Computer Networks 179 (2020) 107361

Contents lists available at ScienceDirect

Computer Networks

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

Rateless802.11: Extending WiFi applicability in extremely poor channels☆

Tao Huang[a], Bin Tang[a,*], Baoliu Ye[a,*], Zhihao Qu[b], Sanglu Lu[a]

[a] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[b] College of Computer and Information, Hohai University, Nanjing, China

ARTICLE INFO

Keywords:
802.11
Convolutional codes
Rateless codes
Scrambler seed
Belief-propagation

ABSTRACT

Due to its limited error correction capability, 802.11 can hardly work in scenarios where the channel fading or interference is very severe, such as industrial internet of things. In this paper, we propose Rateless802.11, a cross-layer scheme which can work as a middleware over common commodity 802.11 devices, that extends the applicability of 802.11 in extremely poor channels. Rateless802.11 concatenates LT codes with 802.11 convolutional codes in the transmitter side to introduce proper redundancies in an incremental manner, so that the uncorrupted bits of MAC protocol data units (MPDU) can be adequately exploited, enhancing the error correction capability significantly. In order to ensure its compatibility with commodity devices, Rateless802.11 adopts a well-designed process flow of received packets at the receiver side, within which some novel decoding procedures are employed to achieve a better decoding performance. Specifically, since the random seed used in the scrambling process could be polluted during the descrambling process in commodity 802.11 device due to the poor channel condition, we propose a method to protect and recover the seed based on the linearity of scrambling sequences. Moreover, we propose a belief-propagation based integrated decoder, which decodes convolutional codes and LT codes together in a joint manner, leading to much less information-loss and decoding delay than serial decoding. Both numerical and real-trace driven simulations show that Rateless802.11 improves the throughput of 802.11 in extremely poor channels by several to dozens of times compared with state-of-the-art solutions.

1. Introduction

In common IEEE 802.11 implementations, once a receiver fails to decode a received MAC protocol data unit (MPDU), i.e., not passing checksum verification, it will simply discard it and wait for retransmission. Despite its simplicity, this retransmission based scheme is inefficient in a sense that there could be plenty of uncorrupted bits in the discarded packets. Moreover, when the channel fading or unavoidable interference is very severe, for example in the industrial internet of things scenarios with intense coexistence of wireless devices [2], this scheme can hardly work since 802.11 has very limited error correction capability, whose least coding rate is only 1/2.

To improve the error correction capability of 802.11, many approaches have been proposed by exploiting the uncorrupted bits in corrupted MPDUs. Generally, they can be classified into the following two categories:

- *Hybrid automatic repeat-request (ARQ)* based scheme, e.g., [3–5], where the receiver first determines and reserves the uncorrupted bits, and then requests partial retransmission of those corrupted ones. Such approaches work well if the erroneous bits are not too dense, but they still hardly handle severe channel fading or unavoidable interference where a lot of errors occur within MPDUs since the identification of a limited number of uncorrupted bits would incur significant overhead.
- *Cross-layer forwarding error correction (FEC)* based scheme, where outer codes are concatenated with the convolutional codes of 802.11 to exploit these uncorrupted bits. Previous approaches [6,7] employ fixed-rate outer codes, which require accurate estimation of the packet delivery status based on methods like [8,9], to determine proper coding rates of outer codes in advance. Unfortunately, such approaches are less practical when the channel condition varies quickly.

Apart from the above approaches dedicated to improving the error correction capability of 802.11, there are some latest work trying to mitigate interference introduced by other communication technologies such as ZigBee and Bluetooth [10–14]. Their main idea is that collaborations between different technologies can be enabled based on approaches called cross-technology communication (CTC), where heterogeneous wireless devices can directly communicate with each other, such that a more efficient channel coordination among these devices

☆ This work was partly presented in IEEE Wireless Communications and Networking Conference (WCNC), 2019 [1].

* Corresponding author.
E-mail addresses: tb@nju.edu.cn (B. Tang), yebl@nju.edu.cn (B. Ye), quzhihao@hhu.edu.cn (Z. Qu), sanglu@nju.edu.cn (S. Lu).

https://doi.org/10.1016/j.comnet.2020.107361
Received 4 December 2019; Received in revised form 20 April 2020; Accepted 8 June 2020
Available online 10 June 2020

can be provided. However, these approaches still fail to address severe channel fading as well as interference from hidden terminals.

To extend the applicability of 802.11 in extremely poor channels, in this paper, we focus on utilizing those uncorrupted bits in a more efficient manner, while getting rid of accurate channel status estimation. We propose Rateless802.11, a novel cross-layer FEC based scheme that employs rateless codes—LT codes [15]—as outer codes, and present a well-designed processing flow to correct these corrupted bits at the receiver side, such that additional redundancies for network-layer payload can be incrementally introduced to provide adequate error correction capability. Rateless802.11 consists of three components, named link control, encoding, and decoding, which are well-designed to be compatible with various 802.11 specifications such that Rateless802.11 can work as a middleware over common commodity 802.11 devices. The link control component manages the transmission process of network-layer payload, deciding the introduction of additional redundancies and the acknowledgment of the transmitter. The encoding component is responsible for generating these additional redundancies using LT codes, constructing proper MAC service data units (MSDUs), and forwarding them to 802.11 devices. The most critical decoding component is dedicated to correcting various errors emerged in the receiving path (i.e., the noisy channel for propagating RF signals, and the demodulator, decoder, and descrambler at the receiver side), and reconstructing the network-layer payload.

Towards efficient error correction in Rateless802.11, two challenges should be well addressed. The first is that, while the channel fading or unavoidable interference is very severe, a large number of errors emerged in the receiving path come from the descrambling process, since the randomly generated 802.11 scrambler seed (c.f. Section 5.1) embedded in the header of the physical-layer conformance procedure (PLCP) protocol data units (PPDUs) might not be accurately recovered by the receiver. Moreover, common commodity 802.11 devices usually do not support the customization of such scrambler seed. The second challenge is how to efficiently decode the concatenated codes consisting of LT codes and 802.11 convolutional codes, since the natural serial decoding approaches could incur significant additional information-loss during the decoding, which would degrade the error correction capability. Besides, the decoding process is required to be compatible with commodity 802.11 devices and not incur significant delay.

In the design of Rateless802.11, to address the first challenge, we insert some metadata with specific structures into each MSDU to be transmitted. By exploiting the linearity of scrambling sequences generated from scrambler seeds, these metadata can help protect and recover the valid scrambler seeds for the descrambling process, such that descrambling errors within MPDUs can be eliminated. To address the second challenge, we propose a belief-propagation based integrated decoder named IntBP, which takes the received physical-layer data, i.e., PPDUs or their corresponding baseband symbols as input, and decodes LT codes and convolutional codes in a joint manner, so as to effectively avoid information-loss. In particular, IntBP can make full use of computing resources for parallel computing, and it can be executed in an incremental manner during the reception of each PPDU, both of which lead to a low decoding delay. Additionally, since common commodity 802.11 devices cannot offer detailed physical-layer data, we use an encoder of convolutional codes to obtain pseudo-PPDUs based on these received corrupted MPDUs and feed them into IntBP.

The main contributions of this paper are summarized as follows:

- We propose Rateless802.11, a novel scheme working as a middleware over common commodity 802.11 devices to extend 802.11 applicability in extremely poor channels. By concatenating LT codes with 802.11 convolutional codes, Rateless802.11 can provide additional redundancies properly for network-layer payload without accurate channel status estimation, and exploit the uncorrupted bits of MPDUs adequately.
- By employing the metadata with specific structures, we provide a method to protect and recover the valid scrambler seeds for the descrambling process, which effectively eliminates these descrambling errors within MPDUs. We also present a belief-propagation based integrated decoder for the concatenated codes, which incurs much less information-loss and extra decoding delay than natural serial decoding approaches.
- We evaluate the performance of Rateless802.11 via both numerical and real-trace driven simulations. Performance evaluation results show that Rateless802.11 improves the throughput of 802.11 in extremely poor channels by several to dozens of times compared with state-of-the-art solutions.

The rest of this paper is organized as follows. Section 2 discusses about related work. Section 3 introduces the mechanism and setups of Rateless802.11. Section 4, Section 5, and Section 6 introduce the link control and encoding process, the scrambler seed protection method, and the decoding process of Rateless802.11, respectively. Section 7 presents the analysis of complexity and delay, and Section 8 presents the evaluation results. Finally, Section 9 concludes.

2. Related work

In the lastest specifications, 802.11 has been extended to support various communication techniques, like low-density parity check (LDPC) codes [16] and multiple-input and multiple-output (MIMO) in 802.11n, downlink multiuser MIMO (MU-MIMO) in 802.11ac [17], uplink MU-MIMO in 802.11ax [18], millimeter wave communication in 802.11ad [19], and sub-1GHz communication in 802.11af [20]. However, in all these specifications, issues regarding utilizing uncorrupted bits of MPDUs and communication in extremely poor channels have not been well resolved yet.

2.1. Hybrid ARQ for 802.11

To improve the performance of 802.11 by exploiting the uncorrupted bits in MPDUs, the first category of related work is using hybrid ARQ to request partial retransmission, e.g., [3–5], where the receiver first determines and reserves the uncorrupted bits, and then requests retransmission of those corrupted ones. Such approaches work well if the erroneous bits are not too dense, but they still hardly handle severe channel fading or unavoidable interference where amounts of errors occur within MPDUs. For example, [4] divides payload to transmit into chunks each of which is attached with a checksum, such that receivers can figure out those corrupted chunks and request partial retransmission. If the channel condition is rather poor, for example, in a binary symmetric channel (BSC) of capacity 1/2, i.e., crossover probability in BSC is around 0.11, this approach works badly (c.f. Fig. 2 in Section 3), where the error ratio of each 32-bit chunk is around 50% for 802.11 1/2 convolutional code—the least coding rate of 802.11. That is, most chunks may keep on requiring retransmissions until the channel condition becomes better.

2.2. Cross-Layer FEC for 802.11

The second category is using cross-layer FEC to exploit these uncorrupted bits, e.g., [6,7], where fixed-rate outer codes are concatenated with 802.11 convolutional codes. These approaches need to accurately estimate the capacity of the channel and determine the best coding rate of the outer codes in advance. For example, in [7,9], the bitwise bit-error-ratio (BER) is estimated at the 802.11 transmitter side and different sourceword bits are provided with unequal protection using Reed-Solomon (RS) codes. It is less practical while the channel condition varies quickly, and according to our experiments (c.f. Section 8.2.4), the protection provided by RS codes is quite limited if the estimated BER is not very small. For example, in BSC of capacity 1/2, the error

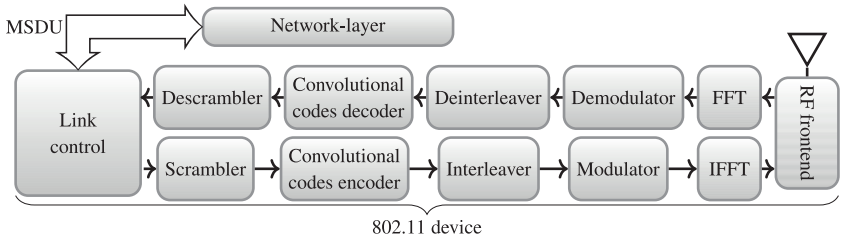


Fig. 1. Processing flow of 802.11g.

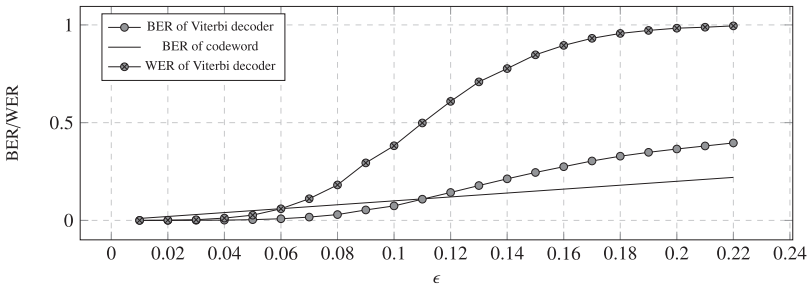


Fig. 2. BER and word-error-ratio (WER, error ratio of each 32-bit chunk) of 802.11 1/2 convolutional code decoded by the Viterbi decoder in BSC with different crossover probability ϵ.

ratio of each 127 bytes is around 80% for the concatenated code consisting of a (255,127) RS code[1] and 802.11 1/2 convolutional code—the total coding rate is around 1/4.

2.3. Others for 802.11

In addition to the above two categories of work which focus on utilizing the uncorrupted bits in MPDUs while not requiring any modification of the physical-layer coding and modulation standards of 802.11, there are some other related work. It includes content-aware cross-layer design based work like Chen *et al.* [21], which provide a content-aware retransmission limit adaptation scheme for video streaming over 802.11, and Khan *et al.* [22], which provide a cross-layer strategy to optimally utilize the application and physical-layer resources for unequal error protection of wavelet coded scalable video. However, neither of them can handle those extremely poor channels.

There are also some other work concentrating on improving the performance of 802.11 by studying the delivery status of MPDUs in selective fading channels, like [8,25,23,24]. Among them, Vutukuru *et al.* [23] present SoftPHY to estimate the bitwise BER of delivered MPDUs at the receiver side using physical-layer information, while methods provided in [8,25,24] manage to estimate the average BER to determine the most appropriate coding rate at the transmitter side. Different from these work which avoid modification of the physical-layer standard of 802.11, related work like [26–28] redesign the coding and modulation process to address the coding rate selection problem and selective fading problem during wideband wireless communication.

2.4. Rateless codes

Rateless codes such as LT codes [15] and Raptor codes [29] are widely used for erasure correction in many applications [30]. Regarding rateless codes in noisy channels, Iannucci *et al.* [31] design a practical link control suitable for rateless codes to guarantee reliable communication, where very little feedback from receivers is needed. Gudipati *et al.* [32] present a rateless coding scheme called strider codes to automatically handle collisions, which achieves the capacity of additive white Gaussian noise channel (AWGNC) at any signal-noise-ratio (SNR). Perry *et al.* propose a hashing function based rateless coding scheme called spinal codes [33], which can achieve the capacity of BSC and AWGNC as well. However, all these proposals cannot be directly taken as the outer codes in current 802.11 specifications due to the restrictions of their inputs and outputs.

[1] In (255,127) RS code, every 127 input bytes are encoded into 255 output bytes.

3. Design of Rateless802.11

In this section, we introduce the main mechanism of Rateless802.11 and show its compatibility with 802.11 devices for different setups.

3.1. Errors in 802.11 MPDUs

We first introduce the processing flow of common commodity 802.11 devices. In conventional 802.11, as illustrated in Fig. 1, MSDUs offered by network-layer for transmission are wrapped in MPDUs first and processed by the scrambler of 802.11 devices to add randomness. The scrambled bit sequence is then processed by the convolutional codes encoder (or LDPC codes encoder in later 802.11 specifications) to add redundancies. To avoid burst errors, these encoded bits are further interleaved and mapped to different subcarriers of orthogonal frequency-division multiplexing (OFDM) symbols by the interleaver. If MIMO is not enabled, after the processing of modulator and inverse fast fourier transformation (IFFT) block, time-domain baseband samples of MPDUs are obtained. By concatenating preambles of training sequence and the samples of signal field, they can finally be interpolated, up-converted to passband and sent out.

During the receiving path, the received passband signal is down-converted to baseband and sampled first. After sequentially executing the faster fourier transformation (FFT) block, the demodulator, the deinterleaver, the Viterbi algorithm based convolutional codes decoder (or belief-propagation based LDPC codes decoder) [7], and the descrambler, MPDUs successfully decoded can forward their MSDUs to network-layer while those fail to decode are usually discarded and retransmissions of them are waited.

The error correction capability of 802.11 is very limited to deal with severe channel fading or unavoidable interference, especially for its convolutional codes, which has the error-spreading problem (i.e., the BER of recovered sourceword is higher than that of codeword before decoding) resulted by the Viterbi decoder as shown in Fig. 2. In addition to these errors in 802.11 MPDUs incurred by the Viterbi decoder, the descrambling process can also introduce an amount of errors if its recovered scrambler seed is an error. Both of them raise challenges to designing schemes based on commodity 802.11 devices to handle those

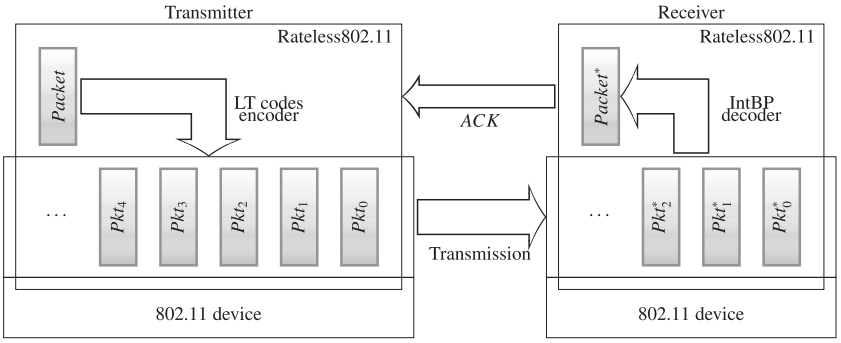


Fig. 3. Mechanism overview.

extremely poor channels using corrupted MPDUs only. In this paper, we present Rateless802.11 to extend the applicability of these convolutional codes based 802.11 devices in extremely poor channels. For devices using LDPC codes for error correction, our proposal can be extended to support them by simply modifying the decoding process. We list some of its simulation results in Section 8.4, but do not discuss it in detail here.

3.2. Mechanism of Rateless802.11

Our Rateless802.11 is essentially a MAC-layer FEC based scheme, which can be applied over common commodity 802.11 devices, where protocols like CSMA/CA for carrier transmission are not affected, i.e., ordinary interference can be avoided normally. It is designed to handle extremely poor channels caused by fading or interference from hidden terminals or devices of other protocols, by incrementally introducing redundancies. Its basic mechanism is illustrated in Fig. 3. Given network-layer payload denoted by *Packet*, it is encoded by the LT codes encoder into a sequence of packets, i.e., Pkt_0, Pkt_1, ⋯, which are sent out sequentially through the 802.11 device. At the receiver side, once the transmission of these packets is detected, the decoding process begins. Once the *Packet** decoded based on the captured packets, i.e., Pkt_0^*, Pkt_1^*, ⋯, is verified to be a success, an *ACK* is feedback.

To ensure that the above mechanism can efficiently handle those extremely poor channels and is fully compatible with common commodity 802.11 devices, in Rateless802.11, we provide a method to protect and recover the valid scrambler seeds for descrambling MPDUs, and propose the belief-propagation based integrated decoder (IntBP) to jointly decode convolutional codes and LT codes, so as to avoid information-loss during serial decoding approaches, guarantee its error correction capability, and reduce its decoding delay.

3.3. Transmitting process of Rateless802.11

More specifically, at the transmitter side, as shown in Fig. 4, the network-layer payload is appended with its frame check sequence (FCS) first. The sourceword formed by them is then encoded into a long sequence of parity bits with LT codes encoder. After that, the payload, FCS and parity bits, i.e., the inter codeword, are sliced into different chunks, appended with different headers of metadata, and form the MSDUs. These MSDUs are then forwarded to 802.11 devices and wrapped in PPDUs. Metadata appended to each chunk consists of the sequence for 802.11 scrambler seed protection, the type of the MSDU, the identities of the transmitter and receiver, the index of the chunk, the length of the chunk, etc., which are protected with repeated fixed-rate LT codes individually. Its detailed structure is shown in Fig. 5.

3.4. Receiving process of Rateless802.11

Based on the capabilities of the receiver side 802.11 device, we present two setups in the receiving process of Rateless802.11, i.e., the **Regular-Setup** and the **Advanced-Setup** as shown in Table 1.

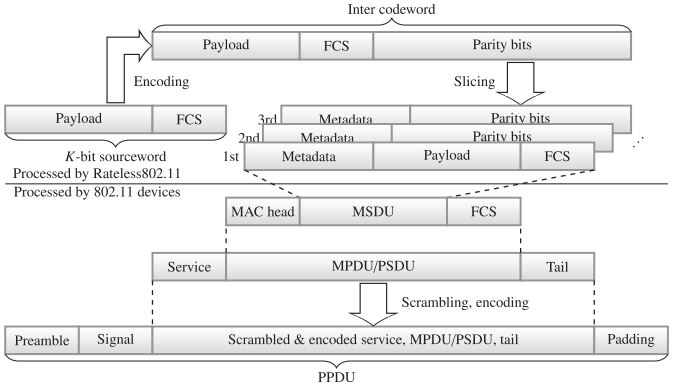


Fig. 4. Packet structure of Rateless802.11.

Table 1
Setup and requirement of Rateless802.11.

Setup	Transmitter requirement	Receiver requirement
Regular-Setup	Disable conventional retransmission	Offer MPDU with FCS error and SNR
Advanced-Setup	Disable conventional retransmission	Offer PPDU or its baseband symbols and SNR

Specifically, for common commodity 802.11 devices which can only offer MPDUs with FCS errors, we have the Regular-Setup of Rateless802.11. As shown in Fig. 6, MSDUs within these corrupted MPDUs are forwarded to the middleware Rateless802.11 for processing. Within the middleware, the valid scrambler seeds for descrambling are recovered first based on the scrambler seed protection sequences embedded in metadata within these MSDUs (c.f. Section 5.3). Then, with these recovered valid scrambler seeds, the convolutional codes encoder is taken to encode these MSDUs into pseudo-PPDUs, and these pseudo-PPDUs are further processed by IntBP, which recovers the information bits within metadata, the payload, and the FCS (c.f. Section 6.1.1). Finally, the obtained payload can be forwarded by the link control component to the network-layer.

For 802.11 devices like WARP [34] which can offer PPDUs or even their corresponding baseband symbols, i.e., samples before demodulation, we have the Advanced-Setup of Rateless802.11. As shown in Fig. 7, the scrambler seeds using at the transmitter side can be directly recovered first (c.f. Section 5.4). With the recovered scrambler seeds and the captured PPDUs or their baseband symbols as input, IntBP is directly taken to recover the information bits within metadata, the payload, and the FCS (c.f. Section 6.1.2). Finally, the payload is forwarded to the network-layer by the link control component.

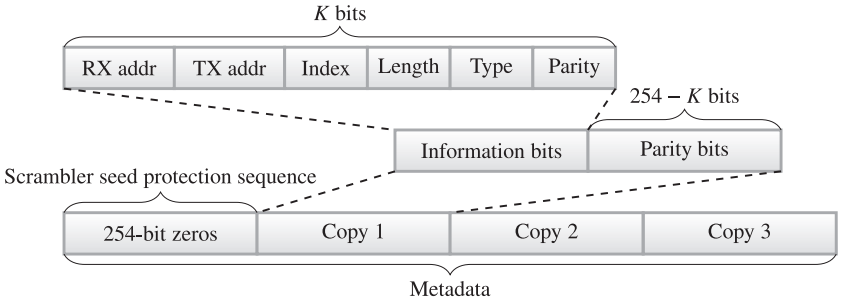


Fig. 5. Metadata structure of Ratelss802.11.

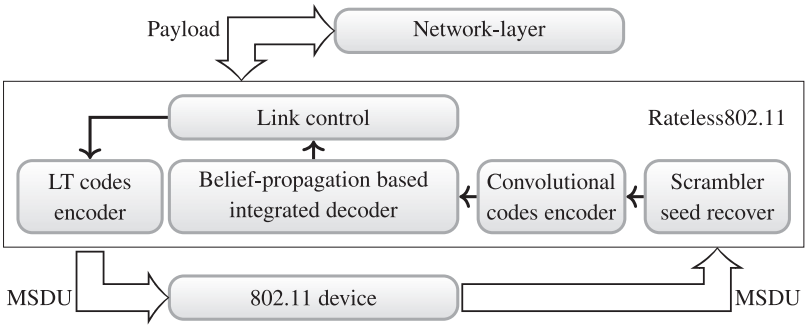


Fig. 6. Processing flow of Rateless802.11 with Regular-Setup.

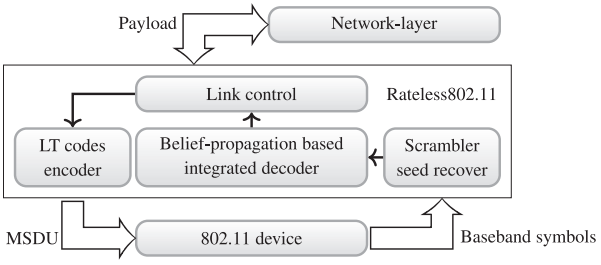


Fig. 7. Processing flow of Rateless802.11 with Advanced-Setup.

4. Link control and encoding process of Rateless802.11

4.1. Link control

We provide a simple link control within Rateless802.11 which manages the transmission process of network-layer payload. It maintains the queues of PPDUs which are ready to send and have been captured for payloads during sending and receiving, respectively. The link control of Rateless802.11 is essentially a variation of stop-and-wait. A unicast example is illustrated in Fig. 8. In this example, $PPDU_0^*$ is captured by the receiver first but fails to decode since too many errors are within it. Since no *ACK* is captured by the transmitter till the timer expires, $PPDU_1$ is sent out, which is not even detected by the receiver. As still no *ACKs* are captured, $PPDU_2$ and $PPDU_3$ are sent out successively at the transmitter side. Payload is finally recovered based on $PPDU_0^*$, $PPDU_2^*$ and $PPDU_3^*$, and an *ACK* is feedback which ends the unicast.

4.2. Payload encoding process

From K-bit sourceword, i.e., payload and FCS, the LT codes encoder generates a long sequence of parity bits, each of which is obtained as follows.

– Pick a degree d randomly according to some given degree distribution.

– Choose uniformly at random $d-1$ sourceword bits and return their XOR-sum.

The distribution of degree d can be characterized by the following polynomial

$$\mu(x) = \sum_i \mu_i x^{i-1}, \tag{1}$$

where the coefficient μ_i denotes the fraction of degree $d = i + 1$. Here, we consider the use of a pseudorandom seed to sample degrees and bits, which is known by both the transmitter and the receiver beforehand. As Fig. 4 shows, the K-bit sourceword is then wrapped in the first PPDU, and the long sequence of parity bits is further sliced into different chunks and wrapped in subsequent PPDUs. The size of each chunk of parity bits can be dynamically adjusted.

4.3. Metadata encoding process

Each PPDU also contains the metadata appended to the sourceword or chunk of parity bits. Fig. 5 illustrates its encoding process where K information bits, i.e., the identities of the transmitter and receiver, the index and the length of the chunk, etc., are first appended with $254 - K$ parity bits generated by LT codes encoder. Such 254-bit copy is then repeated for 3 times to further deal with the error-spreading problem of convolutional codes and appended to the 254-bit scrambler seed protection sequence. All the 1016 bits form the metadata within each PPDU. The reason why both the scrambler seed protection sequence and the LT codes codeword copy have size of 254 bits is highly related to the scrambling process of 802.11 devices, which is explained in detail in Section 5 and Section 8.

5. Scrambler seed protection of Rateless802.11

In this section, we introduce the scrambler seed protection method in Rateless802.11. We start by reviewing the scrambling process of 802.11 devices.

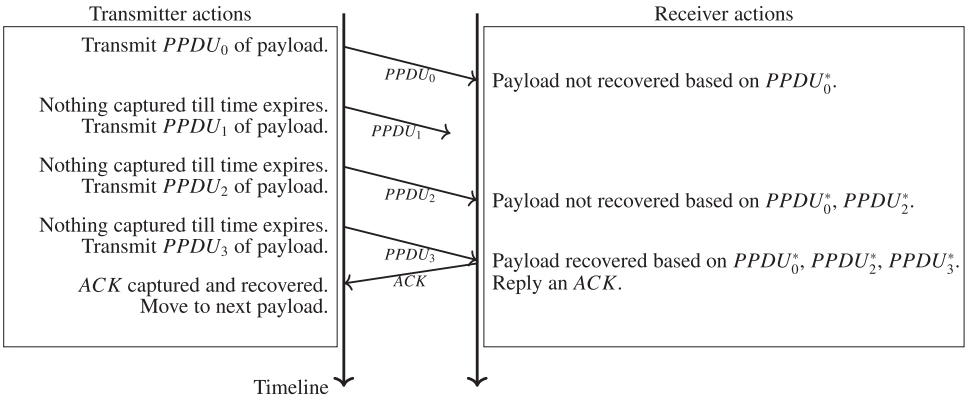


Fig. 8. An illustration of unicast in Rateless802.11.

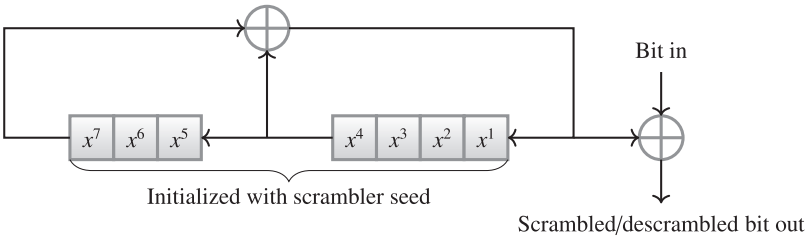


Fig. 9. Scrambling process of 802.11 devices where the scrambling sequence is generated based on the scrambler seed through the shift and XOR operations. By performing bitwise XOR of the scrambling sequence with the input bit stream, the scrambled/descrambled bit stream can be obtained.

5.1. Scrambling process of 802.11 devices

The scrambling process of 802.11 devices can be viewed as the bitwise XOR of the sequence consisting of service field, MPDU and tail, with a scrambling sequence. The first 7 bits of each scrambling sequence is the scrambler seed to generate this whole scrambling sequence, and the rest bits are generated as shown in Fig. 9. Usually, such 7-bit seed is assigned randomly by hardware and most commodity 802.11 devices do not support manual setting by software. Note that the first 7 bits of service field after scrambling is exactly the 7-bit seed since the service field is initialized with all zeros.

At the receiver side, after obtaining the decoding result of the Viterbi decoder for convolutional codes, i.e., the sequence consisting of scrambled service field, MPDU, and tail, it takes the first 7 bits of service field as the scrambler seed to generate the whole scrambling sequence and executes the descrambling process, i.e., performing bitwise XOR. Note that if the obtained scrambler seed contains error bits, the descrambling process will introduce a great number of errors to MPDU, where around half of its bits can be erroneous after descrambling, and the MPDU has to be discarded.

Most related MAC-layer FEC based work for 802.11 like [6,7] do not pay attention to the problem that descrambling can introduce enormous errors. They simply assume that the channel status is not that bad, such that the scrambler seed recovered at the receiver side is correct, or the scrambler seed is fixed and known to each receiver, which is usually too ideal since common commodity 802.11 devices do not support the customization of these scrambler seeds. While these assumptions are reasonable when the channel condition is good, they fail to hold for extremely poor channels. In the following, we present a method to protect the scrambler seed in Rateless802.11.

5.2. Scrambler seed protection sequence

The generation of each scrambling sequence can be viewed as encoding each 7-bit seed with a linear code. Therefore, the bitwise XOR of any two sequences within all the 128 (i.e., 2^7) scrambling sequences is still a valid scrambling sequence. Additionally, it can be verified that each of the 128 scrambling sequences is cyclic with 127-bit period, i.e., its i-th bit is exactly the same as its $(i+127)$-th bit. Based on this property, we insert a 254-bit all-zero sequence, referred to as the scrambler seed protection sequence, into the metadata as shown in Fig. 5. Let $\mathbf{e}$ be the scrambler seed using at the transmitter side. After performing scrambling at the transmitter side, such 254-bit all-zero sequence is scrambled into $scram(\mathbf{e})$, where function $scram(\cdot)$ returns the corresponding 254 bits within the scrambling sequence generated from $\mathbf{e}$. Note that $scram(\mathbf{e})$ contains exactly 2 periods of the scrambling sequence. By utilizing such 2 periods of the scrambling sequence, we can recover the scrambler seed even if the channel fading or unavoidable interference is very severe.

5.3. Scrambler seed recovery in Regular-Setup

In Regular-Setup, only the decoded and descrambled MPDU can be offered to the middleware Rateless802.11. Within such decoded and descrambled MPDU, let $scram(\mathbf{e})^* \oplus scram(\mathbf{e}')$ denote the scrambler seed protection sequence. Here $scram(\mathbf{e})^*$ is the 254-bit scrambler seed protection sequence decoded by the receiver side Viterbi decoder, $\mathbf{e}'$ is the scrambler seed recovered at the receiver side ($\mathbf{e}'$ does not necessarily equal $\mathbf{e}$ using at the transmitter side), and $\oplus$ denotes bitwise XOR. Ideally, we should recover both unknown seeds $\mathbf{e}$ and $\mathbf{e}'$, and perform descrambling separately, so as to successfully descramble the MPDU. However, considering the linearity of the scrambling sequences, i.e., $scram(\mathbf{e} \oplus \mathbf{e}') = scram(\mathbf{e}) \oplus scram(\mathbf{e}')$, our idea is that we can directly recover the valid seed $\mathbf{e} \oplus \mathbf{e}'$ instead. Let $\mathrm{H}(\cdot)$ denote Hamming distance between input sequences. By enumeration according to the following formula

$$(\mathbf{e} \oplus \mathbf{e}')^* = \arg\min_{\mathbf{e} \oplus \mathbf{e}'} \mathrm{H}\big(scram(\mathbf{e} \oplus \mathbf{e}'), scram(\mathbf{e})^* \oplus scram(\mathbf{e}')\big), \tag{2}$$

we can obtain the valid scrambler seed $(\mathbf{e} \oplus \mathbf{e}')^*$, perform the descrambling process for the decoded and descrambled MPDU offered by the device, and obtain the final descrambled MPDU. Note that the computational complexity of this enumeration process is acceptable since the number of optional $\mathbf{e} \oplus \mathbf{e}'$ is only 128.

5.4. Scrambler seed recovery in Advanced-Setup

In Advanced-Setup, the PPDU or its corresponding baseband symbols can be offered to the middleware Rateless802.11. Let $\mathbf{y}$ be the received

bit sequence or baseband symbol sequence corresponding to the 254-bit scrambler seed protection sequence within the PPDU. We can recover the valid $\mathbf{e}$ directly by enumeration according to the following formula

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \mathrm{P}\Big(conv\big(scram(\mathbf{e})\big)|\mathbf{y}\Big), \tag{3}$$

where $\mathrm{P}(\cdot)$ denotes probability and $conv(\cdot)$ denotes convolutional codes encoder. Since the number of optional $\mathbf{e}$ is 128, the computational complexity of this enumeration process is also acceptable.

6. Decoding process of Rateless802.11

In this section, we focus on the decoding process of Rateless802.11 and introduce a belief-propagation based integrated decoder named IntBP. Let LTConv denote the codes concatenating LT codes and convolutional codes for payload and FCS, LTRepConv the codes concatenating repeated LT codes and convolutional codes for information bits within metadata. Note that LTConv and LTRepConv are decoded by IntBP in similar manners. We introduce IntBP for LTConv first, and then we introduce IntBP for LTRepConv. At the end of this section, we show how to execute IntBP in an incremental manner during the reception of each PPDU, which further reduces the decoding delay.

6.1. Mechanism of IntBP for LTConv

To decode concatenated codes like LTConv, the most natural approach is decoding them serially, i.e., decoding convolutional codes (inner codes) first and then decoding LT codes (outer codes) based on the decoding results of inner codes. Such serial decoding approaches can be viewed as the following two-step optimization with a maximum a posteriori (MAP) decoder

$$\begin{aligned}\mathbf{x}_i^* &= \arg\max_{\mathbf{x}_i} \mathrm{P}(\mathbf{x}_i|\mathbf{y}_i),\\ \mathbf{b}^* &= \arg\max_{\mathbf{b}} \mathrm{P}\big(\mathbf{b}|\cdots,\mathbf{x}_i^* \oplus scram(\mathbf{e}_i),\cdots\big),\end{aligned} \tag{4}$$

where $\mathbf{y}_i$ is the received bit sequence or baseband symbol sequence corresponding to the LTConv codeword within $PPDU_i$, $\mathbf{x}_i^*$ denotes the decoding results of convolutional codes, $\mathbf{e}_i$ is the scrambler seed using at the transmitter side, $scram(\cdot)$ produces the scrambling sequence, and $\mathbf{b}^*$ denotes the decoding results of LT codes.

Instead of decoding such concatenated codes serially, which can incur information-loss and additional decoding delay during the two-step optimization, our idea is to treat the inner codes and outer codes as a whole and to decode them in a joint manner, which can be viewed as directly solving

$$\mathbf{b}^* = \arg\max_{\mathbf{b}} \mathrm{P}(\mathbf{b}|\cdots,\mathbf{y}_i,\mathbf{e}_i,\cdots), \tag{5}$$

using a single-step optimization. In addition to helping avoid information-loss and offering an opportunity to reduce the decoding delay, it is worth noting that when the channel fading or unavoidable interference is very severe, not only convolutional codes of 802.11 have the error-spreading problem while decoding by the Viterbi decoder, but also LT codes have the error-floor problem [35], both of which can be well solved by decoding the concatenated codes in a joint manner.

To further decrease the computational complexity of (5), we take the sum-product algorithm which operates on the factor-graph in a message-passing manner [36] to simplify the decoding process. Our proposal IntBP is essentially a variation of the sum-product algorithm for decoding the concatenated codes LTConv, by propagating messages in the form of probability mass function (PMF, i.e., the belief).

6.1.1. IntBP in Regular-Setup for LTConv

In Regular-Setup, within the decoded and descrambled MPDU, only the inter codeword of LTConv, i.e., $\mathbf{x}_i^* \oplus scram(\mathbf{e}_i')$, can be offered to the middleware Rateless802.11, where $\mathbf{e}_i'$ is the scrambler seed recovered at the receiver side for $PPDU_i$. In order to use IntBP, we feed the codeword bit sequence $\mathbf{y}_i^*$ within the pseudo-$PPDU_i$ into IntBP, and treat bits of $\mathbf{y}_i^*$ as the output symbols of memoryless BSC. That is, IntBP in Regular-Setup can be viewed as solving

$$\mathbf{b}^* = \arg\max_{\mathbf{b}} \mathrm{P}(\mathbf{b}|\cdots,\mathbf{y}_i^*,\mathbf{0},\cdots), \tag{6}$$

using the sum-product algorithm operating on the factor-graph, where we take the convolutional codes encoder to obtain $\mathbf{y}_i^*$ within pseudo-$PPDU_i$, i.e.,

$$\mathbf{y}_i^* = conv\Big(\mathbf{x}_i^* \oplus scram(\mathbf{e}_i') \oplus scram\big((\mathbf{e}_i \oplus \mathbf{e}_i')^*\big)\Big). \tag{7}$$

Here $(\mathbf{e}_i \oplus \mathbf{e}_i')^*$ is obtained through the scrambler seed protection method for Regular-Setup in Section 5.3.

6.1.2. IntBP in Advanced-Setup for LTConv

In Advanced-Setup, if what offered to the middleware Rateless802.11 is the received bit sequence corresponding to the LTConv codeword within $PPDU_i$, we let $\mathbf{y}_i$ denote the bit sequence, and treat its bits as the output symbols of memoryless BSC. If what offered to Rateless802.11 is the baseband symbol sequence corresponding to the LTConv codeword, we let $\mathbf{y}_i$ be the baseband symbol sequence after equalization, and treat its symbols as the output of memoryless binary-input additive white Gaussian noise channel (BI-AWGNC) for binary phase-shift keying (BPSK) modulation. That is, IntBP in Advanced-Setup can be viewed as solving

$$\mathbf{b}^* = \arg\max_{\mathbf{b}} \mathrm{P}(\mathbf{b}|\cdots,\mathbf{y}_i,\mathbf{e}_i^*,\cdots), \tag{8}$$

using the sum-product algorithm operating on the factor-graph, where $\mathbf{e}_i^*$ is obtained through the scrambler seed protection method for Advanced-Setup in Section 5.4.

6.2. Executing IntBP for LTConv

The factor-graph of LTConv is illustrated in Fig. 10 which is a bipartite graph including two kinds of nodes, variable nodes and check nodes. In this factor-graph, B_i, $i \geq 0$ is the LT codes sourceword variable node, corresponding to random variable $b_i \in \{0, 1\}$, the i-th LT codes sourceword bit. X_i, $i \geq 0$ is the inter codeword variable node, corresponding to random variable $x_i \in \{0, 1\}$, the i-th inter codeword bit. S_i, $i \geq 0$ is the convolutional codes state variable node, corresponding to random variable $s_i \in \{000000, 000001, 000010, \cdots, 111111\}$ in binary form. C_i, $i \geq 0$ is the convolutional codes codeword variable node, corresponding to random variable $c_{2i}, c_{2i+1} \in \{0, 1\}$, the convolutional codes codeword bits while the coding rate is 1/2. Besides these variable nodes, G_i is the LT codes check node, denoting the relationship between X_i and all its connected B_j. F_i is the convolutional codes check node, denoting the relationship between X_i, C_i, and its connected S_i, S_{i+1}.

Let y_i, $i \geq 0$ denote the channel output symbols corresponding to convolutional codes codeword bit c_i, where $y_i \in \{0, 1\}$ in BSC and $y_i \in \mathbb{R}$ in BI-AWGNC. Let $\mathcal{G}$ denote the factor-graph shown in Fig. 10. IntBP for LTConv is presented in Algorithm 1, which is executed when each PPDU or pseudo-PPDU is obtained. It takes the factor-graph $\mathcal{G}$, the scrambler seed $\mathbf{e}$, and the channel output symbols $y_0, y_1, \cdots, y_{2i}, y_{2i+1}, \cdots$, as input, propagates messages between variable nodes and check nodes in its MaxIterationNum iterations, and exports final decoding result $b_0^*, \cdots, b_{K-1}^*$ by maximum the bitwise likelihood using the following formula

$$b_i^* = \arg\max_{b_i} A \prod_{(B_i,G_j)\in\mathcal{G}} \mathrm{P}_{G_j\to B_i}(b_i), \tag{9}$$

where $\mathrm{P}_{G_j\to B_i}(b_i)$ denotes the message propagated from node G_j to B_i, in the form of PMF of random variable b_i.

6.2.1. Variable node message in IntBP for LTConv

The PMF propagated from C_i to F_i is given by

$$\mathrm{P}_{C_i\to F_i}(c_{2i}, c_{2i+1}) = A \cdot \mathrm{P}(y_{2i}|c_{2i}) \cdot \mathrm{P}(y_{2i+1}|c_{2i+1}), \tag{10}$$

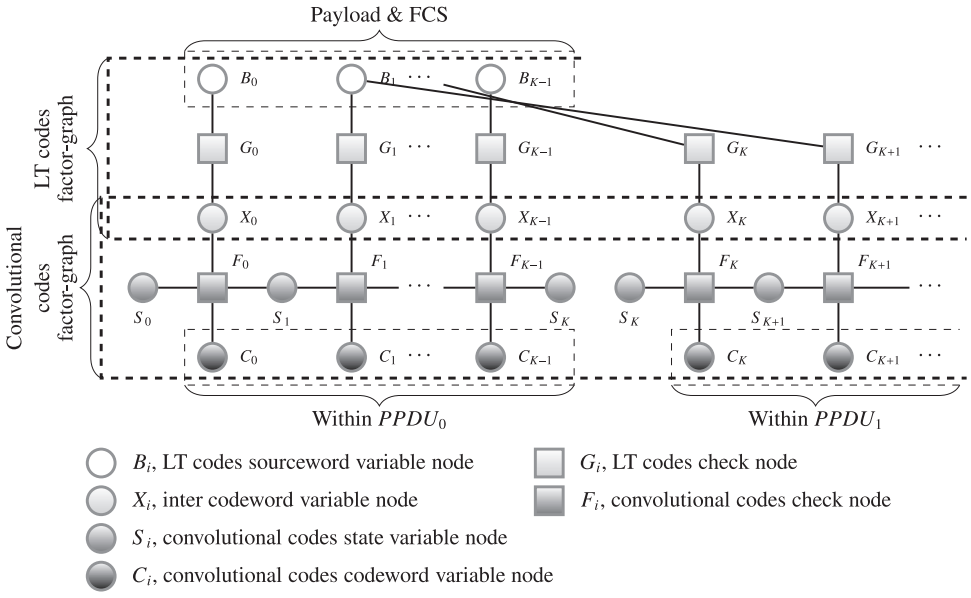


Fig. 10. The factor-graph of LTConv for IntBP.

Algorithm 1 IntBP for LTConv.

Input: $y_0, y_1, \cdots, y_{2i}, y_{2i+1}, \cdots, \mathcal{G}$, **e**
Output: $b_0^*, \cdots, b_{K-1}^*$
1: **for** iterator = 0 to MaxIterationNum − 1 **do**
2: propagate PMFs of all variable nodes given by (10), (13) - (17) to their adjacent check nodes in $\mathcal{G}$
3: propagate PMFs of all check nodes given by (18), (20) - (24) to their adjacent variable nodes in $\mathcal{G}$
4: **end for**
5: obtain $b_0^*, \cdots, b_{K-1}^*$ with (9)
6: **if** $b_0^*, \cdots, b_{K-1}^*$ passes CRC32 **then**
7: **return** $b_0^*, \cdots, b_{K-1}^*$
8: **end if**

where A is the normalization coefficient. In BSC with crossover probability ϵ, i.e., in Regular-Setup or in Advanced-Setup that only offers bit sequence, for $k \in \{0, 1\}$,

$$\mathrm{P}(y_{2i+k}|c_{2i+k}) = \begin{cases} \epsilon, & y_{2i+k} \neq c_{2i+k}, \\ 1-\epsilon, & \text{o.w.}. \end{cases} \tag{11}$$

In BI-AWGNC with SNR $= 1/\sigma^2$, i.e., in Advanced-Setup that offers baseband symbol sequence, for $k \in \{0, 1\}$,

$$\mathrm{P}(y_{2i+k}|c_{2i+k}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_{2i+k}-2c_{2i+k}+1)^2}{2\sigma^2}}. \tag{12}$$

The PMF propagated from X_i to F_i is given by

$$\mathrm{P}_{X_i\to F_i}(x_i) = \begin{cases} 1/2, & \text{1st iteration}, \\ \mathrm{P}_{G_i\to X_i}(x_i), & x_i \text{ is scrambled by 0}, \\ 1-\mathrm{P}_{G_i\to X_i}(x_i), & \text{o.w.}. \end{cases} \tag{13}$$

The PMF propagated from X_i to G_i is given by

$$\mathrm{P}_{X_i\to G_i}(x_i) = \begin{cases} 1/2, & \text{1st iteration}, \\ \mathrm{P}_{F_i\to X_i}(x_i), & x_i \text{ is scrambled by 0}, \\ 1-\mathrm{P}_{F_i\to X_i}(x_i), & \text{o.w.}. \end{cases} \tag{14}$$

The PMF propagated from S_i to F_{i-1} is given by

$$\mathrm{P}_{S_i\to F_{i-1}}(s_i) = \begin{cases} 1/64, & S_i \text{ has no right neighbor}, \\ \mathrm{P}_{F_i\to S_i}(s_i), & \text{o.w.}. \end{cases} \tag{15}$$

The PMF propagated from S_i to F_i is given by

$$\mathrm{P}_{S_i\to F_i}(s_i) = \begin{cases} 1/64, & S_i \text{ has no left neighbor}, \\ \mathrm{P}_{F_{i-1}\to S_i}(s_i), & \text{o.w.}. \end{cases} \tag{16}$$

The PMF propagated from B_i to G_j is given by

$$\mathrm{P}_{B_i\to G_j}(b_i) = \begin{cases} 1/2, & \text{1st iteration}, \\ A \prod\limits_{\substack{(B_i,G_k)\in\mathcal{G} \\ k\neq j}} \mathrm{P}_{G_k\to B_i}(b_i), & \text{o.w.}. \end{cases} \tag{17}$$

6.2.2. Check node message in IntBP for LTConv

The PMF propagated from F_i to X_i is given by

$$\begin{aligned} \mathrm{P}_{F_i\to X_i}(x_i) = A \sum_{\sim x_i} \big[& \mathrm{I}_{\mathrm{conv}}(s_i, x_i, s_{i+1}, c_{2i}, c_{2i+1}) \\ & \cdot \mathrm{P}_{S_i\to F_i}(s_i) \cdot \mathrm{P}_{S_{i+1}\to F_i}(s_{i+1}) \\ & \cdot \mathrm{P}_{C_i\to F_i}(c_{2i}, c_{2i+1}) \big], \end{aligned} \tag{18}$$

where edges $(S_i, F_i), (S_{i+1}, F_i) \in \mathcal{G}$ and $\sim x_i$ means all random variables except x_i. $\mathrm{I}_{\mathrm{conv}}(s_i, x_i, s_{i+1}, c_{2i}, c_{2i+1})$ is indicator function showing the relationship between inter codeword bits, convolutional codes codeword bits and values of states. It is given as

$$\begin{aligned} \mathrm{I}_{\mathrm{conv}}(s_i, x_i, s_{i+1}, c_{2i}, c_{2i+1}) = & \delta\big((2s_i + x_i) \bmod 64 - s_{i+1}\big) \\ & \cdot\delta\Big((x_i + s_i + \Big\lfloor\frac{s_i}{2}\Big\rfloor + \Big\lfloor\frac{s_i}{4}\Big\rfloor + \Big\lfloor\frac{s_i}{32}\Big\rfloor) \bmod 2 - c_{2i}\Big) \\ & \cdot\delta\Big((x_i + \Big\lfloor\frac{s_i}{2}\Big\rfloor + \Big\lfloor\frac{s_i}{4}\Big\rfloor + \Big\lfloor\frac{s_i}{16}\Big\rfloor + \Big\lfloor\frac{s_i}{32}\Big\rfloor) \bmod 2 - c_{2i+1}\Big). \end{aligned} \tag{19}$$

where $\delta(\cdot)$ is the Dirac delta function.

The PMF propagated from F_i to S_{i+1} is given by

$$\begin{aligned} \mathrm{P}_{F_i\to S_{i+1}}(s_{i+1}) = A \sum_{\sim s_{i+1}} \big[& \mathrm{I}_{\mathrm{conv}}(s_i, x_i, s_{i+1}, c_{2i}, c_{2i+1}) \\ & \cdot \mathrm{P}_{S_i\to F_i}(s_i) \cdot \mathrm{P}_{X_i\to F_i}(x_i) \\ & \cdot \mathrm{P}_{C_i\to F_i}(c_{2i}, c_{2i+1}) \big], \end{aligned} \tag{20}$$

where edge $(S_i, F_i) \in \mathcal{G}$.

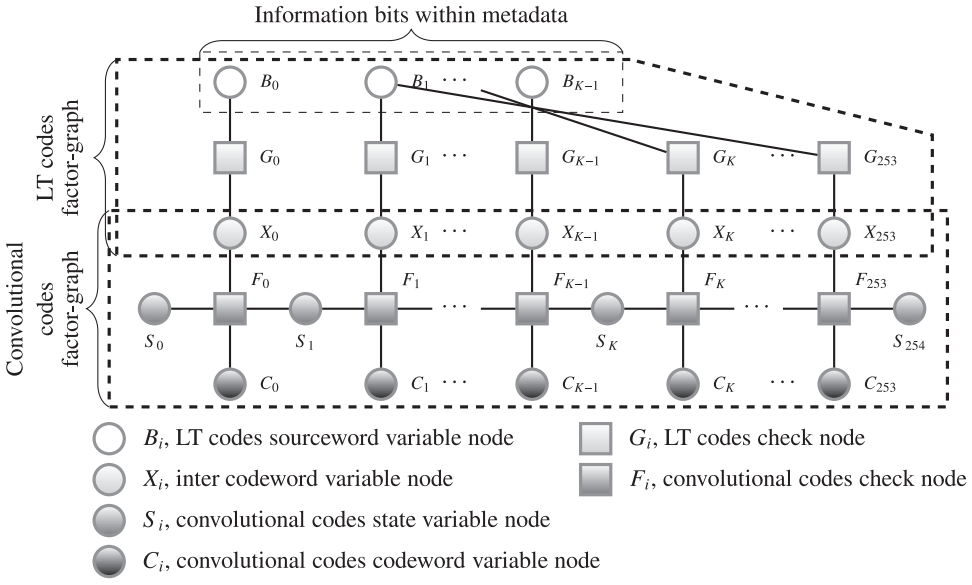


Fig. 11. The factor-graph of LTRepConv for IntBP.

The PMF propagated from F_i to S_i is given by

$$\begin{aligned}\mathrm{P}_{F_i\to S_i}(s_i) = A\sum_{\sim s_i}\big[&\mathrm{I}_{\mathrm{conv}}(s_i,x_i,s_{i+1},c_{2i},c_{2i+1})\\ &\cdot\mathrm{P}_{S_{i+1}\to F_i}(s_{i+1})\cdot\mathrm{P}_{X_i\to F_i}(x_i)\\ &\cdot\mathrm{P}_{C_i\to F_i}(c_{2i},c_{2i+1})\big],\end{aligned}\tag{21}$$

where edge $(S_{i+1}, F_i) \in \mathcal{G}$.

The PMF propagated from F_i to C_i is given by

$$\begin{aligned}\mathrm{P}_{F_i\to C_i}(c_{2i},c_{2i+1}) = A\sum_{\sim c_{2i},c_{2i+1}}\big[&\mathrm{I}_{\mathrm{conv}}(s_i,x_i,s_{i+1},c_{2i},c_{2i+1})\\ &\cdot\mathrm{P}_{S_i\to F_i}(s_i)\cdot\mathrm{P}_{S_{i+1}\to F_i}(s_{i+1})\\ &\cdot\mathrm{P}_{X_i\to F_i}(x_i)\big],\end{aligned}\tag{22}$$

where edges $(S_i, F_i), (S_{i+1}, F_i) \in \mathcal{G}$.

The PMF propagated from G_i to X_i is given by

$$\begin{aligned}\mathrm{P}_{G_i\to X_i}(x_i) = A\sum_{\sim x_i}\Big[&\delta\Big(\Big(\sum_{(B_k,G_i)\in\mathcal{G}} b_k\Big) \bmod 2 - x_i\Big)\\ &\cdot\prod_{(B_k,G_i)\in\mathcal{G}}\mathrm{P}_{B_k\to G_i}(b_k)\Big].\end{aligned}\tag{23}$$

The PMF propagated from G_j to B_i is given by

$$\begin{aligned}\mathrm{P}_{G_j\to B_i}(b_i) = A\sum_{\sim b_i}\Big[&\delta\Big(\Big(\sum_{(B_k,G_j)\in\mathcal{G}} b_k\Big) \bmod 2 - x_j\Big)\\ &\cdot\prod_{\substack{(B_k,G_j)\in\mathcal{G}\\ k\neq i}}\mathrm{P}_{B_k\to G_j}(b_k)\cdot\mathrm{P}_{X_j\to G_j}(x_j)\Big].\end{aligned}\tag{24}$$

6.3. Executing IntBP for LTRepconv

As we mentioned above, LTConv for payload and FCS and LTRepConv for information bits within metadata can be decoded by IntBP in similar manners. What we should do is replace the variable node message (10) with the following function

$$\begin{aligned}\mathrm{P}_{C_i\to F_i}(c_{2i},c_{2i+1}) = A\cdot&\mathrm{P}(y_{2i}|c_{2i})\cdot\mathrm{P}(y_{2i+1}|c_{2i+1})\\ &\mathrm{P}(y_{2i+508}|c_{2i})\cdot\mathrm{P}(y_{2i+509}|c_{2i+1})\\ &\mathrm{P}(y_{2i+1016}|c_{2i})\cdot\mathrm{P}(y_{2i+1017}|c_{2i+1}),\end{aligned}\tag{25}$$

and execute IntBP on the factor-graph shown in Fig. 11. Here y_{2i}, y_{2i+1}, y_{2i+508}, y_{2i+509}, $y_{2i+1016}$, and $y_{2i+1017}$ $(0 \le i < 254)$ denote the $2i$-th and $2i+1$-th channel output symbols corresponding to the first, the second, and the third 254-bit LT codes codeword copy for the K information bits within metadata, respectively.

6.4. Executing IntBP for LTConv in an incremental manner

Fig. 12 illustrates IntBP for LTConv executed in an incremental manner in Rateless802.11 with Advanced-Setup. As the figure illustrates, upon the reception of newly arrived channel output symbol y_{2i+2} and y_{2i+3}, new nodes are added to build the factor-graph first, and then messages are propagated as ' → ' shows between nodes[2] Let $\mathcal{G}$ denote the factor-graph built during decoding. The detailed decoding process of IntBP for LTConv in an incremental manner is presented in Algorithm 2.

Algorithm 2 IntBP for LTConv in an incremental manner.

Input: **e** and $y_0, y_1, \cdots, y_{2i}, y_{2i+1}, \cdots$
Output: $b_0^*, \cdots, b_{K-1}^*$
1: add $B_0, \cdots, B_{K-1}$ into the empty factor-graph $\mathcal{G}$
2: **while true do**
3: **if** y_{2i}, y_{2i+1} are received **then**
4: add F_i, G_i, their adjacent X_i, C_i, S_i, S_{i+1}, and their edges into $\mathcal{G}$
5: let MaxIterationNum = 1, take $y_0, y_1, \cdots, y_{2i}, y_{2i+1}$, $\mathcal{G}$, **e** as input, execute Algorithm 1
6: **end if**
7: **end while**

7. Analysis of complexity and delay

7.1. Complexity of transmitting process

We can find that the computational complexity of the transmitting process of Rateless802.11 is dominated by the encoding process of LT codes for the sourceword consisting of payload and FCS (c.f. the packet structure plotted in Fig. 4). In our proposal, the LT codes check node degree distribution (1) is taken from RFC 5053 section 5.4.4.2 [37], where the largest degree $d = 41$, i.e., a bounded integer constant. That is, the computational complexity of generating each individual parity bit of LT codes codeword in Section 4.2 is $\mathcal{O}(1)$, and the computational complexity of the encoding process is given by

$$\mathcal{O}(N-K),\tag{26}$$

[2] In real setup, PPDUs are transmitted in the unit of OFDM symbol, each of which usually contains 48 bits for BPSK at 2.4 GHz. That is at least 48 channel output symbols can be moved in during each iteration for Advanced-Setup.

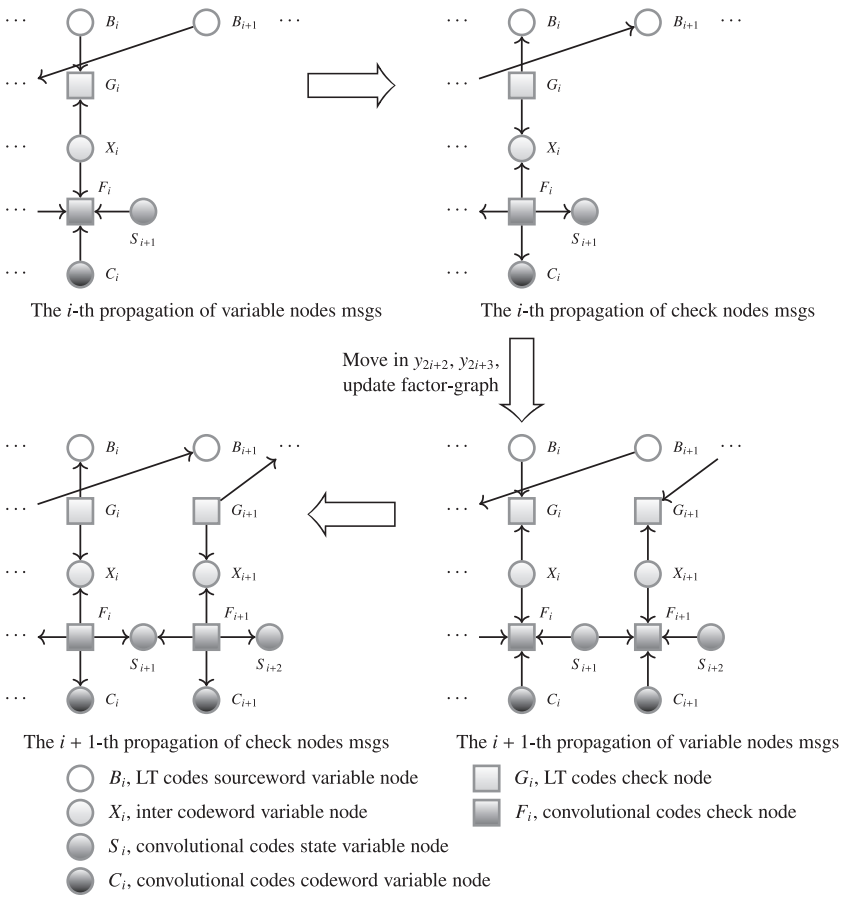


Fig. 12. IntBP for LTConv in an incremental manner.

where K is the length of sourceword (i.e., payload and FCS), and N is the length of inter codeword, in the unit of bits. It is much smaller compared with the $\mathcal{O}(KM)$ computational complexity of the encoding process in the state-of-the-art solution [7] (noted as RS codes scheme), where the bitwise bit-error-ratio of MPDU is estimated at the transmitter side, and payload bits are provided with additional unequal protection using Reed-Solomon (RS) codes. Here M denotes the length of each RS codes codeword, in the unit of symbols in finite field GF(256).

7.2. Complexity of receiving process

Given the fact that the computational complexity of using the convolutional codes encoder to obtain pseudo-PPDUs is $\mathcal{O}(K)$, we have the conclusion that for both Regular-Setup and Advanced-Setup, the computational complexity of the receiving process is dominated by IntBP for LTConv, which is introduced in detail in Algorithm 1, or Algorithm 2 if it is executed in an incremental manner.

Suppose after receiving $2N$ channel output symbols, Algorithm 1 and 2 terminate, i.e., the inter codeword has a length of N. During each of their iterations, the computational complexity of obtaining all messages given by equation (10) (13) (14) (15) (16) is $\mathcal{O}(N)$. The computational complexity of obtaining all messages given by (17) is $\mathcal{O}\big(N(N-K)\big)$, since the number of edges between all B_i nodes and all G_i nodes in the factor-graph shown in Fig. 10 is at most $\mathcal{O}(N)$, each with at most $N-K$ neighbor edges. The computational complexity of obtaining all messages given by equation (18) (20) (21) (22) (23) (24) is $\mathcal{O}(N)$. The computational complexity of obtaining all b_i^* based on (9) is $\mathcal{O}\big(K(N-K+1)\big)$, since the sourceword has a length of K, and each node B_i has at most $N-K+1$ neighbor nodes. The computational complexity of verifying the CRC32 of a length K sourceword is $\mathcal{O}(K)$. As a conclusion, the computational complexity of each iteration is $\mathcal{O}\big(N(N-K)\big)$. Suppose such $2N$ channel output symbols belong to N/K different PPDUs, the total computational complexity of IntBP and that executed in an incremental manner is given by

$$\begin{cases} \mathcal{O}\big(CN^2(N-K)/K\big), \\ \mathcal{O}\big(N^2(N-K)\big), \end{cases} \tag{27}$$

respectively, where C is determined by the MaxIterationNum specified in Algorithm 1. It is acceptable compared with the $\mathcal{O}(NM^2)$ computational complexity of the receiving process in the RS codes scheme, where RS codes codeword is usually decoded by methods like Berlekamp-Welch algorithm [38]. It is also acceptable compared with the computational complexity of natural serial decoding approaches, i.e., BP-BP and VD-BP, characterized by $\mathcal{O}\big(CN^2(N-K)/K\big)$ as well. Here VD-BP and BP-BP denote serial decoding approaches, where convolutional codes are decoded first by the Viterbi decoder or the belief-propagation (BP) based decoder [23], and their results are fed into the belief-propagation based LT codes decoder for further decoding. Note that BP-BP imposes the same requirement to 802.11 devices as Rateless802.11 with Advanced-Setup.

7.3. Delay analysis

Here we present a brief study of all the major time cost of Rateless802.11 components and the delay incurred by them. Specifically, from the top to the bottom of Fig. 13, we give an illustration of the possible time cost of different decoding approaches (including BP-BP, VD-BP, IntBP in Regular-Setup, IntBP in Advanced-Setup, IntBP executed in an incremental manner in Advanced-Setup), the time cost of transmitting each PPDU, and that of encoding each PPDU. Considering 802.11 usually has a limited coverage range ($\leq$ 300 m) [17] [18], we ignore the signal propagation time (≤ 1 μs) from the transmitter to the receiver in Fig. 13, since it is much smaller compared with the length of

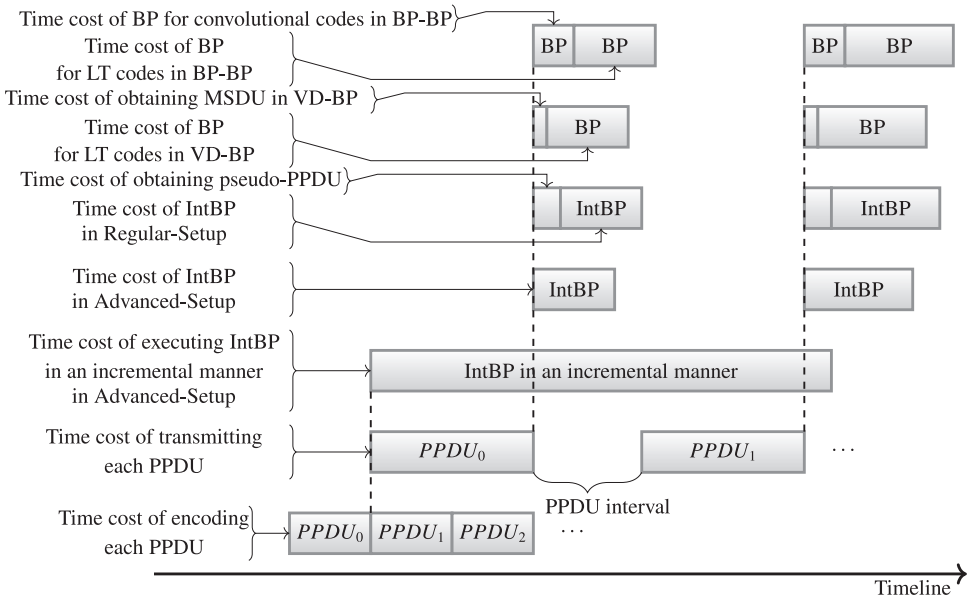


Fig. 13. Major time cost of Rateless802.11 components.

1 OFDM symbol (4 μs in 20 MHz). We also ignore the time cost of data copy within memory incurred by the communication between the link control component of Rateless802.11 and the network-layer, since the memory bandwidth can usually be up to several gigabytes per second.

We can find that (i) when the channel can be handled well by conventional 802.11, the additional delay incurred by our proposal comes from the time cost of encoding the first PPDU and that of decoding the concatenated codes; (ii) when the channel is rather poor that conventional 802.11 cannot handle, the additional delay incurred by our proposal also comes from the transmission time of subsequent PPDUs and the interval between them, if the decoding time cost does not exceed the summation of transmission time and the interval; (iii) otherwise, the additional delay of Rateless802.11 is dominated by the decoding time after receiving each PPDU.

Moreover, we can find that compared with serial decoding approaches for the concatenated codes characterized by (4), IntBP reduces the two-step optimization based decoding pipeline into a single-step optimization, where messages propagated in each of its iterations shown in Algorithm 1 can be obtained completely through parallel calculation. That is, in Rateless802.11 with Advanced-Setup, compared with BP-BP and VD-BP, it offers an opportunity to reduce the decoding time cost and additional decoding delay of the concatenated codes, if adequate computing resources can be provided for parallel computing, e.g., the FPGA chips in WARP [34] and GPU chipsets on ordinary devices. Besides, in Rateless802.11 with Advanced-Setup where adequate parallel computing resources can be provided, executing IntBP in an incremental manner as the channel output symbols are received would further reduce the additional decoding delay, even though it may have a longer decoding time cost.

8. Performance evaluation of Rateless802.11

8.1. Basic settings

In this section, we present both numerical and real-trace driven simulations for Rateless802.11. The bandwidth of 802.11 is set as 20 MHz and the interval between PPDUs is set as 10 μs, i.e., the length of short-interframe-space (SIFS) in 802.11g. The coding rate of convolutional codes is set as 1/2 and the modulation scheme is set as BPSK. The payload and FCS in PPDU_0 have 6048 bits in total. Each metadata consists 127 information bits which are encoded into 254-bit LT codes codeword first and then repeated 3 times. The LT codes check node degree distribution is taken from the aforementioned specification RFC5053 [37]. Given these settings, the time cost of encoding each PPDU with 6048 parity bits by a single thread on Dell PowerEdge R740 is around 0.35 ms, and the time cost of transmitting each of them is around 1.23 ms.

8.2. Numerical simulations

8.2.1. Recovery of scrambler seed

We first evaluate our scrambler seed protection method. For BSC with different crossover probability ϵ, we compare the error ratio of recovering the valid scrambler seed in Rateless802.11 with Regular-Setup and Advanced-Setup proposed in Section 5.3 and Section 5.4, respectively, and the error ratio of recovering the service field scrambler seed in conventional 802.11 by the Viterbi decoder. The comparison results are shown in Fig. 14. It is easy to find that our scrambler seed protection method can work very well even in extremely poor channels which conventional 802.11 cannot handle.

8.2.2. Recovery of information bits within metadata

Then we evaluate the performance of protecting and recovering the 127 information bits within metadata in Rateless802.11. For BSC with different crossover probability ϵ, we compare the error ratio of recovering such 127 bits in Rateless802.11 with Regular-Setup and Advanced-Setup proposed in Section 6.3, and the error ratio of recovering any 127 bits in conventional 802.11 by the Viterbi decoder. The comparison results are shown in Fig. 15 where errors introduced by the descrambling process are not counted. Still, our proposal can work well even in those extremely poor channels.

8.2.3. Obtainment of pseudo-PPDUs

To give a sense of the fact that feeding pseudo-PPDUs into IntBP helps improve the decoding performance in Rateless802.11 with Regular-Setup, in Fig. 16, we list the BER of these pseudo-PPDUs generated by the convolutional codes encoder in BSC with different crossover probability. From this figure, we can find that when the crossover probability $\epsilon \leq 0.125$, the BER of LTConv codeword in pseudo-PPDUs is always smaller than that of the received PPDUs, i.e., BER shrinks in these

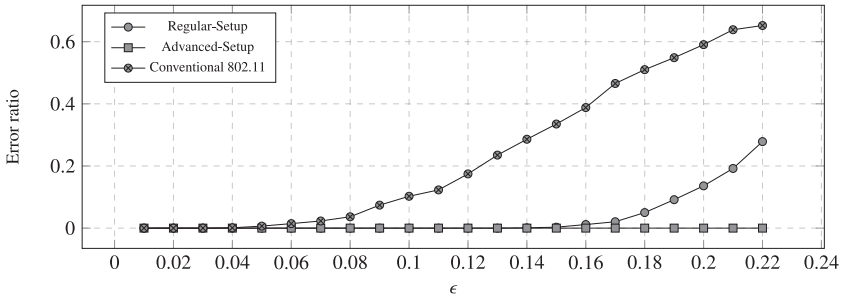


Fig. 14. Error ratio of recovering scrambler seed in BSC.

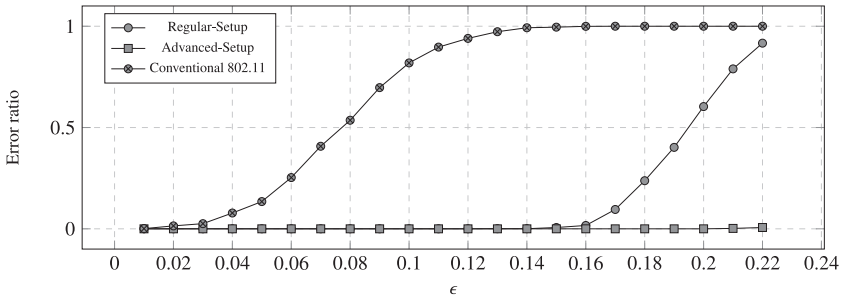


Fig. 15. Error ratio of recovering information bits within metadata in BSC.

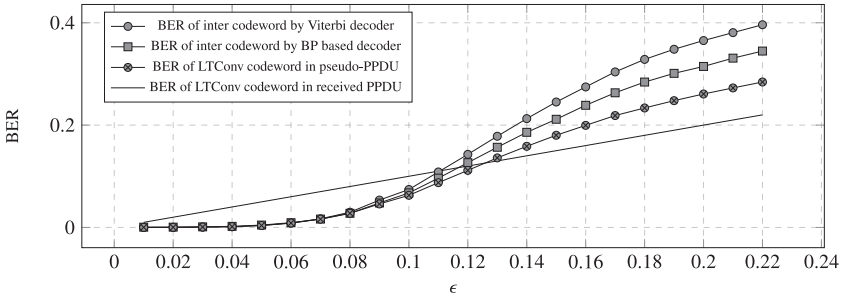


Fig. 16. BER of inter codeword decoded by Viterbi decoder, by the BP based decoder, and BER of LTConv codeword in pseudo-PPDU generated by convolutional codes encoder.

pseudo-PPDUs. In addition, when $\epsilon \geq 0.09$, we can find that the BER in pseudo-PPDUs is also much smaller than that of inter codeword decoded by both the Viterbi decoder and the belief-propagation based decoder, i.e., the input of our IntBP has less errors compared with that of the decoder for LT codes in the natural serial decoding approaches. These two factors can help improve the error correction capability of our proposal in all kinds of channel conditions.

8.2.4. Rateless802.11 vs. RS codes scheme

Finally, we compare the performance of Rateless802.11 with the RS codes scheme, the state-of-the-art solution presented in [7], which operates in finite field GF(256). Given the fact that in Rateless802.11, IntBP and IntBP executed in an incremental manner do not make a significant difference to the simulation results (except the decoding time cost), we omit their difference in the performance evaluations.

Specifically, Fig. 17(a) and Fig. 17(b) show the packet-error-ratio (PER) of Rateless802.11 and the RS codes scheme in BSC with different crossover probability ϵ, which the least coding rate of conventional 802.11 cannot handle. For example, according to our experiments, when $\epsilon = 0.0417$, its PER≐89.0%, and when $\epsilon \geq 0.06$, its PER≐100.0%. In these figures, R^{-1} denotes the reciprocal of total coding rate R for payload, and for figures plotting PER, neither errors introduced by the descrambling process nor the errors resulted by failing to recover metadata are counted for all schemes.

Fig. 18 (a) shows their corresponding throughput of payload, while the minimum coding rate of payload is set around 1/24. Here for $i = 1, 2, 3, 4, \cdots$ PPDU_i of Rateless802.11 contains 1728, 3456, 5184, 5184, $\cdots$ parity bits, and the coding rate of payload in the RS codes scheme is simply the one providing the greatest throughput. Fig. 18(b) shows their throughput while the scrambler seed protection method is not introduced. Additionally, to be fair to the RS codes scheme which does not provide additional protection to its metadata appended to MAC header, we also plot Fig. 18(c) where we assume that all schemes can recover their metadata well and do not count their overhead. Note that for figures plotting throughput, we assume the incurred additional decoding delay of all schemes is shorter than the interval between PPDUs.

We can find that for small ϵ, all schemes work well and the RS codes scheme is slightly better. If channels become worse, our proposal works much better. For example, for extremely poor channel whose $\epsilon = 0.11$, Rateless802.11 can incrementally approach reliable communication below coding rate 1/8 for both setups, while that of the RS codes scheme is far below 1/24, and in terms of throughput, the improvement is larger than 90 times. In addition, by comparing Fig. 18(a) and Fig. 18(b), we also find that when $\epsilon \geq 0.1461$, the scrambler seed protection method can help improve the throughput by more than 50%. It is also worth noting that by carefully choosing the payload size and parity bits size in each PPDU to reduce overhead, throughput of Rateless802.11 can be further improved.

8.3. Real-Trace driven evaluations

8.3.1. Capture of real-trace

Here we compare the performance of IntBP with BP-BP and VD-BP for Rateless802.11 with real-trace. The minimum coding rate of payload is set as 1/32 and each PPDU_i ($i \geq 1$) contains 6048 parity bits. To capture the baseband RF signal samples of Rateless802.11 during a relative

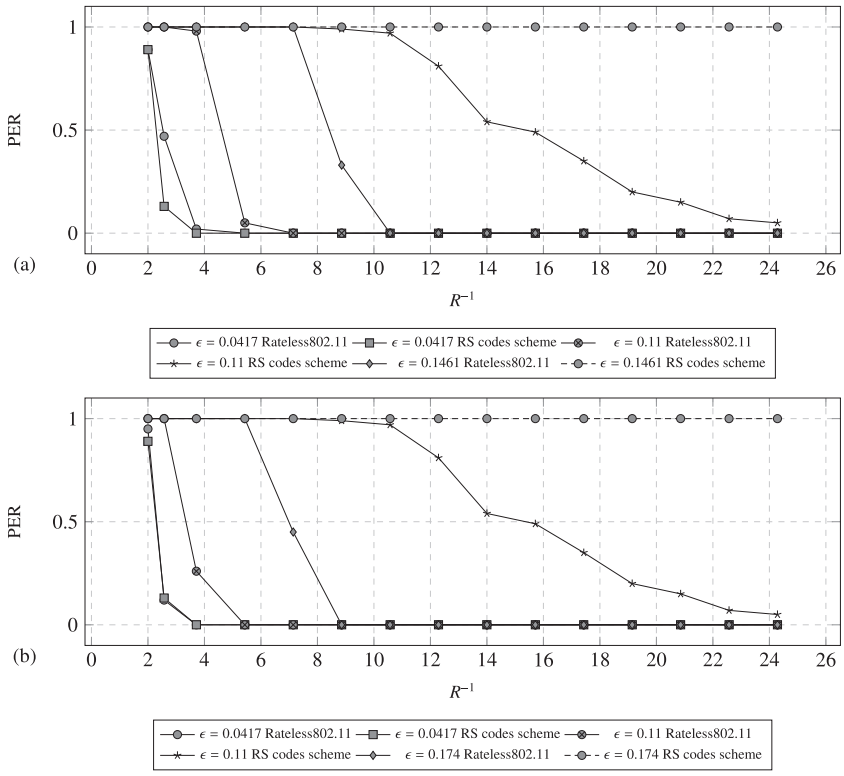


Fig. 17. In BSC with different crossover probability ϵ, we list the PER of (a) Rateless802.11 with Regular-Setup and the RS codes scheme; (b) Rateless802.11 with Advanced-Setup and the RS codes scheme .

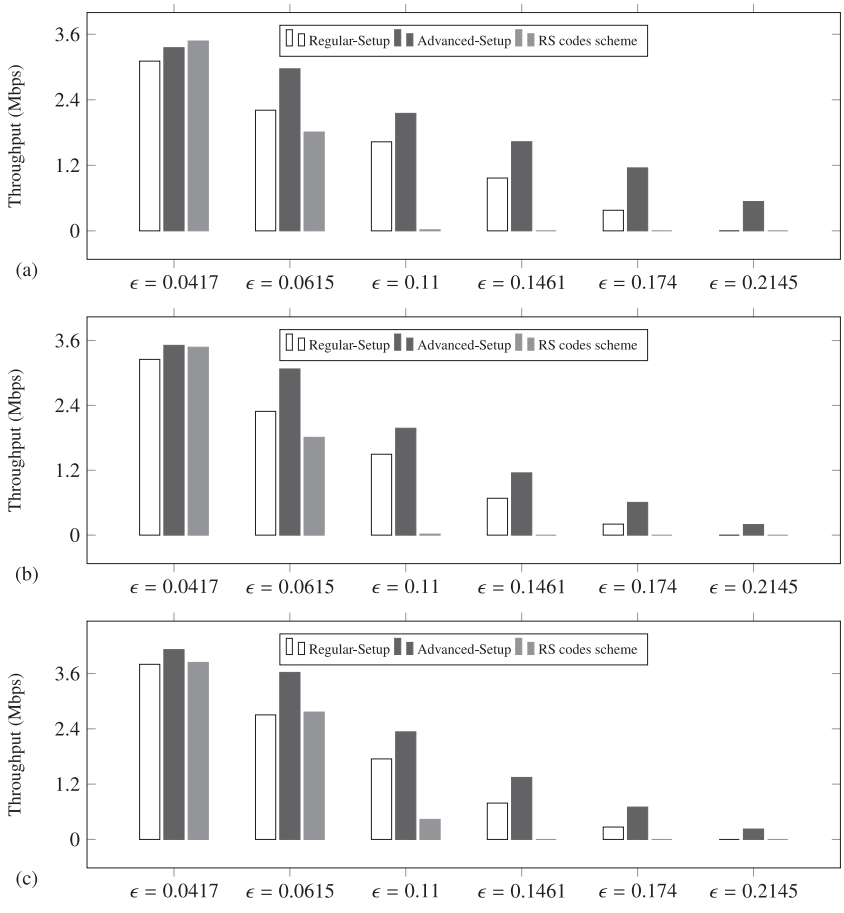


Fig. 18. In BSC with different crossover probability ϵ, we list the throughput of (a) Rateless802.11 and the RS codes scheme; (b) Rateless802.11 and the RS codes scheme, where scrambler seed protection is not applied; (c) Rateless802.11 and the RS codes scheme, where scrambler seed protection is not applied and decoding errors of metadata is not counted .

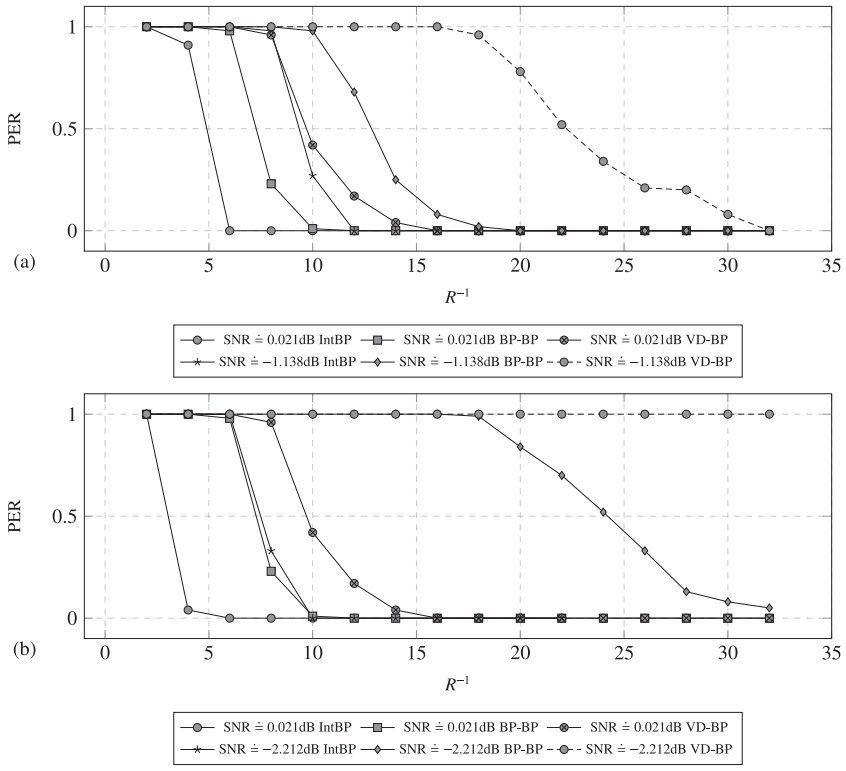


Fig. 19. Driven by real-trace, we list the PER of (a) IntBP (Regular-Setup), BP-BP, and VD-BP; (b) IntBP (Advanced-Setup), BP-BP, and VD-BP .

static channel condition and avoid the influence introduced by carrier frequency offset, sampling frequency offset, noises of other transmitters and environment, we connect the TX port and RX port of a single universal software radio peripheral (USRP) device—N210—together with a cable. Through the corresponded TX path and RX path of the USRP, we send and receive the baseband samples of Rateless802.11 simultaneously. By carefully tuning the multiplication constant of the baseband samples to send, we can control the SNR of the channel.

8.3.2. IntBP vs. VD-BP, and BP-BP

Fig. 19 (a), Fig. 19(b), Fig. 20(a), Fig. 20(b), and Fig. 20(c) show the performance of IntBP, BP-BP, and VD-BP in extremely poor channels that the least coding rate of 802.11 cannot handle. For example, according to our experiments, when SNR $\leq$ 0.021 dB, its PER≐100.0%. It is straightforward to see that IntBP dramatically improves the decoding reliability, especially for the Advanced-Setup. Also, the scrambler seed protection method can help improve throughput by more than 50% when SNR ≤ -1.138 dB.

8.3.3. Spectral efficiency

We also list the spectral efficiency of our proposal in Fig. 21(a) and Fig. 21(b), corresponding to the throughput shown in Fig. 20(a), Fig. 20(b), and Fig. 20(c). The spectral efficiency is defined as

$$\frac{\text{throughput of payload}}{\text{Shannon capacity}}, \tag{28}$$

where the Shannon capacity of the 20 MHz channel is 23.492 Mbps, 18.785 Mbps, 15.160 Mbps, and 10.862 Mbps, when the SNR≐0.021 dB, −1.138 dB, −2.212 dB, and −3.828 dB, respectively. From Fig. 21(a) and Fig. 21(b), we can find that for extremely poor channels whose SNR $\leq$ 0.021 dB, the spectral efficiency of our proposal can be up to 7.2% and 10.2% in Regular-Setup and Advanced-Setup, respectively. In contrast, according to our experiments, conventional 802.11 with least coding rate can only well handle channels whose SNR $\geq$ 5.849 dB (i.e., the Shannon capacity of the 20 MHz channel is greater than 61.888 Mbps), where the spectral efficiency is at most 9.1% (i.e., the throughput equals 5.605 Mbps).

8.3.4. Time cost of decoding

To give a sense that the decoding time cost can be reduced by decoding the concatenated codes in a joint manner when adequate parallel computing resources are provided, we compare that of IntBP with BP-BP and VD-BP in Fig. 22. Given the fact that the time cost of obtaining each pseudo-PPDU is very short ($\leq$ 0.3 ms), here we omit the difference of IntBP in Regular-Setup and Advanced-Setup. We take Dell PowerEdge R740, which is equipped with two 2nd generation Intel(R) Xeon(R) scalable processors with up to 28 cores per processor, to simulate these platforms where adequate parallel computing resources can be provided. Specifically, in VD-BP, for each PPDU, we allocate 1 thread (i.e., 1 core) to decode its convolutional codes using the Viterbi decoder, and $R^{-1}/2$ threads to decode the LT codes in parallel, when the coding rate of the concatenated codes is R. In BP-BP, for each PPDU, we allocate 1 thread to decode its convolutional codes using the belief-propagation based decoder, and $R^{-1}/2$ threads to decode the LT codes as well. In IntBP, for each PPDU, we allocate 1 thread to calculate the messages related to its convolutional codes, and $R^{-1}/2$ threads to calculate the messages related to the LT codes in parallel. We can find that the decoding time cost of IntBP is always much shorter than that of BP-BP, which is consistent with our expectation since IntBP reduces the two-step optimization based decoding pipeline into a single-step. Additionally, when $R \geq 12$, we can find that the time cost of IntBP is almost the same as that of VD-BP. This is because the time cost of Viterbi decoder in VD-BP is relatively short (around 42 ms), and the decoding time of IntBP is dominated by that of obtaining messages related to LT codes, which is almost equal to that of decoding LT codes in VD-BP.

8.4. Extension to 802.11 with LDPC codes

In Fig. 23, we also list some simulation results of our proposal applied in 802.11 devices using LDPC codes for error correction. Specifically, we let the payload and FCS have 5832 bits in total. Each codeword of the LDPC codes has 648 bits, and the coding rate is set as 1/2, i.e., its least coding rate. We compare the PER of IntBP and BP-BP in BSC with different crossover probability ϵ, which the 1/2 LDPC

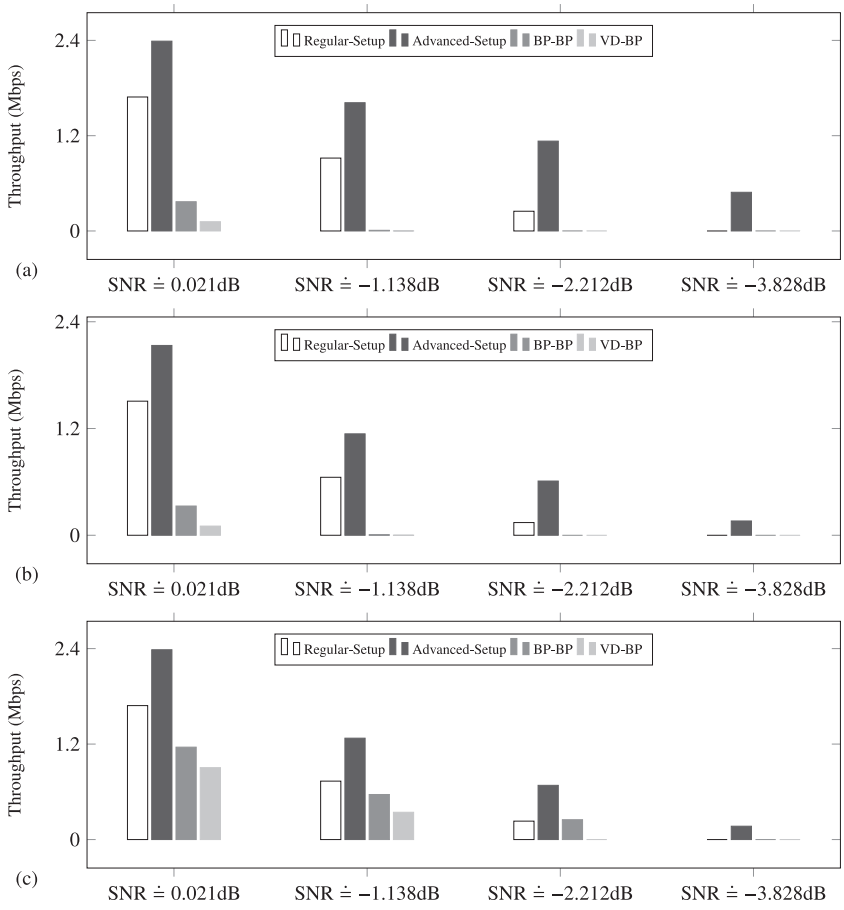


Fig. 20. Driven by real-trace, we list the throughput of (a) IntBP, BP-BP, and VD-BP; (b) IntBP, BP-BP, and VD-BP, where scrambler seed protection is not applied; (c) IntBP, BP-BP, and VD-BP, where scrambler seed protection is not applied, and decoding errors of metadata are not counted .

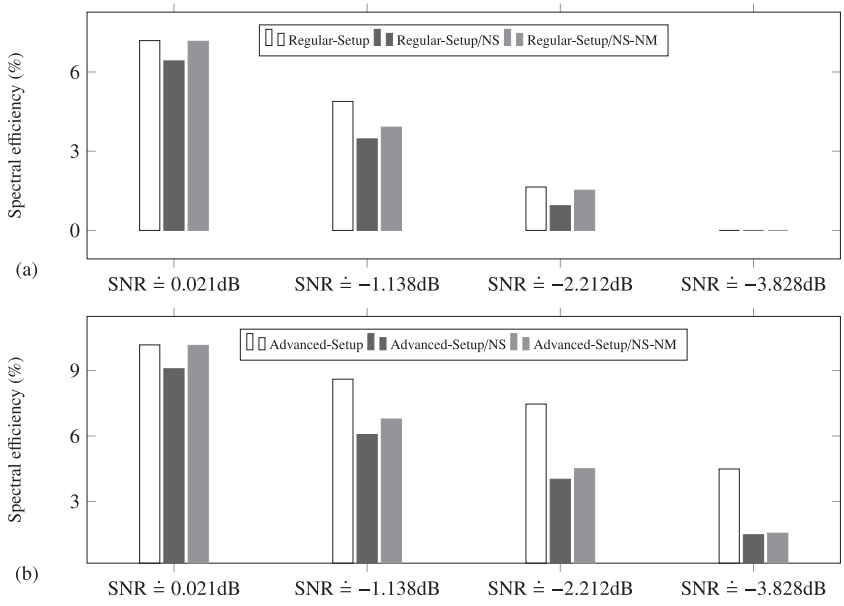


Fig. 21. Driven by real-trace, we list the spectral efficiency of (a) Regular-Setup, Regular-Setup where scrambler seed protection is not applied (denoted by Regular-Setup/NS), and Regular-Setup where scrambler seed protection is not applied and decoding errors of metadata are not counted (denoted by Regular-Setup/NS-NM); (b) Advanced-Setup, Advanced-Setup where scrambler seed protection is not applied (denoted by Regular-Setup/NS), and Advanced-Setup where scrambler seed protection is not applied and decoding errors of metadata are not counted (denoted by Regular-Setup/NS-NM); .

codes cannot handle. For example, according to our experiments, when $\epsilon \geq 0.12$, its PER≐100.0%. In this figure, BP-BP denotes the serial decoding approach, where the LDPC codes of 802.11 are decoded first by the BP based decoder, and its results are fed into the BP based LT codes decoder for further decoding. In our proposal, IntBP decodes the concatenated codes consisting of LT codes and LDPC codes in a joint manner, by iteratively feeding their decoding results to each other. We can find that our proposal works very well in these extremely poor channels. For example, when $\epsilon = 0.1461$, Rateless802.11 can incrementally approach reliable communication below coding rate 1/6, while that of BP-BP is far below 1/12. We can also find that concatenating LT codes with LDPC codes works better than concatenating LT codes with convolutional codes. For example, when $\epsilon = 0.2145$, concatenating LT codes with LDPC codes can incrementally approach reliable communication below coding rate 1/10, while that of concatenating LT codes with convolutional codes is below 1/16.

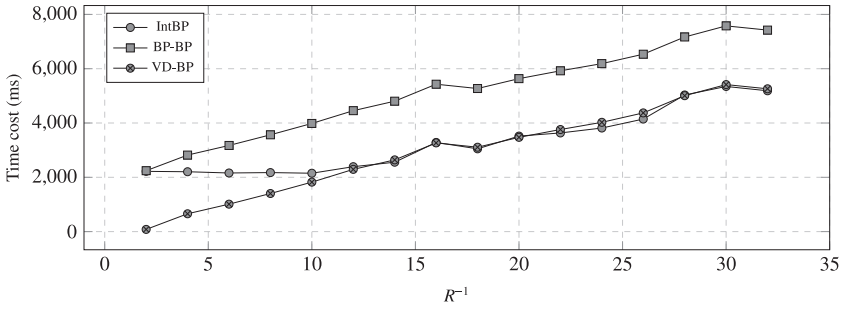


Fig. 22. Time cost of IntBP, BP-BP, and VD-BP for different coding rate R.

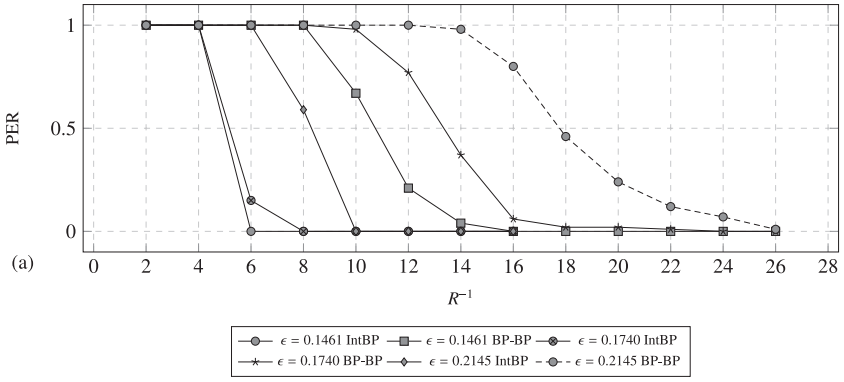


Fig. 23. In BSC with different crossover probability ϵ, we list the PER of IntBP (Advanced-Setup) and BP-BP, when 802.11 devices are using LDPC codes for error correction .

9. Conclusion

In this paper, we propose Rateless802.11, a middleware to extend 802.11 applicability in extremely poor channels. Rateless802.11 is compatible with common commodity 802.11 devices, which employs LT codes as outer codes, and adopts a well-designed processing flow to correct various errors introduced in the receiving path, such that additional redundancies can be incrementally introduced without accurate channel estimation, and fully provide their error correction capability. Both numerical and real-trace driven simulations show that Rateless802.11 dramatically improves the throughput of 802.11 in extremely poor channels compared with state-of-the-art solutions. In the future, we would like to evaluate our proposal in more complicated channels and try to further optimize its encoding strategy.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Tao Huang: Conceptualization, Methodology, Formal analysis, Software, Writing - original draft. **Bin Tang:** Supervision, Methodology, Writing - original draft. **Baoliu Ye:** Supervision, Writing - review & editing. **Zhihao Qu:** Validation, Writing - review & editing. **Sanglu Lu:** Resources, Project administration.

Acknowledgment

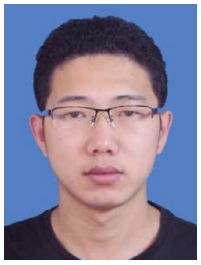
This work was supported in part by National Key R&D Program of China under Grant 2018YFB1004704, in part by the National Natural Science Foundation of China under Grant 61832005 and Grant 61872171, in part by the Key R&D Program of Jiangsu Province under Grant BE2017152, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20190058, the Science and Technology Program of State Grid Corporation of China under Grant SGSHXT00JFJS1900092, and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] T. Huang, B. Tang, B. Ye, S. Lu, Rateless802.11: Architecture design and performance optimization, in: Proc. IEEE Wireless Communications and Networking Conference, WCNC, 2019.
[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, M. Gidlund, Industrial internet of things: challenges, opportunities, and directions, IEEE Trans. Industrial Informatics 14 (11) (2018) 4724–4734.
[3] K.C.-J. Lin, N. Kushman, D. Katabi, ZipTx: Harnessing partial packets in 802.11 networks, in: Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM, 2008, pp. 351–362.
[4] B. Han, A. Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, R.R. Miller, Maranello: Practical partial packet recovery for 802.11, in: Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI, 2010, pp. 205–218.
[5] J. Xie, W. Hu, Z. Zhang, Revisiting partial packet recovery in 802.11 wireless lans, in: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys, 2011, pp. 281–292.
[6] S. Choi, Y. Choi, I. Lee, IEEE 802.11 Mac-level FEC scheme with retransmission combining, IEEE Trans. Wireless Communications 5 (1) (2006) 203–211.
[7] Y. Xie, Z. Li, M. Li, K. Jamieson, Augmenting wide-band 802.11 transmissions via unequal packet bit protection, in: 35th Annual IEEE International Conference on Computer Communications, INFOCOM, 2016, pp. 1–9.
[8] M.O. Khan, L. Qiu, Accurate wifi packet delivery rate estimation and applications, in: 35th Annual IEEE International Conference on Computer Communications, INFOCOM, 2016, pp. 1–9.
[9] Z. Li, Y. Xie, M. Li, K. Jamieson, Recitation: Rehearsing wireless packet reception in software, in: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MOBICOM, 2015, pp. 291–303.
[10] Z. Li, T. He, Webee: Physical-layer cross-technology communication via emulation, in: Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MOBICOM, 2017, pp. 2–14.
[11] A. Bhartia, Y. Chen, S. Rallapalli, L. Qiu, Harnessing frequency diversity in wi-fi networks, in: Proceedings of the 17th Annual International Conference on Mobile Computing and Networking, MOBICOM, 2011, pp. 253–264.
[12] S. Wang, S.M. Kim, T. He, Symbol-level cross-technology communication via payload encoding, in: Proceedings of the 38th IEEE International Conference on Distributed Computing Systems, ICDCS, 2018, pp. 500–510.
[13] X. Zheng, Y. He, X. Guo, Stripcomm: Interference-resilient cross-technology communication in coexisting environments, in: Proceedings of the 2018 IEEE Conference on Computer Communications, INFOCOM, 2018, pp. 171–179.

[14] X. Guo, Y. He, X. Zheng, Z. Yu, Y. Liu, Lego-fi: Transmitter-transparent CTC with cross-demapping, in: Proceedings of the 2019 IEEE Conference on Computer Communications, INFOCOM, 2019, pp. 2125–2133.
[15] M. Luby, LT codes, in: Proceedings of the 43rd Symposium on Foundations of Computer Science, FOCS, 2002, p. 271.
[16] R.G. Gallager, Low-density parity-check codes, IRE Trans. Information Theory 8 (1) (1962) 21–28.
[17] O. Bejarano, E.W. Knightly, M. Park, IEEE 802.11Ac: from channelization to multi-user MIMO, IEEE Commun. Mag. 51 (10) (2013) 84–90.
[18] E.M. Khorov, A. Kiryanov, A.I. Lyakhov, G. Bianchi, A tutorial on IEEE 802.11ax high efficiency wlans, IEEE Commun. Surv. Tutorials 21 (1) (2019) 197–216.
[19] T. Nitsche, C. Cordeiro, A.B. Flores, E.W. Knightly, E. Perahia, J. Widmer, IEEE 802.11Ad: directional 60 ghz communication for multi-gigabit-per-second wi-fi [invited paper], IEEE Commun. Mag. 52 (12) (2014) 132–141.
[20] A.B. Flores, R.E. Guerra, E.W. Knightly, P. Ecclesine, S. Pandey, IEEE 802.11Af: a standard for TV white space spectrum sharing, IEEE Commun. Mag. 51 (10) (2013) 92–100.
[21] C. Chen, C. Lin, Y. Chen, Unequal error protection for video streaming over wireless lans using content-aware packet retry limit, in: Proceedings of the 2006 IEEE International Conference on Multimedia and Expo, ICME, 2006, pp. 1961–1964.
[22] M.A. Khan, A.A. Moinuddin, E. Khan, M. Ghanbari, Optimized cross-layered unequal error protection for SPIHT coded wireless video transmission, 62, 2016, pp. 876–889.
[23] M. Vutukuru, H. Balakrishnan, K. Jamieson, Cross-layer wireless bit rate adaptation, in: Proceedings of the ACM SIGCOMM 2009 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 3–14.
[24] D. Halperin, W. Hu, A. Sheth, D. Wetherall, Predictable 802.11 packet delivery from wireless channel measurements, in: Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2010, pp. 159–170.
[25] B. Chen, Z. Zhou, Y. Zhao, H. Yu, Efficient error estimating coding: Feasibility and applications, in: Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2010, pp. 3–14.
[26] H. Lu, W. Gao, Continuous wireless link rates for internet of things, in: Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN, 2018, pp. 48–59.
[27] A. Bhartia, Y. Chen, S. Rallapalli, L. Qiu, Harnessing frequency diversity in wi-fi networks, in: Proceedings of the 17th Annual International Conference on Mobile Computing and Networking, MOBICOM, 2011, pp. 253–264.
[28] H. Rahul, F. Edalat, D. Katabi, C. Sodini, Frequency-aware rate adaptation and MAC protocols, in: Proceedings of the 15th Annual International Conference on Mobile Computing and Networking, MOBICOM, 2009, pp. 193–204.
[29] A. Shokrollahi, Raptor codes, IEEE Trans. Information Theory 52 (6) (2006) 2551–2567.
[30] D.J. MacKay, Fountain codes, IEE Proceedings-Communications 152 (6) (2005) 1062–1068.
[31] P.A. Iannucci, J. Perry, H. Balakrishnan, D. Shah, No symbol left behind: A link-layer protocol for rateless codes, in: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, MOBICOM, 2012, pp. 17–28.
[32] A. Gudipati, S. Katti, Strider: Automatic rate adaptation and collision handling, in: Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2011, pp. 158–169.
[33] J. Perry, P. Iannucci, K. Fleming, H. Balakrishnan, D. Shah, Spinal codes, in: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2012, pp. 49–60.
[34] P. Murphy, A. Sabharwal, B. Aazhang, Design of WARP: A wireless open-access research platform, in: Proceedings of the 14th European Signal Processing Conference, EUSIPCO, 2006, pp. 1–5.
[35] R. Palanki, J.S. Yedidia, Rateless codes on noisy channels, in: Proceedings of the 2004 IEEE International Symposium on Information Theory, ISIT, 2004, p. 37.
[36] F.R. Kschischang, B.J. Frey, H. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Information Theory 47 (2) (2001) 498–519.
[37] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, Raptor forward error correction scheme for object delivery, RFC 5053 (2007) 1–46.
[38] W. Ryan, S. Lin, Channel codes: classical and modern, Cambridge University Press, 2009, doi:10.1017/CBO9780511803253.

Tao Huang received his B.S. Degree in Computer Science and Technology and M.S. Degree in Computer Software and Theory from Nanjing University (NJU), China, in 2012 and 2015. He is now a Ph.D. candidate with Department of Computer Science and Technology, NJU. His current research interests include wireless communication in 802.11 and 5G NR, multiuser detection, and over-the-air computation.



Bin Tang received his B.S. and Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 2007, and 2014, respectively. He is currently an assistant researcher in the Department of Computer Science & Technology, Nanjing University, China. His current research interests lie in the area of communications, network coding, and distributed computing, with a focus on the application of network coding to data dissemination in various networking environments.

Baoliu Ye is a full professor at Department of Computer Science and Technology, Nanjing University. He received his Ph.D. in computer science from Nanjing University, China in 2004. He served as a visiting researcher of the University of Aizu, Japan from March 2005 to July 2006, and the Dean of School of Computer and Infomation, Hohai University since January 2018. His current research interests mainly include distributed systems, cloud computing, wireless networks with over 70 papers published in major conferences and journals. Prof. Ye served as the TPC co-chair of HotPOST12, HotPOST11, P2PNet10. He is the regent of CCF, the Secretary-General of CCF Technical Committee of Distributed Computing and Systems, and a member of IEEE.

Zhihao Qu received his B.S. and Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 2009, and 2018, respectively. He is currently an assistant researcher in the College of Computer and Information, Hohai University, China. His research interests are mainly in the areas of wireless networks, edge computing, and distributed machine learning.

SangLu Lu received her B.S., M.S., and Ph.D. degrees from Nanjing University in 1992, 1995, and 1997, respectively, all in computer science. She is currently a professor in the Department of Computer Science & Technology and the National Key Laboratory for Novel Software Technology. Her research interests include distributed computing, wireless networks and pervasive computing. She has published over 100 papers in referred journals and conferences in the above areas. She is a distinguished member of CCF and a member of IEEE.