Performance Evaluation 68 (2011) 1118–1135

Contents lists available at SciVerse ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

FEC-based AP downlink transmission schemes for multiple flows: Combining the reliability and throughput enhancement of intra- and inter-flow coding

Chih-Chun Wang [a], Dimitrios Koutsonikolas [b], Y. Charlie Hu [a,*], Ness Shroff [c]

[a] Purdue University, United States
[b] University at Buffalo, SUNY, United States
[c] The Ohio State University, United States

ARTICLE INFO

Article history:
Available online 3 August 2011

Keywords:
FEC
Intra-flow coding
Inter-flow coding

ABSTRACT

We consider downlink access point (AP) networks and the corresponding reliable transmission schemes. It is well known that one can protect the network traffic against packet erasures by forward-error-correcting-codes (FEC). In addition to ensuring reliable delivery, FECs could substantially reduce the amount of feedback traffic, which is critical when designing high-performance AP protocols.

In this work, we generalize the FEC-based schemes, also known as intra-flow coding schemes, for multiple downlink flows. In contrast with the classic approach that performs FEC *separately* on individual flows, we propose a new protocol MU-FEC, which incorporates the recent idea of *inter-flow coding* to further enhance the achievable throughput. Specifically, MU-FEC guarantees 100% reliability, is oblivious and robust to the underlying erasure probabilities, has near-optimal throughput higher than any existing inter-flow coding protocols, and can be practically implemented on top of 802.11. The design of MU-FEC consists of three components: batch-based operations, a systematic phase-based network coding decision policy, and smooth integration of inter-flow and intra-flow coding. We analytically show that MU-FEC can achieve much higher throughput than intra- or inter-flow coding alone, and validate its performance gain via extensive simulations. To our knowledge, MU-FEC is the first practical protocol that leverages both intra-flow and inter-flow network coding to solve a real-world problem in single-hop wireless networks.

1. Introduction

Consider a typical scenario of an 802.11 WLAN. Multiple clients are associated to the access point (AP). The AP forwards the traffic between each client and the wired Internet and uses 802.11 unicast for packet transmissions between itself and the clients. We focus on the downlink traffic, *i.e.,* in the direction from the AP to each client, as the downlink traffic dominates the uplink traffic in typical AP deployments.

To deal with packet losses due to the inherent lossy wireless medium and due to interference and collisions from other clients and other WLANs in the neighborhood, the 802.11 protocol employs a simple retransmission mechanism for unicast communication. A sender waits for an acknowledgment (ACK) after each packet transmission and retransmits the packet if the ACK is not received after a short, fixed amount of time. Each packet is retransmitted up to a maximum number of

* Corresponding author.
E-mail address: ychu@purdue.edu (Y.C. Hu).

0166-5316/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.peva.2011.07.021

times and is then dropped. Under high loss rates, this simple retransmission mechanism can cause significant overhead and severely limit the throughput of the WLAN, as the AP may spend a significant portion of the airtime retransmitting lost packets. This motivates the need for novel, more efficient retransmission schemes.

It is well known that one can use Forward Error-correcting Codes (FECs), also known as intra-flow network coding [1], to ensure 100% delivery ratio as well. That is, the AP keeps sending randomly mixed packets generated from a single batch. Once the destination receives a sufficient number of packets for decoding, a *batch ACK* is sent back to the AP. AP can then move on to the next batch. The benefits of FEC-based schemes are multi-fold. First, 100% reliability is guaranteed. Second, when there is only a single flow in the network, the achievable throughput approaches the theoretic optimum. Third, the amount of feedback traffic is substantially reduced and the network thus experiences less interference. Fourth, without complicated timeout mechanisms, the FEC scheme is oblivious to the underlying erasure probabilities and offers robust performance even when the channel conditions evolves over time. The main cost of a FEC scheme, in exchange of the aforementioned benefits, is the incurred delay during batch decoding. An FEC scheme is thus an ideal candidate for high-rate data traffic.

The above FEC scheme can also be viewed as an *intra-flow coding* scheme as packet-mixing is performed within the single flow. In contrast, a different form of network coding, termed *inter-flow coding*, can be used to significantly enhance the achievable throughput (while leaving the reliability and robustness parts untouched). Specifically, consider an AP and two clients C_1, C_2. The AP has two packets, p_1, p_2, one for each client, respectively. In a lossy wireless network, it may happen that both packets are lost and the AP would have to retransmit both of them. However, due to the broadcast nature of the wireless medium, it can also happen that C_1 received the packet p_2 destined to C_2 and C_2 received the packet p_1 destined to C_1. In that case, the AP can XOR the two packets and broadcast the combined packet $p_1 \oplus p_2$. Then C_1 can extract its own packet p_1 by XORing $p_1 \oplus p_2$ with p_2 and similarly, C_2 can extract p_2 by XORing $p_1 \oplus p_2$ with p_1, thus saving one transmission. This simple idea can be extended beyond the two-client example, further improving the achievable throughput.

The potential benefits of inter-flow coding based AP downlink transmission scheme were first considered by MU-ARQ [2]. For subsequent reference, we termed the ideas in [2] the MU-ARQ principle. A practical implementation of the MU-ARQ principle was proposed in [3]. Nonetheless, the throughput benefits of MU-ARQ rely on carefully coordinating multiple concurrent flows. This further exacerbates the problem of designing the corresponding ACK and retransmission mechanisms. Therefore, complicated time-out mechanisms with unpredictable time dynamics have to be implemented [3]. The resulting protocol is thus sensitive to the number of flows to be coded together and the underlying packet erasure probabilities, and *does not guarantee* 100% *reliability*.

In this paper, we propose a new protocol MU-FEC, which combines the robustness of intra-flow coding with the throughput enhancement feature of inter-flow coding. Specifically, MU-FEC inherits the same simple interactive relationship between the AP and the users of any FEC-based scheme: AP keeps sending coded packets until receiving batch ACK from the users. This thus enables reliable and robust performance that is oblivious to the underlying packet erasure probabilities without resorting to complicated time-out mechanisms. Moreover, with new advanced network code design (describing how to generate the FEC coded packets), MU-FEC also fully captures the throughput advantage of MU-ARQ inter-flow coding principle. As will be discussed further in Section 2, the existing inter-flow coding protocol [3] faces an NP-hard challenge: Deciding which flows should be mixed together for the next coded packet transmission. Several heuristics are thus used in ER [3], which further reduce the achievable throughput and make its performance sensitive to the underlying network environment. In MU-FEC, the inherent FEC-based structure (intra-flow coding) allows us to robustly circumvent the above challenge without resorting to heuristics and thus no performance degradation. Furthermore, we show analytically that MU-FEC achieves the optimal broadcast channel capacity with feedback [4–6], which strictly outperforms the theoretically achievable rate of the MU-ARQ principle. The main cost of MU-FEC is the incurred delay during batch decoding, which is inevitable for any FEC-based schemes.[1]

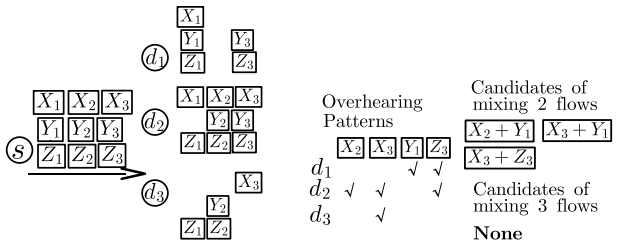
Our extensive simulations also show that MU-FEC achieves throughput *close to the theoretically optimal* in both low- and high-loss rate environments. The gap between MU-FEC and the best possible throughput of any schemes, termed the *theoretic gap*, is at most 9%–16% for MU-FEC. For comparison, the theoretic gap of the ER protocol is 30%–38%. Note that ER is sensitive to parameter tuning. If we assume that the AP is able to fine tune the parameters of ER on the fly (such optimization is done by us, not by the original paper [3]), then the theoretic gap of ER can be reduced to 14%–21%. As a result, MU-FEC with reliable delivery outperforms unreliable ER protocol by 12%–42% depending on whether on-the-fly fine tuning of ER is allowed. Based on FECs, the performance of MU-FEC is also very robust in various network environments, consistently outperforming ER under both homogeneous and heterogeneous loss patterns. In short, MU-FEC successfully combines the benefits of both intra- and inter-flow coding and solves a real-world problem in single-hop wireless networks.

2. Previous results

2.1. From the MU-ARQ principle to the theoretic capacity

One of the earliest papers regarding inter-flow coding for AP networks is the MU-ARQ principle in [2]. The analysis of the MU-ARQ principle is straightforward under several idealistic assumptions, including zero-cost feedback, infinitely large

[1] The delay is mostly determined by the batch size. For the same batch size, the delay of MU-FEC is smaller than that of the existing FEC schemes since the higher throughput of MU-FEC ensures that each user can accumulate packets for decoding in a faster way than the existing FEC schemes.



(a) The overhearing patterns. (b) Coding opportunities.

Fig. 1. Illustration of the MU-ARQ protocol.

batch sizes, and perfect channel symmetry. Accordingly, [2] quantifies the *throughput efficiency* of the MU-ARQ principle, which is also known as the *sum-rate capacity* and is defined as

$$\eta = \frac{MN}{T} \tag{1}$$

where M is the number of clients, N is the number of packets at the AP for each client, and T is the overall number of time slots required to finish the transmission of the MN packets. Obviously, η is at most 1 by definition as each time slot can carry at most one packet even with perfect channels. The higher the throughput efficiency η, the larger throughput is for the entire system. The throughput efficiency of MU-ARQ for infinitely large N is computed in [2] by

$$\lim_{N\to\infty} \eta_{\text{MU-ARQ}} = \frac{1-(1-p)^M}{1+\frac{1-p}{Mp^2}\left(1-(1-p)^M - Mp(1-p)^{M-1}\right)} \tag{2}$$

where it is assumed that the probability that a packet sent by the AP is received by Client C_i successfully is p for all $i \in [M]$, and the success events are independent for different Clients $C_i \neq C_j$.

The achievable throughput of the MU-ARQ is significantly higher than the traditional FEC-only schemes, which have $\eta_{\text{FEC.only}} = p$. On the other hand, recent theoretic studies [4–6] show that the throughput efficiency of any scheme must obey

$$\eta \leq \frac{M}{\sum_{k=1}^{M} \frac{1}{1-(1-p)^k}}. \tag{3}$$

Moreover, theoretically one can design an optimal scheme that achieves η^* equal to the outer bound (3) and thus strictly outperforms an MU-ARQ scheme (since "(3) $\geq$ (2) $\geq \eta_{\text{FEC.only}} = p$" always holds).

The remaining question is thus whether the throughput benefits of the theoretically optimal schemes in [4–6] can be robustly realized in an ever-changing practical environment when considering the necessary overheads. Our work answers this question in an affirmative way.

2.2. Limitations of the MU-ARQ protocol

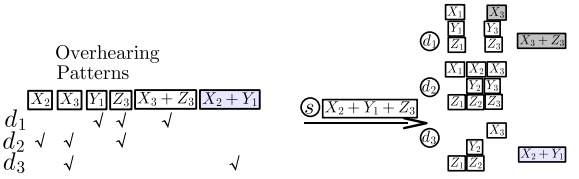
There are several inherent challenges when implementing the idealistic MU-ARQ principle. We use the example in Fig. 1 for illustration.

Suppose in the initial stage, the AP s transmits 9 native packets among which X_1 to X_3 are intended for client d_1; Y_1 to Y_3 are intended for d_2; and Z_1 to Z_3 are intended for client d_3, respectively. All 9 packets are sent using wireless broadcast, and due to the randomness of the wireless channel, clients d_1 to d_3 may have different overhearing patterns, as illustrated in Fig. 1(a).

MU-ARQ takes advantage of the "diversity" of the overhearing patterns at the clients. For example, packet X_2 is heard by d_2 but not by $\{d_1, d_3\}$, packet X_3 is heard by both d_2 and d_3 but not d_1. Packet Y_1 is heard only by d_1 and packet Z_3 is heard only by $\{d_1, d_2\}$. See Fig. 1(b) for the list of the 4 overheard packets that may lead to potential coding opportunities. MU-ARQ then searches for the inter-flow coding opportunities that combine 2 flows (or even 3 flows) together. See Fig. 1(b) for the opportunities mixing 2 flows. Since there is no Y packet that is heard by both $\{d_1, d_3\}$, there is no coded packet in this example that can simultaneously mix 3 flows.

An inherent limitation of MU-ARQ is the following: *Issue* 1: *The protocol is overly conservative and exploits coding opportunities in a passive way*. Continue our example in Fig. 1. Suppose the AP decides to send $[X_3+Z_3]$ and then $[X_2+Y_1]$. Due to the channel randomness, $[X_3+Z_3]$ and $[X_2+Y_1]$ are heard by d_1 and by d_3, respectively; see Fig. 2(a) for the overhearing pattern table. By hearing $[X_3+Z_3]$, d_1 can now decode X_3 as illustrated in Fig. 2(b).

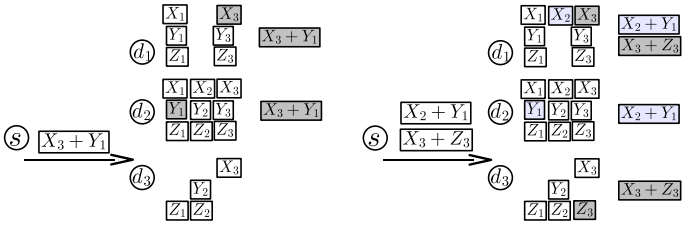
The remaining question is thus *what is the optimal choice of the next coded packet*. MU-ARQ answers this question in the following conservative way. We first notice that the coded packet $[X_2+Y_1]$ heard by d_3 cannot be used to decode any



(a) The overhearing patterns when the AP sends $[X_3 + Z_3]$ and then $[X_2 + Y_1]$.

(b) Sending $[X_2 + Y_1 + Z_3]$ will take advantage of the previously overheard $[X_2 + Y_1]$ packet.

Fig. 2. Aggressive exploitation of the overheard coded packets.



(a) MU-ARQ sending $[X_3 + Y_1]$ first. At least two more transmissions are necessary.

(b) MU-ARQ sending $[X_3 + Z_3]$ and $[X_2 + Y_1]$ instead. No further transmission is necessary.

Fig. 3. The impact of different choices of coded packets.

additional information, since d_3 has heard neither X_2 nor Y_1 in the past. As a result, MU-ARQ discards the overheard coded packet $[X_2 + Y_1]$ at d_3. Since each of d_1 to d_3 only needs one additional packet (more explicitly, they need X_2, Y_1, and Z_3, respectively), MU-ARQ checks whether they can be mixed together. Since X_2 is now heard only by d_2 (resp. Y_1 is heard only by d_1) as illustrated in Fig. 2(a), these three packets will not be mixed together by MU-ARQ. As a result, MU-ARQ needs at least two additional packets to complete transmission.

On the other hand, we can *aggressively exploit the coded packet* $[X_2+Y_1]$ *overheard by* d_3. That is, we can broadcast a coded packet $[X_2 + Y_1 + Z_3]$. If d_1 receives such packet, d_1 can *decode* the desired X_2 packet by subtracting Y_1 and Z_3 based on d_1's existing knowledge about Y_1 and Z_3. If d_2 receives such packet, d_2 can decode the desired Y_1 packet by subtracting X_2 and Z_3 based on d_2's existing knowledge about X_2 and Z_3. For d_3, d_3 has neither the knowledge about X_2 nor the knowledge about Y_1, which is the reason why MU-ARQ chooses not to mix the three packets together in the first place. However, we note that although d_3 does not know the individual packets X_2 and Y_1, d_3 did overhear a coded packet $[X_2 + Y_1]$. Therefore, d_3 can still decode the desired Z_3 by *subtracting the previously heard coded packet* $[X_2 + Y_1]$ from the new received packet. In this way, we *recoup the benefits of overheard coded packets that are not immediately decodable*, which further pushes throughput efficiency from the MU-ARQ scheme (2) closer to the optimal (3).

Issue 2: *The performance of ER with a finite batch size is sensitive to the decision of which coded packet to send in each time slot*. Previously, we assumed that the AP first sends $[X_3 + Z_3]$ and then $[X_2 + Y_1]$. In Fig. 1(b) we note that there is actually one other coding opportunity $[X_3 + Y_1]$ in addition to the previous choices $[X_3 + Z_3]$ and $[X_2 + Y_1]$. What if the AP decides to send $[X_3 + Y_1]$ first (instead of the previous choices)? Also suppose that both d_1 and d_2 receive $[X_3 + Y_1]$ and use it to decode X_3 and Y_1, respectively. After the first packet, the new overhearing pattern becomes Fig. 3(a).

Now d_2 has received all three Y packets; d_1 only needs to receive one more packet X_2; d_3 only needs to receive one more packet Z_3; but X_2 and Z_3 *cannot be coded together* since X_2 is not overheard by d_3. As a result, we need at least two more time slots to complete transmission. The total number of necessary transmissions is thus $1 + 2 = 3$ when we send $[X_3 + Y_1]$ first. In contrast, suppose we use the original choices $[X_2 + Y_1]$ and $[X_3 + Z_3]$ and they are heard by $\{d_1, d_2\}$ and by $\{d_1, d_3\}$, respectively; see Fig. 3(b) for the overhearing pattern table. In this case, *all three clients* d_1 *to* d_3 *can decode all the desired* X, Y, *and* Z *packets after two transmissions*. The decision of not sending $[X_3 + Y_1]$ first thus saves 1 transmission.

Deciding which packets to be sent first turns out to be highly non-trivial (NP-hard in some cases) [3]. By assuming the wireless channels are noiseless, [3] proposed to use an integer-programming (IP) solver for this problem. However, since in practice the wireless channel is not perfect, the coding decision of ER is far from optimal even when using the optimal IP solver.

The above two limitations plus the timeout-based design substantial hinder the performance of ER. For example, as pointed out in [3] and verified empirically in Section 6, the performance gain of [3] degrades substantially when the number of to-be-mixed flows increases and when in a high-loss rate environment. We argue that such performance loss is caused strictly by the implementation choice of ER. The intuition behind is that the more number of flows to be mixed together, the

more coding opportunities to choose from, and the higher the throughput should be. And the higher the loss rate is, the more "diverse" of the overhearing pattern (more frequently to see a packet received by one but not the other when compared to a packet received by both clients), the higher the coding gain should be. Such intuition also fits the theoretic prediction in (2) and (3). As we will see in Section 3, by using the FEC (equivalently intra-flow coding) as a building block, we can successfully circumvent the above limitations. The resulting MU-FEC protocol demonstrates properly monotonic increasing improvement with respect to the number of flows and the loss rates, and achieves robust throughput under various network environments.

2.3. Related work

Recent results that combine intra- and inter-flow coding. The smooth combination of the benefits of intra- and inter-flow coding has been a grand challenge for wireless network designers. Recently, [7] proposes to combine intra- and inter-flow coding in a wireless relay environment. Compared to our work, their approach of combining intra- and inter-flow coding is performed at the batch-level. That is, intra-flow coding is performed within a batch, which alters the "overhearing pattern from a batch's perspective". Then inter-flow coding is performed based on the overhearing batches. The *C*&*M* protocol [8] also combines the two types of coding at the batch level but in reverse order: a node first creates a batch of inter-flow coded packets by mixing packets belonging to different flows and then sends out linear combinations of the created coded packets. Finally, I^{2}MIX [9] simply performs random linear network coding on all $\sum_{i=1}^{M} N_i$ packets (assuming M flows and a batch of N_i packets for flow i). As we will see in Section 3.1, this approach can be highly inefficient. Note that all these works refer to a topology and traffic pattern fundamentally different from the one we are considering in this paper—multihop wireless mesh networks with multihop flows intersecting at some intermediate router vs. single hop WLANs with single hop flows from the same source (AP) to multiple clients.

In our work, the integration of intra- and inter-flow coding is performed at the packet level. MU-FEC makes the joint intra- and inter-flow coding decision for each packet based on the overhearing patterns at the packet level. As can be seen, such low-level packet-based combination is key to the robustness and high performance of MU-FEC.

Theoretic studies. The 1-hop WLAN corresponds to the classic "broadcast channel" problem in the information theory society, which has been an active research subject in the past four decades. Some example related works in this extremely rich literature include the 2-user feedback capacity exploration for the erasure channel [5,10], the Gaussian channel [11], and the discrete memoryless channels [12]. The results in this work can also be viewed as a generalization of the *index coding* problem [13] from the noiseless channels to the random wireless broadcast erasure channels.

Other network coding based retransmission schemes. ER [3] was the first practical implementation of the MU-ARQ principle. The design of ER aims to capitalize the throughput benefits of inter-flow coding while addressing several practical issues, such as feedback frequency, packet delay time-out, link asymmetry, that were previously ignored in MU-ARQ. Recently, XORR [14], was proposed to improve the performance of ER by considering the impact of different link data rates on the coding decision and incorporating a network-coding-aware opportunistic scheduler to the AP, to provide fairness among clients of different link qualities. However, XORR's coding strategy can be far from optimal. As in MU-ARQ/ER, XORR also drops overheard coded packets that are not immediately decodable, and only considers the head-of-line packet of each flow when making coding decisions, which also limits the coding gain. Note that the idea of MU-ARQ appeared in dozens of papers other than [2]. Some other theory papers that use the same idea of MU-ARQ can be found in [15,16]. However, none of them achieves the theoretic optimum (3) for ≥ 3 clients.

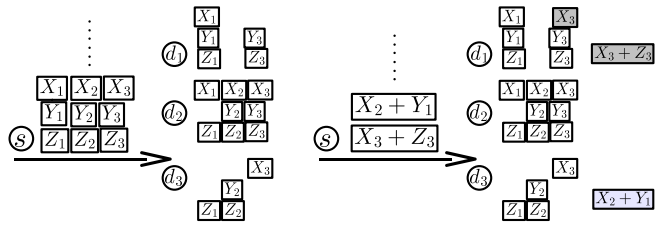
3. MU-FEC: main ideas

The MU-FEC protocol consists of three important design components: (i) Batch-based operations, (ii) A systematic phase-based coding decision policy, and (iii) The smooth combination of inter-flow coding and intra-flow coding. MU-FEC not only enables practical implementation but also admits provably optimal theoretical performance, when there is a sufficiently large number of clients in the system. In this section, we discuss the main ideas in the design of MU-FEC. We then give a detailed description of the protocol in Section 4.

3.1. Prerequisite: the batch-based approach

We first note that it is generally more beneficial to wait for more coding opportunities before starting inter-flow coding. However, as it is impractical to wait indefinitely for new coding opportunities, we use a batch-based design that curtails the delay impact and also regulates the associated header size for each coded packet. As will be seen later, this batch-based approach enables new systematic design of recouping overheard coded packets and scheduling new coded transmissions that are difficult to implement in a timer-based approach [3].

Batch-based operation: Each flow i has N_i packets to transmit in each batch. If there are M flows to be served by the AP, then an *inter-flow batch* contains $\sum_{i=1}^{M} N_i$ packets. The protocol will intelligently decide how to mix the $\sum_{i=1}^{M} N_i$ packets. Our goal is to finish the transmission of the N_i packets for each flow in the shortest amount of time.



(a) The Phase 1 operation. (b) The Phase 2 operation.

Fig. 4. Phase-based operation of the proposed algorithm.

3.2. Addressing Issue 1: phase-based inter-flow coding

A critical observation toward recouping the benefits of overheard coded packets is that for any given packet, the number of overhearing clients is monotonically increasing. For example, suppose a packet X intended for d_1 is heard by d_2, and we retransmit X for the second time. This time, X is received only by d_3. Even though d_2 does not receive X in the second time, d_2 still knows X from the first transmission. Therefore, X is now overheard by both d_2 and d_3. The number of clients overhearing X thus increases monotonically over time. Similar statements hold for inter-flow coded packets as well. That is, if a coded packet is created by mixing two flows, then the overheard coded packet can only create new coding opportunities for mixing >2 flows. In our example of Fig. 2(b), the coded packet $[X_2 + Y_1]$ is created for mixing 2 flows. Once $[X_2 + Y_1]$ is overheard by d_3, it can be used for mixing 3 flows.

The above observation prompts the following phase-based design. We define a *k-flow packet* as a packet that mixes exactly k flows together, k ranging from 1 to M. Any k-flow packet thus serves k clients simultaneously. If a k-flow packet is heard by a client other than the intended k clients, it is not immediately decodable. As discussed previously, MU-FEC will recoup this kind of packet and use it as side information when mixing over $>k$ flows in the future. Since overhearing a k-flow packet may only create a new opportunity for mixing h flows with some $h > k$, an optimal solution is to send out k-flow packets first, and hopefully this will create new opportunities for mixing h flows together where $h > k$. Based on this reasoning, we divide an inter-flow batch into M phases. Each batch goes through Phases 1 to M in sequence. During Phase k, the AP only sends out k-flow packets; overhearing these packets can create new coding opportunities of mixing $h > k$ flows in the future phases.

An example of phase-based operations: We again use a 3-client scenario as our running example, see Fig. 4. Each inter-flow batch contains $N_1 + N_2 + N_3 = 9$ packets. In Phase 1 (see Fig. 4(a)), the AP transmits packets mixing only 1 flow, i.e., no inter-flow coding is performed. As can be seen, some Phase 1 packets successfully reach the intended client while some do not. The latter are called the overheard packets, and in our example, X_2, X_3, Y_1, and Z_3 are overheard packets. These overheard Phase-1 packets will later be used in Phases 2 and 3.

After spending some time in Phase 1, AP switches to Phase 2 based on the feedback from the clients. In Phase 2, AP starts to send packets mixing exactly 2 flows, see Fig. 4(b). In our example, based on the overhearing record, Phase 2 sends packets like $[X_3 + Z_3]$ and $[X_2 + Y_1]$ that mix the overheard Phase-1 packets.

Again, after spending some time in Phase 2, AP switches to Phase 3 based on the feedback from the clients. In Phase 3, AP sends packets mixing exactly 3 flows. One such example is the $[X_2 + Y_1 + Z_3]$ packet described in Fig. 2(b). The way we create it is by mixing two packets $[X_2 + Y_1]$ and $[Z_3]$. Again, $[Z_3]$ is an overheard packet transmitted in Phase 1. $[X_2 + Y_1]$ is an overheard packet transmitted in Phase 2, which is now recouped in the MU-FEC protocol (in contrast with being discarded in ER). The Phase-3 packet $[X_2 + Y_1 + Z_3]$ is indeed constructed by overheard packets in the previous Phases k, $k < 3$. By monotonically progressing from Phases 1 to 3, we increase the maximum amount of k-flow coding opportunities for each individual phase.

3.3. Addressing Issue 2: intra-flow coding within a single phase

The phase-based operation gives higher priority to sending k-flow packets for a smaller k. However, as discussed in the example of Fig. 3, even when we are only sending packets mixing the same number of flows, it is still critical to decide which packet should be sent earlier. We observe that the problem of deciding which packet to send under a noisy broadcast channel model is essentially identical to the packet forwarding problem in the original *opportunistic routing* protocol ExOR [17]. Motivated from the success of using *random intra-flow coding* to efficiently solve the packet forwarding problem (the MORE protocol [18]), MU-FEC uses the same heuristic, i.e., random intra-flow coding (with the coding coefficients randomly chosen from a Galois Field GF(2^b)), to simplify the packet selection decision, without resorting to complicated integer programming solvers or simplified suboptimal heuristics.

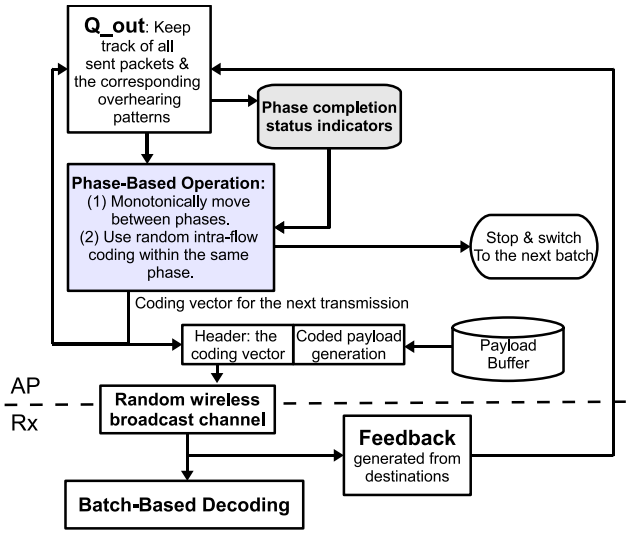


Fig. 5. MU-FEC flow chart.

3.4. When to stop (in a given phase)? a new rank-aware phase transition

The key concept of any FEC (intra-flow coding) scheme is to use feedback to inform the AP that a client has collected a sufficient number of packets and the AP can stop transmission. Since we now have several phases within a batch, the key question becomes when to stop the transmission in the current Phase k and move on to Phase $(k+1)$; and when to stop transmission for the current batch and move on to the next batch. Note that if we move to the next phase too late, then the throughput suffers as it takes longer than necessary to finish transmission. If we move to the next phase too early, some clients may fail to receive all the desired information. Since the "information of network coded traffic" is quantified by the rank of the corresponding linear space, we design a near-optimal mechanism that traces the evolution of "the ranks for the individual phases" and use it as an indicator when to move from one phase to the other.

4. MU-FEC design

4.1. Protocol overview

A flow chart of the protocol operation at the AP and the client is given in Fig. 5.

The AP maintains M buffers $B_i,\ i = 1, \ldots, M$ (jointly shown as *Payload Buffer* in Fig. 5); the i-th buffer stores the payloads of the N_i packets for the current batch. To fully exploit/recoup all possible coding opportunities, the AP also needs to keep track of all data packets in the air. For that purpose, the AP keeps a queue Q_{out} that stores the inter-flow coding vectors (not the payloads) of all the outgoing packets for the current batch. Each inter-flow coding vector $\mathbf{v}$ stores the coding coefficients for the $N_{\text{total}} \triangleq \sum_{i=1}^{M} N_i$ packets of the M flows, i.e., it is a row vector of size N_{total}. We set the maximum allowable number of vectors in Q_{out} to be $10 \times \left(\sum_{i=1}^{M} N_i\right)$. Each coding vector is associated with an *overhearing bit map* and a *creation bit map*, each of size M bits, and a unique 2-byte sequence number. The overhearing bit map records which client has overheard this packet. Since each outgoing coding vector/packet is created by mixing a subset of M flows, the creation bit map of an inter-flow vector is used to indicate which flows were used to generate this vector.

The unique sequence number for each inter-flow coding vector is to facilitate cumulative feedback. The sequence number is defined within the same inter-flow batch, not within the same flow. Therefore, each client periodically sends back a list of sequence numbers acknowledging which coded/uncoded packets it has received without the need to distinguish which flows the packets belong to. Upon the receipt of the list of ACK'ed sequence numbers, the Q_{out} updates the corresponding overhearing map accordingly.

The Q_{out} buffer is used by the phase-based operation to make the coding decision, which starts from Phase 1 and ends in Phase M. When the phase-based operation is in Phase k, the AP generates a coded k-flow packet and broadcasts it whenever there is a transmission opportunity. More explicitly, the phase-based operation generates the inter-flow coding vector for the next to-be-transmitted packet. The coding vector is then used by a separate routine to construct the coded payload from the payload of native packets stored in the payload buffer.

As discussed in Section 3, the monotonic progression from Phase 1 to Phase M gives higher priority to coding over a smaller number of flows. On the other hand, we also want to quickly move to higher phases so that we can code over a large number of flows and thus capitalize the largest amount of coding benefits. To that end, we need a new component:

the *Phase Completion Status Indicators (PCSIs)*. As already mentioned, when the AP receives a feedback packet from a client, the AP will update the overhearing status of Q_{out}. A separate process then takes the updated Q_{out} as input to compute the PCSIs. The PCSIs are then used by the phase-based operation to decide when to switch from Phase k to Phase $(k+1)$, or to decide whether we have finished the last Phase M and are ready to move to the next inter-flow batch.

The clients (Rx) store all overheard packets. Throughout the transmission, each client periodically sends back cumulative ACKs, similar to [3,19], that tell the AP which coded and uncoded packets it has received in the past. Since each packet is associated with an inter-flow coding vector, one can view that the periodic feedback tells the AP which coding vectors have been received by which client.

Unlike the MU-ARQ and ER, MU-FEC uses *batch decoding* to extract the benefits of network coding in a similar way as the MORE protocol and any other FEC-based protocols. More explicitly, each client collects all the packets it has heard and performs decoding at the end of the current inter-flow batch.

4.2. Phase-based operation

In the beginning of each inter-flow batch, we add N_{total} *elementary basis vectors* to Q_{out}. That is, if we concatenate the elementary row vectors vertically, we have an $N_{\text{total}} \times N_{\text{total}}$ identity matrix after initialization. For an elementary row vector corresponding to the i-th client, its i-th bit of the creation bit map is set to one and all other creation bits are set to zero. The creation map thus indicates that this elementary coding vector contains the information from flow j. Since no packet has been transmitted yet (thus no overhearing), all the overhearing bit maps are set to all-zero.

We use $[M]$ to denote the integer set $\{1, 2, \ldots, M\}$. The phase-based operation maintains $(2^M - 1)$ floating-point variables a_S indexed by S. Each subscript S is a non-empty subset of $[M]$. For example, when $M = 2$, we have $(2^M - 1) = 3$ different non-empty subsets $\{1\}$, $\{2\}$, and $\{1, 2\}$. Therefore, we have three floating-point variables $a_{\{1\}}$, $a_{\{2\}}$, and $a_{\{1,2\}}$. We use $|S|$ to denote the number of elements in S. The phase-based operation also maintains a register K, which records the current phase index.

The number of a_S variables grows exponentially with respect to M and is a tradeoff between performance and computational complexity. The value of M depends on the modern microprocessor capabilities. As will be seen, the larger the M value, the higher the throughput. In our implementation, we only use $M = 5$ (using 31 variables). Our results show that mixing only 5 flows together in an efficient way already outperforms the state-of-the-art ER protocol, which mixes all flows together.

Initialization of the path-based operation: In the beginning of each inter-flow batch, let $a_S \leftarrow 0$ for all non-empty set $S \subseteq [M]$. Let $K \leftarrow 1$.

Normal operation in Phase K:

Step I—*Choosing the flows to be coded together*: Whenever there is a transmission opportunity and the AP is in Phase K, the AP will mix K flows together. More explicitly, for any given non-empty subset $S \subseteq [M]$ that has $K = |S|$ elements, the AP can mix together all flows $i \in S$ in Phase K. The first key question is thus how to choose a subset S of size K for the AP to code the corresponding flows together. In MU-FEC, the module "Phase completion status indicators" will output a non-negative integer d_S for each subset S. The larger the d_S is for a given S, the more important it is to code the flows in S. To perform tie-breaking between different S, every time we can transmit a packet, we use the following subroutine to choose S:

1: Consider all non-empty subset $S \subseteq [M]$ satisfying both (i) the size is K, and (ii) the corresponding d_S generated from PCSIs is non-zero.
2: Among those S, choose the S^* with the largest a_S value.
3: Let $a_{S^*} \leftarrow a_{S^*} - \frac{1}{d_{S^*}}$.

Intuition: Using the auxiliary a_S variables, the frequency of selecting a subset S will be proportional to the corresponding importance indicator d_S.

Step II—*Generate a new coding vector*: Once the S^* is determined, use the following subroutine to generate a new inter-flow coding vector $\mathbf{v}$, which is a row vector of dimension N_{total}. We need the following definition for the subroutine.

Definition 1. An inter-flow coding vector $\mathbf{v}$ in Q_{out} is *compatible* to a non-empty subset $S \subseteq [M]$ if both the following two conditions are satisfied: (i) In the creation bit map of $\mathbf{v}$, all the bits outside S are zero. (ii) For all $i \in S$, either the i-th bit of the creation map is 1, or the i-th bit of the overhearing map is 1.

In other words, $\mathbf{v}$ is compatible to S if the subset S "covers" the creation bit map; and jointly the creation plus the overhearing maps "cover" the subset S. Intuitively, a vector $\mathbf{v}$ being compatible to S means that for any $i \in S$, either receiver i has already overheard $\mathbf{v}$ in the past (the overhearing map), or $\mathbf{v}$ carries information that can benefit receiver i. Borrowing the wisdom of intra-flow coding, we randomly mix all such eligible vectors $\mathbf{v}$ together to avoid making the decision which packets to be mixed. In other words, the final output $\mathbf{v}_{\text{out}}$ is the random summation of all the inter-flow coding vectors compatible to S^*. We use this $\mathbf{v}_{\text{out}}$ as the inter-flow coding vector for the to-be-transmitted packet. Store the new $\mathbf{v}_{\text{out}}$ back into Q_{out} (see the flow chart Fig. 5). Set the overhearing map to 0. For all $i \in [M]$, set the i-th bit of the creation map to 1 or 0 depending on whether $i \in S^*$ or not. Append to $\mathbf{v}_{\text{out}}$ a unique 2-byte sequence number. Note that the creation bit map

of this new $\mathbf{v}_{\text{out}}$ will not change anymore. However, the overhearing bit map will be updated in the future according to the feedback from the receivers back to the AP.

Step III—*Construct the coded payload and send the coded packet*: Based on the new inter-flow coding vector $\mathbf{v}_{\text{out}} = (v_1, \ldots, v_{N_{\text{total}}})$, we can assemble the coded payload by summing up the native payloads (stored in the payload buffer) with the corresponding coefficients $v_1, \ldots, v_{N_{\text{total}}}$. The final outgoing coded packet contains the 2-byte sequence number, the new inter-flow coding vector, the coded payload, and the creation bit map. The inclusion of the creation bit map is to reduce the size of the packet header, in particular the part occupied by the inter-flow coding vector. More explicitly, one can prove that for any vector $\mathbf{v}_{\text{out}}$ in MU-FEC, the value of a coordinate is non-zero only if it corresponds to a flow-i packet such that the i-th bit of the creation map is one. Therefore, when sending a k-flow packet, $k < M$, we only include in the packet header the segments of the inter-flow coding vector corresponding to the chosen k flows. The clients use the creation bit map to reconstruct the whole inter-flow coding vector by filling the segments corresponding to the remaining $(M - k)$ flows with 0's.

4.3. Phase completion status indicators

An important component of MU-FEC is the PCSI computation, which decides whether the AP can stop the current Phase k and move to Phase $(k + 1)$. Note that the overhearing maps of the vectors in Q_{out} are updated only after receiving an ACK from the destinations. Therefore, we only need to run the following routine during the initialization of Q_{out} for each inter-flow batch (see Section 4.1), and when the AP receives a cumulative ACK from a client. The computation of the PCSIs is as follows:

There is one PCSI for each non-empty subset $S \subseteq [M]$, which is denoted by d_S. For any given S, the computation of d_S is carried out as follows.

1: $d_S \leftarrow 0$.
2: **for** all $i \in S$ **do**
3: If $|S| < M$, consider all vectors in Q_{out} that satisfy at least one of the following two conditions: **Cond. 1:** it is received by d_i (by checking the i-th overhearing bits); **Cond. 2:** it is *compatible* to at least one $S' \subseteq [M]$ satisfying $|S'| > |S|$. If $|S| = M$, consider all vectors in Q_{out} that satisfy **Cond. 1**.
4: Project[2] all these vectors onto flow i.
5: Compute the rank of the projected vectors. Let r_1 denote the computed rank and store it for later use.
6: If $|S| < M$, consider all vectors in Q_{out} that satisfy at least one of the following three conditions: **Cond. 1**, **Cond. 2** as defined in Line 3, or **Cond. 3:** it is *compatible* to S. If $|S| = M$, consider all vectors in Q_{out} that satisfy at least one of the two conditions: **Cond. 1** and **Cond. 3**.
7: Project all these vectors onto flow i.
8: Compute the rank of the projected vectors. Let r_2 denote the computed rank.
9: $d_S \leftarrow d_S + (r_2 - r_1)$.
10: **end for**

Intuition: The PCSIs answer when we can switch to the next phase. Suppose we are in Phase k and try to mix k flows in S ($|S| = k$). If we know that what we can possibly achieve by mixing $|S|$ flows (in terms of conveying packets to their designated clients) can also be achieved by mixing $|S'|$ flows for some $S' \subseteq [M]$ satisfying $|S'| > |S|$, then it means that the conservative approach of mixing only $|S|$ flows can be superseded by a more aggressive approach of mixing $|S'|$ flows. In this case, we obviously want to start mixing $|S'|$ flows rather than wasting more time on only mixing $|S|$ flows. In essence, PCSIs quantify and compare "what can possibly be achieved" by mixing $|S|$ flows with that of mixing $|S'|$ flows. Such task is achieved by simple rank-based computation, which is based on a critical property of MU-FEC that will be introduced shortly after in Proposition 1 of Section 5.

More explicitly, the rank r_2 computed by PCSIs represents the benefits of "staying in the current Phase k" while the rank r_1 represents the benefits of "moving out of the current Phase k and focusing on the higher-phase S' flows". When the difference is zero, it means that it is time to move to the next phase. Note that d_S is the sum of $r_2 - r_1$ for all $i \in S$. The reason is that mixing S flows potentially benefits all $i \in S$. Therefore, we compute the benefit $r_2 - r_1$ for each individual flow i and sum them up for the computation of d_S.

Deciding whether to move to the next phase/batch: After the computation of d_S for all non-empty subsets $S \subseteq [M]$, the AP needs to decide whether to move to the next phase/batch or not. Suppose the current phase is k for some $1 \leq k \leq M - 1$. We check all $S \subseteq [M]$ with $|S| = k$. If all those S have $d_S = 0$, then it means staying in Phase k is redundant and we move to Phase $(k + 1)$. Otherwise, stay in Phase k. When $k = M$, there is no next phase to move to. We stay in Phase M until all destinations have sent feedback acknowledging that the decoding of the inter-flow batch is successful. Once all destinations have acknowledged the current inter-flow batch, we move to the new batch.

Complexity: The computation of $(2^M - 1)$ PCSIs every time the AP receives a cumulative ACK from a client is a computationally intense component of MU-FEC. (Note that the PCSI computation is on the vector only, not on the payload.)

[2] The projection of an inter-flow coding vector $\mathbf{v}$ to flow i is the subvector of $\mathbf{v}$ corresponding to the symbols of flow i.

However, the computation can be made much more efficient by noticing that there are a lot of repeated computations. For example, any vector in Q_{out} that satisfies **Conds. 1** and **2** in Line 3 for some i and S_1 must also satisfy **Conds. 1** and **2** for the same i and all other S_2 satisfying $|S_2| = |S_1|$. Therefore, the computation of r_1 for the pair (i, S_1) must be identical to that for the pair (i, S_2).

It is also important to notice that the complexity of the PCSI computation in MU-FEC increases exponentially with the number of flows (M), while the complexity of ER/MU-ARQ is exponential with the total number of packets. The dependence on the number of flows allows us to limit M to a small number (by grouping M flows together and only coding packets from those flows). This simple heuristic allows MU-FEC to achieve performance close to the optimal in practice. In contrast, the heuristic used in [3] performs much lower than the optimal coding strategy which can only be discovered if we consider all possible ways of coding all the packets.

4.4. Batch decoding at the client

Each receiver uses the same batch-based decoding algorithm. For simplification, we consider only receiver 1. Rx i for $i > 1$ can be obtained by symmetry.

During the batch transmission, Rx 1 stores all the inter-flow coding vectors in a queue $Q_{\text{rx\_vec}}$, and these vectors in $Q_{\text{rx\_vec}}$ form a $|Q_{\text{rx\_vec}}| \times N_{\text{total}}$ matrix. *We then deliberately rearrange the columns such that the columns corresponding to flow* 1 *are the last* N_1 *columns.* (If focusing on other clients, say Rx 3, we shuffle the columns of flow 3 to the last N_3 columns.) When adding newly received inter-flow coding vector $\mathbf{v}$ to $Q_{\text{rx\_vec}}$, we perform Gaussian elimination on both the coding vector and on the corresponding coded payload, which keeps the $Q_{\text{rx\_vec}}$ matrix upper triangular. Note that this is a two-step process. We first test the independence only for the coding vector. Only when the coding vector is linearly independent, then we process the payload, which reduces the computation complexity.

When decoding, consider the rows of which the only non-zero entries corresponding to flow 1. Since we shuffle the columns of flow 1 to the last N_1 columns and since $Q_{\text{rx\_vec}}$ is upper triangular, those rows must be the last few rows. When the number of those rows is N_1, we perform decoding. That is, we use the last N_1 rows and the corresponding coded payload to decode as if we are performing intra-flow decoding.

Intuition: The critical step is shuffling the flow-1 columns to the last N_1 columns. In this way, Gaussian elimination will automatically eliminate the "interference" caused by other flows so that we can decode the flow-1 packets as if decoding from intra-flow coded packets. (The only non-zero entries in the last few rows are all in the columns of flow 1.) As a result, we can decode the flow-1 packets from inter-flow coded packets as if decoding from intra-flow coded packets.

4.5. Client feedback reduction

We notice that with the PCSI computation, we move to Phase 2 only when each packet is overheard by at least one of the M receivers. Therefore, it takes at least $N_{\text{total}} \triangleq \sum_{i=1}^{M} N_i$ transmissions to finish Phase 1. *Since feedback is used only to decide when to move to the next phase, the clients do not need to transmit feedback back to the AP during the first* N_{total} *transmissions*. As a result, to reduce the amount of feedback traffic, at the beginning of each batch, clients do not send any feedback and use the arrival time and the sequence numbers of their received packets to estimate when the AP will finish the first N_{total} transmissions. After that, each client starts sending periodic feedback.

In contrast, feedback in ER is used to determine the retransmission time of each individual packet and hence, it has to be sent periodically for the whole duration of the protocol operation.

5. Correctness guarantee and performance analysis

The MU-FEC protocol is designed with rigorous mathematical foundation. In this section, we describe a key property of the MU-FEC protocol and provide the corresponding performance analysis.

For any receiver Rx i, we notice that by construction the corresponding $Q_{\text{rx\_vec}}$ is an upper triangular matrix. Let r_{proj} denote the rank of the submatrix of $Q_{\text{rx\_vec}}$ induced by the columns of flow i. Let r_{lower} denote the number of the last few rows for which all the non-zero entries are in columns of flow i. We have the following proposition.

Proposition 1. *For any time instant,* $r_{\text{proj}} = r_{\text{lower}}$.

Intuition: This proposition says that the "rank" of the projected space on flow i (r_{proj}) can be fully extracted by Gaussian elimination when focusing on the last r_{lower} rows for decoding (the decodable rows). Therefore, the inter-flow coding problem is now simplified to an intra-flow coding problem since the rank of the projected space on each flow (as if coding only among the flow of interest) is equal to the rank of information space after inter-flow decoding (based on Gaussian elimination that removes interference). Moreover, the AP can now simply look at the individual projected spaces of all the clients to decide whether to move to the next phase (r_{proj}) and MU-FEC automatically guarantees that the clients can receive coded packets (r_{lower}) by removing undesired packets of other clients.

It is worth emphasizing that the basic structure of MU-FEC is no different than the traditional FEC-based schemes. Namely, the AP keeps transmitting and uses the feedback to move to the next phase and/or next batch. The advanced coding

computation, such as the PCSIs, can be viewed as generalizing FECs to accommodate the ideas of inter-flow coding. Moreover, the computation at the clients is minimal. The client simply needs to perform Gaussian elimination and periodically sends a bit map indicating which packets it has received. All the advanced computation is performed within the AP.

The most intriguing feature of MU-FEC is that such simple feedback and FEC-based design *is also theoretically optimal.* That is, under the same theoretic setting as used in MU-ARQ [2], our practice-oriented *MU-FEC strictly outperforms the overly idealistic MU-ARQ principle and achieves throughput efficiency* (3)*, the best possible throughput over any idealistic/practical schemes one can envision*.

To rigorously state the above optimality claim, we use the same channel model as used in Eqs. (1) and (2) of Section 2. Unlike MU-ARQ [2], which assumes instant feedback, we assume periodic feedback is sent every $F > 1$ time slots using a separate channel, and prove that

Proposition 2. *The throughput efficiency of MU-FEC achieves the theoretic upper bound in* (3) *when* $\mathsf{GF}(2^b) \to \infty$ *and* $N \to \infty$*, where N is the common batch size for all flows and* $\mathsf{GF}(2^b)$ *is the finite field.*

Remark. The feedback period F affects how fast a client can send its reception status back to the AP. The intuition behind Proposition 2 is that for any fixed F, the inefficiency of MU-FEC caused by the AP not getting the latest reception status is negligible when an infinitely large batch size is used. Note that even for small batch sizes $N_i = 32$–48, the batch-based design of MU-FEC can greatly mitigate the impact of not receiving instant feedback. This also ensures that the impact of the loss of feedback packets is absorbed without too much throughput degradation.

The proofs of Propositions 1 and 2 are relegated to the Appendix.

6. Simulation study

In this section, we compare the performance of MU-FEC against that of ER and 802.11 using the Glomosim simulator [20]. Since a major contribution of MU-FEC is to realize the throughput gain of inter-flow coding, we still use the ER protocol for comparison even though ER does not guarantee 100% reliability. For performance comparison with respect to FEC-only protocols, since there are several different FEC protocols, ex: Reed–Solomon codes, fountain codes, intra-flow coding, we use the theoretic value $\eta_{\text{FEC.only}} \cdot (\text{erasure-free rate})$ as proxy, also see Section 2.1, which will be strictly larger than any practical FEC scheme with overhead.

6.1. Methodology

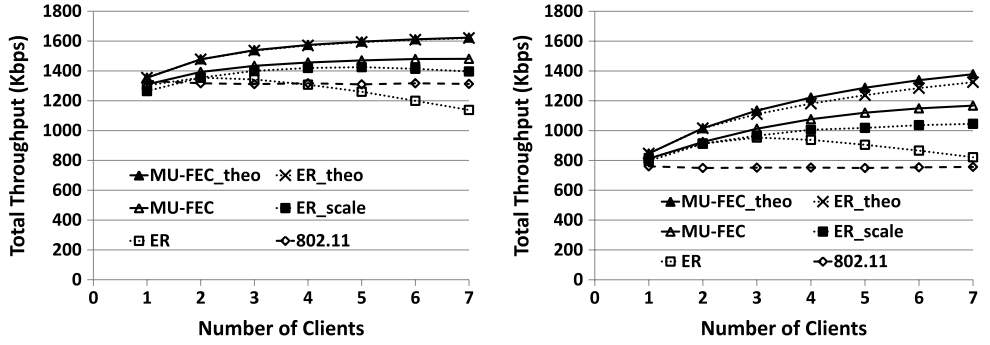
We follow the evaluation methodology in [3]. We place an AP in the center of the simulation area and up to 20 clients uniformly on a circle around the AP. To evaluate the performance of the protocols under different loss scenarios, the clients are placed close to the AP and we generate link loss rates in a controlled manner, by artificially dropping packets at each client following a Bernoulli model. We consider both homogeneous and heterogeneous loss rates. In the homogeneous case, the loss rates of all links are the same, varied between 10% and 90% in different simulations. In the heterogeneous case, the loss rate for each link is randomly selected between 0 and an upper bound varying from 10% to 90%.

In every simulation run, the AP transmits 1500-byte packets for 100 s. The results in the following sections are averages over 10 runs. Note that the standard deviations are very low in all cases except for the heterogeneous loss scenarios in Section 6.4. Hence, in the interest of clarity, we do not plot the standard deviation in the rest of the results.

We noticed the performance of ER heavily depends on three parameters: retransmission queue threshold, retransmission queue timeout, and period of cumulative ACKs. Further, the optimal set of values for the three parameters depends on the number of clients. [3] uses a threshold of 25 packets and a timeout of 250 ms and does not discuss the third parameter. Note also that [3] used a simplified simulator that did not model timing dynamics. Hence, it did not evaluate the actual ER implementation, but a simplified version of the protocol, where a sender sends a constant-size batch of packets at a time and then (after instant feedback from the clients) retransmits lost packets until all of them are received by the destined clients. With this simplified (but not practical) version, the authors were not able to study the interdependence of the three parameters of the actual implementation.

In ER's favor, we decided to tune the three parameters to maximize ER's performance. We skip the details of tuning due to space limitation. We found that using a base ACK period of 20 ms, and scaling all three parameters by K when there are K clients, gave the best performance among different configurations. We denote this version of ER as ER_scale. We repeated the same procedure to choose the ACK period for MU-FEC, the only parameter in our protocol, and we set the ACK period to $10 \cdot K$ ms for MU-FEC. Finally, we use a batch of 48 packets for each flow and a Galois Field $\mathsf{GF}(2^4)$ (half byte).

We used the *Aggregate Throughput* as our evaluation metric. The throughput per client is defined as the total number of unique packets (excluding duplicates) received for 802.11 and ER, or the total number of decoded packets for MU-FEC, multiplied by the packet payload size, and divided by the total time required to collect/decode those packets. The aggregate throughput is the sum of the throughputs of all clients.



(a) 20% loss. (b) 50% loss.

Fig. 6. Theory vs. practice: Throughput comparison for different numbers of clients under homogeneous losses.

6.2. Overall result

Fig. 6(a) and (b) plot the throughput of MU-FEC, ER, and 802.11 when the number of clients varies from 1 to 7, for a homogeneous loss rate of 20% and 50%, respectively. For ER, we plot two versions, the original one and the version where we scale all the parameters with the number of clients, denoted as ER_scale. For MU-FEC, we ignore for now the computational complexity issue and group all flows together (i.e., $M = 7$).

We also include in these two graphs the theoretically predicted throughput for MU-FEC (MU-FEC_theo) and for ER (ER_theo or MU-ARQ). To plot these curves, we first ran a simulation with a single client using 802.11 under a 0% loss scenario and measured the throughput (1693 kbps). We consider this value as the theoretically optimal throughput when there is no packet loss.[3] We then calculated the theoretical throughput for 2–7 clients, by multiplying this optimal throughput value with the throughput efficiency, given by Eqs. (2) and (3), for ER_theo and MU-FEC_theo, respectively.

We notice that the theoretic FEC-only throughputs for 20% and 50% loss rate are 1354 kbps and 847 kbps, respectively, which are significantly lower than our practical MU-FEC implementation even when mixing only $M \geq 3$ clients. Therefore, for the following, we focus on throughput comparison to the unreliable ER scheme. We make the following observations:

First, the theoretically optimal throughput of ER is *strictly lower* compared to the theoretically optimal throughput of MU-FEC. While the difference between the two curves is negligible for low loss rates (Fig. 6(a)), it can be clearly observed under high loss rates (Fig. 6(b)). In the same figure, we also observe that the gap between the two curves increases with the number of clients.

Second, the systematic design of MU-FEC allows the protocol to achieve throughput *close to the theoretically optimal* even under high loss rates. Comparing the theoretical and practical throughput for each protocol, we observe that the gap between the two values for MU-FEC is smaller than for ER. Specifically, the gap for MU-FEC is at most 9% of the theoretical value under 20% loss and at most 16% under 50% loss. For example, when mixing 7 flows in a noisy environment of 50% loss rate, MU-FEC, with all the necessary overhead for coding and coordinating 7 flows simultaneously, achieves at least 84% of the throughput of any possible scheme (including the theoretic schemes that do not incur any overhead). In contrast, the gap for ER_scale can be as high as 14% of the theoretical value under 20% loss and as high as 21% under 50% loss. The gaps for the original version of the ER protocol are even larger, up to 30% under 20% loss and up to 38% under 50% loss.

Third, comparing the practical throughput values for each protocol, we observe that MU-FEC clearly outperforms ER, especially under high loss rates. Under a 50% loss, MU-FEC outperforms ER_scale by up to 12% and the original ER by up to 42%.

Fourth, the two graphs show the *robustness* of the systematic design MU-FEC over the ad-hoc design of ER. In both figures, the throughput of MU-FEC increases with the number of clients, following the same trend as the theoretical curves. In contrast, the throughput of the scaled version of ER increases at a much slower rate under 50% loss and starts dropping for more than 5 clients under 20% loss. Note also, that in order for ER_scale to achieve performance close to that of MU-FEC, we had to finely tune 3 different parameters, which requires a significant amount of effort in practice. Without tuning, the protocol cannot scale beyond 3–4 clients.

In the remaining of Section 6, we compare the scalability of the two protocols with the number of clients and with the loss rate. Since the computational complexity of our protocol increases exponentially with the number of flows M, in the remaining simulations we always use $M = 5$ (i.e., we divide the flows into groups of 5 and we only mix together packets from flows in the same group).

[3] This value 1693-kbps is also used as the theoretic (erasure-freerate) when computing the FEC-only throughput $\eta_{\text{FEC.only}} \cdot (\text{erasure-free rate}) = p \cdot 1693$ kbps where p is the packet delivery probability.

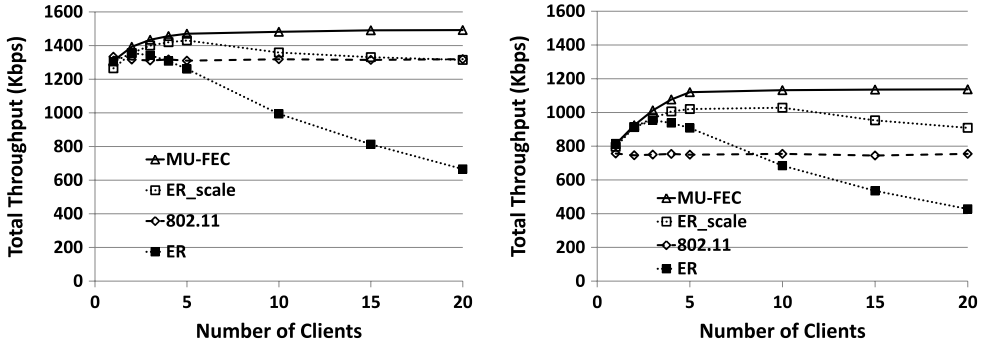


Fig. 7. Throughput comparison for different numbers of clients under homogeneous losses.

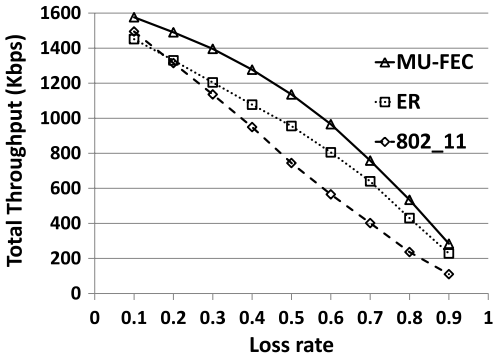


Fig. 8. Throughput comparison with 15 clients under varying homogeneous loss rates.

6.3. Varying the number of clients

Performance comparison. Fig. 7(a)–(b) plot the throughput of MU-FEC, ER, and 802.11 with 1–20 clients for a loss rate of 20% and 50%. We make the following observations:

First, we observe that the performance of ER drops dramatically with the number of clients. Even 802.11 outperforms ER with more than 4 clients under a 20% loss rate and with more than 10 clients under a 50% loss rate. Note again the complete ER protocol was evaluated in [3] with only up to 6 clients.

Second, ER_scale substantially outperforms ER and always outperforms 802.11 (but not much for the case of 20 clients). Still, its performance degrades with the number of clients.

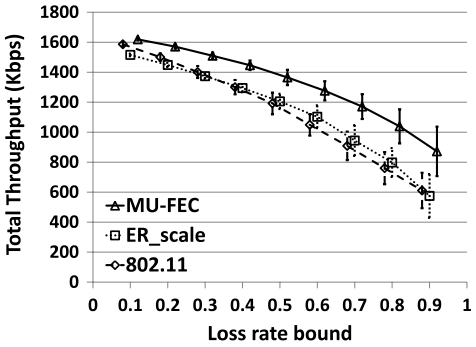
Third, MU-FEC outperforms ER_scale. The throughput gain of MU-FEC over ER_scale is as high as 14% under a 20% loss rate and as high as 25% under a 50% loss rate, with 20 clients. Compared to original ER, MU-FEC's throughput gain is much higher, up to 126% and 166% under 20% and 50% loss rates, respectively. Finally, compared to 802.11, MU-FEC's throughput gain is as high as 13% under a 20% loss rate and as high as 51% under a 50% loss rate. Moreover, the protocol scales well with the number of clients in spite of only allowing to encode packets from groups of 5 flows.

To understand where the gains of MU-FEC come from, we looked at the breakdown of coded and uncoded transmissions for the three protocols. Details are omitted due to lack of space. In summary, with ER and ER_scale, the number of retransmitted packets increases with the number of clients, and the percentage of coded transmissions decreases. In contrast, the fraction of packets in each phase with MU-FEC remains unchanged with the number of clients, which shows that MU-FEC scales well with the number of clients.

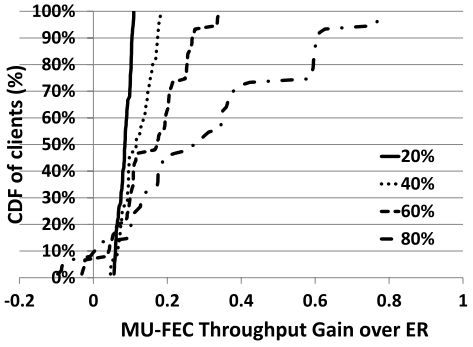
6.4. Varying the loss rate

We now evaluate the performance for varying loss rates. Since ER_scale performs much better than the original ER without scaling, we will only use this version in the remaining of this section.

Homogeneous losses. Fig. 8 plots the aggregate throughput with MU-FEC, ER_scale, and 802.11, with 15 clients, when the loss rate varies from 10% to 90%. We observe that MU-FEC outperforms ER_scale and 802.11 for all loss rates and the improvement is higher with higher loss rates, which also agrees with the theoretical results. The throughput gain of MU-FEC over ER_scale varies from 9% (with 10% loss rate) up to 25% (with 90% loss rate) and the gain over 802.11 varies from 5% up to 161%.



(a) Throughput comparison under varying loss rate bounds.



(b) CDFs of the gain of MU-FEC over ER for 4 different loss rate bounds.

Fig. 9. Performance comparison with 15 clients under heterogeneous losses. For clarity, the points of the MU-FEC and 802.11 curves in Fig. 9(a) are offseted horizontally.

Heterogeneous losses. We now consider heterogeneous loss rates. This scenario can arise when different clients have different loss rates either because they are located in different distances from the AP or due to multipath fading or because they experience different numbers of collisions (when other clients or neighboring APs act as hidden terminals, e.g., in urban environments).

Fig. 9(a) plots the aggregate throughput with MU-FEC, ER_scale, and 802.11, with 15 clients, when the loss rate bound varies from 10% to 90%. Again, MU-FEC outperforms ER_scale and 802.11 for all loss rate bounds and the improvement is higher with higher loss rate bounds, i.e., with higher heterogeneity. The throughput gain of MU-FEC over ER_scale varies from 7% (with 10% loss rate) up to 52% (with 90% loss rate) and the gain over 802.11 varies from 2% up to 43%. Note that in the heterogeneous case, ER_scale barely outperforms 802.11, while the gain of MU-FEC over 802.11 is robust.

Fig. 9(b) plots the cumulative distribution function (CDF) of the *per-client* throughput gain of MU-FEC over ER_scale for all 10 scenarios (i.e., for $10 \times 15 = 150$ clients) and for 4 different loss rate bounds. The throughput gain of MU-FEC over ER_scale for a client c is defined as $\frac{T^c_{\text{MU-FEC}} - T^c_{\text{ER}}}{T^c_{\text{ER}}}$, where $T^c_{\text{MU-FEC}}$, T^c_{ER} is the throughput of client c with MU-FEC and ER_scale, respectively. We observe that MU-FEC offers *higher* throughput than ER_scale to *all* clients under low (20% loss bound) and moderate (40% loss bound) loss rates and heterogeneity, and to more than 90% of the clients under high loss rates and heterogeneity. This shows again the robustness of the protocol. The median throughput improvement is 9%, 12%, 17%, and 29%, under a 20%, 40%, 60%, and 80% loss rate bound, respectively. The 90-th percentile is much higher, especially under high heterogeneity and loss rates.

7. Conclusion

In this paper, we developed a new network coding based retransmission protocol for WLANs, called MU-FEC. The design of MU-FEC is accompanied by a theoretical underpinning yet enables practical implementation on off-the-shelf 802.11 hardware. We showed that, when the batch size N is sufficiently large, MU-FEC attains the *provably optimal* inter-flow coding gain that is *strictly better* than that of the MU-ARQ principle, especially when the number of clients is from moderate to large. Even though the complexity of MU-FEC is exponential with the number of flows M, the use of intra-flow coding allows us to greatly simplify the computation (by grouping M flows together and only coding packets from those flows) and at the same time to achieve performance close to the optimal in practice. Our performance evaluation, through extensive simulations shows that the performance of MU-FEC is very robust and consistently outperforms ER in a variety of scenarios.

For future work we plan to incorporate bitrate adaptation into MU-FEC, and devise online batch size selection algorithms to allow the smooth operation of the protocol under higher layer protocols that set delay requirements (e.g., streaming protocols or TCP).

Appendix. Proofs

We first introduce the following lemmas, which will be used to prove Proposition 1.

Lemma 1. *For any time instant of the AP, suppose the AP is in Phase l and a packet is generated according to a set S satisfying* $|S| = l$. *Consider one client* $i_0 \in S$. *If the* i_0*-th client receives that packet, then after Client-*i_0 *performs Gaussian elimination, the new vector will have zero elements for all columns corresponding to flow-j, for all* $j \neq i_0$.

Intuition: This proposition says that for any client $i_0 \in S$, all the "interference" from flow j, $j \neq i_0$ can be canceled. This is similar to the "immediate decodability concept" of COPE. That is, for the COPE protocol, upon the reception of one packet mixing all flows in S satisfying $i_0 \in S$, Client i_0 can cancel all the interference and decode one more *native packet*. The

difference for MU-FEC is that now d_{i_0} can cancel all the interference and obtain one more *linearly independent intra-flow coded packet*.

Proof. We prove this by double induction on the size of $|S|$ and on the k-th packet in Phase $|S|$.

When $|S| = 1$ and $k = 1$, there is only one element i_0 in S. The vector pool used to generate this packet must only consist of the elementary basis vectors for flow i_0 that are put in the queues during initialization. As a result, if that packet is received by Client-i_0, then after Gaussian elimination the new vector will have zero elements for all columns corresponding to flow-j, for all $j \neq i_0$. The lemma is satisfied.

Based upon the basic case $|S| = 1$ and $k = 1$, we will prove the following two cases. *Case* 1: for any integer $h \geq 1$, $l > 1$, if Lemma 1 holds for $|S| = h$ and $k \leq l - 1$ and for $|S| \leq h - 1$ and all k, then Lemma 1 holds for $|S| = h$ and $k = l$ as well. *Case* 2: for any $h > 1$, if Lemma 1 holds for $|S| \leq h - 1$ and any k, then Lemma 1 holds for $|S| = h$ and $k = 1$ as well.

Case 1: For $|S| = h$ and $k = l$, since the compatibility condition requires that S covers the creation bit map of the vectors of interest and since we only move between phases in a monotonic fashion, the vector pool used to generate this packet must only consist of (i) old Phase $|S|$ packets generated using the same S, or (ii) packets generated from some $S_1 \subset S$ and $i_0 \in S_1$, or (iii) packets generated from some $S_1 \subset S$ and $i_0 \notin S_1$. Among the vector pool, those packets of type-(i), by induction, will have zero elements for all columns corresponding to flow-j after Gaussian elimination, for all $j \neq i_0$. Those packets of type-(ii), by induction, will have zero elements for all columns corresponding to flow-j after Gaussian elimination, for all $j \neq i_0$. Those packets of type-(iii) will be transparent to Client-i_0, since $i_0 \notin S_1$ implies that such packet have been overheard by Client-i_0 (the compatibility condition implies that the overhearing bit map plus the creation bit map cover S_1). Since the newly generated packet is a linear combination of packets of one of the three types and all of them have zero elements after Gaussian elimination for columns of flow $j \neq i_0$, the same statement holds for the generated packet as well. Case 1 is thus proved.

Case 2: For $|S| = h$ and $k = 1$, the vector pool used to generate this packet must only consist of (ii) packets generated from some $S_1 \subset S$ and $i_0 \in S_1$, or (iii) packets generated from some $S_1 \subset S$ and $i_0 \notin S_1$. Those packets of type-(ii), by induction, will have zero elements for all columns corresponding to flow-j after Gaussian elimination, for all $j \neq i_0$. Those packets of type-(iii) will be transparent to Client-i_0, since $i_0 \notin S_1$ implies that such packet have been overheard by Client-i_0. By the same reason as in Case 1, we have proved Case 2. □

Lemma 2. *For any time instant of the AP, suppose the AP is in Phase l and a packet is generated according to a set S satisfying $|S| = l$. Consider one client $i_0 \notin S$. Then the inter-flow coding vector of the newly generated packet will have zero elements for all columns corresponding to flow-i_0.*

Proof. We prove this by double induction on the size of $|S|$ and on the k-th packet in Phase $|S|$.

When $|S| = 1$ and $k = 1$, there is only one element j in S, $j \neq i_0$. The vector pool used to generate this packet must only consist of the elementary basis vectors for flow j that are put in the queues during initialization. As a result, the new vector will have zero elements for all columns corresponding to flow-i_0.

Based upon the basic case such that $|S| = 1$ and $k = 1$, we will prove the following two cases. *Case* 1: for any integer $h \geq 1$, $l > 1$, if Lemma 2 holds for $|S| = h$ and $k \leq l - 1$ and for $|S| \leq h - 1$ and all k, then Lemma 2 holds for $|S| = h$ and $k = l$ as well. *Case* 2: for any $h > 1$, if Lemma 2 holds for $|S| \leq h - 1$ and any k, then Lemma 2 holds for $|S| = h$ and $k = 1$ as well.

Case 1: For $|S| = h$ and $k = l$, since the compatibility condition requires that S covers the creation bit map of the vectors of interest and since we only move between phases in a monotonic fashion, the vector pool used to generate this packet must only consist of (i) old Phase $|S|$ packets generated using the same S, or (ii) packets generated from some $S_1 \subset S$ and $i_0 \notin S_1$. Among the vector pool, those packets of type-(i), by induction, will have zero elements for all columns corresponding to flow-i_0. Those packets of type-(ii), by induction, will have zero elements for all columns corresponding to flow-i_0. Since the newly generated packet is a linear combination of the above two types of packets and both of them will have zero elements for the columns of flow i_0, the same statement holds for the generated packet as well. Case 1 is thus proved.

Case 2: For $|S| = h$ and $k = 1$, the vector pool used to generate this packet must only consist of (ii) the packets generated from some $S_1 \subset S$ and $i_0 \notin S_1$. Those type-(ii) packets, by induction, will have zero elements for all columns corresponding to flow-i_0. By the same reason as in Case 1, we have proved Case 2. □

In the following, we prove Proposition 1.

Proof. Suppose the joint decoding matrix $Q_{\text{rx\_vec}}$ is of size $k \times \sum_{i=1}^{M} N_i$. We will prove Proposition 1 by induction of the value of k. When $Q_{\text{rx\_vec}}$ is of size $0 \times \sum N_i$, Proposition 1 is obviously true.

Suppose it is true for the size of $Q_{\text{rx\_vec}}$ being $k_0 \times \sum N_i$ for some $k_0 \geq 0$. When the next packet arrives, we have two cases. *Case* 1: $i_0 \in S$. Then by Lemma 1, after Gaussian elimination, we have that i_0's columns have a vector $\mathbf{v}$ and all other places are zero. *Case* 1.1: $\mathbf{v} = 0$. In this case, the rank of the lower right submatrix does not increase. We need to show that the projection of the original packets (before Gaussian elimination) also does not increase. This is true since $\mathbf{v} = 0$ implies that the new inter-flow vector is in the space of the already received inter-flow coding vectors in $Q_{\text{rx\_vec}}$. Therefore the projected vector is also in the space of the projection of the already received inter-flow vectors. Therefore the rank of the projected space also does not increase. *Case* 1.2: $\mathbf{v} \neq 0$. In this case, the rank of the lower right submatrix increases. Since the rank

of the projected space is no smaller than the rank of the lower right submatrix, the rank of the projected space must also increase by one. Therefore, for both subcases, we have $r_{\text{proj}} = r_{\text{lower}}$.

Case 2: $i_0 \notin S$. Then by Lemma 2, even before Gaussian elimination, the inter-flow coding vector has zero elements in the columns of flow i_0. Therefore, the rank of the projected space does not increase. As a result, the rank of the lower-right submatrix also does not increase, as the rank of the projection is no less than the rank of the lower right submatrix. The above proves that $r_{\text{proj}} = r_{\text{lower}}$. The proof is complete. □

For the following, we provide the proof of Proposition 2.

Proof. The proof of Proposition 2 consists of two parts: *Part* 1: The proof that the best possible throughput is bounded by (3), and *Part* 2: MU-FEC indeed achieves the throughput depicted by (3).

Proof of Part 1. One can prove Part 1 by the degraded channel argument first used in [11] for the broadcast additive white Gaussian channels and later generalized [5] for broadcast packet erasure channels (PECs) for the special case of 2 clients. We first fix a permutation π for integers 1 to M. Consider a new broadcast channel with $(M-1)$ artificially created information pipes sending information from client $C_{\pi(i)}$ to $C_{\pi(i+1)}$ for $i = 1$ to $(M-1)$. With the auxiliary pipes, any client $C_{\pi(i)}$, not only observes its corresponding output but also has all the information of its "upstream receivers" $C_{\pi(l)}$ for all $l = 1, \ldots, i-1$. Since we only create new pipes, any achievable rates of the original 1-to-M broadcast PEC must also be achievable in the new 1-to-M broadcast PEC. The capacity of the new 1-to-M broadcast PEC with feedback is thus an outer bound on the capacity of the original 1-to-M broadcast PEC with feedback.

On the other hand, the new 1-to-M broadcast PEC is a *physically degraded* broadcast channel with the new success probability of $C_{\pi(k)}$ being $1-(1-p)^k$ instead of p. The reason is that any one of clients $C_{\pi(1)}$ to $C_{\pi(k)}$ in the original channel receives the output will imply that the $C_{\pi(k)}$ in the new PEC receives it due to the added information pipes. In [12] proves that feedback does not increase the capacity of any physically degraded 1–2 broadcast channel. By extending the derivation steps in [12], one can easily prove that feedback does not increase the capacity of any physically degraded 1-to-M broadcast channel for arbitrary M values. Therefore the capacity of the new 1-to-M broadcast PEC with feedback is identical to the capacity of the new 1-to-M broadcast PEC without feedback. Since the capacity of the new 1-to-M PEC with feedback can be described by

$$\sum_{k=1}^{M} \frac{R_{\pi(k)}}{1-(1-p)^k} \leq 1 \tag{4}$$

where $R_{\pi(k)}$ is the throughput for client $C_{\pi(k)}$, Eq. (4) must be an outer bound of the capacity of the original 1-to-M PEC with feedback. Note that the above procedure can be repeated for any permutation π. The overall capacity outer bound is thus described by the $M!$ inequalities over the permutations. By maximizing the sum capacity $\sum_{i=1}^{M} R_i$ over the above polytope, we thus have

$$\eta = \sum_{i=1}^{M} R_i \leq \frac{M}{\sum_{k=1}^{M} \frac{1}{1-(1-p)^k}}. \tag{5}$$

See [5,4] for detailed steps and the corresponding references. □

Proof of Part 2. We first note that for a sufficiently large batch size, the impact of sending non-instant, periodic feedback is fully absorbed and becomes negligible from the throughput's perspective. Therefore, to analyze the performance of MU-FEC, we can assume that we do have instant feedback $F = 1$.

Secondly, since we also assume that all M flows have the same batch size $N_i = N$, the assumption that the channels are symmetric implies that for any two sets S_1 and S_2 satisfying $|S_1| = |S_2|$, the corresponding PCSIs d_{S_1} and d_{S_2} will become 0 *almost simultaneously*. Therefore, we can use $f(|S|)$ to denote the number of time slots it takes to mix all flows in S before the PCSI outputs $d_S = 0$. For example, $f(1)$ is the number of time slots to finish transmitting flow-i_0 packets in Phase 1 with the chosen flow set being $S = \{i_0\}$, before the PCSI outputs $d_S = 0$. We then note that anytime one of the M clients receives a flow-i_0 packet in Phase 1 will reduce the PCSI $d_{\{i_0\}}$ by one, and before the beginning of Phase 1 we have $d_{\{i_0\}} = N$. Therefore it must take $f(1) \approx \frac{N}{1-(1-p)^M}$ time slots to finish transmission of $S = \{i_0\}$ in Phase 1. Since there are M different choices of S in Phase 1, overall it takes $\binom{M}{1} f(1)$ time slots to finish Phase 1.

Similarly, it takes $\binom{M}{k} f(k)$ time slots to finish Phase k. Therefore, to finish all M phases, we need $\sum_{k=1}^{M} \binom{M}{k} f(k)$ time slots. The throughput efficiency of MU-FEC is thus

$$\frac{MN}{\sum_{k=1}^{M} \binom{M}{k} f(k)}. \tag{6}$$

For the following, we show that the expression of $f(k)$ can be computed by the following iterative formula:

$$\forall k \in \{1, \ldots, M\},$$
$$f(k) + \sum_{j=1}^{k-1} \binom{k-1}{j-1} f(j) = \frac{N}{1-(1-p)^{M-(k-1)}}. \tag{7}$$

To derive the above equality, suppose that we are focusing on $S = \{1, 2, 3, \ldots, k\}$ and sending flow-S packets. The right-hand side of (7) thus corresponds to the total number of time slots it takes for a client-1 packet to be heard by at least one of Clients $1, k+1, k+2, \ldots, M$. Note that all those $f(k)$ time slots for sending flow-$S = \{1, 2, 3, \ldots, k\}$ must be counted toward part of it as it mixes a session-1 packets that has been heard only by Clients $2, 3, \ldots, k$. On the other hand, there are also other time slots that will be counted toward the total number $\frac{N}{1-(1-p)^{M-(k-1)}}$ of time slots. For example, when we send a session-1 packet that has been heard by none of $\{2, 3, \ldots, k\}$ (a Phase-1 operation), totally there are $f(1)$ such time slots that will be counted toward the right-hand side of (7). Similarly, when we send a session-1 packet that has been heard by only one of $\{2, 3, \ldots, k\}$, totally there are $f(2)$ such time slots that will be counted toward the right-hand side of (7). But there are $\binom{k-1}{1}$ ways of being heard by only one of $\{2, 3, \ldots, k\}$. Therefore, the total contribution becomes $\binom{k-1}{1} f(2)$. As a result, the left-hand side of (7) summarizes how many time slots are counted before all client-1 packets are heard by at least one of Clients $1, k+1, k+2, \ldots, M$.

To prove that (3) and (6) are identical, the remaining task is to prove that the total number of time slots used in our scheme is

$$\sum_{k=1}^{M} \binom{M}{k} f(k) = \sum_{k=1}^{M} \frac{N}{1-(1-p)^k}. \tag{8}$$

Eq. (8) can be derived from (7) by simple arithmetic computation. (8) then ensures the equivalence between (3) and (6). The proof of Part 2 is complete. □

References

[1] M. Ghaderi, D. Towsley, J. Kurose, Reliability gain of network coding in lossy wireless networks, in: Proc. of IEEE INFOCOM, 2008.
[2] P. Larsson, N. Johansson, Multi-user ARQ, in: Proc. of IEEE VTC-Spring, 2006.
[3] E. Rozner, A.P. Iyer, Y. Mehta, L. Qiu, M. Jafry, ER: efficient retransmission scheme for wireless lans, in: Proc. of CoNEXT, 2007.
[4] C.-C. Wang, Capacity of 1-to-K broadcast packet erasure channels with channel output feedback, in: Proc. 48th Annual Allerton Conf. on Comm., Contr., and Computing, 2010.
[5] L. Georgiadis, L. Tassiulas, Broadcast erasure channel with feedback—capacity and algorithms, in: Proc. of NetCod, 2009.
[6] M. Gatzianas, L. Georgiadis, L. Tassiulas, Multiuser broadcast erasure channel with feedback—capacity and algorithms, in: Proc. of NetCoop, 2010.
[7] H. Seferoglu, A.M. nad, K.K. Ramakrishnan, I2NC: intra- and inter-session network coding for unicast flows in wireless networks, in: Proc. of IEEE INFOCOM, 2011.
[8] X. Zhu, H. Yue, Y. Fang, Y. Wang, A batched network coding scheme for wireless networks, ACM Wirel. Netw. 15 (2009).
[9] C. Qin, Y. Xian, C. Gray, N. Santhapuri, S. Nelakuditi, I^{2}MIX: integration of intra-flow and inter-flow wireless network coding, in: Proc. of IEEE International Workshop on Wireless Network Coding WiNC, 2008.
[10] F. Xue, X. Yang, Network coding and packet-erasure broadcast channel, in: Proc. of IEEE SECON, 2008.
[11] L. Ozarow, S. Leung-Yan-Cheong, An achievable region and outer bound for the Gaussian broadcast channel with feedback, IEEE Trans. Inform. Theory 30 (4) (1984).
[12] A. El Gamal, The feedback capacity of degraded broadcast channels, IEEE Trans. Inform. Theory 25 (2) (1978).
[13] Z. Bar-Yossev, Y. Birk, T. Jayram, T. Kol, Index coding with side information, in: Proc. of IEEE FOCS, 2006.
[14] F. chun Kuo, K. Tan, X. Li, J. Zhang, X. Fu, XOR rescue: exploiting network coding in lossy wireless networks, in: Proc. of IEEE SECON, 2009.
[15] D. Nguyen, T. Nguyen, B. Bose, Wireless broadcasting using network coding, in: Proc. of NetCod Workshop, 2007.
[16] A. Eryilmaz, A. Ozdaglar, M. Medard, On delay performance gains from network coding, in: Proc. of IEEE CISS, 2006.
[17] S. Biswas, R. Morris, ExOR: opportunistic multi-hop routing for wireless networks, in: Proc of ACM SIGCOMM, 2005.
[18] S. Chachulski, M. Jennings, S. Katti, D. Katabi, Trading structure for randomness in wireless opportunistic routing, in: ACM SIGCOMM, 2007.
[19] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, J. Crowcroft, Xors in the air: practical wireless network coding, in: Proc. of ACM SIGCOMM, August 2006.
[20] X. Zeng, R. Bagrodia, M. Gerla, Glomosim: a library for parallel simulation of large-scale wireless networks, in: Proc. of PADS Workshop, May 1998.

Chih-Chun Wang joined the School of Electrical and Computer Engineering at Purdue University in 2006 as an Assistant Professor. He received the B.E. degree from the National Taiwan University in 1999, and the M.S. and Ph.D. degrees in E.E. from Princeton University in 2002 and 2005, respectively. His current research interests are in the graph-theoretic and algorithmic analysis of iterative decoding and of network coding. His other research interests fall in the general areas of optimal control, information theory, detection theory, coding theory, iterative decoding algorithms, and network coding. He received the NSF CAREER Award in 2009.

Dimitrios Koutsonikolas received the Ph.D. degree in Electrical and Computer Engineering from Purdue University, West Lafayette, IN, in 2010. He worked as a Post-Doctoral Research Associate at Purdue University from September to December 2010. He is currently an Assistant Professor of Computer Science and Engineering at the University at Buffalo, State University of New York. His research interests are broadly in experimental wireless networking and mobile computing. He is a member of IEEE, ACM, and USENIX.

Y. Charlie Hu is a Professor of Electrical and Computer Engineering and Computer Science (by courtesy) at Purdue University. He received his Ph.D. degree in Computer Science from the Harvard University in 1997. From 1997 to 2001, he was a research scientist at Rice University. His research interests include operating systems, distributed systems, computer networking, and wireless networking. He has published over 120 papers in these areas. Dr. Hu received the NSF CAREER Award in 2003. He is a senior member of IEEE and a distinguished scientist of ACM.

Ness Shroff currently holds the Ohio Eminent Scholar Chaired Professorship in Networking and Communications, in the departments of ECE and CSE at the Ohio State University. He is also a Guest Chaired Professor of Wireless Networking in the department of Electronic Engineering at Tsinghua University, China. Previously, he was a Professor of ECE at Purdue University and the Director of the Center for Wireless Systems and Applications (CWSA), a university-wide center on wireless systems and applications. His research interests span the areas of wireless and wireline communication networks, where he investigates fundamental problems in the design, performance, pricing, and security of these networks. Dr. Shroff has received numerous awards for his networking research, including the NSF CAREER award, the best paper awards for IEEE INFOCOM 06 and IEEE INFOCOM 08.