Smart Sender: A Practical Rate Adaptation Algorithm for Multirate IEEE 802.11 WLANs

Qiuyan Xia, *Student Member, IEEE,* and Mounir Hamdi, *Senior Member, IEEE*

***Abstract*— Wireless Local Area Networks (WLANs) have become increasingly popular due to the recent availability of affordable devices providing multirate capabilities. Under time-varying wireless channels, a device needs to tune its transmission rate dynamically for more efficient utilization of the physical link. Hence, rate adaptation mechanism, which is intentionally unspecified by the IEEE 802.11 standards, is critical to the system performance. In this paper, we propose a practical rate adaptation algorithm, *Smart Sender*, which utilizes both statistics and the received signal strength indicator (RSSI) of ACK packets to determine the transmission rate that maximizes the throughput. We implement our algorithm in commercial WLAN products and carry out extensive experiments for performance evaluation. The results demonstrate that using both statistics and RSSI of ACKs greatly improves system throughput and responsiveness under various wireless environments.**

***Index Terms*— rate adaptation, IEEE 802.11, WLAN, multirate, RSSI.**

I. Introduction

IN RECENT years, Wireless Local Area Network (WLAN) technology has been evolving at a rapid pace. Most of the commercial WLAN products are based on the IEEE 802.11 standard [1], which specifies the Medium Access Control (MAC) and Physical (PHY) layers for WLAN systems. Two medium access mechanisms are defined in 802.11: the Distributed Coordination Function (DCF) is a mandatory, contention-based access protocol; the Point Coordination Function (PCF) is a priority-based, contention-free protocol. In the basic access mode of the DCF, only DATA-ACK is exchanged between the sender and the receiver. The DCF also defines an optional access mode to avoid collisions from hidden nodes, which requires the sender and receiver to exchange short RTS (Request To Send) and CTS (Clear To Send) control frames before the actual data transmission.

There are currently three PHY layer extensions, 802.11a/b/g [2]–[4], all providing multiple data rates at the physical layer by using different modulation and coding schemes. For example, 802.11b supports four data rates of 1, 2, 5.5 and 11 Mbps at 2.4 GHz band; 802.11a defines eight data rates ranging from 6 Mbps up to 54 Mbps at 5 GHz band; 802.11g, operating at 2.4 GHz band, is backwards-compatible with 802.11b and offers twelve rates up to the maximum 54 Mbps. Typically, higher data rates are achieved by more efficient modulation schemes. A high level modulation can be used when the channel Signal-to-Noise Ratio (SNR) is sufficiently high such that the received signal can be properly decoded. Therefore, a tradeoff emerges between the data rate and the Bit Error Rate (BER) under a given SNR.

With the multirate capability, it is desirable to always transmit data at the highest possible rate given current channel conditions. However, in wireless systems, the radio propagation environments vary over time and space due to factors such as signal attenuation and fading, motion of objects, interference, causing variations in the received SNR. As a result, there is no single rate that can be optimal under all scenarios. As the multirate enhancements are within the PHY layer protocols, it is the MAC layer mechanisms that are required to exploit this capability. Intuitively, when the channel quality is poor, lower transmission rate is preferred to maintain a smaller BER; otherwise, the transmission rate should be higher to improve the throughput. The problem of dynamically selecting an appropriate transmission rate out of multiple available data rates is referred to as *Rate Adaptation*. Its objective is to maximize system throughput by tuning the user-available MAC layer parameter, i.e., data rate, to current channel conditions.

The implementation of a rate adaptation algorithm is intentionally left open to the vendors. The 802.11 standards only specify which rate set is allowed to use for certain types of data frames, but not when and how to switch among different rates. In addition, no signaling mechanism is available for the receiver to notify the sender about the channel conditions. Typically, in 802.11 WLANs, the data rate to be used for a particular transmission is solely determined by the sender. The actual transmission rate is encoded in the PLCP (Physical Layer Convergence Procedure) header, which is transmitted at a fixed rate (basic rate) supported by all stations in the WLAN system. After receiving the header, the receiver can switch to the rate indicated in the PLCP header to receive the remaining frame. As presented above, the higher the transmission rate is, the larger SNR is required at the receiver to maintain the same communication quality. To determine the best transmission rate at a given time, the sender needs to know the Channel State Information (CSI) in advance, i.e., SNR at the receiver side. Since SNR is time-varying, its accurate value is not available to the sender in reality. The sender has to determine the transmission rate based on limited, indirect CSI feedback such as ACK, FER and retry count.

In this paper, we focus on rate adaptation for 802.11a WLANs based on the DCF, as the PCF is rarely used in

Manuscript received December 10, 2006; revised May 14, 2007; accepted June 12, 2007. The associate editor coordinating the review of this paper and approving it for publication was S. Aissa. This work was supported by a grant from Research Grants Council (RGC) under contract HKUST6260/04E.

Q. Xia and M. Hamdi are with the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China (e-mail: {xiaqy, hamdi}@cse.ust.hk).

Digital Object Identifier 10.1109/TWC.2008.061047.

commercial WLAN devices. Nevertheless, our algorithm can be easily extended to other physical layers such as the 802.11g PHY. Our objective is to design a rate adaptation algorithm which works well in a variety of channel conditions; besides, no protocol modification is required so that it can be easily deployed with current products. Specifically, we investigate a new MAC layer mechanism, *Smart Sender*, which performs the following tasks efficiently:

- It dynamically monitors and adapts to changes (transient or sustained) in the link quality by tuning the transmission bit rate.
- It maintains both long-term and short-term statistics, which are carefully selected to better reflect the network configurations and channel conditions.
- It maximizes channel capacity by opportunistically transmitting at the highest feasible rate analogous with "water-filling" concept in the information theory.
- To improve system responsiveness, it uses the Received Signal Strength Indicator (RSSI) of ACKs to predict the dynamics of the receiver-perceived link conditions for rate adaptation at the sender.

We demonstrate that the combination of these methods can significantly improve system performance.

The rest of the paper is organized as follows. In Section II, we briefly review the related work. We describe the experiment methodology in Section III. The proposed *Smart Sender* rate adaptation algorithm is presented in Section IV. In Section V, we introduce practical implementation of our algorithm on commercial wireless adapters. Section VI gives extensive performance evaluations of our algorithm compared with other state-of-the-art algorithms. Finally, this paper concludes with Section VII.

II. Related Work

In this section, we briefly review rate adaptation algorithms grouped into two categories, statistic based and signal measurement based, according to the type of CSI employed. Description of practical algorithms are followed as they will be compared with our scheme for performance evaluation.

A. Categories of Rate Adaptation Algorithms

Based on the type of CSI used for channel quality estimation, rate adaptation algorithms can be roughly divided into two categories: *statistic based* schemes (ARF [5], Dynamic ST [6], ONOE [7], AARF/AMRR [8], SampleRate [9]) and *signal measurement based* schemes (RBAR [10], Goodput analysis [11], OAR [12], RSS measurement [13]). There are still others designed with specific considerations, such as [14]–[16]. Performance evaluations on these rate adaptation algorithms are available in [17] [18].

1) Statistic based schemes: For statistic based schemes, channel quality estimation and rate selection are performed by the sender. Usually, statistics such as ACK, retry count, and FER are maintained as feedback at the sender. The rate is either adapted when the values of the indicators exceed their thresholds (ARF, ONOE, AARF/AMRR), or statistically selected out as the best one to use (SampleRate). Such techniques are widely used in current 802.11 products. However, it is hard to decide proper thresholds or statistic models, hence they are not very adaptive to time-varying channel conditions; also, adaption is relatively slow due to the use of indirect, long-term statistics for channel estimation. The Auto Rate Fallback (ARF) algorithm [5] uses a success threshold $ST = 10$ and a failure threshold $FT = 2$ to decide the rate increment/decrement behavior. However, it cannot react quickly to fast channel variations. Also, it may over-react when the channel quality is stable over a relatively long period. A dynamic rate adaptation algorithm proposed in [6] further develops ARF by using one failure threshold $FT = 1$ and two success thresholds ST_1 and ST_2 to distinguish between slow and fast changing channel conditions. In [8], AARF is proposed to alleviate the regular failure problem of ARF, by increasing the period between successive failed attempts mostly experienced in stable channel conditions. It adapts ST using a Binary Exponential Backoff (BEB) procedure to better reflect the channel conditions.

2) Signal measurement based schemes: Signal measurement based schemes use direct link quality metric (e.g., SNR, RSSI etc.) for optimal rate selection. The measured signal strength is used to look up the best data rate based on predefined mappings. These schemes have the advantage that the channel quality (indicated by the SNR, which is firmly related to the BER of the link) is obtained immediately, without a need to wait a long time before the feedback statistics are collected, and that no data is sent at rates higher than the current (optimal) rate. Despite these advantages, they have not been applied in practice so far, due to the difficulties to obtain reliable SNR estimates of the radio channel, and the fact that the SNR needed at the sender is the one observed at the receiver. In RBAR [10], the receiver estimates the channel condition using a sample of instantaneously received signal strength at the end of the RTS reception, and then piggybacks the selected transmission rate to the sender in CTS. OAR [12] improves RBAR by exploiting the time-varying nature of the channel and opportunistically sending multiple packets at high rate when the channel is favorable. MAD [19] further extends OAR exploiting the multiuser diversity. In [11], a link adaptor computes offline a best PHY mode table and at runtime, it monitors the SNR variations in the channel and performs a table lookup. In [13], signal strength of received frames and number of retransmissions are utilized to estimate the channel condition. Based on the statistics, a table is continuously adapted that maps RSS into throughput for different data rates.

B. Introduction to the AR5212 Chipset and MADWiFi driver

We carried out experiments with wireless adapters based on Atheros AR5212 chipset [20]. It maintains several FIFO (First In First Out) queues of transmission descriptors to schedule packets for transmission. Each descriptor contains detailed control information for a frame's transmission and a status field that records how the transmission is completed [8]. Among them, the most relevant information is an ordered set of 4 pairs of rate and transmission count fields, referred to as "multirate retry series", and the transmission status field including the sub-fields such as "ok", "excessive retries", "fail count", "final transmit index", "ACK signal strength". There

are four retry series (r_0/c_0, ..., r_3/c_3), each with a rate r_i and a limit c_i, which means that transmission attempts at rate r_i is at most c_i times ($i = 0, 1, 2, 3$). Typically, the AR5212 chipset supports default on-board multirate retry. However, the users can either disable this function, or customize the retry rate and retry number for each series. Whenever data is received from the upper layer, the driver prepares a descriptor for it, by filling in the fields with proper initial values, and inserts the descriptor into one of the FIFO queues. As soon as the wireless medium is available, the head-of-line frame is transmitted at the rate r_0. Other retry series are automatically carried out if necessary, as specified by the multirate retry series. Finally, the transmission status is returned to the descriptor for the reference of users of the driver.

A complete Linux driver for AR5210/5211/5212 chipsets is available from the Multiband Atheros Driver for WiFi (MADWiFi) [7]. Three rate adaptation algorithms, namely ONOE, AMRR [8] and SampleRate [9], are available in MADWiFi. We briefly review ONOE and SampleRate (more popular) below; useful descriptions can also be found in [17].

- *ONOE*: ONOE is an ARF-like rate adaptation algorithm. It runs periodically analyzing transmit statistics for each destination. The current rate for the destination is associated with a counter of credits. If transmissions at the current rate are judged to be good in that interval, a "tx_upper credit" is issued; otherwise, the credit counter is deducted. If the total credits at the current rate exceeds the $rate\_raise\_threshold$, the transmission rate is raised. If no packets have succeeded, or every packet needs retry in average, the transmission rate is decreased to the next lower one. We note that though ONOE is much less sensitive to an individual packet failure than ARF, it is also more conservative. When a better channel occurs, it takes at least 10 seconds to scale up. Similarly, if the channel is deteriorating, it can only step down to the next lower rate for each interval and eventually, there are too many packet losses before it finds a proper rate.
- *SampleRate*: SampleRate sends most data packets at the bit rate that would provide the highest throughput, no matter how lossy the link may be. Different from ONOE, it selects the bit rate on a per-frame basis. SampleRate periodically sends packets at rates other than the current one to estimate whether another rate will offer better throughput. For each destination, it keeps a record of the number of successive failures, the number of successful transmits and the total transmission time for a bit rate. Then it chooses the rate which may provide the most throughput based on estimates of the expected per-frame transmission time at each rate. SampleRate can generally achieve a higher throughput in low-quality links.

III. Experiment Methodology

A. Experiment Settings

We consider the network topology shown in Fig. 1. It consists of three laptops (N_1, N_2, N_3), all equipped with 3com 11a/b/g wireless PC cards (3CRPAG175 with XJACK Antenna). These cards are based on the AR5212 chipset. All laptops are running Linux (Fedora core 3 with kernel 2.6.1.667) with the MADWiFi driver including the above three algorithms as well as the *Smart Sender* algorithm. We configure the wireless cards to work under 802.11a. They are associated with the AP (Access Point) N_0, which is a 3com wireless 11a/b/g access point (3CRWE454A72). The AP is connected to a 100 Mbps Ethernet switch, which extends the wireless LAN to a wired LAN, where a Dell PC (N_4) is located. All equipment are placed in a typical office environment with lots of concrete walls and moving objects.

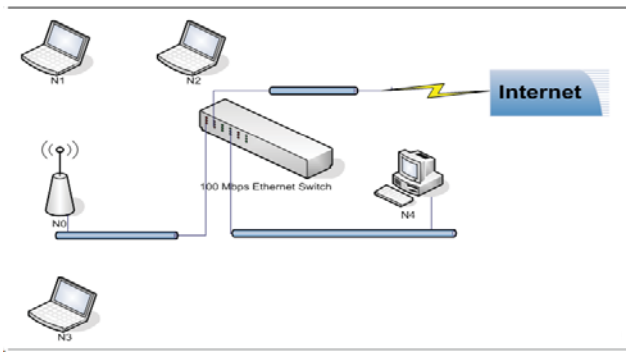


Fig. 1. Network topology in the experiments: N_0-N_3 form an infrastructure wireless LAN, which is connected to a wired LAN via a 100 Mbps Ethernet switch. A server PC N_4 is located at the wired LAN.

B. Experiment Methodology

We use "Netperf" [21] to generate continuous saturated UDP/TCP traffic and report the achieved application level throughput. Before our experiments, an additional wired throughput check is performed. We connect the server PC and the laptop being used for testing with a crossover Ethernet cable. The 100 Mbps Ethernet connection reports TCP throughput about 89 Mbps. This verifies that the wireless link is not affected by host hardware issues. Furthermore, the server PC resides on an independent subnet from the campus network, which ensures that the "server PC-to-AP" connection is not affected by traffic outside of the test setup. The paths between the AP and the laptops may not be LOS (Line-Of-Sight). In the experiments, we first place the laptops at different locations in static scenarios. To get large SNR variations in mobility scenarios, we also move the laptop at the speeds of 1 to 2 m/s. Specifically, we classify a wireless link into four types in terms of signal strength and varying speed listed in Table I. Fig. 2 shows the RSSI profiles of these links sampled in a 250 seconds interval.

We run experiments in different settings with static/mobile clients, in good/bad channel qualities, with/without contending stations. To minimize the uncertainty and make the comparisons meaningful, we use Channel 60 of 802.11a for no-contention scenarios, since it is orthogonal to other busy channels used by our department. We measure the stability (by average throughput), responsiveness (by response time), and contention performance of the rate adaptation algorithms in realistic scenarios. Stability means a rate adaptation algorithm can achieve desired performance references and is robust to disturbances. Note that an unstable algorithm may cause the bit rate diverge from the desired value and hardly recover to

TABLE I
LINK TYPES IN THE EXPERIMENTS

Link Types	LT 1: High-Stable Link	LT 2: Low-Stable Link	LT 3: High-Varying Link	LT 4: Low-Varying Link
Received signal strength $RSSI$	$> RSSI_{high1}$	$< RSSI_{low1}$	$> RSSI_{high2}$	$< RSSI_{low2}$
Varying speed in $RSSI$	slight	slight	large	large

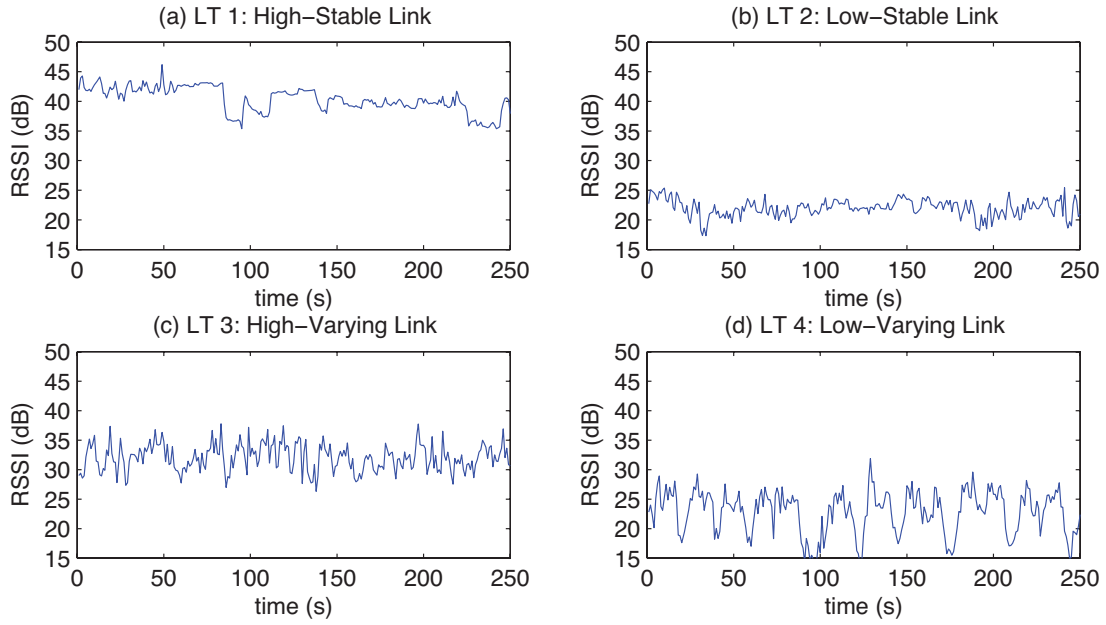


Fig. 2. RSSI profiles of four different link types.

a regular state. Responsiveness measures how fast it takes for the algorithm output to achieve the new desired value (i.e., bit rate), in reacting to environment changes. Both metrics measure the efficiency of the rate adaptation mechanisms.

IV. *Smart Sender* RATE ADAPTATION ALGORITHM

In this section, we focus on designing practical sender-based algorithms with an "ignorant" receiver, that can still be self-tuning and fast responsive. Our scheme is based on the following observations:

- The static nature of some previously proposed algorithms with static thresholds makes them less versatile to different channel conditions;
- Signal strength measurements should not be directly used for rate selection; however, they can be helpful in statistic based algorithms if properly used [22].
- Rate adaptation algorithms must be robust against congestion losses, although it may not be able to differentiate between congestion loss and wireless loss (which is extremely difficult without special effort).

Specifically, we propose the *Smart Sender* algorithm, which predicts channel dynamics based on collected feedback. The key component is to decide when to change the rate, and which rate to switch to. To achieve our goals, both statistics and SNR-related information are incorporated in rate adaptation. The former are continuously collected and updated; based on them, an optimization metric, i.e., the throughput, is calculated and acts as a primary rate-switch-decision trigger. The later aids to safeguard the selected rate, and improve system responsiveness, like in [14].

A. Statistic Adaptor

The statistic adaptor of *Smart Sender* is throughput-centered. It monitors the transmission history and computes a long-term transmission rate that provides the best throughput. In order to select one out of several candidates, *Smart Sender* probes at a new rate that may outperform the current rate, provided that: the expected throughput at the new rate exceeds the current rate; or transmissions at the current rate are suboptimal, and scaling up/down can actually improve the performance. As long as the statistic adaptor observes that the probing rate does not perform so well as expected, it stops probing immediately and resorts to the previous long-term transmission rate, which is selected based on periodical analysis of transmission history. During a transmission period, the long-term transmission rate is not changed if it is optimal in the sense that it provides the highest and most stable throughput. However, one can hardly judge this since no exact channel knowledge is known a priori. Here we adopt both long-term and short-term statistics to decide whether and when to probe at a new rate. It is based on the following observations: for system stability, long-term statistics (throughput, retry count) can decide whether a probe is justified; for system responsiveness, the consecutively received ACKs reflect improving channel conditions and the lost ACKs may indicate

a deteriorating channel. By probing at a promising rate, the sender can tune to the optimal rate as soon as possible.

1) Long-term statistics for system stability: Previous ARF-like schemes only use heuristics with consecutive successes/failures to decide an up/down scale, which may lead to premature rate switch decisions. Here we propose to use long-term throughput/retry statistics to justify the rate change. For this end, expected transmission time (ETT) is utilized which inherently takes account of the packet size, data rates and retransmissions into account. Base on ETT, expected goodput (EGP) can be calculated. A switch from the current rate $R_{cur} = R_i$ to the next higher/lower rate $R_{new} = R_{i+1}$ or R_{i-1} is possible only when the relationship in (1) is satisfied:

$$EGP(R_{cur}) < P_i \cdot EGP(R_{new}), \quad where \tag{1}$$

$$\begin{aligned} EGP(R_{cur}) &= \frac{\overline{pktSize}}{\overline{ETT}(R_{cur})}, \quad R_{cur} = R_i; \\ EGP(R_{new}) &= \frac{\overline{pktSize}}{\overline{ETT}(R_{new})}, \quad R_{new} = R_{i\pm1}. \end{aligned} \tag{2}$$

$\overline{ETT}(R_i)$ can be calculated using exponentially weighted moving average (EWMA):

$$\overline{ETT}_t(R_i) = \lambda \cdot \overline{ETT}_{t-1}(R_i) + (1-\lambda) \cdot ETT_t(R_i). \tag{3}$$

In (3), $ETT_t(R_i)$, the expected transmission time for tth packet at rate R_i is calculated as the sum of transmission time ($ett(j,k)$) at current or possibly following lower rates for this packet, assuming a total number of $m = \sum m(R_j)$ attempts are made to transmit the packet:

$$ETT_t(R_i) = DIFS + \sum_{j=i,k=k_0(R_i)}^{j\le i,k=m-1} ett(j,k), \quad where \tag{4}$$

$$\begin{aligned} ett(j,k) &= BK_{avg}(k) + PHY_{overhead} + T_{MACdata}(j,k) \\ &+ T_{prop} + SIFS + T_{ACK} + T_{prop}, \; where \end{aligned} \tag{5}$$

$$\begin{aligned} T_{MACdata}(j,k) &= \frac{S_{MACdata}}{R_{j,k}}, \\ T_{ACK} &= \frac{S_{ACK}}{R_{basic}}, \\ BK_{avg}(k) &= \min(2^{k-1}CW_{min}T_{slot}, CW_{max}). \end{aligned} \tag{6}$$

In (4), $ett(j,k)$ represents the time needed for one attempt at k_{th} attempt at the rate $R_{j,k}$. $\overline{ETT}(R_{new})$ is calculated using previous transmission history before adopting the new rate. If R_{new} is never used, a default value applies assuming a given mean packet size and no retransmissions. $DIFS$ and $SIFS$ are DCF Inter Frame Space and Short Inter Frame Space respectively. BK_{avg} is the average backoff time as a function of minimum contention window CW_{min} and backoff stage k. $PHYoverhead$ is the physical layer overhead (preamble, PLCP header, etc.). T_{prop}, $T_{MACdata}(j,k)$, T_{ACK} is the propagation delay, transmission time of MAC payload at the rate $R_{j,k}$, and transmission time of ACK respectively. Since we incorporate the multirate retry mechanism, the retry rate $R_{j,k}(k > 0)$ may differ from the current long-term transmission rate $R_{i,0} = R_i$, dependent on the retry stage k. The parameter $0 < P_i \le 1$ is used to ensure that the throughput advantage exists for some confidence interval. In our implementation, we set $P_i = p \cdot \frac{R_l}{R_h}$ (proportional to the ratio between the lower and the higher rate values). The intuition behind this is that, the larger the rate values differ, the harder the rate switch is.

2) Short-term statistics for system responsiveness: Consecutive successes/failures are indications of an improving/deteriorating channel. However, a channel usually behaviors diversely and does not follow a fixed success/failure pattern. Hence, static thresholds may degrade the system performance. As they are used to trigger the rate switch decision, their values must be carefully designed. When combined with the "multirate retry" mechanism, we define three counters: "consecutive successes" denotes the number of consecutive packets succeeding at the first attempt; "consecutive ACKs" indicates the number of consecutive packets that are sent successfully at the first retry rate $r_0 = R_i$ (possibly with several retries); and "consecutive failures" means the number of consecutive packets that fail at all of the attempts with the first retry rate r_0. The rationale behind this is that as long as the failed packet finally succeeds at r_0, we do not count it in consecutive failures, since previously failed attempts of this packet may be caused by collision or fast fading, which does not account for stable deteriorations in channel quality, and is handled by the "multirate retry" strategy. Based on the above analysis, we analyze failure threshold FT and success threshold ST settings in the next sections:

3) FT settings: FT for "consecutive failures" should be small to make the algorithm react quickly to deteriorating channel conditions: a high value may cause too many failed attempts before the long-term rate is reduced. When transmitting smaller packets at lower rate, FT should be larger. Hence, we use 2 different FT's, $FT_1 = 2$ and $FT_2 = 4$; at runtime, one of them is chosen according to the ratio calculated by (7):

$$I_{FT} = \frac{S_{refMACdata} \cdot R_{max}}{S_{MACdata} \cdot R_{cur}}. \tag{7}$$

where $S_{refMACdata}$ and $S_{MACdata}$ are the reference MAC data size and current MAC data size; R_{max} and R_{cur} are the maximum data rate and the current data rate. The default FT uses the smaller one FT_1; whenever the calculated I_{FT} is larger than the threshold I_{thFT}, FT is set to FT_2.

4) ST settings: ST specifies the number of successful packets in a row at the first attempt before it believes the link quality has improved so that it should use a higher rate. A fixed value of ST, like in ARF, is very sensitive to the changing speed of link quality. Under stable channel conditions, the rate adaptation algorithm can finally reach the best rate; after that, it periodically attempts to transmit at a higher data rate, which causes repeated failures. Therefore, an algorithm with dynamic ST is preferred. When the channel condition changes slowly, it is better to increase ST so as to minimize the undesired rate increments. On the other hand, when the channel condition is fluctuating, it is critical to find an optimal rate and stay there for as long as possible, to avoid jitter.

In this paper, we propose a dynamic ST scheme similar to that of TCP's congestion control behavior. We first briefly review the basic TCP rules. The sending behavior of a TCP

sender is determined by its state (either slow-start or congestion avoidance) and the congestion window W. The sender begins at the slow-start phase with initial congestion window $W = 2$. Packets are sent back-to-back until a total of W packets are in flight. The sender stops if a window of packets are outstanding and waits until an ACK is received. During slow start, for each arrival of TCP ACK, the sender increases W by one, which means W effectively doubles every RTT (round trip time). In congestion avoidance, W is increased by $1/W$ for each ACK and hence one every RTT. Whenever a loss is detected, the sender halves the window and enters/stays in the congestion avoidance state. The key idea behind TCP congestion control is to probe the available bandwidth on the network and to adjust the transmission rate accordingly. To achieve this objective, TCP adopted the additive increase multiplicative decrease (AIMD) mechanism. By additively increasing W until detecting packet loss, which is regarded as an implicit notification of congestion, TCP sender probes available bandwidth and tries to use the bandwidth most effectively. When it detects packet loss, TCP sender decreases W multiplicatively, which reduces the transmission rate.

There is a correspondence between the nature of this problem in WLANs and the nature of TCP congestion window setting in Internet. The key idea of rate adaptation is to probe the available bandwidth on the wireless link and to adjust the transmission rate accordingly. To achieve this objective, we adopt the multiplicative increase linear decrease (MILD) mechanism for ST. The reason of using "MI" instead of "AI" is that we now have a reduced "RTT" when former ST is reached and the bit rate is increased on the wireless link, rather than an invariant "RTT" in the TCP case. Similar reason justifies "LD". If previous ST has been reached and the bit rate is increased, we conclude that the channel quality is improving, and since the rate is already higher now, we should increase ST to stay in the current high rate as long as possible. By this way, we make sure that if the channel quality is improving quickly, the new ST can be reached soon again; otherwise, the undesired rate increment attempt is postponed. Here we mark the transmissions of the first packet at the new rate as "recovery"; if transmission failures at the new rate immediately follow the rate increment, while resort to the previous rate, we also increase ST to discourage further rate increments. In other cases when the rate is decreased, we reduce ST to encourage potential rate increments. Specifically, we have an initial threshold value ST_{min}. When conditions are satisfied for increasing ST, we increase it multiplicatively by a factor of α. When conditions are satisfied for decreasing ST, we decrease it linearly by a factor of β. Throughout this process, ST is bounded in the interval $[ST_{min}, ST_{max}]$.

The parameters $[ST_{min}, ST_{max}]$ and (α, β) need to be carefully chosen. Among them, ST_{min} and α are of the most importance. We have the following facts:

- TCP uses small values of initial window size (2), additive increase parameter (1), and multiplicative parameter (2).
- The *coherence time* of the indoor radio channel, a measure of the average time duration over which the radio channel is stationary, is on the order of tens to hundreds of packet transmission time.

Therefore, we examined several small values in [6, 20] for ST_{min} and [1.5, 3] for α respectively. The results show that the combination of $ST_{min} = 8$ and $\alpha = 2$ performs slightly better than others under most of the scenarios, while larger values perform worse. ST_{max} and β are set to simple values such as 50 and 6. We note that ST should also account for the packet size and data rate. Possible solutions include adopting several ST's for different packet size intervals, or updating the counters according the actual packet size and transmission rate. We will discuss these issues in our future work.

B. RSSIA Regulator

In current design, wireless interface reports to the upper layer about the signal strength of a received frame, denoted by RSSI. Among them, RSSI of ACK is utilized to reflect the channel quality at the receiver, which can avoid complicated signal strength measurement as in [14]. For safety use of RSSIA, a RSSIA regulator is developed to calculate the average RSSI of recently received ACKs and predict the dynamics of the channel condition based on the most recent RSSIA values. Firstly, the RSSIA regulator acts as a safeguard which limits the range of new rates to choose from. When ACKs are continuously received, the average RSSIA is calculated by EWMA, upon which feasible rates are predicated. The rate selected by the statistic adaptor must be within the safety region of the ones bounded by RSSIA. For example, suppose the statistic adaptor decides to increase to a new rate larger than all of the feasible rates, the increment action is postponed. Secondly, the RSSIA regulator is also used for fast recovery to a new rate. Once $\max(ST_{min}, ST/2)$ is reached, if the regulator observes improving channel quality by increasing average RSSIAs, the sender may trigger the rate-up action immediately, without waiting to reach ST. Note that $\max(ST_{min}, ST/2)$ is still required to ensure that such improvement is a long-term effect instead of a transient fluctuation. Finally, RSSIA readings can be viewed as an indicator for contention level on the channel. We should not reduce the rate too aggressively when average RSSIA is stable. This point is not included in our current implementation but is left to future work.

V. Practical Implementation of *Smart Sender* in MADWiFi

A. Statistic Counters

A "smart sender" takes the full responsibility of maintaining for a certain destination several counters, such as successful/failed transmissions, and retry counters for each data rate. They are evolved in the calculation for long-term statistics. In addition, short-term throughput statistics for currently used rate, such as "consecutive successes/ACKs/failures" are maintained: if the first attempt succeeds, the *success* counter is increased by one; if some attempt at the first retry rate succeeds, the *acked* counter is increased by one; otherwise, the *failure* counter is increased by one and the *success/acked* counters are reset to zero. Generally speaking, if the *success/failure* counter reaches ST/FT and (1) is satisfied, the current rate for the corresponding destination can be increased/decreased. As long as the current rate is changed, the "success/acked/failure" counters are reset to zero. As presented in Section IV-B,

other control logics are added into this basic structure, for the purpose of safety and fast responsiveness.

1) When to probe up: Up-scale probing is triggered when one of the following conditions is satisfied:

- The consecutive success counter reaches ST, and the feasible rate looked up according to the RSSI threshold table is larger than the current rate;
- The consecutive success counter reaches ST, and the expected throughput satisfies (1);
- The consecutive success counter reaches $ST/2$ and the RSSIs of continuously received ACKs are judged to be improving quickly.

2) When to probe down: Accordingly, down-scale probing is triggered when one of the following conditions is satisfied:

- The consecutive failure counter reaches FT, and the feasible rate looked up according to the RSSI threshold table is smaller than the current rate;
- The consecutive failure counter reaches FT, and the expected throughput satisfies (1);
- The consecutive failure counter reaches FT and the RSSIs of continuously received ACKs are judged to be deteriorating quickly.

B. Multirate Retry Mechanism

To resolve transient channel variations, the *Smart Sender* algorithm keeps the multirate retry mechanism with four retry series (like that in ONOE), but modifies the values of c_i's to make them suitable for our rate adaptation goals. Among the four rate/count pairs, the allowed retry number for the first retry series c_0, is most relevant to system responsiveness. A high value makes the second and later attempts use the same rate as the first, which may in turn cause additional retransmissions in the case where the channel condition is really degrading. Therefore, we set c_0 to 2, which means that the first and second attempts use the long-term rate r_0; after that, the possible following retries use lower rates. The other reason is that, with two attempts at the first retry rate, we do not rush to a reduced rate once the first attempt fails, because it is very probably that such a failure is caused by a collision. Similarly, to ensure fast responsiveness to short-term channel variations, c_1, c_2 and c_3 are all set to a smaller value of 1, since the collision probabilities for these later attempts are much smaller.

C. State Transitions

The visible long-term transmission rate for a node, $txRate$ (the index to the current sorted rate set), is adjusted according to the sender's state. The main idea for state transitions is illustrated in Fig. 3. Specifically, a sender has two states, the "Tx" state and the "Probe" state. The statistics for these states are updated/reset periodically according to a timer, with a default timeout interval of 1000 ms. At the very beginning of a new round, the sender is at its "Tx" state and transmits packets at $r_0 = txRate$. Meanwhile, the sender monitors the transmission results of the current rate and if it observes a promising rate by $may\_probe()$, which can possibly improve the throughput, the sender sets up a new rate $probeRate$ and

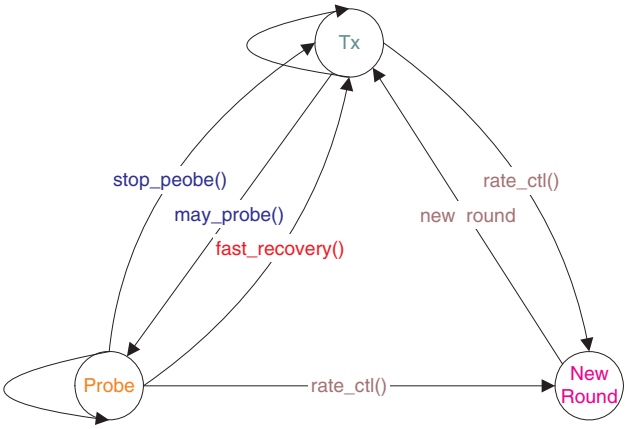


Fig. 3. State transitions at the sender: in each round, the sender goes between the "Tx" and "Probe" state; when the timer expires, the sender begins a new round with an initial state of "Tx".

enters into the "Probe" state. The following queued packets are sent using $r_0 = probeRate$, but $txRate$ is not altered. Similarly, the sender keeps on monitoring the probing results at the $probeRate$ and if it concludes that the $probeRate$ offers a better performance than the $txRate$, the sender quickly recovers by setting $r_0 = txRate = probeRate$ and enters into a new "Tx" state; otherwise, the sender stops probing and goes back to the "Tx" state. After settling r_0, the remaining rate series r_1, r_2, r_3 are determined in the same way as in ONOE. Whenever a state transition occurs, the associated statistics are updated. This process continues until the timer expires, then the statistic counters are reset and a new round is started. The three procedures, $may\_probe()$, $stop\_probe()$ and $fast\_recovery()$ describe when the sender is allowed to change its current state, which is critical to the throughput performance.

Appendix I provides the pseudo code description of the proposed algorithm. The flowchart of data paths and control logics of the algorithm is shown in Fig. 4. In *Smart Sender*, the sender collects statistics in the function $tx\_complete()$. After a frame transmission, the returned hardware status in the descriptor is either "ok" ($status = 0$) or "error" ($status \neq 0$), which is distinguished in $tx\_complete()$. Note that the transmission "ok" status does not necessarily means that the first attempt is successful, but that the frame may be successfully transmitted after several retransmissions. Accordingly, the transmission "error" status means that all attempts of the frame fail and it is discarded finally when the retry limit is exceeded. The average RSSIA is used to look up feasible rates, and predict channel dynamics, which are implemented in the functions of $lookup\_rssiThresholdTable()$, $fast\_up()$, and $fast\_down()$. Note that at the "Tx" state, the sender also uses a threshold-based scheme like that in ARF, with FT set to 2 or 4. Moreover, to resolve the problems of ARF with a static ST, it uses MILD to dynamically adapt ST, which is bounded in the interval $[ST_{Min}, ST_{Max}]$. Whenever the threshold is reached, along with the condition (1) satisfied, the sender starts the "Probe" state by setting $curRate = probeRate$. The rate selection is made in the function $findrate()$, which decides the series 0 rate r_0 and retry number c_0 for a new

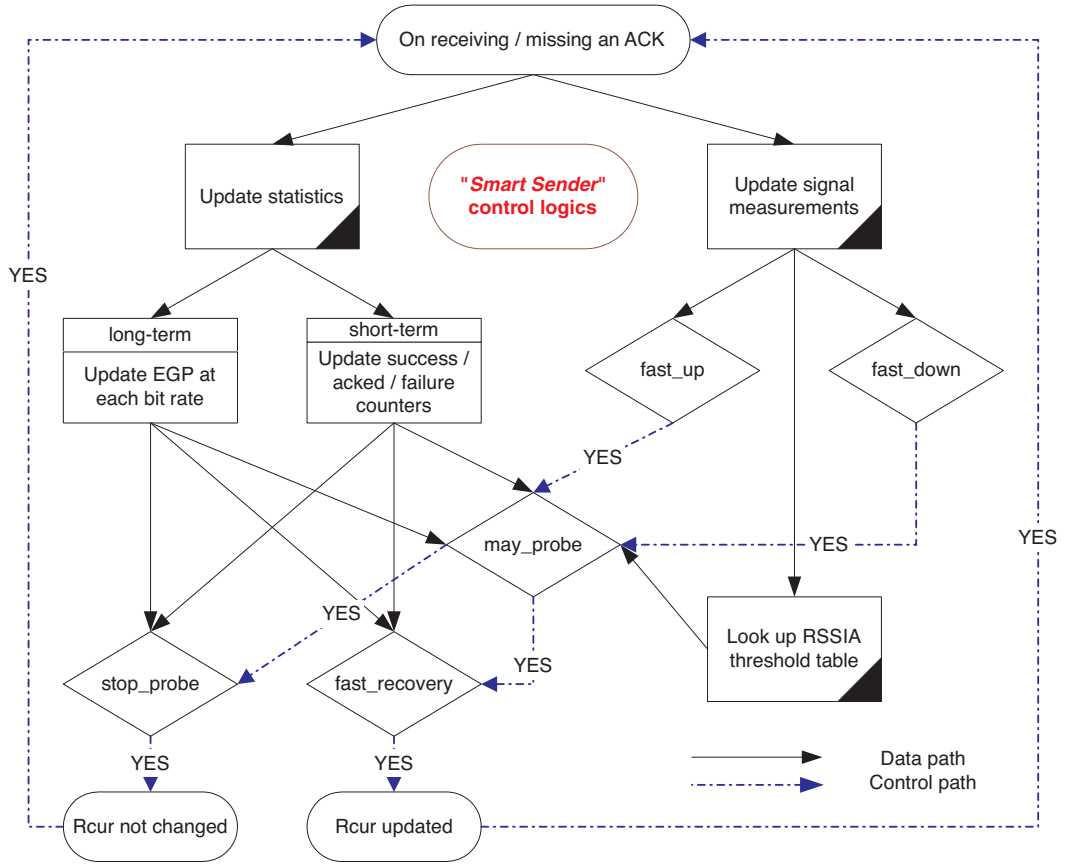


Fig. 4. Data paths and control logics of the "*Smart Sender*" algorithm.

outgoing packet. With the multirate retry mechanism enabled, r_0 is set to the current rate $curRate$, which is either the long-term transmission rate $txRate$, or the probing rate $probeRate$. Then, the other multirate retry series are decided in the function $setupxtxdesc()$, which completes the initialization of the transmission descriptor for that packet. Finally, the head-of-line packet is transmitted when the medium becomes available.

VI. Performance Evaluation

After implementation of the *Smart Sender* algorithm on real hardware, we conduct extensive experiments to study its performance under various link conditions as introduced in Section III. We compare the throughput/responsiveness with existing algorithms in MADWiFi, shown in Table III. Table II summarizes 802.11a MAC/PHY parameters and other control parameters used in our experiments. We present main performance results in this section.

A. UDP/TCP Throughput

Fig. 8(1)-(4) study the rate distribution probability of ONOE and *Smart Sender* transmitting UDP traffic under different link types. The total throughput in these scenarios is shown in Fig. 5a. In all scenarios, *Smart Sender* manages to send more packets at higher rates than ONOE, mainly due to the "probe" and "fast recovery" mechanisms. In Fig. 8(1), where the link is stable and with high quality, both ONOE and *Smart Sender* can achieve similar optimal throughput.

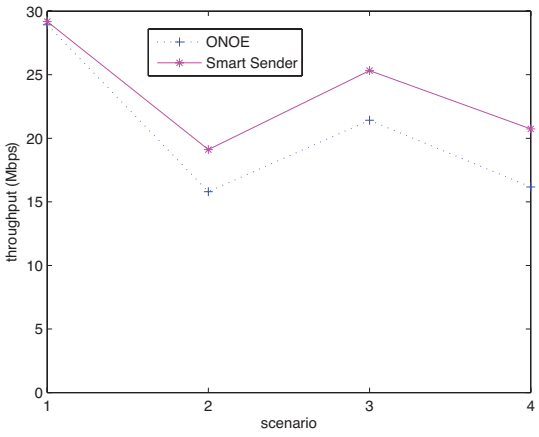
TABLE II
MAC/PHY and Control Parameters Used in Experiments

CW_{Min}	15	ST_{max}	50
CW_{Max}	1023	ST_{min}	8
SlotTime	$9\,\mu s$	FT_1	2
SIFSTime	$16\,\mu s$	FT_2	4
DIFSTime	$28\,\mu s$	α	2
Preamble Duration	$16\,\mu s$	β	6
PLCP Header	$4\,\mu s$	$rateinterval$	$1000\,ms$
TCP Packet Size	1500 Bytes	$enough$	20 packets
UDP Packet Size	1472 Bytes	$c_0,\ c_1,\ c_2,\ c_3$	2, 1, 1, 1

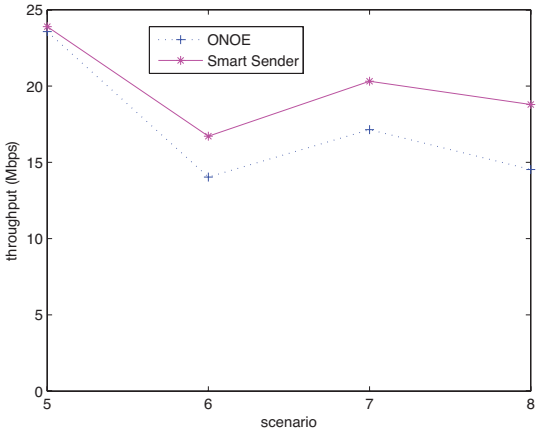
However, ONOE spends much longer time, 10 seconds at the initial rate 36 Mbps before scaling up to 48 Mbps and spends another 10 seconds at 48 Mbps before finally reaching the best 54 Mbps; on the other hand, *Smart Sender* can quickly recover to 54 Mbps after several transmissions at the beginning of the first second. Therefore, *Smart Sender* still achieves slightly better performance than ONOE. In Fig. 8(2), the channel quality is poor and the optimal long-term rate is around 24 or 36 Mbps. Since ONOE requires 10 credits to trigger a rate increase, it spends too much transmission time at 24 Mbps and can only transmit at 36 Mbps for less time. *Smart Sender* does not restrict the rate increase action and can respond to the variations quickly enough (several packet transmission time), so the packets sent at 36 Mbps is much more than in ONOE. Similarly, in Fig. 8(3), the

TABLE III
PRONS/CONS OF THREE ALGORITHMS

Adjustment interval	Algorithm	CSI	Comparison
Per-window	ONOE	statistic based	stable long-term throughput
Per-packet	SampleRate	statistic based	short-term responsiveness
Per-packet	*Smart Sender*	statistics and RSSIA	stability and responsiveness

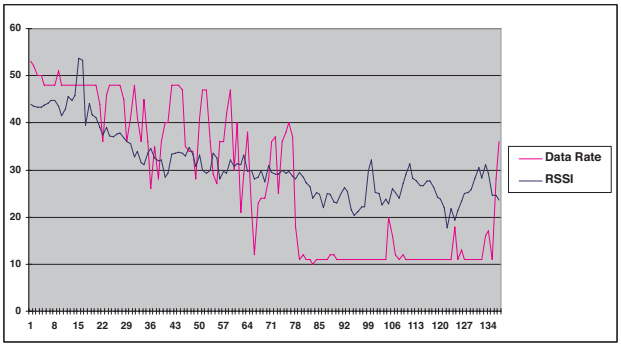


[UDP throughput in scenario 1-4]

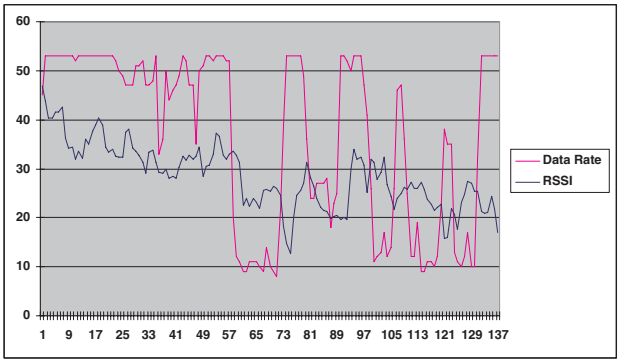


[TCP throughput in Scenario 5-8]

Fig. 5. UDP/TCP throughput comparison of ONOE and *Smart Sender* in all scenarios.



[SampleRate trace]



[*Smart Sender* trace]

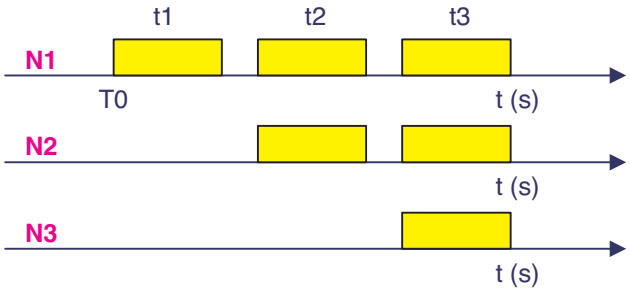
Fig. 6. UDP $N_1 \rightarrow N_0$: Data rate under varying channel conditions.

channel quality is good but N_1 is moving around, causing the received signal strength to vary over time at the AP. Therefore, ONOE can hardly accumulate 10 credits to scale up at 48 Mbps. However, when *Smart Sender* observes an improving channel, it quickly probes at 54 Mbps, and when it senses that the channel quality is deteriorating, it either stops probing or retracts to 48 Mbps immediately. This is also true in Fig. 8(4). Fig. 8(5)-(8) study the rate distribution of both rate adaptation algorithms transmitting TCP traffic. The achieved TCP throughput (Fig. 5b) is less than that of UDP, because of additional protocol overhead such as TCP ACK. These results demonstrate similar characteristics of *Smart Sender* as in UDP scenarios.

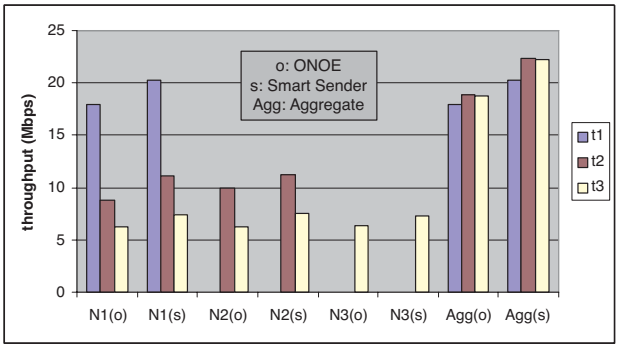
Another feature of *Smart Sender* is, though it reacts to the result of single packet transmission, it is stable by using long-term statistics and RSSIAs to bound rate switch actions and avoid jitter. In *Smart Sender*, the dominant factor to decide a rate switch is throughput. If the current long-term rate can offer better throughput, no adjustment for the long-term rate is made though it may be suboptimal under some transient channel conditions; indeed, transient variations are resolved by the "multirate retry". Even if *Smart Sender* selects a wrong rate to probe, it can perceive the error through a few packet transmissions and stop probing at that rate in the current round.

B. Responsiveness

Since ONOE is based on per-window adaptation, which is inherently less responsive, in this section, we compare the responsiveness of *Smart Sender* with SampleRate. Fig. 6 shows the trace profiles of both algorithms. For SampleRate, the rate decreases when the channel quality drops for certain interval, but with a phase lag. Also, the rate decreases are not smooth but with lots of glitches. However, with *Smart Sender*, the sender can predict and tune to the channel variations quickly with little delay. *Smart Sender* is opportunistic, which can quickly switch to a higher rate once a good channel occurs, and down-scale while observing a string of failures.



[Time-line for the transmissions of N_1, N_2 and N_3]



[Throughput of N_1, N_2 and N_3 under contention]

Fig. 7. UDP throughput comparison of N_1, N_2 and N_3 under contention.

C. Throughput under Contention

In the last experiment, we evaluate the UDP performance of ONOE and *Smart Sender* under contention. In this scenario, N_1 is always moving; N_2 and N_3 are stationary. At T_0, N_1 starts the UDP traffic to N_0, which lasts for 30 seconds (t_1); then N_1 waits for 10 seconds; this pattern is repeated twice more (transmitting in t_2 and t_3). N_2 starts the UDP traffic to N_0 at the time $T_0 + 40$, which also lasts for 30 seconds (t_2); then N_2 waits for 10 seconds; this pattern is repeated once more (in t_3); The UDP traffic from N_3 to N_0 starts at the time T_0+80, lasts for 30 seconds (t_3) and ends. Therefore, the contention levels in the intervals of t_1, t_2 and t_3 are increasing along with the time. The time-line for the transmission/stop intervals of the three nodes is shown in Fig. 7a.

The throughput performance is shown in Fig. 7b. As expected, when there is no contention during t_1, *Smart Sender* outperforms ONOE. Besides, in t_2 and t_3 intervals, we can still observe higher throughput of *Smart Sender* than ONOE. We conclude that one reason for this is due to the use of RSSIA, which can bound the range of feasible rates and limit the premature rate decrement actions. Also, note that in *Smart Sender*, rate decrement is possible when consecutive failures occur, which is, with a high probability, caused by imperfect channel quality rather than collisions. Therefore, *Smart Sender* is more suitable for occasional contention errors.

VII. Conclusion

Rate adaptation algorithms are extremely important for WLANs with multirate capabilities. Previously proposed solutions are based on either statistics of ACKs, retransmissions, or signal strength measurements. In this paper, we have proposed and evaluated a novel sender-based ARF-like algorithm, *Smart Sender*, which combines statistic based methods with signal measurement based methods to get the best of both worlds. It does not require any changes to the IEEE 802.11 standard. The novelty of our rate adaptation scheme lies in its great adaptability to a variety of channel conditions robust enough to be easily adopted for current wireless hardware. This rate adaptation framework also provides hints for systems (such as MIMO) used in future high-speed WLANs.

Appendix I
Smart Sender Algorithm

Smart Sender algorithm is described here for convenience.

Initially, $probe = 0$; {at the "Tx" state}
$tx\_complete()$:
update try counter of packet transmissions at each rate **try[ ]**;
update average RSSIA with newly received ACK $\overline{\textbf{rssi}}$;
$feasibleRate = lookup\_rssiThresholdTable(\overline{rssi0})$;
{calculate $feasibleRate$ from current RSSIA}
$bestRate = find\_best\_rate\_by\_stats(\textbf{try[ ]})$;
{calculate $bestRate$ from transmission statistics}
if $status == 0$ **then**
  $err = 0$;
  if $retryCount == 0$ **then**
    $success++$; $acked++$; $failure = 0$;
    {consecutive successes/ACKs at $curRate$ without retries}
  else if *succeed at the first retry rate* **then**
    $acked++$; $success = 0$; $failure = 0$;
    {consecutive ACKs at $curRate$ with retry}
  else
    $success = 0$; $acked = 0$; $failure++$;
    {consecutive failures other than $curRate$ with retries}
  end if
else
  $success = 0$; $acked = 0$; $failure = 0$; $err++$;
  {consecutive errors at $curRate$ exceeding retry limit}
end if
if $!probe$ && $prRate = may\_probe() != -1$ **then**
  $curRate = probeRate = prRate$;
  $probe = 1$;
else if $probe$ && $fast\_recovery()$ **then**
  $curRate = txRate = probeRate$;
  $probe = 0$;
else if $probe$ && $stop\_probe()$ **then**
  $curRate = txRate$;
  $probe = 0$;
end if

$findrate()$:
{find series0 rate before a frame transmission}
if $mrretry$ **then**
  $rix = curRate$; $try0 = 2$;
else
  $rix = fixedRate$; $try0 = TXMAXTRY$;
end if

$setupxtxdesc()$:
{set up mrretry descriptor before a frame transmission}

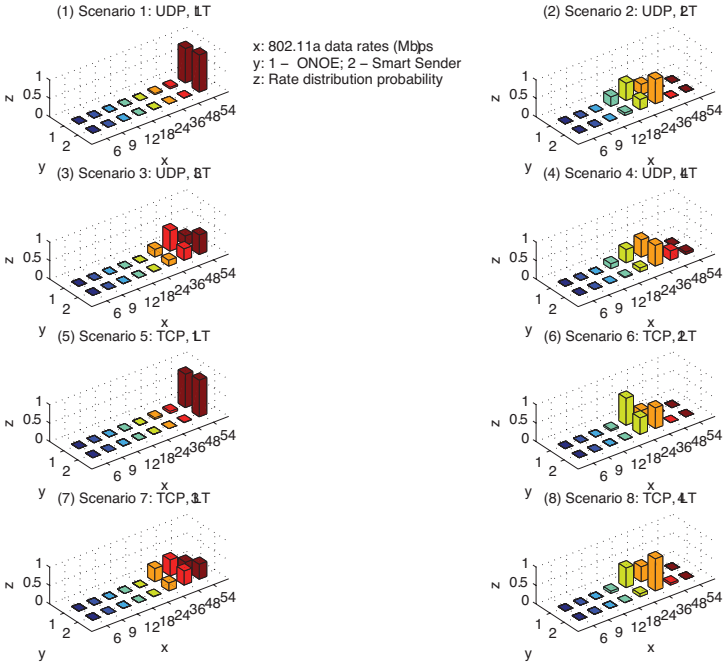


Fig. 8. Rate distribution probability for UDP/TCP traffic on different links.

$rate0 = curRate$;
$rate1 = --rate0 > 0?rate0 : 0$;
$rate2 = --rate0 > 0?rate0 : 0$;
$rate3 = 0$;
$hal\_setupxtxdesc(..., rate1, 1, rate2, 1, rate3, 1)$;

$rate\_ctl()$:
update/reset statistics for the new round;

$may\_probe()$:
$prRate = -1$;
if !$maxRate()$ && $success \geq \max(ST_{min}, ST/2)$ **then**
  if $(feasibleRate > txRate \mid\mid egp[txRate + 1] \cdot P_i > egp[txRate])$ && $success \geq ST$ **then**
    $prRate = txRate + 1$;
  else if $fast\_up(\mathbf{rssi})$ **then**
    $prRate = txRate + 1$;
  end if
  if $prRate > txRate$ **then**
    $recovery = 1$; {mark the first probe}
    $ST* = \alpha$; $ST = min(ST, ST_{max})$;
  else
    $recovery = 0$;
  end if
end if
if $acked == 0$ **then**
  if !$minRate()$ & &$failure \geq FT$ **then**
    if $feasibleRate < txRate \mid\mid fast\_down(\overline{\mathbf{rssi}}) \mid\mid egp[txRate - 1] \cdot P_i > egp[txRate]$ **then**
      $prRate = txRate - 1$;
    end if
  else if $err > 0$ **then**
    $prRate = bestRate$;
  end if
  if $recovery$ **then**
    $ST* = \alpha$; $ST = min(ST, ST_{max})$;
  else if $txRate > prRate$ **then**
    $ST- = \beta$; $ST = max(ST, ST_{min})$;
  end if
  $recovery = 0$;
end if
if $prRate! = -1$ && $canProbe[prRate] == -1$ **then**
  $prRate = -1$;
end if
return $prRate$;

$stop\_probe()$:
if $(enough()$ && $egp[probeRate] < egp[txRate]) \mid\mid (failure > FT) \mid\mid (err > 0)$ **then**
  $canProbe[probeRate] = -1$;
  return 1;
end if
return 0;

$fast\_recovery()$:
if $(enough()$ && $egp[probeRate] \cdot P_i > egp[txRate]) \mid\mid (success \geq ST)$ **then**
  $canProbe[probeRate] = 1$;
  return 1;
end if
return 0;

REFERENCES

[1] IEEE Std 802.11-1999, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Std., Aug. 1999.
[2] IEEE Std 802.11a-1999, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band*, Std., Sep. 1999.

[3] IEEE Std 802.11b-1999, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer Extension in the 2.4 GHz Band*, Std., Sep. 1999.
[4] IEEE Std 802.11g-2003, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Further Higher Data Rate Extension in the 2.4 GHz Band*, Std., Jun. 2003.
[5] A. Kamerman and L. Monteban, "WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band," *Bell Labs Technical Journal*, pp. 118–133, Summer 1997.
[6] P. Chevillat, J. Jelitto, A. N. Barreto, and H. L. Truong, "A Dynamic Link Adaptation Algorithm for IEEE 802.11a Wireless LANs," in *Proc. ICC'03*, May 2003, pp. 1141–1145.
[7] "MADWIFI." [Online]. Available: http://madwifi.org/
[8] M. Lacage, M. Manshaei, and T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach," in *Proc. MSWiM'04*, Venice, Oct. 2004, pp. 126–134.
[9] J. C. Bicket, "Bit-rate Selection in Wireless Networks," Master's thesis, Feb. 2005.
[10] G. Holland, N. Vaidya, and P. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," in *Proc. ACM MOBICOM'01*, Rome, Italy, Jul. 2001, pp. 236–251.
[11] D. Qiao, S. Choi, and K. G. Shin, "Goodput Analysis and Link Adaptation for IEEE 802.11a Wireless LANs," *IEEE Trans. Mobile Computing*, vol. 1, no. 4, pp. 278–292, October-December 2002.
[12] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Oar: an opportunistic auto-rate media access protocol for ad hoc networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 39–53, Jan. 2005.
[13] J. del Prado Pavon and S. Choi, "Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement," in *Proc. ICC'03*, vol. 2, Anchorage, Alaska, May 2003, pp. 1108–1113.
[14] I. Haratcherev, K. Langendoen, R. Lagendijk, and H. Sips, "Hybrid Rate Control for IEEE 802.11," in *Proc. MobiWac'04*, Philadelphia, PA, Oct. 2004, pp. 10–18.
[15] M. Manshaei, T. Turletti, and M. M. Krunz, "Media-Oriented Transmission Mode Selection in 802.11 Wireless LANs," in *Proc. WCNC'04*, Atlanta, Georgia, Mar. 2004, pp. 1525–3511.
[16] J. Kim, S. Kim, S. Choi, and D. Qiao, "CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs," in *Proc. INFOCOM'06*, Barcelona, Spain, Apr. 2006, pp. 1–11.
[17] S. Pal, S. R. Kundu, K. Basu, and S. K. Das, "IEEE 802.11 Rate Control Algorithms: Experimentation and Performance Evaluation in Infrastructure Mode," in *Proc. PAM'06*, Adelaide, Australia, Mar. 2006.
[18] Y. Yang, M. Marina, and R. Bagrodia, "Experimental Evaluation of Application Performance with 802.11 PHY Rate Adaptation Mechanisms in Diverse Environments," in *Proc. WCNC'06*, Las Vegas, USA, Apr. 2006, pp. 2273–2278.
[19] Z. Ji, Y. Yang, J. Zhou, M. Takai, and R. Bagrodia, "Exploiting Medium Access Diversity in Rate Adaptive Wireless LANs," in *Proc. ACM MOBICOM'01*, Philadelphia, PA, Sep. 2004, pp. 345–359.
[20] "ATHEROS Communications." [Online]. Available: http://www.atheros.com/pt/index.html/
[21] Netperf, "The Networking Benchmark." [Online]. Available: http://www.netperf.org/
[22] M. Souryal, L. Klein-Berndt, L. Miller, and N. Moayeri, "Link Assessment in an Indoor 802.11 Network."

Qiuyan Xia (S'05) received her B.S. degree in Computer Science from the Nanjing University, Nanjing, China, in 2003. She is currently working towards the Ph.D degree in Computer Science and Engineering, at the Hong Kong University of Science and Technology, Hong Kong, China. Her research interests include cross layer design and optimization, adaptive algorithms, link adaptation, resource management for the IEEE 802.11 WLANs, and quality of service provisioning.

Mounir Hamdi (S'90-M'91-SM'06) received the B.S. degree in Electrical Engineering from the University of Louisiana in 1985, and the MS and the PhD degrees in Electrical Engineering from the University of Pittsburgh in 1987 and 1991, respectively.

He has been a faculty member in the Department of Computer Science at the Hong Kong University of Science and Technology since 1991, where he is now Full Professor of Computer Science, Director of the Computer Engineering Program, and Director of the Master of Science in Information Technology. He is a member of the University Senate and University Council. In 1999 to 2000 he held visiting professor positions at Stanford University, USA, and the Swiss Federal Institute of Technology, Lausanne, Switzerland. His general area of research is in high-speed wired/wireless networking in which he has published more than 220 research publications, received numerous research grants, and graduated more 20 PhD/Master students. In addition, he has frequently consulted for companies in the USA, Europe and Asia on high-performance Internet routers and switches as well as high-speed wireless LANs.

Dr. Hamdi is/was on the Editorial Board of IEEE Transactions on Communications, IEEE Communication Magazine, Computer Networks, Wireless Communications and Mobile Computing, and Parallel Computing. He was a guest editor of IEEE Communications Magazine, guest editor-in-chief of two special issues of IEEE Journal on Selected Areas of Communications, and a guest editor of Optical Networks Magazine, and has chaired more than 7 international conferences and workshops including The IEEE International High Performance Switching and Routing Conference, the IEEE GLOBECOM/ICC Optical networking workshop, the IEEE ICC High-speed Access Workshop, and the IEEE IPPS HiNets Workshop. He is/was the Chair of IEEE Communications Society Technical Committee on Transmissions, Access and Optical Systems, and Vice-Chair of the Optical Networking Technical Committee, as well as member of the ComSoc technical activities council. He is/was on the technical program committees of more than 120 international conferences and workshops. He received the best paper award at the International Conference on Information and Networking in 1998 out of 152 papers. In addition to his commitment to research and professional service, he is also a dedicated teacher. He received the best 10 lecturers award and the distinguished engineering teaching appreciation award from the Hong Kong University of Science and Technology, and various grants targeted towards the improvement of teaching methodologies, delivery and technology. He is a member of IEEE and ACM.