Retransmission Rate Selection for Block-based Partial Packet Recovery

Haonan Lu, Shuaiyuan Zhou, and Zhenghao Zhang
Computer Science Department
Florida State University Tallahassee, FL 32306, USA

***Abstract*—In a Wi-Fi network, *partial packets* often exist which are packets with some errors. Recent works show that much better efficiency can be achieved by *repairing* such packets instead of retransmitting such packets. In this paper, we propose a rate selection scheme for the repair packets, refereed to as *ReSel*. ReSel is developed under the Maranello partial packet recovery framework and is motivated by two observations. First, the repair packet is sent shortly after the original packet and will likely experience similar channel conditions as the original packet. Therefore, given the recent failure on the original packet, a lower data rate should be used for the repair packet. Second, a Maranello receiver will send a NACK packet to the sender, which can be used to carry important information for the selection of the data rate. We build a rate selection table for ReSel with experimental data. We implement ReSel in the firmware and our experiments show that ReSel significantly improves the link throughput and reduces packet jitter. We also find that the repair packet is transmitted only once in the majority of the cases and hence ReSel is capable of selecting the appropriate data rate.**

I. Introduction

In a Wi-Fi network, *partial packets* often exist which are packets with some errors. The default 802.11 protocol retransmits such packets; however, recent works show that much better efficiency can be achieved by *repairing* instead of retransmitting such packets. Among the repair methods, block-retransmission has the lowest complexity, with which a packet is divided into blocks and only the corrupted blocks are retransmitted in a *repair packet*. Block-retransmission has been implemented in the firmware of commodity wireless cards such as the Broadcom 4306 card in Maranello [6].

In this paper, we study partial packet recovery under the Maranello framework and propose *ReSel* for the rate selection for the repair packet. Wi-Fi supports multiple data rates and lower data rates are typically more resilient to noise than higher rates. In the current Maranello, the repair packet is first transmitted at the same data rate as the original packet which we refer to as the *the default rate*. This strategy may be suboptimal because the default data rate has been attempted and *failed* on the original packet transmission. The repair packet is most likely transmitted within several milliseconds of the original packet such that it will likely experience similar channel conditions as the original packet. Given a failure in the recent past, the probability for a failure significantly increases. A new rate selection algorithm is therefore needed.

Rate selection has been studied extensively [1], [2], [3], [4], [5]. Typically, the rate selection algorithm runs in the device driver and determines the rate based on statistics such as transmission successes or failures and packet delivery time. The key difference between ReSel and the existing rate selection algorithms is that ReSel selects rate based on the *instantaneous* channel conditions while existing rate selection algorithms select rates based on the *average* channel conditions. As mentioned earlier, the repair packet is expected to be transmitted shortly after the original packet. On the other hand, other rate selection algorithms have to assign a rate to a packet which may be handed down by the upper layer at any time when the channel state information may not be available; the best rate in this case should be the rate that best matches the average channel condition to minimize packet loss. Therefore, ReSel and the existing algorithms solve different problems.

The main challenge in the design of ReSel is the unavailability of the channel state information (CSI). Ideally, if the CSI is available, ReSel can adopt a similar approach as in [1] by selecting a rate with a reasonably low loss ratio according to the CSI. However, most commodity wireless cards do not report the CSI to the firmware except the RSSI which is the average signal to noise ratio. While RSSI is usful, it is well-known that it is insufficient for determining the data rate because the packet loss is also determined by the fading condition, i.e., whether the fading is flat or non-flat. A non-flat fading channel results in uneven signal strengths across the OFDM subcarriers and higher packet loss ratio than a flat fading channel with the same RSSI.

Fortunately, we note that it is possible to extract additional information for rate selection under the Maranello framework because the sender actually knows the number of corrupted blocks in a packet. Intuitively, the more the corrupted blocks, the worse the channel conditions. For example, under the same RSSI, more corrupted blocks means a more non-flat fading channel for which a lower data rate may be needed. Therefore, ReSel takes two numbers as input, namely the RSSI and the number of corrupted blocks of the original packet. We design the rate selection strategy based on extensive experimental data and implement ReSel as an extension to Maranello. Because the same approach can be used for all data rates, we explain our approach using 36 Mbps as an example. Our experiments show that ReSel can achieve significant performance gain comparing to the original Maranello by selecting the appropriate data rate for the repair packet, because the repair packet is transmitted only once in the majority of the cases.

The rest of the paper is organized as follows. Section II

discusses the related works. Section III briefly introduces Maranello. Section IV describes ReSel. Section V gives the experimental results. Section VI concludes the paper.

II. Related Work

Rate selection is a classic topic for wireless networks [1], [2], [3], [4], [5]. As mentioned earlier, comparing to existing works, ReSel solves a different problem, focusing on the rate selection upon a loss event for the repair packet. The solutions in ReSel is also different because ReSel exploits information only available in the Maranello framework.

Partial packet recovery has also been studied extensively in recent years [6], [7], [9], [10], [8]. The existing approaches include using error correction codes [7], retransmitting the corrupted parts [6], [9], [10], or both [8]. We focus on the Maranello framework in this paper because it can be implemented in the firmware level without any involvement of the upper layers, and has several key advantages such as minimum disturbance to packet delay. To the best of our knowledge, the rate selection for the repair packet has not been studied before in existing works.

III. A Brief Introduction to Maranello

Maranello is implemented on the OpenFWWF platform [12] which is an open source firmware that enables programmable MAC on the Broadcom 4306 wireless card. In Maranello, packets are divided into blocks. On the first attempt, the sender transmits the original packet. If the receiver receives the packet correctly, it sends an 802.11 ACK. If the receiver receives a partial packet, it calculates the checksums of the blocks, and sends the checksums in a NACK packet to the sender. When the sender receives the NACK packet, it compares the locally computed checksums based on the original packet and the checksums in the NACK packet to find the corrupted blocks, and retransmits such blocks in the repair packet. In the current implementation, the repair packet is transmitted at the same rate as the original packet up to two times before being transmitted at other data rates.

IV. ReSel

ReSel is a simple modification to the original Marannello. At the receiver side, we modify the firmware to piggyback the measured RSSI of the original packet in the NACK packet. At the sender side, we modify the firmware to read the RSSI value and learn the number of corrupted blocks, which are used as input in a simple table lookup to determine the rate of the repair packet. The core of ReSel, clearly, is this rate table, which we build with experimental data and is explained in this section.

A. Clustering Cases into Bins

We denote the RSSI value as α and the number of corrupted blocks as β. Ideally, for each (α, β), we may run experiments and attempt all possible rates to find the best rate. The technical challenge is that the memory size in the wireless card processor is very limited and cannot hold much data for data collection. We therefore create bins for α and β values and collect data for each bin. We first divide the RSSI into three ranges: less than 20 dB, within 20 and 26 dB, and greater than 26 dB. Marannello repairs packets with up to 14 corrupted blocks. We further divide the number of corrupted blocks into 4 ranges. In total, we create 12 bins, which is within the capabilities of the processor. The range table is shown in Table 1.

	β range 1	β range 2	β range 3	β range 4
$\alpha < 20$	[1,2]	[3,5]	[6,8]	[9,14]
$20 \leq \alpha \leq 26$	[1,2]	[3,4]	[5,7]	[8,14]
$\alpha > 26$	1	[2,3]	[4,6]	[7,14]

TABLE 1
Range table

	36	24	18	12	6
$\alpha < 20$, $\beta \in [1,2]$	0.919	0.983	0.983	0.982	0.980
$\alpha < 20$, $\beta \in [3,5]$	0.788	0.970	0.974	0.975	0.972
$\alpha < 20$, $\beta \in [6,8]$	0.550	0.935	0.959	0.959	0.963
$\alpha < 20$, $\beta \in [9,14]$	0.288	0.829	0.962	0.964	0.953
$20 \leq \alpha \leq 26$, $\beta \in [1,2]$	0.836	0.928	0.940	0.936	0.931
$20 \leq \alpha \leq 26$, $\beta \in [3,4]$	0.693	0.907	0.922	0.937	0.928
$20 \leq \alpha \leq 26$, $\beta \in [5,7]$	0.481	0.852	0.929	0.921	0.913
$20 \leq \alpha \leq 26$, $\beta \in [8,14]$	0.378	0.779	0.905	0.929	0.935
$\alpha > 26$, $\beta = 1$	0.875	0.956	0.952	0.954	0.956
$\alpha > 26$, $\beta \in [2,3]$	0.787	0.951	0.954	0.958	0.955
$\alpha > 26$, $\beta \in [4,6]$	0.627	0.938	0.952	0.958	0.946
$\alpha > 26$, $\beta \in [7,14]$	0.539	0.904	0.962	0.959	0.945

TABLE 2
Packet Receiving Ratio Table

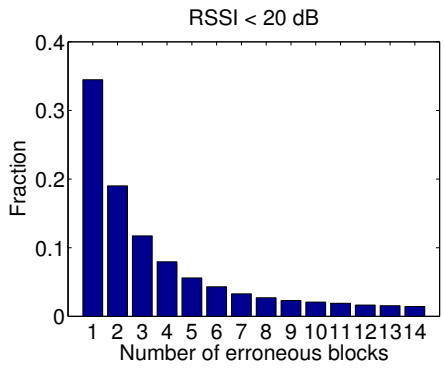
The RSSI range is selected because we find that they represent bad channel, functional channel, and good channel in typical indoor environments, respectively. The β range is also determined based on measurements. We show in Fig. 1 the number of corrupted blocks in different α ranges collected in 10 experiments. We can see that the distributions of β are different for different α ranges. We basically attempt to cluster β with similar values into the same group.

B. Packet Receiving Ratio (PRR) Measurements

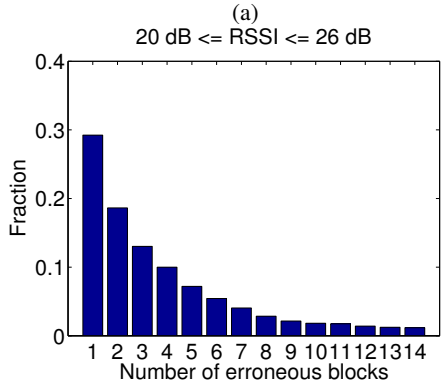
We conduct a measurement campaign to determine the best data rate for each bin. Basically, the sender tries the data rates in a round-robin manner and records the *Packet Receiving Ratio (PRR)* for each rate. To be more specific, when the sender receives a NACK packet, it finds the values of α and β, and finds the bin they belong to. The sender keeps a round-robin pointer for each bin and uses the rate pointed by the pointer as the rate for the repair packet. After the repair packet is sent, the pointer is updated. The set of rates include five 802.11g rates that are not higher than 36 Mbps, because the repair packet should not be transmitted at rate higher than the rate for the original packet. We record the PRR for each rate in each bin. Table 2 shows raw measurement results.

C. Determining the Rate Table

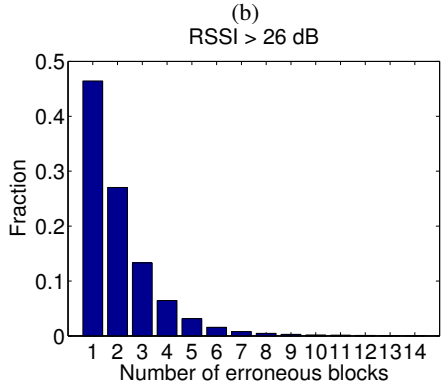
The PRR measurements contain interesting information. For example, it can be seen that within the same RSSI range,



(a)



(b)



(c)

Fig. 1. The distribution of the corrupted block number. (a). $\alpha < 20$. (b). $20 \leq \alpha \leq 26$. (c). $\alpha > 26$.

the PRRs vary significantly for different number of corrupted blocks. We suspect that this is because the channel fading conditions are different; channels with more subcarriers in deep fading tend to result in more corrupted blocks as well as lower PRRs. The clear correlation between PRR and the number of corrupted blocks confirms that the number of corrupted blocks is a useful input to the system.

We can also see that for the same RSSI range and corrupted block number range, the PRRs do not exhibit significant differences for rates 18 Mbps or lower; in fact the PRRs for such rates are all very high. Therefore, the lowest data rate we need to use for the retransmission is 18 Mbps. On the other hand, the PRRs for 36 Mbps are usually lower than desired. Therefore, we basically need to choose between 24 Mbps and 18 Mbps. We use 24 Mbps if the difference between the PRRs for the two rates is very small; otherwise we use 18 Mbps. The rate table is shown in Table 3.

$\alpha < 20$, $\beta \in [1, 2]$	24
$\alpha < 20$, $\beta \in [3, 5]$	24
$\alpha < 20$, $\beta \in [6, 8]$	24
$\alpha < 20$, $\beta \in [9, 14]$	18
$20 \leq \alpha \leq 26$, $\beta \in [1, 2]$	24
$20 \leq \alpha \leq 26$, $\beta \in [3, 4]$	24
$20 \leq \alpha \leq 26$, $\beta \in [5, 7]$	18
$20 \leq \alpha \leq 26$, $\beta \in [8, 14]$	18
$\alpha > 26$, $\beta = 1$	24
$\alpha > 26$, $\beta \in [2, 3]$	24
$\alpha > 26$, $\beta \in [4, 6]$	24
$\alpha > 26$, $\beta \in [7, 14]$	18

TABLE 3
RATE TABLE

The rate table in Table 3 determines the rate when a NACK is received. If no NACK is received after the original packet transmission, the rate table cannot be applied because there is no feedback from the receiver. In this case, ReSel always uses 18 Mbps for the retransmission of the packet. This is because if no NACK is received, the channel is likely in a poor condition such that low data rates should be used.

V. EVALUATION

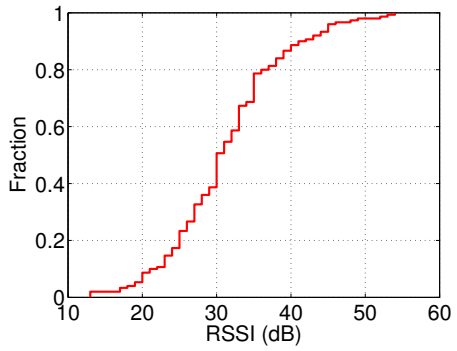
We evaluate ReSel with experiments. We set up one computer as the sender and one computer as the receiver, and run the iperf software [11] to collect the link throughput and jitter. We use the firmware to collect statistics such as *erasure ratio* and *partial ratio*, defined as the fraction of erasure packets and partial packets, as well as the *retry number*, defined as the number of transmissions of the repair packet. The experiments are repeated 150 times at randomly selected locations on 802.11 b/g channel 3. We fix the data rate to be 36 Mbps.

A. The Channels in the Experiments

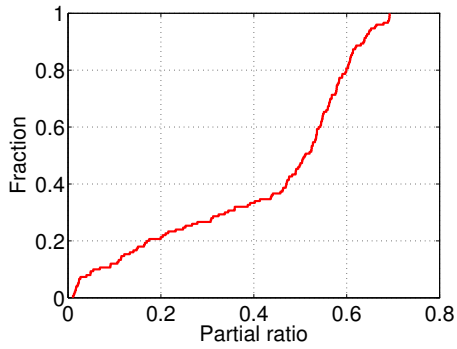
We show in Fig. 2 the channel conditions in our experiments, represented by the Cumulative Distribution Function (CDF) of the RSSI, the partial ratio, and the erasure ratio. We can see that our experiments include a wide variety of RSSIs and partial ratios. The erasure ratios tend to be small numbers because the link cannot be set up between the sender and the receiver if the erasure ratio is too high.

B. Throughput Comparison

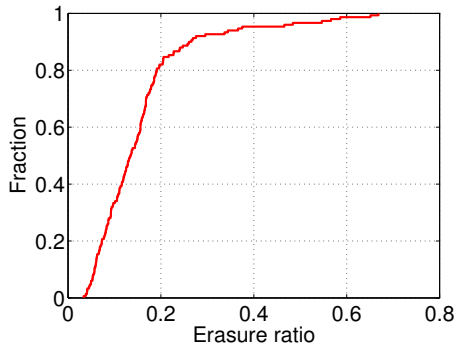
Fig. 3(a) shows the CDF of the link throughput of the original Maranello and ReSel. We can see that ReSel achieves significant gain over the original Maranello. The median throughput of ReSel and the original Maranello are 9.73 Mbps and 7.41 Mbps, respectively, with a gain of 31%. Fig. 3(b) shows the scattered plot of the original Maranello and ReSel. In the scattered plot, a point on the 45-degree line represents a link in which two compared schemes have the same throughput. We can see that ReSel outperforms the original Maranello in most of the links. This is remarkable in the sense that the gain is achieved by a very simple modification to the firmware.



(a)

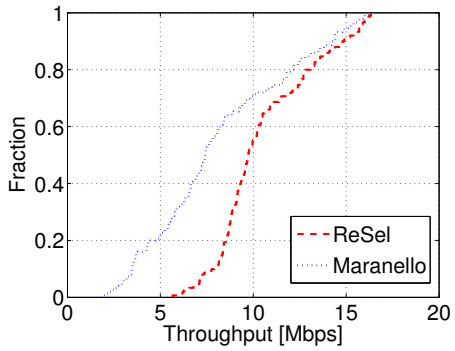


(b)

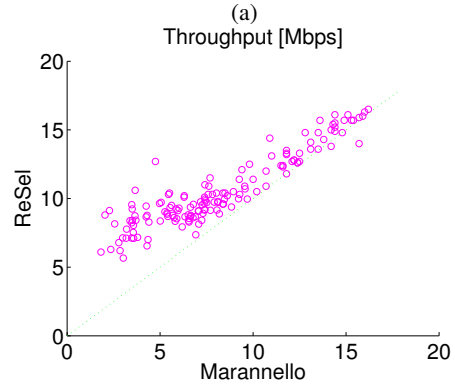


(c)

Fig. 2. Channel statistics. (a). RSSI. (b). Partial ratio. (c). Erasure ratio.



(a)



(b)

Fig. 3. (a). CDF of the throughput. (b). Scattered plot of the throughput.

C. Jitter Comparison

Fig. 4(a) shows the CDF of the jitter of the original Maranello and ReSel. We can see that the jitter of ReSel is much less than the original Maranello. The median jitter of ReSel and the original Maranello are 3.28 ms and 4.24 ms, respectively, with a reduction of 23%. Fig. 3(b) shows the scattered plot of the original Maranello and ReSel, where we can see that ReSel achieves less jitters than the original Maranello in most of the links.

D. Retry Number Comparison

The gain of ReSel over the original Maranello is due to the better selection of the data rates to reduce the number of retransmissions. Fig. 5(a) shows the CDF of the retry number of the original Maranello and ReSel. We can see that the retry number of ReSel is almost always very close to 1 while the retry number of the original Maranello are much larger. The median retry number of ReSel is 1.10 while the median retry number of the original Maranello is 1.47. We note that the minimum retry number is 1 because the repair packet must be sent at least once. With ReSel, in over 92% of the experiments, the retry number is below 1.2. In this sense, we have achieved our goal of selecting a good data rate because most repair packets are received in the first attempt in a wide variety of channel conditions. Fig. 5(b) shows the scattered plot of the original Maranello and ReSel, where we can see that ReSel uses less retransmissions than the original Maranello in most of the links.

E. Analyzing the Gain

To further analyze the gain, we show the throughput and jitter ratios of ReSel over the original Maranello as functions of the RSSI, the partial ratio, and the erasure ratio in Fig. 6, Fig. 7, and Fig. 8, respectively. We need to mention that we have added a small random perturbation within $[0, 0.2]$ to the RSSI values in Fig. 6, such that different experiments sharing the same RSSI reading will not overlap in the figure.

We can see that there does not seem to be a strong correlation between RSSI and the gains, because the throughput and jitter ratios seem to be rather uniformly distributed over all RSSI values. This confirms that RSSI alone is not a good indicator of the channel and the gains. On the other hand, the gains exhibit notable patterns with regarding to the partial ratio and the erasure ratio. As the jitter ratio can be explained

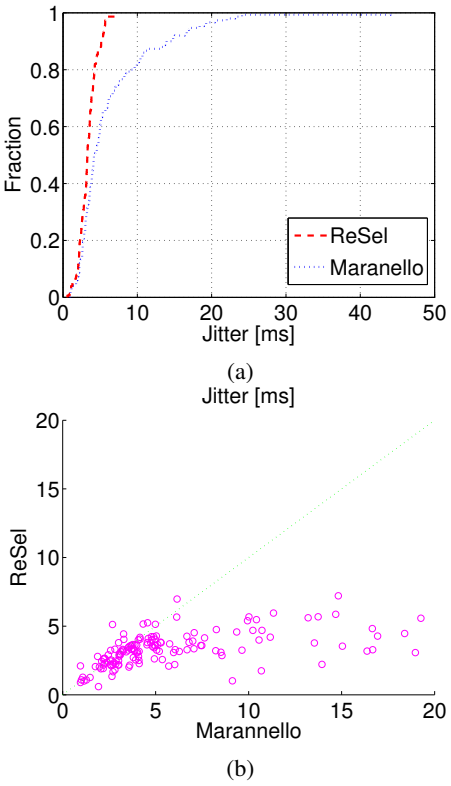


Fig. 4. (a). CDF of the jitter. (b). Scattered plot of the jitter.

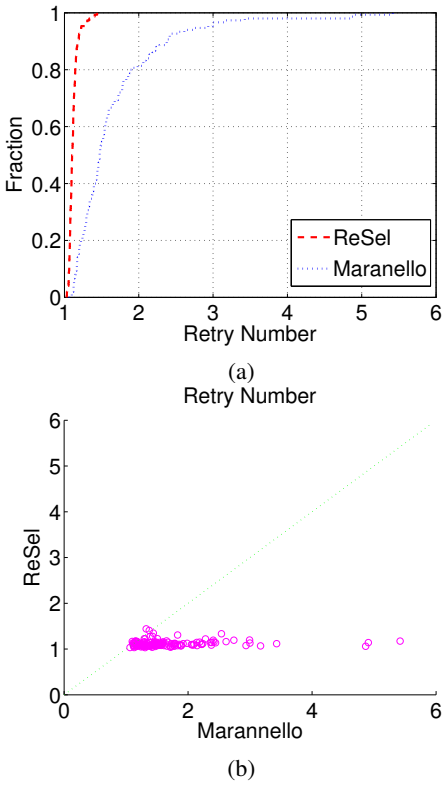


Fig. 5. (a). CDF of the retry number. (b). Scattered plot of the retry number.

similarly as the throughput ratio, we use the throughput ratio as an example:

- The throughput ratio increases as the partial ratio increase. This is not a surprise because the gain of ReSel is due to exploiting partial packets; the more partial packets, the higher the gain.
- The throughput ratio increases as the erasure ratio increases when the erasure ratio is below 0.2, which is likely because a higher erasure ratio leads to a higher partial ratio when the erasure ratio is below 0.2. When the erasure ratio is higher than 0.2, the throughput ratio ceases to increase, which is likely because more packets are lost due to erasure and cannot be repaired when the erasure ratio is higher than 0.2.

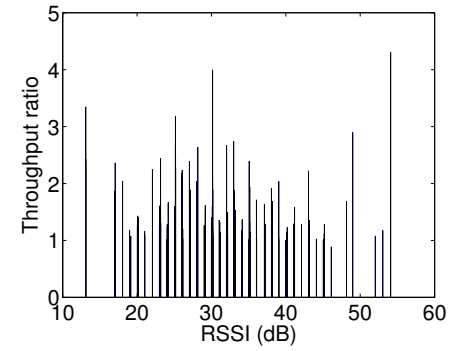
F. Remarks

Our experiments are carried out at a single data rate because we wish to compare the two schemes in a simple and well-defined scenario. We must mention that the performance of the original Maranello may be better if the rate is not fixed. This is because when the rate is not fixed, the driver will run an auto-rate algorithm and assign fallback rates for retransmissions upon too many failures. On the other hand, in our experiments, the rate is fixed at 36 Mbps and the diver does not provide a fallback rate, hence the original Maranello will always use 36 Mbps for retransmissions which may lead to more retransmissions.

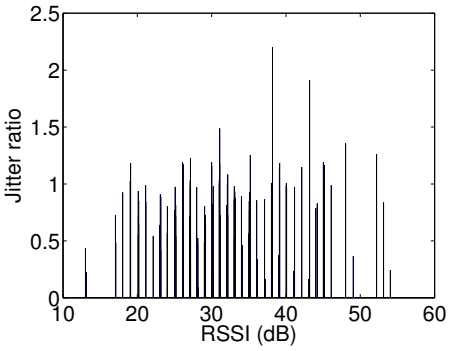
We note that according to [6], the fallback rate is not used until the third transmission of the repair packet in the current firmware implementation. On the other hand, we can see in Fig. 5(a) that the original Maranello uses no more than two transmissions for the repair packet for more than 80% of the links. Fig. 3(b) shows that ReSel achieves notable throughput gains in the majority of the links, including the links in which original Maranello uses no more than two retransmissions. We therefore suspect that ReSel can still achieve significant gains over the original Maranello even when auto-rate is enabled.

VI. Conclusion

In this paper, we propose ReSel for selecting the rates for the repair packets under the Maranello partial packet recovery framework. ReSel is motivated by two observations. First, the repair packet should use a different data rate from the original packet because the original packet has been transmitted and failed in the recent past which is likely within the channel coherence time. Second, the NACK packet sent by the Maranello receiver can be used to piggyback information such as RSSI and can be used to infer the channel condition. We build the rate selection table for ReSel with experimental data. We implement ReSel in the firmware and our experiments show that ReSel significantly improves the link throughput and reduces packet jitter. ReSel achieves the throughput gain and reduces the jitter because it is capable of selecting the appropriate data rate such that the repair packet is transmitted only once in the majority of the cases. Our future work includes the study of repair data rate selection when the auto-rate algorithm is enabled in the driver.

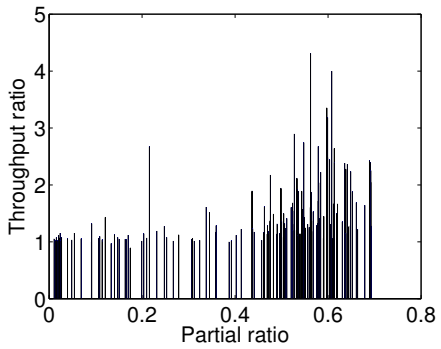


(a)

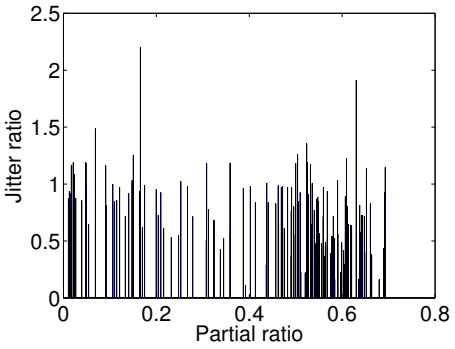


(b)

Fig. 6. RSSI and performance. (a). Throughput. (b). Jitter.

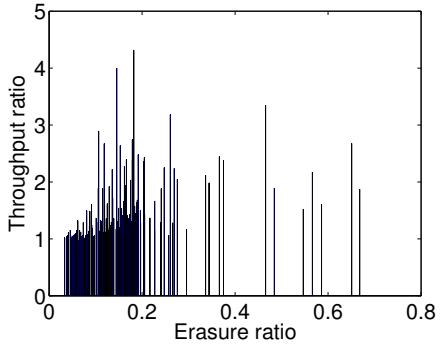


(a)

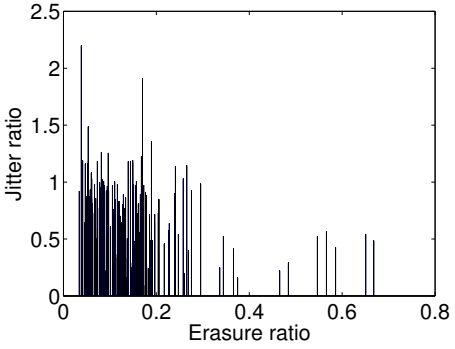


(b)

Fig. 7. Partial ratio and performance. (a). Throughput. (b). Jitter.



(a)



(b)

Fig. 8. Erasure ratio and performance. (a). Throughput. (b). Jitter.

REFERENCES

[1] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," in *ACM SIGCOMM*, 2010.

[2] G. Judd, X. Wang, and P. Steenkiste, "Efficient channel-aware rate adaptation in dynamic environments," in *ACM MobiSys*, 2008.

[3] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi, "AccuRate: Constellation based rate estimation in wireless networks," in *USENIX NSDI*, 2010.

[4] M. Vutukuru, H. Balakrishnan, and K. Jamieson, "Cross-layer wireless bit rate adaptation," in *ACM SIGCOMM*, 2009.

[5] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *ACM MobiCom*, 2006.

[6] B. Han, A, Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. Miller, "Maranello: Practical partial packet recovery for 802.11," in *NSDI, 2010*.

[7] K. Lin, N. Kushman, and D. Katabi, "ZipTx: Harnessing partial packets in 802.11 networks," in *ACM Mobicom*, 2008.

[8] J. Xie, W. Hu, and Z. Zhang, "Revisiting partial packet recovery in 802.11 wireless LANs," in *ACM Mobisys*, 2011.

[9] K. Jamieson and H. Balakrishnan, "PPR: Partial packet recovery for wireless networks," in *ACM SIGCOMM*, 2007.

[10] A. P. Iyer, G. Deshpande, E. Rozner, A. Bhartia, and L. Qiu, "Fast resilient jumbo frames in wireless LANs," in *IEEE IWQoS, 2009*, Charleston, SC, June 2009.

[11] iperf, *http://iperf.sourceforge.net/*.

[12] F. Gringoli and L. Nava, "Open FirmWare for WiFi networks: a UniBS NTW group project," *http://www.ing.unibs.it/˜openfwwf/*.