Abstract—Traditional rate adaptation algorithms (RAAs) suffer under congested scenarios as they are unable to differentiate between packet errors due to poor physical channel conditions, from those due to collisions arising from contention for the channel. This degrades the rate selection which results in a dramatic loss of throughput both for individual users and the system. This paper presents ‘SmartRate’, a throughput and packet size aware, passive measurement based, client-side RAA. It employs 802.11 MAC fragmentation for physical PER isolation, and employs a dynamic sampling regime to collect per-rate statistics. It uses a novel RSSI based volatility adaptivity mechanism to fine tune various parameters of the algorithm under different channel conditions (stationary or mobile). We compare SmartRate against SampleRate and AMRR in our wireless testbed and show that SmartRate outperforms them in single and multi-user cases.

I. INTRODUCTION

The explosion of WiFi enabled smart phones, netbooks, notebooks, iPads and iPods present unique challenges, where the stations (devices) are more mobile and have to compete for crowded wireless channel access. A new rate adaptation algorithm (RAA) which is designed from the ground up and is aware of these problems is needed.

The existing algorithms suffer from a number of drawbacks. First, their inability to isolate the physical errors causes huge loss in throughput in congested scenarios. Second, non-adaptive static parameters set in the algorithms for tasks – like sampling and rate decision updates – result in sub-optimal performance under different channel conditions (e.g. the parameters might be optimal for a stationary situation, but would degrade performance under mobile situations). Third, existing algorithms generally exclusively rely on a single key metric for all their decisions, e.g. PER, ignoring other available information like RSSI. By ignoring the full picture available to them, RAAs will invariably suffer under extreme conditions. Fourth and lastly, these algorithms do not take into account the power consumption in terms of computing needed to make decisions. Thus they might drain the battery unnecessarily under stationary conditions.

Current RAAs do not distinguish between errors caused by contention or physical reasons and thus suffer badly in multi-user environments. We used frame reservation by using fragmentation to isolate the second (and subsequent) fragments from contention errors in [1], and by employing this technique we enhanced the performance of two existing algorithms (SampleRate [2] and AMRR [3]) in congested scenarios, for both single user and system level, in [4]. However we did not address the wider systemic problems in these RAAs there. An important reason underlying their shortcoming is that these algorithms were designed and developed for single users with limited mobility. They are blind to other station’s presence and hence have poor performance in congested scenarios. In addition the design of the algorithms themselves is not very adaptive and can not cope with the increasingly complex behaviour of the wireless devices currently in use.

In this paper, we look at designing a new dynamic RAA, SmartRate, which could sense the change in channel condition and is able to isolate the physical packet errors from contention ones. The following are the key novel contributions:

1) This is the first RAA proposed that can achieve clean separation of Physical PER (PPER) for different packet sizes through the use of fragmentation that fits with the IEEE standard. Compared to our in-principle work on fragmentation, we had to solve practical issues and design tradeoffs to make this work within a RAA, which results in compromises involving packet-size selection, and hence sampling strategy.

2) We use RSSI in a novel way, to detect channel volatility, that exploits the useful information it carries whilst avoiding its pitfalls. In addition, SmartRate is robust as it fine tunes the design parameters of the algorithm according to the channel conditions.

3) We use throughput aware PPER, so that we can benefit both from the knowledge of true PPER and the benefit of aiming directly for the performance metric, throughput.

The paper is organized as follows. Section II describes our wireless testbed. Section III discusses related work. Section IV discusses the challenges associated with the design of a new RAA. Section V presents the design and architecture of SmartRate. Section VI evaluates the performance of this algorithm in our testbed and compares it to other RAAs.

II. TESTBED

Our testbed consists of wireless clients, access points, PCs and traffic generators. In total we have 16 nodes, all placed in a conference room. We use Linux with open source driver code, and focus on 802.11{a} to avoid interference from neighboring 802.11{b,g} networks. Experiments were conducted in a secluded office both during the day and at night. We use the 802.11e extension to 802.11a in best effort mode only, where it behaves essentially like 802.11a. The possible data rates for the data packets are $r = \{6, 9, 12, 18, 24, 36, 48, 54\} \text{ Mbps}$. 

978-1-4577-0351-5/11/$26.00 \copyright 2011 IEEE
The wireless hardware consists of Linksys [5] and Netgear wireless cards, based on Atheros 5212 chipsets [6], which can operate in each of 802.11{a,b,g}. We used the Multiband Atheros Driver for Wireless Fidelity (MADWIFI version 0.9.4) [7], a Linux kernel driver for Atheros-based Wireless LAN devices. The PCs are each Pentium III or higher, running Fedora Core 5, kernel version 2.6.18. All wireless nodes are connected to a central machine via a wired Ethernet network to control and manage experiments. The control software sets up the wireless network, loads the drivers, starts the traffic generators, takes the measurements (via two sniffers), and stores the experiment results. We make extensive use of accurate passive monitoring. Atheros based wireless cards provide a monitor mode, similar to promiscuous mode in wired Ethernet, whereby all valid packets within range of the card on a given channel are captured. Monitored IP packets have MAC headers and physical layer PRISM headers appended, and are subsequently caught via tcpdump using the libpcap [8] capture library and stored. The packet traces are processed by programs we wrote in C++ and Matlab to access the information required at different protocol levels.

We use IEEE 802.11’s Long Retry Counter (LRC) to calculate Packet Error Rate (PER) precisely at the sender. The MAC retry bit is used for PER estimation of packets captured by the monitor or sniffers but originating from other stations. For competing background IP traffic we use D-ITG [9] to generate UDP traffic of various types.

III. RELATED WORK

A congested environment experimental study of rate adaptation (control) algorithms in the ORBIT testbed [10] was presented in [11], which shows that all error-threshold [12], [3], [2] based algorithms suffer from contention errors and unnecessarily decrease the throughput.

Recently CARA [13] has been proposed which can distinguish between physical errors and collision errors by employing RTS/CTS. In [14], a RAA is presented which uses the CARA based RTS/CTS scheme to avoid collision based errors in rate estimation. Their RAA is a much improved version of ONOE, a native MADWIFI rate adaptation algorithm. However, they do not consider the modern multi-rate retry feature of current cards nor do they test it on MADWIFI based platform. In addition their evaluation is limited to a single AP. Two estimators for separating physical errors are presented in [15]. The first separates the physical error based on channel utilisation. The second (less efficient) one tries to separate out errors based on CRC checking of the physical header compared to the MAC frame CRC. In [16] a simple passive heuristic based on the correlation between contention error and channel utilisation, is applied on the SampleRate algorithm. This heuristic may not hold in all cases, and their approach is vendor (Atheros) specific if the overhead of monitoring mode is to be avoided. In [17] a scheme is described to distinguish collisions from physical errors, which requires comparing frames transmitted from the AP to those received at the client station. The need for cooperation from the AP limits the practical applicability in real networks.

In [18] a new rate adaptation algorithm CHARM is presented, which uses RSSI as its primary metric for rate selection. It improves on other SNR-based techniques (like [19]) by using dynamic SNR based thresholds and relies on the receiver frame’s RSSI to estimate the channel path. CHARM has many short comings. First, it is not compatible with legacy devices. Second, if a device is using a power control algorithm (which should be prevalent in modern cards for energy efficiency reasons) then their technique suffers greatly. Third, they use channel monitoring to collect the information regarding other nodes and to snoop on probe packets of other nodes, which is not energy efficient and will not work in secure networks.

We now describe SampleRate and Atheros Multi Rate Retry or AMRR, each freely available with MADWIFI. Each employs algorithms designed to take measurements, keep per-rate statistics, and then act on them. SampleRate represents a class based on throughput and AMRR on error rates. Adaptive Auto Rate Fallback (ARF) [20] is a simple extension of ARF, where if a transmission attempt at a higher rate fails the time to the next try is exponentially increased. The MADWIFI implementation, AMRR, exploits the multi-rate aspect of Atheros cards whereby the retransmissions occur at a lower rate using the multi-retry chain. A higher rate is tried if at the current rate less than 10% of packets are retransmitted, and a lower one if more than 30% are retried. SampleRate [2] tries to find the data rate with the highest throughput by maintaining throughput statistics for each rate. SampleRate also samples a different rate for 10% of its packets. In [2] it is shown to perform better than ONOE and AMRR for a single station.

Two approaches of isolating the physical error from contention based on channel utilisation and fragmentation are detailed in our prior work [1], where we show that fragmentation based approach outperforms the other approach. In [4] we evaluate the RTS/CTS and fragmentation based approach at both user and system level for AMRR and SampleRate. In [4], we only change the input of RAAs (SampleRate and AMRR) by passing only isolated physical PER information. We do not in any way change the underlying RAAs but still enhance their performance in congested cases. However [4] does not describe how one should implement the isolation technique in a RAA designed from scratch. In this work we design a new algorithm, SmartRate, which avoids the traditional pitfalls, and compare its performance against SampleRate and AMRR.

IV. RAA DESIGN CHALLENGES

In this section we examine the key issues for designing a new robust, passive and independent RAA.

A. Using RSSI

The Received Signal Strength Indicator (RSSI) is measured over the preamble and physical header of a received frame and can be converted to an SNR value. It is used as a key metric for rate selection in [19], [18]. The problems using RSSI for rate selection are: First, it is only available over the PHY layer,
transmitted at the base rate, and thus does not provide a SNR for every rate or over entire frame length; Second, it can only be measured at the receiver and thus need to be passed back to the sender, which would involve a feedback mechanism lacking in the 802.11 standard; Third, it is a rough measure of SNR which depends on a particular manufacturer for its accuracy, resolution and precision.

The change in wireless channel causes the variation in the RSSI of received frames. The absolute value of an RSSI is difficult to interpret, but the variations of this value is symbiotic of change in wireless channel. RSSI is a per-frame metric available to every 802.11 device. We get high resolution data as it comes with every frame and is independent of packet size and data rate. As the RSSI is only available at the receiver, the Data frame RSSI could not be used, instead we use the ACK frame, acknowledged for every Data frame by the receiver. This ACK frame is close in time to the Data frame and thus would on average experience the same channel condition as that of data frame. We do not have to even assume channel symmetry for using the ACK frame’s RSSI at the sender, as we are not using its absolute value, but its variation. SmartRate uses the ACK’s RSSI variations for channel volatility estimation.

B. Coping With Volatility

By volatility we mean rapid changes in the wireless channel quality. Low volatility is generally found in statistically stationary and immobile (or static) environments, while high volatility could be caused by a moving (mobile) station or movement of objects or people in the wireless path. Channel volatility is a hallmark of wireless links. Current RAAs do not detect these different volatility cases and thus lead to sub-optimal results. E.g., in a mobile environment where the channel is changing in the timescale of tens of milliseconds, a rate selection decision that has to wait for half a second would suffer by not adapting to the channel quickly. Similarly, a RAA which is making rate decision rapidly in a stationary environment (changing in the timescale of seconds), would be wasting energy by doing the unnecessary computations.

The channel volatility impacts frames PER and RSSI. We use RSSI metric, independent of packet size and data rate (thus PER is not suitable), for estimating channel volatility. The rapidly changing environment causes a rapid change in the RSSI value and would result in higher variance and vice versa. In SmartRate, we extract a signal for channel volatility over an interval by discretizing the magnitude of the RSSI variance into three environments, namely: low (stationary), medium (for mild variations caused by the movement of objects or people in the wireless path) and high (mobile) volatility.

C. Physical PER Isolation

The increasingly crowded wireless access add to the challenges faced by the stations. RAA performance suffers in congested scenarios if it confuses PPER with contention PER as shown in [4] for SampleRate and AMRR. In [4] we provide a fragmentation based solution to enhance the throughput of SampleRate and AMRR in congested cases, however it does not describe how it could be incorporated inside a RAA.

SmartRate uses fragmentation to achieve PPER isolation. The fragmentation based method does the isolation by dividing a packet into fragments and use only the protected frames for PER measurement. Each packet above a minimum threshold could be fragmented, but this would induce an extra overhead. Thus we fragment only a fraction of the packets, which makes the isolation efficient and effective by sacrificing little throughput. But the enormous advantage in congested scenarios makes it justified. In SmartRate, all packets used in PER measurements are fragmented, and only second and subsequent fragments are used for PPER measurement.

D. Packet Size Specificity

Packets of different sizes could experience different PER for the same channel condition. Most RAAs (e.g. AMRR) ignore this size dependence and consider one representative size for all packets. SampleRate categorizes the packets into three bins and thus provides an improvement over other RAAs, but relies passively on the user traffic for statistics collection.

A RAA would require a large measurement set to cover all packet sizes, which would be really hard. A way to reduce this complexity is to divide the size distribution into three bins, centred around the tri-modal size distribution represented in the Internet. It serves three purposes: First, as those sizes are often used, their statistics could easily be gathered; Second, it simplifies the state requirement on the memory and computation; Third, the corresponding rate measurements will be directly relevant for the majority of packet sizes seen. A key issue is size control of the protected frames, which are smaller than the packets they are derived from. The largest fragmented frame is half of $\text{maxframesize}$ (2346 bytes for 802.11 MAC). In SmartRate, our approach to obtain the characteristic of the large frame is to use a simple model using the largest available protected fragment. Specifically the PER of a fragmented packet depends on the PER of its fragments. If we assume that the fragment’s PER is independent and identical (assuming equal size fragments and only considering physical errors) then PPER of the packet can be represented as

$$
\text{PPER} = 1 - (1 - \text{PER}_{\text{frag}})^{\frac{1}{\text{l}}},
$$

Here $\text{PER}_{\text{frag}}$ is the PER for the protected fragment. $\text{Fm}$ is the fragmentation threshold and $\text{l}$ is the packet size.

E. Data Sampling

All RAAs have to test a data rate (known as $\text{sampling}$) to know its characteristics. The PER dependence on packet size complicates the picture further and needs size based sampling. This two dimensional sampling raises the issues of memory allocation and statistics gathering in the context of a non-uniform packet size distribution traffic flow. Sampling a rate causes overhead and loss of throughput by: wasting transmission time in sampling lower rates; or sampling more error prone higher rates.

We need to sample rate and size. Sizing is explained in the previous section so we focus now on rate. Current rate is
the RAA recommended rate used by a normal unfragmented packet, while \textit{sample} rate is used to collect statistics for decision purposes using fragmented \textit{sampled} packets. Rates should not be uniformly sampled, as there is a natural asymmetry between exploration above and below the current rate, as below punish the system while up is quick even if it fails. SmartRate adapts sampling according to the rate’s previous behaviour, e.g., if a higher rate has not been successful then its frequency of sampling is lowered, controlled by its PPER. From the available rate set, a subset of \textit{allowed} rates is chosen which are the ones RAA will ever recommend. This allows for the known problems with some rates (e.g., the 9 Mbps rates is shown to perform worse than 12 Mbps in [2]) and are thus excluded from the sampling decisions to reduce overheads. In addition the network settings might not allow certain rates.

F. Adaptive Decision Making

Any algorithm based on a static scheme working in a dynamic environment would not be optimal in all cases. The selection mechanism is invoked periodically to update the rate decisions according to the rate statistics available. Two key parts of this are the update interval and relevancy of previous statistics. Generally the RAAs have a fixed update interval (e.g. 2 and .5 sec intervals for SampleRate and AMRR respectively). The relevancy of the collected statistics depends on channel conditions. For example, in a low volatility case the channel would not change rapidly and a low weight could be given to recently collected data and vice versa.

There are many aspects of a RAA which if adapted according to the environment would make it more robust, like: rate selection interval, rate and packet size sampling, and the value of thresholds or weights depending on statistical confidence. This update (selection) time interval is an important design variable, which should respond to the changing environment. In SmartRate, the update interval is made a non-increasing function of volatility level. In SmartRate, we use exponential weighting moving average (EWMA) smoothing on the collected statistics in each interval. As the time-scale changes with the environment, this naturally gives more weight to recent statistics in high volatility cases.

G. Optimizing Throughput

SampleRate estimates the throughput by explicitly taking into account the time wasted by the retransmission of the errored packets. AMRR does not take this approach and hence suffer in some cases. E.g., in the 802.11b wireless network setting, a 11 Mbps data rate with a 35% PER would still have higher throughput than the next available rate of 5.5 Mbps (assuming extreme case of 0% PER). In such a case AMRR would select the later as the best performing data rate, while the SampleRate would select the former. While SampleRate choice seems better, it needs to calculate the throughput for each data rate, which is computationally a very expensive operation and is subjected to error of various sorts.

Once we have the characteristics of the data rates, we need to select the best according to their throughput. Throughput optimization criteria needs to be efficient and robust. By \textit{throughput aware} rate selection, we mean a rate selection based on maximizing the global goodput, even though the underlying measurement might not be throughput. In SmartRate, the metric measured directly is PPER for different (discretised) packet sizes and rates. Specifically, the rate delivering the highest throughput needs to be selected for each particular packet size bin. We use throughput weighted success probability (1-PPER) as our selection metric (obtained by multiplying the physical success probability (1-PPER) with the IP throughput of a packet) and recommend the rate with the highest value at the start of each update interval. See Section V-E for more details. In SmartRate, we optimise throughput without having to try to calculate it on every frame (like SampleRate) making it computationally efficient.

As shown in Table I SampleRate and AMRR fail to tackle some of these issues. SmartRate is described next.

<table>
<thead>
<tr>
<th>Feature</th>
<th>SampleRate</th>
<th>AMRR</th>
<th>SmartRate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling</td>
<td>Done, 10%</td>
<td>Done, Adaptive</td>
<td>Done, Adaptive</td>
</tr>
<tr>
<td>Decision Interval</td>
<td>2 sec</td>
<td>5 sec</td>
<td>1-2.5 sec Adaptive</td>
</tr>
<tr>
<td>Main stat</td>
<td>Throughput</td>
<td>Total PPER</td>
<td>Throughput weighted PPER</td>
</tr>
<tr>
<td>Packet Isolation</td>
<td>None</td>
<td>None</td>
<td>Fragmentation based</td>
</tr>
<tr>
<td>Packet Size Bins</td>
<td>3</td>
<td>1</td>
<td>3, active sizing</td>
</tr>
</tbody>
</table>

V. SMARTRATE: ARCHITECTURE AND DESIGN

The scheme is passive measurement based, and requires no co-operation from other stations nor modifications to the 802.11 MAC. It is incrementally deployable and benefits each user as well as the whole system. Preferred (and backup) rate decisions are made at the beginning of each ‘time bin’, known as \textit{interval}, based on measurements made over previous intervals, and are used by all normal packets over the interval. Rate decisions are PER based, obtained using fragmentation to isolate PPER, and estimates are stored per-rate. The preferred rate is selected based on weighting the per-rate PPER estimates to determine that with the highest expected throughput. To reduce overhead, fragmentation is only performed for particular ‘sampled packets’, which explore rate space and packet size. Variability in RSSI is used as a volatility detector, and used to tune key parameters, in particular the time interval width.

Conceptually this algorithm can be applied to any 802.11 based device, but here we focus on the architecture provided by the Atheros based devices, where an RAA works as a Rate Control Module (RCM) in the driver. Let available rates be \( r = \{r_{\min},...,r_{\max}\} \), current preferred rate be \( r^* \), backup rate \( r_b^* \) and sampled rate be \( r_s \). We describe the various aspects of SmartRate algorithm here.

A. Statistics Collection

Statistics are collected for measurement of volatility and PER over packet and rate space. Volatility (caused by mobility or environment) is accessed via RSSI (see Volatility Adaptivity V-B). Physical PER is measured using the LRC counter of packet fragments, for discretised packet size bins and for all rates (see PPER Measurement subsection V-D).
B. Volatility Adaptivity

The purpose is to optimise different algorithmic parameters for static versus highly variable environments by using RSSI.

1) Estimation: Our main metric is sample variance calculated on the ACK frames RSSI values received over each update interval using an on-line algorithm. We discretise the sample variance values into three levels: low, medium and high volatility, \( v = \{L, M, H\} \), based on ratios relative to the full RSSI range (i.e. map range into \([0,1]\)). We currently use ratios \( \{v_L, v_M\} = \{0.05, 0.1\} \), which we found to work well empirically for Atheros based devices. These values may need to be recalibrated for other device manufacturers.

2) Time Scale Dynamics: The rate update interval, \( \tau \), is the basic element which drives the whole algorithm. We set \( \tau_n \) for the \( nth \) interval to discrete values chosen from \([500, 250, 100]\) msec as a function of \( v \) and the current value. The algorithm basically sets \( \tau_n = \tau(v) \), but delays this in downward (less volatile) direction because: we need to be conservative and stay reactive if the environment is more volatile; to avoid unstable dynamics due to variance being close to a state transition (causing ‘flip-flop’ behaviour). The delay lasts for the duration of maximum time interval, e.g. it would last 500 msec for transition from high to low volatility.

On the other hand a switch from low to medium to high should be done immediately, since again, if the environment has become more volatile the algorithm needs to adapt to it as soon as possible. This asymmetric behaviour ensures the robustness of SmartRate. The parameters in SmartRate which are a function of volatility are:

- Underlying intervals \( \tau_n \) driving RCM as above.
- Sampling rate parameters \( \alpha_s(v) \) for overall rate sampling, \( ratedown(v) \) and \( rateup(v) \) for trading off rate coverage versus system performance [see next Section].
- \( \alpha(v) \) controlling timescale of EWMA for recommended rate (effectively gives the number of update intervals over which PPER estimates are made/averaged) [Section V-D].

C. Sampling Algorithm

The purpose of sampling is to explore the rate and packet size space for subsequent PPER measurement. If the packet is too small (smaller than fragmentation threshold i.e. 256 bytes), then it won’t be sampled. Packets will be fragmented into three frames at most (two would be better for overhead, but three needed for packet size variety). Rates are selected out of an ‘allowed rate’ vector, which is a subset of all possible rates.

There is a retry chain in Atheros based cards which specifies the number of times any Data frame can be retransmitted and at what rates. Up to four different rates with associated retry limits could be specified. For example, for normal packets the first rate in the chain is the recommended rate, \( r^* \), while the backup rate, \( r_b^* \), ideally is usually placed as the second entry in the chain. The third and fourth rates are generally specified to try to ensure the eventual delivery of the frame. The frame is discarded if all the rates and retries are exhausted, and a frame transmission failure error is reported. A compact way of writing this retry chain for normal packets in SmartRate is \( \{r^*; 2, (r_b^*; 2), (r_{min}; 4)\} \), which means two attempts at rate \( r^* \), two at rate \( r_b^* \) and four at rate \( r_{min} \).

1) Sampling Rate: Packets of a station are sampled using average per-packet rate \( \alpha_s(v) \) according to the volatility. Currently the values are \( \alpha_s = [2, 4, .6] \) for \( v = \{L, M, H\} \). The increase in the sampling frequency with increasing volatility makes the algorithm more vigilant in the changing environment. The rate coverage depends on the following principles:

- Uniform covering is not desirable and should be adaptive.
- Natural asymmetry between exploration above and below \( r^* \), since below punishes system, up is quick even if fails.
- Only the 1st attempt of 2nd and 3rd fragments can yield protected measurements, therefore rate coverage is performed only on them.
- For all others, rates are irrelevant for measurement and so are chosen to minimize system impact.

From these observations, the sampling is divided into three parts: sampling rates higher than the current rate, denoted by \( rateup(v) \); sampling rates lower than current rates, specified by \( ratedown(v) \); sampling current rate specified by \( currentrate(v) = 1 - ratedown(v) - rateup(v) \). It is important to avoid a performance hit by sampling lower rates too often under good conditions. The vector \( (ratedown(v), currentrate(v), rateup(v)) \) controls the proportion of sampled packets and hence attempted relevant fragments sent (below,at,above) \( r^* \). Currently the values we use according to the volatility are: \( ratedown = [0.1, 0.1, 0.2] \), \( rateup = [0.5, 0.5, 0.6] \) and \( currentrate = [0.4, 0.4, 0.2] \), for \( v = \{L, M, H\} \). Note that these rates apply to packets already sampled.

We start with a uniform distribution of the \( \text{resampling} \) (already sampled packets) rates in the up \((r^*, r_{max})\) and down \((r_{min}, r^*)\) direction. Then we learn about the success probabilities (1-PPER) of these rates, and use this resulting distribution for rate sampling. So the share of any rate having very poor success probability would be reduced and allocated to other rates. We set the minimum success probability to 1% (instead of 0), so that those rates still have a chance of getting sampled. In addition, with every change in volatility state this distribution is again set to uniform, and it relearns the sampling pattern according to the channel condition. This adaptive sampling regime ensures that we avoid sampling unsuccessful rates most of the time.

a) First Fragment: It is treated the same as normal packets in terms of rate-retry chain as it does not take part in measurement and thus the rates are chosen to optimize throughput and minimize user and system impact.

b) Subsequent Fragments: Principle is to control rate on the first attempt, but allow subsequent attempts to follow the normal packet retry chain. As before, only the first attempt is protected and is useful in measurement, the subsequent attempts could be corrupted by the collision based errors and hence sent to minimize system impact. Hence

- \( Rateup \) case: Rate \( r_s \) chosen from allowed rates in \((r^*, r_{max})\) according to their success probabilities distri-
bution and rateup proportion. If \( r^* = r_{\text{max}} \), don’t sample. Retry chain is \( \{(r_s, 1), (r^*, 1), (r^*, 2), (r_{\text{min}}, 4)\} \).

- **Currentrate case**: Same as for normal packets.
- **Ratedown case**: Rate \( r_s \) chosen according to success probability distribution and ratedown proportion from allowed rates in \( [r_{\text{min}}, r^*] \). Retry chain first entry is \( (r_s, 1) \), the rest are as for normal packets with first entry omitted, i.e. \( \{(r_s, 1), (r^*, 1), (r^*, 2), (r_{\text{min}}, 4)\} \).

2) **Packet Size Coverage**: Distribution of packet sizes control what is possible (typically‘tri-modal’ [40 576 1500’], but not always). Fragmentation alters frame size and makes sizes above 1500/2 unavailable, at least for TCP/IP packets. So, we need to bin normal packets coarsely to have enough data per bin. We gather statistics for three packet size bins covering main modes of tri-modal size distribution. i.e. \((S, M, L) = (\text{min} - 300, 301 - 900, 901 - \text{max})\), which are controlled with parameters \( \{\text{psize}M, \text{psize}L\} = \{300, 900\} \).

We base all samples on 1500 byte packets as they are typically plentiful (highest size TCP/IP packet), and gives greatest flexibility for fragment sizes. We make use of two fragment thresholds: \( F_m = 750 \), which divides the packet into two 750 bytes frames; \( F_m = 730 \), which gives three fragments, two 730 bytes and one 40 byte fragment (typical size for TCP ACK packet). We fragment the packets twice more often than thrice to reduce overhead. We use \( F_{m\text{ratio}} \) parameter to control this and currently set it to \( F_{m\text{ratio}} = 0.6 \).

Measurements per packet size bins are derived from these protected fragments as:

- S : use 40 bytes.
- M : use 730 and 750 bytes, we don’t distinguish between them for simplicity.
- L : infers 1500 from middle bin based on bit error model, i.e. \( \text{PPER}_{1500} = 1 - (1 - \text{PPER}_\text{middlebin})^2 \) (see (1)).

3) **Sampling dynamics**: We pre-filter packets to find suitable sized packets (currently 1500 bytes TCP data packets). We need to guard against a low supply of packets. We use a simple RSSI model for initialization and also use it in case of data shortage (an extreme case of having not enough measurements). Empirically the RSSI values could be categorized into four types: bad, ok, good and excellent, as rough signal strength estimates. Safe rates could then simply be assigned to these RSSI estimates. This assignment will vary but could be empirically derived for each card manufacturer. We assign the rates \( r^N = \{6, 12, 24, 48\} \) to average RSSI values \( \text{RSSI}_{\text{Avg}} = \{0.25, 0.34, 0.42, 0.71\} \text{RSSI}_{\text{MAX}} \) (of the full RSSI range, i.e. map range into [0,1]), to all bin sizes, for Atheros based devices in IEEE 802.11a/g operating setting. We sample the packets passing the filter periodically for rate and then select packet size as above for the collection of statistics.

D. **Physical Packet Error Rate Measurement**

Our algorithm is based on PPER estimates made over each update interval. We need to combine those measurements over adjacent intervals. The two important aspects are: number of samples in the current interval; their importance and quality compared to the previous intervals. The number of samples cannot be controlled, as it is a function of a station’s traffic pattern. The relative importance of the current measurements can be guessed from the volatility. i.e. under high volatility, current measurements are more important than past measurements which may be irrelevant. Under low volatility, previous measurements can be used to reduce estimation variance.

We use a configurable exponential weighted mean average (EWMA), as it allows: time intervals to be used easily even when their duration changes; replaces complex estimates of statistical quality and provides ‘fallback logic’ when samples are very rare; forgets the past gracefully but allows estimation timescale to be controlled.

1) **PPER Measurement Per Time Interval**: PPER estimate for a rate \( r \), of frame length \( l \), over time interval ‘\( n \)’ is the proportion of relevant (second and subsequent) fragment frames whose LRC field exceeds 0:

\[
\text{PPER}(n, r, l) = \frac{\text{No. of relevant fragments of rate } r \text{ and size } l \text{ with LRC \geq 0}}{\text{Total no. of 2nd fragments sent}}
\]

2) **EWMA PPER**: The exponentially averaged PPER of the above is given by:

\[
\text{PPER}_{\text{EWMA}}(n, r, l) = \alpha \text{PPER}(n, r, l) + (1 - \alpha)\text{PPER}_{\text{EWMA}}(n - 1, r, l)
\]

We use the timescale of the measurements to select the smoothing factor. Thus in the above equation the smoothing factor \( \alpha \) depends on the effective averaging time scale, i.e. \( \text{mt timescale} = [ts_L, ts_M, ts_H] \) for \( v = [L, M, H] \). Currently we use \([ts_L, ts_M, ts_H] = [5, 2.5, 1])sec and calculate \( \alpha = \frac{\text{mt timescale}(v)}{v}+1 \), where \( \text{mt timescale}(v) \) is the number of time intervals. Therefore \( \alpha(v) = [0.18, 0.18, 0.18] \). The input parameters here are meaningful timescales. We choose 5 sec for the low volatility; 2.5 sec for med, and 1 sec for high volatility as the effective measurement intervals. We selected the auto-regressive parameter \( \alpha \) according to the effective time-scale and the update interval width. Although currently the \( \alpha \) vector has equal entries, this mechanism is there to be more general and it is expected that further experience with SmartRate will reveal better settings.

E. **Throughput Aware PPER**

We are measuring success probability (1 - PPER), which has to be made throughput conscious. To do this first calculate the IP transmission throughput for a packet of size \( l \) and rate \( r \) and then multiply it with the success probability (1- PPER) of this rate, to get expected throughput. As packets are discretised into three size bins, we have to pick a representative size for each bin for the purpose of its transmission time calculation. We choose \([40, 750, 1500])bytes = [S, M, L] \) as our representative packets because of their nearness to the tri-modal distribution of the packets in the Internet (apart from the middle one, which is used for inference of the large bin packet and thus chosen as 750 bytes).

1) **Calculating Transmission Time**: We calculate the transmission time \( T(l, r) \) for an IP packet of sizes \( l = \{40, 750, 1500\} \) and rates \( r = \{r_{\text{min}}, ..., r_{\text{max}}\} \) by taking into account all the physical and MAC layer overheads of the 802.11 wireless network. The Atheros driver provides a function named \textit{ath_hal_computeptime} which takes its input
the data rate and packet size and computes the packet transmission time (generally used to calculate the time duration field of the IEEE 802.11 MAC header).

2) Expected Throughput: Multiplying the corresponding success probability with the reciprocal of transmission time gives the Expected Throughput (Exp._Thr):

\[ \text{Exp._Thr}(r; l) = \frac{1}{T(l, r)}(1 - \text{PPER}_{\text{WMA}}(l, r)) \]

which is later used for selection of throughput aware rate.

F. Rate Adaptation Algorithm

We can now finally state the RAA. We initialize the recommended rates with \( r^p \) (see subsection V-C) according to average RSSI estimates. Also when the algorithm gets started, it remains in high volatility state for a minute so as to be more vigilant. The success probabilities are considered 100% at startup for all bin sizes. Once we start collecting measurements then at the start of every update interval, controlled by the interval width, the following selections are made: The current optimal rate \( r^* \) and backup rate \( r^*_p \) for: \( r^* = \max(\text{Exp._Thr}(r; p)) \) and \( r^*_p = 2 \min\max(\text{Exp._Thr}(r; p)) \), for each \( p \) in \( \{S, M, L\} \), where \( p \) is the representative packet size of the corresponding bin.

When the driver asks the RCM to provide a rate for packet transmission, first we find out the size bin the packet belongs to, i.e. \( p \) and then apply the following rate retry chain:

- Normal packet: return \( \{ (r^*_p, 2), (r^*_p, 2), (r_{\text{min}}, 4) \} \).
- Sampled packet: as per sampling algorithm.

We implemented the SmartRate for MADWIFI as a Linux Kernel Module (LKM). It interacts with the driver when it is invoked and starts to provide the rate information to the driver, and receives the error rate measurements from the driver.

VI. PERFORMANCE

We describe the working of an implementation of SmartRate and evaluate it on our wireless testbed by comparing it against AMRR and SampleRate. First, we show the volatility detection mechanism at work and how it performs under good channel conditions. Second, we present a single user case evaluation of SampleRate, AMRR and SmartRate for six different channel conditions. Lastly, we show the system level performance of all three algorithms under congested scenarios.

A. Channel Volatility Detection

To create volatility within the testbed we perform the following experiment. We move a station, connected to an AP from a good channel condition (above RSSI=50), to a bad one (around average RSSI=20), and then move it back again. The purpose is to show how the volatility detection mechanism works by first detecting the volatility and then adjusting the time interval accordingly. We start by sending a 10 Mbps 1500 bytes IP UDP (Poisson) stream from the station to the AP and move the station away from the AP. We did this on the top floor of our departmental building. We present the result of this experiment in Figure 1. It shows the mean and variance of the RSSI in the top left plot (when we move the station from good to bad environment). Notice the sudden jump in the variance of the RSSI coincides with the sudden drop in the mean value. This change in RSSI variance is used as volatility detection, which changes the time interval accordingly (shown in the bottom left plot). Then we move back to the original location and the subsequent mean, variance and interval are shown on the right top (RSSI mean and variance) and bottom (time Interval) plots. This experiment shows that RSSI change is successfully detected by the volatility detection mechanism of our algorithm which adjusts the parameters accordingly. Notice in case there is no packet transmission, in a given interval the RSSI mean drops to 0, which causes no change in sample variance.

B. SmartRate in Action

Now we show how SmartRate works experimentally. We did this experiment in a good channel condition (average RSSI=52) in a stationary environment. We first run our mapping tool (See [4] for details), where different packet sizes (fragmented and unfragmented) are sent at different rates, to establish the characteristics of the channel and show the resulting map in Figure 2. Each data point in Figure 2 represents a mean PPER value of approximately 50,000 sent packets. It shows that the channel condition is good and the only competition of rates would be between 48 Mbps and 54 Mbps. It also shows the packet size impact on the PER. We also send packets using three fragments and show the individual fragment characteristics. Having established the ground truth with the map, ideally we would like our algorithm to, in effect, estimate this map using as few packets as possible and base the rate decisions accordingly. We start a 20 Mbps, 1500 bytes UDP (Poisson) stream from the station running SmartRate to the AP with no cross traffic. At the end of the 3 minute experiment the SmartRate algorithm’s statistics are read from...
contains the RAA. We test their design and principles under different scenarios including the challenging and crucially important case of heavy congestion.

1) Single User Multiple Locations: In addition to the same driver, the best way to compare against other algorithms is to use the same station and same access point. We do all of the above as the only variable we would like to evaluate are the RCMs of SampleRate, AMRR and SmartRate. We select the following simple experiment with no cross traffic to begin the evaluation. We send simple UDP packet (Poisson) streams from the station to the AP for six different location progressively going from good to bad environments. We run streams of 10 Mbps and 20 Mbps UDP 1500 bytes IP packets using DITG. We do this for each RCM for a duration of 3 minutes and each time we reinitialize the driver and the RCM.

We do this thrice for each RAA and average the outcome.

The most important factors to consider for evaluation of an RAA are the data rate they used most of the time and what was their IP throughput. We show the mean weighted rate and IP throughput of the RCMs for the six different locations in Figure 4. The plots show the performance of the three AAAs under two medium loads: 10 Mbps (left column) 20 Mbps (right). When the channel conditions are good (left three points), shown by the average RSSI of a particular location on the horizontal axis, all the algorithms show good performance (no surprises there!). But as the situation gets progressively worse (towards lower RSSI values), the algorithms start to show differences. In particular, SmartRate out performs the others, in almost all cases. But once the channel condition is very bad (the right most point on all plots), it does not matter which data rate to use. These experiments show that the SmartRate is indeed making smart decisions in each location and recommends a better rate even under single user condition

C. Comparison with SampleRate and AMRR

Now we compare SmartRate with the two other well known RAs. As each of them are also implemented as Linux Kernel Modules and the driver version used in all algorithms is the same, the only variable is the rate control module which

the module. They are summarized and shown in Figure 3. In the left most plot we depict the PPER map as evaluated by SmartRate, by using intelligent sampling of the rate and packet size space. It bears a strikingly close resemblance to the map in Figure 2. Based on these PPER estimates, the expected throughput estimates are made and shown in the middle plot of Figure 3. The recommended pick for 1500 byte packets is 48 Mbps as shown by the map in agreement with the data rate actually sent as shown on the right most plot. The 40 bytes packet has the 54 Mbps rate selected according to its IP throughput as evinced by the true map (but the difference between the different rates is negligible because of the packet size effect of IEEE 802.11 MAC). Notice the scant amount of time wasted in sampling lower data rates as shown by the right most plot. We have shown the complete working of SmartRate in a good environment, making measurement of the channel and then acting intelligently.

Fig. 2. Map for a good location (mean RSSI=52). The PER of three unfragmented packets shown by circles (diameters represent the packet size) along with the PER of three fragments of a 1472 byte packet (diamonds).

Fig. 3. SmartRate Explanation: the left plot shows the PPER as measured by the algorithm. The middle graph converts this into expected throughput, the highest of which is used as recommended rate r∗. As shown by the right plot, most of the packets are sent using this selected rate. Also notice the sampling of the rate (most sampling done towards the higher rate, while negligible sampling towards lower data rates). All three sizes are sampled as also shown by the packets of different sizes appearing in other data rates because of packet rate and size sampling.

Fig. 4. SampleRate, AMRR and SmartRate compared on a single station under six different channel conditions. The left and right columns show the performance results of 10 Mbps and 20 Mbps IP packets (UDP) streams (Poisson) respectively. The top row shows the weighted mean rate and the bottom row plots the IP throughput. The channel condition progressively gets worse as we move on the right on the horizontal axis on all plots as represented by their average RSSI. SmartRate outperforms the other RAs.
VII. Conclusion

We have presented a new RAA, SmartRate, which outperforms both SampleRate and AMRR under single user stationary cases, and especially under congested scenarios. It uses adaptive sampling dynamics and makes informed decisions on two key measurements, RSSI and PPER. We hope to explore the dynamics of SmartRate in mobile scenarios in the future. The main reason we compare SmartRate against SampleRate and AMRR is that their code and description is freely available. A comparison with newly proposed algorithms is difficult because some of them are either implemented on propriety hardware and not openly available (like [14]), or their code is not available freely (like [18]). A comparative analysis with these new algorithms would also be considered in the future.

References