Smart Retransmission and Rate Adaptation in WiFi

Muhammad O Khan* Lili Qiu* Apurv Bhartia* Kate Ching-Ju Lin+
The University of Texas at Austin* Academia Sinica+

Abstract—**Transmission failures are common in wireless networks due to dynamic channel conditions and unpredictable interference. To efficiently recover from failures, we propose a smart retransmission scheme where the receiver combines information received from multiple failed transmissions associated with the same frame. The smart retransmission has two distinguishing features: (i) it can simultaneously support partial retransmission and combines bits with low confidence, and (ii) it has the first combining-aware rate adaptation scheme, which selects the data rates for all transmissions associated with the same frame to maximize overall throughput. We find that combining-aware rate adaptation is essential to harnessing the combining gain. Using trace-driven simulation and USRP testbed experiments, we demonstrate the feasibility and effectiveness of our approach, and show it significantly out-performs the existing schemes, such as WiFi, partial packet recovery (PPR) [14], and SOFT [32] in terms of both throughput and energy.**

I. Introduction

Motivation: Wireless channel condition is hard to predict due to mobility, dynamic interference, and environmental changes. Therefore, transmission failures are common in wireless networks. Traditional systems retransmit an entire frame upon every failure. This is inefficient since a significant portion of bits may have already been correctly received and retransmitting these bits is wasteful. Recognizing this inefficiency, PPR [14] proposes to extract physical layer hints to determine if bits are correct and discards/retransmits the bits with lower confidence. PPR is an interesting concept. Maximizing its full potential requires addressing several important challenges.

- It is hard to tell exactly which bits are corrupted. In order to ensure all incorrect bits are retransmitted, a conservative threshold has to be used. Many bits below the threshold may well be received correctly. Completely discarding them is wasteful. It would be beneficial if we can take a step further by leveraging information from bits with lower confidence to further reduce retransmission overhead. In other words, we should not only support partial retransmission, but also combine bits from multiple failed transmissions.
- We observe that a receiver decodes data by first demodulating the incoming signals to bits and then performing FEC decoding, as shown in Figure 1. Combining can be applied to the raw analog signal, the demodulated data before FEC, or the FEC decoded data. Where should combining take place to maximize the gain while achieving high flexibility?
- How do partial retransmission and combining affect rate adaptation? Existing schemes (*e.g.*, PPR [14] and SOFT [32]) show significant performance benefit under a fixed data rate. However, their gains diminish under standard rate adaptation, which tends to pick a conservative rate and leaves little opportunity for partial retransmission and combining. Therefore we should re-visit rate adaptation to take advantage of partial retransmission and combining.

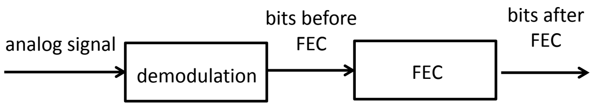


Fig. 1. Process of data decoding.

Our approach: In this paper, we seek to address these challenges in turn. We propose combining-aware partial packet recovery in which we retransmit the bits with lower confidence. For any bits that are received more than once, we combine them to improve accuracy.

To maximize the combining gain, we examine how combining accuracy changes when combining takes place at different stages. We find the combining gain is highest when it is performed before FEC decoding. However, combining analog signals has several limitations: (i) it is unclear how to combine signals transmitted using different modulations or FEC codings since it requires converting signals from one modulation/FEC coding to another before combining and there is no known solution to perform such conversion while maintaining the confidence of demodulated bits, and (ii) it is unclear how to combine signals from partial retransmissions because when different transmissions contain different signals, the same bit may be spread over different parts of the signals (*e.g.*, a bit may be the first bit in a QAM-16 symbol during the first transmission, but the second bit in a QAM-16 symbol during the second transmission). Bit position in a symbol is important since it affects the confidence level.

Therefore we propose combining the demodulated bits before FEC so that it is independent of modulation, thereby simultaneously achieving high combining gain and flexibility. In particular, we estimate Log-Likelihood Ratio (LLR) for each demodulated signal based on the signal-to-noise ratio (SNR) obtained from the preamble. To support both partial retransmission and combining, we use the LLR to determine which bit(s) before FEC are likely to be corrupted and combine bits from multiple transmissions by adding up their LLR and feeding the resulting LLR to a FEC decoder.

To maximize effectiveness, we observe that traditional rate adaptation tries to select the rate that maximizes throughput of the current transmission, where throughput is computed based on packet delivery rate. This essentially means that utility is 0 when the current transmission fails. This usually results in a conservative rate, which leads to successful delivery of the frame in one try and leaves little opportunity for partial packet recovery and combining. In order to fully take advantage of combining, the rate adaptation design should be revisited. In particular, the use of combining means that a failed transmission still conveys useful information by increasing LLR and has positive utility. To capture this notion, we formulate a new rate adaptation problem whose goal is to minimize the total time of the transmission and retransmissions to deliver a frame. This significantly changes the rate adaptation problem. We develop a novel rate adaptation scheme to select the rates for all the transmissions associated with a frame in order to maximize throughput, or equivalently, to minimize expected transmission time. Our main insight is that by leveraging combining, we can potentially use a more aggressive data rate due to a much lower retransmission cost. This creates an opportunity for partial retransmissions and combining.

Our major contributions can be summarized as follow:

- We study different combining options and show combining after demodulation but before FEC decoding achieves the best of both worlds – high combining gain and high flexibility (*i.e.*, supporting partial retransmissions and different data rates for retransmissions). We further design a practical approach to perform partial retransmission and combine bits based on the log-likelihood estimation (Section III).
- We develop a novel combining-aware rate adaptation scheme to effectively harness combining gains by taking into account the utility of a failed transmission and selecting the rate with the minimum total transmission and retransmission time of a frame. Our approach uses the standard data rates in IEEE 802.11 and is easy to deploy (Section IV).
- We demonstrate its effectiveness using trace-driven simulation and testbed experiments. Our results show WiFi, PPR, and SOFT perform similarly using the existing rate adaptation, which selects the rate that can deliver most frames successfully in one try. In comparison, our approach can take advantage of partial information delivered in failed transmissions and may choose higher rates. In simulation, our scheme out-performs WiFi on average by 26-31.5%, PPR by 13.5-19.5%, SOFT by 22-23% in terms of throughput for 4000-byte frames. It reduces energy by 18-25% over WiFi, 8.5-15% over PPR, and 11-18% over SOFT. For testbed, our scheme outperforms WiFi and SOFT by 11-52% in terms of throughput for 1000-byte frames over a range of experimental scenarios (Section VI–VII).

II. Related Work

Leveraging spatial diversity: MRD [22] leverages receptions at multiple APs to collectively recover the received frame. It considers the bits that are different in different receptions as corrupted and exhaustively searches over different combinations to find the one that passes CRC. SOFT [32] improves over MRD by using PHY-layer hints and combining different receptions using weighted averages instead of exhaustive search. [9] proposes a novel quantization scheme that allows different APs to efficiently share their received signals for combining. This is orthogonal to our focus.

Leveraging temporal diversity: [14] uses PHY-layer hint to identify data with low confidence and only retransmits that data. It develops a dynamic programming approach to balance the feedback overhead and retransmission overhead. Unlike PPR, which extracts the high-confidence bits from different transmissions, we observe that it is more efficient to combine different bits based on their log-likelihood estimates. SOFT [32] also has an extension to exploit temporal diversity in the downlink by combining the original transmission and retransmission(s). However, it retransmits the entire frame instead of partial retransmissions. Several extensions have been proposed since then. For example, [2] uses algebraic consistency rule instead of PHY-layer hint to detect erroneous bits, and [24] proposes shuffling constellation points in retransmissions to further enhance the effectiveness of combining.

This paper proposes the following three significant enhancements. First, it combines the best of both approaches – partial retransmission in PPR and PHY-layer-hint directed combining in SOFT. Second, different from SOFT and PPR, our approach combines the demodulated data before FEC to achieve high combining gain and flexibility as shown in Section III. Most importantly, both PPR and SOFT use the existing rate adaptation scheme, while we propose a new rate adaptation scheme to exploit the combining gain. Rate adaptation is essential to leverage the combining gain. While [14], [32] shows significant benefit over PPR and SOFT under fixed rates, our evaluation shows that their benefit under auto rate is rather limited. This is because an existing rate adaptation tends to choose a rate that lets most transmissions succeed in one try and leaves little room for PPR and SOFT to improve upon. In comparison, combining-aware rate adaptation increases combining opportunity and improves efficiency.

Rate adaptation: There has been significant work on rate adaptation. SampleRate [6] is a widely used rate adaptation scheme. It uses probes to select the rate that minimizes the expected transmission time. MadWiFi uses ONOE [23], which estimates long-term loss rate and uses thresholding for rate selection. RRAA [31] reports the pitfalls of several common practices and develops a robust rate adaptation based on short-term loss rate estimates. [17] proposes a new rate adaptation scheme based on error estimating coding. [11] shows effective SNR gives a good prediction of link performance and develops a rate adaptation based on it. The significant benefits and increasing popularity of MIMO devices have also motivated considerable work on rate adaptation for MIMO (*e.g.*, [25]). These works are all orthogonal to our rate adaptation scheme in that we develop the first combining-aware rate adaptation scheme that optimizes a new objective function: finding the data rates for a group of transmissions associated with the same frame to minimize the total transmission time when these transmissions are combined. Our approach can easily support MIMO.

Rateless codes: Several rateless codes have been proposed for wireless communication, such as LT [19], Raptor [27], Strider [10], and Spinal [26] codes. Some rateless codes have been shown to approach the Shannon capacity in real networks. However, the existing rateless codes have the following major limitations. (i) Rateless code has high complexity. For example, Strider's complexity is KN where N is the length of the packet and K is the block size and typically set to 33. Spinal code involves $O((n/k)B * L * 2^{(k*d)})$ hashes and $O((n/k)B * 2^k)$ comparisons, where n is the total data, k is the block size, B is the beamwidth (*i.e.*, number of nodes kept at each step), d is the depth of the tree, n/k is the size of each packet, and L is the number of packets transmitted before successful decoding. (ii) Rateless codes can incur large delay since they may require lots of transmissions to successfully deliver a frame and some of the rateless codes require block coding structure, which further increases delay, and (iii) Rateless codes are hard to deploy since they are significantly different from the modulation/demodulation schemes in the existing wireless systems. In contrast, our smart retransmission overcomes these limitations: it has low complexity and only needs a small number of retransmissions (if any). It is also easier to deploy since it uses the standard modulation and FEC codings that already exist in IEEE 802.11. So our approach can be viewed as a simple and practical approximation to the rateless codes with low delay.

Theoretical Analysis: There have been significant theoretical analyses of combining gain. The idea of combining was first proposed in [7]. There is significant work on incremental redundancy (*i.e.*, partial retransmission) [18], [20], [28], which are grouped under Hybrid ARQ. While these works provide useful insights, they do not address the interaction between combining, partial retransmissions, and rate adaptation. Our work is unique since it provides a single framework that unifies combining and partial retransmission with rate adaptation.

III. Signal Combining

Signal combining is a well-known technique in wireless communication and is used to provide spatial and temporal diversity. Signal combining leverages multiple receptions of the same data to get a better estimate of the received signal. In this section, we first describe and compare different combining approaches. Then we present our approach to combine partial retransmissions while maximizing the combining gain.

A. Different Combining Schemes

Overview of three combining schemes: As mentioned in Section I, signal combining can be performed at various stages of decoding: (i) before demodulation, (ii) after demodulation but before FEC decoding, and (iii) after FEC decoding.

In (i), the same signal should be transmitted. The receiver computes a weighted sum of the received signals, and then demodulates the resulting signal as usual. It is commonly referred to as symbol combining. Figure 2 shows an example of (i), where two BPSK signals $R1$ and $R2$ are combined and demodulated correctly to 1.

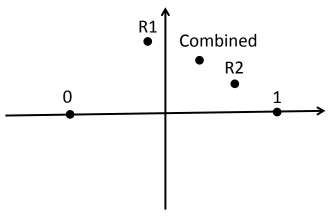


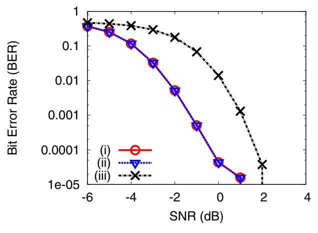
Fig. 2. Symbol combining.

In (ii), a receiver first demodulates an analog signal into digital bits. Based on the distance between the received signal and constellation point, the receiver computes log-likelihood (LLR) value. LLR is the logarithm of the ratio of probabilities of a 0 bit being transmitted versus a 1 bit being transmitted given a symbol r is received. The LLR for a bit b is computed as follows:

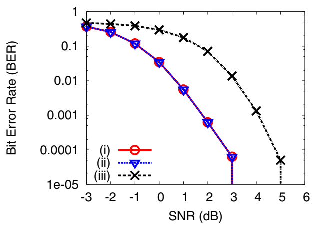
$$L(b) = \log\left(\frac{Pr(b=0|r=(x,y))}{Pr(b=1|r=(x,y))}\right) \quad (1)$$

where r is the received signal, (x, y) is the x and y coordinates of the received signal in the constellation map, b is the transmitted bit. If we assume the probabilities of all symbols are equal, the LLR of an AWGN channel is $L(b) = \log\left(\frac{\sum_{s\in S_0(b)} e^{-\frac{1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}{\sum_{s\in S_1(b)} e^{-\frac{1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}\right)$, where $S_0(b)$ and $S_1(b)$ are ideal symbols with bits 0 and 1 at the given bit position b, respectively, and σ^2 is noise variance of baseband signal and can be obtained using the preamble. To combine multiple received data, the receiver sums up LLR values from all the receptions and feeds the resulting LLR to an FEC decoder (*e.g.*, Viterbi decoder) to enhance the decoding rate. There are two types of decoders: hard decoder, which takes a hard decision as an input (*e.g.*, 0 or 1), and a soft decoder, which leverages more fine-grained LLR values to improve decoding accuracy. Our scheme supports both hard decoder and soft decoder. We focus on the hard decoder in our evaluation since its delivery rate can be estimated accurately (which is required for rate adaptation) whereas our experiments show that the delivery rate of a soft decoder (with or without combining) depends on not only effective SNR but also the mapping between symbols and SNR and is much harder to estimate. Therefore we defer our evaluation of the soft decoder to our future work.

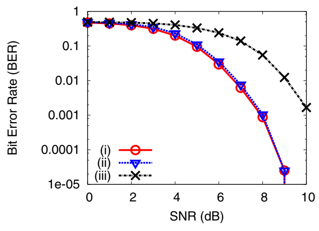
In (iii), a receiver performs FEC decoding on each received data separately and outputs LLR. Our evaluation uses Bahl Cocke Jelinek Raviv (BCJR) Maximum a Posteriori (MAP) decoding algorithm [3], [4]. Its performance is the same as Viterbi decoder when both use soft decoding, but slightly worse than Viterbi when both use hard decoding. Then the receiver adds up the LLR for each bit, and uses a thresholding to determine the final value for the bit (*e.g.*, the bit is 1 if the LLR sum is greater than 0, and 0 otherwise).



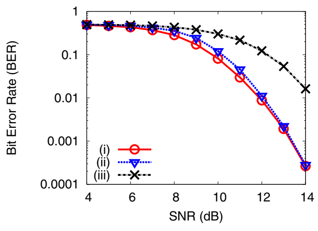
(a) BPSK



(b) QPSK

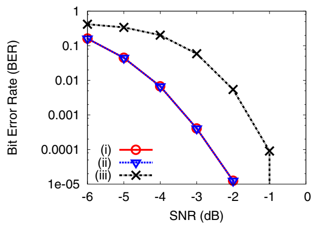


(c) QAM16

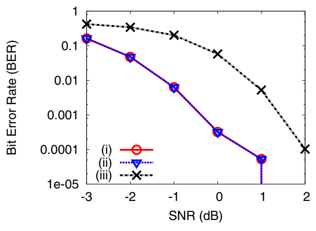


(d) QAM64

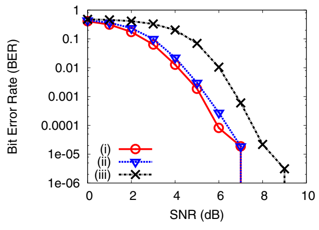
Fig. 3. Hard combining gain comparison, where (i) combining before demodulation, (ii) combining after demodulation but before FEC decoding, and (iii) combing after FEC decoding.



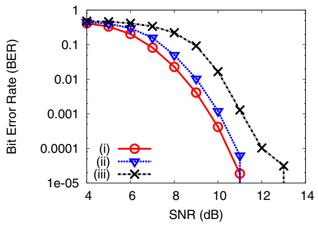
(a) BPSK



(b) QPSK



(c) QAM16



(d) QAM64

Fig. 4. Soft combining gain comparison, where (i) combining before demodulation, (ii) combining after demodulation but before FEC decoding, and (iii) combing after FEC decoding.

Comparison of the three combining schemes: We compare the combining gains using different schemes as follow. We first generate signals according to a selected modulation. We then add AWGN noise to the signal whose magnitude is determined by SNR. We send each signal twice and then calculate the bit error rate (BER) of each scheme over 320,000 bits. Figure 3 compares different combining schemes using a hard decoder, which uses 1/2 convolutional FEC code with BPSK, QPSK, QAM-16, and QAM-64. Figure 4 summarizes the performance of different combining schemes using a soft decoder.

In all cases, we observe (i) $\approx$ (ii) $>$ (iii). (i) and (ii) perform similarly, and only occasionally (i) is slightly better than (ii). However, (i) is limited by the constraint that the retransmitted symbols must be the same as the original symbols. This implies that same set of symbols must be retransmitted and the retransmissions should also use the same data rate (*i.e.*, the same modulation and FEC code) as the original transmissions. This severely limits the applicability of symbol combining and makes it hard to support partial retransmission and rate adaptation for retransmissions.

On the other end, (iii) is completely independent of the data rate (*i.e.*, modulation and FEC). Data transmitted using different rates can be easily combined at the digital level. Moreover, it can easily support partial retransmissions. However, as shown in Figure 3 and 4, its combining gain is lower than (i) and (ii) by up to 2 dB since it cannot leverage the FEC coding gain.

(ii) achieves the best of both worlds. Its combining gain is close to the maximum combining gain: (ii) yields the same SNR gains for BPSK and QPSK as (i), and is at most 1 dB lower than (i) under a higher order modulation. Furthermore, (ii) is flexible and can support partial retransmissions and different modulation as we will show in Section III-B. Therefore we use (ii). Note that (ii) gives the same gain as (i) under BPSK and QPSK, but is slightly worse under QAM because demodulation is independent across different bits in BPSK and QPSK, which makes combining before and after demodulation equivalent in BPSK and QPSK, whereas demodulation of different bits is correlated in QAM and combining before demodulation allows a bit to benefit from the combining gain from other bits in the same symbol, thereby improving demodulation success rate.

B. Combining Partial Retransmissions

Partial data retransmission is another technique used to reduce retransmission overhead. In WiFi, it is common to have corrupted frames that have a small number of erroneous bits. Therefore, retransmitting an entire frame is wasteful. PPR [14] uses PHY-layer hints to identify the bits that are likely to be in error and only retransmits those bits. The concept of PPR is interesting. However, there are three major limitations of PPR.

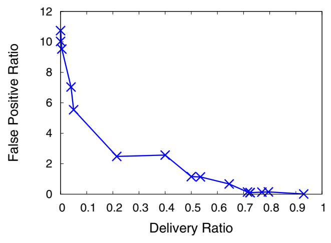


Fig. 5. PPR false positive ratio.

First, it is hard to determine an appropriate LLR threshold (after FEC decoding) for deciding which bits are correct and which bits are in error. This is because LLR threshold must be high enough that all bits in error are chosen to be retransmitted. This is necessary because PPR selects bits for retransmission after FEC decoding. However, if a large threshold is selected, a significant number of bits that were received correctly also end up being retransmitted. Figure 5 plots the false positive ratio (*i.e.*, the ratio between the number of bits that are correct but deemed in error versus the total number of actual erroneous bits) versus the frame delivery rate. Each point in the curve is generated by simulating 200 frame transmissions over a given SNR. For each frame, we determine the LLR threshold (after FEC decoding) for which all erroneous bits are selected for retransmission. For the selected threshold, we determine the number of false positives and calculate the false positive ratio for that frame. Finally, we take the average of false positive ratios over all 200 frames and plot it against the delivery ratio of the 200 frames. We simulate over a range of SNR values to generate the curve. As we can see, the false positive rate can be very high. When the delivery rate is 0.4 or lower, the false positive ratio is over 2, indicating that we may retransmit over twice the number of erroneous bits. When the delivery rate reduces further, the false positive ratio becomes 6 or higher.

Second, PPR discards the bits below the threshold. Given a significant number of bits below the threshold may well be correct, it is important to use these bits to improve decoding.

Third, PPR is applied after FEC decoding. As shown in Figure 3, the benefit of combining after FEC is lower than the other alternatives.

Most importantly, PPR does not modify the rate adaptation and uses the rate adaptation that is not aware of partial packet recovery. As a result, in most cases the selected rate allows a packet to be delivered successfully in one shot and gives little opportunity for partial packet recovery. Our evaluation confirms this intuition and shows PPR performs similar to WiFi under the existing rate adaptation.

Motivated by the benefit and limitations of PPR, we propose an approach to combine partial retransmissions. As PPR, we also identify likely erroneous bits and perform partial retransmissions. Our approach differs from PPR in the following ways. First, instead of using a fixed threshold to determine if a bit is correct, we search for bits for retransmissions that maximize throughput. Our approach does not require that all erroneous bits be retransmitted since combining is done before FEC in our scheme and the FEC can tolerate a few erroneous bits. We no longer need to select a threshold, but let the rate adaptation scheme automatically balances the cost of retransmission and delivery rate, as described in Section IV. Second, we do not discard bits likely to be received in error. Instead, we combine all the received bits to increase the data decoding probability. Third, we propose a combining aware rate adaptation scheme, which explicitly takes into account partial retransmission and combining to select a more appropriate rate.

To reduce feedback overhead, we retransmit in a unit of OFDM subcarriers. Due to temporal stability within a subcarrier and frequency diversity across subcarriers, symbols transmitted on the same subcarrier are likely to experience similar SNR and symbols transmitted on different subcarriers tend to experience different SNR. We compute Bit Error Rate (BER) of each subcarrier, and sort the subcarriers in an decreasing order of BER. Our rate adaptation, described in Section IV, finds the worst k subcarriers for retransmission (*i.e.*, all bits on these subcarriers are retransmitted). The feedback contains a bitmap of subcarriers, where 1 indicates the corresponding subcarrier requires retransmission. The transmitter concatenates all the bits required for retransmission in the order of OFDM symbol index and then an increasing order of subcarrier index, and maps them sequentially to all the subcarriers during retransmissions. Since the receiver selects which bits to retransmit, it knows the mapping used by the transmitter. The combining is performed by summing up the LLR values for any bit that has more than one reception and feeding the resulting LLR values to the Viterbi decoder. By leveraging information received previously despite having low confidence, we can achieve high decoding rate and reduce retransmission overhead.

C. *Benefits of Combining*

In general, combining is beneficial for three main reasons. First, combining reduces retransmission cost and may potentially allow us to choose a higher data rate. Since wireless losses are probabilistic, sometimes frames can still be delivered successfully at a higher rate. Even when they are not successfully delivered, the retransmission overhead is smaller and does not degrade throughput significantly. Second, combining is useful under frequent fluctuations in the wireless channel (*e.g.*, arising from mobility). Under unpredictable wireless channel, losses are common. By leveraging information from previously failed transmission, combining reduces the cost of failures. Third, combining is also useful under dynamic interference due to reduced retransmission cost in case of collision.

The same combining scheme works for all the scenarios. In all cases, we use both preambles and data symbols to compute SNR for each subcarrier. For the preamble, since we already know the reference symbol, we simply use the distance between the expected and received constellation point to compute SNR. For the data symbol, since we do not know the ground truth, we use the distance between the received and the few most closest constellation points to derive SNR. Then we compute the variance $\sigma^2 = 1/SNR$, and derive LLR as $L(b) = \log\left(\frac{\sum_{s \in S_0(b)} e^{-\frac{1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}{\sum_{s \in S_1(b)} e^{-\frac{1}{\sigma^2}((x-s_x)^2+(y-s_y)^2)}}\right)$. The LLR formula is general and applicable with or without interference, since the variance in the formula captures background noise when there is no interference, and captures both background noise and interference otherwise.

IV. Combining-aware Rate Adaptation

Rate search: Traditional rate adaptation selects a rate to minimize the time for the current transmission. That is, it treats each transmission and retransmission as isolated units and picks a rate to optimize these individual units independently. However, we observe that multiple transmissions associated with the same frame should be considered as one unit and our goal is not to optimize individual transmissions, but to optimize the entire process of successfully delivering one frame.

The transmission time depends on the data rate used and the amount of data to be transmitted. For example, one could either retransmit more data bits at a higher rate (less reliably) or retransmit fewer bits but at a lower rate (more reliably). Therefore our rate adaptation should search not only for the data rate used for each transmission but also how much data to retransmit every time.

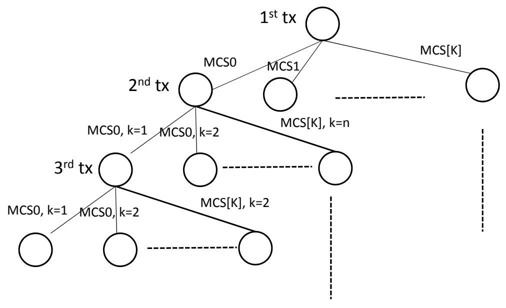


Fig. 6. Rate search tree.

To make the search scalable and reduce feedback overhead, we retransmit in a unit of subcarriers. It means that if a subcarrier is selected for retransmission, then all the bits that are transmitted on that subcarrier during the original transmission are retransmitted. To select the subcarriers for retransmission, we calculate the average BER for each subcarrier, and then sort the subcarriers in a decreasing order of BER. We incrementally add one subcarrier at a time to retransmit, starting with the worst subcarrier (highest BER). Each time we search over all the data rates and compute the expected transmission time and delivery rate. If the delivery rate is below a threshold, we repeat the search process for the next retransmission. This process continues until we populate the rate search tree. Figure 6 shows an example of rate search tree, where each branch of the tree corresponds to a transmission strategy. As it shows, the left-most branch corresponds to transmitting at the rate of MCS[0] for the first time, transmitting the worst subcarrier in the first transmission at the rate of MCS[0] for the second time, and transmitting the worst subcarrier (after combining) at the rate of MCS[0] for the third time. The branch terminates as soon as all the bits in a frame are successfully delivered or the maximum number of retransmissions is reached. Our evaluation considers frame delivery rate over 99% or 2 retries as the stopping criterion. Once we build the rate search tree, the best transmission strategy for a frame is simply the branch that yields the minimum transmission time. It should be noted that the BER based ordering of the subcarriers may change after retransmission, as the receiver combines all the transmissions associated with the same bit and re-sorts the subcarriers according to the average BER (after combining).

The transmission time of each branch is calculated as $TotalTime = \sum_i Time_i = \sum_i (size_i/rate_i + overhead_i)$, where $size_i$ and $rate_i$ are size and data rate of the i-th transmission and $overhead_i$ is the overhead of the i-th transmission. $size_i$ depends on the delivery rate after combining all transmissions that have occurred so far. $overhead_i$ includes the header overhead and MAC overhead such as DIFS, SIFS, and feedback overhead. Our implementation has two types of feedback: partial ACK and complete ACK. A partial ACK contains one OFDM symbol to specify the bitmap of which subcarriers to retransmit and the data rate to use for retransmission. A complete ACK is the same as a WiFi ACK.

modulation	BER
BPSK	$Q(\sqrt{2snr})$
QPSK	$Q(\sqrt{snr})$
QAM-16 (bits 0 and 2)	$\frac{1}{2}Q(\sqrt{snr/5})$
QAM-16 (bits 1 and 3)	$Q(\sqrt{snr/5})$
QAM-64 (bits 0 and 3)	$\frac{1}{4}Q(\sqrt{snr/21})$
QAM-64 (bits 1 and 4)	$\frac{1}{2}Q(\sqrt{snr/21})$
QAM-64 (bits 2 and 5)	$Q(\sqrt{snr/21})$

TABLE I
BER AS A FUNCTION OF SNR FOR DIFFERENT MODULATION

Computing delivery ratio: An important step in rate search is to compute the delivery rate. We compute it as follows.

- Derive uncoded BER (*i.e.*, BER before FEC decoding): Our goal is to compute the probability of each bit being in error after combining multiple transmissions. To achieve this, we (i) derive SNR for each bit, (ii) sum up the SNR of all the transmissions involving the same bit, and (iii) map the combined SNR to BER using Table I.
  Below we elaborate step (i). For the received bits, we calculate the LLR value for each bit using the expression (1). We then calculate the probability of having an error in a received bit using $P_{\text{b}}(k) = 1/(1+e^{|s_k|})$ where $|s_k|$ is the absolute LLR value of bit k and $P_{\text{b}}(k)$ is the probability of bit k being in error [30]. We then use the inverse of the formulas in Table I to map BER to SNR of individual bits.
  Note that we calculate the BER separately for each bit location in a given symbol. For example, each QAM-16 symbol has four bits $b0$, $b1$, $b2$ and $b3$. As shown in Table I, the BER experienced by $b0$ and $b2$ is different from that of $b1$ and $b3$. This is because the way constellation points are located as shown in Figure 7. Therefore, it is necessary to have a separate BER estimate for each bit location to achieve high accuracy. We empirically derive BER for QAM as shown in Table I.

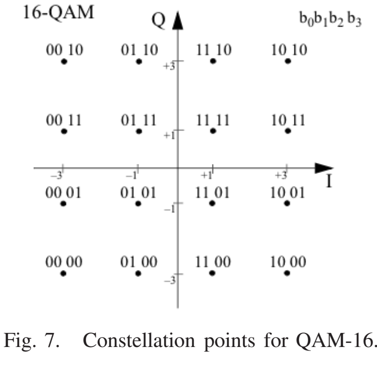


Fig. 7. Constellation points for QAM-16.

To compute the uncoded BER of future transmissions, which is required for rate search, we use the channel state information (CSI) of the previous frame to make prediction. The CSI consists of $M \times N$ matrices H_s, each of which specifies amplitude and phase between pairs of N transmitting and M receiving antennas on subcarrier s. The CSI is estimated using the preamble of a frame, and allows us to compute SNR for each subcarrier. For simplicity, we use the previous CSI for prediction. Alternatively, one may also use exponential weighted moving average (EWMA) (*i.e.*, $y = \alpha x + (1 - \alpha)y$, where x is the current sample, y is the prediction) or Holt-Winter algorithm [15] for prediction.

- Compute Effective SNR: We calculate the effective SNR by averaging BER calculated in the first step and then mapping it back to the SNR for each modulation using the procedure specified in [11].
- Lookup Delivery Ratio: We use a standard pre-computed lookup table (*e.g.*, as in [11]) to get the delivery ratio for each data rate based on the effective SNRs calculated in the previous step. The table contains delivery ratios for SNR values over a range of -10 to 30 dBs and packet sizes from 25 to 4000 bytes. The table is generated offline in Matlab by sending and decoding 1000 packets for each SNR value over a flat fading channel. This step is the same as [11].

The above steps work for both SISO and MIMO.

Speed up: To enhance the efficiency of our search, we prune the search tree in the following way. We perform breath-first search over the rate search tree and keep the top M branches at each level in terms of throughput so far (computed as the expected number of bits delivered so far divided by the transmission time including the overhead), and prune all the other branches. We only explore the top branches deeper. We control the value of M to balance the tradeoff between the computation time and optimality. Our evaluation uses $M = 4$. Furthermore, when we search for the number of subcarriers to retransmit, we stop whenever adding a subcarrier to retransmit reduces throughput. Finally, we limit the depth of a tree by looking ahead 2 transmissions.

Energy minimization: Our rate search is general and can be used to optimize different objectives. In particular, we can use the same search algorithm to minimize energy consumption. Specifically, we compute the energy consumption using an energy model, derived from measurement. For example, as shown in [16], energy consumption of a transmission and reception is a linear function of expected transmission time (ETT), where the slopes can be derived using real energy measurements. Therefore, we can compute ETT for each transmission to derive energy and select the branch that leads to the lowest energy. Our evaluation in Section VI shows our approach saves considerable energy.

Supporting MIMO: Our rate search can support MIMO. There are two differences in MIMO. First, there are more rates under MIMO, and the rate search tree becomes larger. So careful pruning is important to keep computation time low. For example, we can search around the neighborhoods of the previous rates. Second, we use MIMO post-processed SNR (pp-SNR) to compute the new delivery rate after performing a new retransmission under MIMO. In spatial diversity, transmissions between different transmitter and receiver antennas are spatially combined and we can apply the formulas in [16] to compute pp-SNR. In spatial multiplexing, a sender stripes a large frame across multiple antennas and transmits these multiple streams simultaneously. We derive BER for each subcarrier as usual, and compute effective SNR by averaging BER over all subcarriers and all spatial streams. As the data rate increases in MIMO, frame aggregation can be used (as usual) to minimize MAC/PHY overhead in the retransmission.

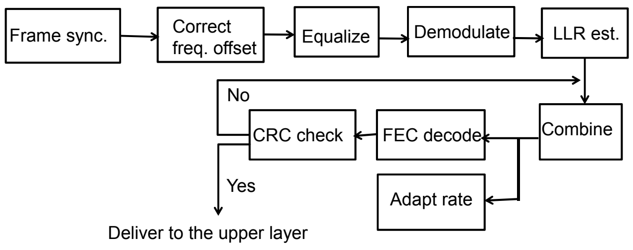


Fig. 8. USRP implementation

V. Protocol Implementation

We implement our approach in USRP [29]. To reduce feedback overhead, we let the receiver measure log-likelihood, perform rate search, and feed back the selected rate to the sender. The receiver feedback not only includes the data rate for retransmission but also which subcarriers to retransmit using a simple bitmap, where 1 at the i-th position indicates that the sender should retransmit data that was sent on the i-th subcarrier in the original transmission. To enhance reliability of feedback, we transmit the feedback at the lowest data rate in our implementation.

As shown in Figure 8, the receiver processes the incoming signals as follow. It first uses cross correlation to detect the beginning of an incoming frame [8], corrects for frequency offset [8], estimates the channel coefficient by dividing the received preamble by the known preamble. Then it uses the same preamble to compute the noise variance, demodulates the analog signals to digital bits, and derives LLR for each bit. It further combines the LLR values for the bits that have been received earlier and passes to FEC decoder. If it fails, the bits and their LLR values are stored for future combining. Otherwise, they are delivered to the upper layer. Meanwhile, based on the combined results and latest channel estimate, the receiver selects the rate as described in Section IV.

Our implementation uses a bandwidth of 1 MHz, 80 OFDM subcarriers, FFT window of size 128, cyclic prefix of 32 samples, and FPGA running at 100MHz. We use the convolutional coding implementation from the IT++ public library [13].

2.49GHz frequency is used to avoid external interference from the campus network. We modify the default GNU Radio OFDM implementation to realize symbol combining, receiver feedback, retransmission, and rate adaptation.

VI. Simulation Evaluation

A. Evaluation Methodology

In this section, we first use trace driven simulation to compare different schemes in Matlab. In Section VII, we further compare using USRP implementation. We collect three channel traces from static environments, and another three traces from mobile environments with human walking speed. The three mobile traces are collected in an office environment using 1 moving receiver and 3 static senders. The three static senders are 7m away from each other. Each trace corresponds to one of the three senders transmitting while the receiver is moved at a walking speed.

We use Intel Wi-Fi Link 5300 (iwl5300) IEEE 802.11 a/b/g/n wireless network adapters to collect the CSI of each frame preamble across all subcarriers. These NICs have three antennas. We use 802.11n, and enable all three antennas at both the sender and receiver. The modified driver [12] reports the channel matrices for 30 subcarrier groups, which is about one group for every two subcarriers in a 20 MHz channel according to the standard [1] (*i.e.*, 4 groups have one subcarrier each, and the other 26 groups have two subcarriers each). We use a transmission power of 15 dBm. Three MIMO streams are sent, so the NICS report CSI in the form of 3×3 matrices for each frame. In our evaluation, we account for the SIFS/DIFS and feedback overhead, which includes MAC-layer ACKs and feedback of which subcarriers to retransmit.

We compare the following approaches. All schemes except Smart and SmartNoCombine use Effective SNR [11] based rate adaptation, which is state-of-the-art rate adaptation scheme and is shown to capture frequency diversity quite accurately. Effective SNR maps the SNR of each subcarrier to BER, averages the BER across subcarriers, converts it back to SNR, and uses effective SNR to look up the frame delivery rate table. Finally, it selects the rate to maximize throughput.

- WiFi: Whenever a frame does not pass CRC check, the entire frame is retransmitted. The receiver discards a failed transmission, and decodes a retransmission independently.
- SOFT: When a frame is corrupted, the entire frame is retransmitted. It combines symbols from all transmissions using maximal ratio combining (MRC) and then performs demodulation and FEC decoding on the combined signals. If the combined result passes CRC, it is delivered to the upper layer. Otherwise, an entire frame is retransmitted until it succeeds.
- PPR: Low-confidence bits are retransmitted. The receiver uses a confidence metric like LLR to extract the bits likely to be correct from each reception and merges them together. Our implementation differs from the original PPR [14] in that (i) we use LLR values from demodulation to identify the low-confidence bits and retransmit coded bits (*i.e.*, before FEC decoding), which is better than after FEC decoding in PPR as shown in Section III, and (ii) we retransmit in units of subcarriers to reduce feedback overhead.
- Smart retransmission and rate adaptation (Smart): This is our main approach as described in Section III and Section IV, which supports combining partial retransmissions before FEC decoding and combining-aware rate adaptation.
- SmartNoComb: This approach is the same as Smart, but only enables partial retransmission and disables combining. The rate adaptation also uses a similar tree-based rate search in Section IV except that we do not consider the combining gain (*i.e.*, the retransmission bits are taken as they are without combining with the previous receptions). The difference between PPR and SmartNoComb reflects the benefit of partial retransmission aware rate adaptation, and the difference between SmartNoComb and Smart reflects the combining gain.
- SmartUnaware: This is the same as Smart Retransmission except that the rate adaptation uses effective SNR per transmission and does not optimize rate selection across multiple transmissions. This is a useful reference for us to quantify the benefit of combining-aware rate adaptation.

B. Performance Results

Static networks: We first evaluate the performance using the static traces. Since the channel is usually stable within a transmission time of a frame, the SNR of preambles is used for trace-driven evaluation. This makes it easy to use the same traces to evaluate different frame sizes. We scale the SNR across subcarriers uniformly up and down to get a complete SNR range. In each run, we transmit 400 frames. We multiply the symbols in a frame by the channel coefficients as specified in the CSI trace, and add appropriate amount of Additive White Gaussian Noise (AWGN). The CSI traces are recorded assuming that the noise variance is 1. The frame is then decoded. Based on the scheme being evaluated, the appropriate rate adaptation scheme is called to calculate the appropriate data rate for the next transmission. We use a max retransmission count of 7, after which the frame is dropped.

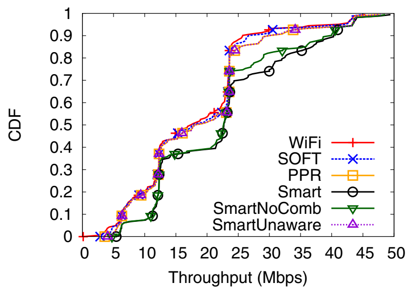


Fig. 9. Intel static traces with 4000-byte frames: the average throughput of WiFi, SOFT, PPR, Smart, SmartNoComb and 17.7 Mbps, 18.1 Mbps, 18.67 Mbps, 22.31 Mbps, 21.59 Mbps and 18.67 Mbps respectively.

Figure 9 compares our scheme with the existing schemes using 4000-byte frames. This is a common setting for IEEE 802.11n, since frame aggregation is turned on by default in commodity software drivers [5]. On average, our scheme out-performs WiFi by 26%, SOFT by 23%, PPR by 19.5%, SmartNoComb by 3.5%, and SmartUnaware by 19.5%. These numbers indicate: (i) Smart yields the best performance by using both combining and combining-aware rate adaptation, (ii) SmartNoComb is the next best performer due to its partial retransmission-aware rate adaptation, and (iii) WiFi, SOFT, PPR, and SmartUnaware all perform similarly since the rate adaptation tends to pick a rate that allows the transmission to be delivered successfully in one try and leaves little opportunity for PPR, SOFT, and SmartUnaware to improve further. Therefore combining-aware rate adaptation is essential to the performance. Furthermore, if we consider the traces where 75% of the frames result in retransmission, Smart shows 15% performance improvement over SmartNoComb. This shows the combining provides significant benefit during retransmissions. Note that the maximum frame size allowed in 802.11n is even larger: up to 64 KB [21], and the gain of our scheme will increase further with the frame size.

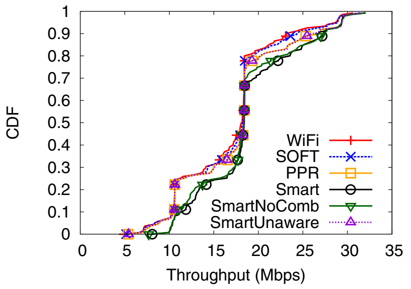


Fig. 10. Intel static traces with 1000-byte frames: the average throughput of WiFi, SOFT, PPR, SmartNoComb ,SmartUnaware and Smart are 16.72 Mbps, 16.93 Mbps, 17.36 Mbps, 18.38 Mbps, 17.35, and 18.65 Mbps, respectively.

Figure 10 compares different schemes using 1000-byte frames. Our schemes continue to perform the best. On average, it out-performs WiFi by 12%, SOFT by 10%, PPR and Unaware combining rate adaptation by 8%, and SmartNoComb by 1.5%. The reduced benefit is due to relatively larger MAC overhead for a smaller frame size. Note that the selected rate curves contain a few large jumps. This is because our static traces contain a few subcarriers with very low SNR. Due to such strong frequency selectivity, 1/2 FEC with higher modulation out-performs 3/4 FEC using lower modulation and the 3/4 FEC date rates never get selected.

Mobile networks: Next we compare the performance using the mobile traces. Figure 11 shows the throughput under 4000-byte frames. As it shows, on average, our scheme out-performs WiFi by 31.5%, SOFT by 22%, PPR by 13.5%, SmartNoComb by 4.5%, and SmartUnaware by 13%. The performance is consistent with the static traces: Smart continues to perform the best, followed by SmartNoComb. The other four schemes perform similarly due to ineffective rate adaptation. Again, when we consider the traces where 75% of the frames are retransmitted, Smart out-performs SmartNoComb by 12%,

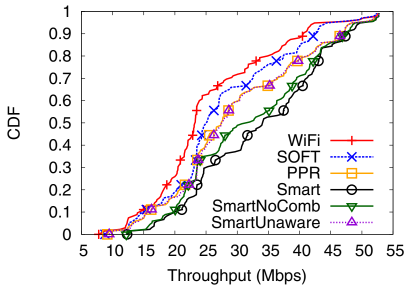


Fig. 11. Intel mobile traces with 4000-byte frames: the average throughput of WiFi, SOFT, PPR, Smart, SmartNoComb and SmartUnaware are 25.55 Mbps, 27.59 Mbps, 29.62 Mbps, 33.62 Mbps, 32.16 Mbps and 29.76 Mbps respectively.

which shows the benefit of combining especially during retransmission. Compared with the static traces, the throughput curves of our mobile traces are more smooth since QPSK with 3/4 FEC works sometimes and the rate smoothly changes from QPSK 1/2 FEC to QPSK 3/4 FEC to QAM-16 1/2 FEC.

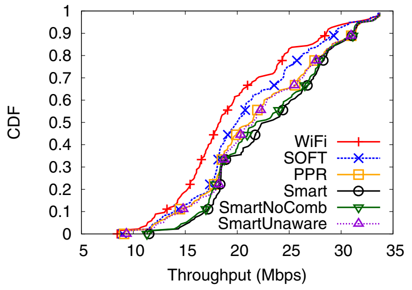
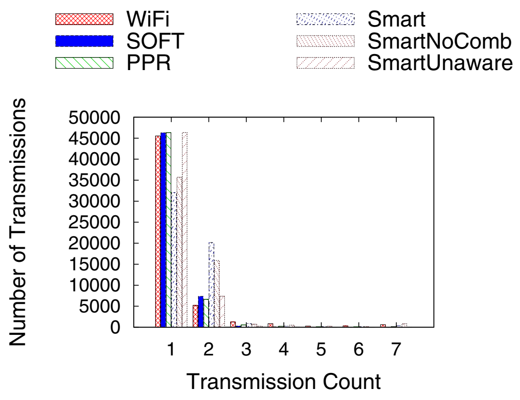


Fig. 12. Intel mobile traces with 1000-byte frames: the average throughput of WiFi, SOFT, PPR, Smart, SmartNoComb and SmartUnaware are 19.58 Mbps, 20.97 Mbps, 21.94 Mbps, 23.25 Mbps, 22.86 Mbps and 22.13 Mbps respectively.

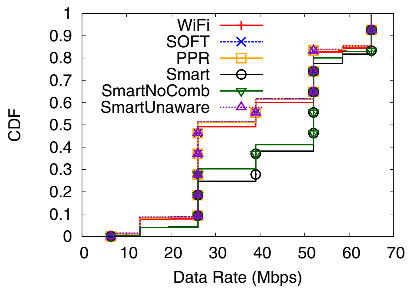
Figure 12 further compares the performance using 1000-byte frames in the mobile traces. On average, Smart out-performs WiFi by 19%, SOFT by 11%, PPR by 6%, Smart-NoComb by 2%, and SmartUnaware by 5%. The relative rankings of different schemes remain the same: Smart and SmartNoComb continue to perform well owing to their more effective rate adaptation.

To further understand the dynamics of different schemes, we examine the number of transmissions and data rates used by each scheme for 4000-byte frames in the mobile traces. Figure 13(a) compares the number of transmissions under different schemes. As we would expect, all schemes except SmartNoComb and Smart Retransmission complete most of their transmissions in one try. Specifically, WiFi, SOFT, PPR and SmartUnaware complete 84-86% of the transmissions in 1 try. In comparison, since the rate adaptation in Smart and SmartNoComb are both aware of partial retransmissions and can select a more aggressive rate, they complete 60% and 66% of their transmissions in first try, respectively, and 37% and 29% in two tries. The 6% gap between Smart and SmartNo-Comb shows that combining allows us to select an even more aggressive rate, thereby achieving higher throughput.

Figure 13(b) further compares the data rates used in different schemes. As we would expect, Smart and SmartNoComb



(a) Number of transmissions



(b) CDF of data rates

Fig. 13. Comparison of numbers of transmissions and data rates in mobile traces using 4000-byte frames.

tend to select higher rates, whereas the rates selected by the other four schemes are considerably lower. For example, the median rate selected by Smart is over 50 Mbps, compared to 30 Mbps in WiFi.

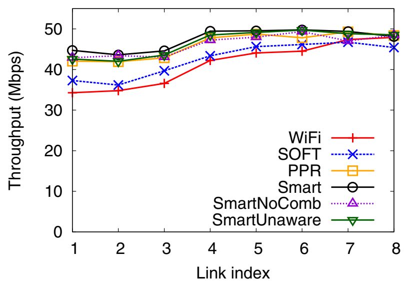
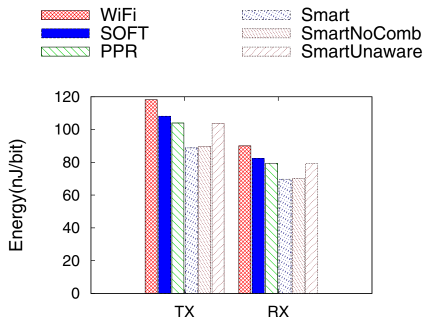
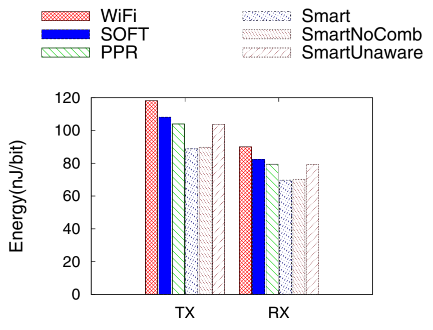


Fig. 14. Throughput under interference with interference link power scaled down by 0.1 and collision probability is 10%.

Under interference: Next we evaluate the performance under interference. We select a trace of one link as the foreground traffic, and a trace of another link as the interference and let these two transmissions collide randomly with a varying probability. As shown in Figure 14, Smart out-performs the other schemes under interference. The throughput gains over WiFi are up to 30%. As before, SmartNoComb continues to perform well due to its effective rate adaptation. The benefit of PPR and SmartUnaware over WiFi increases in the presence of interference because interference allows them to perform partial retransmissions more frequently. In comparison, the performance benefit of SOFT is still limited because it retransmits complete frames even though significant portions of the frames have already been received correctly.



(a) Static traces



(b) Mobile traces

Fig. 15. Energy consumption using Intel traces with 4000-byte frames.

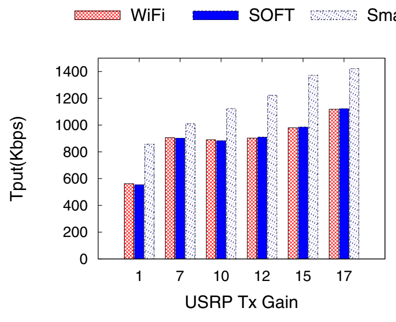
Energy comparison: As mentioned earlier, Smart Retransmission can not only improve throughput, but also reduce energy consumption since it reduces transmission time. We assume the access point (AP) runs the rate adaptation scheme, and quantify the energy consumption of a client that is either transmitting to or receiving from the AP. Figure 15(a) and (b) compares the average energy consumption of all schemes using 4000-byte frames in the static and mobile traces, respectively. As we can see, in the static traces, Smart reduces WiFi transmitter energy by 25%, SOFT by 18%, PPR by 15%, SmartNoComb by 1%, and SmartUnaware by 15%. The receiver also shows similar energy savings. For mobile trace, the energy savings are 18% over WiFi, 11% over SOFT, 8.5% over PPR, 7% over SmartUnaware and 3% over SmartNoComb. Similarly, the Smart receiver consumes less energy due to reduced reception time.

VII. Testbed Experiments

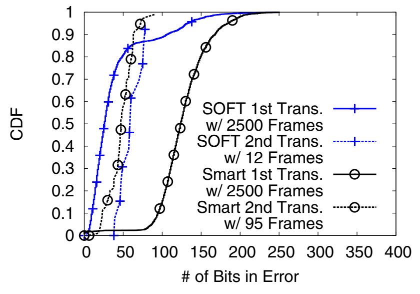
Next we evaluate the performance of our scheme using USRP. We have two USRP nodes serve as a transmitter and receiver, and the third USRP node is used to inject narrowband interference. Both flows use 1 MHz channel. The interference allows us to introduce frequency diversity (common in a 20 MHz WiFi channel) to a 1MHz USRP channel. The center frequency of the interfering USRP is selected so that it partially overlaps with the foreground flow for a given number of subcarriers. As in the simulation, we account for the feedback overhead in the testbed experiments. We ignore the DIFS/SIFS overhead since they are negligible compared to the data transmission time under 1 MHz channel.

Varying Transmit Power: We first evaluate the performance of our scheme by varying the transmit power of the foreground flow through changing its TX gain. The transmit power and bandwidth of the background interfering node are both fixed. Figure 16(a) shows the average throughput of 1000-byte frames using Smart, SOFT, and WiFi as we vary TX gains. For each TX gain value, we average across five runs. Smart outperforms both WiFi and SOFT by 11–52% due to its combining-aware rate adaptation and combining gain. This effect is most prominent for Tx-Gain 15 when both WiFi and SOFT transmit at MCS-3 while Smart transmits at MCS-5 during the first transmission and uses the MCS-7 for retransmission, thereby achieving a much higher throughput. SOFT performs similar to WiFi since the selected rate lets most transmissions succeed in one try and there is little opportunity for SOFT to leverage the combining gain.

Figure 16(b) shows the CDF of the number of erroneous bits in a frame after the first and second transmissions for SOFT and Smart under Tx-Gain 15. As mentioned above, Smart selects MCS-5 for the first transmission, which results in more bits in error when compared to SOFT, which selects MCS-3. However, the number of erroneous bits after the second transmission drop quickly in Smart due to its effective combining and rate adaptation. The numbers of erroneous bits in Smart and SOFT do not need to be zero in order for a frame to be delivered successfully due to the use of FEC.



(a) Throughput



(b) CDF of number of erroneous bits

Fig. 16. Throughput under varying transmission power and CDF of number of erroneous bits in 1000-byte frames.

Varying Interference Bandwidth: Next we fix the transmit power and the bandwidth of both transmitters, and vary the amount of the overlapping bandwidth between the foreground and background flows by changing the center frequency of the background interference. The center frequency of the foreground flow is 2490MHz, and the center frequency of the background interference varies from 2490.7MHz to 2490.6MHz to 2490.2MHz. These center frequencies correspond to an overlapping region of 30%, 40% and 80%.

Figure 17 compares the average throughput of Smart, SOFT, and WiFi for these settings using 1000-byte frames. The throughput is averaged over five runs. Smart again shows significant throughput improvement: it outperforms WiFi and SOFT by 40%, 14% and 36% under these overlap settings, respectively. The improvement is the largest when the number of bad subcarriers is small because Smart needs to retransmit only a few subcarriers. However, even when 80% of the subcarriers have interference, Smart is able to out-perform WiFi by 36% due to its effective rate selection.

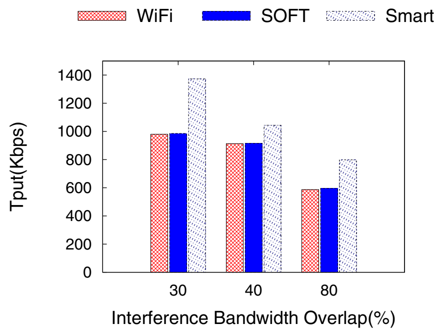


Fig. 17. Throughput under varying interference bandwidth.

VIII. Conclusion

This paper presents Smart Retransmission to harness temporal diversity by combining demodulated bits before FEC from multiple (possibly partial) transmissions and using combining-aware rate adaptation. Our evaluation shows combining-aware rate adaptation is essential to harnessing the gain of partial retransmission and combining. The concept of selecting the data rate that is aware of reduced retransmission cost is general, and can be applied to PPR and SOFT, as in SmartNoComb. Moreover, we show that this approach can not only increase throughput but also save energy.

Acknowledgements: This work is supported in part by NSF Grants CNS-1343383 and CNS-1017549, and the National Science Council of R.O.C. under contract No. MOST 103-2221-E-001-017-MY2 and MOST 104-2628-E-001-004-MY2 Academia Sinica of R.O.C. under project No. AS-104-TP-A05. We thank reviewers especially Chunyi Peng for their insightful feedback.

References

[1] LAN/MAN Standards Commmittee of the IEEE Computer Society. Part 11: Wireless LAN Medium Access Control and Physical Layer (PHY) Specifications. *IEEE Standard 802.11*, 2009. http://standards.ieee.org/getieee802/download/802.11n-2009.pdf.

[2] G. Angelopoulos, A. Chandrakasan, and M. Medard. PRAC: exploiting partial packets without cross-layer or feedback information. In *Proc. of IEEE ICC*, 2014.

[3] From BCJR to turobo decoding: MAP algorithms made easier. http://paginas.fe.up.pt/~sam/textos/From%20BCJR%20to%20turbo.pdf.

[4] BCJR decoding: The BCJR algorithm. http://www.ii.uib.no/~eirik/INF244/Lectures/Lecture11.pdf.

[5] G. Bhanage, R. Mahindra, I. Seskar, and D. Raychaudhuri. Implication of MAC frame aggregation on empirical wireless experimentation. In *Proc. of IEEE GLOBECOM*, 2009.

[6] J. Bicket. Bit-rate selection in wireless networks. In *MIT Master's Thesis*, 2005.

[7] D. Chase. Code combining–a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets. *IEEE Transactions on Communications*, 33(5):385–393, May 1985.

[8] S. Gollakota and D. Katabi. ZigZag Decoding: Combating hidden terminals in wireless networks. In *Proc. of ACM SIGCOMM*, 2008.

[9] M. Gowda, S. Sen, R. R. Choudhury, and S.-J. Lee. Cooperative packet recovery in enterprise WLANs. In *Proc. of IEEE INFOCOM*, 2013.

[10] A. Gudipati and S. Katti. Strider: Automatic rate adaptation and collision handling. In *Proc. of ACM SIGCOMM*, 2011.

[11] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *Proc. of ACM SIGCOMM*, 2010.

[12] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Tool release: gathering 802.11n traces with channel state information. *SIGCOMM Comput. Commun. Rev.*, 41(1), Jan. 2011.

[13] IT++ C++ Library. http://itpp.sourceforge.net/4.3.1/.

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

[15] P. S. Kalekar. Time series forecasting using holt-winters exponential smoothing. Dec. 2004. http://www.it.iitb.ac.in/~praj/acads/seminar/04329008_ExponentialSmoothing.pdf.

[16] M. O. Khan, V. Dave, Y.-C. Chen, O. Jensen, L. Qiu, A. Bhartia, and S. Rallapalli. Model-driven energy-aware rate adaptation. In *Proc. of ACM MobiHoc*, July 2013.

[17] Q. Li, B. Li, Y. Liu, and L. Zhang. An adaptive modulation selection scheme based on error estimating coding. In *Proc. of MSN*, 2011.

[18] S. Lin and P. Yu. A hybrid ARQ scheme with parity retransmission for error control of satellite channels. *IEEE Transactions on Communications*, 30(7):1701–1719, July 1982.

[19] M. Luby. LT codes. In *Proc. of FOCS*, 2003.

[20] D. Mandelbaum. An adaptive-feedback coding scheme using incremental redundancy (corresp.). *IEEE Tran. on Info. Theory*, 20(3), 1974.

[21] Meraki white paper: 802.11n technology. *Meraki White Paper*, 2011.

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

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

[24] J. Ou, Y. Zheng, and M. Li. MISC: merging incorrect symbols using constellation diversity for 802.11 retransmission. In *Proc. of IEEE INFOCOM*, 2014.

[25] I. Pefkianakis, Y. Hu, S. H. Wong, H. Yang, and S. Lu. MIMO rate adaptation in 802.11n wireless networks. In *Proc. of MobiCom*, 2010.

[26] J. Perry, P. A. Iannucci, K. E. Fleming, H. Balakrishnan, and D. Shah. A rateless wireless communication system using spinal codes. In *Proc. of ACM SIGCOMM*, Aug. 2012.

[27] A. Shokrollahi. Raptor codes. *IEEE Tran. on Info. Theory*, 2006.

[28] E. Uhlemann, L. Rasmussen, A. Grant, and P. Wiberg. Optimal incremental-redundancy strategy for type-II hybrid ARQ. In *Proc. of Information Theory*, 2003.

[29] USRP. http://www.ettus.com/.

[30] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *Proc. of SIGCOMM*, 2009.

[31] 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.

[32] G. Woo, P. Kheradpour, D. Shen, and D. Katabi. Beyond the bits: Cooperative packet recovery using physical layer information. In *Proc. of ACM MobiCom*, 2007.