BRAVE: Bit-Rate Adaptation in Vehicular Environments

Pralhad Deshpande, Samir R. Das
Computer Science Department
Stony Brook University
Stony Brook, NY 11794, USA

ABSTRACT

Rate selection in a wireless network is the problem of estimating the current channel conditions and determining the best physical layer bit rate for the outgoing frames in order to maximize the current throughput. All rate adaptation algorithms in literature arrive at an estimate of the current channel conditions by considering the recent history often in the order of seconds. In vehicular WiFi access networks, the constantly changing wireless channel conditions make the channel history quickly irrelevant. We develop BRAVE – an SNR-based rate adaptation algorithm, which only considers short history (500 ms) to make rate selection decisions. We show that a coarse-grained training approach is sufficient to estimate the SNR thresholds for rate selection as opposed to previous approaches that train on a per environment or a per AP basis. We study three frame-based rate adaptation algorithms and a popular SNR-based rate adaptation algorithm along with BRAVE and highlight their shortcomings in the rapidly changing vehicular WiFi access environment. In order to compare the algorithms under repeatable channel conditions, we also develop and use a novel emulation methodology where a software radio-based programmable noise generator is used to emulate varying link quality under vehicular mobility. We show that BRAVE performs significantly better than several prominent frame-based and the SNR-based rate adaptation algorithms.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communications—*Vehicular Communications*

General Terms

Performance, Design, Measurement, Experimentation.

Keywords

Vehicular Internet Access, WiFi, Bit-Rate Adaptation.

VANET'12, June 25, 2012, Low Wood Bay, Lake District, UK.

1. INTRODUCTION

The need for continuous innovation in the face of stiff competition has pushed major car manufacturers towards adopting the "connected car" concept. The term "connected car" refers to the integration of smartphone apps and content into the car. Typically this is done via the car's dashboard, enabling users to listen to online music, access Web data and stream video to the car's passengers. The current systems simply use a cellular connection as the backhaul. Such a system promises persistent albeit low bandwidth connectivity which might not be sufficient for bandwidth intensive applications. In addition, cellular data connections are expensive, particularly given that most operators are moving away from flat-rate pricing and introducing usage-based pricing. Thus, offloading mobile data to urban WiFi networks has been considered by several research groups [11, 16, 12, 8]. This not only provides an opportunity for cost savings, but also could provide a higher average bandwidth in case of dense AP deployments if occasional disconnections can be tolerated [10].

Choosing an appropriate physical (PHY) layer rate is important when a WiFi network is accessed with vehicular mobility. The 802.11 protocol specification allows for multiple transmission rates at the PHY layer each having a different modulation and coding scheme. For example, the 802.11b offers 4 different bit rates ranging from 1 to 11 Mbps and 802.11g offers 8 different bit rates ranging from 6 to 54 Mbps. 802.11b/g interfaces are quite common and thus typically there is a choice of as many as 12 different bit rates. Generally speaking, higher bit rates result in higher throughputs if the link quality is good but suffer from higher bit error rate (BER) when the link quality deteriorates. Conversely, lower bit rates provide lower throughput but are more resilient to lower quality links. 802.11 interfaces typically deploy a Rate Adaptation Algorithm (RAA) to optimize the chosen rate so as to maximize the overall throughput.

While RAAs have been widely studied, most of them [21, 5, 25] are targeted for traditional indoor wireless networks and are incapable of quickly adapting to the rapidly changing link quality experienced by the mobile vehicular client [17]. They tend to either under-estimate or over-estimate the real-time link quality and hence end up choosing incorrect bit rates. The unsuitability of industry-standard RAAs towards vehicular WiFi access is also highlighted in our past measurements in [10]. We also noticed that throughputs dropped with increasing vehicular speed since the link quality changed more rapidly at higher speeds adversely affecting the performance of the RAA.

All RAAs try to predict the real-time link quality by examining the past history – often over several seconds or longer. Generally speaking, RAAs belong to one of two classes. *Frame-based* algorithms maintain an estimation window for the link layer metrics

such as frame errors, while the *SNR-based* algorithms maintain an estimation window for physical layer metrics such as received signal strength indicator (RSSI) that indicate SNR. In our knowledge the literature does not provide a clear guidance as to what type of RAA should be used in what scenario. Also, the performance of these RAAs in vehicular environments has not been studied. All evaluations are limited to indoor environments only. Indoor evaluations hardly indicate performance under vehicular mobility as the link quality fluctuates rapidly in the latter case and often on a per-packet basis. Thus, the RAA must adapt very quickly.

In this paper we develop a new SNR-based RAA called BRAVE (Bit Rate Adaptation in Vehicular Environment), particularly targeting vehicular mobility. BRAVE estimates upcoming SNRs to be experienced on the link based on a short history of recent SNRs and exploits its knowledge of BER performances for different SNRs. We employ a coarse-grained training approach to estimate the SNR thresholds for rate selection as opposed to previous approaches that train on a per environment [9] or a per AP basis [18]. We evaluate the performance of BRAVE relative to several prominent RAAs. Given the difficulty of doing a head-to-head comparison under vehicular mobility (vehicular experiments are hard to repeat), we also use a novel emulation methodology. Here, a software radio-based programmable noise generator is used to emulate varying signal quality under vehicular mobility. We show that BRAVE performs significantly better than four other popular RAAs.

The rest of the paper is organized as follows. In Section 2 we describe the testbed setup. Section 3 describes a set of measurements that help us understand the characteristics of the dynamically varying WiFi channel. This leads to the development of the BRAVE algorithm that is described in Section 4. Section 5 evaluates BRAVE along with four other RAAs. The shortcomings of the other RAAs are highlighted. Section 6 describes the emulation experiments and Section 7 the related work. We conclude in Section 8.

2. TESTBED

For our study, we use a metro-scale WiFi deployment in the Long Island area in New York (population roughly 7 million). This service is called 'Optimum WiFi' [6] and is provided by Cablevision, a local cable TV provider and ISP. The WiFi network extends to the more populous areas of Long Island where our study is conducted. It also extends to parts of New York City, Pennsylvania, Connecticut and New Jersey, where Cablevision has service. The entire network has roughly 18,000 APs. WiFi access is provided free to all subscribers of Cablevision's TV or Internet services. We note that there are several hundreds of such metro-scale WiFi deployments in USA alone [3].

We use two Dell Latitude laptops running Linux to be carried in the car. One laptop acts as the vehicular client and the other acts as a sniffer. The original miniPCI WiFi interfaces from the laptops are removed and replaced by carrier-grade interfaces (Ubiquity XR2 [7]) with transmit power set to 25 dBm. The interfaces use Atheros chipset supported by the latest Madwifi driver (version 0.9.4) which we used. The WiFi cards are connected to 8 dBi omni-directional antennas using two 2 meter long co-axial cables. The antennas are carried at the back of the car as shown in Figure 2.

The client and the sniffer machines are connected using ethernet. A program on the client machine continuously monitors the current channel and notifies the sniffer machine in case of a channel change. Thus the sniffer always remains in the same channel as the client machine.

On the client, we set the WiFi card's AP selection mode to automatic, i.e. the Madwifi driver decides on the best AP to connect to. DHCP is used to obtain the IP address. The Optimum WiFi network allows clients to retain the IP address between associations; thus DHCP delay is incurred only once at the beginning. Optimum WiFi uses a browser-based authentication for the initial network access. This is done manually. Again this step is needed only once. Authentication is retained across associations.

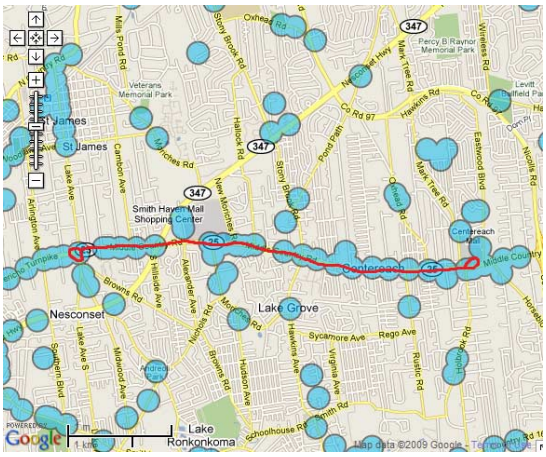


Figure 1: Map of the stretch of the road used for the driving experiments, along with the route shown in red. Approximate WiFi coverages are shown (from [6]).

The client generates upload traffic using the well known network performance evaluation utility `iperf` (version 2.0.5) . The iperf client is configured to send out 1400 byte UDP packets in a back-to-back manner. The sniffer logs all traffic on the current channel.

Figure 1 shows the road stretch used in the experiments with approximate locations of WiFi APs. This is a 9 mile round-trip drive and is used to conduct all our experiments.

3. UNDERSTANDING DYNAMIC CHANNELS

To design an RAA in vehicular scenario it is important to understand the channel dynamics that the algorithm must adapt to. It is also important to use a good measure of channel conditions. Following the general motivations behind the design of a major SNR-based RAA, CHARM [18], we use SNR as an indicator for channel conditions. It is well known that SNR determines the bit error rate (BER) for a wireless channel for a given PHY layer rate. Thus, direct measurement of SNR is expected to provide a simple mechanism to estimate the most effective rate to be used. This measurement can be done on a per-packet basis providing a method for quick adaptation. This is in contrast to frame-based methods that require several packet transmissions to build up a reliable statistics for the error rate.

All 802.11 interfaces report RSSI for each received frames. Depending on the interface and driver, the RSSI can be the measured SNR or the measured received signal strength. In the latter case, SNR can be calculated from a separately reported noise floor. In our reported experiments, RSSI is the same as SNR reported directly by the driver.

In the following we carry out a set of measurements to understand the error performance of 802.11b/g interface under vehicular mobility. This will form the basis of the BRAVE algorithm that we will describe subsequently.

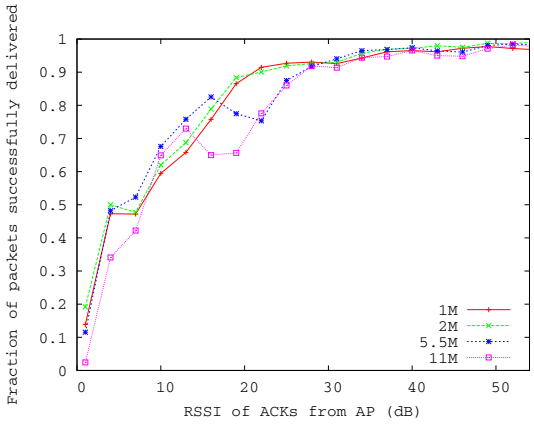
Figure 2: Two 8 dBi omni-directional antennas are mounted at the back of the car using a bicycle rack.

3.1 The CycleRate Algorithm

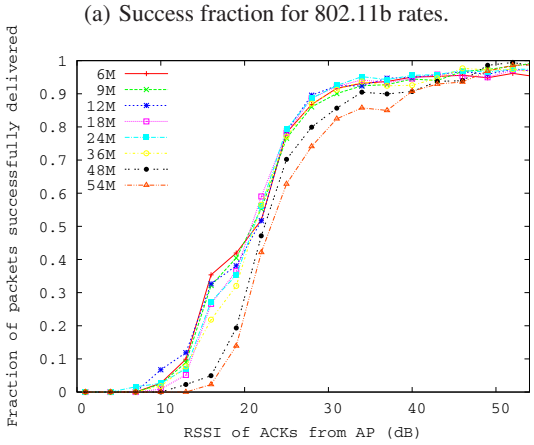
To understand the error performance of all 802.11b/g bit rates under different channel conditions we create a rate selection algorithm that simply loops through all the available bit rates on a per packet basis. We disable retries so that every packet is transmitted exactly once. Thus, there is no exponentially increasing backoff delays between transmissions. We call this the CycleRate algorithm. On average, the duration of one complete cycle is approximately 27 ms. However, this is much larger than the channel coherence time in out-door environments. In [9] the authors have determined the channel coherence time to be as small as 300 μs in realistic vehicular WiFi access settings. Thus, we do not expect the different PHY layer rates to be directly comparable for error performance within a single cycle of the CycleRate algorithm but on average we believe our measurement methodology provides a good first order approximation of their error performances at different SNR levels.

We conduct a driving experiment using the CycleRate algorithm using our experimental setup described in Section 2. UDP upload from the vehicular client to a server on the Internet is used as the traffic load. A saturated load is used – UDP packets are transmitted as fast as possible. The chosen packet size is 1400 bytes. We choose to use upload experiments instead of downloads as the WiFi APs are not under our experimental control. The assumption, however, is that the wireless channels are symmetric and the lessons learnt in these experiments are equally applicable for download scenarios.

In Figure 3 we plot the fraction of packets successfully delivered at different RSSI values. The RSSI is measured off the last received ACK packet. We see that 802.11b and 802.11g have distinctly different packet loss characteristics. The DSSS modulation scheme used for 802.11b rates tends to be much more resilient to packet losses than the OFDM modulation scheme used for 802.11g rates. Within each group, the lower rates show further resilience to packet loss because bit error rate (BER) increases with bits/symbol and lower rates are encoded with smaller bits/symbol. This is more clearly seen in case of 802.11g rates than in case of 802.11b rates. Note that the theoretical proportionality of BER to



(a) Success fraction for 802.11b rates.



(b) Success fraction for 802.11g rates.

Figure 3: Fraction of packets successfully delivered at varying RSSI levels for various 802.11b/g rates.

bits/symbol holds only when the channel remains coherent across the measurement duration. It is practically impossible to attain large channel coherence times in our measurement scenario. This reflects in our analysis in Figure 3 where in several cases the theoretical proportionality of BER to bits/symbol is not held. The 802.11b rates seem to be robust even at low RSSI levels but 802.11g rates start becoming useful when the RSSI is above 20 dB. It is clear that an RAA should use OFDM rates at higher RSSI levels and DSSS rates at lower RSSI levels.

3.2 Predicting RSSI

Predicting the channel quality is critical in determining the best bit rate. The study of temporal correlation between RSSI values under vehicular mobility is useful in this regard. To study this, we use the autocorrelation in the time series of RSSIs to estimate how well the recent RSSI values can predict the RSSI values to be seen in the near future.

The prediction model can use one of several standard techniques such as autoregressive (AR) model, moving average (MA) model, autoregressive moving average (ARMA) model, etc. In [10] we have used the AR model successfully to predict the future throughputs based on throughput measurements in the past in a similar vehicular environment. Thus, we limit our analysis to the AR model.

For this analysis we consider a time series of average RSSI values over 500 ms time slots. We evaluate the performance of the AR(k) model for several values of k to see the quality of estima-

AR(k) Model	MSE(Y_{est})/VAR(Y)
AR(1)	0.932
AR(2)	0.928
AR(4)	0.924
AR(8)	0.922
AR(16)	0.920

Table 1: Prediction accuracy using AR(k) model.

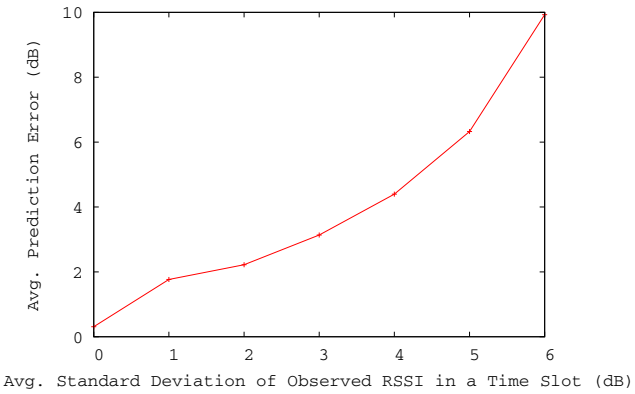


Figure 4: Prediction error increases with increasing standard deviation of the RSSI values within a time slot.

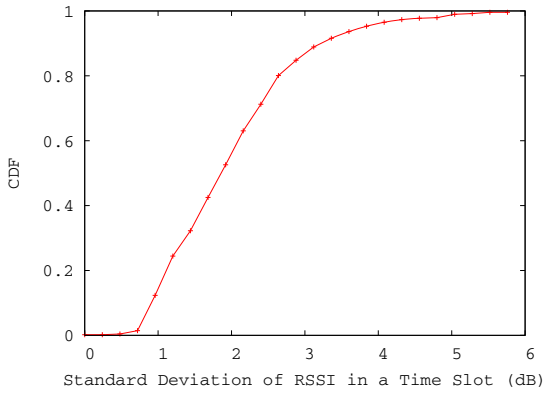


Figure 5: CDF of standard deviation computed over RSSI values in 500 ms time intervals.

RSSI Range	*AGGRO* MRR Chain	*SAFE* MRR Chain
Under 20dB	11, 5.5, 2, 1	11, 1, 1, 1
20 – 28 dB	48, 36, 11, 1	48, 11, 5.5, 1
Over 28 dB	54, 48, 36, 1	54, 11, 5.5, 1

Table 2: Rates in the Multi-Rate Retry array for BRAVE. Each rate is attempted only once. *SAFE* mode is used in case the standard deviation of RSSI values in the previous time slot is more than 3 dB.

tion. Table 1 shows the mean-squared estimation error divided by the sample variance (MSE/s^2). This is based on a standard AR(k) fitting method used in the mathematical software `GNU Octave, version 3.0.5` [2], when applied on the average RSSI per time slot time series as described above.[1] Note that there is very little difference with different values of k. This implies that the simplest AR(1) model can provide us with a good prediction of the average RSSI in the next 500 ms time slot.

It is interesting to study whether a higher variability in the RSSI values within a time slot leads to better prediction of the RSSI values in the next time slot. In Figure 4 we plot the standard deviation of the RSSI values within a time slot versus the error between the observed and the predicted RSSI of the next time slot. For this analysis we consider only those time slots where we have at least 10 RSSI values in order to obtain statistically meaningful results. The predicted RSSI in this case is determined using the AR(1) model described above. We see that as the standard deviation increases prediction error also increases. We consider an error of 3 dB or higher as significant. A lower difference is not likely to influence the bit error rate appreciably as per Figure 3. Note that the error is 3 dB when the standard deviation is approximately 3 dB.

Figure 5 shows the CDF of the standard deviation of RSSI values within each 500 ms time slot. Again, only those time slots are considered where we have at least 10 RSSI values. Notice that in approximately 90% of the cases the standard deviation is within 3 dB indicating that the prediction error will be within 3 dB. From this we conclude that in 90% of the cases the predictability of RSSI using the AR(1) model is acceptable for the purpose of a rate selection mechanism.

[1]If x_t is the average RSSI at the t-th time slot, the AR(k) model estimates x_t as $\hat{x_t} = c + \sum_{i=1}^{k} \phi_i x_{t-i} + \epsilon_t$, where c is a constant, ϕ_i's are the parameters of the model, and ϵ_t is white noise. Each time slot is considered to be 500 ms.

4. THE BRAVE PROTOCOL

The BRAVE (Bit-Rate Adaptation in Vehicular Environments) protocol makes use of the popular Madwifi device driver [4]. The Madwifi Hardware Abstraction Layer (HAL) provides the capability of multi-rate retry (MRR). The MRR array can be populated with up to four different rates and the number of retry attempts for each rate. These rates are to be attempted sequentially for the given number of times until ACK is received or the total number of retries is exhausted. In the latter case the frame is dropped.

BRAVE selects bit rates for outgoing frames based on the RSSI values of the previously received ACKs. It makes use of the MRR array by appropriately populating the rates that should be attempted. We now describe the operation of BRAVE.

The BRAVE protocol operates in two modes – *AGGRO* (aggressive) and *SAFE* (conservative). The *AGGRO* mode is used when the predictability of future RSSI values is good and the *SAFE* mode is used when the predictability is poor. Based on the analysis in the previous section, the standard deviation of prior RSSI values is used as an indicator of predictability.

From Figure 3 one can deduce that at RSSI levels below 20 dB the 802.11b (DSSS) rates provide the highest throughputs and above 20 dB the 802.11g (OFDM) rates provide the highest throughputs. Table 2 shows the rates used by BRAVE in both the *AGGRO* and *SAFE* modes at different RSSI levels. For the *AGGRO* mode, the three rates that provide the highest throughput at every RSSI level make up the first three elements of the MRR array. The fourth element is always chosen to be the base rate of 1 Mbps as this rate has the lowest error rate at any RSSI level. For the *SAFE* mode, BRAVE sets the first MRR array element as the rate that provides the best throughput from the entire rate set and the remaining elements are chosen from the 802.11b rate set. This is because 802.11b rates show higher resilience to frame loss. Overall, the BRAVE protocol works as follow.

1. BRAVE makes rate adaptation decisions every time slot (500 ms).
2. At startup, BRAVE uses the $SAFE$ mode and assumes the average RSSI over the previous time slot to be 0.
3. If at least 10 ACKs are received in the previous time slot of 500 ms and the standard deviation of the RSSI values in the last slot is less than 3, BRAVE switches to the $AGGRO$ mode. Otherwise the mode remains as $SAFE$.
4. Depending on the average RSSI in the previous time slot and the mode of operation, the MRR array is populated with values from Table 2. Each rate is tried only once.

In its current implementation BRAVE uses static RSSI threshold levels (as in Table 2) based on prior analysis with data collected using the CycleRate algorithm described in Section 3.1. Our expectation is that a training similar to the above needs to be done on a per Metro-area WiFi network basis to create a base line. This is feasible as the APs belonging to a network deployment tend to be identically configured both in hardware and software. Such a training component can be potentially incorporated within the BRAVE protocol itself.

5. EVALUATION

In this section we compare BRAVE with four other RAAs. They will be described momentarily. All experiments are done on the 9 mile stretch of road shown in Figure 1 over multiple drives, each drive using one specific RAA. The AP density in this stretch of the road is quite high. The experimental setup is the same as described in Section 2. Unfortunately, the vehicular experiments are not repeatable, as it is virtually impossible to ensure that the vehicle is driven exactly at the same speed at every location on the road on the repeated drives. Thus, the wireless link quality and AP associations vary differently across time for different drives. It is possible to 'ride out' the impact of this variability by using a very large sample size for the RAAs (i.e., many drives). However, this would be an enormous experimental effort given that several RAAs are to be evaluated. Thus, a different evaluation method is used.

Table 3 shows the performance of each of the algorithms under conditions of good and poor link quality. Link quality is considered to be good if the RSSI is 20 dB or higher and considered to be poor otherwise. We show the usage percentage for each rate and the percentage of times each rate is successful. This RSSI-based grouping allows us to evaluate how an algorithm performs when faced with different channel conditions.

In the subsections below we describe and evaluate each of the algorithms used in this analysis.

5.1 SampleRate

SampleRate [21] is a frame-based RAA. It is the default RAA in Madwifi. *SampleRate* adapts rates based on transmission statistics over a sliding window. The transmission time for a frame is defined as the time to send the frame successfully. This includes time for retransmissions and backoffs. The bit rate chosen is the one that achieved the smallest *average transmission time* in the last sampling period of 10 seconds. This way, *SampleRate* tries to achieve the best average throughput in the long term. *SampleRate* starts transmitting at the highest rate and decreases the rate immediately if it experiences four consecutive transmission failures. To explore other rates that could provide better transmission times, *SampleRate* randomly selects the rate from the set of rates whose average transmission time is less than the average lossless transmission time of the rate in use, as the rate for every tenth frame. Also those rates that have experienced 4 consecutive transmission failures are excluded from this sampling.

From the above description it can be easily deduced that *SampleRate* is not the ideal choice for highly dynamic channel conditions. The sampling period of 10 seconds is too large given the highly dynamic nature of the vehicular wireless channel. Also, a random loss of a probe packet which is non-reflective of the channel conditions proves to be costly to *SampleRate* since the statistics for the probed bit-rates are built slowly – alternate rates are probed at every tenth packet. This makes *SampleRate* prone to under-selecting rates.

From Table 3 we can see that even when the link quality is good, *SampleRate* relies heavily on the 11 Mbps and 5.5 Mbps bit rates. When it does choose higher OFDM bit rates (24 Mbps and 36 Mbps), the link quality tends to be excellent. This is highlighted by the fact that 83.2% and 87.9% of packets sent at these bit rates receive link layer ACKs. *SampleRate* is also quite slow in reacting to a drop in link quality. It sends out about 40% of its packets at rates above 11 Mbps even when very few packets are actually acknowledged. Thus, we conclude that *SampleRate* is slow in reacting to changing channel conditions, and tends to under-select bit rates when the link quality is good and over-select bit rates when the link quality is poor.

5.2 AMRR

The *Adaptive Multirate Retry (AMRR)* algorithm [20] is also a frame-based algorithm whose implementation is a part of Madwifi. It also adapts its rate based on transmission statistics but the period over which rate selection decisions are made is shorter. *AMRR* increases the current bit rate by one if in each of the last 10 successive 500 ms time-slots at least 10 packets were transmitted and less than 10% of packet transmissions failed. The bit rate is decreased by one if more than 33% of the packets fail in a 500 ms time slot. In *AMRR* the selected rate and two rates below it constitute the first 3 elements of the MRR chain. The fourth rate is set to 1 Mbps. One transmission attempt is made per rate in the MRR chain.

Referring to Table 3 *AMRR* tends to be very aggressive in rate selection when the link is good. The three highest rates are selected over 72% of times. However, when the link quality is poor, the same aggressiveness tends to pull it away from the more stable DSSS rates to higher OFDM rates that are not as successful. Close to half of the packets are sent at rates higher than 11 Mbps and almost none of these packets are successfully delivered. Hence overall, *AMRR* is guilty of over-selecting rates.

5.3 CHARM

Channel-aware Rate Adaptation Algorithm (CHARM) [18] is an SNR based rate adaptation algorithm. *CHARM* uses channel reciprocity to obtain channel information. The technique is based on predicting the path loss to a receiver in the near future by passively overhearing messages sent by the receiver. *CHARM* automatically adapts the signal thresholds that are used by the transmitter to select the transmission rate. This adaptation is however very slow – in the order of a few seconds.

CHARM is known to outperform popular frame based RAAs like *SampleRate* in indoor environments [18]. However in outdoor environments at vehicular speeds *CHARM* performs very poorly. Our experiments show that *CHARM* continues to predict the link quality as being excellent even when the link quality is very poor in reality. Referring to Table 3, in case the link quality is good, *CHARM* almost exclusively uses the highest available bit rate. However, *CHARM* is very slow in reacting to deteriorating link quality and

Bit Rate (Mbps)	*SampleRate*				*AMRR*				*CHARM*			
	% Used		% ACKed		% Used		% ACKed		% Used		% ACKed	
	Good link	Poor link	Good link	Poor link	Good link	Poor link	Good link	Poor link	Good link	Poor link	Good link	Poor link
1	1.61	24.0	59.8	5.80	3.61	43.9	59.3	16.2	0.03	0.83	5.97	0.44
2	1.89	6.73	74.6	20.9	0.75	1.52	66.7	21.7	0.00	0.16	0	0.78
5.5	18.0	12.3	86.0	18.4	1.12	1.04	79.6	50.1	0.02	0.51	0	0.36
6	7.25	0.23	90.6	1.40	1.67	3.51	55.9	0.42	0	0	0	0
9	0.02	2.77	3.37	0	1.87	1.33	35.6	0	0	0	0	0
11	26.7	11.6	79.1	14.7	4.89	1.52	84.0	23.7	0.03	1.05	0	0.08
12	7.36	2.19	77.1	0.15	5.09	7.58	63.8	0.04	0	0	0	0
18	3.85	5.60	77.3	0	3.19	10.1	60.8	0	0.05	4.72	0	0
24	9.53	7.05	83.2	0	5.01	10.5	63.3	0	0.07	4.73	0.74	0
36	14.8	11.3	87.9	0	7.31	18.8	60.7	0	0.09	3.99	0	0
48	5.22	7.35	82.2	0	10.8	0	71.7	0	0.00	6.24	0	0
54	3.54	8.72	73.8	0.03	54.5	0	92.2	0	99.6	77.7	49.7	0.52

Bit Rate (Mbps)	*RapidSample*				*BRAVE*			
	% Used		% ACKed		% Used		% ACKed	
	Good link	Poor link	Good link	Poor link	Good link	Poor link	Good link	Poor link
1	8.56	34.8	85.9	63.4	0	0.04	0	70
2	6.04	3.81	96.6	83.7	0.05	2.20	100	99.6
5.5	7.65	3.84	98.1	83.5	0.71	19.6	93.0	88.7
6	7.06	3.72	56.1	0.65	0	0	0	0
9	7.78	3.26	54.3	0.51	0	0	0	0
11	8.39	3.79	97.8	75.9	5.50	77.2	86.9	74.5
12	7.08	3.15	60.8	0.83	0	0	0	0
18	6.90	3.94	61.5	0.71	0	0	0	0
24	7.67	3.88	61.5	0.53	0	0	0	0
36	7.75	6.69	59.4	0.25	1.66	0.28	80.7	22.1
48	9.37	10.2	54.2	0.62	9.25	0.33	82.1	16.0
54	15.6	18.7	50.2	0.35	82.8	0.35	88.8	4.70

Table 3: Physical layer bit rates used for different RAAs. When the RSSI is below 20 dB the link is classified as being a 'Poor link'. Otherwise, it is classified as a 'Good link'. The percentage of transmissions using specific bit rates and percentage of transmissions ACKed are shown.

continues to use the highest bit rate over three quarters of the time even when the link quality deteriorates.

5.4 RapidSample

RapidSample [23] is a frame-based algorithm that has been developed specifically for mobile environments. The algorithm starts with the fastest bit rate. If a packet fails to get a link layer ACK it records the time of failure and switches to the next lower bit rate. After success at a particular bit rate for more than 5 ms the sender attempts to sample the highest bit rate that has not failed in the last 10 ms and there is no lower bit rate that has failed in the last 10 ms. If the faster rate fails, it reverts to the original bit rate; otherwise it adopts the new bit rate.

We have implemented *RapidSample* as a Madwifi rate adaptation module. It is important to note that *RapidSample* does not rely on the MRR mechanism and hence there are no retries at the hardware level. Every packet loss is reported to the algorithm and a different rate selection is made. We reflect this in our implementation by disabling multi-rate retry in Madwifi. This makes *RapidSample* very succeptible to packet losses that happen frequently in outdoor mobile environments. Also, RapidSample is tuned for indoor channel coherence times (approximately 10 ms) and walking speeds. Outdoor channel coherence time is much shorter.

From Table 3, we see that *RapidSample* uses all bit rates approximately equally when the link quality is good. This is unlike other RAAs which tend to converge at one stable rate. This highlights the problem of being overly sensitive to transmission failures which is compounded by the absence of multi-rate retry capability. During poor link conditions *RapidSample*, tends to converge at the lowest and the highest bit rates. Approximately half of transmissions happen at these two rates. The convergence at 1 Mbps bit rate can be explained by the increased number of packet losses due to poor link quality. The convergence at 54 Mbps bit rate can be explained by the tendency of the algorithm to jump to the highest bit rate that has not failed in the last 10 ms.

5.5 BRAVE

We implement the proposed *BRAVE* protocol as a rate adaptation module in the Madwifi driver (version 0.9.4). The operation of the *BRAVE* protocol has been previously described in Section 4. *BRAVE* correctly estimates the link quality and appropriately selects bit rates for outgoing packets.

From Table 3 note that *BRAVE* selects the highest 3 bit rates almost 93% of the times when the link quality is good. Also the packets sent at these rates are ACKed with a very high probability. In case of poor link quality, BRAVE selects the loss resilient 802.11b bit rates around 99% of the times. The link quality is overestimated less than 1

6. VEHICULAR LINK EMULATION

We mentioned previously the non-repeatability of vehicular experiments that prevented us from doing a real head-to-head comparison of RAAs. In this section we describe an indoor emulation environment that can vary wireless link qualities in a repeatable fashion. Though realistic outdoor channels cannot be emulated, variability in the link qualities in a repeatable fashion still is quite useful for a first-order comparison of RAAs. The essential idea is to use a software radio as a programmable noise generator.

6.1 Emulation Setup

The emulation setup as shown in Figure 6 consists of two 802.11 devices in the 'ad hoc' mode placed 1 meter apart. This ensures

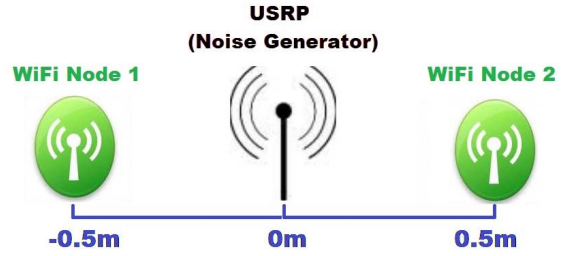


Figure 6: Emulation setup showing two WiFi nodes 1 meter apart with the USRP noise generator exactly in the middle.

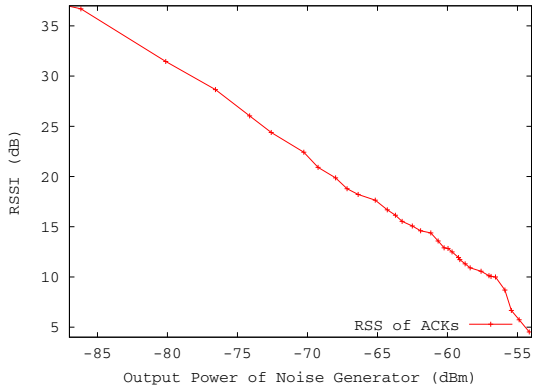


Figure 7: RSSI of received ACKs with changing output power of noise generator.

that the nominal link quality is excellent between the two nodes. We use a Universal Software Radio Peripheral (USRP) [1] device in between the two WiFi nodes acting as a noise generator. The USRP has a 2.4GHz radio transceiver and is programmed to generate Gaussian white noise over a 4 MHz bandwidth centered around the center frequency of the chosen 802.11 channel. Even though the channel width used by the USRP is small – only 4 MHz compared to 20 MHz for 802.11 – the noise floor at the 802.11 receiver can still be raised enough by increasing the transmit power of the noise generator such that the link quality is impacted adversely.

The output power from the noise generator can be increased by programmatically increasing the amplitude of the noise. This progressively worsens the link quality between the two WiFi nodes. Figure 7 shows the effect of increasing the output power from the noise generator on RSSI of the received ACKs at the transmitter validating that the generated noise has an intended effect. Notice that the RSSI decreases linearly with increasing output power from the noise generator.

6.2 Emulating Drives

We use the emulation setup described in the previous section to emulate a 50 km drive for each RAA. We assume that 50 APs are placed at regular intervals along the drive. Our goal is to vary the noise generated by the USRP such that the link quality between the

two WiFi nodes will be similar to what it would be for an actual drive at various points of time.

Choosing an appropriate propagation model is important for the emulation set up. We note that since the vehicular antenna height is short, we need to use near-ground propagation models that take into account effects due to the lack of Fresnel zone clearance. We choose the model developed and validated for 802.11 signal propagation in the 2.4 GHz band as reported in [13]. The model is as follows:

$$P_{loss} = 40Log_{10}D + 20Log_{10}F - 20Log_{10}h_t h_r,$$

where: P_{loss} is the path loss in dB, D is the distance in kilometers, F is the frequency in MHz, and h_t and h_r are the heights of the transmitting and receiving antennas in meters.

This model estimates the path loss for 802.11 WLAN line-of-sight links with antenna between 1 and 2.5 meters in height. In our emulation we assume the APs to be at a height of 2.5 m and the vehicular client to be at a height of 1 m.

The output power from the noise generator is varied using a script in order to emulate the changing distance from the APs at different driving speeds. Three different driving speeds are emulated in this fashion – 5 m/s (or 18 km/hr), 10 m/s (or 36 km/hr) and 15 m/s (or 54 km/hr). The speed remains constant during the emulation. The 'ad hoc' mode is used specifically to ensure that there is no issue with AP-client association and there is no handoff. This ensures that there is no influence from any handoff protocol on the overall throughput performance. Note that use of 'infrastructure' mode instead of 'ad hoc' mode may reduce the repeatability of the emulation. The AP-client association messages are broadcast in nature and thus are highly susceptible to losses. Hence the handoff delays can vary widely during repeat experiments. The same saturated UDP traffic is used as before and the received throughput is plotted for each RAA for different speeds (see Figure 8). Note that this emulation perhaps over-estimates the absolute performance as the experiments are stationary and indoors, and no handoff delay is modeled. Its sole use is relative comparison between different RAAs. We see that *BRAVE* still performs the best. The rest of the RAAs perform much worse, specifically at higher vehicular speeds.

At slow speeds *AMRR* seems to be better than SampleRate because of its aggressive nature. See the relevant discussion in Section V. This over-aggressiveness hurts *AMRR* at higher speeds since now it over-selects bit rates. *SampleRate*'s nature of under-selecting bit rates helps it in competition with *AMRR* at higher speeds. *CHARM* over-selects bit rates very aggressively and mostly relies on the 54 Mbps bit rate even when the link quality deteriorates. With the rapidly changing link quality experienced under vehicular mobility, *CHARM* incurs heavy retransmissions and packet losses hurting its performance significantly.

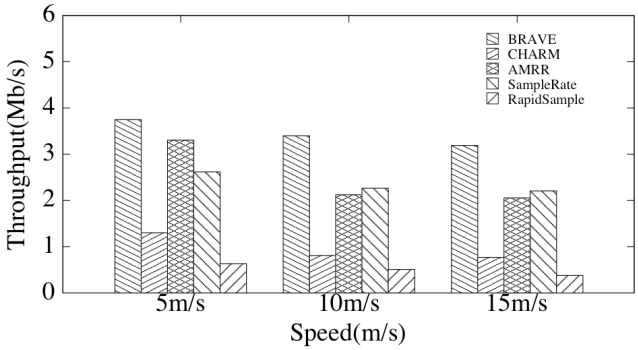


Figure 8: Throughput performance of rate adaptation algorithms in the 50 km emulated drive for three different speeds.

7. RELATED WORK

7.1 Frame Based Rate Adaptation

Frame based rate adaptation algorithms rely on link layer frame delivery statistics.

Auto Rate Fallback (ARF) [19] is a simple rate adaptation algorithm proposed in 1996 that drops the transmission rate on successive packet losses and increases the rate on successive successful packet transmissions. Since then many other rate adaptation algorithms have been proposed. Adaptive Auto Rate Fallback (AARF) [20] improves upon ARF by dynamically choosing the threshold for increasing and decreasing the bitrate instead of assuming a fixed value. This adds a measure of stability to the protocol. ONOE [5] is another simple but popular algorithm whose implementation is available in the Madwifi driver code. It decreases the bitrate when packets need at least 1 retry on average and increases it if less than 10% of the packets require retransmissions.

SampleRate [21] and Adaptive Multi-Rate Retry (AMRR) [20] are the other two popular algorithms whose implemenations are available in the Madwifi driver. We have previously described their operation in Sections 5.1 and 5.2.

All the above algorithms are designed for indoor WiFi access scenarios where channel conditions are relatively stable. In outdoor vehicular WiFi access scenarios the packet loss rates and variance in the channel conditions are much higher and the frame based algorithms tend to either under-estimate or over-estimate the link quality depending on the sizes of their estimation windows and rate adaptation procedures.

7.2 SNR Based Rate Adaptation

In case of signal-to-noise ratio (SNR) based rate adaptation algorithms the bit-rate adaptation is dependent on PHY layer statistics like RSSI.

Receiver Based AutoRate (RBAR) [15] is an SNR based scheme that relies on the RTS/CTS mechanism and a pre-computed wireless channel model to choose an appropriate bit-rate. In this scheme, the transmitter sends an RTS frame before every data frame and the receiver measures the SNR and compares it with SNR thresholds from the channel model to select an appropriate bit-rate. This bit-rate is conveyed to the transmitter as part of the CTS frame. The obvious downside of this scheme is the overhead introduced by the RTS/CTS frames.

CHARM [18] is another SNR based algorithm which uses channel reciprocity to obtain channel information while avoiding the RTS/CTS overhead. We have already described its operation in Section 5.3.

7.3 Rate Adaptation in Mobile Environments

A recent paper [23] has proposed a hint-aware rate selection algorithm which explores the possibility of using mobility hints for bit-rate selection. They propose switching between the RapidSample algorithm (described in Section 5.4) and the SampleRate algorithm (described in Section 5.1) based on whether the client is mobile or not, respectively. Our analysis in Section 5.4 shows that in outdoor vehicular environments RapidSample under-performs in comparison to all other algorithms.

The CARS algorithm proposed in [22] also accepts sensor hints to make rate selection decisions. Unlike the hint-aware rate selec-

tion algorithm in [23], CARS has been designed for out-door vehicular WiFi access scenarios. In the CARS scheme, the communicating nodes exchange GPS locations in order to track distance and relative speeds between the two nodes. This forms the current "context". Each node maintains a context to link quality mapping and fast rate adaptation is performed based on the current context. However, this scheme has the drawback of requiring a lot of training data to be collected to account for every possible context.

Camp and Knightly [9] have proposed a cross-layer framework for rate adaptation in vehicular networks. The study concludes that both frame-based and SNR-based algorithms suffer from the smaller channel coherence time in out-door vehicular environments and training SNR-based protocols according to the environment's coherence time can improve throughput performance.

SoftRate [24] is a bit rate adaptation algorithm that uses physical layer hints to make rate adaptation decisions. SoftRate is known to out-perform most RAAs in slow and very fast fading scenarios, but tends to suffer under intermediate mobility where the channel changes every few packets.

The Strider [14] system is designed to achieve optimal rate adaptation by the use of a novel code which is rateless and collision-resilient. The current implementation of this system is on a `GNU Radio` platform and only uses 6.25 MHz width instead of the full 20 MHz width required for WiFi.

8. CONCLUSIONS

The popularity of outdoor metro-scale WiFi networks is growing and there is a new interest in off-loading cellular data to WiFi networks for cost and/or congestion savings. Thus, optimizing WiFi access from moving vehicles continues to be an important issue. Physical layer rate selection is a large component of this optimization. However, all rate adaptation algorithms for WiFi networks have been studied only indoors with fairly static scenarios or with very limited mobility (walking speeds). These algorithms often uses a significant amount of history (over several seconds) to make a rate selection decision, rendering their use fairly limited in very dynamic scenarios under vehicular mobility. We have addressed this problem by developing *BRAVE*, a rate adaptation algorithm that uses some amount of training to understand SNR versus error rate performance and then dynamically uses only a brief (sub-second) history to determine the rate to be used. *BRAVE* has been compared with four other prominent rate adaptation algorithms in a metro-scale WiFi deployment exposing the advantages of using *BRAVE* vis-a-vis the other protocols. A head-to-head comparison in an emulation environment finally establishes its superiority, specifically at higher vehicular speeds.

9. ACKNOWLEDGEMENT

This work was partially supported by NSF grants CNS-0751121, CNS-0831791, CNS-1117719 and a gift from Northrop Grumman Corporation.

10. REFERENCES

[1] Ettus Research LLC. `http://www.ettus.com/`.

[2] GNU Octave, version 3.0.5. `http://www.gnu.org/software/octave/`.

[3] List of cities and counties with large WiFi networks. `http://www.muniwireless.com/reports/Mar-28-2009-list-of-cities.pdf`.

[4] Multiband Atheros driver for WiFi (MADWIFI). `http://sourceforge.net/projects/madwifi/`.

[5] Onoe Rate Control. `http://madwifi.org/browser/trunk/ath_rate/onoe`.

[6] OptimumWiFi. `http://www.optimum.net/MyServices/WiFi/`.

[7] Ubiquity Networks, Inc. `http://www.ubnt.com`.

[8] Aruna Balasubramanian, Ratul Mahajan, and Arun Venkataramani. Augmenting Mobile 3G Using WiFi. In *Proceedings of the ACM MobiSys Conference*, 2010.

[9] J. Camp and E. Knightly. Modulation Rate Adaptation in Urban and Vehicular Environments: Cross-layer Implementation and Experimental Evaluation. *IEEE/ACM Trans. on Networking*, 2010.

[10] Pralhad Deshpande, Xiaoxiao Hou, and Samir Das. Perfomance Comparison of 3G and Metro-Scale WiFi for Vehicular Network Access. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2010.

[11] Pralhad Deshpande, Anand Kashyap, Chul Sung, and Samir R. Das. Predictive methods for improved vehicular WiFi access. In *Proceedings of the ACM MobiSys Conference*, 2009.

[12] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: A WiFi-Based Vehicular Content Delivery Network. In *Proceedings of the ACM MobiCom Conference*, 2008.

[13] D.B. Green and A.S. Obaidat. An Accurate Line of Sight Propagation Performance Model for Ad-hoc 802.11 Wireless LAN (WLAN) Devices. In *Proceedings of IEEE International Conference on Communications (ICC)*, 2002.

[14] Aditya Gudipati and Sachin Katti. Strider: Automatic Rate Adaptation and Collision Handling. In *Proceedings of the ACM SIGCOMM conference*, 2011.

[15] G. Holland, N. Vaidya, and P. Bahl. A Rate-adaptive MAC Protocol for Multi-hop Wireless Networks. In *Proceedings of ACM MobiCom*, 2001.

[16] Xiaoxiao Hou, Pralhad Deshpande, and Samir R. Das. Moving Bits from 3G to Metro-Scale WiFi for Vehicular Network Access: An Integrated Transport Layer Solution. In *Proceedings of the IEEE ICNP 2011 conference*.

[17] Ajinkya Uday Joshi and Purushottam Kulkarni. Vehicular WiFi Access and Rate Adaptation. In *Poster in SIGCOMM*, 2010.

[18] G. Judd, X. Wang, and P. Steenkiste. Efficient Channel-aware Rate Adaptation in Dynamic Environments. In *Proceeding of the ACM MobiSys*, 2008.

[19] A. Kamerman and L. Monteban. WaveLAN®-II: A High-performance Wireless LAN for the Unlicensed Band. *Bell Labs technical journal*, 1997.

[20] M. Lacage, M.H. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *Proceedings of the ACM MSWIM*, 2004.

[21] Robert T. Morris and John C. Bicket. Bit-rate selection in wireless networks. Technical report, Masters thesis, MIT, 2005.

[22] P. Shankar, T. Nadeem, J. Rosca, and L. Iftode. Cars: Context-aware Rate Selection for Vehicular Networks. In *Proceedings of the IEEE ICNP*, 2008.

[23] L.R. Sivalingam, C.C. Newport, H. Balakrishnan, S.R. Madden, et al. Improving Wireless Network Performance Using Sensor Hints. In *Proceedings of the USENIX NSDI*, 2011.

[24] Mythili Vutukuru, Hari Balakrishnan, and Kyle Jamieson. Cross-Layer Wireless Bit Rate Adaptation. In *Proceedings of the ACM SIGCOMM*, 2009.

[25] Starsky H. Y. Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *Proceedings of the ACM Mobicom*, 2006.