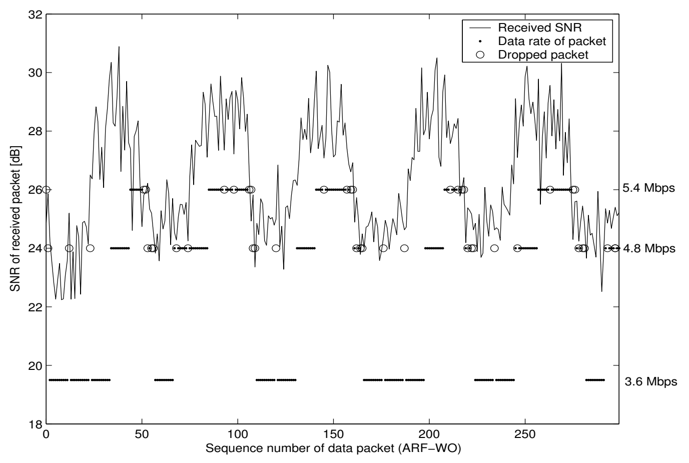


Fig. 11. Delivery Ratio in Time-Varying Channels (Slow to Fast)

of ARF because of the significant chance that the channel worsens before the data transmission.

3) A real time-varying channel: Finally, we present some results from a real time-varying channel. As described in Section III, this channel is created by using an oscillating metal fan in a non-LOS channel. This measurement is based on 256 Byte packets and a fixed power of 12.9 dBm



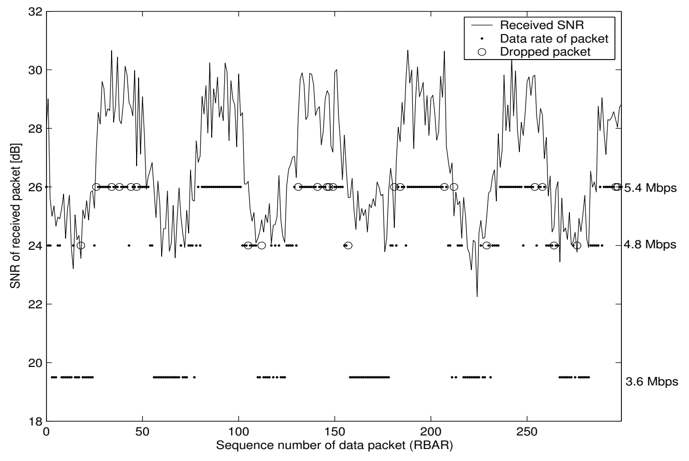


Fig. 12. MAC trace in real time-varying channel.

Figure 12 shows a trace of ARF on the top and of RBAR on the bottom in the same format as the other MAC traces. The received SNR shows the variation due to the fan very clearly and the overall SNR range is 5-6 dB. We see essentially the behavior we expect, ARF can track the channel when it has enough time to ramp up, while RBAR tracks the channel more closely. We do note that the sharp spikes (perhaps from the rotating blades) do seem to cause more drops for RBAR than some of our previous results have shown.

V. Future Work and Conclusions

Possible future work falls broadly into two categories, experimental work and algorithm development and validation. Experimentally, probably the most important avenues to explore next involve varying the kinds of channels we can experiment with. Ideally, we would be able to create reasonably reproducible time varying channels with deeper fades than those we explored briefly here. Also, here we have not explored any of the issues that arise when the channel become frequency selective, in which case, for example, average SNR will become a less reliable predictor of the best rate.

Algorithmically, there is also significant work to do. We have developed an algorithm, Transmitter Base Autorate (TBAR), that uses channel reciprocity to measure the channel, without the need for feedback. Space limitations have prevented us from presenting it here. In general, our results suggest that a variety of adaptive strategies might be worth pursuing, tracking the speed of the channel and trading off between RBAR and ARF like approaches. We believe that real world testing would greatly enhance the practicality of any such approach.

In conclusion, we have used the Hydra testbed, which provides a flexible software platform to implement a wide variety of cross-layer protocols, to experiment with several rate adaptation protocols. These experiments have been performed using realistic PHYs and MACs and over both actual and emulated channels. The results do not clearly favor one rate control technique over another, but they do give rise to an understanding of the systems of interest that would be hard to gain by using simulation alone.

References

[1] A. Kamerman and L. Monteban, "WaveLAN II: A high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal*, vol. 2, no. 3, pp. 118–133, 1997.

[2] G. Holland, N. H. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Mobile Computing and Networking*, 2001, pp. 236–251. [Online]. Available: citeseer.ist.psu.edu/holland01rateadaptive.html

[3] K. Mandke, S.-H. Choi, G. Kim, R. Grant, R. C. Daniels, W. Kim, S. M. Nettles, and R. W. H. Jr., "Early Results on Hydra: A Flexible MAC/PHY Multihop Testbed," in *Proceedings of the 65th IEEE Vehicular Technology Conference*, Apr. 2007, pp. 1896–1900.

[4] "MadWifi." [Online]. Available: http://sourceforge.net/projects/madwifi

[5] M. Lacage, M. H. Manshaei, and T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach," in *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04)*, Venice, Italy, Oct. 2004, pp. 126–134.

[6] J. C. Bicket, "Bit-rate Selection in Wireless Networks," Master's thesis, MIT, Feb. 2005.

[7] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic Media Access for Multirate Ad hoc Networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Atlanta, GA, Sep. 2002.

[8] J. del Prado Pavon and S. Choi, "Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement," in *Proceedings of IEEE International Conference on Communications (ICC'03)*, Anchorage, AK, May 2003, pp. 1108–1113.

[9] I. Haratcherev, K. Langendoen, R. Lagendijk, and H. Slips, "Hybrid Rate Control for ieee 802.11," in *ACM International Workshop on Mobility Management and Wireless Access Protocols (MobiWac'04)*, Philadelphia, PA, Oct. 2004.

[10] K. Mandke, R. C. Daniels, S.-H. Choi, S. M. Nettles, and J. Robert W. Heath, "Physical Concerns for Cross-Layer Prototyping and Wireless Network Experimentation," in *The Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, Montreal, QC, Canada, Sep. 2007.

[11] "GNU Radio: Universal Software Radio Peripheral." [Online]. Available: http://www.comsec.com/wiki?UniversalSoftwareRadioPeripheral

[12] "GNU Software Radio." [Online]. Available: http://www.gnu.org/software/gnuradio/

[13] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5 GHz Band*, Part 11 standard ed., IEEE 802.11 Working Group, Sep 1999.

[14] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, IEEE 802.11 Working Group, Piscataway, NJ, 2003.

[15] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000. [Online]. Available: citeseer.ist.psu.edu/article/kohler00click.html

[16] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE 802.11 Working Group, Piscataway, NJ, 1997.

[17] D. Qiao, S. Choi, and K. G. Shin, "Goodput Analysis and Link Adaptation for IEEE 802.11a Wireless LANs," *IEEE Transactions on Mobile Computing*, vol. 1, pp. 278–292, Oct. 2002.

[18] W. C. Jakes, *Microwave Mobile Communications*. Wiley, 1974.