History-Aware Rate Adaptation in 802.11 Wireless Networks

Ioannis Pefkianakis*, Yun Hu†, Songwu Lu*
*UCLA Computer Science, †USTC Computer Science
Email: *{pefkian, slu}@cs.ucla.edu, † geajy@mail.ustc.edu.cn

***Abstract*—Rate adaptation (RA) is a mechanism unspecified by the 802.11 standards, yet critical to the system performance. Although many different design directions have been studied the past years [1]–[14], there are still little insights learned of how short-term channel's past performance can be utilized to limit transmissions at low throughput rates. In this paper, we conduct a systematic experimental study to expose the importance of *history aware rate adaptation* and explore new techniques to address this space. To this end, we design and implement HA-RRAA, a new robust RA algorithm which uses short-term loss ratio to opportunistically guide its rate selection, a cost-effective, adaptive RTS filter to prevent collision losses from triggering rate decrease and an adaptive probe time window to limit excessive probing at high lossy rates. Our experimental results show gains up to 63% of HA-RRAA over RRAA, RRAA+, SampleRate and ARF, in realistic field trials.**

***Keywords*-Rate Adaptation, 802.11, Design, Experimentation**

I. Introduction

IEEE 802.11 standard mandates multiple transmission rates at the physical layer (PHY) that use different modulation and coding schemes. *Rate Adaptation* (RA) exploiting such multi-rate capability, selects the best transmission rate and dynamically adapts its decision to the time-varying and location-dependent channel quality. The wide span of available rate options and the dynamic 802.11 wireless channel, make RA a challenging problem but yet unspecified by the 802.11 standards.

A number of algorithms for rate adaptation [1]–[14], have been proposed in the literature, and some [2], [6], [7] have also been used in real products. Metrics and techniques which have been widely studied for channel estimation, include probe packets [2], [3], [7], consecutive successes/losses [2], [3], PHY-layer feedback such as SNR [4], [10], [12], and long-term statistics [7]. However, there are still little insights learned of how short-term channel's past performance can be utilized to select the best throughput rate for transmission.

In this paper, we conduct a systematic experimental study to expose the importance of *history aware rate adaptation* and explore new techniques to address this space. We first conduct controlled static-client fixed rate experiments and we evaluate popular rate adaptation algorithms in various indoor locations. We observe that even in static settings the wireless channel can switch from highly dynamic time periods to relatively stable intervals. To our surprise, popular RA designs which cannot efficiently capture short-term channel's past performance, can yield up to 29% goodput[1] loss over the fixed best goodput rate.

To address these challenges, we design and implement History-Aware RRAA (HA-RRAA), an improved RRAA [6] which is based on a novel *Adaptive Probe Time Window* mechanism to limit excessive probing at high lossy rates. HA-RRAA also applies: a) *Fast Adaptation* mechanism to remain responsive to highly dynamic channel scenarios. b) *Cost-Effective Adaptive RTS* filter to suppress collision losses with minimal overhead.

The main contributions of this paper can be summarized as follows: We first empirically study rate adaptation, using an IEEE 802.11 programmable AP platform. Our study includes both controlled static-client fixed rate experiments and performance evaluation of popular RA designs in various locations. It reveals significant limitations of existing RAs to adequately utilize short-term channel's past performance. We second propose a simple generic adaptive probe time window mechanism to capture short-term history. We design HA-RRAA, an improved version of RRAA which incorporates probe time window, while it also applies novel fast adaptation and frame collision reaction mechanisms. We implement HA-RRAA in a programmable AP platform and we compare it with RRAA, RRAA+, SampleRate and ARF. Our evaluation results show gains up to 63% in realistic field trials.

The rest of the paper is organized as follows. Section II gives an overview of state of the art RA solutions and discusses their limitations to effectively capture short-term channel's past history. Section III describes our experimental methodology and Section IV presents the importance of history-aware rate adaptation using a case study. Sections V and VI present our proposed design and our implementation and evaluation effort respectively. Finally, Section VII concludes the paper.

II. State of the Art Rate Adaptation

In this section we give an overview of state of the art RAs, while we discuss their limitations to utilize their knowledge of short-term channel's past performance.

[1]Goodput is defined as effective throughput by excluding protocol overheads.

A. An overview

There were many RA proposals the past few years [1]–[14]. As the current 802.11 compliant devices do not expose PHY layer information to the device driver, and receiver's explicit feedback to the sender is not available, cross layer designs [4], [10]–[13] have been less practical. In this paper we mainly focus on popular algorithms (ARF [2], AARF [3], SampleRate [7], RRAA [6]), which make decisions solely based on the ACK, sent upon successful delivery of a DATA frame. ARF, AARF and SampleRate are *probing-based* designs in which a few data frames are periodically transmitted at a rate different from the current one to "probe" the channel. Probing success or failure indicates if the probing rate can be used for the following transmissions. On the other hand, RRAA uses a short-term loss ratio to assess the channel and opportunistically adapts the runtime transmission rate to dynamic channel variations.

B. Learning from the past: A critique

State of the art RAs do not adequately utilize the knowledge of channel's past short-term performance, which can lead to significant probing overhead as we show in Section IV. RRAA and ARF decide the transmission rate of the subsequent frames, solely based on the performance of the current transmission rate, without considering the outcome of the past probing in adjacent rates. AARF [3] seeks to fix the above limitation of ARF by doubling, the probing threshold when a probe packet fails. RRAA+ [14] improves RRAA by increasing/decreasing the probability $p[R]$ of transmitting at a rate R, when probing at R succeeds/fails respectively. Finally, SampleRate seeks to limit sampling at lossy rates by excluding from selection for 10 seconds (MADWIFI implementation), rates which faced 4 successive failures. However, the history-aware mechanisms of AARF, RRAA+ and SampleRate do not adequately address the two main dimensions of the problem.

When loss happens? RRAA+ and SampleRate consider *rate's but not channel's past performance* and update their probing thresholds for a rate R only when they probe to this rate. Keeping stale history, can lead to performance degradation especially in scenarios of intense channel dynamics (e.g. mobility).

How severe is the loss? The above history-aware mechanisms will adapt probing to lossy rates when transmissions to these rates fail without considering how significant was the loss. For example RRAA+ will halve $p[R]$ when it moves to lower rates independently if the loss ratio for R was 40% or 100%. AARF will double the probing threshold independently if the probe frame was hardware retried 1 or 10 times, while SampleRate does not distinguish cases where probe frame will face less or more than 4 successive failures.

In Sections IV, VI-B we validate our claims using real experiments.

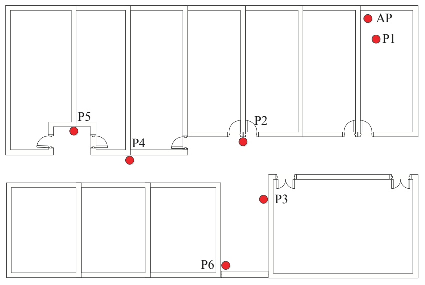

Figure 1. Experimental floorplan.

III. Experimental Methodology

The results presented in this paper are obtained from real experiments. In this section, we describe our experimental platform, setup and methodology.

Experimental Platform: We conduct all experiments on a programmable AP platform, which uses Atheros AR5416 2.4/5 GHz 802.11a/b/g/n capable chipset. The 802.11 MAC is implemented in the FPGA firmware, to which we have access. Our platform has several appealing features that facilitate our research on rate adaptation. First, we can program our own rate adaptation algorithms and run them at the AP. Second we can perform per-frame tracing of various metrics of interests, such as frame hardware retries and SNR values. Third we can configure many different parameters in real time on a per-frame basis such as: a) the maximum retry count, b) RTS option, c) the transmission rate for each frame retry. Finally, the feedback delay from the hardware layer is small, which implies that timely link-layer information is available to rate adaptation.

Experimental Setup and Methodology: We conduct all our experiments in a campus setting. Figure 1 shows the floorplan of the building where we run experiments. We placed our clients in various locations between spots P1 and P6, while our access point located at AP spot, serves as the sender of the wireless traffic. All the algorithms are implemented on the AP side. For our experiments we used various adapters as Linksys WPC600N 802.11a/b/g/n and CISCO Aironet 802.11a/b/g. For the results presented in the paper we used AirPort Extreme, Atheros (0x168C, 0x86) adapter. We conduct both static-client fixed-rate experiments and we also evaluate popular RA designs in controlled settings (interference-free) and field trials, using both TCP and UDP traffic. The results presented in the paper are for 1.5KB frames.

IV. Learning from Short-Term History: A Case Study

We start this work by raising two simple questions. How important is for RA designs to consider channel's short-term past performance? Are the existing RAs history-aware? To systematically answer these questions, we conduct fixed rate experiments in many different locations and we study in per-frame granularity their run-time performance (loss and goodput). To make sure that our observations are attributed to channel dynamics and not to collisions from hidden stations, we switch to 5GHz band (802.11a) at channel 36, which was clean during our experiments.

From our experiments, we observe time periods of highly dynamic channel to be followed by periods where channel is relatively stable (longer than 10 seconds in our experiments). The channel behavior highly depends on the environment as stated in [8]. In Figure 2, we present as a case study a 13-second trace of frame loss evolution for a scenario, where client was placed at P3 and the rate was fixed at 48Mbps. We observe that frame loss presents intense variations during the first 5 seconds of our trace, while it is relatively stable after the 5th second. More specifically, during the first 5 seconds, frame loss can vary from 0% to 69.4%, while from 5.4th to 10th second (time interval of 4.6 seconds) loss ranges from 86% to 91%. Overall, from 5.2th to 13th second, loss is greater than 42%. The average loss of our trace is 54.9%.

An efficient RA algorithm should be responsive to rapid channel changes and at the same time should limit transmissions at high lossy rates. For our case study scenario, rate adaptation should switch to 48Mbps when its loss is low and should move to lower than 48Mbps rates, when its loss is constantly very high (after 5th second). How state of the art RAs perform in this scenario?

To answer this question, we first extensively evaluate the performance of all 802.11a rates at location P3 and we present the results in Table I. We observe that rates lower than 48Mbps give loss smaller than 4%, while 48Mbps gives a significant average loss of 62.8%. Then, we study the performance of RRAA, ARF and SampleRate at that location. As RRAA and ARF do not keep any state about rates other than the current one, they keep probing low goodput rates. From Table I we see that RRAA and ARF transmit 46% and 30% of the total frames respectively at high lossy 48Mbps, 54Mbps rates. As a result, RRAA, ARF present 29%, 14.8% lower goodput comparing to the fixed best goodput rate, which is 36Mbps (Table I). Although SampleRate seeks to exclude lossy rates from sampling, as discussed in Section II-B, it still transmits 7% of the frames at high lossy 48Mbps, 54Mbps rates. This results in 6.7% decrease in goodput over the fixed best goodput rate.

Our case study reveals the need of mechanisms, which will eliminate the rates that consistently offer low goodput, by probing to these rates less frequently over time. At the same time, rate adaptation should be able to exploit the short-term opportunistic gains offered by the wireless channel.

Rates	RRAA Rate Distr.(%)	SampleRate Rate Distr.(%)	ARF Rate Distr.(%)	Fixed Rate Goodput (Mbps)	Fixed Rate Loss (%)
6				5.39	0.64
9				7.74	1.54
12				10.24	0.49
18				14.73	0.80
24	1	6	5	18.65	1.96
36	53	87	65	25.64	3.41
48	46	6	29.5	12.49	62.84
54		1	0.5	0	100
Goodput (Mbps)	18.2	23.87	21.85		
Loss (%)	33.13	9.28	22.03		

Table I
RATE DISTRIBUTION, GOODPUT AND LOSS PERFORMANCE OF POPULAR RAS AT LOCATION P3.

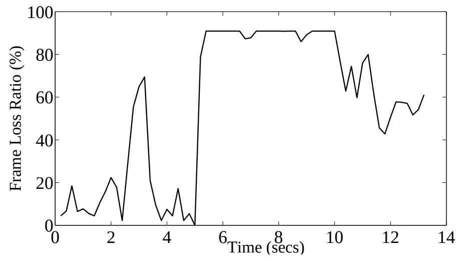

Figure 2. Frame loss ratio of 48Mbps over a 13 seconds trace. Each point is an average over 200 milliseconds.

V. DESIGN HISTORY-AWARE RATE ADAPTATION

In this section, we first present an adaptive probe time window mechanism to efficiently capture short-term channel's past performance. Then we design History Aware RRAA (HA-RRAA), an improved RRAA, which utilizes our proposed mechanism.

A. *Adaptive probe time window*

Adaptive probe time window mechanism: a) exponentially increases a probe timer upon transmission failure, b) resets the timer upon transmission success, c) bounds the timer in $[0, T_{max}]$. From one hand, the exponentially increased probe time window eliminates the rates that consistently offer lower goodput by probing to these rates less frequently over time. On the other hand, by bounding and reseting appropriately the probe window, it remains adaptive to fast channel dynamics. Probe time window is $T(R) = T_C \times 2^{exp}$, where the exponent factor exp represents the number of probing failures (upper-bounded by 10 in our prototype) and T_C represents channel coherence. Similar mechanisms have only been discussed in 802.11n context [1].

B. *History-aware RRAA*

History-aware RRAA is based on our proposed time window scheme, to limit probing overhead at lossy rates. HA-RRAA further utilizes the short-term loss statistics offered by RRAA to capture the magnitude of losses, and linearly increases probe window with loss. The new adaptive probe time window is revised as:

$$T(R) = T_C \times 2^{exp} \times max(1, \frac{P(R)}{P_0}) \qquad (1)$$

where $P(R)$ is the short-term loss ratio of the rate R and P_0 is a loss normalization factor set to 10% in our prototype.

HA-RRAA keeps only one probe window $T(R^+)$ for the next higher rate of the current transmission rate R.

Procedure 1 HA – RRAA: Input (Ack Frame), Output (R)

```
1:  R=highest_rate;
2:  timer=ewnd(R);
3:  while true do
4:      rcv_tx_status(last_frame);
5:      P = update_loss_ratio();
6:      if timer == 0 then
7:          if P>PMTL then
8:              if R!=$T\_R^+$ then
9:                  reset(exp, T);
10:             end if
11:             T =update_probewnd(R,P,exp);
12:             $T\_R^+$ = R;
13:             exp++;
14:             R = next_lower_rate(R);
15:         else
16:             if R==$T\_R^+$ then
17:                 reset(exp, T);
18:             end if
19:             if P<PORI and T==0 then
20:                 R = next_high_rate(R);
21:             end if
22:         end if
23:         timer = ewnd(R);
24:     end if
25:     send(next_frame, R);
26:     timer−−;
27:     T−−;
28: end while
```

When it moves downward from a rate R^+ to R, it will update probe window based on equation 1, while it will increase exponential exp by one. HA-RRAA will reset probe window in two cases: a) When transmission to the rate R^+ is successful (does not result in moving downwards). b) When channel's further deterioration results in moving from R to the next lower rate R^-. The pseudocode of HA-RRAA is presented in Procedure 1.

C. Handling mobility and hidden terminals

HA-RRAA further improves RRAA in mobility and hidden terminal settings.

Fast Adaptation: HA-RRAA uses a fast adaptation mechanism to be responsive to fast channel deterioration. It maintains a small window of frames (10 frames in our prototype) and computes the loss ratio inside this window. If the loss ratio $P \geq P_{Thresh}$ it moves to the next lower rate. We set $P_{Thresh} = 90\%$ for our implementation.

Cost-Effective Adaptive RTS Filter: HA-RRAA improves RRAA's adaptive RTS (A-RTS) filter to amortize RTS/CTS overhead. RRAA maintains a RTS window (RTSwnd), within which all frames are sent with RTS on. Initially RTSwnd is set to 0 and then it is adapted as follows. When the last frame is lost without RTS, RTSwnd increments by one because the loss was potentially caused by collisions. However, when the last frame transmission was lost with RTS, or succeeded without RTS, RTSwnd is halved because the last frame clearly did not experience collisions.

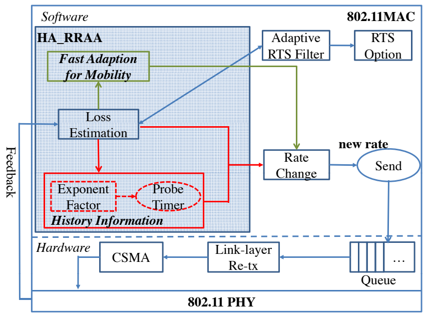

Figure 3. HA-RRAA's architecture.

Although A-RTS is trying to mitigate signaling overhead by turning on RTS only when necessary, there can be still significant overhead in the cases where frame's transmission time is small comparing to RTS/CTS overhead. HA-RRAA uses a cost-effective adaptive RTS scheme, which follows the general paradigm of A-RTS to protect data frame transmissions, but it does not blindly turn on RTS to avoid significant overhead especially at high rate options. Instead, it turns on RTS only when the overhead is outweighed by the potential savings. HA-RRAA first estimates the RTS/CTS overhead (T_{RCTS}), which is the channel time used for transmitting RTS/CTS signaling messages. Second it computes the transmission time of the frame as $T_{frame} = \frac{FRAME}{R} + T_{overhead}$ where $FRAME$ is the MAC-layer frame size, R is the transmission rate and $T_{overhead}$ includes the various 802.11 protocols overheads (SIFS, DIFS, ACK). Finally, HA-RRAA will turn RTS on only if the following condition holds: $T_{frame} \geq k \cdot T_{RCTS}$, where k is a benefit/cost ratio set to 1.5 in our prototype. The intuition besides this condition is that, without RTS/CTS, the frame may need at least one retry to get through when collision occurs.

D. Putting everything together

In Figure 3 we present the complete architecture of HA-RRAA. Upon the reception of MAC-layer feedback, loss estimation module updates both history information module to set the probe time window and mobility fast adaptation module. It also interacts with A-RTS filter to update RTS window.

VI. Implementation and Evaluation

We implement HA-RRAA on a programmable AP platform and we compare it with popular state of the art RA designs (RRAA, RRAA+, SampleRate, ARF) using both controlled testbeds and uncontrolled field trials. In this section, we present our implementation and evaluation effort.

A. Implementation

The main implementation challenge is to incorporate Atheros driver's *software retries* with the RA algorithms. Upon a transmission failure (ACK is not received) a software

	RRAA	RRAA+	SampleRate	ARF
Static UDP	41.1%	6.7%	83.9%	39.6%
Static TCP	17.1%	41.6%	55.0%	33.6%
Mobility	-	4.8%	8.6%	-
Hidden Terminal	8.4%	21.7%	28.5%	1144.3%
Field Trial	5.8%	63.2%	6.0%	51.9%

Table II
MAXIMUM PERFORMANCE GAINS OF HA-RRAA OVER STATE OF THE ART RA DESIGNS.

retry can re-send the data frame in the next lower rate in an effort to get the packet through. A software retry is a pair $\{$rate, number of retries$\}$. The first two transmissions will be done using the transmission rate returned from RA algorithm (say 54Mbps). If both of them fail, the remaining transmissions will be done using lower transmission rates (say 2 retries at 48Mbps, 2 retries at 36Mbps and the last 4 retries at 24Mbps). In our implementation we consider that a failure at a lower rate, implies a failure at higher rates too. For example if the selected rate R (say 48Mbps) from RRAA fails 2 times and R^{-} (say 36Mbps) fails one time then the failed frames considered in HA-RRAA's loss ratio will be 3. On the other hand a successful transmission for the rate R returned by RA design, is a transmission which does not require any hardware retries.

B. Evaluation

In this section, we compare the above RAs both at controlled static and mobile settings and field trials. All the algorithms are implemented on the AP side and traffic is downlink (from AP to client). We summarize the gains of HA-RRAA over the other algorithms in Table II.

Static Clients We first compare RA designs in five different locations (P1-P5) on a 5GHz interference-free channel. In Figures 4, 5, 6 we present the goodput performance of the five algorithms for UDP, intense TCP (4 flows) and sparse (1 flow) TCP traffic respectively. We observe that HA-RRAA outperforms all the other algorithms in all the locations. For UDP traffic HA-RRAA gives goodput gains up to 41.1% over RRAA, up to 6.7% over RRAA+, up to 83.9% over SampleRate and up to 39.6% over ARF. In the static TCP setting, goodput gains are significant as well and can go up to 55%.

HA-RRAA's history mechanism leads to much lower average losses in the most of our static UDP and TCP settings. HA-RRAA presents up to 29.7% lower average loss than RRAA and up to 22.4% lower average loss than ARF. Although SampleRate considers past performance before sampling to higher rates as we discuss in Section II, it yields higher average losses than HA-RRAA that can go up to 9.1%. For the location P3 of our case study presented in Section IV, HA-RRAA gives significant better performance than RRAA, SampleRate and ARF (Figure 4) by transmitting only 2.2% of the total frames at the lossy 48Mbps rate. Although RRAA+ is proven effective in our case study setting by giving almost the same goodput performance and rate distribution with HA-RRAA, in various other traffic and location settings it turns out to be very conservative and to suffer from stale statistics as we discuss in Section II. For example, for multiple flows TCP setting at location P3 (Figure 5), RRAA+ transmits on average 40% of the frames at rates lower than 36Mbps, while the best goodput rates during our experiments were varying between 36Mbps and 48Mbps.

Mobile Clients In our mobility setting, client is moving between locations P1 and P5 at approximately constant pedestrian speed of 1m/s. The channel selected is interference-free and traffic is UDP. From Figure 7 we observe that probe time window does not have any negative effect when client is moving closer to AP as HA-RRAA performs similar to RRAA and ARF. On the other hand, RRAA+ does not have any mechanism to reset long-term history, which makes it less responsive. As a result HA-RRAA outperforms RRAA+ by 4.7%. To adequately evaluate the fast adaptation mechanism proposed in Section V-C, client should move very fast away from the AP (e.g. vehicular network scenario). We leave the evaluation of this setting as a future work.

Hidden Terminal We next evaluate HA-RRAA in a controlled hidden terminal scenario. In our interference setting, an 802.11a client broadcasting packets at P6, acts as a hidden terminal to an 802.11a client at P2, which receives UDP traffic from the AP. To change the intensity of the hidden terminal, we vary the data source rate of the hidden station. In Figure 8 we present the performance of the implemented algorithms in a modest and an intense hidden terminal scenario. In the modest setting (1Mbps data source rate) HA-RRAA is a clear winner over the other designs, with goodput gains up to 50.1%. In the very intense hidden terminal scenario HA-RRAA is slightly worse than RRAA (3.2%) because probe time window may increase upon collision losses, making HA-RRAA more conservative. Overall, because A-RTS filter, RRAA, HA-RRAA and RRAA+ give better performance than ARF and SampleRate.

Field Trials We also conduct a series of uncontrolled field trials to understand how well the algorithms perform under realistic scenarios, in which various sources of dynamics co-exist in a complex manner. Our field trial uses two static clients at locations P2 and P4 and a third client initially placed at P3 and which we periodically move between locations P1 and P5. We run four sets of experiments and each lasted at least half an hour both at 2.4GHz (channel 1) and 5GHz bands (channel 36). At 2.4GHz band, the channel was heavily loaded as we sniffed five radios at channel 1 (the same channel as our AP), three radios at channel 3, one radio at channel 6, four radios at channel 9, two radios at channel 10 and two radios at channel 11. Under this highly

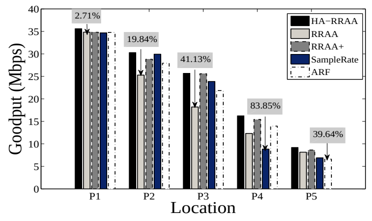

Figure 4. Static 802.11a client at UDP setting.

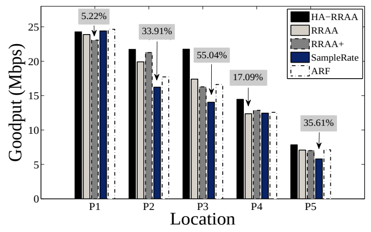

Figure 5. Static 802.11a client at high volume (4 flows) TCP setting.

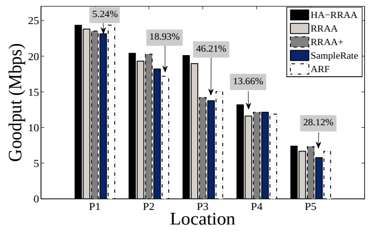

Figure 6. Static 802.11a client at low volume (1 flow) TCP setting.

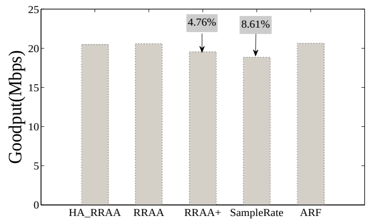

Figure 7. Mobile 802.11a client at UDP setting.

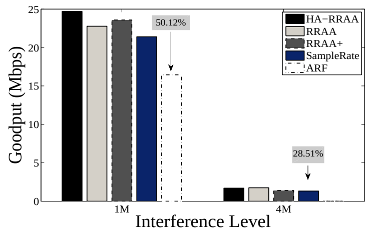

Figure 8. Hidden terminal setting.

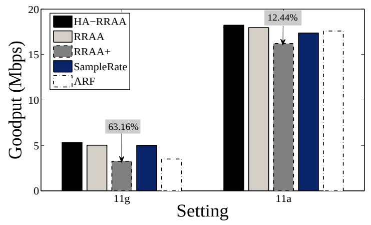

Figure 9. Field trials for 2.4GHz and 5GHz bands.

congested environment HA-RRAA gives 5.8%, 63.2%, 6%, 51.9%, goodput gains over RRAA, RRAA+, SampleRate and ARF respectively, as presented in Figure 9. At the less congested 5GHz band, the performance of the algorithms is significantly better. Still HA-RRAA gives up to 12.4% goodput gains over the other algorithms.

VII. Conclusion

In this paper, we empirically study rate adaptation, using an IEEE 802.11 compliant, programmable AP platform. Our experimental results reveal very dynamic wireless channel, even in indoor static settings. The key insight learned is that, RA algorithms, which cannot eliminate transmissions at consistently lossy rates, while remaining adaptive to fast channel dynamics, can perform much worse than a fixed rate scheme. To this end, we design and implement HA-RRAA, an improved version of RRAA which applies an adaptive loss-proportional and binary exponential growing probe timer, to limit transmissions at low goodput rates. HA-RRAA also incorporates mechanisms to address mobility-induced channel changes and to efficiently handle collision losses. HA-RRAA shows gains up to 63% over RRAA, RRAA+, SampleRate and ARF, in realistic field trials.

References

[1] I. Pefkianakis, Y. Hu, S. H.Y. Wong, H. Yang, S. Lu. MIMO Rate Adaptation in 802.11n Wireless Networks. *ACM MOBICOM'10.*

[2] A. Kamerman and L. Monteban. WaveLAN II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, 1997.

[3] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. *ACM MSWiM'04.*

[4] G. Holland, N. Vaidya and V. Bahl. A Rate-Adaptive MAC Protocol for Multihop Wireless Networks. *ACM MOBICOM'01.*

[5] J. Kim, S. Kim, S. Choi and D. Qiao. CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs. *IEEE INFOCOM'06.*

[6] S. H. Wong, H. Yang, S. Lu and V. Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. *ACM MOBICOM'06.*

[7] J. Bicket. Bit-rate Selection in Wireless Networks. *MIT Master's Thesis*, 2005.

[8] J. Camp and E. Knightly. Modulation Rate Adaptation in Urban and Vehicular Environments: Cross-layer Implementation and Experimental Evaluation. *ACM MOBICOM'08.*

[9] A. Prashanth, A. Kumar, A. Sharma, E. Belding, K. Almeroth and K. Papagiannaki. Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach. *IEEE SECON'08.*

[10] G. Judd , X. Wang and P. Steenkiste. Efficient channel-aware rate adaptation in dynamic environments. *ACM MobiSys'08.*

[11] M. Vutukuru, H. Balakrishnan and K. Jamieson. Cross-Layer Wireless Bit Rate Adaptation. *ACM SIGCOMM'09.*

[12] H. Rahul, F. Edalat, D. Katabi and C. Sodini. Frequency-Aware Rate Adaptation and MAC Protocols. *ACM MOBICOM'09.*

[13] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury. AccuRate: Constellation Based Rate Estimation in Wireless Networks. *USENIX/ACM NSDI'10.*

[14] K. Ramachandran, R. Kokku, H. Zhang, M Gruteser. Symphony: Synchronous Two-Phase Rate and Power Control in 802.11 WLANs. *ACM MobiSys'08.*

[15] W. Kim, M. Khan, K. Truong, S-H. Choi, R. Grant, H. Wright, K. Mandke, R. Daniels, R. Heath and S. Nettles An Experimental Evaluation of Rate Adaptation for Multi-Antenna Systems. *IEEE INFOCOM'09.*