RADAR: Rate-Alert Dynamic RTS/CTS Exchange for Performance Enhancement in Multi-rate Wireless Networks *

Liqiang Zhang and Yu-Jen Cheng
Dept. of Computer & Information Sciences
Indiana University South Bend
South Bend, IN 46615, USA

Xiaobo Zhou
Dept. of Computer Science
University of Colorado at Colorado Springs
Colorado Springs, CO 80918, USA

Abstract

Rate adaptation is a common technique to exploit channel diversity in wireless networks. Despite the many rate adaptation algorithms proposed for 802.11 networks, the ARF (Auto Rate Fallback) remains the most widely adopted scheme in commercial 802.11 products due to its simplicity. However, ARF suffers from some disadvantages. Our research effort revealed the rate avalanche effect that could significantly degrade the network performance of heavily loaded 802.11 networks. In this work, we propose RADAR (Rate-Alert DynAmic Rts/cts exchange) to judiciously exploit dynamic RTS/CTS exchange in multi-rate 802.11 networks. RADAR could effectively suppress the rate avalanche effect while at the same time minimizes the transmission overhead of RTS/CTS exchanges. Being fully compatible with current 802.11 standards, RADAR can be readily implemented in the NIC driver. Through extensive simulations using realistic channel propagation and reception models, we demonstrate that RADAR is a practical and efficient performance enhancement approach for multi-rate 802.11 networks.

1 Introduction

The highly volatile nature of wireless medium poses special challenges to the protocol design for wireless networks. Supporting multiple transmission rates at the physical (PHY) layer that use different modulation and coding schemes is a nature solution to exploit wireless channel diversity. While many current wireless networking standards support multi-rate transmissions, the rate adaptation schemes that guide the rate selection to match the time-varying channel conditions are left undefined in the standards. Recently, a number of rate adaptation schemes have been proposed for 802.11 networks in the literature [3, 4, 5, 6, 7, 8, 10, 13, 14]. These schemes could be roughly divided into two categories: SINR (Signal to Interference plus Noise Ratio)-based [4, 6, 13] and statistics-based [3, 5, 7, 8, 10, 14].

SINR-based approaches are theoretically ideal for rate adaptations, they are hard to implement in practice, however, because of their dependences on the following assumptions: (i) the sender should be able to accurately predict the channel condition (i.e., SINR) at the receiver side, and (ii) the mapping between SINR and the optimal rates are node-independent and known a priori. Unfortunately, these strong assumptions usually do not hold true in today's wireless networks [15]. Consequently, some approximations are often used in these schemes [15]. SINR-based approaches have not been applied in practice so far.

As an alternative to SINR-based approaches, statistics-based schemes estimate link conditions through maintaining statistics about the transmitted data like the achieved throughput [3], consecutive transmission successes/losses [7, 10, 8, 5], and short-term lose ratio [14], etc. For example, as the representative statistics-based approaches, ARF (Auto Rate Fallback) and AARF (Adaptive ARF) adjust data rates by keeping the track of acknowledged/unacknowledged transmissions. Despite some disadvantages, ARF, the first documented rate adaptation scheme, remains the most widely implemented rate adaption scheme in the 802.11 market [8] due to its simplicity.

Aiming at practical solutions, we focus our effort on statistics-based schemes with the objective of enhancing their performance while keeping them compatible with currently-deployed 802.11 WLANs. Our proposed approach is driven by the finding of the rate avalanche effect [15] that could significantly degrade the performance of heavily loaded ARF/AARF-enabled multi-rate 802.11 networks. The rate avalanche effect is a vicious circle that happens when RTS/CTS exchange is not enabled: high collision rates not only lead to retransmissions but also drive nodes to switch to lower data rates; the retransmis-

*This work was supported in part by US National Science Foundation grants CNS-0834230 and CNS-0720524.

sions and the longer channel occupation caused by lower rates will further deteriorate the channel contention, which yields more collisions. One of the major reasons behind the rate avalanche effect is that ARF/AARF lacks the ability to differentiate frame losses caused by collisions from those caused by link errors. Our study revealed that the rate avalanche effect could be effectively ameliorated through turning on the RTS/CTS mechanism. However, as the RTS/CTS exchange itself introduces transmission overhead, the use of the RTS/CTS exchange should be dynamically tuned. To this end, we propose RADAR (Rate-Alert DynAmic Rts/cts exchange) which could effectively suppress the rate avalanche effect while at the same time minimizes the usage of RTS/CTS exchanges.

RADAR detects the rate mismatch (i.e., the rate avalanche effect) through continuously monitoring the transmission rate used for date frames and the RSSI (Received Signal Strength Indicator) measurement of the corresponding ACK frames (called AckRSSI in the rest of the paper). We divide both the rates and AckRSSI values into three ranges. Two online self-calibrated AckRSSI thresholds are used to map the AckRSSI ranges to the rate ranges. A rate mismatch is detected when the average rate used is lower (in terms of range index) than the average AckRSSI measurement. When this happens, the RTS threshold is decreased to enlarge the opportunity of enabling RTS/CTS exchange sequence; otherwise, the RTS threshold is increased gradually. It is worthwhile noting that although RSSI measurements are utilized in RADAR, we assume neither accurate channel measurement nor receiver-sender feedback loop (e.g., receivers measure the channel, and then send the result or rate decision back to senders [1]). We evaluated the performance of RADAR through ns2 simulator [11] where we incorporated new realistic channel propagation and reception models. Experimental results revealed that RADAR could significantly enhance the performance of multi-rate 802.11 networks in all the situations examined.

The rest of the paper is organized as follows. Section II illustrates through examples the rate avalanche effect that could significantly degrade the performance of multi-rate 802.11 networks. In section III, we present the detailed design of RADAR rate-adaptation system. Simulation modeling and performance evaluation results are presented in section IV. We briefly discuss some related work in section V. Finally, section VI concludes the paper.

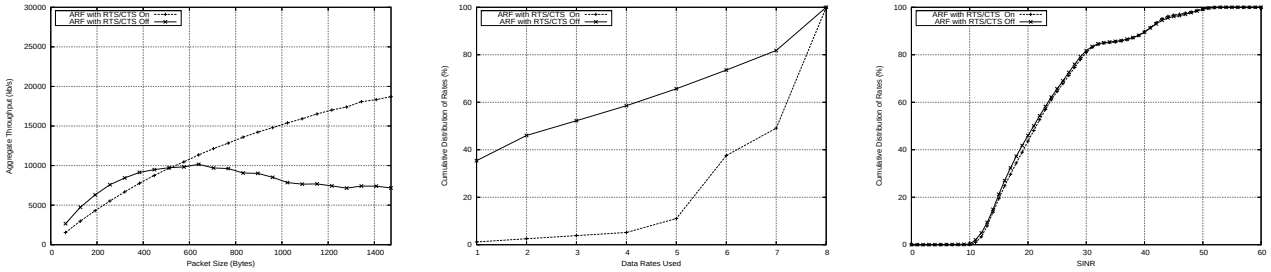
[1]For example, in RBAR [6], a widely-known rate adaptation protocol, receivers estimate the channel conditions based on the RTS frame from senders, make the rate decision and inform the later through a customized CTS frame (thus incompatible to the current 802.11 specifications).

2 The Rate Avalanche Effect

In this section, we illustrate the rate avalanche effect and show how it could significantly degrade the performance of heavily-loaded multi-rate 802.11 networks. Here, by "heavily-loaded networks", we refer to those highly-contending wireless environments where many (e.g., 20 or more) nodes, each of which having intense traffic to transmit, compete to access a common channel. With the great prevalence of 802.11 WLANs and ever-growing multimedia applications, such highly-contending scenarios are not rare today. Flash crowds [9] that often happens in public hotspot WLANs (e.g., in universities, conferences, airports, and coffee shops) and wireless chaos caused by unplanned channel settings in private WLANs often worsen the situations.

To demonstrate the rate avalanche effect, let us start with a simple scenario: fifty 802.11a wireless nodes are randomly deployed in a square area of 80 X 80 $meter^2$, and one AP is installed in the center of the area. There is one UDP-based CBR (constant bit rate) traffic flow sending from each node to the AP. Each flow has the packet arrival rate of 200 packets/second. Nodes are static during the experiment. We first turn RTS/CTS on (i.e., set RTS threshold as 0 byte), and repeat the experiment with different packet sizes from 64 bytes to 1472 bytes with the step size of 64 bytes. Using the exactly same settings, we repeat the experiments with RTS/CTS turned off (i.e., with RTS threshold set as 3000 bytes). Fig. 1(a) compares the aggregate throughput for the cases that RTS/CTS are turned on and the cases that RTS/CTS are turned off (referred as rts-on and rts-off respectively in the rest text). As clearly shown in the figure, while RTS/CTS-off delivers slightly higher throughput than RTS/CTS-on when the packet size is smaller than 512 bytes, it achieves much lower performance than the later when the packet size is higher than 512 bytes.

The transmission overhead of RTS/CTS exchange can easily explain the performance gain of rts-off over rts-on. However, what have caused the performance deterioration of rts-off after the cross-point in Fig. 1(a)? With hidden nodes and SINR differences being eliminated from the possible causes [15], we identified that the major reason is the sharp rate dropping caused by collisions. With 1 to 8 representing the rates from 6 to $54Mbps$, Fig. 1(b) compares the cumulative distribution of the rates used in data transmissions in rts-on and rts-off, clearly showing that a much higher percentage of lower-rates are used in rts-off than in rts-on. The rate dropping in rts-off is due to the lack of loss-differentiation ability of ARF – each unacknowledged transmission, even caused by a collision, is attributed to link errors and then used to direct the rate adaptations. It is interesting to see that RTS/CTS exchange not only leads to less collisions but also helps to reduce the fault reactions to



(a) (b) (c)

Figure 1. Illustrating the rate avalanche effect. (a) compares the aggregate throughput of rts-off and rts-on with the setting of 50 nodes and CBR traffic flows; (b) compares the cumulative distribution of the date rates used in (a) for the case with packet size = 1024 bytes; (c) shows the cumulative distribution of the SINR measured at the receiver for all the frames received with or without errors, indicating that SINR is not the reason for the rate discrepancy shown in (b).

the unacknowledged transmissions.

It is worthwhile noting that the rate avalanche effect does not only happen in ARF-enabled multi-rate networks, instead, it comes with all statistics-based rate adaptation schemes as long as they lack an effective method to differentiate collisions from link errors. More results and discussions about the rate avalanche effect can be found in [15].

3 The RADAR Rate-adaptation System

The results presented in the previous section suggest us to utilize the RTS/CTS exchange wisely in rate adaptations – to enable it only when necessary, so that the rate avalanche effect could be effectively suppressed while at the same time the transmission overhead is minimized. To dynamically enable/disenable RTS/CTS exchange on a per frame basis, we surely could exploit the RTS threshold. However, our study reveals [15] that a pre-defined RTS threshold only leads to sub-optimal network performance. The optimal RTS threshold depends on many factors, such as, the number of competing nodes, the geographic distribution of nodes, and node mobility, etc. All of these factors can vary over time.

Driven by this, besides the rate adaptation loop, RADAR also embraces an RTS threshold adaptation loop. The rate adaptation loop could be some original rate controller that mainly relies on transmission statistics to guide rate switching, for example, ARF or AARF; while the RTS threshold adaptation loop is a delicately designed self-calibrated controller that takes advantage of some extra feedback information about the wireless channel. These two loops are loosely coordinated, in the sense that the RTS threshold adaptation loop does not directly put force on the rate adjustment. Instead, it helps to create benign interactions among nodes that reduces the situations that lead to false-reactions of the rate controller. On the other side, the rate controller does nothing directly on the RTS threshold tuning except feeding rate settings to the later as a part of statistics collection. Figure 2 outlines the structure of the RADAR rate adaptation system.

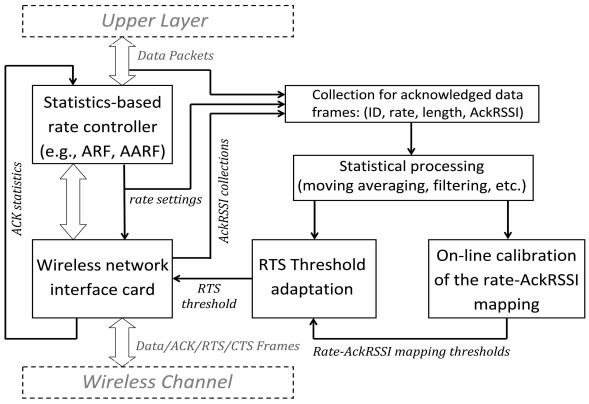


Figure 2. The RADAR rate adaptation system

Next we briefly present the basic principles of the RTS threshold adaptation. Choosing a best RTS threshold at run time that optimally balances the benefit of using RTS/CTS exchange on ameliorating the rate avalanche effect against its transmission overhead is a challenging issue. Firstly, using the (locally) achieved throughput or its variance as the feedback to guide the adaptation is inappropriate, since the throughput depends on not only the RTS threshold setting but also several other factors, such as, the varying channel condition caused by node mobility, the local traffic load of a node, the number of competing nodes, etc. Secondly, the choice of a RTS threshold value is a per node decision, however, its impact on the network performance is global (i.e., network-wide). This makes a model-based approach difficult to achieve. Having these considerations in mind, we choose to use the recent history of data rates and channel condition measurements to guide the RTS adaptation. The rationale is rather simple: if data rates does not match the

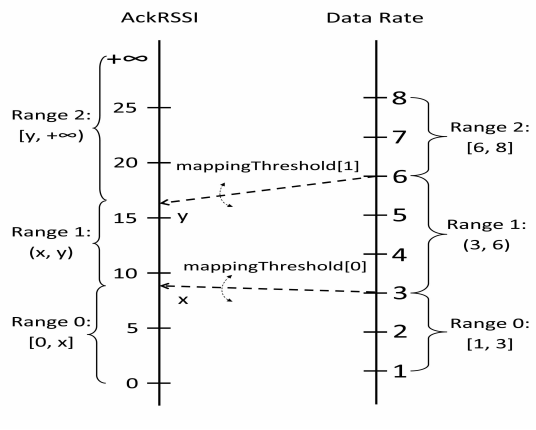


Figure 3. The dynamic mapping between rate and AckRSSI ranges

channel condition well, which indicates the influence of rate avalanche effect, we should lower down the RTS threshold so that more data frames will be rigged with RTS/CTS exchange; otherwise, a higher RTS threshold should be used to reduce the transmission overhead.

However, in real implementation, we face the following difficulties: (1) channel condition measurements available in current wireless cards, for example, RSSI (Received Signal Strength Indicator) values, are only approximations to SINR, and are often noisy and uncalibrated; (2) the mapping between RSSI and optimal rate is hardware-dependent, interference-dependent, fuzzy, and unknown a priori. To address these issues, instead of applying point-to-point mapping, we divide both the rates and RSSI values into three indexed ranges and make use of range-to-range mapping (see Figure 3). Two online self-calibrated RSSI thresholds are used to map the RSSI ranges to the rate ranges. To further simplify the design, we measure the RSSI of ACK frames at senders, instead of doing the measurements for data frames at receivers then sending them back. The later approach needs to modify the format of ACK frame thus incompatible to the 802.11 standard. We present the detailed design of RTS threshold adaptation and the online self-calibration of rate-AckRSSI mapping in the followed text.

3.1 The Adaptation of RTS Threshold

The RTS threshold adaptation is described in Algorithm 1, where we adjust the RTS threshold in an additive increase/multiplicative-decrease (AIMD) manner, similar to the way of TCP avoiding congestion. When rate mismatch is detected (line 20), the RTS threshold is cut half (line 22); otherwise, it is increased gradually (line 25). The AIMD algorithm enables a timely reaction to the rate avalanche effect and a minimized using of RTS/CTS exchange.

As previously discussed, the rate avalanche effect is detected by continuously monitoring the data rates and AckRSSI samples, which are averaged periodically. The sampling window is bounded by a specific number of acknowledged data transmissions and a sampling timer, whichever is reached first. To detect the rate avalanche, we then need to first identify the indexes for the averaged data rates and AckRSSI measurements as shown in Figure 3. If the index of averaged data rate is lower than the averaged AckRSSI measurements, a rate mismatch is identified. To avoid unnecessary continuous decreasing of the RTS threshold caused by avalanche-polluted samples, we will skip a specific number, e.g., 15, of samples each time after the rate avalanche being detected. This gives the rate adaptation loop some time to correct its rate selection before the next examine point.

Algorithm 1 The adaptation of the RTS threshold

```
 1: rtsThresholdAdaptation():
 2:   while(1)
 3:   {
 4:     ackCount = 0;
 5:     reset sampling timer;
 6:     while (ackCount < samplingWindow
 7:             && samplingTimer not expired)
 8:     {
 9:       send/resend a data frame at a rate (w/ or w/o RTS/CTS);
10:       if (data transmission is acknowledged)
11:       {
12:         measure the AckRSSI;
13:         if (dataFrameLength > RTSThreshold)
14:           rateRssiMappingCalibration();
15:         ackCount++;
16:       }
17:     }

18:     avgDataRate ⇐ moving average of the rates for acked data frames;
19:     avgAckRssi ⇐ moving average of the AckRSSIs;

20:     if (dataRateIndex(avgDataRate) < ackRssiIndex(avgAckRssi)
21:     {
22:       RTSThreshold = max(RTSThreshold_MIN, RTSThreshold/2);
23:       filter out the avalanche-polluted samples;
24:     } else
25:       RTSThreshold = min(RTSThreshold_MAX,
26:                          RTSThreshold + RTSThresholdIncStep);
27:     }
28:   }

29: dataRateIndex():
30:   return the index of the range where the rate locates;

31: ackRssiIndex():
32:   return the index of the range where the RSSI value locates;
```

Obviously, the detection of rate mismatch highly depends on the thresholds that map the rate ranges to the RSSI ranges. To address the noisy and fuzzy nature of the rate-RSSI mapping, we need to online self-calibrate the map-

ping thresholds. This is reflected in line 14 of Algorithm 1, where a function named rateRssiMappingCalibration() is repeatedly invoked. We discuss the details of this function in the next subsection.

3.2 The Online Self-calibration of the Rate-AckRSSI Mapping

Let us first get back to Figure 3 which illustrates the idea of range-to-range mapping. As shown in the figure, the three ranges for date rates have fixed boundaries, i.e., $[1, 3]$, $(3, 6)$, and $[6, 8]$, while the AckRSSI ranges, namely, $[0, x]$, (x, y), and $[y, +\infty)$, are all floating. We define the x and the y here as the mapping thresholds, which are assigned some initial values at the beginning and are self-calibrated through the online learning later.

Algorithm 2 The online self-calibration of the Rate-AckRSSI mapping

```
 1: rateRssiMappingCalibration():
 2:   ri = dataRateIndex(dataRate);
 3:   si = ackRssiIndex (AckRSSI);
 4:   indexDifference = ri - si;
 5:   if (indexDifference > 0)
 6:   {
 7:     mappingThresholdTooHighIndicator[ri-1] += indexDifference;
 8:     if (mappingThresholdTooHighIndicator[ri-1] ≥ 3)
 9:     {
10:       if(mappingThresholdTooLowIndicator[ri-1] == 0)
11:         dropMappingThreshold(ri-1, indexDifference);
12:       resetMappingThresholdIndicators(ri-1);
13:     }
14:   }
15:   if (indexDifference < 0)
16:   {
17:     mappingThresholdTooLowIndicator[ri] += indexDifference;
18:     if (mappingThresholdTooLowIndicator[ri] ≥ 3)
19:     {
20:       if(mappingThresholdTooHighIndicator[ri] == 0)
21:         raiseMappingThreshold(ri, indexDifference);
22:       resetMappingThresholdIndicators(ri);
23:     }
24:   }

25: dropMappingThreshold(i, d):
26:   mappingThreshold[i] -= d;
27:   adjust adjacent threshold if necessary to avoid boundary crossing;

28: raiseMappingThreshold(i, d):
29:   mappingThreshold[i] += d;
30:   adjust adjacent threshold if necessary to avoid boundary crossing;

31: resetMappingThresholdIndicators(i):
32:   mappingThresholdTooHighIndicator[i] = 0;
33:   mappingThresholdTooLowIndicator[i] = 0;
```

Algorithm 2 presents the details of the online self-calibration, which has a rather simple rationale behind. Let's explain it using an example. If a data frame is transmitted with a rate 5 which has a range index of 1, while the AckRSSI measurement falls into range 0 according to the mapping thresholds, we take it as a sign indicating the $mappingThreshold[0]$ is too high; it should be lower down so that the rate and the AckRSSI could fall into the same range (i.e., 1); on the other hand, if, for the same date frame, the AckRSSI measurement falls into range 2, then the mappingThreshold[1] should be raised for the same reason. To avoid erroneous reactions as well as to balance the stability and agility, some filtering processing are applied (e.g., lines 7-10 and 17-20 in algorithm 2) before the mapping thresholds are adjusted.

There is, however, an important issue to address: on one side we use (rate, AckRSSI) sample pairs to calibrate mappings thresholds and on the other side we use mapping thresholds to judge if a rate is a match to its AckRSSI; is there an "arguing in a circle" kind of flaw in the process? Well, yes, if the sample pairs are not filtered; no, if we carefully choose the sample pairs to calibrate the mapping thresholds. As shown in Algorithm 1, besides filtering out those avalanche-polluted samples, RADAR only allows those RTS/CTS enabled sample pairs to be used for calibration (line 13). We trust the rate controller to guide the rate to the right track (i.e., providing trustworthy samples to the online calibration loop) when it is not troubled by the rate avalanche effect.

4 Performance Evaluation

In the section we presents the performance evaluation of RADAR. Due to the expensiveness of building a testbed that contains a large number of 802.11 nodes, we adopt a simulation-based study using the ns2 [11]. However, ns2 does not have a concrete PHY implementation that is needed for our investigation, this has driven us to redesign the PHY implementation incorporating some important features. In the next text, we briefly summarize these features and then present numerical results to demonstrate the performance of RADAR.

4.1 Simulation Modeling and Set Up

Our simulation setup targets an 802.11a-based multi-rate network, although the proposed approach applies to 802.11b/g/n networks as well. Two flavors of RADAR are implemented in ns2: one embraces the ARF as the rate controller, which we refer to as RADAR(ARF); the other one embraces AARF as the rate controller, which we refer to as RADAR(AARF). We compare RADAR with original ARF/AARF to demonstrate its advantages. To achieve trustworthy results, we made significant redesign on the PHY implementation of ns2. The original design in ns2 has two major deficiencies: (1) all the three channel propagation models existed in ns2, namely, Friis free-space

model, two-ray-ground model, and shadowing model, are not sufficient to accurately simulate the small scale multi-path fading effects which are critical for modeling mobile networks; (2) the threshold-based frame reception model is over-simplified, far from being able to model the error performance of the 802.11a OFDM-based PHY layer. To this end, we developed a more realistic channel propagation model that combines the log-distance path loss model and the Ricean propagation model, and a FER (frame error rate) based frame reception model that capture the detailed error characteristics of modulation/coding schemes employed by 802.11a PHY. For more details please refer to [15].

Unless otherwise specified, for all the experiments reported in this paper, we have the same setting for the paramters used in Algorithms 1 and 2 as shown in Table 1.

Parameter	Value/Initial Value
samplingWindow	5
samplingTimer	50 ms
RTSThreshold_MIN	0 $byte$
RTSThreshold_MAX	2332 $bytes$
RTSThresholdIncStep	64 $bytes$
mappingThreshold[1]	30
mappingThreshold[0]	0

Table 1. The setting of parameters for Algorithms 1 and 2

4.2 Numerical Results

Figure 4 compares the performance of RADAR with ARF and AARF. Here we use a similar scenario setting as in section 3: fifty 802.11a wireless nodes are randomly deployed in a square area of 80 X 80 $meter^2$, and one AP is installed in the center of the area. In the experiments for Figures 4(a) and 4(b), each node sends a pair of H.263 video streams that are based on the trace file captured for movie Jurassic Park I, while in the experiments for Figures 4(c) and 4(d), each node sends a CBR traffic flow with a packet arrival rate of 200 $packets/second$. Nodes are static in Figures 4(a) and 4(c), however, mobile in Figures 4(b) and 4(d). RWPM mobility model is used with the maximum velocity bounded by 10 $meters/second$. As clearly shown in these figures, RADAR significantly outperforms ARF and AARF. For the setting of H.263 video streams, RADAR achieves about 10% higher aggregate throughput than both ARF and AARF when the later ones have RTS/CTS enabled (i.e., have a RTS threshold value of 0 $byte$), and more than 200% higher throughput than the later ones when they have RTS/CTS disabled (i.e., have a RTS threshold value of 3000 $bytes$). The performance gain of RADAR for CBR traffic flows depends on the packet size. However, clearly RADAR again does a much better job than ARF and AARF, no matter RTS/CTS is turned on or off.

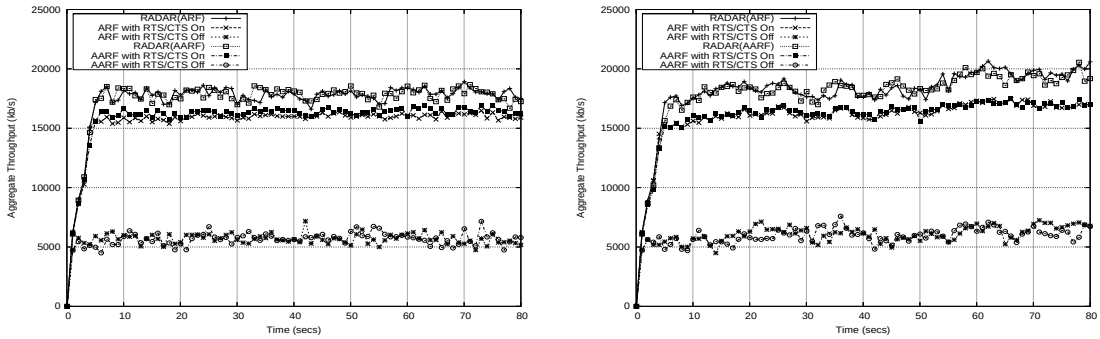
We also investigated how AckRSSI measurement errors and the link asymmetry problem affect the performance of RADAR. Our analysis and experimental results demonstrate RADAR has high robustness against RSSI measurement errors and link asymmetry. We skipped the analysis and results here, however, due to the space limitations.

5 Related Work

ARF [7] was originally designed for Lucent Technologies' WaveLAN-II WLAN devices. It is a typical statistics-based rate adaptation algorithm which incrementally increases or decreases the rate by keeping the track of acknowledged/unacknowledged transmissions. Some deficiencies of ARF have been revealed by previous research efforts: it is unable to adapt effectively to fast-changing channel conditions [6]; On the other hand, if the channel conditions do not change at all, or change very slowly, ARF will try to use a higher rate every 10 consecutive transmission successes or after a timer expiration, which results in increased retransmission attempts and thus harms the throughput [10]. Aiming to enhance the performance of ARF when facing rather stable channel conditions, AARF [7] proposes to adapt the threshold for the number of consecutive successful transmissions after which a higher rate is tried.

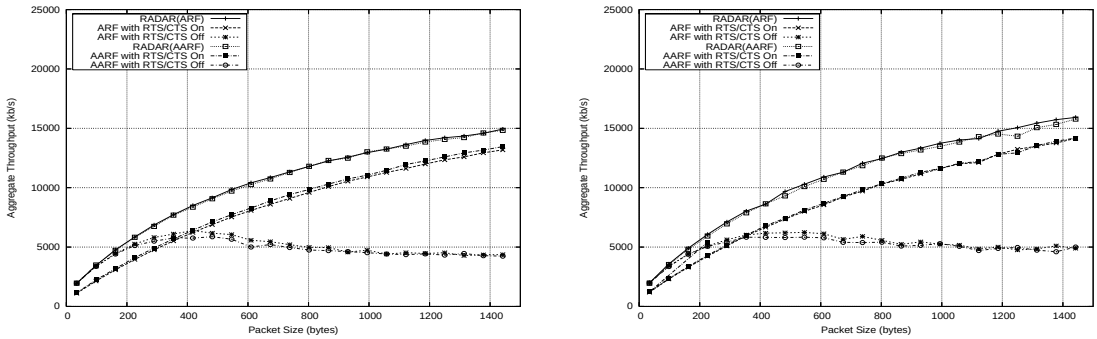
The RTS/CTS exchange was defined as an optional mechanism in DCF (Distributed Coordination Function) access method in IEEE 802.11 standard to deal with the hidden node problem. However, in most infrastructure-based WLANs, it is turned off due to the transmission overhead it introduces. Besides the transmission overhead and its effectiveness to solve hidden node problem, other possible effects of the RTS/CTS exchange are rarely studied. One exception is [2], where Bianchi in the well-known work pointed out that in a heavily-contending WLAN environment, the RTS/CTS exchange might help to reduce collisions and therefore enhance network performance even when no hidden node presents. However, Bianchi's analytical model assumed ideal channel conditions (i.e. error-free links) and only single-rate transmissions were considered.

Recently, several studies have revealed the importance of the loss differentiation to the performance of DCF [1] as well as to the rate adaptation schemes [12, 8]. Pang et al. in [12] and Kim et al. in [8] also revealed that RTS/CTS exchange could be utilized to enhance the loss differentiation ability. However, none of them gave a thorough study on the impact of RTS/CTS exchange. Our previous work [15] investigated the rate avalanche effect and gave a thorough study on the effect of RTS/CTS exchange on the performance of multi-rate 802.11 networks under various network conditions, which has formed the basis of this work.



(a) Static setting (b) Mobile setting

Results for H.263 video trace based VBR streams



(c) Static setting (d) Mobile setting

Results for CBR traffic flows, experiments are repeated for different packet sizes

Figure 4. Performance Comparison: RADAR vs. ARF/AARF

6 Conclusions

In this paper, we proposed a novel rate adaptation approach called RADAR for 802.11 multi-rate networks. Rooted from statistics-based schemes, RADAR exploits channel estimations and dynamic use of RTS/CTS exchange to significantly enhance their performance. It inherits some advantages from both categories of rate adaptation schemes, i.e., SINR-based and statistics-based. Our performance evaluation reveals its effectiveness and robustness.

References

[1] I. Aad, "Quality of Service in Wireless Local Area Networks," *Ph.D. Dissertation*, INRIA, France, Oct. 2002.

[2] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications*, vol. 18, No. 3, pp. 535-547, Mar. 2000.

[3] J. Bicket, "Bit-rate Selection in Wireless Networks," *MIT Master Thesis*, 2005.

[4] C. Chen et al., "Rate-adaptive Framing for Interfered Wireless Networks," *in Proc. of Infocom'07*, Anchorage, Alaska, May 2007.

[5] I. Haratcherev et al. "Hybrid Rate Control for IEEE 802.11," *in Proc. of ACM MOBIWAC'04*, Philadelphia, PA, Sept. 2004.

[6] G. Holland et al., "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," *in Proc. of Mobicom'01*, Rome, Italy, July 2001.

[7] A. Kamerman and L. Monteban, "WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band," *Bell Labs Technical Journal*, pp. 118-133, Summer 1997.

[8] J. Kim et al., "CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs," *in Proc. of Infocom'06*, Barcelona, Spain, April 2006.

[9] A. Jordash et al., "IQU: Practical Queue-based User Association Management for WLANs," *in Proc. of Mobicom'06*, Marina del Rey, CA, September 2006.

[10] M. Lacage et al., "IEEE 802.11 Rate Adaptation: A Practical Approach," *in Proc. of ACM MSWiM'04*, Venezia, Italy, Oct. 2004.

[11] The Network Simulator – ns-2, http://www.isi.edu/nsnam/ns.

[12] Q. Pang et al., "A Rate Adaptation Algorithm for IEEE 802.11 WLANs Based on MAC-Layer Loss Differentiation," *in Proc. of BROADNETS 2005*, Boston, MA, October 2005.

[13] B. Sadeghi et al., "Opportunistic Media Access for Multirate Ad Hoc Networks," *in Proc. of Mobicom'02*, Atlanta, Georiga, Sept. 2002.

[14] S. Wong et al., "Robust Rate Adaptation for 802.11 Wireless Networks," *in Proc. of Mobicom'06*, Los Angeles, CA, Sept. 2006.

[15] L. Zhang, Y. Cheng, and X. Zhou, "Effects of RTS/CTS Exchange on the Performance of Multi-rate 802.11 WLANs," *technical report, available at http://www.cs.iusb.edu/~liqzhang/RTSCTS-Multirate.pdf.*