A Practical Rate Adaptation Algorithm for IEEE 802.11 Networks

by

Kehao Zhang

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 9, 2010

Keywords: IEEE 802.11, Rate Adaptation, FRRA Algorithm

Abstract

IEEE 802.11 standard has evolved from the early basic transmission rate to today's multiple transmission rates. The performance of IEEE 802.11 devices has been improved by exploiting multiple transmission rates. However, to take advantage of such evolution, a mechanism is required to choose the most appropriate rate under different circumstances, namely *rate adaptation*.

Rate adaptation is critical for improving system performance by exploiting multiple transmission rates provided by the current IEEE 802.11 WLANs (Wireless LANs) and adjusting the data rate accordingly under different channel conditions. The key challenge in designing such an algorithm is to select the most suitable data rate under different environments in order to maximize the throughput over wireless links. For a given channel, the poorer the channel quality, the lower should the data rate be. However, in a congestion dominated network, lowering the data rate does not help the situation, instead, it further decreases throughput because lower transmission rate will increase both transmission range and transmission time and therefore introduces more collisions.

Multiple rate adaptation schemes for IEEE 802.11 WLANs have been proposed and studied. The first generation rate adaptation schemes work well in collision free environments. However, in a congestion dominated environment, these schemes perform poorly because they do not differentiate frame losses caused by collision from those frame losses caused by channel degradation and unnecessarily decrease the data rate. The second generation rate adaptation schemes use Request To Send/Clear To Send (RTS/CTS) control frames to differentiate frame losses. However, there exist several problems when using these control frames. One obvious problem is the introduction of overhead which may lower network performance especially when the data frame size is small. The second problem is that, in IEEE 802.11 standard, the RTS is always transmitted at the lowest rate which may waste bandwidth under certain circumstances. The third problem is deciding when to turn on and off RTS/CTS control frames to reduce the overhead.

Besides the problems mentioned above, most current rate adaptation schemes only consider the situation when the channel quality is good or when there are a lot of collision. Little or no action has been taken when the signal strength is low. According to our study, special actions need to be taken when the channel quality is poor. In our proposed algorithms, we have made several adjustments to accommodate this situation.

This dissertation first gives several guidelines on how to design an efficient rate adaptation scheme and then presents two practical rate adaptation algorithms called *Advanced Rate Adaptation (ARA)* and *Fast Recovery Rate Adaptation (FRRA)*. These two algorithms fully satisfy the proposed guidelines and are implemented along with four other representative rate adaptation schemes on a Linux-based testbed. The proposed algorithms and other selected rate adaptation schemes are evaluated extensively in both controlled and public field tests. Experiment results show that ARA and FRRA outperform other rate adaptation schemes in most scenarios.

Acknowledgments

I am grateful to my advisor Dr. Alvin S. Lim for his direction, assistance, and guidance. I could not express enough thanks to him while I was writing this dissertation.

I wish to thank my committee members Dr. Kai H. Chang, Dr. Xiao Qin and Dr. Zhongyang Cheng. Their guidance and advices are very valuable and helpful.

I am also indebted to my colleague Dr. Shaoen Wu who supported me in a number of ways.

My deepest gratitude goes to my parents and my little brother. I believe I have been blessed by the God for granting me such wonderful and supportive family. Especially my father Dr. Shaoqin Zhang, he is my spirit leader and guides me in my journey of life.

Finally, special thanks to my beloved wife Xuejie Chen. It is her love and support that makes this dissertation possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the last decade, IEEE 802.11 [2] Wireless LAN (WLAN) has gained wide popularity for broadband wireless networking. IEEE 802.11 standard has been evolving from the earlier basic transmission rate to today's multiple transmission rates. The performance of IEEE 802.11 devices has been improved by exploiting multiple transmission rates. However, to take advantage of such evolution, a mechanism is required to choose the most appropriate rate under different circumstances, namely *rate adaptation*.

Rate adaptation is a link-layer mechanism critical to the system performance in IEEE 802.11 WLANs (Wireless LANs) and has been studied extensively in recent years. The key idea of this mechanism is to select the most appropriate rate under different circumstances. When the channel condition is degraded, the link may not be able to support current rate, therefore, frame losses occur and a lower data rate may be more desirable. However, when a frame is lost due to collision instead of channel degradation, the data rate should not be decreased for the following two reasons: 1) A lower rate may exacerbate medium congestion because of longer frame transmission time and wider transmission range (more interference); 2) A lower rate is wasteful of bandwidth and unnecessary as channel conditions may well support a higher rate.

Rate adaptation on IEEE 802.11 networks has been extensively studied in the past years and many schemes [10–20] have been proposed. These schemes can be classified into two generations: without loss differentiation and with loss differentiation. For the schemes without loss differentiation, they do not differentiate the cause of frame losses and reduce the data rate when the transmission failure count or failure rate reaches a certain threshold. By doing this, these schemes assume that all frame losses are caused by channel degradation with limited congestion losses [10, 13, 14].

Such an assumption is valid as long as the MAC layer deploys some collision avoidance mechanisms such as the use of Request To Send/Clear To Send (RTS/CTS) control frames to eliminate or minimize the congestion losses. Some first generation schemes [11, 12, 15] rely on these control frames to adapt the rate and minimize the collision losses. However, according to the IEEE 802.11 standard, the use of RTS/CTS control frames is optional and recommended only when the frame size is larger than the *RTS Threshold (2347 bytes)* and thus limit the use of these schemes.

To solve the limitation of the first generation rate adaptation schemes, researchers have proposed the second generation rate adaptation schemes [19, 20] to adjust the data rate accordingly in different situations: channel degradation or collision. Though researchers use different methods to assess the channel conditions and to differentiate the frame losses, most of them believe that the data rate should be decreased only if the frame loss is caused by channel degradation and should remain the same if the frame loss is caused by collision. Some of the second generation rate adaptation schemes [19] use RTS/CTS control frames to differentiate frame losses caused by channel degradation and collision. However, there exist several problems: 1) The introduction of overhead. This problem is obvious since for each transmitted data frame, RTS/CTS exchange occurs before the data frame; 2) In IEEE 802.11 standard, the RTS is always transmitted at the lowest rate and this is not desirable in certain circumstances. For example, when the channel strength is strong, transmitting RTS control frame at the lowest rate is a waste of bandwidth. Furthermore, transmitting at the lowest rate may cause the RTS control frame to collide with other frames due to longer transmission duration and wider interference; 3) The decision concerning when to turn on and when to turn off the RTS/CTS control frame. Therefore, it is clear that RTS/CTS should not always be turned on especially when the data frame size is small. This is one of the challenges that one must face when designing an efficient rate adaptation algorithm.

Besides the ability to differentiate the cause of frame losses, another challenge for designing a proper rate adaptation algorithm is how to adapt to channel variations. Since the wireless signal is not stable and may change even in several microseconds, an efficient rate adaptation scheme should take the opportunity to increase the data rate when the channel strength becomes strong and decrease the data rate when the channel strength becomes weak.

Moreover, when a IEEE 802.11 WLANs adaptor is just turned on, the rate adaptation scheme will select an initial rate for transmission. Normally, the initial rate is selected from the highest rate, the medium rate, and the lowest rate. Most of the existing schemes just use the medium rate or the lowest rate in the supported rate set. However, this decision is independent of the actual channel conditions and therefore may be inappropriate. The initial rate should be chosen carefully in order to fully utilize current channel condition. For example, in a situation where the channel strength is strong, selecting the lowest transmission rate as the initial rate may yield a poor throughput. Therefore, selecting an appropriate initial transmission rate is another challenge one has to face when designing the rate adaptation algorithm.

Another important factor in designing an efficient rate adaptation scheme is the rate adjustment metric. The metric can be classified into two categories, either using a threshold or using statistics from the past. Each method has its advantages and disadvantages. Using a threshold in the metric is simple and easy to implement but may not be accurate enough whereas using statistics from the past is more complex and may not be able to adjust the rate quickly. More details about the rate adaptation metric will be discussed in Section 5.1.

Another challenge in designing a rate adaptation scheme is their compatibility. Many existing rate adaptation schemes [11, 12, 18] require modification of the current IEEE 802.11 standard and therefore are hard to implement and to collect experimental results. These schemes normally only have theoretical meanings.

The contributions of this dissertation are listed below:

- Investigates these challenges and proposes several guidelines for designing an efficient rate adaptation algorithm.

- Implements the proposed rate adaptation algorithms (ARA and FRRA) and other four representative rate adaptation schemes on a Linux-based testbed. Extensively controlled and field experiments have been conducted to compare the proposed algorithms with the selected schemes.

From the experiment results gathered from the implemented testbed, the proposed algorithms outperform other rate adaptation schemes in most scenarios. The good performance of the proposed algorithms come from: 1) The ability to precisely differentiate the frame losses caused by channel degradation and collision; 2) The ability to quickly recover from a frame loss, regardless of whether this frame loss is caused by channel degradation or collision. Detailed discussion on why the proposed algorithms perform better than other rate adaptation schemes can be found in Chapter 5.

This dissertation is organized as follows: It first gives the motivation and objectives of this study in Chapter 2. Then, it presents the background information in Chapter 3 and reviews existing rate adaptation schemes in Chapter 4. Chapter 5 shows in detail how the proposed algorithms are designed and implemented. The evaluation and analysis of the proposed algorithms are explored in Chapter 7. Finally, Chapter 8 concludes this dissertation and Chapter 9 shows what needs to be done in the future.

Chapter 2

Motivation and Objectives

Rate adaptation is critical to the system performance in IEEE 802.11 [2] WLANs (Wireless LANs), however, this link-layer mechanism is left unspecified in the IEEE 802.11 standard. In recent years, researchers have been studying this mechanism and have proposed many schemes. These proposed rate adaptation schemes can be classified into two categories: without loss differentiation and with loss differentiation. The first generation rate adaptation schemes [10–14, 16] do not differentiate the frame losses and assume all the losses are due to channel fading. While this is true in certain cases, it is not applicable in most environments. The second generation rate adaptation schemes [18–20] try to differentiate the frame losses and may yield much higher throughput in real situations.

Though the second generation rate adaptation schemes differentiate the frame losses, some of these schemes [18] are not compatible with the current IEEE 802.11 standard and therefore cannot be implemented on current commercial product. Other second generation rate adaptation schemes [19, 20] are compatible with the IEEE 802.11 standard. However, these schemes can only determine the frame losses caused by channel fading and cannot precisely determine the frame losses caused by collision.

For the frame losses caused by channel fading, the transmission rate should be decreased in order to adapt to the environmental changes. For the frame losses caused by collision, decreasing the data rate may not help the situation but to make it worse since a lower data rate results in longer transmission time and wider broadcast range, which will lead to even more collisions. Therefore, we should *not* decrease the transmission rate if a frame loss is caused by collision. Since different causes of frame losses require different actions, if a rate adaptation scheme cannot precisely determine the cause of these losses, an inappropriate action may be taken and therefore results in throughput degradation. To design an efficient rate adaptation scheme, one has to design a mechanism that can precisely differentiate the frame losses caused by channel degradation and collision.

Besides the ability to determine the different causes of frame losses, an efficient rate adaptation scheme should be able to recover from these losses quickly. However, existing rate adaptation schemes do not have an efficient mechanism to recover from such losses because they cannot precisely know what the current situation is.

In this dissertation, two practical rate adaptation algorithms have been developed that can differentiate frame losses caused by channel degradation or collision. They also can recover from frame losses very quickly.

Chapter 3

Background

This chapter first describes the fundamental access method of IEEE 802.11 [2] MAC. Then it investigates the literature on the characteristics of IEEE 802.11 wireless channels.

3.1 IEEE 802.11 Channel Access

In this section, we first introduce the basic channel access mechanism of IEEE 802.11 Distributed Coordination Function (DCF). Then we explain the RTS/CTS exchange mechanism in IEEE 802.11.

3.1.1 IEEE 802.11 CSMA/CA

The backoff procedure for DCF is shown in Figure 3.1 (Figure 3.1 is adopted from [2], Section 9.2.5.2, Figure 9-6 Backoff procedure). The basic medium access method of the IEEE 802.11 MAC is a DCF known as carrier sense multiple access with collision avoidance (CSMA/CA). The CSMA/CA protocol is designed to reduce the collision probability when multiple stations are accessing the same medium. The highest probability of a collision occurs when the medium becomes idle following a busy medium. The reason is because multiple senders could have been waiting for the medium to become idle again. That is why a random backoff procedure is necessary in order to resolve the medium contention conflicts. The DCF is based on CSMA/CA as illustrated in Figure 3.2.

When a station wants to transmit data, it first senses the medium to determine whether the medium is idle. If the channel is not occupied by another station, the transmission may proceed. The CSMA/CA distributed algorithm forces a gap of a minimum duration (DIFS) to exist between contiguous frames. A transmitting station must ensure that the medium is not busy for the duration

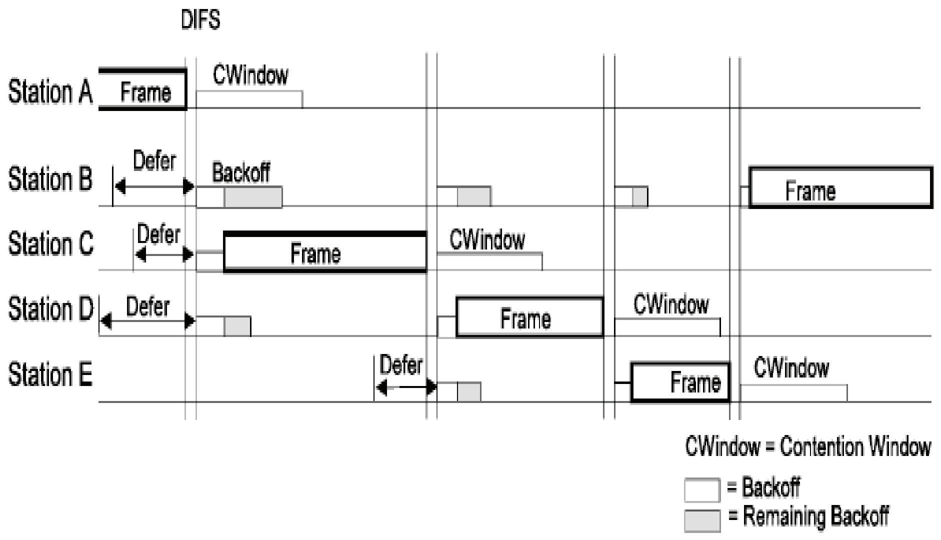

Figure 3.1: Backoff Procedure for IEEE 802.11 DCF

of DIFS. If the medium is occupied during this period, the transmitting station has to defer it's transmission until the end of the current transmission. After deferring, this station shall select a random backoff interval and starts decreasing this interval counter while the medium is idle. A transmission is successful when the sender receives an ACK from the receiver. If no ACK is received in a given time, the sender will schedule a retransmission.

It has to be pointed out that even with random backoff, a frame may still collide with other frames when two or more stations finish the backoff procedure simultaneously. Such collision cannot be resolved due to the nature of the DCF, and the situation may become worse as the number of transmitting stations increases. A frame is corrupted when it collides with other frames or there exists channel errors.

3.1.2 RTS/CTS Exchange in IEEE 802.11

Both the RTS and CTS frames contain a Duration Field that defines the time period that the medium is to be reserved to transmit the actual data frame and the returning ACK frame. Due to the broadcasting nature of RTS and CTS frames. All the stations within the sender range (can hear RTS) and the receiver range (can hear CTS) may learn the medium reservation by reading the

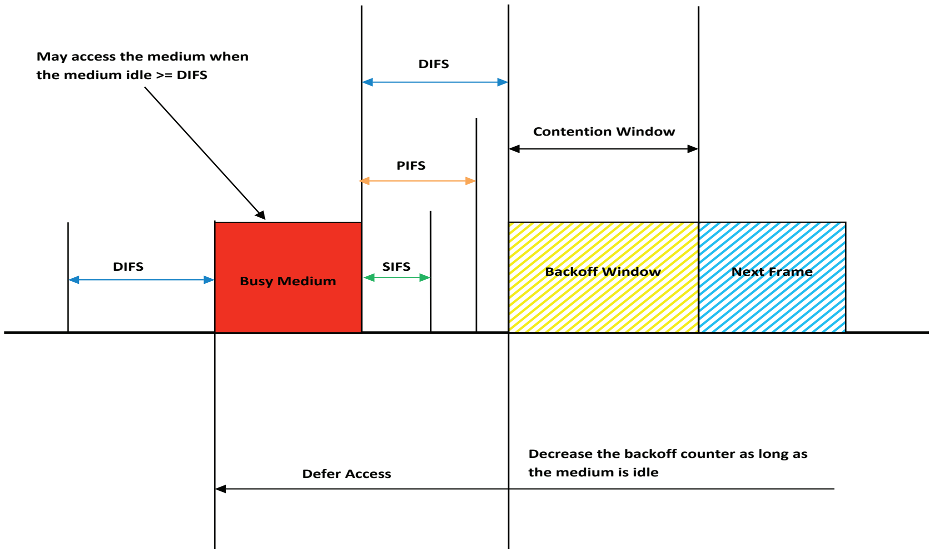
May access the medium when the medium idle >= DIFS
DIFS
PIFS
Contention Window
DIFS
Busy Medium
SIFS
Backoff Window
Next Frame
Defer Access
Decrease the backoff counter as long as the medium is idle

Figure 3.2: Basic Medium Access Protocol of IEEE 802.11 DCF

Duration Field in the RTS or CTS frame. Therefore, it is possible that one sender that is unable to hear from another sender and yet still know about the impending use of the medium to transmit a data frame by listening to a CTS frame. Through the exchange of RTS and CTS frames, the hidden terminal problem is solved.

The RTS/CTS exchange can also perform as a collision inference and a transmission path check. If the return CTS is not heard by the station which sends the RTS frame, there exists a high probability that a collision occurred due to the fact that RTS uses robust transmission rate.

The procedure of RTS/CTS exchange is illustrated in Figure 3.3, (Figure 3.3 is adpoted from [2], Section 9.2.5.4, Figure 9-7 RTS/CTS/data/ACK and NAV setting). From Figure 3.3, it is clear that the wireless channel is reserved for the transmitting station after a successful exchange of RTS/CTS frames.

According to the IEEE 802.11 standard [2], the decision to make use of the RTS transmission is made solely at the sender side. When the size of the current data frame is equal to or larger than the RTS *threshold*, RTS/CTS exchange occurs before the transmission of the current data. However, in most of the typical IEEE 802.11 devices operating as infrastructure based mode, the RTS threshold is set to the largest value (*i.e, 2347 bytes*), which actually disables the exchange

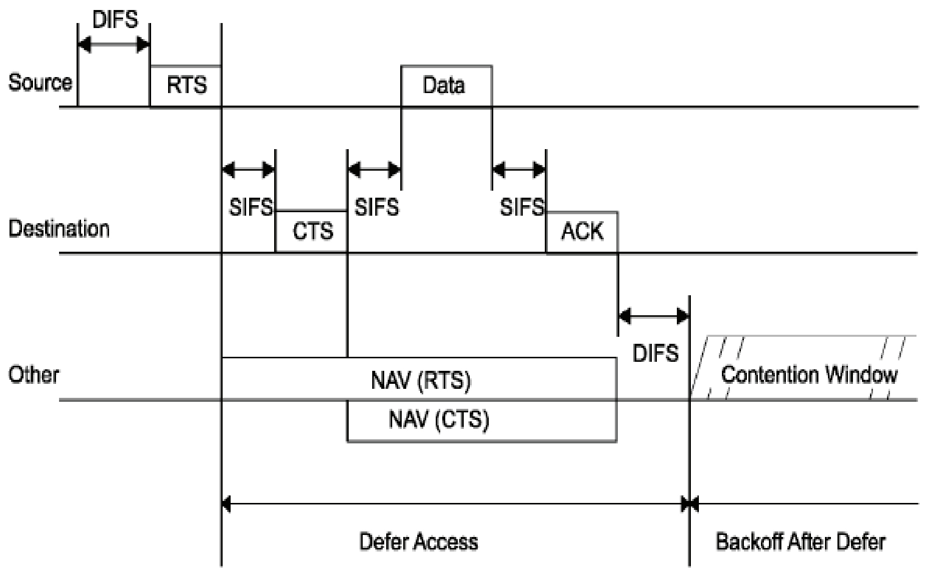

Figure 3.3: RTS/CTS Exchange Procedure in IEEE 802.11

of RTS/CTS control frames in most cases. Although IEEE 802.11 standard defines that the transmission of RTS frame should be triggered by the RTS threshold, using of the RTS frame for other purpose can be found in supplemental standards.

3.2 Investigation of IEEE 802.11 Wireless Channels

This section explains several existing IEEE 802.11 link measurement experiments [5, 6, 8, 9] and the conclusion drawn from their assessments. Most of these assessments were conducted in an outdoor environment, more assessments are needed for indoor environments.

Rodrig *et al.* [6] collected realtime traces from the SIGCOMM 2004 conference. From the observation they provided, around 28% of data frames had to be retransmitted and the retransmission took around 35% of whole transmission time. This observation suggests that *the rate adaptation algorithms implemented in current commercial wireless adaptor drivers is ineffective.*

Aguayo *et al.* [5] used hundreds of node pairs with mounted antennas at the the top of different campus buildings to form a mesh network named Roofnet to perform extensive link-level measurement in an IEEE 802.11b [3] environment. From their experiments, several interesting results are given:

- Regardless the communication distance, most of the links experience frame loss. So the frame loss rate is not closely correlated to the communication distance.

- The fact that most links experience frame loss is probably due to multi-path fading in outdoor environments.

- When loss rate increases due to collision, higher data rates might result in better performance than lower data rates. This indicates that *a good rate adaptation algorithms should not take the loss rate as the sole indicator to adjust the data rate.*

Although Aguayo *et al.* carried out extensive experiments and analyzed in detail the factors impacting IEEE 802.11 networks, their results are limited to IEEE 802.11b network and outdoor environments. Bianchi, Formisano, and Giustiniano [8] pointed out that the physical codings are different between IEEE 802.11b and IEEE 802.11g [4]. Therefore, they may have different characteristics even in the same environment. Bianchi, Oligeri and Potort [8] also found that the link behavior of IEEE 802.11b and IEEE 802.11g is totally different due to the difference in physical coding, therefore, some environments may be suitable for IEEE 802.11b network and not suitable for IEEE 802.11g network.

Chebrolu, Raman, and Sen [7] used an IEEE 802.11 rural network where there is limited interference to monitor several long distance links. Their conclusion somewhat contradicts Aguayo's observation.

- The transmission rate has a great influence upon the frame losses.

- The external interference has a great impact on communication.

The reason why Aguayo [5] and Chebrolu [7] draw two different conclusions is because they were using two different network settings. Even though they were both using IEEE 802.11b outdoor networks, Aguayo was using a collision dominated network whereas Chebrolu was using a rural network.

Souryal *et al.* [9] conducted the IEEE 802.11 link experiment in an indoor environment. They claim that the SNR is a good indication of link robustness. However, they did not specify which IEEE 802.11 standard they were using; therefore the results were not applicable to some IEEE 802.11 standards. One conclusion upon their assessments is that SNR can be a good indicator for rate adaptation algorithms.

The above investigation of the literature for IEEE 802.11 wireless channels shows that not only is rate adaptation schemes implemented in current commercial wireless adaptor drivers in-effective, but also inappropriate indicator (loss rate) has been used for designing rate adaptation algorithms.

Chapter 4
Related Work

During the last decade, IEEE 802.11 [2] Wireless LAN (WLAN) has become the dominant technology for both indoor and outdoor broadband wireless networking. This chapter explains the most typical and latest relevant rate adaptation schemes, it is divided into two sections, namely First Generation and Second Generation. The last section summarizes these rate adaptation schemes.

4.1 First Generation: Rate Adaptation without Loss Differentiation

A frame loss in IEEE 802.11 networks is generally caused by channel/signal fading or collision. First generation rate adaptation schemes do not differentiate the cause of these losses. In IEEE 802.11 networks, a station can use the RTS/CTS control frames to reduce the collision for data transmission. RTS/CTS frames are very small in size. The sender starts the transmission by sending a RTS control frame to the receiver, and the receiver will response with a CTS frame if it is not busy. All other stations that hear the exchange of RTS/CTS control frames will remain silent until the completion of the whole transmission transaction. Therefore, the use of RTS/CTS control frames can minimize collisions. Since the first generation rate adaptation algorithms are intended for networks using RTS/CTS, their lack of loss differentiation is reasonable because most of the data frame losses are due to channel fading.

Auto Rate Fallback (**ARF**) by Kamerman and Monteban [10] is the first rate adaptation algorithm proposed for IEEE 802.11 based wireless networks. It is originally designed for a Lucent Technology wireless LAN product, the WaveLan-II. This algorithm is simple and intuitive. It starts the transmission at the highest rate, when the ACK is missed following a successfully transmitted packet, the first retry is transmitted at the same rate. If the ACK is missed again, the transmission rate is decreased and a timer is started. When either the timer expires or the successively received ACKs reaches a threshold N (N = 10), the transmission rate will increase. A new transmission (probe packet) will be sent at this higher rate. If the new transmission fails, the system will immediately fall back to the lower rate. ARF suffers from two problems. First, it cannot perform efficiently in an environment where the signal strength changes quickly. It either waits for the timer to expire or the successful transmission reaches the threshold before the rate can be upgraded. Both of these conditions take time. Second, if the channel condition does not change at all, ARF still keeps upgrading the rate when the success transmission count reaches threshold, which will lead to a periodical failures and rate oscillations.

Receiver Based AutoRate (**RBAR**) [11] by Holland, Vaidya, and Bahl is the first rate adaptation that takes advantage of the control frames RTS/CTS transmitted at the basic rate. RBAR requires incompatible changes to the IEEE 802.11 standard in two aspects: 1) The original standard transmission time in the header of RTS/CTS is changed to packet size and rate; 2) A new message RSH is introduced to finalize the tentative reservation information in CTS. The RBAR algorithm mandate the use of RTS/CTS mechanism. A pair of RTS/CTS control frame exchange appears before the start of each data frame transmission. When the receiver receives the RTS control frame, it calculates the transmission rate for the upcoming data frame based on the Signal to Noise Ratio (SNR) of the received RTS frame and a set of SNR thresholds. The rate is then sent back to the source in the CTS frame. This algorithm suffers from several major problems: 1) The modification of the RTS/CTS control frame is not compatible with current IEEE 802.11 standard and therefore cannot be deployed in current IEEE 802.11 networks; 2) The RTS/CTS frame exchange needs to appear before the transmission of each data frame even if there are no hidden terminals exist and this is a great waste of the bandwidth; 3) It assumes that the SNR of a transmitted packet is available at the receiver, which is not always true.

Sadeghi *et al.* proposed a rate adaptation scheme called *Opportunistic Auto Rate* (**OAR**) [12]. This scheme is based on the observation that channel coherence times are typically several packets transmission times. The goal of this scheme is to achieve temporal fairness instead of throughput fairness. A throughput gain can be obtained when the channel can support a higher transmission rate instead of the basic transmission rate. The key idea of OAR is to use fake fragmentation. In the IEEE 802.11 standard, the fragmentation mechanism provides a simple and practical way to hold the channel for multiple packets. In OAR, it first uses RBAR rate adaptation scheme to get the available data rate through the exchange of RTS/CTS control frames. Then if the data rate is above the base rate, the *more fragment* flag in *frame control* Field of MAC header is set until $\lfloor \frac{available\_rate}{base\_rate} \rfloor$ packets have been transmitted. For example, suppose current base rate is 2 Mbps, and the channel can support up to 11 Mbps. OAR will try to send $\lfloor \frac{11}{2} \rfloor = 5$ packets compared to the original 1 packet at base rate.

Pavon and Choi proposed the *Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement*(**LA-RSS**) [13]. This rate adaptation scheme is based upon two facts: 1) The frame error probability depends on the received frame length and its transmission rate; 2) The transmitting station can estimate channel behavior by keeping track of the Received Signal Strength (RSS) measured from the frames sent by the Access Point (AP). Upon receiving any frame, the station will update the averaged RSS_avg using the RSS measured from the received frame. In this algorithm, a threshold $Th[i, j]$ is defined as a two dimensional array where i is the transmission rate for IEEE 802.11b [3] with a maximum number of 4 corresponding to the transmission rate from 1 Mbps to 11 Mbps and j is the category of frame length. This algorithm classifies the frame length into three types: 0-100 bytes, 100-1000 bytes and 1000-2400 bytes. Therefore, the threshold $Th[i, j]$ holds 12 values and each value is used to indicate the minimum RSS value required for a particular transmission. The initial value for each threshold is set to 0 and will be updated dynamically when the station starts to operate. For example, $Th[2, 2]$ is the estimated minimum RSS for a frame whose length is between 100-1000 bytes and transmits at 2 Mbps. Two algorithms are defined in order to update the threshold $Th[i, j]$:

$Th[i, j] = (1 - \alpha) \times Th[i, j] + \alpha \times RSS,$

$RSS\_avg = (1 - \beta) \times RSS\_avg + \beta \times RSS.$

Where RSS is the received signal strength measured from the latest frame received. The values of α and β are between 0-1. Besides the problem it does not differentiate the cause of frame losses, this scheme is designed for IEEE 802.11b. For IEEE 802.11g, there are 12 different transmission rate. Therefore, the threshold $Th[i, j]$ will hold 36 values, which greatly increase the algorithm complexity and inaccuracy.

Adaptive Auto Rate Fallback (**AARF**) by Lacage, Manshaei, and Turletti [14] is a rate adaptation scheme based upon ARF. It reduces the ARF oscillation problem by using a changeable threshold N for the successful transmissions required to increase the data rate. AARF increases the data rate after N consecutive successful transmissions and decrease the rate after two consecutive package failure. If the first transmission fails after the rate is just increased, AARF falls back to the lower rate and also doubles the threshold required to increase the rate to $2N$ (with a maximum of 50).

Rate adaptation scheme **SampleRate** [16] by Bicket is based on transmission statistics over cycles. It starts transmission at the highest rate. Every tenth data packet, SampleRate picks a random rate which may do better than the current one to send the data packet. If the selected rate provides smaller transmission time, it will switch to this rate. SampleRate reduces the number of rates it must sample by eliminating those rates that cannot provide better performance than the current one being used. This scheme stops sampling when it encounters four consecutive losses. The process of this scheme is explained in the following example: In IEEE 802.11b, for a packet of 1500 bytes, the transmission time are about 12995 μs, 6834 μs, 2976 μs, 1873 μs for four data rates 1 Mbps, 2 Mbps, 5.5 Mbps and 11 Mbps, respectively (from Figure 5-1 in [16]). Suppose current data frames are transmitted at 11 Mbps, after nine transmission, the average transmission time is 3761 μs, which is bigger than the lossless transmission time of 5.5 Mbps. Then the tenth package will be transmitted at 5.5 Mbps assuming that this rate might yield better performance. Note that 1 Mbps and 2 Mbps are not selected because their lossless transmission time for the packet is higher than the average transmission time at the current rate. SampleRate aims to achieve the best average throughput in the long run.

4.2 Second Generation: Rate Adaptation with Loss Differentiation

The first generation rate adaptation schemes are effective in environments with limited collision losses. They are working either in an environment without collision or in a mixed environment exchanging RTS/CTS control frame before the transmission of each data frame to reduce the collision. However, in reality, most data frames are transmitted in a mixed environment. The exchange of RTS/CTS control frame before the transmission of each data frame introduces a lot of overhead which dramatically reduce the throughput. Even worse, since the first generation rate adaptation algorithms do not differentiate frame losses, they treat all the losses as channel fading which may not be appropriate in certain situations. To respond accurately to a frame loss, recent rate adaptation schemes use several ways to differentiate the cause of frame losses and thus respond accordingly to these losses. In the following section, these schemes are explained.

Pang, Leung and Liew proposed a rate adaptation scheme with loss differentiation ability called Loss Differentiating-ARF (**LD-ARF**) [18] for IEEE 802.11 WLANs by combining ARF with a loss-differentiating MAC [17] they developed. The loss differentiation is performed at the receiver side in LD-ARF. This scheme assumes there are no hidden terminals in a WLAN and all stations can hear each other. The authors argue that the frame header in IEEE 802.11 is small and may not be corrupted by channel fading. If the frame header is corrupted, then it must be caused by collision because if two stations transmit in the same time slot, the whole frame is corrupted including the frame header. Therefore, if a frame header can be decoded by the receiver while the payload cannot, then the cause of the frame loss is due to channel fading. Otherwise, the frame loss is due to collision. If a frame loss is diagnosed as due to channel degradation, a negative ACK *NACK* is sent back to the sender to reduce its rate with the assumption that the source address is still available in the decoded frame header. Besides the loss differentiation, all other operations are the same as in ARF. This scheme suffers from several problems. First, the author assume that the frame header in IEEE 802.11 can only be corrupted by collision, which is not always true. Second, it is possible that the source address is corrupted which will cause the *NACK* unable to send back to the source. Thus, the scheme will not work in this situation. Third, modification of the IEEE 802.11 standard makes the scheme impractical.

Scheme *Collision-Aware Rate Adaptation* (**CARA**) [19] by Kim *et al.* believes that the effectiveness of a rate adaptation algorithm depends on how fast it can respond to the change of channel condition and also depends on how the collision can be detected and processed. CARA assumes that the transmission error probability of a RTS control frame is negligible because of its small size and robust transmission rate. Therefore, all the failure transmission of RTS control frame are due to collision. In this algorithm, it mandate the use of RTS/CTS control frames in case of a frame loss. A data frame is first transmitted without the support of RTS/CTS control frames, if the transmission fails, the exchange of RTS/CTS will be initiated for the next retransmission attempt. If RTS control frame failed, then CARA will keep sending RTS until it successfully receives the CTS. However, if the RTS/CTS exchange is successful but the retransmission fails, the rate will be decreased because the loss must be caused by channel degradation. The following data frames are transmitted at a lower rate without the exchange of RTS/CTS until a data frame fails again. Other than turning on RTS/CTS control frames, CARA adjusts the data rate in the same way as ARF.

The *Robust Rate Adaptation Algorithm* scheme (**RRAA**) by Wong *et al.* [20] is a statistical rate adaptation scheme with loss differentiation. It consists of three modules: Loss Estimation, Rate Change and Adaptive RTS Filter. The Loss Estimation module is used to assess the channel condition by using a time window (5 - 40 frames) to keep track of the frame loss ratio. The Rate Change module decides whether to change or keep the current rate based on the estimated loss ratio. The Adaptive RTS Filter module is used to selectively turn on or turn off RTS/CTS exchange to reduce the collision losses. In RRAA, each rate is associated with three parameters, namely the estimated window size, the Maximum Tolerable Loss threshold (P_{MTL}), and the Opportunistic Rate Increase threshold (P_{ORI}). RRAA starts transmission at the highest rate, whenever a rate is chosen, it is used to transmit the next estimated window size frames. The loss ratio P is calculated based on how many frames have been lost within the window size. When the window finishes, a new rate is chosen based on this loss ratio. If the loss ratio $P > P_{MTL}$, the rate is decreased.

If $P < P_{ORI}$, the rate is increased. If $P_{ORI} <= P <= P_{MTL}$, the rate remains unchanged and the estimated window slides forward to calculate the loss ratio continuously for the current rate. RRAA uses a small trick to adapt the rate change more quickly by calculating the best (worst) possible loss ratio. The best (worst) possible loss ratio is calculated by assuming all the following frames in the estimated window are transmitted successful (failure). If the best possible loss ratio is already larger than P_{MTL}, the rate is decreased immediately and a new estimated window starts. Similarly, the rate is immediately increased if the worst possible loss ration is already smaller than P_{ORI}. To solve the hidden terminal problems, RRAA uses a mechanism called adaptive RTS filter. The key idea of this mechanism is using a RTS window size (RTS_{wnd}), all data frames within this RTS_{wnd} are transmitted with RTS turned on. The RTS_{wnd} is initially set to 0. If the last frame is lost without RTS turned on, RTS_{wnd} increases by one because last frame loss may be caused by collision. However, if the last frame fails after the RTS is turned on or the last frame succeeds without RTS turned on, RTS_{wnd} is halved because last frame did not experience collision. if last frame succeeds with RTS turned on, RTS_{wnd} remain unchanged. However, RRAA is not efficient in some situations. It does not change data rate even if the frame loss is detected due to channel degradation because it does not adjust its rate until the end of each estimated window.

4.3 Summary of Rate Adaptation Algorithms

The above rate adaptation schemes can be grouped under different criteria: whether it differentiate the frame losses, where the data rate is adjusted, and what is the indicator of channel conditions. Table 4.1 summarizes these criteria.

Schemes	**Loss Differentiation**	**Based Location**	**Condition Indicator**
ARF	No	Sender Based	Loss ratio
RBAR	No	Receiver Based	SNR
OAR	No	Receiver Based	SNR
LA-RSS	No	Sender Based	RSS
AARF	No	Sender Based	Loss ratio
SampleRate	No	Sender Based	Loss ratio
LD-ARF	Yes	Receiver Based	Loss ratio
CARA	Yes	Sender Based	Loss ratio
RRAA	Yes	Sender Based	Loss ratio

Table 4.1: Summary of Rate Adaptation Schemes

Chapter 5

Design

This chapter first gives several guidelines in designing an efficient rate adaptation scheme for IEEE 802.11 [2] networks. Then it analyzes the problems in current rate adaptation schemes. Finally, the proposed algorithms are explained in detail.

5.1 Design Guidelines

This section proposes several design guidelines for designing an efficient rate adaptation scheme.

5.1.1 Ability to Differentiate Frame Losses

In the IEEE 802.11 standard, the RTS/CTS control frame exchange is disabled by default due to the overhead. Without the help of these control frames, first generation rate adaptation schemes treat all the frame losses as from channel fading and decreases the data rate whenever the frame failure rate reaches certain threshold. Although this method works when there is no collision, it will fail in a complicated environment where both channel degradation and collision occur. This failure is caused by the inability to differentiate frame losses. If a frame loss is caused by collision, decreasing the data rate will not help solve the problem but will make it worse. The reason is because a lower transmission rate has longer transmission time and wider broadcast range, which will lead to even more collisions in this situation. Therefore, an efficient rate adaptation scheme should be able to differentiate frame losses and respond according to these different causes.

5.1.2 Fast Response to the Variation of Channels

An efficient rate adaptation scheme should be able to adapt to environment changes. Otherwise, the scheme may lose the opportunity to transmit at a higher data rate or keep sending the data at a high data rate when successful delivery is not possible. An efficient rate adaptation scheme should be able to grab the opportunity by increasing the data rate when the signal strength is strong and decreasing the data rate quickly when the signal strength drops dramatically.

5.1.3 A Suitable Initial Rate

In a multi-rate wireless network, when the driver for a mobile station is initialized and attached to a node, it will set the initial transmission rate. Basically, the initial rate is set with three choices: the highest rate, the medium rate and the lowest rate. Most of the current rate adaptation schemes use the medium rate as their initial rate, which is 5.5 Mbps for IEEE 802.11b and 36 Mbps for IEEE 802.11g. This makes sense since the current environment condition is unknown. However, it is not efficient when the channel condition is extremely good or poor. An efficient rate adaptation scheme should take these extreme cases into consideration and set an appropriate rate as the initial rate.

5.1.4 Ability to Accommodate Poor Channel Conditions

When the channel condition is poor, a higher data rate will tend to result in more corrupted frames. Most people believe that using a low data rate may get better results. Therefore, when the signal strength is very low, most existing rate adaptation schemes tends to decrease the data rate to the minimum rate (1 Mbps). However, using the minimum rate does not necessarily gives a better performance than the other rates. Figure 5.1 shows the average TCP throughput in a poor channel and collision free environment by using different fixed rates. From Figure 5.1, we can see that the average throughput increases when we decrease the data rate from the maximum transmission rate of 54 Mbps. The throughput reaches its peak when the data rate is 12 Mbps, after which, the average throughput starts decreasing. We know that the throughput is related to the transmitted frames and the delivery ratio. From Figure 5.1, we can see that at the minimum rate, the delivery ratio is close to 1, which means almost all of the frames have been transmitted successfully. However, its low data rate transmits fewer frames in a given amount of time. On the contrary, at 12 Mbps, even though more than half of the frames have been corrupted, it still gives a much better throughput since it can transmit more frames compared to the minimum rate. Figure 5.1 shows that selecting the lowest data rate in a poor channel environment may not be appropriate.

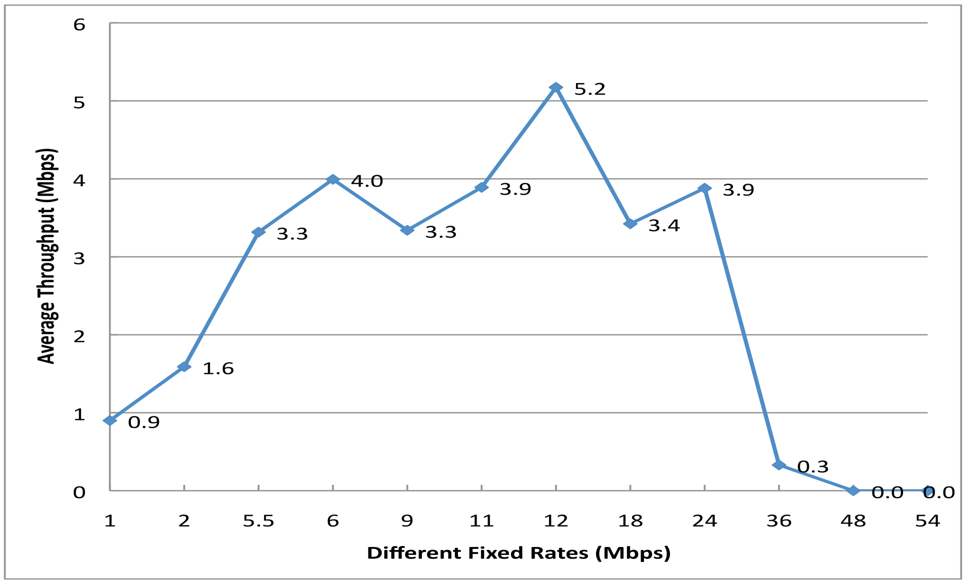

Figure 5.1: TCP Throughput by Using Fixed Rates in a Poor Channel Quality Environment

5.1.5 Ability to Adapt Different Protocols

Currently, there are two different transport protocols that are commonly used, namely Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP provides reliable and ordered delivery of a stream of bytes between stations. Therefore, TCP is mainly used for information and data retrieval applications, for example, web browser, email and file transfer. On the other hand, UDP uses a simple transmission model without the hand-shaking process used in TCP, and it does not guarantee reliability and data integrity. It is mainly used for real-time applications which are normally time-sensitive. In such systems, dropping packets is preferable to waiting for delayed packets.

TCP uses a network congestion control algorithm in order to achieve congestion avoidance. For each connection, TCP maintains a congestion window, limiting the total number of unacknowledged packets between two stations. It uses a mechanism called slow start to manage the congestion window. The window is initialized as one maximum segment size (MSS) which means it can accept one packet at the beginning of a connection. Then the windows starts increasing by 1 MSS as each packet is acknowledged. This actually makes the congestion window to increase exponentially. When the congestion window exceeds certain threshold, the algorithm enters a new state called congestion avoidance. In this state, as long as non-duplicate ACKs are received, the congestion window is additively increased by one MSS every round trip time. However, when a packet is lost, the algorithm will reduce congestion window to 1 MSS, and reset to the slow start state (TCP Tahoe). The TCP's slow start process is shown in Figure 5.2.

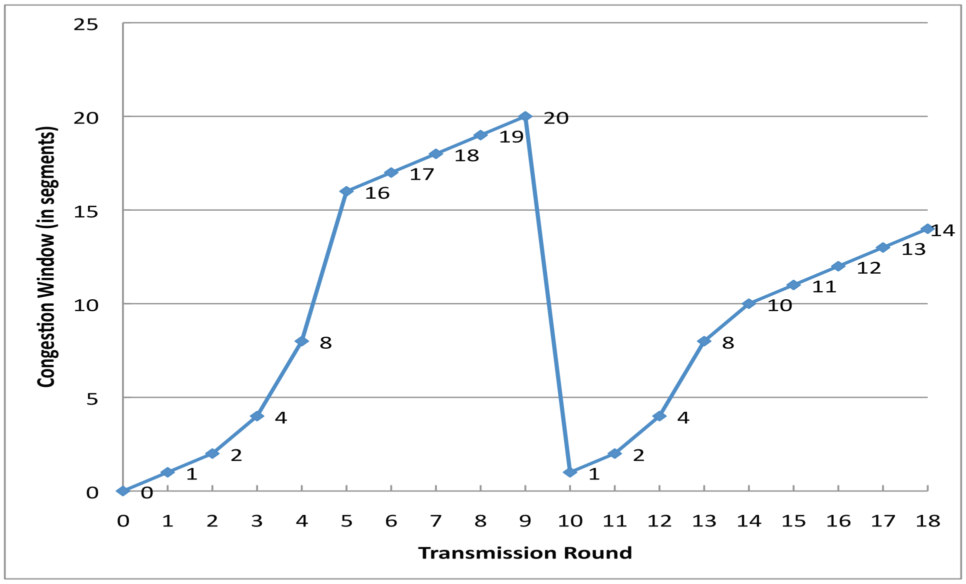

Figure 5.2: TCP Tahoe's Slow Start Process

Figure 5.2 shows that even one packet failure may cause the TCP congestion window to decrease dramatically and the threshold be halved. Therefore, it is very important to keep a low loss ratio in order to get better performance in a TCP traffic. This is especially important in a collision dominated network, where there are a lot of packets will be corrupted due to collision. Therefore, the use of RTS/CTS control frames is necessary because the exchange of RTS/CTS frame reserves the bandwidth and decreases the probability that a packet will collide with other packets.

Contrary to TCP, UDP does not have any congestion control mechanism. If the receiver cannot process the data, it will simply drop those packets. Therefore, to get a better performance in a UDP traffic, the sender should not introduce too much overhead. However, in a collision dominated network, the use of RTS/CTS contro frame is still necessary in order to avoid too much collision.

An efficient rate adaptation scheme should consider the different characteristics of TCP and UDP traffics and should react differently depending on the current traffic type.

5.1.6 A Good Metric to Adjust Data Rate

Some of the current rate adaptation schemes adjust the data rate by monitoring the consecutive success count (with threshold normally set to 10) and the consecutive failure count (with threshold normally set to 2). While this method is simple, it is not accurate. According to *RRAA* [20], the probability of successfully transmitting a data packet following ten consecutive success is only 28.5%. And the probability of a failure in a data transmission after two consecutive failure is only 36.8%. These statistics show that the consecutive success or failure count should not be used as the metric to adjust the data rate.

Besides monitoring the success and failure count, other schemes try to use the signal to noise ratio (SNR) as the indicator to adjust the rate. However, according to SampleRate [16] and RRAA [20], SNR is *NOT* a good indicator of the channel condition and therefore should not be used in the metric.

5.1.7 Compatible with Current Commercial Product

A practical rate adaptation scheme should be compatible with current commercial product, which means we should not make changes to the IEEE 802.11 standard. Several existing rata adaptation schemes [11, 12, 18] require modification of the IEEE 802.11 standard and therefore are hard to implement on the current commercial wireless network adaptors.

The above guidelines give us some basic ideas on how to design an efficient and practical rate adaptation scheme. The next section will explain the proposed algorithms as well as how they satisfy these guidelines.

5.2 RTS Control Frame

RTS control frame has several characteristics which make it suitable for use as a probe frame. First, it has a very small size; according to the IEEE 802.11 standard, it is only 20 bytes in length. Compared to a normal data frame, which is usually larger than 1000 bytes, a smaller probe frame has a smaller probability of colliding with other frames due to the shorter transmission time. Second, RTS/CTS control frames have the ability to reserve the bandwidth. Therefore, a data frame following a successful exchange of RTS/CTS rarely collide with other frames. Furthermore, a failure frame after such exchange must be caused by channel degradation since the bandwidth has already been reserved. Third, RTS transmission rate is adjustable in our testbed. Even though transmitting the RTS at rates other than the lowest does not strictly follows the IEEE 802.11 standard, it does not modify the content of a RTS frame and therefore is still compatible with current commercial products.

To demonstrate that the use of RTS as the probe frame is helpful in a collision dominated network, we use several equations to calculate the probability that a RTS frame will not collide with other frames.

Suppose in an IEEE 802.11 network, there are N stations that has a frame available to transmit at all time. All the data frames are of equal length and the transmission of one frame occupies T

time slots. Let τ denote the probability that a station transmits in a randomly chosen slot time (τ can be obtained using Bianchi's work [21]). Now, suppose one of these stations transmit a frame.

Let q represent the probability that no other stations will transmit in any generic slot occupied by the frame currently being transmitted. q can be calculated as follows:

$$q = (1-\tau)^{N-1} \tag{5.1}$$

Since the entire frame must not collide with other frames, all the T slot time occupied by the transmitting frame must not overlap with any transmissions from other stations. Therefore, the probability P that the entire frame does not collide with other frames can be expressed as:

$$P = q^N = (1-\tau)^{T(N-1)} \tag{5.2}$$

Suppose the transmission of one RTS frame takes R time slots and the transmission of one data frame takes D time slots. Through the use of RTS, the probability for a data frame not to collide with other frames is increased from the original $(1-\tau)^{D(N-1)}$ to $(1-\tau)^{R(N-1)}$.

Next, let's see some examples. Table 5.1 lists some important parameters in IEEE 802.11g (Suppose no other IEEE 802.11b stations connect to the AP).

Parameters	**Value**
Slot Time	9 μs
SIFS	10 μs
DIFS	2 $\times$ Slot Time + SIFS = 28 μs
Physical Layer Overhead	20 μs
Signal Extension	6 μs
CW_{min}	15 Slots
CW_{max}	1023 Slots
RTS Length	20 Bytes
CTS Length	14 Bytes
ACK Frame Size	14 Bytes
MAC Header Length	36 Bytes

Table 5.1: Summary of IEEE 802.11g Parameters

Considering a WLAN where there are five stations and the MAC layer data payload is 1500 Bytes. With the MAC layer header, the frame length is 1536 Bytes. The time needed to transmit this data frame is: $Physical\ Layer\ Overhead + \frac{1536\times 8}{Transmission\ Rate}$. Therefore, it will occupy 230, 60 and 28 slots at transmission rate 6 Mbps, 24 Mbps and 54 Mbps, respectively. Similarly, transmitting a RTS frame will occupy 6, 3 and 3 slots respectively at these rates. Since IEEE 802.11 uses binary exponential backoff mechanism, the probability that a station will transmit a frame in a randomly chosen slot (τ) is small. Based on these information and Equation 5.2, Figure 5.3 plots the probability for a RTS frame not to collide with other frames.

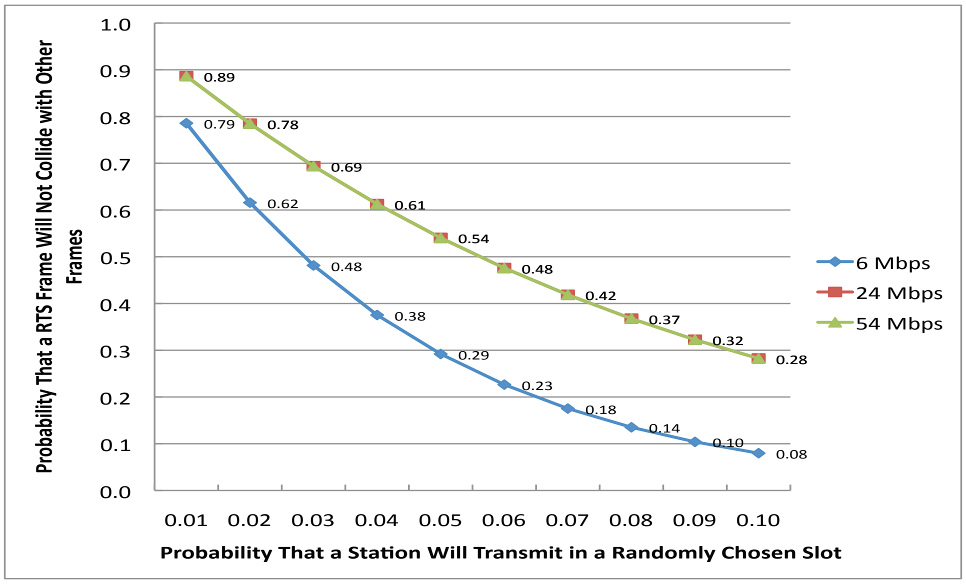

Figure 5.3: Probability for a RTS Frame Not To Collide with Other Frames

From Figure 5.3, we can clearly see that a RTS frame has a high probability not to collide with other frames. Especially when τ is small, the probability of no collision can reach as high as 0.89 when the transmission rate is above 24 Mbps. Also, through the exchange of RTS/CTS, the probability that the original data frame does not collide with other frames has increased to the probability that a RTS frame not to collide with other frames.

It has to be pointed out that the previous calculated RTS transmission time is only the transmission time for RTS itself, it does not include the standard DIFS, SIFS, and IEEE 802.11 ACK. To calculate the total transmission time of a data frame with RTS/CTS and IEEE 802.11 ACK in

IEEE 802.11g only environment, we should use the following equation:

$$Total\,Transmission\,Time =$$
$$DIFS + Physical\,Overhead + RTS + Propagation\,Time +$$
$$SIFS + Physical\,Overhead + CTS + Propagation\,Time +$$
$$SIFS + Physical\,Overhead + DataTime + Propagation\,Time + Signal\,\,Extension +$$
$$SIFS + Physical\,Overhead + IEEE\,802.11\,ACK + Propagation\,Time + Signal\,\,Extension$$

5.3 Algorithms

This section first introduces our original design of a rate adaptation algorithm, named Advanced Rate Adaptation Algorithm (**ARA**). Then it explains how it is extended to a better algorithm called Fast Recovery Rate Adaptation Algorithm (**FRRA**).

5.3.1 Advanced Rate Adaptation Algorithm (ARA)

The key idea of ARA is to make use of the RTS control frame. According to the IEEE 802.11 standard [2], the decision to make use of the RTS transmission is made solely at the sender side. When the size of the current data frame is equal or larger than the *RTS threshold*, the RTS/CTS exchange occurs before the transmission of current data. However, in most typical IEEE 802.11 devices operating in infrastructure based mode, the RTS threshold is set to the largest value (i.e, *2347 bytes*), which actually disabled the exchange of RTS/CTS in most cases. Although the IEEE 802.11 standard defines that the transmission of RTS frame should be triggered by the RTS threshold, use of the RTS frame for other purpose can be found in supplemental standards.

The following section discusses how ARA satisfies the proposed guidelines.

Ability to Differentiate Frame Losses

As explained before, an efficient rate adaptation algorithm should be able to differentiate the cause of frame losses, because different causes of frame losses require different actions. ARA satisfies this guideline by taking advantage of the RTS control frames. When a data frame is transmitted unsuccessfully, a RTS will be sent at the same rate as the failed data frame. If the corresponding CTS can be received successfully, then the current channel has a high probability of supportting this data rate since the RTS is transmitted at the same rate as the failed data frame. Therefore, the failure of previous data frame is probability caused by collision. Remember that such RTS frame has a very low probability of colliding with other frames due to its small size and high transmission rate. If such RTS frame is transmitted successfully but the following data frame still fails, then it is very obvious that the data loss is caused by channel degradation since the bandwidth has already been reserved by the successful exchange of RTS/CTS. By using this method, ARA precisely differentiate frame losses caused by channel degradation from collision.

Fast Response to the Variation of Channels

ARA uses a fast and precise rate adjustment mechanism. As explained above, if the RTS/CTS exchange is successful but the following data frame still fails, the failure must be caused by channel degradation. Therefore, the data rate will be decreased in response for such environment changes. ARA does not act like other rate adaptation schemes which use a threshold to decrease the transmission rate. Using a threshold to adjust data rate is not only inaccurate, but also responds slowly to the variation of the channels.

A Suitable Initial Rate

ARA uses the highest data rate, 11 Mbps for IEEE 802.11b [3] and 54 Mbps for IEEE 802.11g [4] as the initial rate. The reason is because ARA has the ability to quickly identify the cause of frame losses and be able to adjust the transmission rate efficiently in response of channel degradation. Therefore, in an environment where signal strength is strong, ARA does not need to adjust the data rate. Whereas in an environment where signal strength is very weak, ARA can quickly decrease the data rate to a suitable level when it finds out that the channel cannot support the current rate any more.

Ability to Accommodate Poor Channel Condition

ARA is originally designed for the situation where the channel condition is good. Therefore, it does not adapt to the situations where the channel condition is poor.

Ability to Adapt Different Protocols

ARA concentrates on the TCP traffic and does not adapt to UDP traffic.

A Good Metric to Adjust Data Rate

Most of the current rate adaptation schemes use consecutive success count or consecutive failure count as their metric to adjust the data rate. However, this method is not only inaccurate, but also loses the opportunity to grab the short gain period of strong signal strength. ARA uses RTS and the following data frame to identify channel degradation and therefore be able to decrease the transmission rate quickly and precisely. ARA also uses the success count to increase the data rate but does not need to be consecutive. As stated previously, the probability of a successful transmission after ten consecutive success transmission is only 28.5%. Think of a situation when the channel quality is improving, however, transmission error still occurs due to the broadcast nature of the wireless transmission. A single failure of the frame may cause the consecutive success count to reset to 0, which means those rate adaptation schemes have to recount this counter and wait it to reach the threshold before they can increase the data rate. This is not desirable since the current channel quality is improving. ARA does not reset the success count as long as the data rate is not changed in order to grab the period of strong signal quickly.

Compatible with Current Commercial Product

ARA uses the RTS control frame but extend the usage because the RTS may be sent at other rates besides the basic rate. However, no modification is made inside the content of a RTS frame which makes this algorithm still compatible with current commercial product.

ARA is the original design of a rate adaptation algorithm and it satisfies most of the proposed guidelines. It is specially designed for TCP traffics and works very well in situations where the channel quality is not poor. Figure 5.4 shows the state transition of ARA algorithm.

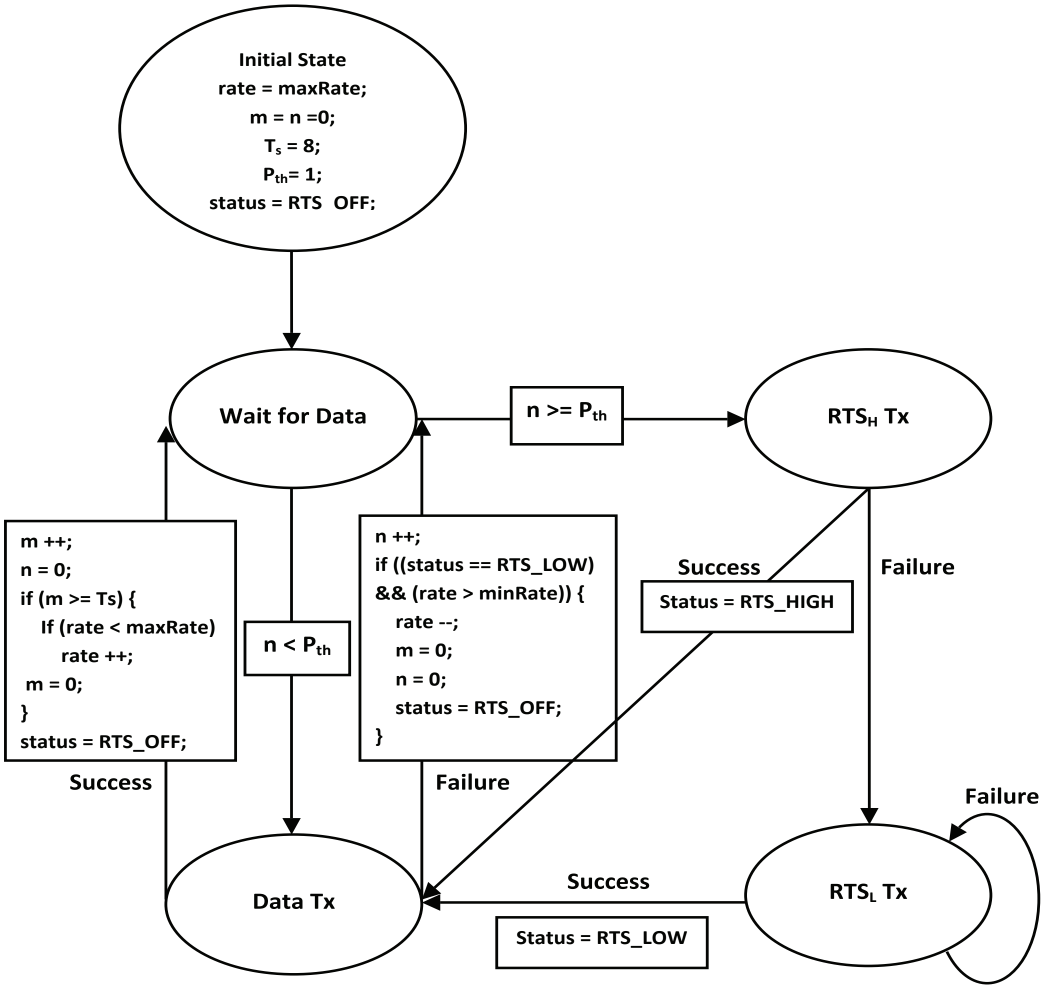

Figure 5.4: ARA Algorithm

5.3.2 Fast Recovery Rate Adaptation Algorithm (FRRA)

FRRA is based upon ARA. However, it extends ARA in several aspects. First of all, it uses states to best identify current network status. The advantage of using states is very obvious: we can easily know the current status and therefore can precisely adjust the transmission rate. As explained in the previous chapter, different cause of frame losses requires differnet actions. We should not decrease the data rate when a frame loss is caused by collision and should not keep the current transmission rate when the channel starts to degrade. By using the predefined states, FRRA knows the exact current network status and therefore can easily make adjustments to the data rate. Basically, it has five states, namely NORMAL, PRE_COLLISION, COLLISION, PRE_FADING and FADING.

Besides using the predefined states, FRRA considers both TCP traffic and UDP traffic while ARA only concentrates on the TCP traffic. Since a TCP traffic has quite different characteristics from a UDP traffic as we have explained in Section 5.1.5. It is very important that a rate adaptation scheme takes both types of traffic into consideration.

As we have explained in the previous section, ARA works well in a situation where the signal strength is not poor. FRRA not only considers the situation where the channel quality is good, but also considers the situation when the channel quality is poor. From Figure 5.1, we can see that a lower data rate gives a higher delivery ratio. However, a higher delivery ratio does not necessarily means a better throughput. This is because a lower data rate normally transmits fewer frames in a given amount of time. Therefore, even though most of the frames can be delivered successfully, the throughput may not be high. FRRA improves the throughput by limiting the minimum rate to 6 Mbps. Note that even though Figure 5.1 shows that when the data rate is 12 Mbps, the highest throughput can be achieved, it is wise to select a lower data rate as the minimum rate in order to make the rate adaptation scheme work in a exremely low signal situation. Using 6 Mbps as the minimum rate not only gives us close to maximum throughput, but also guarantees that FRRA may work well even in a extremely poor channel environment.

Another improvement of FRRA over ARA is the fault tolerance feature. We know that a frame failure occurs even when there is no channel degradation and collision due to the wireless broadcast feature. This kind of failure may cause the rate adaptation schemes to misjudge the cause of frame losses and therefore react incorrectly. FRRA solve this problem by using a probe frame at different states in order to quickly recover from an inappropriate state.

Next, let's look at how FRRA fully satisfies the proposed guidelines we mentioned before.

Ability to Differentiate Frame Losses

FRRA uses the similar idea as ARA to differentiate frame losses. Whenever a frame is lost, FRRA will first send a RTS frame at the same rate as the lost data frame. If the RTS can be transmitted successfully, then the previously lost frame has a high probability of collision since the RTS is using the same transmission rate as the failed data frame. If the RTS/CTS exchange cannot be done at the current rate, FRRA will initiate another RTS/CTS exchange by using the lowest transmission rate. If this exchange is successful but the frame still cannot be transmitted successfully, FRRA will assume that the current channel cannot support the transmission rate and believe the previous frame loss was caused by channel degradation since the successful exchange of RTS/CTS has already reserved the bandwidth. By using this technique, FRRA can precisely differentiate the different causes of frame losses.

Fast Response to the Variation of Channels

In order to adapt to change in the environment, FRRA uses states to best describe the current situation. It has five predefined states, namely NORMAL, PRE_COLLISION, COLLISION, PRE_FADING, and FADING.

NORMAL state means current data rate is steady and there is no interference from another station. At this state, FRRA will keep sending packets without the exchange of RTS/CTS control frames in order to reduce the overhead. If the success count reaches certain threshold, FRRA will try to increase the data rate. After the rate is increased to the next level, if the following packet is transmitted failed, FRRA will immediately fall back to the original rate and the threshold is doubled. This process is similar to AARF [14] to avoid rate oscillation.

At the NORMAL state, whenever a frame failure occurs, FRRA will first assume the lost is caused by collision. It enters a new state called PRE_COLLISION which means predicted collision. In this state, FRRA will initiate the exchange of RTS/CTS control frames to determine the exact cause of this loss. Note that the RTS frame is transmitted by using the *same rate* as the failed data frame, this is the key difference between FRRA and other rate adaptation schemes. If the RTS/CTS exchange is successful, FRRA will know that the channel is still supporting current data rate and the previous frame failure was caused by collision. Therefore, FRRA will change the current state to COLLISION. If the RTS/CTS exchange is unsuccessful, FRRA believes there is a high probability that the previous frame loss was caused by channel degradation since the RTS frame is very small in size and will not easy collide with other frames. FRRA will further judge the cause of this frame loss by entering the PRE_FADING state.

In the COLLISION state, FRRA will set up a RTS Counter ($rtsCounter$) and a RTS Window ($rtsWnd$) to best support the current situation. $rtsCounter$ is initially set to 0 and $rtsWnd$ is initially set to 5 which means the next 5 data frames will be transmitted with the help of RTS/CTS exchange. For each successfully transmitted frame, the $rtsCounter$ will be increased by 1. When it reaches the $rtsCounter$, FRRA will go back to NORMAL state. This is necessary because we do not know whether the collision station still exists. If the collision station does not exist and we are still using RTS/CTS exchange before each transmitted frame, a lot of bandwidth will be wasted due to the unnecessarily transmission of RTS/CTS. It has to be pointed out that, although there are other rate adaptation schemes that also use a RTS Window, FRRA is different in some aspects. The RTS/CTS exchange of FRRA is done at the current transmission rate instead of the lowest rate (1 Mbps) as in other rate adaptation schemes. Since the lowest transmission rate has a longer transmission time and a wider broadcast range, using the lowest rate may cause the RTS frame itself to collide with other frames and therefore may lead these rate adaptation schemes to misjudge the current situation. FRRA transmits the RTS frame at a higher data rate to reduce the probability of RTS collision and therefore provides a better performance in a collision dominated environment. In the COLLISION state, if the RTS/CTS exchange at the current rate is successful but the following data frame is corrupted, FRRA believes that the channel condition has become worse and it will enter a new state called PRE_FADING which means it predicted fading.

In the PRE_FADING state, one RTS frame will be sent at the *lowest* transmission rate. If both the RTS/CTS exchange and the following data frame can be transmitted successfully, then FRRA may have misjudged the cause of the previous frame loss since the new frame is transmitted by using the same data rate as the failed frame. Therefore, FRRA will immediately fall back to the NORMAL state and continue to increase the success counter. Note that the success counter is resumed instead of restarting from 0. This is because the current rate has not been adjusted, therefore, the success counter is still valuable to determine how many frames has been transmitted successfully at this rate. However, if the RTS/CTS exchange still fails or the RTS/CTS exchange is successful but the following data frame cannot be delivered, then obviously the channel cannot support the current transmission rate. Therefore, FRRA will enter the FADING state.

In the FADING state, a RTS will be sent at the lowest rate, if the following data frame cannot be delivered after the successful exchange of RTS/CTS, then the data rate will be decreased to the next level and the state will reenter the NORMAL state. However, if the RTS frame cannot be delivered successfully, then current channel quality must be very poor since the RTS is already transmitted by using the lowest rate. At this situation, FRRA will keep sending RTS frames until it can be transmitted successfully.

A Suitable Initial Rate

FRRA uses the highest transmission rate (54 Mbps) as the initial rate. Most rate adaptation schemes use the medium rate (36 Mbps) as their initial transmission rate because they do not know what is current optimum transmission rate. These rate adaptation schemes either uses statistic method or use the consecutive success or failure count and set up a threshold. Only when the counter reaches the threshold can the current rate be changed. However, this method makes these rate adaptation schemes hard to reach the optimum rate since the adjustment takes time. FRRA can precisely determine the cause of frame losses and can quickly reaches the optimum rate even when the current channel quality is very poor. Therefore, selecting the highest transmission rate gives a better performance when the signal strength is strong and does not decrease the performance when the signal strength is poor since it can decrease the data rate to a lower level very quickly.

Ability to Accommodate Poor Channel Condition

From Figure 5.1, we can clearly see that when the channel condition is very poor, transmitting at the highest rate will not give any throughput since all the transmitted frames will be corrupted. Therefore, we need to decrease the data rate in order to adapt to this situation. However, using the minimum rate (1 Mbps) only ensures that most frames can be delivered successfully. It does not necessarily means that we can get a higher throughput due to the limited transmission data rate. In the design of FRRA, we set the minimum transmission rate to 6 Mbps in order to get the best performance.

We have to point out that transmitting at a lower rate does not only gives a lower throughput, but also makes the rate adaptation algorithm harder to recover to a higher data rate when the channel condition improves. Most of the rate adaptation algorithms can only increase their transmission rate step by step, for example, from 1 Mbps to 2 Mbps, and so on. Therefore, it is unwise to drop the data rate to a very low level. In FRRA, we set the minimum rate that the algorithm may reach to be 6 Mbps. This not only guarantees that we get a better throughput when the channel condition is poor, but also makes it easier to recover from the lower data rate should the channel condition improve.

Ability to Adapt Different Protocols

FRRA is designed not only for TCP, but also for UDP. For a TCP traffic, FRRA uses the help of RTS/CTS control frames to minimize the possibility of packet losses. As we have explained in Section 5.1.5, the loss ratio is a key factor for improving TCP performance.

For a UDP traffic, even though the use of RTS/CTS frames introduces overhead, FRRA monitors current state and avoids unnecessarily use of these control frames in order to get a better performance in a UDP connection.

A Good Metric to Adjust Data Rate

FRRA uses different states to best describe different situations, therefore, it is very easy and clear as how we should adjust the data rate. The decision is quite clear, the NORMAL state means current transmission rate is steady and there is no collision. All the frames are delivered immediately without overhead and nothing needs to be done for the data rate. Only when a frame loss occurs will FRRA change to other states. The COLLISION state indicates there are collision stations and therefore FRRA will keep the current data rate and turns on the RTS/CTS exchange to minimize collision. At FADING state, FRRA knows that the channel condition cannot support the transmission rate and will simply decrease the data rate.

Compatible with Current Commercial Product

The only thing FRRA changes is the transmission of RTS frames at a higher rate rather than the lowest rate. However, this change is not against the IEEE 802.11 standard and can still be implemented in current commercial products.

By satisfying all the above guidelines, a practical and efficient rate adaptation algorithm is proposed and implemented. The state transition of FRRA is shown in Figure 5.5.

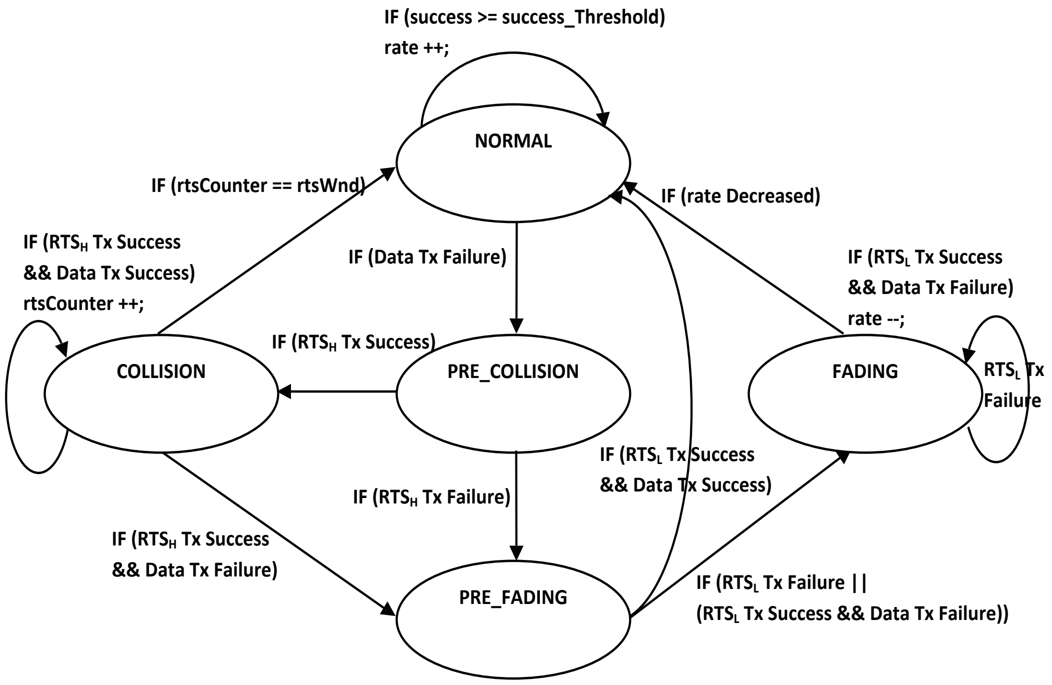

Figure 5.5: FRRA Algorithm

Chapter 6
Implementation

This chapter first explains the Madwifi [1] driver, which is an open source IEEE 802.11 device driver for Atheros cards in Linux and FreeBSD. Then it discusses how the two proposed algorithms are implemented on Madwifi.

6.1 Madwifi Driver

We use Madwifi 0.9.3.2 as the device driver for our Proxim wireless cards to implement the proposed rate adaptation algorithms. In this section, we first explain the basic structure of Madwifi driver, then we will use FRRA as an example to illustrate how to compile, debug, install and run this algorithm in our Linux-based testbed.

6.1.1 Madwifi Basic Structure

The basic structure of Madwifi driver is shown in Figure 6.1. As shown in the figure, the Madwifi driver consists of four components: HAL, ath, NetIEEE 802.11 and Rate Control. Among these components, HAL(Hardware Abstraction Layer) acts as an interface for the other components to access the hardware firmware. The ath component encapsulates HAL and provides specific callbacks for other components. NetIEEE 802.11 component implements most of the IEEE 802.11 features such as frame assembly and disassembly, data encryption etc. Rate Control components is the key component when implementing a new rate adaptation algorithm. It is responsible for selecting a transmission rate for each data frame. Depending on the status of a transmitted frame, different rate adaptation algorithms might make different decisions in order to maximize the network performance.

In our implementation of the proposed algorithms, we made each algorithm an individual module in the operating system. The detailed information on how the proposed algorithms can be built into modules will be shown in the next section.

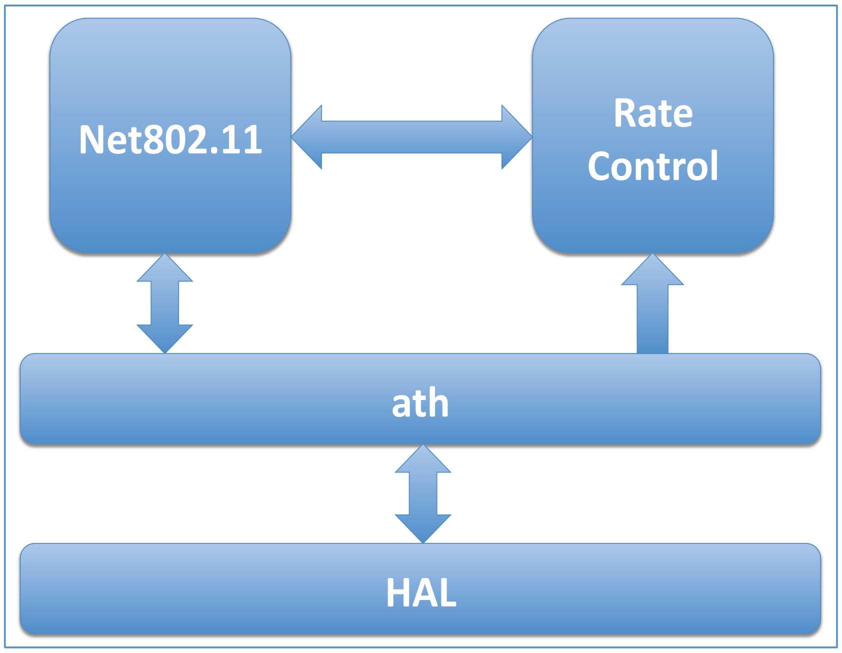

Figure 6.1: Madwifi Basic Structure

6.1.2 Compile, Debug, Install & Run The Proposed Algorithms

This section shows how we install the proposed algorithms as modules in our Linux-based testbed.

Compile The Algorithm

- Open folder madwifi/ath_rate, create a new folder named *frra*.
- Put FRRA's source code into folder *frra*.

- Create a makefile in this folder, the makefile should be similar to other existing rate adaptation algorithms in Madwifi.

- Edit madwifi/net80211/ieee80211_rate.h and add *IEEE80211_RATE_FRRA* to the enum.

- Edit madwifi/net80211/ieee80211_rate.c and add *[IEEE80211_RATE_FRRA] = "ath_rate_frra"* to the module names.

- Edit madwifi/ath_rate/makefile and add *frra* to the appropriate position.

- Edit madwifi/scripts/madwifi-unload.bash (for the unload of Madwifi modules) and add *frra* to the appropriate position.

- Edit madwifi/scripts/find-madwifi-modules.sh (for the removal of Madwifi driver) and add *frra* to the appropriate position.

Debug The Algorithm

In order to debug, first we need to enable debug in source code:

- Enable debug in source code, this can be done by adding the following code:

  ```
  #define FRRA_DEBUG
  #ifdef FRRA_DEBUG
  enum {
  ATH_DEBUG_RATE = 0x00000010 /* rate control */
  };
  ```

- Use "DPRINTF" in the source code to print out the debug message.

  The format for DPRINTF is:

  ```
  DPRINTF(sc, "Debug Message with Variable Value %d", variable);
  ```

  or

  ```
  DPRINTF(sc, "%s", "Debug Message");
  ```

- To see the debug message, type the following command as the root user:

  athdebug rate

  dmesg

Install The Algorithm Into The Linux Kernel:

The procedure to install a new rate adaptation algorithm as a module into the Linux kernel is shown below:

- Remove previous madwifi driver from Linux Kernel:

  Open a terminal, change to folder madwifi_0.9.3.2/scripts and type:

  ./madwifi-unload.bash

  ./find-madwifi-modules.sh $(uname -r) and select *r* (for remove) when prompted.

- Install the new madwifi driver:

  In the madwifi root directory, type:

  make

  make install

Run The Algorithm

- To run FRRA, open a terminal and type the following command as root:

  madwifi-unload.bash

  modprobe ath_pci ratectl=frra

6.2 ARA Implementation

ARA is the original proposed rate adaptation algorithm and concentrates on TCP traffics. It first defines two threshold parameters named Ts (Ts = 5) and P_{th} (P_{th} = 1), Ts is the success count threshold which is used for the algorithm to upgrade the rate to a higher level, and P_{th} is used to check whether previous data frame has failed. If the data is transmitted unsuccessfully, the algorithm will enter a state while RTS/CTS exchange will occur to help differentiate the cause of frame loss. ARA also holds two rate index sets, namely rix and cix. rix is used to hold the current data frame transmission rate index and cix is used to hold the current RTS transmission rate index. A rate index corresponds to the real transmission rate, for example, in IEEE 802.11g, there are in total 12 different transmission rate ranging from 1 Mbps to 54 Mbps and the rate index ranges from 0 to 11 to correspond to these 12 rates.

At the beginning of the state transition, ARA sets the initial rate to be the maximum rate, which is 11 Mbps for IEEE 802.11b and 54 Mbps for IEEE 802.11g. It also initialize variables m, n, and $status$. Variable m is used to hold the number of frames being successfully delivered at current rate and variable n is the consecutive failure count for any lost data packet. It has to be pointed out that even though n is the consecutive failure count, it is *NOT* used in the rate adjustment metric to decrease the data rate. This variable is only used as a condition test to decide whether ARA needs to change to another state. Variable $status$ tells the algorithm whether a normal RTS/CTS exchange (RTS is sent at the lowest rate, denoted as RTS_L Tx) is successful before the transmitting of next data frame. If the exchange is successful, this variable will be set as RTS_ON, otherwise (including the situation where RTS is sent at the same rate as the failed frame), it is set as Normal.

After the variables has been initialized, ARA will wait for the coming data frames. Whenever a data frame is found in the transmission queue, ARA will try to send the data without the help of RTS. If the data is delivered successfully, ARA will increase the success count m by 1 and reset the failure count n to 0. It will also reset variable $status$ to NORMAL because next frame will be transmitted without the help of RTS/CTS control frame. When m reaches the threshold Ts and the current transmission rate is not at the maximum rate, the rate will be increased to a higher level. If the transmission fails, ARA will increase the failure count n by 1. Note that at this point, we do not reset m yet.

When n is equals to 1, it means the previous data transmission has failed. ARA will try to differentiate the cause of this failure. It will first send RTS at the same rate as the failed data ($cix =$

rix), denoted as RTS_H Tx. If the corresponding CTS can be received successfully, several things can be confirmed: 1) The channel condition may support current rate since RTS is transmitted at the same rate as the failed frame; 2) There is a high probability that the previous frame loss was caused by collision; 3) The bandwidth has been reserved because of the successful exchange of RTS/CTS. Therefore, the next data frame will have a high probability to be successful. Since the cause of the previous data frame loss is collision, we do not need to adjust the data rate index rix. And success count m will be resumed because the channel condition does not change. It has to be pointed out that m *does not need to reset to 0 even when a data frame failure occurs, as long as the channel condition does not change,* m *should not be reset and therefore does not necessarily need to be consecutive*. This is quite different with other rate adaptation algorithms and one of the reason why ARA is more efficient than the other algorithms.

However, it is still possible that RTS_H Tx may fail. This happens when the channel condition is degraded. Since RTS is very small and has a low probability of colliding, there is a high probability that the failure is caused by channel fading. In order to confirm this, ARA will send the RTS at the lowest speed ($cix = 0$), denoted by RTS_L Tx, and also the variable $status$ will be set to RTS_ON. At this point, there is a high probability that RTS_L Tx will be transmitted successfully because of the small size and robust transmission rate. After the corresponding CTS has been received, the bandwidth has been reserved for the next data packet. Recall that the data rate index rix is still not changed in the above steps. Therefore the bandwidth has been reserved through RTS_L, but the data is still transmitted using the same rate index rix as the previous failed data. It is very likely that this frame will not be delivered successfully. Since the cause of this failure is clearly due to channel fading, the data rate will be decreased to a lower level and we reset every variables and start the transmission from beginning.

It is very interesting that in some rare situations, the signal strength is so weak that no frame can be delivered successfully. ARA will enter a state that it will keep sending RTS at the lowest rate. In this case, the RTS is serving as the probe packet to monitor the change in channel conditions. ARA will not deliver any data frame until at least one successful exchange of RTS/CTS has occured. At this point, ARA will try to resume the transmission.

6.3 FRRA Implementation

As we have explained in Section 5.3.2, FRRA uses pre-defined states to best describe the current environment status.

The initial state of FRRA is NORMAL, which suppose the wireless device is in a stable environment with limited collision (for example, Home). The RTS/CTS exchange feature is disabled in this state in order to reduce the overhead. If no frame loss occurs, FRRA will stay in the NORMAL state and keeps transmitting data at the current rate until the threshold is reached.

If a frame is lost, FRRA first assumes that the loss is caused by collision. It changes its state to PRE_COLLISION, which means it predicted collision state. At this state, FRRA will send a RTS frame at the same rate as the lost data frame. If this control frame can be transmitted successfully, then there is a high probability that the previous lost data frame is caused by collision. The reason is because the RTS is transmitted by using the same speed as the lost data frame. If the channel condition cannot support current rate, then RTS will also be corrupted by using this rate. Therefore, FRRA will treat the current environment as collision dominated network and enter a new state called COLLISION. If the RTS control cannot be transmitted successfully, then FRRA will enter a state called PRE_FADING, which means it predicted fading because the frame loss may be caused by the channel degradation.

Suppose FRRA is in the COLLISION state, it will turn on the RTS/CTS exchange in order to reduce collision. To enable RTS/CTS, FRRA setups a RTS window ($rtsWnd$). Within this window, all the data frames are transmitted after the RTS/CTS exchange. The key challenge here is to select an appropriate RTS Window size. Setting it to a large size may help to reduce collision, but introduces too many overheads. Besides, the RTS window also has different effects on TCP and UDP traffics. Figure 6.2 and Figure 6.3 show the average TCP throughput and average UDP goodput in a collision free environment where the signal strength is strong.

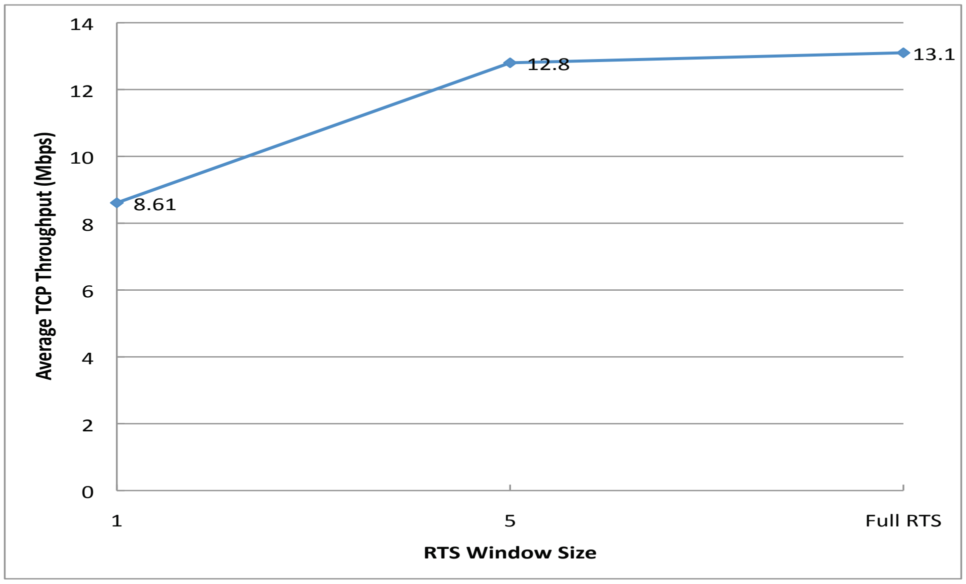

Figure 6.2: TCP Throughput for Different RTS Window Size in a Collision Free Network

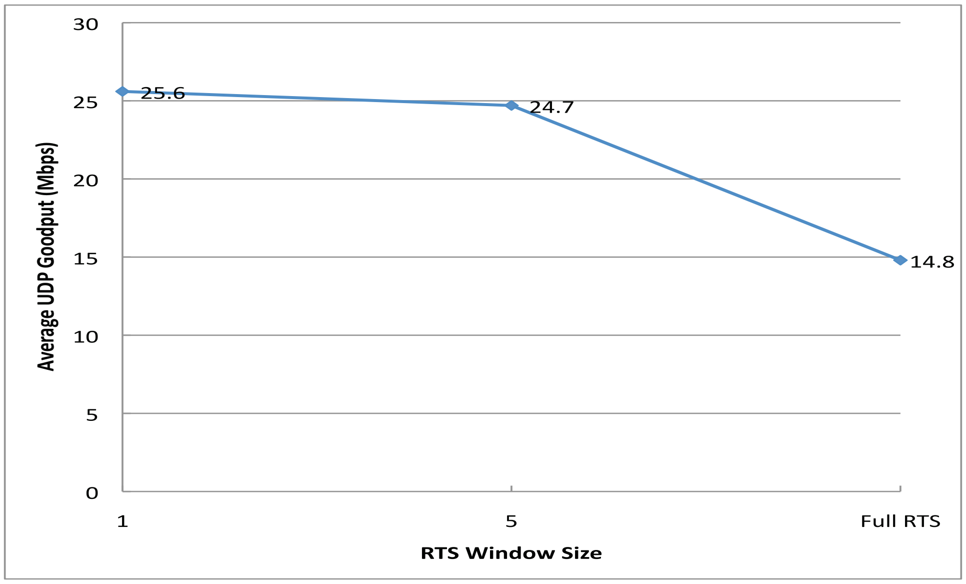

Figure 6.3: UDP Goodput for Different RTS Window Size in a Collision Free Network

From these two figures, we can clearly see that when we set the RTS Window size to be too small, for example 1, as shown in the figures, it may achieve a good performance for UDP traffic since it does not introduce too much overhead by sending RTS. However, the throughput of TCP is poor due to a higher failure rate as shown in Figure 6.4. As we have explained in the previous chapter, the failure rate is very important for a TCP connection. Even one packet failure may cause the TCP congestion window to decease to 1 (TCP Tahoe) and cause the throughput to be dramatically decreased. Consider the fact that there are more than thousands of packets delivered in one connection, setting the RTS Window size to be very small is not desirable.

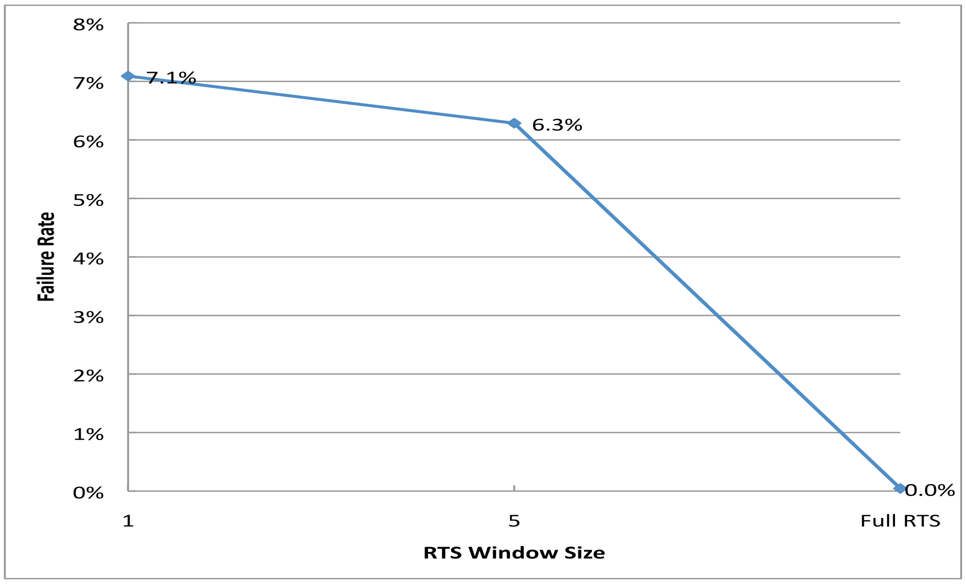

Figure 6.4: Failure Rate for Different RTS Window Size in a Collision Free Network by Using TCP Connection

What about turning on the RTS/CTS all the time? Although this may lower the failure rate, it introduces too much overhead causing UDP goodput to suffer, as shown in Figure 6.3.

Besides TCP throughput and UDP goodput, another important factor in a UDP traffic is jitter. Jitter is often used as a measure of the variation in packet delay. A network with constant latency has no variation and the jitter will be zero. We know that UDP is mostly used in real-time systems; therefore, jitter is a very important factor in the assessment of UDP traffic. Figure 6.5 shows the average jitter for different RTS Window sizes in UDP traffic. From this figure, we can clearly see that when we set the RTS Window size to be too small or too large, the jitter will be increased.

Considering the TCP throughput, UDP goodput and jitter in a UDP traffic, in the implementation of FRRA, we set the initial value of $rtsWnd$ to be 5.

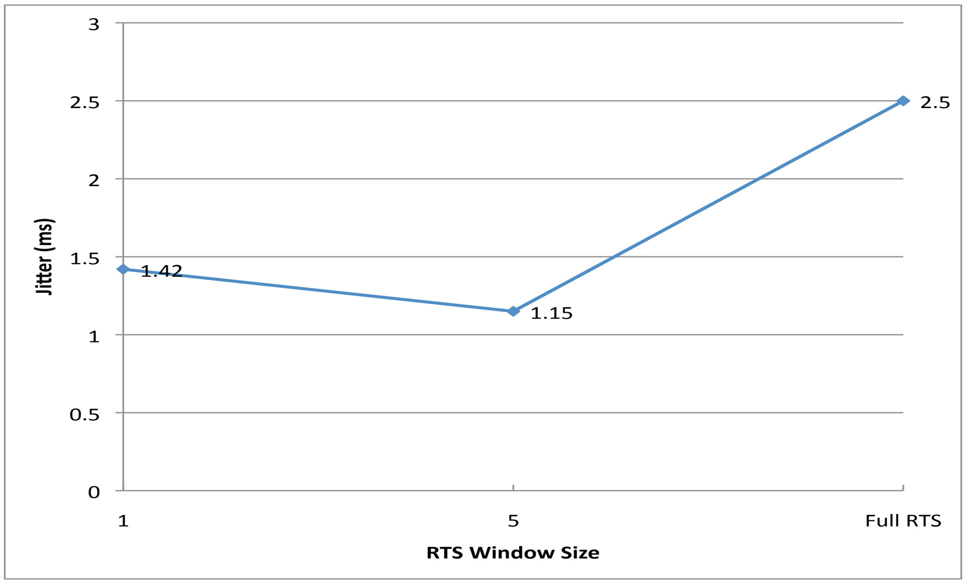

Figure 6.5: Jitter for Different RTS Window Size in a Collision Free Network by Using UDP Connection

Besides the RTS Window, FRRA also sets up another variable called $rtsCounter$; this variable is used to monitor how many data frames have been transmitted with the help of RTS/CTS exchange. For each data frame that uses the RTS/CTS, the $rtsCounter$ is increased by 1. When the $rtsCounter$ reaches $rtsWnd$, FRRA will return to NORMAL state. This is to identify whether the collision stations still exist. If there are no more collision stations, staying in the COLLISION state will introduce a lot of overheads and result in performance decrease.

Suppose the $rtsConter$ reaches $rtsWnd$ and FRRA recovers to the NORMAL state, but the following data frame transmission fails. Then there is a high chance that the collision stations still exist since the previous data frame can be transmitted successfully with the help of RTS/CTS exchange. FRRA will immediate return to the COLLISION state and the $rtsWnd$ will be increased by 1 to indicate that we are currently still in a collision dominated network. However, if this frame can be delivered successfully at NORMAL state, then the collision station might not exist any more. FRRA will stay in the NORMAL state and decrease the $rtsWnd$ by 1 to reflect this change.

In the COLLISION state, if a data frame transmission fails, then it is very possible that the loss is caused by channel degradation since the exchange of RTS/CTS has already reserved the bandwidth. In this case, FRRA will enter a state called PRE_FADING meaning it predicted fading.

Note that FRRA will also enter PRE_FADING if the initial RTS/CTS exchange fails, as shown in Figure 5.5.

In the PRE_FADING state, FRRA will first try to send a RTS frame by using the lowest transmission rate (1 Mbps). If the RTS/CTS exchange is not successful, or the exchange is successful but the following frame fails, then it is clear that the channel cannot support the current transmission rate. FRRA will enter a new state called FADING to decrease the transmission rate. However, if a frame can be transmitted at this state, then FRRA might have misjudged the cause of previous failed frame. Therefore, it will immediately fall back to the NORMAL state.

The FADING state is the only state that FRRA will decrease the data rate. At this state, if current transmission rate is not minimum, FRRA will decrease the data rate to next level and return to NORMAL state hoping that the newly decreased rate can be supported by the changed channel and the whole process starts again.

Chapter 7

Evaluation

In order to compare and evaluate the proposed rate adaptation algorithms, several existing rate adaptation schemes are selected. For those schemes whose rate adjustment metric is based upon RSSI or SNR, they are not chosen for the following two reasons: 1) RSSI or SNR is not a good indicator of channel conditions in [5,16]; 2) Current WLAN adaptors do not provide RSSI or SNR specification mapping to data rates, which makes the implementation quite difficult. Therefore, only those schemes with their rate adaptation metric based upon loss rate are selected. Table 7.1 summarizes the characteristics of these selected rate adaptation schemes.

Loss Differentiation	**Consecutive success or failure Count**	**Loss Statistics**
No	AARF	SampleRate
Yes	CARA	RRAA

Table 7.1: Representative Rate Adaptation Schemes

7.1 Methodology

Our Linux-based testbed consists of one wireless router and multiple laptops. Each laptop is equipped with one wireless PC Card. The detailed information for the hardware and software configuration can be found in Table 7.4 and Table 7.5 respectively. The proposed rate adaptation algorithms are implemented on the client (laptops) side because of the open architecture feature. We select these PC Cards for several reasons: 1) These PC Cards use Atheros AR5212 chipset, this chipset allow us to easily switch between IEEE 802.11b/g modes for experimental purpose; 2) Atheros AR5212 chipset is supported by the open source driver Madwifi [1]. Madwifi driver is designed in layers which makes it easy to make modifications to the driver; 3) The Madwifi driver is organized into different modules, and the rate adaptation mechanism is deployed as an individual module. This architecture makes the implementation of a new rate adaptation algorithm easier; 4) The Madwifi driver also provides a lot of useful tools which we can use to collect the communication statistics for each experiment.

The experiments are conducted on an indoor Linux-based testbed and campus network, each experiment runs for 2 minutes in order to cover the channel variations. To minimize interference from other people near our lab, we run these experiments from 12:00 AM to 5:00 AM when there are limited people activities in the building.

The two proposed algorithms are implemented on a Madwifi driver, which is the only available open source IEEE 802.11 device driver for Atheros cards in Linux. To test both the TCP and UDP performance of different rate adaptation schemes, we use Iperf [22] as the network testing tool.

In order to see how the channel conditions affect the performance of these rate adaptation schemes, we have selected three different locations where the signal level is dramatically different. Table 7.2 summarizes the channel conditions at these locations and Table 7.3 shows the channel conditions for our campus wireless network. To avoid the heavily congested wireless channels (1, 6, 11) in our lab, we select Channel 9 to do our expeirments.

Locations	**Channel**	**Signal Level**	**Noise Level**	**Link Quality**
Location 1	9	-44 dBm	-95 dBm	51/70
Location 2	9	-58 dBm	-95 dBm	37/70
Location 3	9	-67 dBm	-94 dBm	27/70

Table 7.2: Channel Conditions at Different Locations

AP Name	**Channel**	**Signal Level**	**Noise Level**	**Link Quality**
tsunami	1	-48 dBm	-96 dBm	48/70

Table 7.3: Channel Conditions for Campus Wireless Network

Besides interference from channels, the frame size also has a great impact on the performance of the rate adaptation schemes. Figure 7.1 shows how different datagram size might affect the performance of these rate adaptation schemes in location 1 where no collision stations exist.

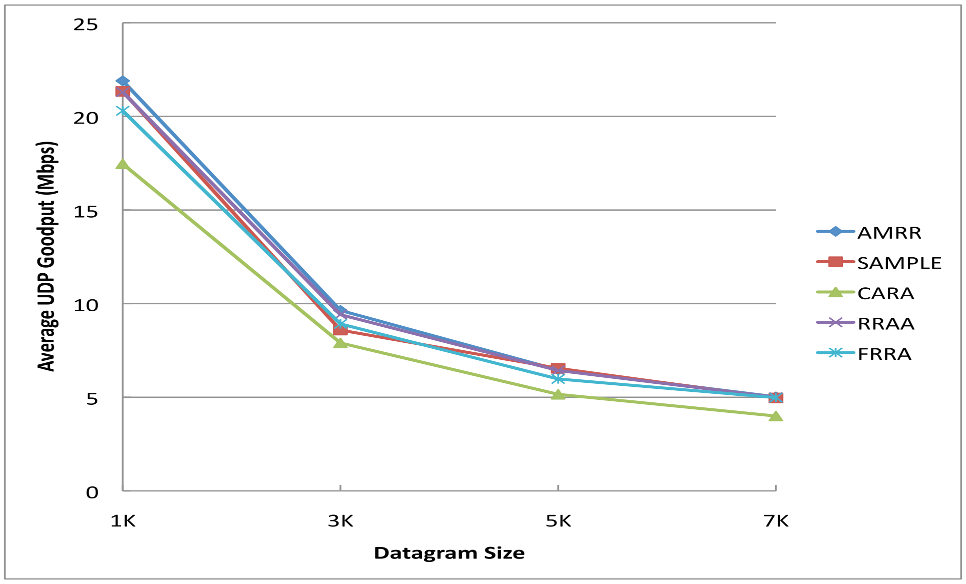

Figure 7.1: Impact of Different Datagram Size to a UDP Traffic

From Figure 7.1, we can see that as the datagram size increases, the UDP goodput decreases. This is obvious because a larger frame means more transmission time, and more transmission time results a higher failure rate.

Considering the above factors, in our controlled experiments, we choose Channel 9 to avoid interference with other wireless channels. We set the TCP maximum segment size (MSS) to be 1460 bytes and the UDP datagram size to be 1470 byte. These two sizes are actually the default sizes in Iperf. Besides the packet size or datagram size, we set the TCP window size to be 250k and the UDP bandwidth to be 54 Mbps in Iperf. For each experiment, we let the rate adaptation scheme runs for 2 minutes. After each experiment, for a TCP connection, we record the throughput, total transmitted frames, failure rate and RTS ratio. For a UDP connection, we record the goodput, total transmitted frames, delivery ratio, RTS ratio and jitter. The whole process is repeated several times to collect the average communication statistics.

7.1.1 System Configuration

This section lists the system configuration for the evaluation. Table 7.4 summarizes the hardware configuration and Table 7.5 shows the software configuration as well as the tools used in the experiments.

Hardware	**Manufacturer**	**Model**
Router/Access Point	D-Link	DIR-655
Wireless Cards	Proxim	Orinoco Gold 8470-FC
Chipset	Atheros	AR5212 (b/g) Rev.01

Table 7.4: Hardware Configuration

Software	**Name**	**Version**
Operating System	Fedora 7	2.6.21-1.3194.fc7
Wireless Driver	Madwifi	0.9.3.2
Network Testing Tool	Iperf	2.0.4

Table 7.5: Software Configuration

7.1.2 Network Configuration And Topology

This section shows the different network configuration used in the experiments. Different types of experiment environments are designed to evaluate the performance of the proposed algorithms and those selected rate adaptation schemes. The settings include the situations where the wireless station is static or mobile, and whether the station is in a collision free network or in a collision dominated network. Besides these considerations, we also consider the impact of different traffic types. TCP traffics are normally used in information and data retrieval applications such as web browser and email etc., whereas UDP traffics are normally used in real-time systems such as VOIP or audio and video streaming. Since different traffic types are used in different applications, both TCP and UDP traffics are tested in most scenarios.

Static Station in a Collision Free Network: This is a typical home network with only one fixed server and one static wireless station. The purpose of this experiment is to evaluate how different rate adaptation schemes perform in a simple environment where the frame loss is only caused by channel fading. The network configuration is shown in Figure 7.2.

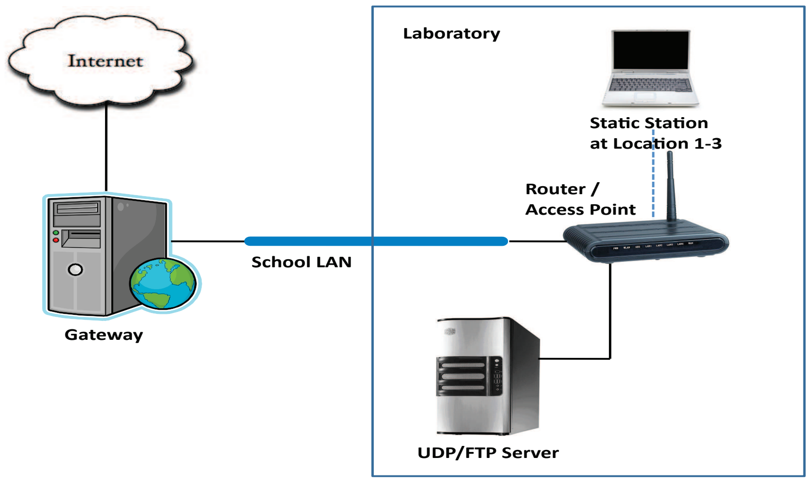

Figure 7.2: Static Station in a Collision Free Network

Static Station in a Collision Dominated Network: Such network topology is often seen in an office environment where many users access the WLAN network at the same time. In such an environment, many users may access the AP simultaneous which may lead to a lot of collision. This is especially true when there are hidden terminals. To simulate this environment, we set the number of collision stations from one to six to see the impact of these collision stations. The collision stations are placed within the AP's range and may send traffic to the AP within the experiment time. Each collision station is loaded with SampleRate, which is the default rate adaptation module in Madwifi. The collision stations always use the same traffic type as the client station, which means TCP VS. TCP and UDP VS. UDP. Figure 7.3 shows the network configuration for this scenario.

Static Station in a Mixed Environment: The network configuration of this scenario is similar to the previous network configuration. However, in such environment, both channel fading and collision have a great impact on frame losses. In our lab, we use six collision stations to generate

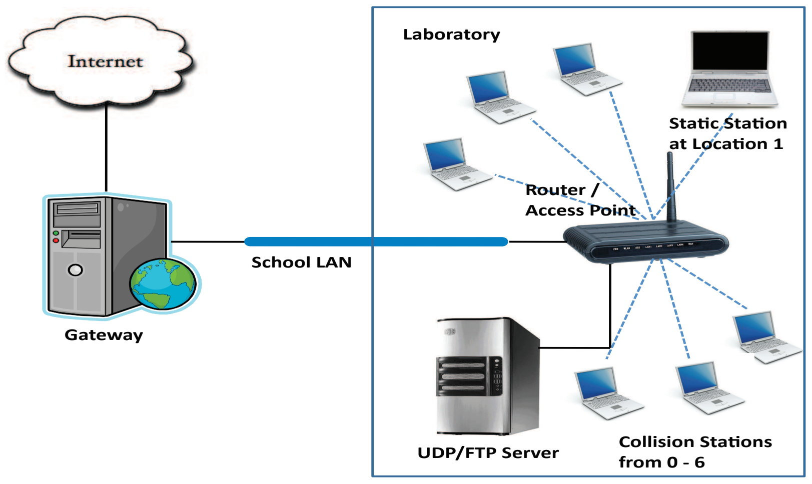

Figure 7.3: Static Station in a Collision Dominated Network

collisions and put the client station at Location 2 for channel fading. Figure 7.4 shows the network configuration for this scenario.

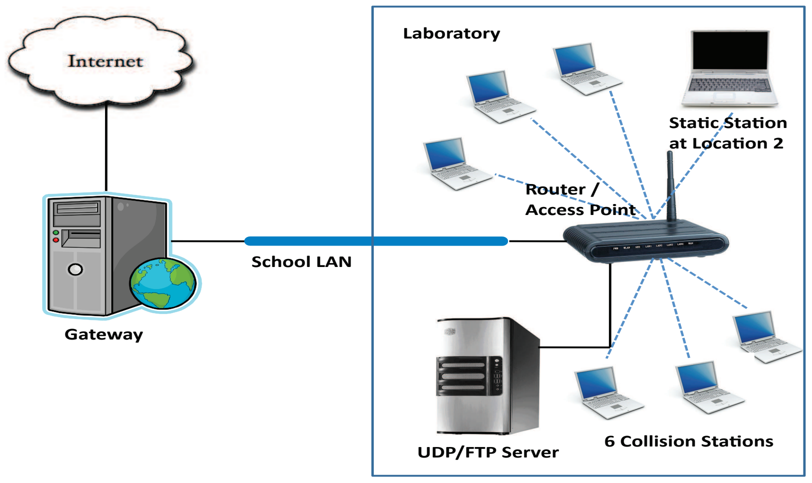

Figure 7.4: Static Station in a Mixed Environment

Mobile Station in a Collision Free Network: This scenario emulates a person using his PDA phone to access the WLAN network at home to make a VoIP call or synchronize business emails while moving around in his house. The purpose of this experiment is to measure the impact of the mobility on the performance of these rate adaptation schemes. The network configuration for this experiment is shown in Figure 7.5.

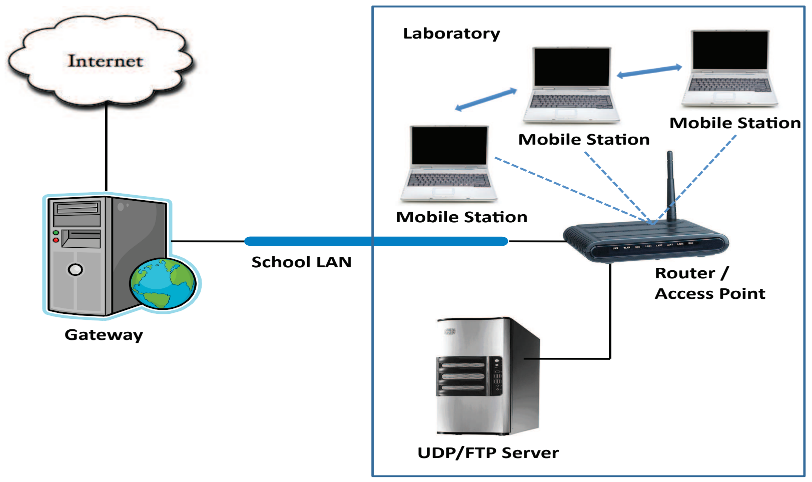

Figure 7.5: Mobile Station in a Collision Free Network

Mobile station in a Collision Dominated Network: This experiment emulates a hot spot with people moving around to access a wireless network. Measurement from such setting shows the impact of both mobility and collisions on the performance of these rate adaptation schemes. The network configuration for this scenario is shown in Figure 7.6.

Static Station in Campus network: This is the field test by using a laptop connecting to our school access point and keep sending data to a server connected to the school LAN. Figure 7.7 shows part of the campus map where the location for the laptop and the server are marked. The network configuration for this field test is shown in Figure 7.8.

7.1.3 Network Test Tool (Iperf) Configuration

Iperf [22] is a commonly used network testing tool which is supported by the National Laboratory for Applied Network Research [23]. It is an open source software and can be installed and run on different platforms including Windows, Linux and Unix. It can create both TCP and UDP

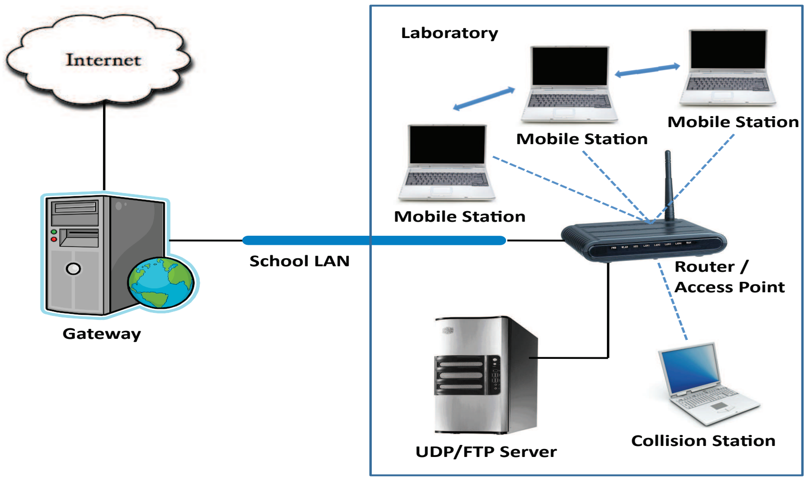

Figure 7.6: Mobile Station in a Collision Dominated Network

data streams to test the network throughput. Iperf provides various commands for easy configuration of the network. Table 7.6 summarizes some of the most useful commands. When Iperf is used for TCP testing, it measures the throughput of the payload. When it is used for UDP testing, it allows the user to set the maximum available bandwidth and specify the datagram size and provides results for datagram throughput and packet losses.

Command Line Option	**Required Parameter**	**Description**
-s	None	Run Iperf in server mode.
-c	Server name or address	Run Iperf in client mode and connect to a server.
-p	Port number	The server port for the server to listen on.
-t	Time	The time in seconds to transmit for.
-i	Time interval	The interval time in seconds between periodic reports.
-w	TCP window size	Sets the socket buffer sizes to the specified value.
-u	None	Use UDP rather than TCP.
-b	Bandwidth	The UDP bandwidth to send at, in bits/sec.
-f	Format	A letter specifying the format to print bandwidth numbers in.
-h	None	Print out a summary of commands and quit.

Table 7.6: Iperf Most useful Commands

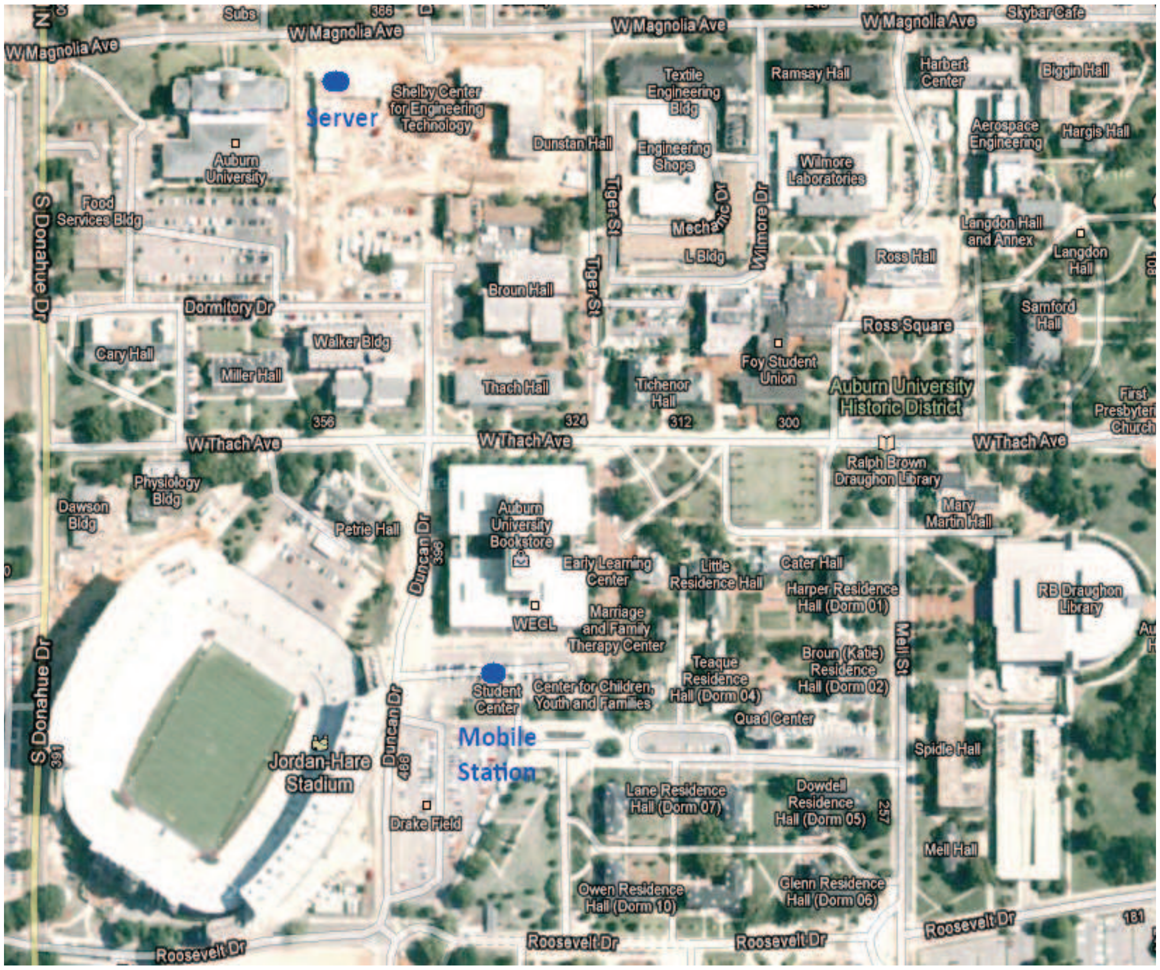

Figure 7.7: The Campus Map for Field Test

Disable Firewall

In order to use a different port number rather than the default one (5001), the firewall in Linux has to be disabled.

- To disable the firewall in Fedora 7, open the terminal and type the following command as root:
  yum install system-config-securitylevel-tui
  /usr/sbin/system-config-securitylevel-tui

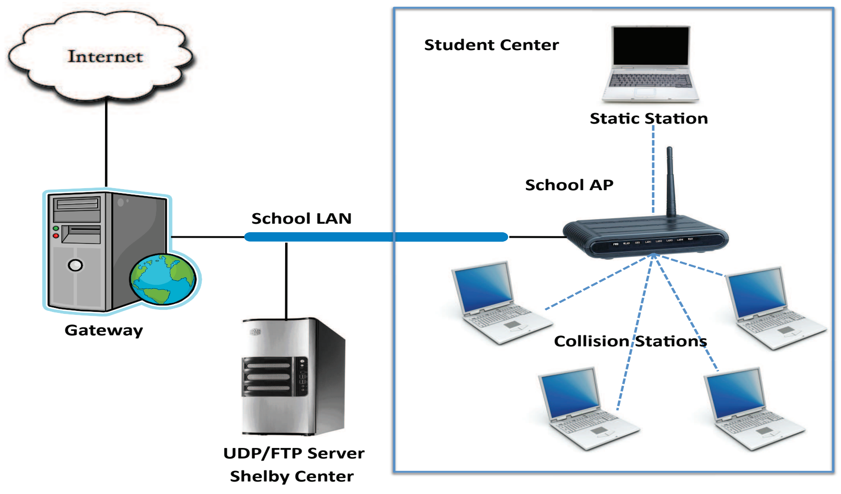

Figure 7.8: Static Station in The Campus Network

Iperf TCP Testing

- To use Iperf for the TCP testing, on the server side, type:

  iperf -s -p portnumber

  For example: *iperf -s -p 5001*

- On the client side, type:

  iperf -c serveraddress -p portnumber -t timeperiod -i timeinterval -w tcpwindow

  For example: *iperf -c 192.168.1.100 -p 5001 -t 120 -i 5 -w 250k*

Iperf UDP Testing

- To use Iperf for the UDP testing, on the server side, type:

  iperf -s -p port number -u

  For example: *iperf -s -p 5001 -u*

- On the client side, type:

  iperf -c serveraddress -p portnumber -t timeperiod -i timeinterval -u -b bandwidth

  For example: *iperf -c 192.168.1.100 -p 5001 -t 120 -i 5 -u -b 54m*

7.1.4 Collect Statistics After The Experiments

- To collect the statistics after one experiment (total transmitted frames, failed frames, RTS enabled frames, jitter for UDP traffic etc.), open a terminal and type the following command as root:
  athstats

7.2 Experiment Results

This section illustrates the experiment results of ARA and FRRA. For ARA, only TCP traffic are tested. For FRRA, both TCP and UDP traffics are tested. From the data we collected, both ARA and FRRA outperforms other selected rate adaptation schemes in most of the scenarios.

7.2.1 ARA Experiment Results

Static Station in a Collision Free Network: In this scenario, different rate adaptation schemes are run in three different locations, where Location 1 has the strongest signal strength and Location 3 has the weakest signal strength. The experimental results for this scenario are shown in Figure 7.9. Note that AMRR [14] is the implemented version of AARF [14] in Madwifi [1]. As shown in Figure 7.9, ARA performs the best in all these three locations, especially in Location 1 where the signal strength is the strongest. It has to be pointed out that even though ARA performs the best, the throughput of all these schemes are close to each other. The reason is because in this scenario, there is no collision. Therefore, all the frame losses are caused by channel fading.

Static Station in a Collision Dominated Network: This is a situation where both channel fading and collision exists. Figure 7.10 shows the TCP throughput in this senario. ARA performs very well when the signal strength is strong. When the channel quality decreases, both channel fading and collision may cause frame failure. For the first generation algorithms, they cannot differentiate the cause of frame losses and therefore perform poorly in Location 3.

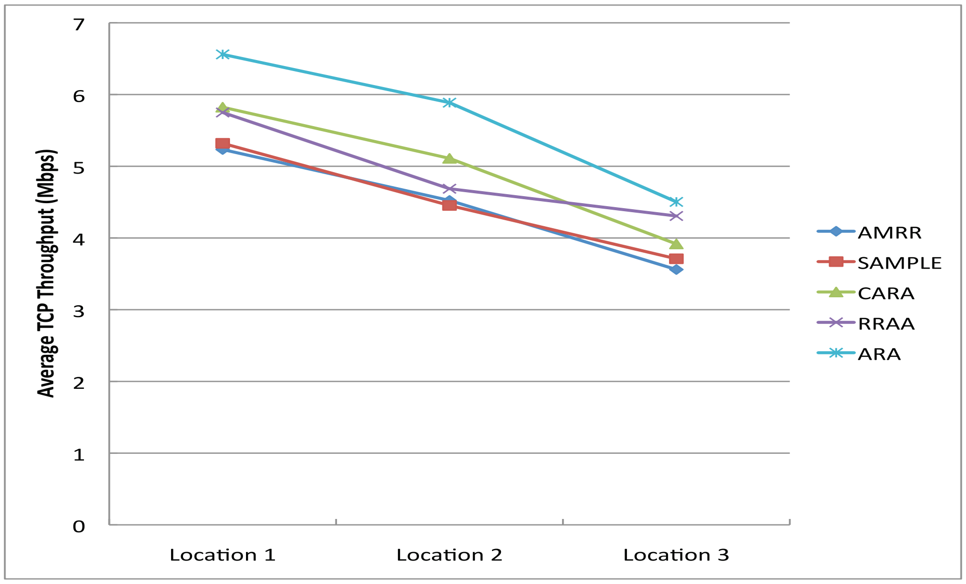

Figure 7.9: ARA: TCP Throughput for a Static Station in a Collision Free Network

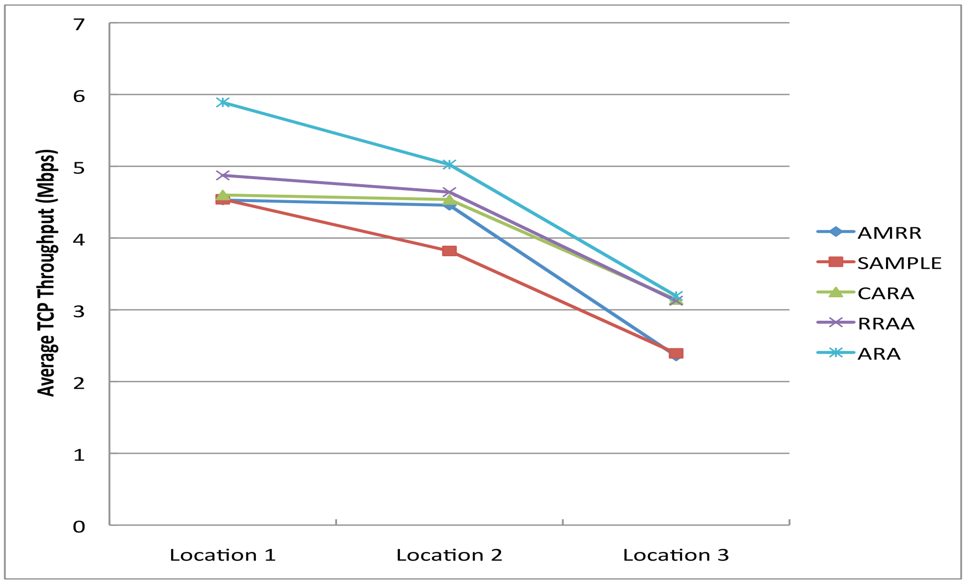

Figure 7.10: ARA: TCP Throughput for a Static Station in a Collision Dominated Network

Mobile Station in a Collision Free Network: In this scenario, a user is carrying a laptop and moving around the access point. The experimental results are shown in Figure 7.11. From this figure, we can see that ARA performs the best among all the schemes, however, the throughput is still close to each other because there is still no collision.

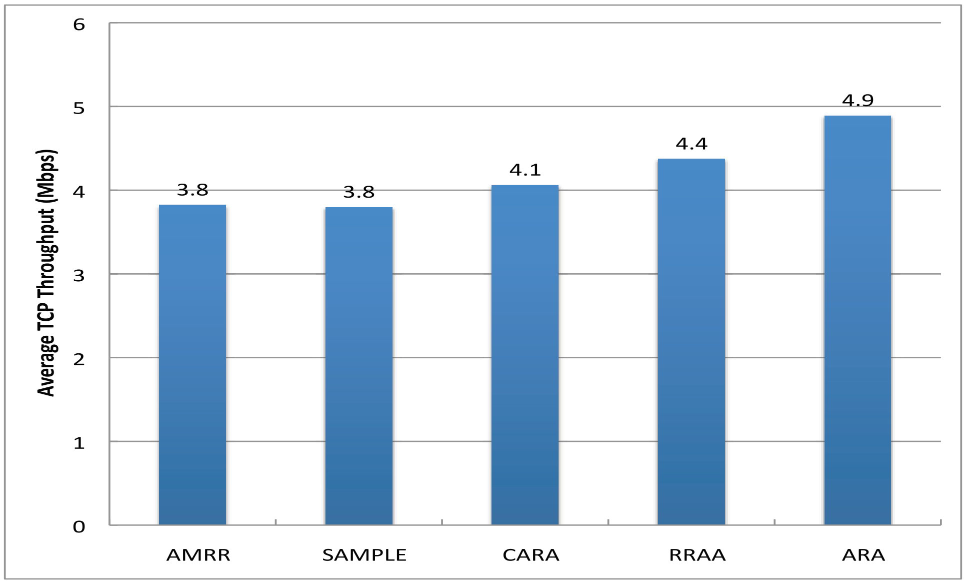

Figure 7.11: ARA: TCP Throughput for a Mobile Station in a Collision Free Network

Static Station in Campus network: This is a realistic scenario, a user is using a laptop in the student center where a lot of other students are using the same campus wireless network. There are a lot of collisions expected in this scenario. The results for this scenario are shown in Figure 7.12.

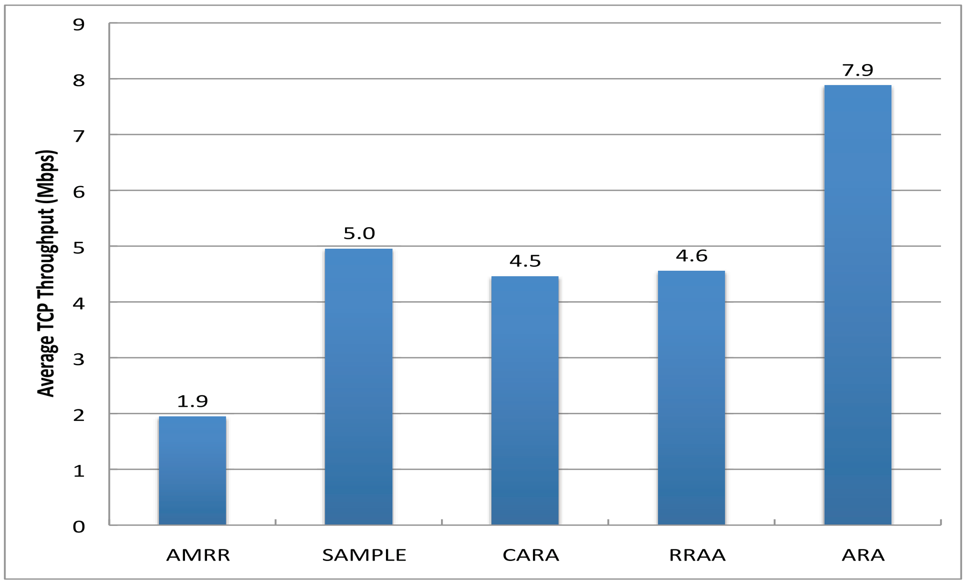

Figure 7.12: ARA: TCP Throughput for a Static Station in The Campus Network

From Figure 7.12, we can clearly see that ARA outperforms all the other rate adaptation schemes and the performance is quite different among these schemes. The reason for this difference is because: 1) In this scenario, there are a lot of collisions. Therefore, most frames losses are caused by collision instead of channel fading. AMRR mistakenly treats all the frame losses as channel fading and therefore keep decreasing the data rate. Which greatly decrease the throughput; 2) Even though SampleRate [16] does not differentiate the cause of frame losses, it transmits the frames in cycles. At each cycle, it will pick a data rate which may theoretically produce a better throughput than the current rate. This prevents SampleRate from decreasing the data rate as AMRR would and therefore still provides good throughput; 3) CARA [19] and RRAA [20] somehow differentiate frame losses and therefore produce a better results than AMRR; 4) ARA precisely differentiate the frame losses and react accordingly. Therefore, ARA provides the best throughput.

7.2.2 FRRA Experiment Results

Static Station in a Collision Free Network: Let's first look at the TCP performance in this scenario. Figure 7.13 shows the average TCP throughput in different locations for FRRA and the four other selected rate adaptation schemes. From this figure, we can see that FRRA has more than 67% throughput improvement over AMRR at Location 1. It has a 30% throughput improvement at Location 2, and more than 110% improvement at Location 3. From these statistics, we can see that FRRA performs very well in different locations. It performs especially well when the channel quality is good or poor.

Figure 7.14 shows the average UDP goodput in these locations. It is very clear that all the algorithms perform very close to each other in this scenario besides CARA. CARA performs the worst when the signal strength is strong. This is because when the signal strength is strong, most of the schemes will use the highest data rate to transmit the data frames. However, due to it's design, CARA transmits some extra RTS and therefore introduces overheads causing the UDP goodput to decrease. This figure also shows that for a UDP traffic in an environment where there is little

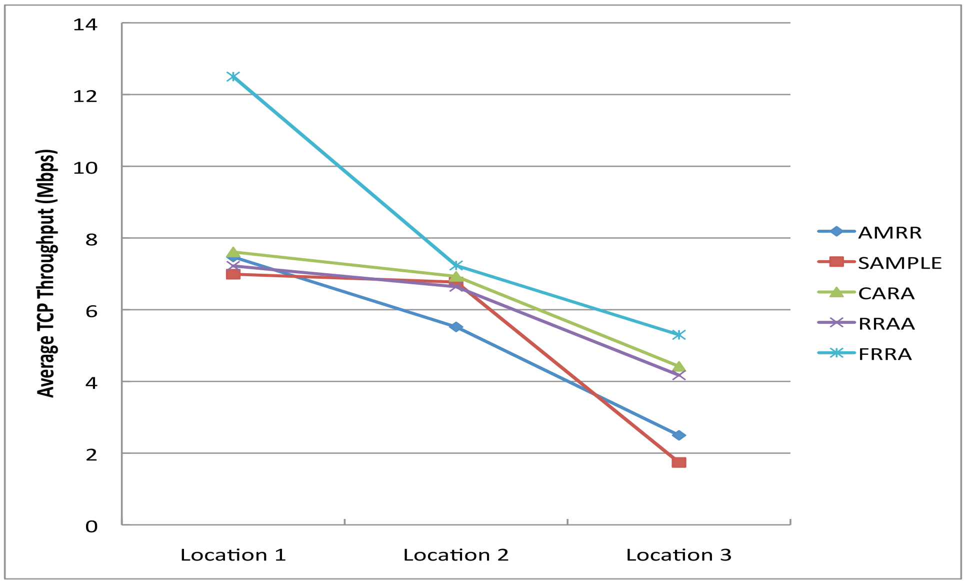

Figure 7.13: FRRA: TCP Throughput for a Static Station in a Collision Free Network

collision, the second generation rate adaptation schemes do not have an advantage over the first generation rate adaptation schemes. The reason is because most of the frame losses are only caused by channel fading, and both the first generation schemes and the second generation rate adaptation schemes can handle such losses.

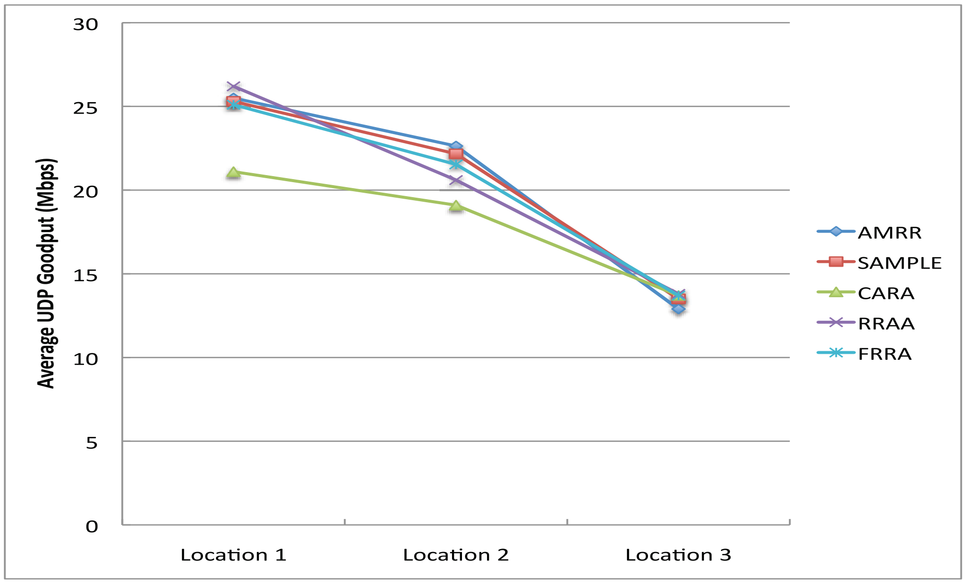

Figure 7.14: FRRA: UDP Goodput for a Static Station in a Collision Free Network

Static Station in a Collision Dominated Network: In this scenario, we want to test how the number of collision stations may affect the performance of different rate adaptation schemes. In our experiments, the number of collision stations are set from zero to six. Both TCP and UDP traffics are tested to see the impact of collisions. In order to minimize the effects of channel degradation, we put the client station in Location 1 where the signal strength is the strongest. Figure 7.15 shows the average TCP throughput for FRRA and other selected rate adaptation schemes.

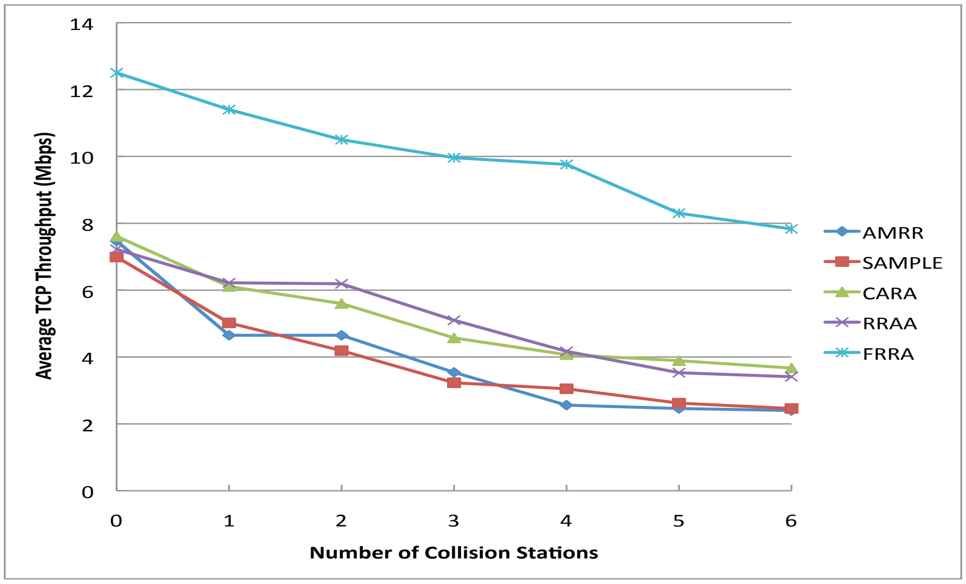

Figure 7.15: FRRA: TCP Throughput for a Static Station in a Collision Dominated Network

As we can see from Figure 7.15, FRRA has a much higher throughput comparing with other selected rate adaptation schemes. When the number of collision stations is increasing, FRRA performs better and better. It has a throughput improvement of 67% over AMRR when there is no collision station. When there are six collision stations, the throughput improvement increases to 226%. We can also see from this figure that when there is any collision station exists, even one collision station makes the TCP performance quite different between the first generation and second generation rate adaptation schemes. This also clearly shows that the second generation rate adaptation schemes take into consideration the frame losses caused by collision and can make some adjustments.

Figure 7.16 shows the total transmitted frames during the experimental period. From Figure 7.16, we can see that the second generation rate adaptation schemes transmitted more frames comparing with the first generation schemes because they consider the collision situation and therefore do not decrease the data rate, and therefore may transmit more frames.

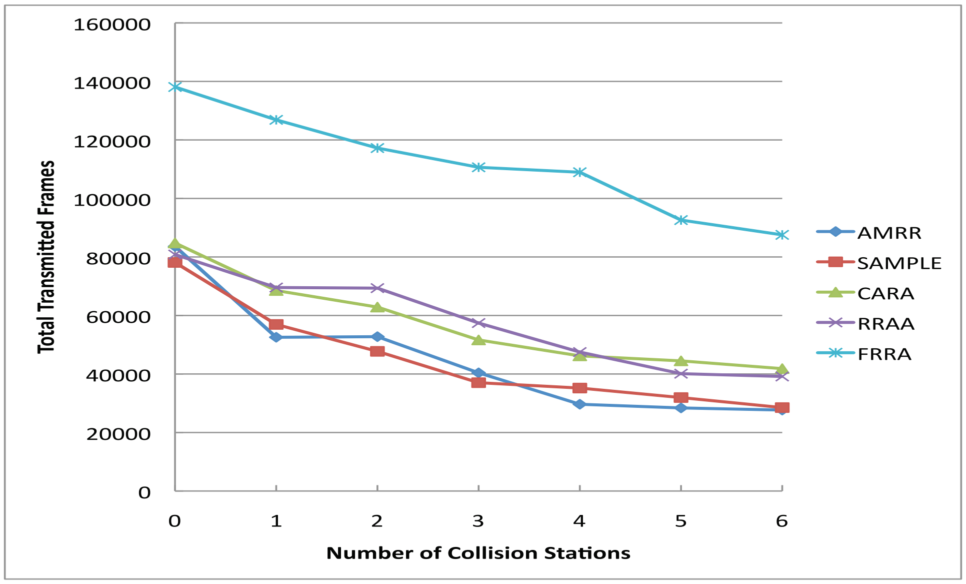

Figure 7.16: FRRA: Total Transmitted Frames for a Static Station in a Collision Dominated Network by Using a TCP Connection

Another important factor for TCP transmission is the failure rate. As we have explained before, TCP has the congestion control mechanism and has a slow start process. Even one packet failure may cause the TCP congestion window to decrease to the minimum and therefore greatly reduces the throughput. Figure 7.17 shows the failure rate for different rate adaptation schemes. As shown in this figure, the second generation rate adaptation schemes has a lower failure rate comparing with the first generation schemes.

To see how the second generation rate adaptation schemes effectively avoid collision in a collision dominated network, we have calculated the RTS enabled frames ratio and shown it in Figure 7.18. As shown in the figure, the first generation rate adaptation schemes does not use RTS control frames to minimize collision because they treat all the frame losses as being caused by channel fading. The second generation rate adaptation schemes use RTS/CTS exchange to control collisions and therefore may get a much better throughput. We can also see from the figure

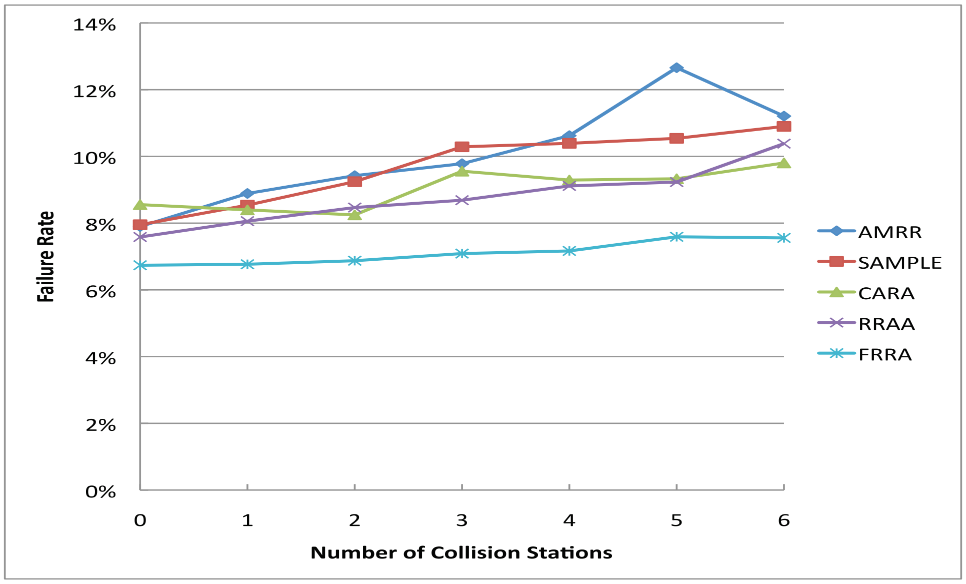

Figure 7.17: FRRA: Failure Rate for a Static Station in a Collision Dominated Network by Using a TCP Connection

that as the collision stations increase, the RTS Ratio for the second generation rate adaptation schemes is increasing to reflect this change. Note that in this figure, FRRA has a RTS ratio around 30% whereas CARA and RRAA has a RTS ratio around 10%. This does not means that FRRA wastes a lot of bandwidth to transmit RTS. This is because for FRRA, most of the RTS frames are transmitted by using a higher data rate (same rate as the failed data frame) whereas CARA and RRAA use the lowest data rate (1 Mbps) to transmit the RTS. Therefore, even though FRRA has a much higher RTS ratio comparing with CARA and RRAA, it still has a higher throughput than these two schemes.

From these statistics, we can know why FRRA has the highest throughput in this scenario: 1) FRRA can precisely detect the cause of frame losses and therefore does not decrease the data rate as the first generation rate adaptation schemes would do. This can be proven from Figure 7.16, where we can see clearly that FRRA transmitted more frames than other rate adaptation schemes; 2) FRRA has a predefined COLLISION state, where all the frames are transmitted with the exchange of RTS/CTS. These exchanges reserve the bandwidth and therefore minimize the collision probability. Figure 7.18 shows the percentage of RTS Enabled Frames during the transmission period. From this figure, we can see that FRRA has a higher RTS enabled frames comparing with

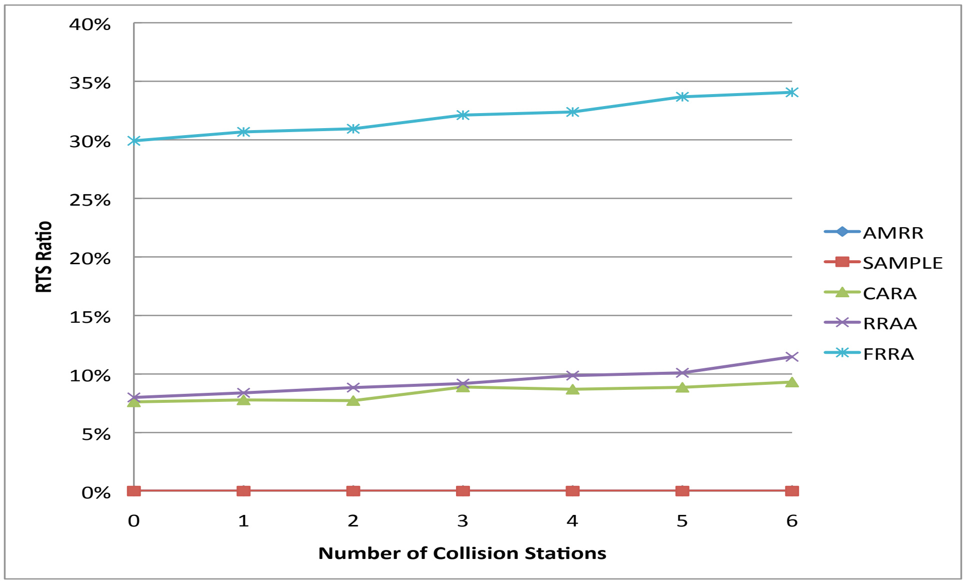

Figure 7.18: FRRA: RTS Ratio for a Static Station in a Collision Dominated Network by Using a TCP Connection

other schemes, which is a good indicator that FRRA is fully aware the collision environment; 3) Through the exchange of RTS/CTS frames, the bandwidth is reserved for the data frame and therefore resulting in a low probability of colliding with other frames. This causes FRRA to have a low failure rate (shown in Figure 7.17) and prevents the TCP from entering the slow start process.

Next, let's look at how FRRA and the other rate adaptation schemes perform in a UDP traffic. Figure 7.19 shows the average goodput as the collision stations increase from 0 to 6. From this figure, we can see all these algorithms perform close to each other.

In order to analyze these schemes' performance, we have listed the communication statistics in different figures. The total transmitted frames for this experiment is shown in Figure 7.20. From this figure, the first thing one might notice is the dramatic decrease of total transmitted frames. This is because as the collision stations increase, there are more stations sharing the channel. Besides the decrease of frames, if we compare it with the total frames transmitted by a TCP traffic in the same environment (shown in Figure 7.16), we will find that there are much more frames transmitted by a UDP traffic than a TCP traffic. This is because UDP does not have congestion control mechanism and has no congestion window. Therefore, it will not enter the slow start process as a TCP traffic would do. Note that as shown in Figure 7.20, FRRA has transmitted fewer frames compared to the

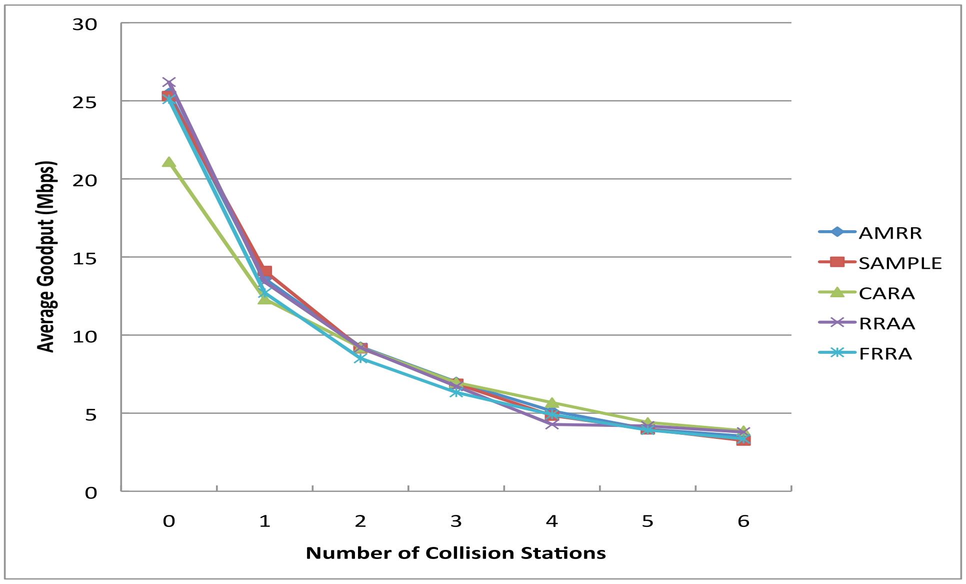

Figure 7.19: FRRA: UDP Goodput for a Static Station in a Collision Dominated Network

other rate adaptation schemes. However, it still gets the similar goodput due to the higher delivery ratio, as shown in Figure 7.21. SampleRate, on the other hand, has the lowest delivery ratio, which means SampleRate is not efficient in this scenario.

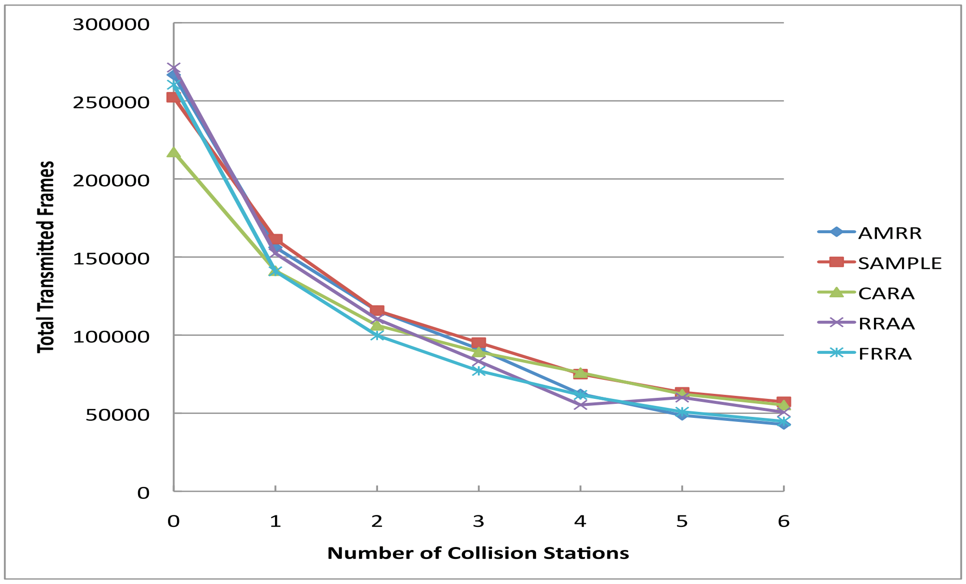

Figure 7.20: FRRA: Total Transmitted Frames for a Static Station in a Collision Dominated Network by Using a UDP Connection

Next, let's look at the RTS ratio of these rate adaptation schemes. Figure 7.22 shows the percentage of the RTS enabled frames. As we can see from the figure, the RTS ratio for a UDP

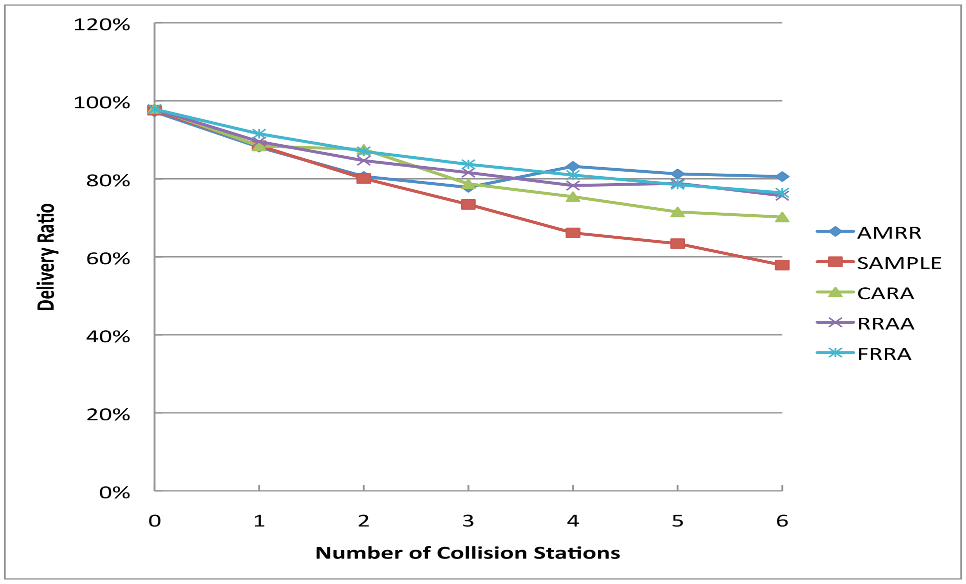

Figure 7.21: FRRA: Delivery Ratio for a Static Station in a Collision Dominated Network by Using a UDP Connection

traffic in a collision dominated network is quite different from a TCP traffic shown in Figure 7.18. Especially FRRA, the RTS ratio for a UDP traffic when there are six collision stations is over 50% comparing with the 30% in a TCP traffic under same environment. This is because a UDP traffic generates more frames and creates more collisions comparing with a TCP traffic under the same circumstance. From Figure 7.22, we can see that FRRA is fully aware the increment of collisions and uses more RTS to minimize the collision.

To summarize, the similar performance of these rate adaption schemes comes from several ways: 1) The experiment is conducted at Location 1. At this location, the signal level is very high. Almost all the schemes use the highest data rate to transmit the data frames. This makes all the schemes to have similar total transmitted frames as shown in Figure 7.20; 2) A UDP traffic is not concerned with the failure rate; the receiver will receive all the frames that it can process. This is quite different from a TCP traffic which is very sensitive to frame failures. And all the algorithms has similar delivery ratio as shown in Figure 7.21. Note that even though SampleRate has a low delivery ratio, it has a higher total transmitted frames. This makes the final goodput similar to the other algorithms; 3) As the number of collision stations increases, the first generation rate adaptation schemes will misjudge the failure as channel degradation and decrease the data rate.

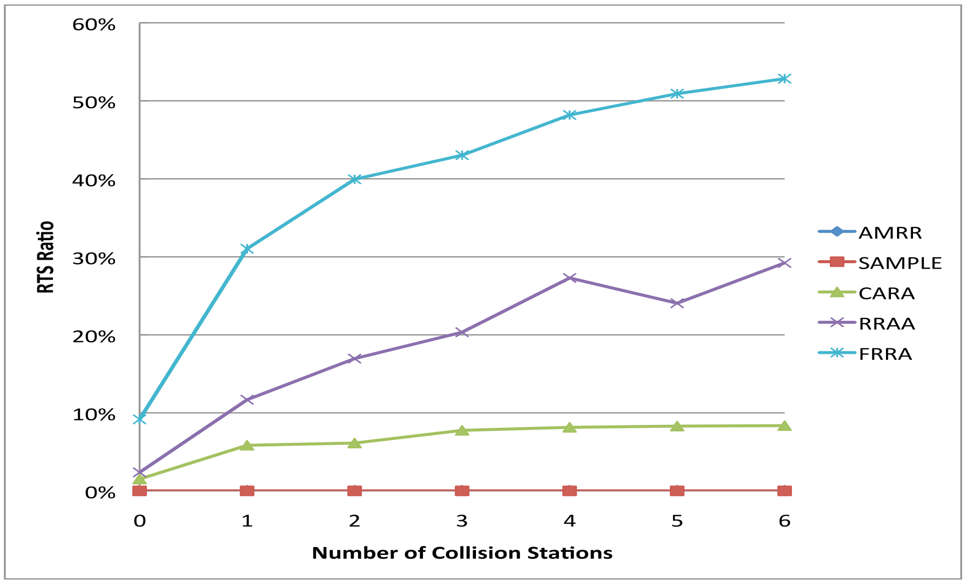

Figure 7.22: FRRA: RTS Ratio for a Static Station in a Collision Dominated Network by Using a UDP Connection

However, from our experiments, even though they decrease the data rate to 11 Mbps, this rate is still high enough to support the transmission since there are many stations competing for the channel slots. Even though FRRA knows the situation and keeps a higher data rate, it still needs to wait for the available channel slots to transmit which makes the higher data rate not very helpful.

In this scenario, FRRA clearly knows the current network is collision dominated and does not decrease the data rate. This can be clearly seen from Figure 7.22, from which we can see that the RTS ratio keeps increasing as the number of collision stations are increasing. RRAA also somewhat knows the situation and increases its RTS Window, but is much smaller compared to FRRA. CARA has almost a constant RTS ratio. AMRR and SampleRate never uses the RTS.

Even though the average goodput for FRRA and the selected rate adaptation schemes are close to each other, there are other factors that are important to UDP traffic. One factor is the delivery ratio, because a low delivery ratio will waste a lot of bandwidth and cause the whole network performance to decrease. As shown in Figure 7.21, FRRA, AMRR and RRAA have the highest delivery ratio whereas SampleRate has the lowest. This means that SampleRate is not very efficient in this scenario.

Figure 7.23 shows the jitter for different rate adaptation schemes. We can see from this figure that FRRA has the lowest jitter whereas AMRR has the highest jitter among all the algorithms when there are multiple collision stations. As shown from Figure 7.23, when the number of collision stations is more than three, the jitter for AMRR and SampleRate starts to increase dramatically. This means that AMRR and SampleRate cannot perform smoothly in a heavily congested network. FRRA, on the other hand, can provide much smooth performance in a real-time system.

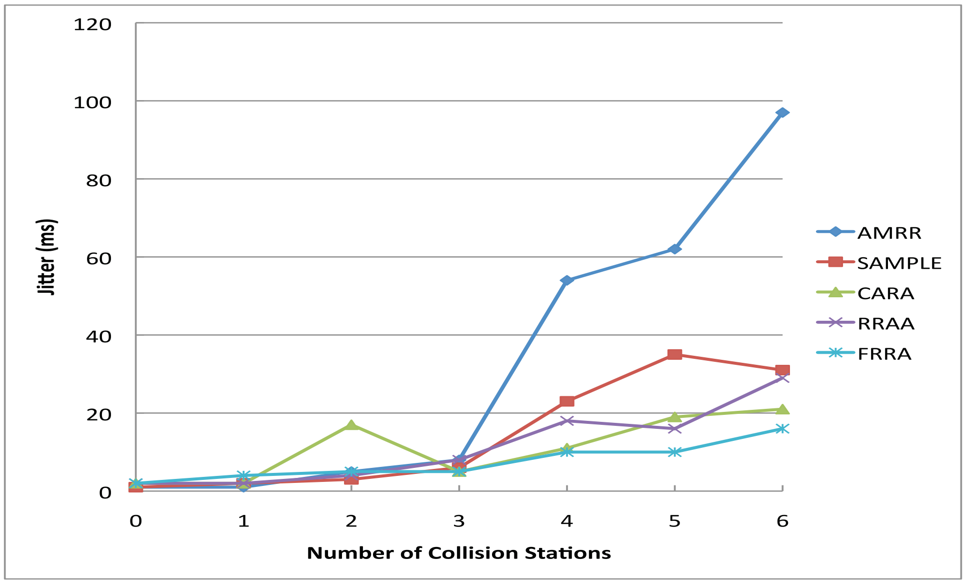

Figure 7.23: FRRA: Jitter for a Static Staion in a Collision Dominated Network by Using a UDP Connection

Static Station in a Mixed Environment: In this scenario, both channel fading and collision may cause frame failures. Figure 7.24 shows the average TCP throughput for FRRA and the selected rate adaptation schemes.

From Figure 7.24, we can see that FRRA performs the best among all the rate adaptation schemes in this scenario. It has a throughput improvement of more than 13% over CARA and RRAA, a 100% improvement over SampleRate and more than 300% improvement over AMRR. This figure also demonstrates the main difference between the first generation rate adaptation schemes and the second generation schemes. The first generation rate adaptation schemes do not differentiate the frame losses and unnecessarily decrease the data rate which further decrease the performance. As shown from Figure 7.25, the second generation rate adaptation schemes

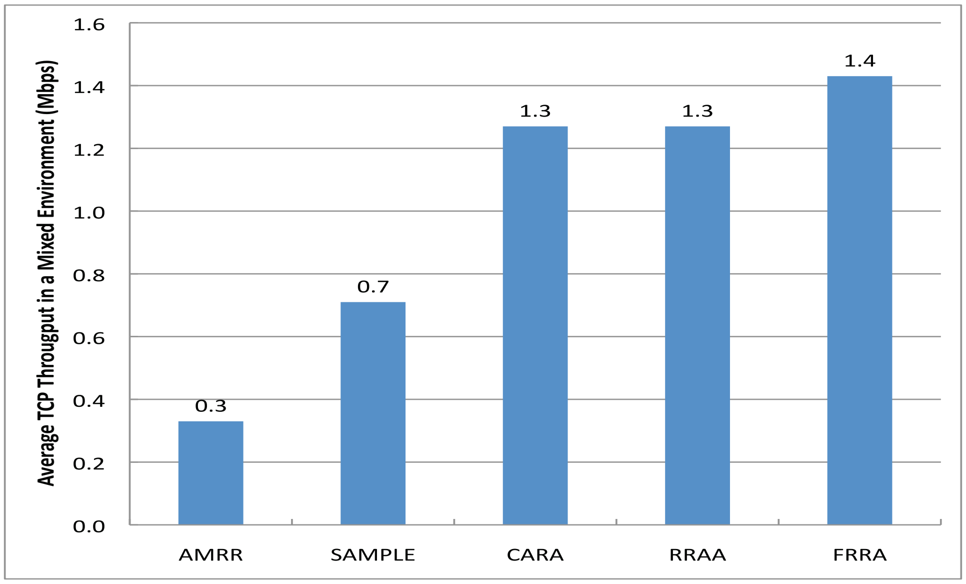

Figure 7.24: FRRA: TCP Throughput for a Static Station in a Mixed Environment

transmitted more frames compared to the first generation schemes. From Figure 7.25, we can see that AMRR has very few frames transmitted because it misjudged the cause of frame losses and decreased the data rate to a very low level.

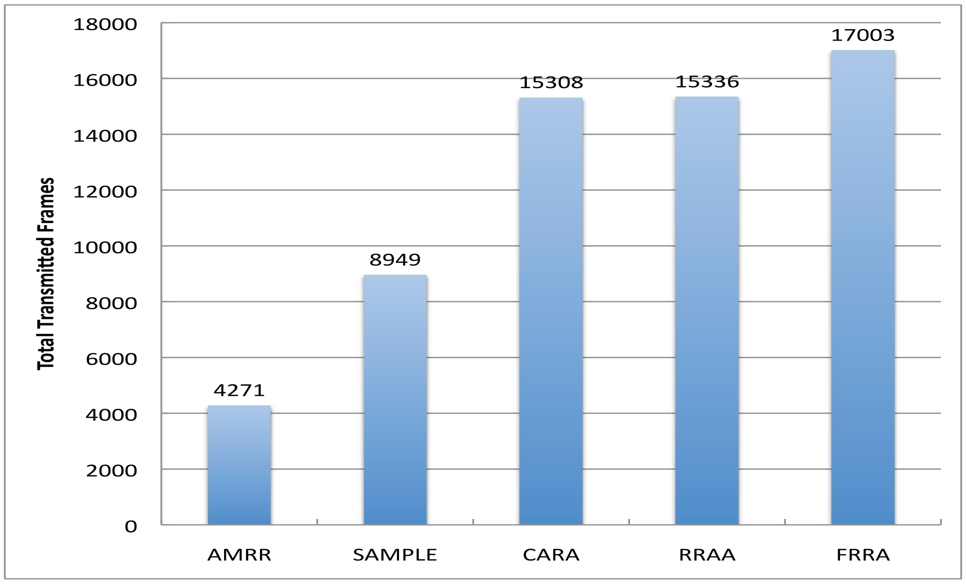

Figure 7.25: FRRA: Total Transmitted Frames for a Static Station in a Mixed Environment by Using a TCP Connection

Figure 7.26 shows the failure rate for different rate adaptation schemes. Note that FRRA does not have the lowest failure rate, this is because it uses a higher data rate to transmit data and transmitted more frames in the experiment. RRAA, on the other hand, provides the lowest failure rate, however, it transmits fewer frames compared to FRRA. Figure 7.24 proves that FRRA has the highest throughput among all the algorithms.

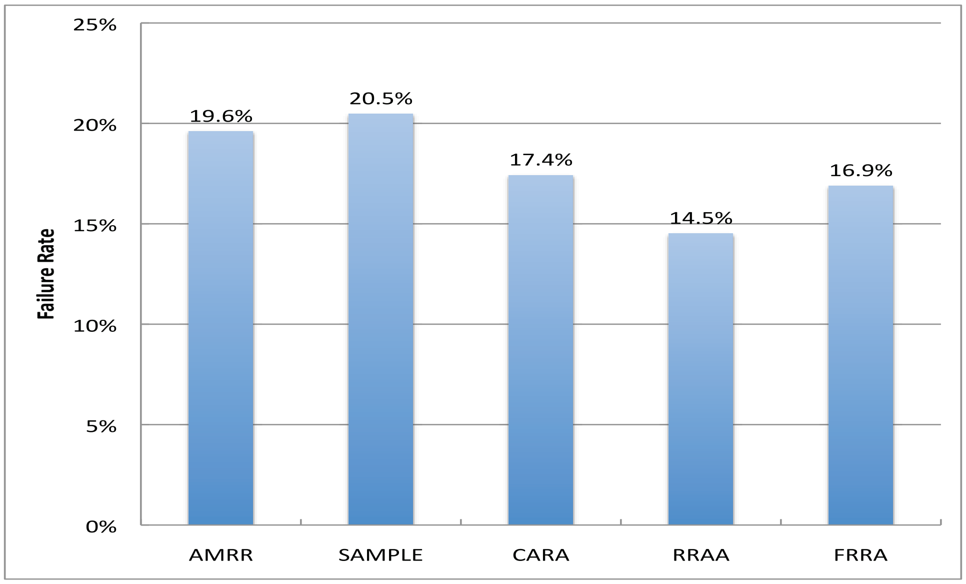

Figure 7.26: FRRA: Failure Rate for a Static Station in a Mixed Environment by Using a TCP Connection

To see how the second generation rate adaptation schemes handle the collision, Figure 7.27 demonstrates the RTS ratio. From this figure, we can see that the second generation schemes use RTS to minimize collision. FRRA is fully aware the existence of collision stations and has a higher RTS ratio compared to CARA and RRAA.

Next, let's see how a UDP traffic performs with different rate adaptation schemes in this scenario. Figure 7.28 shows the average goodput for these schemes.

From Figure 7.28, we can see that FRRA has the highest goodput among all the selected schemes. It has more than 50% improvement over CARA and RRAA, 130% improvement over SampleRate, and around 16 times the goodput of AMRR. As we can see from the figure, the second generation rate adaptation schemes perform much better than the first generation schemes. Normally, a UDP traffic will generate more frames than a TCP traffic since it does not have the congestion control mechanism as in TCP. These frames will cause the first generation rate adaptation schemes to quickly decrease the data rate to a very low level since it cannot differentiate the

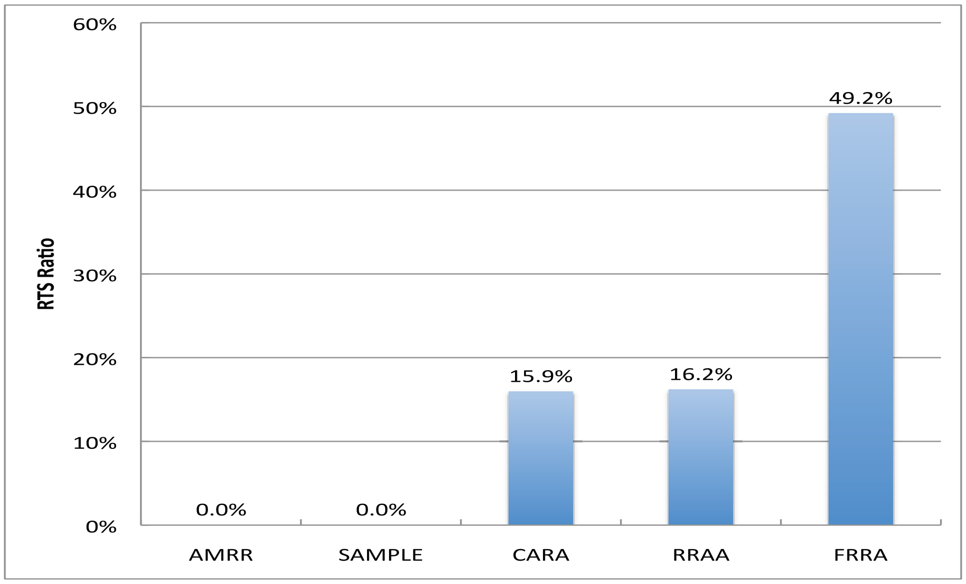

Figure 7.27: FRRA: RTS Ratio for a Static Station in a Mixed Environment by Using a TCP Connection

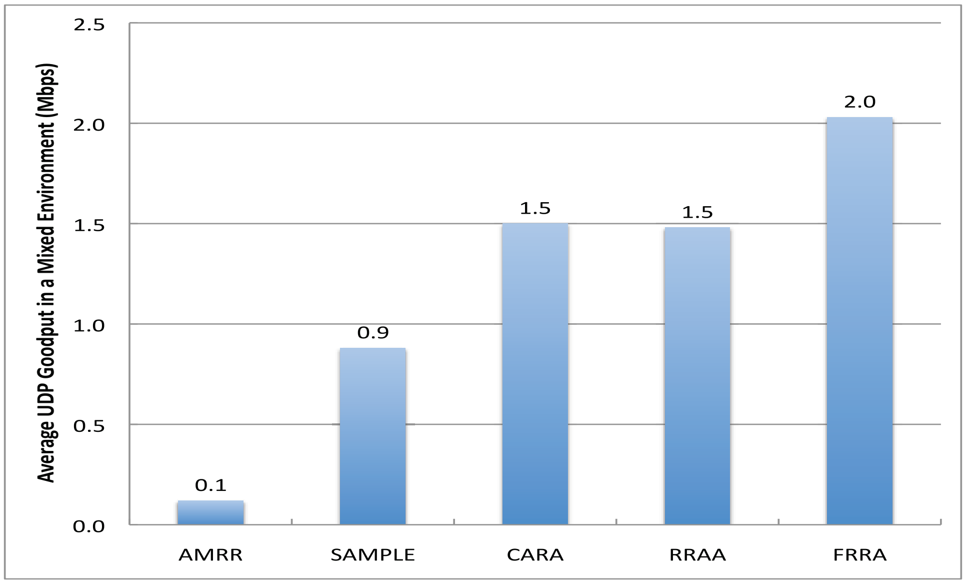

Figure 7.28: FRRA: UDP Goodput for a Static Station in a Mixed Environment

cause of frame losses. Figure 7.29 shows the total transmitted frames in this scenario. From this figure, we can see that AMRR, RRAA and FRRA did not transmit too many frames. However, the reason why they did not transmit a lot of frames is quite different. For AMRR, it treats all the frame losses as channel degradation and keeps decreasing the data rate to a very low level. For RRAA and FRRA, they both know that the current environment is a mixed environment, for channel degradation, they should decrease the data rate so that the channel can support the data rate. For collision, they remain at the current rate and would not decrease it. Therefore, they only transmit those frames that are just suitable for the environment.

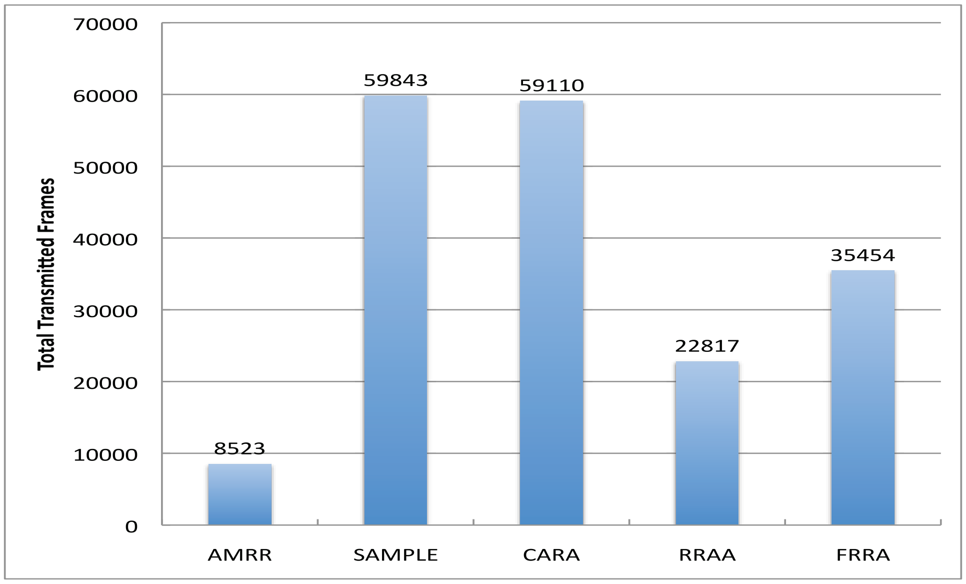

Figure 7.29: FRRA: Total Transmitted Frames for a Static Station in a Mixed Environment by Using a UDP Connection

However, fewer frames transmitted does not necessarily means a low goodput, because we also need to consider the delivery ratio. For AMRR, RRAA, FRRA, even though they have a fewer frames transmitted, they have a high delivery ratio. As shown in Figure 7.30, AMRR has a high delivery ratio because it actually did not transmit many frames. Therefore, even a higher delivery ratio does not provide it a high goodput. For RRAA and FRRA, they both have a very high delivery ratio, therefore, the final goodput is high. For SampleRate and CARA, they produce a better goodput compared to AMRR by generating a lot of frames.

To see how these rate adaptation schemes adapt to the mixed environment, we can check their RTS ratio, which is shown in Figure 7.31. From this figure, we can see that RRAA has a high RTS ratio because RRAA implements a RTS window for a collision dominated network. CARA, though it uses RTS, does not have a RTS window designed for collision. Therefore, its RTS ratio is low compared to RRAA and FRRA. For FRRA, it has the highest RTS ratio among these schemes. Note that FRRA has a higher RTS ratio in a UDP traffic compared to the TCP traffic. This is

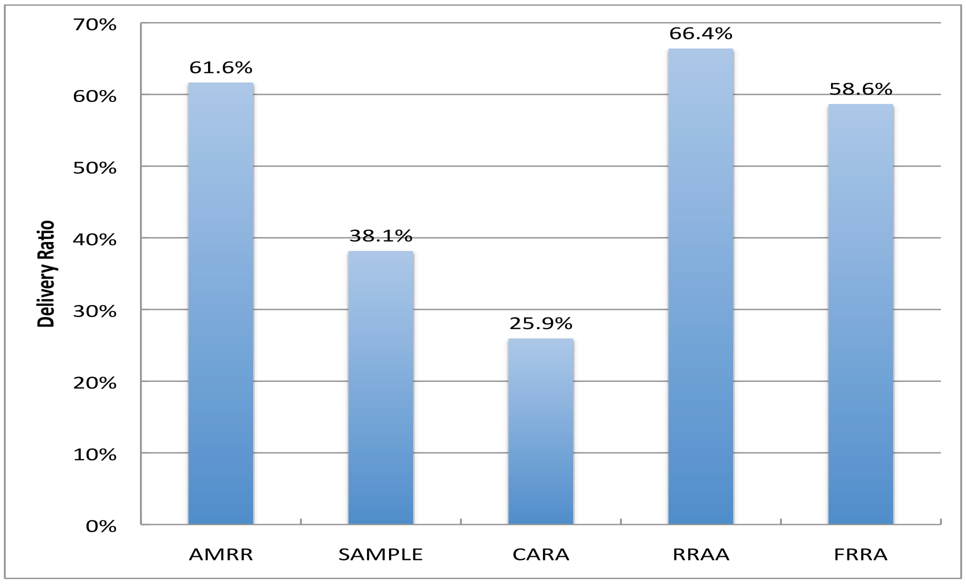

Figure 7.30: FRRA: Delivery Ratio for a Static Station in a Mixed Environment by Using a UDP Connection

because in a UDP traffic, there are more frames transmitted which cause more collisions compared to a TCP traffic. That's why it has a higher RTS ratio in a UDP traffic than in a TCP traffic.

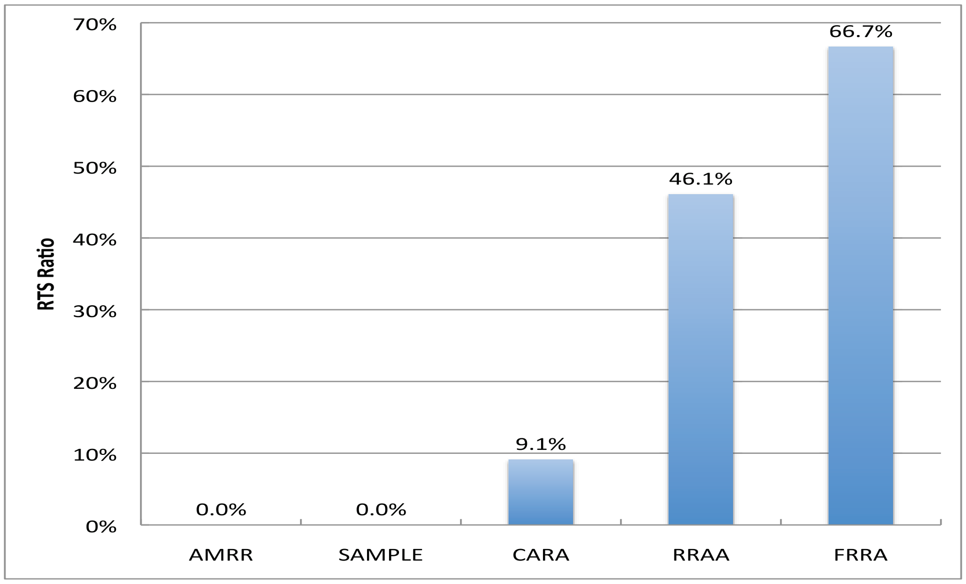

Figure 7.31: FRRA: RTS Ratio for a Static Station in a Mixed Environment by Using a UDP Connection

Finally, let's look at the jitter for these rate adaptation schemes. Figure 7.32 shows the jitter in this scenario. As shown in this figure, AMRR has the highest jitter among all the rate adaptation schemes. This is because AMRR misjudges the environment and decreases its data rate to a very low level. Therefore, the transmission time for each frame is very long causing it to have a much higher jitter. FRRA has the lowest jitter which means it will give the most smooth performance in real-time systems.

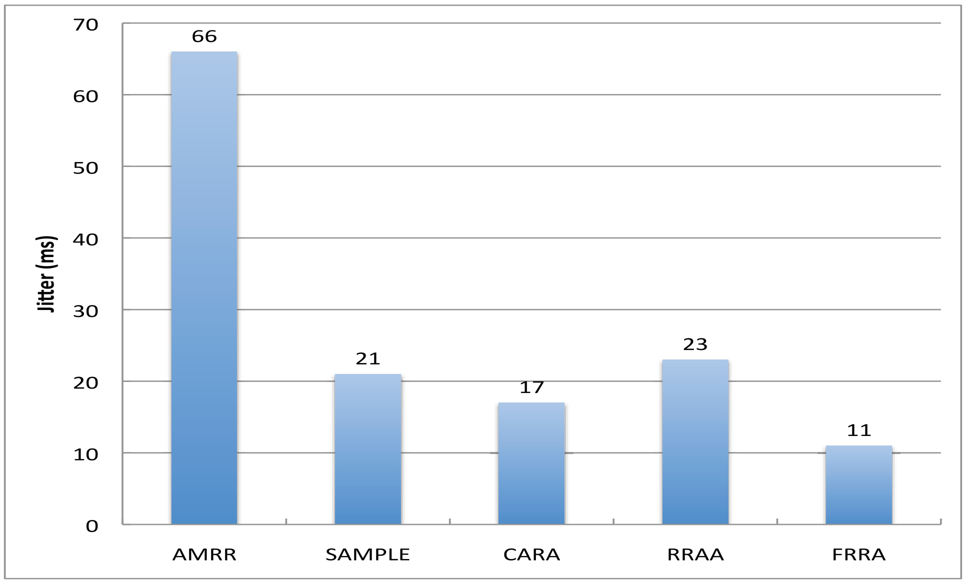

Figure 7.32: FRRA: Jitter for a Static Station in a Mixed Environment by Using a UDP Connection

Mobile Station in a Collision Free Network: To see how the different rate adaptation schemes may adapt to the environmental changes, we have designed these experiments. In these experiments, a user carries a laptop and moves in the lab at a constant speed. The TCP throughput for the mobile station in this scenario is shown in Figure 7.33.

From Figure 7.33, we can see that FRRA has a throughput improvement of 64% over AMRR and a 38% improvement over RRAA. This proves that FRRA can quickly adapt to the channel condition changes. Figure 7.34 shows the total transmitted frames during the experimental period. From Figure 7.34, we can see that FRRA transmitted the most frames, which is one reason why FRRA has a better throughput. Another reason for the better performance of FRRA comes from the low failure rate. Figure 7.35 shows the failure rate for these rate adaptation schemes. We can see that FRRA has the lowest failure rate compared to the other schemes.

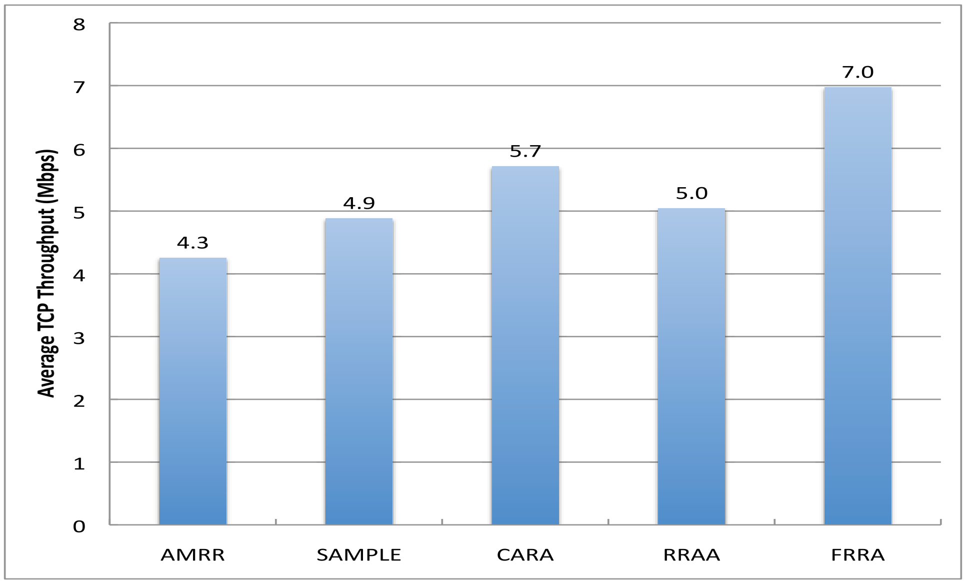

Figure 7.33: FRRA: TCP Throughput for a Mobile Station in a Collision Free Network

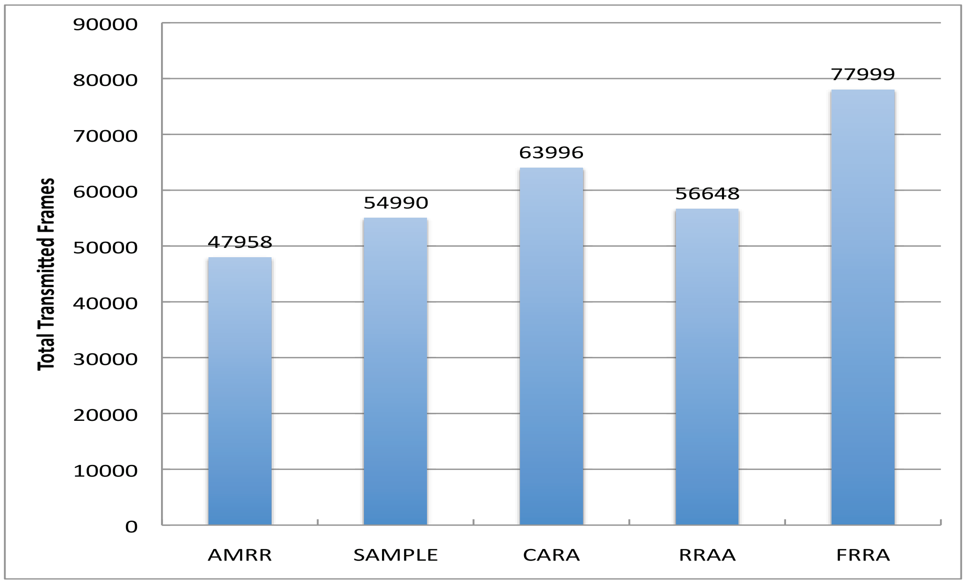

Figure 7.34: FRRA: Total Transmitted Frames for a Mobile Station in a Collision Free Network by Using a TCP Connection

Besides the total transmitted frames and the failure rate, we also listed the RTS ratio in Figure 7.36, similar to the other TCP experiments. FRRA has the highest RTS ratio.

Mobile Station in a Collision Dominated Network: This scenario is similar to the previous one, but we have put six collision stations to make situation more complicated. Figure 7.37 shows the average TCP throughput in such environment. As we can see from this figure, FRRA has

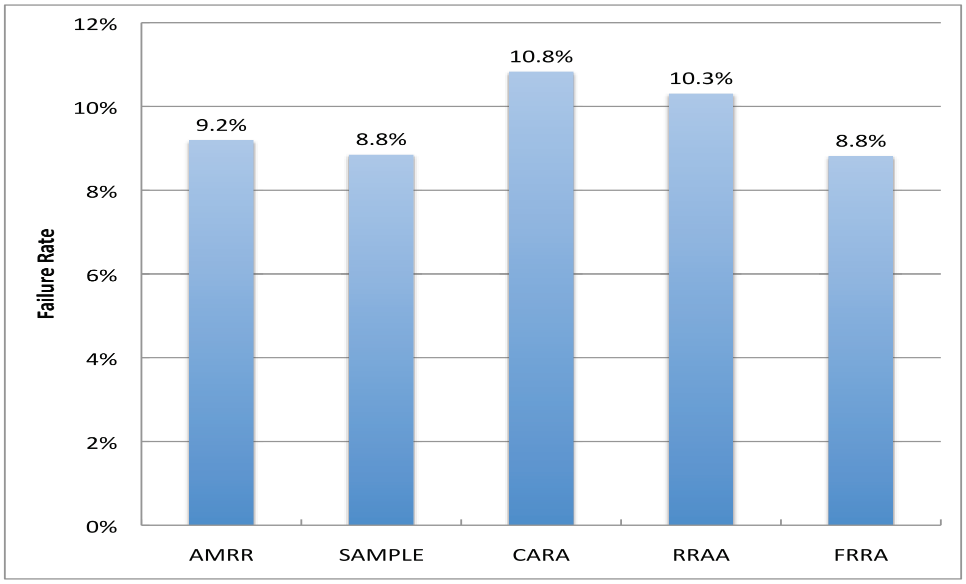

Figure 7.35: FRRA: Failure Rate for a Mobile Station in a Collision Free Network by Using a TCP Connection

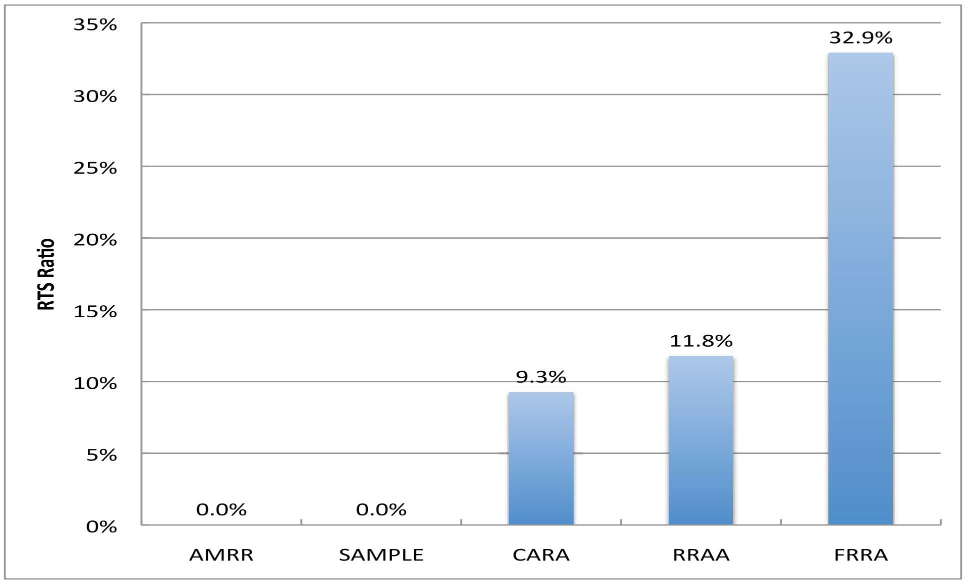

Figure 7.36: FRRA: RTS Ratio for a Mobile Station in a Collision Free Network by Using a TCP Connection

a throughput improvement of more than 114% over the first generation rate adaptation schemes, and more than 42% improvement over the second generation schemes. This shows that FRRA performs extremely well in a collision dominated environment for TCP traffic.

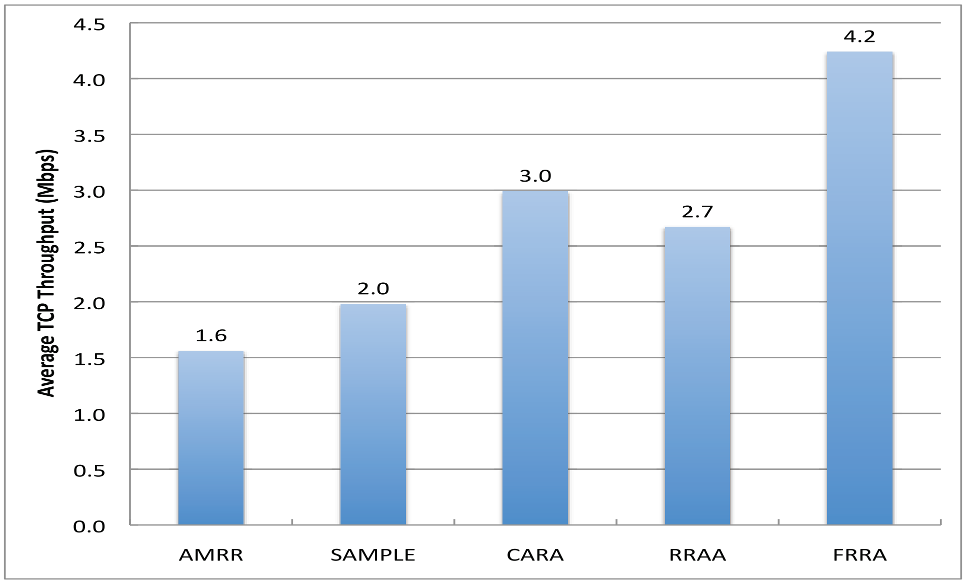

Figure 7.37: FRRA: TCP Throughput for a Mobile Station in a Collision Dominated Network

To analyze why FRRA can achieve such improvement, we need to take a look at the communication statistics. Figure 7.38 shows the total transmitted frames in this experiment. We can see from this figure that the second generation rate adaptation schemes transmitted more frames than the first generation schemes. This is because the second generation schemes differentiate the cause of frame losses and do not decrease the data rate unnecessarily. Therefore, they can transmit more frames in the same amount of time.

Failure rate is also related to the TCP throughput, Figure 7.39 shows the failure rate for these rate adaptation schemes. Note that even though FRRA does not have the lowest failure rate, it still provides the highest throughput because it transmitted more frames by using a more higher data rate.

To see how these rate adaptation schemes adapt the collision situation, we have shown the RTS ratio in Figure 7.40. If we compare this figure with the RTS ratio when there is no collision (Figure 7.35), we can see that for the first generation rate adaptation schemes, their RTS ratio remains zero. For the second generation schemes, CARA's RTS ratio only increased 0.73% and RRAA 's RTS ratio only increased 0.29%, whereas FRRA's RTS ratio has an increment of 9.57%.

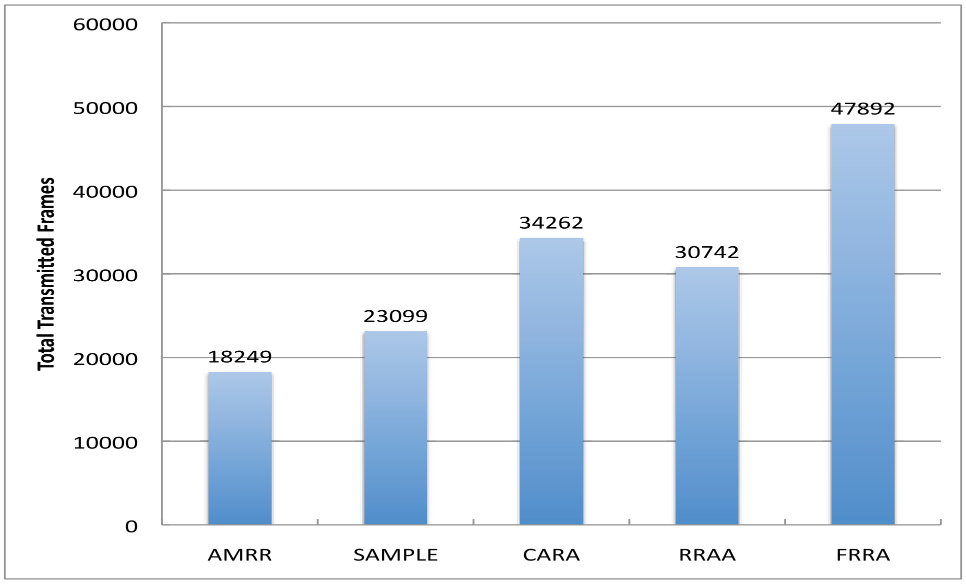

Figure 7.38: FRRA: Total Transmitted Frames for a Mobile Station in a Collision Dominated Network by Using a TCP Connection

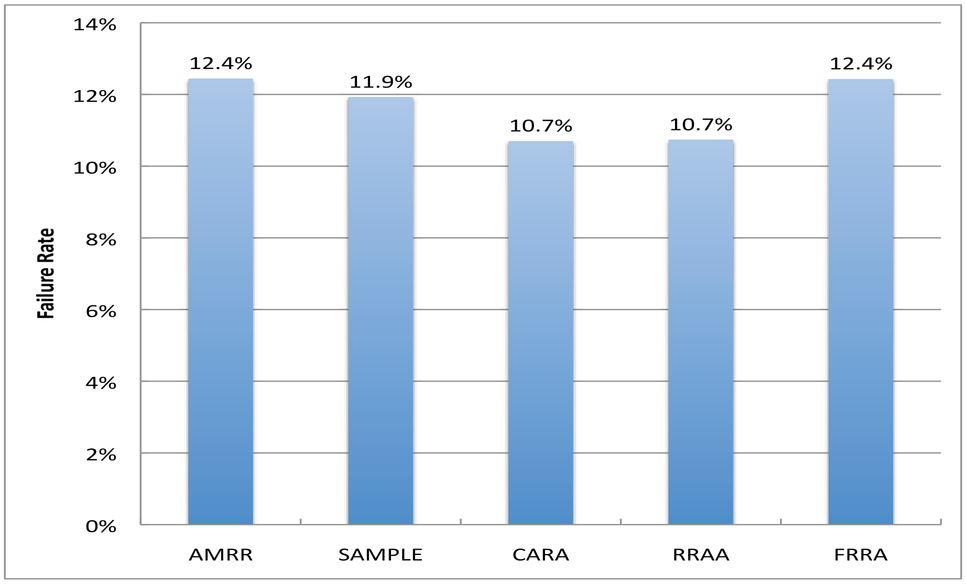

Figure 7.39: FRRA: Failure Rate for a Mobile Station in a Collision Dominated Network by Using a TCP Connection

These statistics clearly shows that FRRA is fully aware the collision situation and therefore greatly increase the exchange of RTS/CTS control frames to minimize the collision.

Next, let's see how UDP traffic performs in this scenario. Figure 7.41 shows the average UDP goodput for different rate adaptation schemes. As shown from this figure, the second generation

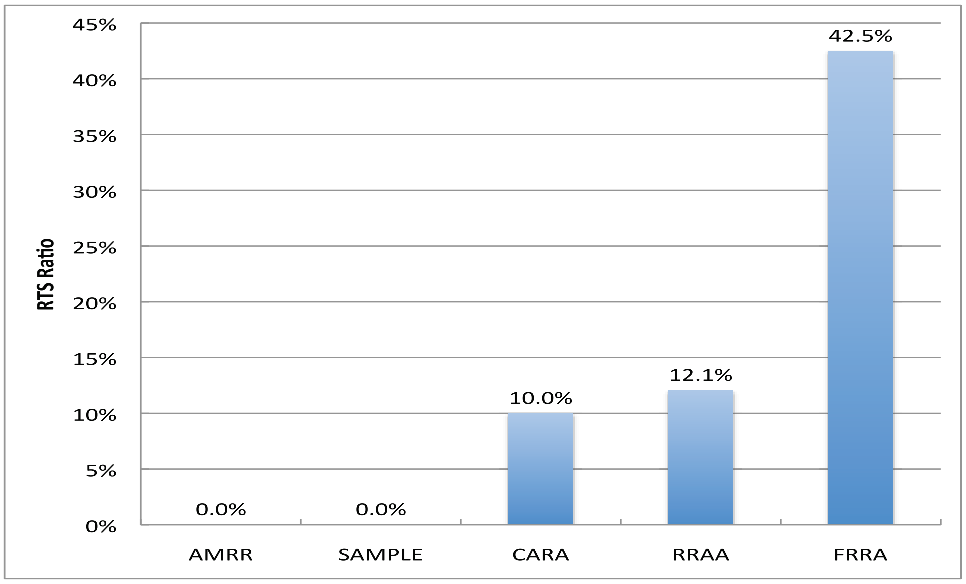

Figure 7.40: FRRA: RTS Ratio for a Mobile Station in a Collision Dominated Network by Using a TCP Connection

rate adaptation schemes perform better than the first generation schemes. AMRR performs the worst due to its inability to differentiate the cause of frame losses. The second generation rate adaptation schemes perform similar in this scenario.

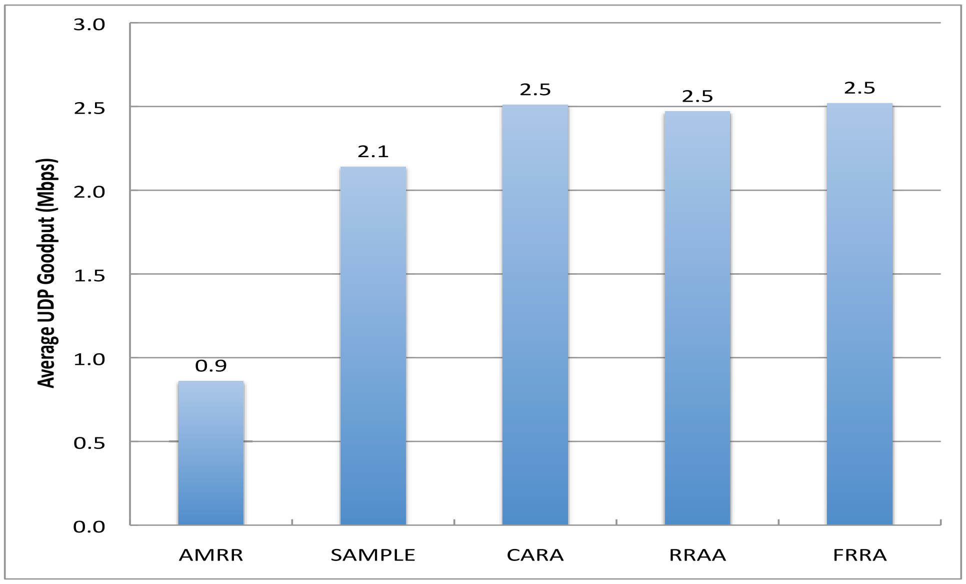

Figure 7.41: FRRA: UDP Goodput for a Mobile Station in a Collision Dominated Network

Figure 7.42 shows the total transmitted frames in this experiment. As we can see from this figure, SampleRate and CARA transmitted the most frames. However, they did not provide the highest goodput because of their low delivery ratio, as shown in Figure 7.43. Similar to a static station in a mixed environment where both channel fading and collision may cause the frame losses, AMRR has a high delivery ratio. However, due to its fewer transmitted frames, AMRR still yields the lowest goodput. FRRA, on the other hand, does not have the highest delivery ratio, but it provides the highest goodput by having a large number of frames transmitted.

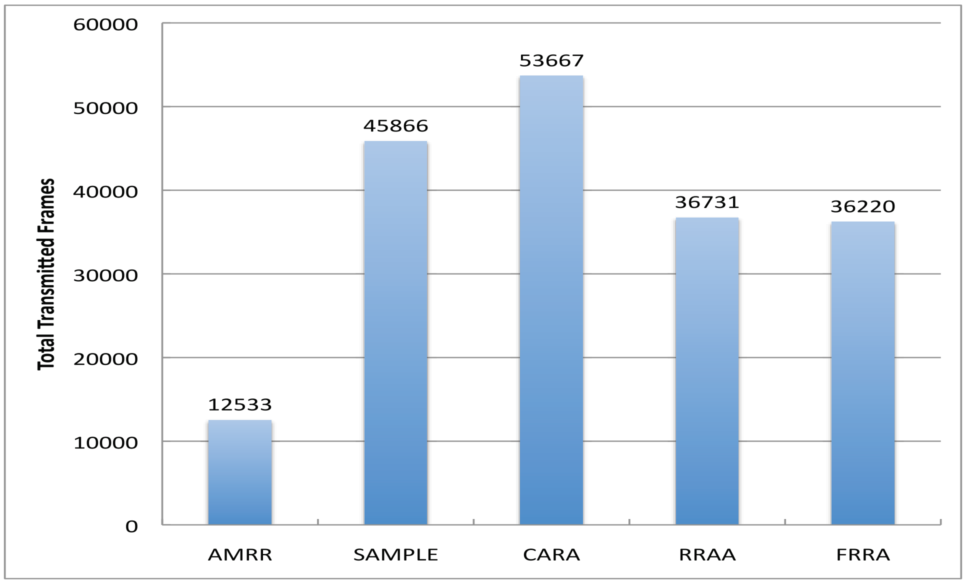

Figure 7.42: FRRA: Total Transmitted Frames for a Mobile Station in a Collision Dominated Network by Using a UDP Connection

To analyze the collision effect, we have illustrated the RTS ratio in Figure 7.44. This figure once again confirms the effectiveness of FRRA in a collision dominated network by having the highest RTS ratio. RRAA also has a high RTS ratio by adjusting its RTS window whereas CARA has a low RTS ratio because it does not implement any RTS window mechanism to adapt to collision.

Finally, the jitter for this scenario is shown in Figure 7.45. SampleRate and FRRA has the smallest jitter whereas AMRR has the largest jitter. The reason why AMRR has the largest jitter is due to its robust transmission rate. FRRA has the smallest jitter because it uses a higher data rate to transmit the data frames.

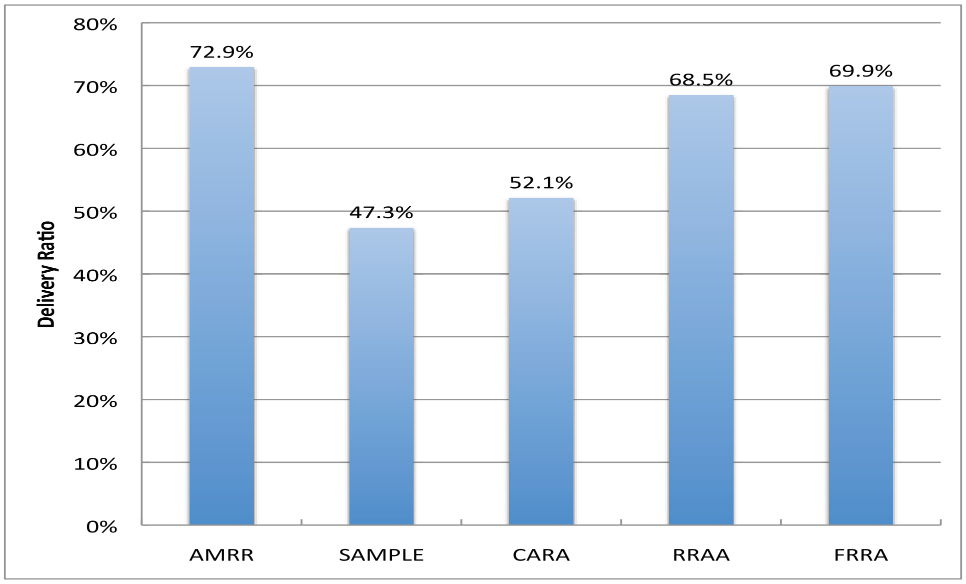

Figure 7.43: FRRA: Delivery Ratio for a Mobile Station in a Collision Dominated Network by Using a UDP Connection

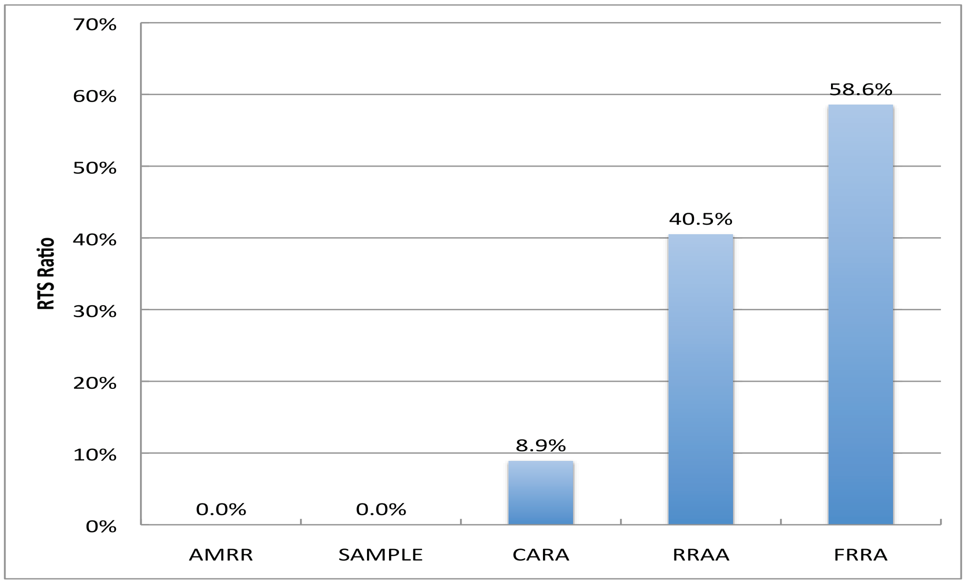

Figure 7.44: FRRA: RTS Ratio for a Mobile Station in a Collision Dominated Network by Using a UDP Connection

Static Station in Campus network: This is the field test for all these rate adaptation schemes. Our campus link condition is shown in Table 7.3. In the field test, we only tested the TCP performance because this access point (tsunami) is mainly used for TCP connections (web browser and

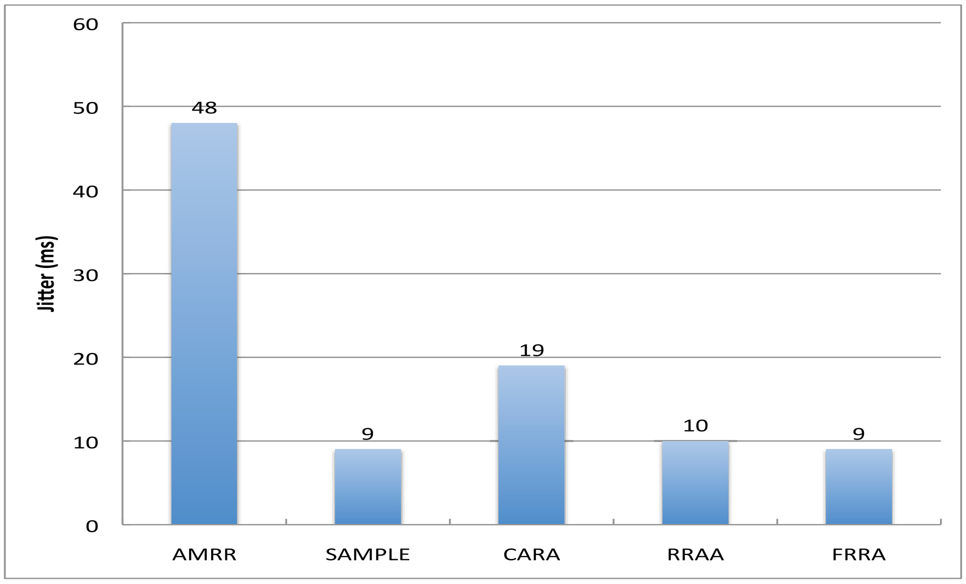

Figure 7.45: FRRA: Jitter for a Mobile Station in a Collision Dominated Network by Using a UDP Connection

email etc). If we use UDP traffics, due to the nature of UDP (No congestion control), the generated frames will congest the network and cause all the other TCP applications to get the minimum throughput.

Our experiments are done in student center at lunch time during weekdays, there are a lot of students using the same AP as us and generating a lot of collisions. We also noticed that the selected AP has a medium signal strength (The signal level is shown in Table 7.3). Therefore, the student center has a similar environment as one of our controlled experiments (A Static Station in a Mixed Environment). For our TCP experiments, the average throughput is shown in Figure 7.46. This figure actually shows a similar results as our controlled experiment as expected with a little variations. These difference comes from several ways: 1) The signal level is different from the controlled experiments. In the field test, the campus AP has a signal level of -48 dBm whereas the controlled experiment has a signal level of -58 dBm at Location 2; 2) In our controlled experiment, we have exactly six collision stations whereas in the field test, the number of collision stations are not fixed due to the mobility of the students; 3) The campus might be using a different model of AP from our controlled experiment; 4) The students might be using different wireless cards and drivers from our controlled experiment. Despite these differences, in this field test, FRRA has the highest average TCP throughput among all the rate adaptation schemes. It has more than 65% throughput improvement over the first generation schemes and more than 26% improvement than the second best algorithm RRAA.

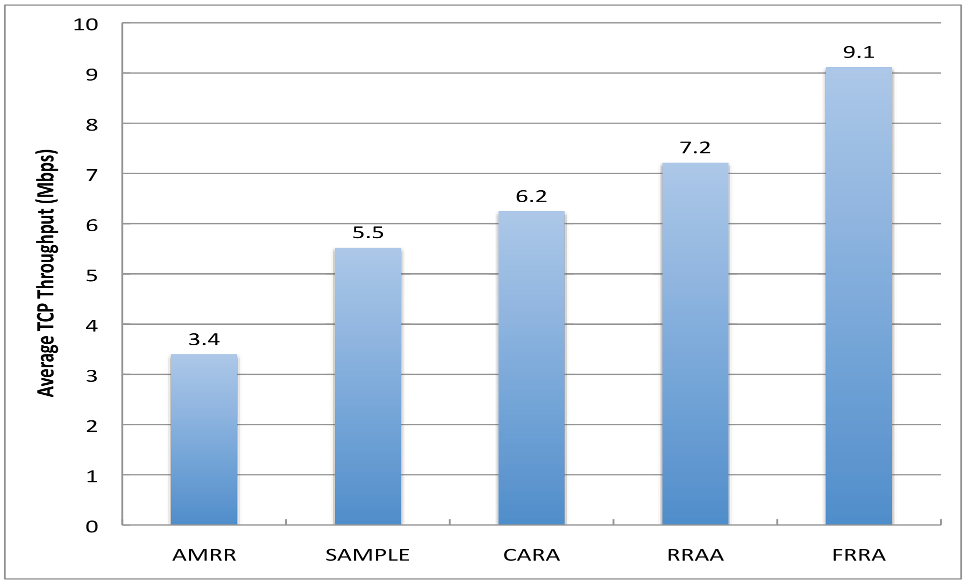

Figure 7.46: FRRA: TCP Throughput for a Static Station in Campus Network

To analyze the performance of these rate adaptation schemes, we have to look at their communication statistics: total transmitted frames, failure rate and RTS ratio. Let's first look at their total transmitted frames in the experiment. Figure 7.47 shows this statistics. As we can see from this figure, FRRA has transmitted the most frames while AMRR transmitted the least. This is obviously one reason why FRRA has the best performance.

Besides the total transmitted frames, another very important factor that affects the throughput of a TCP traffic is its failure rate, the reason has been explained in the previous chapter. Figure 7.48 shows the failure rate in this field test. From the figure, we can see that FRRA has the lowest failure rate while AMRR has the highest. This is the second reason why FRRA has the highest throughput whereas AMRR has the lowest.

To understand why FRRA has the lowest failure rate, we need to understand the environment. The student center is similar to a mixed environment where both channel fading and collision may cause frame failure. To minimize collision, FRRA turned on the RTS to reserve the bandwidth and

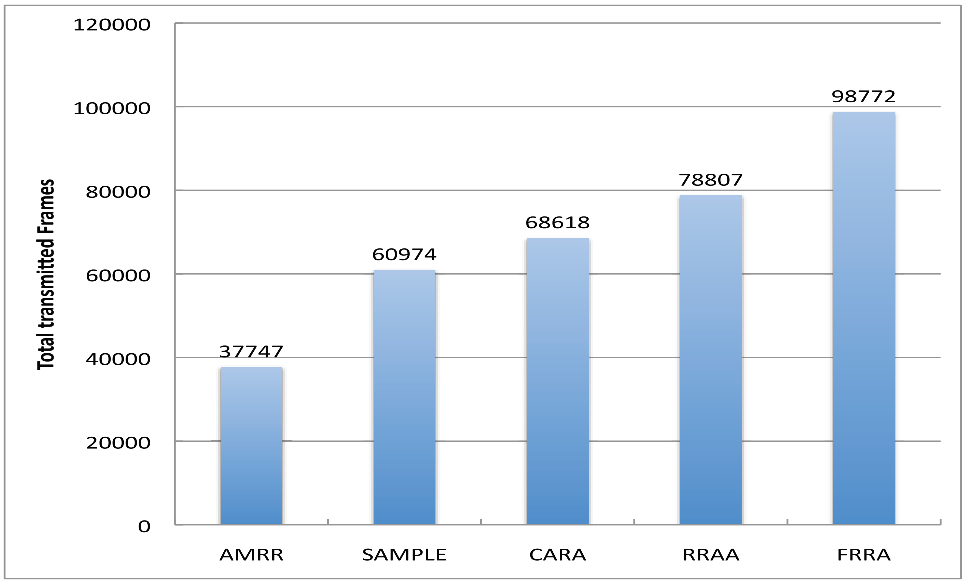

Figure 7.47: FRRA: Total transmitted frames for a Static Station in Campus Network by Using a TCP Connection

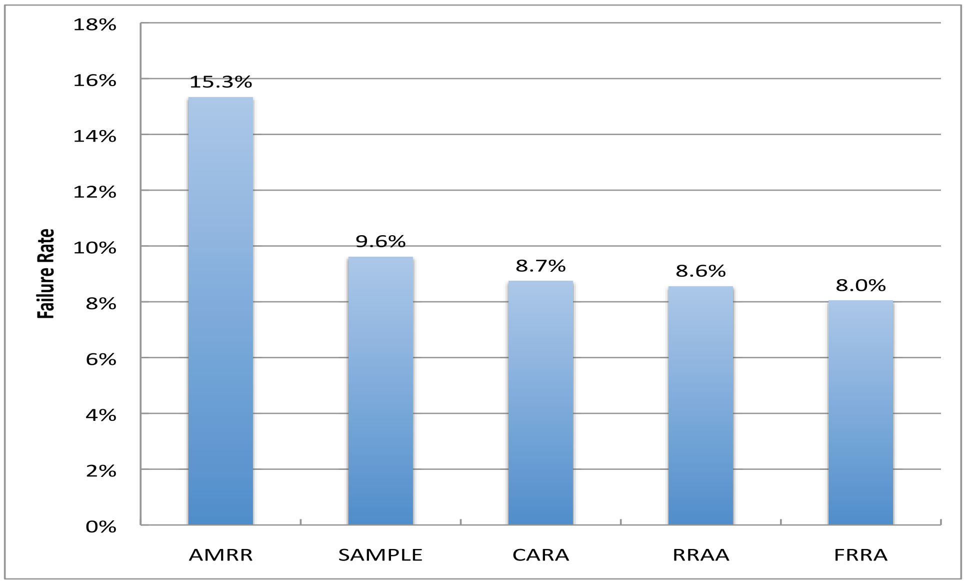

Figure 7.48: FRRA: Failure Rate for a Static Station in Campus Network by Using a TCP Connection

therefore greatly reduce the possibility that a frame may collide. Figure 7.49 shows the RTS ratio for all the selected schemes. From this figure, we can see that FRRA has a RTS ratio around 33%, which clearly indicates that there are some collision stations. The two other second generation rate adaptation schemes has a RTS ratio around 8% which is relatively lower than FRRA. However, we have to point out that even though FRRA has a much higher RTS ratio, it does not means FRRA introduces too much overhead. The reason is because most of the RTS in FRRA is transmitted by using a higher transmission rate (same as the failed data rate) whereas the RTS in CARA and RRAA are transmitted at the lowest data rate (1 Mbps). Therefore, considering the transmission speed of these RTS, FRRA actually does not waste too much time in transmitting the RTS. By using the RTS window, FRRA reduces the probability of collision and therefore reduces the failure rate. This is the third reason why FRRA may achieve the highest throughput.

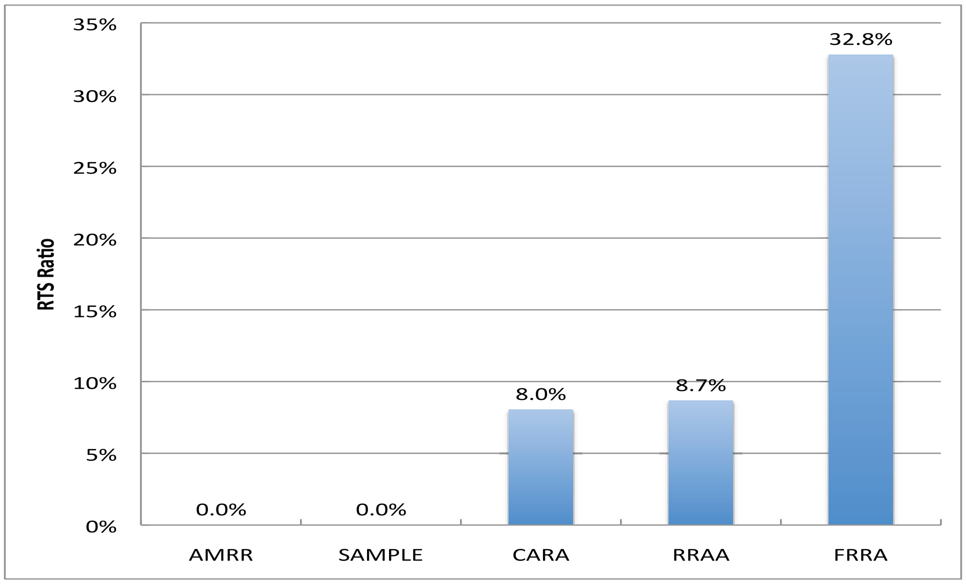

Figure 7.49: FRRA: RTS Ratio for a Static Station in Campus Network by Using a TCP Connection

Chapter 8

Conclusion

This dissertation analyzes the existing representative rate adaptation schemes and divides them into two generations. The first generation schemes do not differentiate the frame losses and consider all the losses to be caused by channel degradation and therefore perform poorly in an environment where there are a lot of collisions. The second generation schemes differentiate the frame losses and perform much better than the first generation schemes in a collision dominated network. However, most of the second generation rate adaptation schemes may differentiate the frame losses caused by channel degradation, they cannot precisely differentiate the frame losses caused by collision.

This dissertation first proposes several guidelines as how to design an effective rate adaptation scheme. Then it introduces a practical rate adaptation scheme called Advanced Rate Adaptation Algorithm (ARA). The key idea of ARA is to transmit the RTS control frame at the same data rate as the previously failed data frame to detect collision. ARA is specially designed for TCP, therefore, only TCP traffics are tested in our experiments.

Based upon ARA, we propose a new rate adaptation scheme called Fast Recovery Rate Adaptation Algorithm (FRRA). It shares the same idea as ARA: Use the RTS control frame as the probe frame.

To improve ARA, we have done several experiments to collect the statistics of these experiment results. According to our study, when the channel quality is poor, both the highest data rate and the lowest data rate do not provide a good throughput for TCP traffic. The highest data rate is not supported by the poor signal and the lowest data rate can only provide a higher deliver ratio but only transmit limited frames in a given amount of time. Therefore, there is a trade off between the number of transmitted frames and the delivery ratio. Besides the problem of fewer transmitted frames, using a lower data rate also makes the rate adaptation schemes harder to fall back to a higher data rate when the channel quality improves. The reason is because before the data rate can be increased, certain threshold must be met for these schemes. This makes the increment of data rate not easy, not to mention that the data rate can only be adjusted step by by step for most rate adaptation schemes. Considering the above facts, FRRA limits the lowest transmission rate to 6 Mbps so that it can quickly recover from the poor signal environment should the channel condition improve.

Besides the above considerations, FRRA uses some pre-defined states to describe current environment. One of these states is called COLLISION, within this state, a RTS Window will be used to minimize the collision through the exchange of RTS/CTS control frames. The size of the RTS window is chosen from the performance of different RTS window sizes. Normally, a higher RTS window size may get better throughput in a TCP traffic but yield a poor goodput in a UDP traffic. Considering both the TCP and UDP traffics, we set the initial size of the RTS window to be 5. However, this RTS window can dynamically change as FRRA moves from one state to another state.

In order to evaluate the proposed algorithms, we have designed several scenarios including controlled experiments and field tests. In these scenarios, ARA and FRRA are compared to four other representative rate adaptation schemes.

From the experiment results, we can see that ARA performs very well in both controlled experiments and field test. Especially in the field test, through a campus wireless network, ARA provides a 300% throughput improvement over AMRR and more than 73% throughput improvement over the second generation rate adaptation schemes.

For FRRA, it performs extremely well in most scenarios for both TCP and UDP traffics. It has a TCP throughput improvement of more than 67% over other selected rate adaptation schemes when there is no collision and the signal strength is strong. When the network is heavily congested (six collision stations), FRRA has a throughput improvement of more than 218% compared to the first generation rate adaptation schemes and more than 113% improvement compared to the second generation schemes. In the field test where both channel fading and collision may cause the collision, FRRA provides a 168% throughput improvement over AMRR and more than 26% improvement over the second generation rate adaptation schemes. It also provides more than 22% improvement over other rate adaptation schemes when the station is in mobility.

For UDP traffics, even though the goodput is close when there is only channel fading or collision causing frame failure. In a mixed environment where both channel fading and collision exist, FRRA has a goodput improvement of more than 130% compared to the first generation rate adaptation schemes and an improvement of more than 35% compared to the second generation schemes. Besides the goodput improvement for UDP traffic, FRRA also has the lowest jitter in these scenarios which means FRRA has a more smooth performance in real-time systems.

Chapter 9

Future Work

As we have explained in this dissertation, TCP traffic has quite different characteristics from UDP traffic. TCP has the congestion control mechanism which makes the failure rate a key factor to performance. UDP, on the other hand, is simple and does not implement any congestion control algorithm. Considering the different characteristics of these two types of traffics, an efficient rate adaptation scheme should not only be able to differentiate the cause of frame losses, but also should consider the different types of traffic.

For future design of rate adaptation schemes, traffic type should first be identified before making any adjustments to the data rate. For a TCP connection, the rate adaptation scheme should concentrate on the failure rate in order to avoid the TCP slow start process. From our experiments, using some RTS/CTS exchanges actually help lower the failure rate for TCP connections. For UDP traffic, using too many RTS/CTS exchanges might be a waste of bandwidth when the network is not heavily congested. Since UDP traffic does not have the slow start process as in TCP traffic, even when some frame are corrupted, there is no congestion window to control the traffic flow.

Another discovery through the experiments involves the high RTS ratio when there is no collision for TCP traffic. As shown in Figure 7.18, FRRA has a RTS ratio of 30% when there is no collision station. Even though FRRA has the highest TCP throughput and most of the RTS are transmitted by using a high data rate instead of the minimum data rate, a RTS ratio of 30% is still high. Further improvement involves how to decrease the probability that a RTS frame is transmitted under this environment.

Besides these problems, another work we can do is to consider the infrastructure of the networks. Currently, these rate adaptation schemes are designed for IEEE 802.11 networks with access points. However, the multiple hops wireless network has gained much attention (for example: sensor networks) recently. The next step is to design an effective rate adaptation scheme for these networks to improve their performance.

Bibliography

[1] Madwifi, “http://madwifi-project.org/.”

[2] IEEE 802.11, “Wireless LAN medium access control (MAC) and physical layer (PHY) specifications,” June 1999.

[3] IEEE 802.11b. “Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: high-speed physical layer (PHY) Extension in the 2.4GHz Band,” September 1999.

[4] IEEE 802.11g. “Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: further higher data rate extension in the 2.4GHz Band,” June 2003.

[5] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris, “Link-level Measurements from an IEEE 802.11b Mesh Network,” SIGCOMM04, August 30-September 3, 2004.

[6] Maya Rodrig, Charles Reis, Ratul Mahajan, David Wetherall, and John Zahorjan, “Measurement-based Characterization of IEEE 802.11 in a Hotspot Setting,” SIGCOMM05 Workshops, August 22-26, 2005.

[7] K. Chebrolu, B. Raman, and S. Sen, “Long-Distance IEEE 802.11b Links: Performance Measurements and Experience,” Mobicom06, 2006.

[8] Giuseppe Bianchi, Fabrizio Formisano, and Domenico Giustiniano, “IEEE 802.11b/g Link Level Measurements for an Outdoor Wireless Campus Network,” Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM’06), 2006.

[9] Michael R. Souryal, Luke Klein-Berndt, Leonard E. Miller, and Nader Moayeri, “Link Assessment in an Indoor IEEE 802.11 Network,” WCNC06, 2006.

[10] Ad Kamerman and Leo Monteban, “WaveLAN-II: a high-performance Wireless LAN for the Unlicensed Band,” Bell Labs Technical Journal, vol.2, no.3, pages 118-133, August 1997.

[11] Gavin Holland, Nitin Vaidya, and Paramvir Bahl, “A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks,” ACM SIGMOBILE, July 2001.

[12] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, “Opportunistic Media Access for Multirate Ad Hoc Networks,” MOBICOM02, September 23-26, 2002.

[13] Javier del Prado Pavon and Sunghyun Choi, “Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement,” ICC, pages 1108-1123, 2003.

[14] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti, “IEEE 802.11 Rate Adaptation: A Practical Approach,” MSWiM’04, pages 126-134, October 2004.

[15] Z. Li, A. Das, A.K. Gupta and S. Nandi, “Full auto rate MAC protocol for wireless ad hoc networks,” IEEE Proceedings on Communication, 3:311-319, June 2005.

[16] John C. Bicket, “Bit-rate Selection in Wireless Networks,” Masters thesis, Massachusetts Institute of Technology, 2005.

[17] Qixiang Pang, Soung C. Liew, and Victor C. M. Leung, “Design of an Effective Loss-Distinguishable MAC Protocol for IEEE 802.11 WLAN,” IEEE Communications Letters, 9:781-783, 2005.

[18] Qixiang Pang, Victor C.M. Leung, and Soung C. Liew, “A Rate Adaptation Algorithm for IEEE 802.11 WLANs Based on MAC-Layer Loss Differentiation,” IEEE BROADNETS 2005 - Broadband Wireless Networking Symposium, pages 709-717, 2005.

[19] Jongseok Kim, Seongkwan Kim, Sunghyun Choi, and Daji Qiao, “CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs,” IEEE INFOCOM’06, April 2006.

[20] Starsky H.Y. Wong, Hao Yang, Songwu Lu and Vaduvur Bharghavan, “Robust Rate Adaptation for IEEE 802.11 Wireless Networks,” MobiCom06, September 23-26, 2006.

[21] Giuseppe Bianchi, “Performance Analysis of the IEEE 802.11 Distributed Coordination Function,” IEEE Journal on Selected Area in Communications, vol.18, no.3, pp. 535-547, March 2000.

[22] Iperf, “http://sourceforge.net/projects/iperf/.”

[23] National Laboratory for Applied Network Research, “http://www.nlanr.net/.”