Fast Resilient Jumbo Frames in Wireless LANs

Anand Padmanabha Iyer, Gaurav Deshpande, Eric Rozner, Apurv Bhartia, Lili Qiu
{aiyer,gauravd,erozner,apurvb,lili}@cs.utexas.edu
University of Texas at Austin

***Abstract*—With the phenomenal growth of wireless networks and applications, it is increasingly important to deliver content efficiently and reliably over wireless links. However, wireless performance is still far from satisfactory due to limited wireless spectrum, inherent lossy wireless medium, and imperfect packet scheduling. While significant research has been done to improve wireless performance, much of the existing work focuses on individual design space. We take a holistic approach to optimizing wireless performance and resilience. We propose Fast Resilient Jumbo frames (FRJ), which exploit the synergy between three important design spaces: (i) frame size selection, (ii) partial packet recovery, and (iii) rate adaptation. While these design spaces are seemingly unrelated, we show that there are strong interactions between them and effectively leveraging these techniques can provide increased robustness and performance benefits in wireless LANs. FRJ uses jumbo frames to boost network throughput under good channel conditions and uses partial packet recovery to efficiently recover packet losses under bad channel conditions. FRJ also utilizes partial recovery aware rate adaptation to maximize throughput under partial recovery. Using real implementation and testbed experiments, we show that FRJ out-performs existing approaches in a wide range of scenarios.**

***Index Terms*—Wireless LAN, jumbo frame, partial recovery, rate adaptation.**

I. Introduction

The popularity of wireless networks has grown at a phenomenal rate. Yet wireless performance is still far from satisfactory due to limited wireless spectrum, inherent lossy wireless medium, and imperfect packet scheduling. Emerging trends such as rapidly growing densities and increasing traffic volumes only exacerbate this problem.

Many novel techniques have been proposed to improve the efficiency and resilience of wireless networks. In particular, sending large frames has been suggested to reduce MAC overhead (*e.g.* [1], [2], [3], [13]); a series of novel packet recovery schemes (*e.g.*, [16], [10], [24], [14]) have been proposed to combat wireless losses; and a variety of rate adaptation algorithms (*e.g.*, [4], [18], [14]) have been developed to automatically adapt PHY sending rate according to the current link condition. While each of these existing techniques are useful, each alone is insufficient and there exist strong interactions between these seemingly orthogonal techniques. We now explain the relationships between these techniques.

Interactions between partial packet recovery and jumbo frames: A natural way to boost network throughput is to use a large frame size, since the MAC overhead for every frame remains constant and its relative overhead becomes smaller when a larger payload size is used. However, the loss rate of a frame tends to increase with frame size. As a result, even though sending a jumbo frame boosts network throughput under no-loss scenarios, its performance would suffer significantly under either collision losses or inherent wireless medium losses. Interestingly, partial packet recovery schemes help reduce the impact of losses so that it enables jumbo frames to achieve good performance under both low loss and high loss environments.

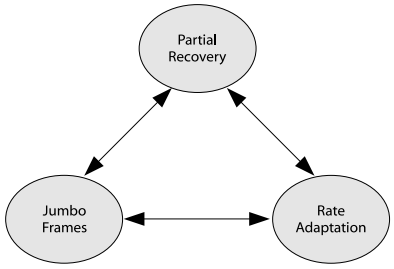


Fig. 1. Interactions between partial packet recovery, jumbo frames, and rate adaptation.

In addition, the use of jumbo frames also improves the effectiveness of partial packet recovery, because it reduces the relative cost of RTS/CTS, which helps to significantly reduce collision losses. Partial recovery techniques work best if the number of erroneous bits in a frame is small, whereas collisions tend to result in large erroneous bits. Therefore reducing collision losses makes it easier for partial packet recovery to succeed.

Interactions between partial packet recovery and rate adaptation: Traditional auto-rate adaptation schemes adapt the transmission rate according to the frame loss rate, which does not reflect the actual loss rate after partial recovery. Therefore the data rates these schemes select tend to be overly conservative when partial recovery is used [14]. In comparison, partial packet recovery reduces the effective data loss rate so that higher transmission rates can be used. In addition, increased transmission rate reduces the medium occupancy duration and hence reduce contention losses, which further improves the success of partial packet recovery.

Interactions between jumbo frames and rate adaptation: As the effectiveness of rate adaptation schemes improves, higher transmission rates are more likely to be chosen.

However, most MAC overhead (*e.g.*, transmission time of RTS/CTS, preamble, DIFS, SIFS) take constant time and its relative overhead compared to useful payload transmission time increases with the transmission rate. For example, consider a transmission of 1500-byte UDP packet in IEEE 802.11a. The MAC overhead under no RTS/CTS is 12% for 6 Mbps and increases to 43% for 54 Mbps, and the corresponding overheads under RTS/CTS are 16% and 53%, respectively. As jumbo frames reduce this relative overhead, the benefit increases with increased transmission rates, and effective rate adaptation helps to maximize the benefit of jumbo frames and vice versa.

Overview: Based on the above insights, in this paper we propose Fast Resilient Jumbo frames (FRJ). FRJ combines seemingly orthogonal techniques in order to increase performance and robustness in wireless LANs. FRJ uses jumbo frames to boost network throughput under good channel conditions and uses partial packet recovery to efficiently recover packet losses under bad channel conditions. It further uses partial recovery aware rate adaptation to maximize effective throughput under partial recovery.

We implement FRJ using the Madwifi driver [15] and Click toolkit [5]. Using real implementation and testbed experiments, we demonstrate that FRJ achieves efficient and resilient performance in wireless LANs. Its improvement over the existing schemes is 10-36% under a single flow and increases to 10-161% under multiple flows.

To summarize, we make the following contributions:

- We show the interactions between the three techniques and develop the FRJ protocol to exploit the synergy between these schemes.
- We leverage partial-packet recovery techniques to support jumbo frames and achieve high throughput under both low loss and high loss conditions.
- We advocate the use of RTS/CTS with jumbo frames to mitigate hidden terminals in multiple flow environments and reduce the relative overhead incurred for each packet transmission.
- We develop a prototype implementation and use testbed experiments to demonstrate FRJ's effectiveness.

The rest of the paper is organized as follows. In Section II, we survey related work. We present our design of FRJ in Section III. We describe our evaluation methodology in Section IV and performance results in Section V. We conclude in Section VI.

II. Related Work

As mentioned earlier, existing literature mainly focuses on individual design space. In this section, we give a brief overview of recent work in these areas.

Jumbo frames: Using jumbo frames to boost wireless network performance has received increasing attention from industry. Original approaches to frame aggregation included Atheros' Super G [1] fast framing and Texas Instruments' frame concatenation feature [2]. These proprietary optimizations combine multiple packets into a single frame. Unlike FRJ, they require specific hardware support. These works and others lead to frame aggregation in 802.11n standard [3]. Similar to FRJ, 802.11n can support block acknowledgements for each combined packet within an aggregated frame. In these schemes, even when a few bits of a packet are corrupted, the entire packet needs to be retransmitted. In contrast, FRJ only needs to retransmit the corrupted segments of a packet, which reduces overhead. In addition, FRJ is a software-based solution and compatible with existing 802.11a/b/g chip-sets, so that we can benefit immediately without hardware modifications.

Partial packet recovery: To protect against wireless link losses, a number of partial packet recovery schemes have been proposed recently. Miu et al. [16] develop MRD, which leverages multiple receivers to recover corrupted packets. When the same packet received at multiple receivers differs in one or more blocks, MRD exhaustively searches over all possible block combinations to find the one that passes the packet checksum. As MRD, SOFT proposed by Woo et al. [24] also takes advantage of multiple receivers for packet recovery. Different from MRD, SOFT has an efficient combining strategy that exploits the physical layer information to maximize the likelihood of packet recovery. Kyle et al. [10] describe the PPR scheme in which a single receiver performs partial packet recovery (PPR). In this scheme, the receiver leverages the confidence information at the physical layer to identify bits with high uncertainty and requests retransmission of these bits. By exploiting physical layer information in software defined radio, PPR out-performs segment-based partial recovery, proposed by Ganti et al. [6], however its performance improvement over the latter is usually around 25%. Like PPR, FRJ uses a single receiver but can be extended to multiple receivers. These previous works all require hardware changes. More recently, Lin et al. [14] propose ZipTx – a software technique based on retransmitting parity codes of corrupted packets. Similar to FRJ, this work also modifies SampleRate [4] to maximize the correct-byte throughput. Like ZipTx, our work is also a software solution and can be implemented directly on existing commodity hardware and software platforms. On the other hand, when PHY layer information is available, FRJ can benefit from it to achieve even higher gain. In addition, different from the existing partial packet recovery schemes, FRJ improves their effectiveness by using jumbo frames.

Rate adaptation: Rate adaptation has received significant research attention and various rate adaptation algorithms have been developed [7], [22], [20], [12], [4], [23], [11], [18]. For example, [18] is the rate adaptation algorithm used in the MadWiFi driver. It uses long-term loss rate estimation and threshold to determine rate changes. SampleRate [4], proposed

by Bicket et al., probes the performance at a random rate every 10 frames, and selects the rate that minimizes expected transmission time including retransmission time. Based on previous works [4], [23] and our own experiments, SampleRate out-performs [18] so we use it as the baseline scheme. More recently, Wong et al. [23] identify the limitations of existing design guidelines for rate adaptation. Based on their observations, they develop Robust Rate Adaptation Algorithm (RRAA), which uses short-term loss ratio to opportunistically change rate and incorporates an adaptive RTS filter to prevent collision losses from reducing data rate. All these existing rate adaptation schemes adapt rate according to frame loss rate. When partial packet recovery is used, the frame loss rate overestimates the actual loss rate experienced by data traffic and causes an unnecessarily low transmission rate to be used. FRJ overcomes this limitation by developing partial recovery aware rate adaptation.

III. Our Approach

In this section, we describe our approach to Fast Resilient Jumbo frame (FRJ) in detail. We focus our description and evaluation on single-hop wireless LANs, but the approach is applicable to multi-hop wireless networks. FRJ consists of two core components: (i) resilient jumbo frames that apply partial recovery to jumbo frames, and (ii) partial recovery aware rate adaptation.

A. Resilient Jumbo Frames

1) Overview: The use of jumbo frames allows us to effectively reduce MAC-layer overhead, thereby achieving high throughput. We modify the MadWiFi driver [15] to support resilient jumbo frames. While our current implementation supports jumbo frames of size up to 3000 bytes due to the limitation in hardware abstraction layer (HAL), such limitation is not inherent and we expect a larger throughput improvement with even larger frame sizes.

As mentioned in Section I, jumbo frames alone are insufficient because they are subject to higher losses in wireless networks. To effectively address this issue, we leverage partial packet recovery techniques to make jumbo frames more resilient to such losses. In particular, we adopt a segment based partial packet recovery scheme described by Ganti et. al in [6], though our design can easily benefit from other partial recovery schemes such as PPR [10] or the systematic codes used in ZipTx [14]. The idea of segment-based recovery is to divide a frame into smaller chunks, each having their own CRC. Upon data corruption, only the corrupted chunks need to be retransmitted to recover the frame, thus saving the overhead of retransmitting the complete frame.

Figure 2 shows our data frame format. It consists of a frame header and a series of segments, each of which contains 32-bit segment CRC and segment payload. The frame header includes frame ID, frame type, segment bitmap, frame length, and frame header CRC. The frame ID is an auto-incremental field, which the sender maintains per destination MAC address it communicates with. The type field is used by the receiver to distinguish between frame types. In our implementation, we use four different frame types - data frame, ACK frame, probe frame and probe response frame. The segment bitmap indicates which segments are present in the current frame. Our implementation use 30 segments per frame so a 32-bit bitmap is sufficient. A bit value of 1 at position i indicates that $segment_i$ is present. Consequently, the initial transmission of a frame will have all the bits set to 1, and the retransmission frame will have 1's only at the bits corresponding to retransmitted segments. A sender packs a data frame from the upper layer according to this format and hands it over to the MAC layer for transmission or retransmission. When a frame arrives at a receiver, if either the preamble is corrupted or the frame header does not pass its CRC check, the entire frame is lost. The likelihood of such losses is generally low due to small header size. Otherwise, the receiver extracts the segments that pass the segment CRC check and informs the sender of the correctly received segments. This will trigger the sender to retransmit unacknowledged segments. When a retransmission arrives, the receiver combines the correctly received new segments with already received segments of the same frame. After the entire frame is received correctly, the frame is then passed to the upper layer.

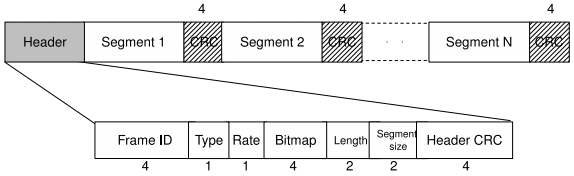


Fig. 2. Data frame format.

2) Receiver Feedback: Receiver feedback in FRJ uses a combination of MAC-layer ACKs and 2.5-layer ACKs. The MAC-layer ACKs are used because the adjustment of backoff window in IEEE 802.11 depends on the presence of synchronous ACKs at MAC-layer [16]. Moreover, the synchronous ACKs at the MAC-layer are more reliable and efficient than 2.5-layer ACKs since they are more compact and designed to avoid collision with nearby transmissions [16]. Therefore MAC-layer ACKs allow senders to quickly remove successfully received frames from its retransmission queue.

In order to support partial recovery, we further use 2.5-layer ACKs. These ACKs are generated after 100 ms or receiving 64 frames since the last ACK time, whichever comes first. These ACKs are cumulative in order to reduce the ACK overhead and minimize the impact of ACK losses. To further improve their reliability, they are transmitted using MAC-layer unicast with a retry count of 16, the maximum retry supported in MadWiFi [15]. To improve their responsiveness, we disable

binary backoff on 2.5-layer ACK frames by setting $CW_{max} = CW_{min}$. Since these ACKs are sent infrequently (*e.g.*, 10 per second), disabling binary backoff has negligible impact on competing data traffic while significantly improving the ACK responsiveness and reducing unnecessary retransmissions.

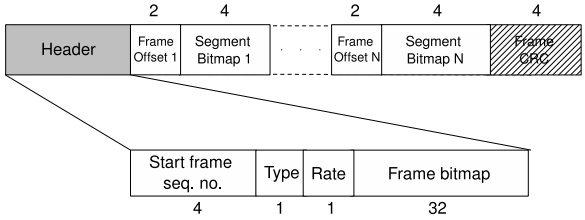


Fig. 3. 2.5-layer ACK frame format.

The 2.5-layer ACKs, whose format is shown in Figure 3, contain frame and segment status, indicating which frames and which segments in those frames are correctly received. To reduce the 2.5-layer ACK overhead, a sender aggregates the status of up to 256 frames into one 2.5-layer ACK, which includes start sequence number ($start$) and 256-bit bitmap ($Bitmap_f$). All the frames up to $start$ are assumed to be received completely correctly. The bits in frame bitmap field indicates the status of a frame: the i-th bit in $Bitmap_f$ is 1 if and only if all segments in the $start + i$-th frame are received correctly, and is 0 when the frame is lost partially or completely. For the partially received frames (these frames have $Bitmap_f$ with a 0 bit), the ACK also reports the status of their segments using tuples of $(offset, bitmap_s)$, where $offset + start$ is the frame ID and $bitmap_s$ is the status of its segments in that frame. The i-th bit in $bitmap_s$ is 1 whenever the i-th segment is received correctly and 0 otherwise. The use of offset allows us to omit the completely lost frames in the segment ACK. We update $start$ so that the largest received packet is no more than $start + 256$. This implies that it is possible that a node may not have received all packets up to $start$ even though it assumes so. The likelihood of such occurrence is low since the bitmap size of 256 is generally large enough even under the highest data rate. Moreover, FRJ is designed to provide best-effort reliability and leaves the upper-layer to ensure full reliability if needed.

3) Retransmission: When a frame is not acknowledged by either a MAC-layer ACK or a 2.5-layer ACK, it requires retransmission. In order to allow partial packet recovery, we disable MAC layer retransmission of data frames by setting the MAC retry count to 0, and retransmit the frames at the 2.5-layer.

The retransmissions can be triggered by either 2.5-layer ACKs or retransmission timeout. A 2.5-layer ACK will cause (partial or complete) retransmission of frame i if (i) it is the first retransmission and either some frames with sequence number higher than i or some segments in frame i are acknowledged, or (ii) it is not the first retransmission and the ACK acknowledges some new segments in frame i. The reason for different treatment between the first and subsequent retransmissions is that the first data transmissions in IEEE 802.11 is in-order delivery (*i.e.*, frames with lower sequence number are received earlier), but this is not the case for subsequent transmissions (*e.g.*, a retransmitted frame at 2.5-layer with smaller sequence number can arrive later than those with larger sequence numbers).

The other retransmissions are triggered by retransmission timeout. We use a standard approach to estimate retransmission timeout (RTO), similar to TCP [19]. Specifically, for every frame that has not been retransmitted, a node measures the time difference between when the frame was transmitted and when the corresponding 2.5-layer ACK was received. Let T denote the measured round-trip time (RTT) of the current frame. Then the node updates its RTO based on smoothened RTT and RTT variance as shown in Figure 4. RTO is initialized based on the PHY transmission data rate. Our evaluation uses $K = 4$, $\alpha = 1/8$, and $\beta = 1/4$ as in [8].

```
if (T is the first RTT measurement)
  SRTT = T;
  RTTVAR = T/2;
  RTO = SRTT + K * RTTVAR;
else
  RTTVAR = (1 − β) × RTTVAR + β × |SRTT − T|;
  SRTT = (1 − α) × SRTT + α × T;
  RTO = SRTT + K * RTTVAR;
end
```

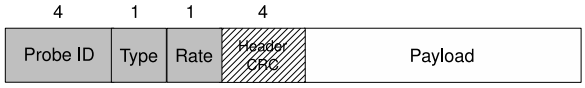
Fig. 4. Estimation of RTO.

B. Partial Recovery Aware Rate Adaptation

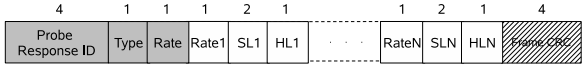
Rate adaptation is critical to the performance of IEEE 802.11 networks. Existing rate adaptation schemes identify the optimal rate based on the frame loss rate. With partial recovery, the frame loss rate over-estimates the actual loss rate experienced by the data traffic, thereby causing traditional rate adaptation schemes to select lower transmission data rate than necessary [14]. In order to fully exploit the benefit of a partial recovery scheme, it is necessary to design a partial recovery aware rate adaptation scheme.

Designing a partial recovery aware rate adaptation scheme poses the following challenges. First, how to accurately and efficiently estimate channel condition at various data rates? Second, how to select the rate that maximizes throughput under partial recovery? This requires us to estimate throughput under partial recovery based on loss statistics. Below we describe our design to address both issues.

1) Estimation of channel condition: To estimate channel condition, a sender periodically broadcasts probes at different data rates. Figure 5(a) shows the probe frame format. The probe ID field is maintained per rate and incremented on every probe, so gaps in probe IDs indicate loss of probe frames due to corrupted header (including preamble). The receiver



(a) Probe frame.



(b) Probe response frame.

Fig. 5. Probe and probe response frame format.

then estimates the channel condition using header loss rate (HL) and segment loss rate (SL), where HL is defined as the fraction of frames lost due to header corruption (or not receiving the packet) and SL is the fraction of corrupted segments. The estimates of HL_r and SL_r (*i.e.*, HL and SL values for rate r that a probe was received) are then communicated to the sender in a probe response packet, whose format is shown in Figure 5(b).

Since the loss of probe response may cause the sender to use incorrect estimates, it is important to make the response reliable. To provide high reliability for the probe response frame, it is transmitted via MAC-layer unicast with the maximum retry count of 16, the maximum in MadWiFi. Furthermore, whenever the maximum MAC-layer retry count is reached, the response frame is further retransmitted at 2.5-layer, thereby achieving full reliability of probe response.

In addition, in order to ensure that the sender receives the probe response in a timely fashion, we disable binary backoff on the probe response by setting $CW_{max} = CW_{min}$. This improves the responsiveness of probe responses with no impact on competing traffic due to their very low frequency (*e.g.*, once per 5 seconds) and allows for quick rate adaptation at the sender.

To limit the probing overhead, a sender transmits probes at three data rates (at most): (i) its current data rate ($CurrRate_r$), (ii) one data rate below its current data rate ($CurrRate_r^-$), and (iii) one data rate above its current data rate ($CurrRate_r^+$). To further reduce probing overhead, we could optionally omit probing at the current data rate and use existing traffic to estimate channel condition. A sender uses the probing frequency of 5 probes per second for each data rate, and its receiver sends probe responses every 5 seconds containing HL and SL estimates at all three data rates. In case no probes have been received over the last 5 seconds at rate r, the receiver sends a default probe response, which contains $HL_r = 1$ and $SL_r = 1$. We use 5 probes per second because our measurement shows that they give accurate loss rate estimation without incurring much overhead (*i.e.*, the loss rate estimation error decreases fast with increasing probing frequency below 5 probes/second and then tapers off afterwards).

2) Rate Selection: Given the loss estimates at different data rates, the sender estimates throughput based on HL_r and SL_r, and selects the data rate that yields the highest throughput using equations in Figure 6.

Specifically, let $Data_i$ denote time to send the i-th data transmission (*e.g.*, 1st transmission indicates the original transmission of the data frame, and 2nd transmission indicates the first retransmission of the frame, etc.), NS_i denote the number of segments in i-th transmission, P_i denote the probability of sending i-th transmission, and HS denote header size. $useRTS$ is 1 only when RTS/CTS is enabled. Contention window (CW), preambleTime, slot time, DIFS, SIFS, RTS, and CTS duration are as specified in the IEEE 802.11 standard [17].

First, to compute the total transmission time of a frame (including all retransmissions), we observe that the MAC/PHY overhead does not change while the data frame size changes depending on how many segments in the previous transmission are lost. The expected time spent in i-th transmission is P_i multiplied by the sum of the overhead and the i-th data transmission time. Therefore the expected total time spent in transmitting a data frame is sum over all i's, where $i = 1..MaxRetries + 1$. This is shown in Equation 1, where

$$RTSOverhead = RTS + SIFS + CTS + SIFS$$

$$DATA = preambleTime + \frac{(HS + NS_i \times segmentSize)}{rate}$$

and $Backoff = \frac{CW_{min}}{2} \times slotTime$. CW_{min} is used to compute average backoff time because the MAC retry count is set to 0 to allow partial recovery in FRJ.

To compute P_i, we note that $P_i = 1$ for the first transmission since each frame should be transmitted at least once. When $i > 1$, the transmission is sent when either the header or at least one segment in the previous transmission is corrupted. This observation leads to Equation 2.

To compute NS_i (*i.e.*, the number of segments in i-th transmission), we observe $NS_1 = 30$, since the initial data frame contains 30 segments. For the subsequent retransmissions, if the previous transmission is lost due to header corruption, the entire frame should be retransmitted; otherwise we retransmit the lost segments. The former is $HL_r \times NS_{i-1}$ and the latter is $(1 - HL_r) \times SL_r \times NS_{i-1}$, where NS_{i-1} is the total number of segments in $i - 1$-th transmission. Therefore, we have Equation 3.

Finally, we estimate throughput based on the following insight: $NS_{MaxRetries+2}$ is the expected number of segments that will be lost when a frame is retransmitted up to MaxRetries (plus 1 original transmission). Therefore the expected data transmitted successfully is $(NS_1 - NS_{MaxRetries+2}) \times SegmentSize$. This size divided by total transmission time T gives expected throughput, as shown in Equation 4.

The rate selection is performed either upon receipt of probe response (sent once every 5 seconds) or till 6-second elapse since the last time rate selection was performed to

$$T = \sum_{i=1..MaxRetries+1} P_i \times (Backoff + DIFS + DATA + SIFS + ACK + useRTS \times RTSOverhead) \quad (1)$$

$$P_i = \begin{cases} 1 & i = 1 \\ P_{i-1} \times (HL + (1 - HL) \times (1 - (1 - SL)^{NS_{i-1}})) & otherwise \end{cases} \quad (2)$$

$$NS_i = \begin{cases} 30 & i = 1 \\ NS_{i-1} \times (HL + (1 - HL) \times SL) & otherwise \end{cases} \quad (3)$$

$$Throughput = (NS_1 - NS_{MaxRetries+2}) \times SegmentSize/T \quad (4)$$

Fig. 6. Throughput calculation in FRJ

accommodate the delay in (re)transmission of probe response. In case the probe response is not received before the rate has to be chosen, the sender estimates HL and SL at the current rate based on the performance of actual data traffic. In addition, since loss rate tends to be lower for a lower data rate, we conservatively assume $HL = 0$ and $SL = 0$ at the rate immediately below the current rate, and apply the above throughput estimation to select the one that gives higher throughput. The rate is re-selected as soon as the next probe response is received or another 6 seconds elapses, whichever comes first.

IV. Experimental Methodology

We evaluate FRJ using testbed experiments. We implement FRJ using the Madwifi driver [15] and Click toolkit [5]. This allows us to understand the performance benefits in real networks. Our testbed consists of 24 DELL Dimension 1100 PCs, located on two adjacent floors of an office building as shown in Figure 7. Each machine has a 2.66 GHz Intel Celeron D Processor 330 with 512 MB of memory and is equipped with a 802.11 a/b/g NetGear WAG511 wireless card. For all our experiments, we use 802.11a to avoid interference from our campus networks that use 802.11b/g. Every node uses an initial PHY transmission rate of 24 Mbps and a transmission power of 18 dBm. We randomly pick source and destination pairs from the testbed and establish CBR transfer with saturated demand between them. We measure throughput over a 60-second transfer for each flow, and compare total throughput, per flow throughput, and Jain's fairness index, which is defined as $(\sum x_i)^2/(n * \sum {x_i}^2)$, where x_i is the throughput of flow i and n is the total number of flows in the network [9].

We compare the following schemes in our evaluation:

1. SampleRate using 1500-byte frames (SR/1500-bytes): This algorithm was developed by Bicket [4]. It is shown to be one of the most competitive rate adaptation schemes [23].
2. SampleRate using 3000-byte frames (SR/3000-bytes): This is the same as the above except it uses a jumbo frame size of 3000 bytes. This is similar to using the fast frame feature in Atheros Super G [1].
3. FRJ: This is the scheme described in Section III. It uses 3000-byte frames and 30 segments per frame for partial recovery.

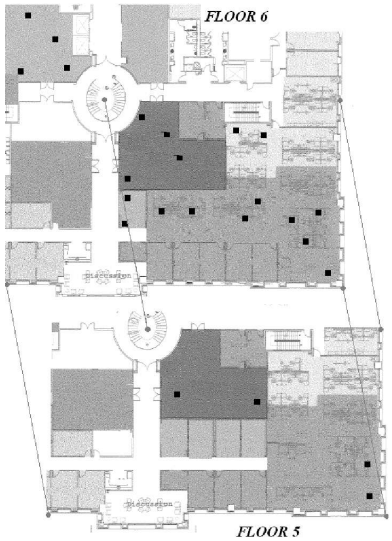


Fig. 7. Node placement in the testbed topology.

All the above schemes retransmit a frame up to 7 times (the default 802.11 retransmission count). In addition, they can work with or without RTS/CTS. In general, when the background traffic is low, the schemes without RTS/CTS yield better performance because of lower overhead and more opportunities to send data frames. The latter is because when using RTS/CTS under lossy links, data frames cannot be sent out until both RTS and CTS frames are successful. As the background traffic increases, the performance under RTS/CTS improves because it prevents data collisions arising from either hidden terminals or the random backoff counter expiring at the same time.

V. Experimental Results

A. Single flow

We first compare the different schemes by plotting the CDF distribution of throughput using a single flow without RTS/CTS. As shown in Figure 8, the performance benefit of FRJ over SR is largest under moderate link conditions. For example, 20-th percentile throughput is 0.68 Mbps for both SR/1500-bytes and SR/3000-bytes and 1.11 Mbps for FRJ; and 80-th percentile throughput is 14.17 Mbps for SR/1500-bytes, 16.93 Mbps for SR/3000-bytes, and 23.81 Mbps for FRJ, resulting in improvement of 40.6% to 68.0%. The larger improvement under moderate link conditions is because under highly reliable links all schemes can effectively utilize bandwidth and in highly lossy links all schemes incur severe preamble losses and suffer. Under moderate link conditions, many losses come from a small number of corrupted segments per frame, which makes the segment-based partial recovery scheme effective.

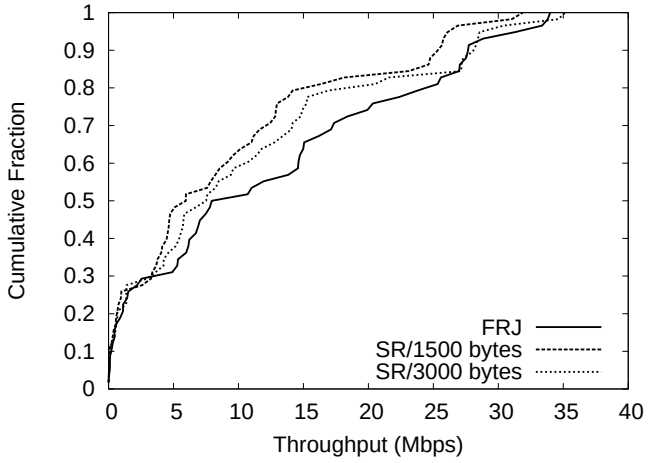


Fig. 8. CDF of both SR protocols and FRJ over all runs in the single flow experiments.

B. Multiple flows

Next we evaluate the performance of FRJ under multiple flow settings. We vary the number of simultaneous flows from 1 to 8, and repeat the experiments 10 times for each number of flows. We compare FRJ with SR/1500-bytes and SR/3000-bytes in terms of total throughput, throughput distribution, and Jain's fairness index.

Figure 9 shows the average total throughput versus the total number of flows with and without RTS/CTS, where the error bars denote the standard deviation of the sampled mean. The standard deviation is generally quite high because we randomly choose flows – some flows are across links with high delivery rate and get high throughput while others are across lossy links and experience low throughput. We make the following observations.

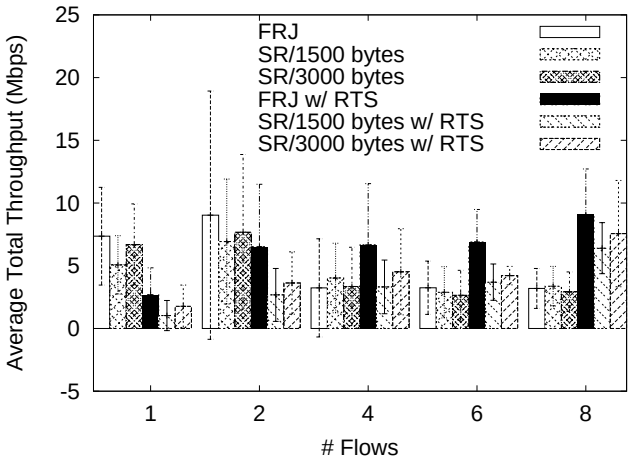


Fig. 9. Average total throughput versus the number of flows.

First, comparing the performance between the schemes under RTS/CTS with their counterparts under no RTS/CTS, we see that the schemes without RTS/CTS perform better under 1 and 2 flows and the schemes with RTS/CTS perform better under more flows. This is expected since the use of RTS/CTS prevents collisions arising from hidden terminals or the backoff counter expiring at the same time and such collisions are more prevalent under a large number of flows.

Second, FRJ without RTS/CTS performs the best among all the schemes under 1 and 2 flows, and FRJ with RTS/CTS performs the best under more flows. In other words, with an appropriate RTS/CTS configuration, FRJ consistently outperforms all SR schemes. Compared with the best performing scheme for each configuration among SR/1500-bytes and SR/3000-bytes with and without RTS/CTS, the benefit of FRJ ranges from 10% to 64%.

Furthermore, in all the runs with RTS/CTS, FRJ consistently out-performs both SR/3000-bytes and SR/1500-bytes. Its improvement over SR/1500-bytes ranges from 42% to 161%, and over SR/3000-bytes ranges from 20% to 80%. In addition, SR/3000-bytes out-performs SR/1500-bytes due to reduced MAC/PHY-layer overhead.

In comparison, the performance difference across various schemes is less pronounced under no RTS/CTS, because in this case collision losses increase and reduce the effectiveness of the jumbo frame, partial recovery, and rate adaptation scheme. Specifically, collision losses often result in corruption of headers or a large fraction of payload, which are harder to recover. Moreover, FRJ's rate adaptation scheme may respond to collision losses by unnecessarily reducing its transmission rate. Techniques to adaptively configure RTS/CTS [23] and diagnose the reason for wireless losses [21] would be very helpful to further improve the effectiveness of FRJ.

We further compare throughput with RTS/CTS using a CDF of per flow throughput over all multiple flow runs,

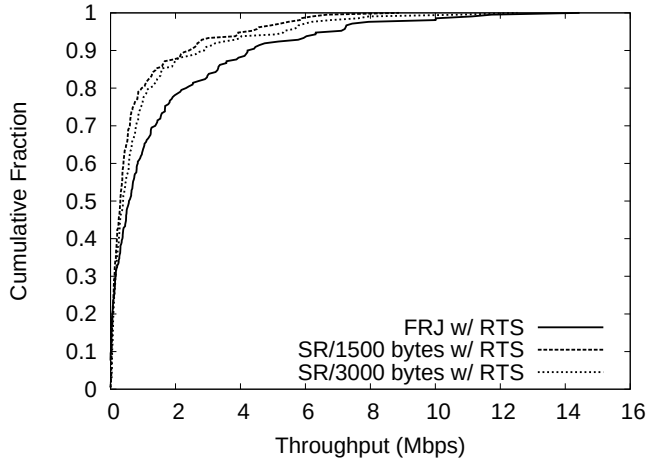


Fig. 10. CDF of per-flow throughput with RTS/CTS over all the multiple flow experiments.

as shown in Figure 10. The median throughput are: 0.30 Mbps for SR/1500-bytes, 0.38 Mbps for SR/3000-bytes, and 0.57 Mbps for FRJ. The average per-flow throughput over all runs (not shown in the figure) are 0.84 Mbps for SR/1500-bytes, 1.05 Mbps for SR/3000-bytes, and 1.68 Mbps for FRJ, which translates to 100% improvement over SR/1500-bytes and 60% improvement over SR/3000-bytes. These numbers further demonstrate the effectiveness of FRJ in achieving good performance.

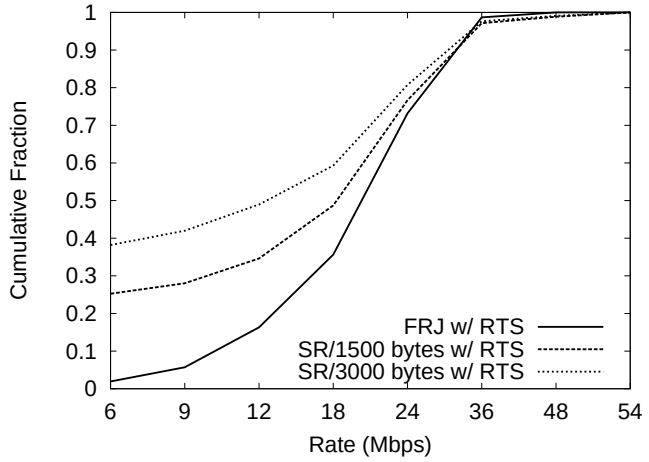


Fig. 11. CDF of selected transmission rate with RTS/CTS over all the multiple flow experiments.

In order to better understand the performance benefits of FRJ, we compare the transmission rate used by different schemes with RTS/CTS. Figure 11 shows a CDF of the transmission rates used for each scheme. The transmission rate used in FRJ is generally higher than both SR/1500-bytes and SR/3000-bytes. For example, 25% of SR/1500-bytes frames and 40% of SR/3000-bytes frames use the lowest transmission rate, while only 2% of FRJ frames use the lowest rate. Moreover, the gap between different CDF curves is larger under low rates because in these cases links tend to be more lossy and partial recovery and partial recovery-aware rate adaptation are more useful. The average transmission rates (not shown in the figure) are: 20.61 Mbps for SR/1500-bytes, 17.75 Mbps for SR/3000-bytes, and 24.03 Mbps for FRJ. SR/3000-bytes uses lower transmission rate than SR/1500-bytes because the former experiences higher frame loss rate due to larger frame size and adapts to lower transmission rate, which offsets the benefit of jumbo frame. In comparison, FRJ is able to use both large frame and high transmission rates by using partial-recovery aware rate adaptation so that it out-performs both schemes.

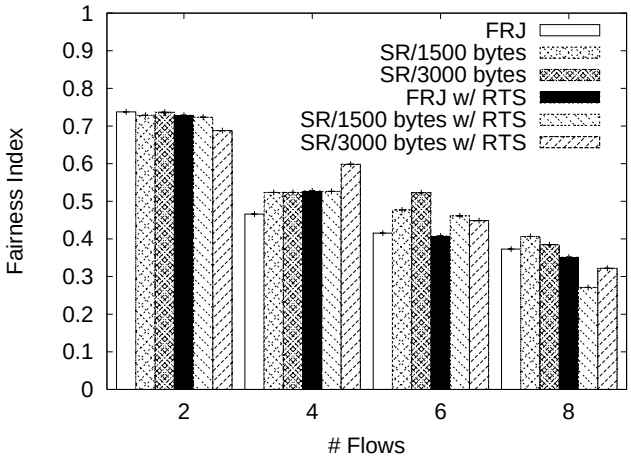


Fig. 12. Average Jain's fairness index versus the number of flows.

Finally, Figure 12 compares Jain's fairness index across different schemes. FRJ's fairness index is comparable to the other schemes – the difference in all cases is within 10% and in most cases is close to 0. This indicates that the performance benefits from FRJ do not come at the cost of some flows unfairly occupying the medium.

VI. Conclusion

In this paper, we develop Fast Resilient Jumbo frame (FRJ) to enhance the efficiency and robustness of wireless performance. We explore the interplay between three seemingly unrelated technologies and show effectively integrating them yields significant performance benefits in wireless LANs. FRJ uses jumbo frames with partial packet recovery to boost network throughput under good channel conditions, and efficiently recover packet losses under bad channel conditions. It further uses partial recovery aware rate adaptation to maximize effective throughput under partial recovery. Our evaluation, based on a real implementation and testbed experiments, shows that FRJ consistently out-performs the existing schemes under different channel and traffic conditions.

The design space for integrating jumbo frames, rate adaptation and partial-packet recovery is large, and many other

design choices are possible. The goal of this paper is to show that there exists synergy between jumbo frame, rate adaptation and partial-packet recovery, and it is important to exploit such synergy to maximize effectiveness.

Moving forward, we are considering the following enhancements to further improve performance. First, the effectiveness of FRJ closely depends on the partial recovery scheme. For simplicity and ease of deployment, we use one of the simplest partial recovery techniques – segment-based partial recovery, and already observe a large benefit. The effectiveness of FRJ can further improve when applied to a more effective partial recovery scheme. In particular, the postamble technique in PPR [10] is effective to reduce header losses and increase the success rate of partial recovery; coding techniques are also useful to partial recovery and reduce control overhead [14]. Second, as shown in Section V-B, the RTS/CTS configuration is important to achieve high performance. This configuration is especially important under lossy links, because a data frame can be transmitted only after both RTS and CTS succeed and RTS/CTS loss can significantly reduce the number of data frames to be transmitted. We are interested in exploring dynamically configurable RTS/CTS. Third, FRJ can be directly applied to multihop wireless networks by improving the performance over each hop. Furthermore, to maximize its effectiveness, it would be useful to design FRJ-aware routing metrics to select paths that maximize effective throughput after partial recovery.

Acknowledgments This work is supported in part by NSF Grants CNS-0546755 and CNS-0627020.

REFERENCES

[1] Atheros Super G. http://www.atheros.com/pt/whitepapers/atheros_superg_whitepaper.pdf.

[2] TI G-plus (802.11g+) Performance-Enhancing Technology. http://focus.ti.com/pdfs/bcg/80211g_plus_wp.pdf.

[3] I. 802.11n Working Group. Wireless LAN medium access control (MAC) and physical layer (PHY) specification, 2007.

[4] J. C. Bicket. Bit-rate selection in wireless networks. *M.S. Thesis, MIT*, Feb. 2005. http://pdos.csail.mit.edu/papers/jbicket-ms.pdf.

[5] Click. http://pdos.csail.mit.edu/click/.

[6] R. Ganti, P. Jayachandran, H. Luo, and T. Abdelzaher. Datalink streaming in wireless sensor networks. In *Proc. of ACM SenSys*, Nov. 2006.

[7] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multihop wireless networks. In *Proc. of ACM MobiCom*, Jul. 2001.

[8] V. Jacobson and M. Karels. Congestion avoidance and control. In *Proc. of ACM SIGCOMM*, Nov. 1988.

[9] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Sep 1984.

[10] K. Jamieson and H. Balakrishnan. PPR: partial packet recovery for wireless networks. In *Proc. of ACM SIGCOMM*, Aug. 2007.

[11] J. Kim, S. Kim, S. Choi, and D. Qiao. CARA: collision-aware rate adaptation for IEEE 802.11 wlans. In *Proc. of IEEE INFOCOM*, Apr. 2006.

[12] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 rate adaptation: A practical approach. In *Proc. of ACM MSWiM*, Oct. 2004.

[13] M. Li, D. Agrawal, D. Ganesan, A. Venkataramani, and H. Agrawal. Bock-switched networks: A new paradigm for wireless transport. In *Proc. of USENIX NSDI*, 2009.

[14] K. C.-J. Lin, N. Kushman, and D. Katabi. Ziptx: Harnessing partial packets in 802.11 networks. In *Proc. of ACM MobiCom*, pages 351–362, New York, NY, USA, 2008. ACM.

[15] Madwifi. http://madwifi.org.

[16] A. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proc. of ACM MobiCom*, Aug. - Sept. 2005.

[17] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[18] ONOE rate control. http://madwifi.org/browser/trunk/ath_rate/onoe.

[19] V. Paxson and M. Allman. Computing TCP's retransmission timer. *IETF Internet DRAFT*, 2000. http://www3.ietf.org/proceedings/00jul/I-D/paxson-tcp-rto-01.txt.

[20] D. Qiao, S. Choi, and K. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. *IEEE TMC*, Oct. 2002.

[21] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee. Diagnosing wireless packet losses in 802.11: Separating collision from weak signal. In *Proc. of IEEE INFOCOM*, Apr. 2008.

[22] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. In *Proc. of ACM MobiCom*, Sept. 2002.

[23] S. H. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation in 802.11 wireless networks. In *Proc. of ACM MobiCom*, Sept. 2006.

[24] G. Woo, P. Kheradpour, and D. Katabi. Beyond the bits: Cooperative packet recovery using PHY information. In *Proc. of ACM MobiCom*, Sept. 2007.