Reliable Wireless Networking via Soft-Source Information Combining

Lihao Zhang© and Soung Chang Liew©, Fellow, IEEE

Abstract—This article puts forth a multistream networking paradigm, referred to as soft-source-information-combining (SSIC), to support wireless Internet of Things (IoT) applications with ultrareliability requirements. For SSIC networking, an SSIC dispatcher at the source dispatches duplicates of packets over multiple streams, which may be established over different physical wireless networks. If a packet on a stream cannot be decoded due to wireless interference or noise, the decoder makes available the packet’s soft information. An aggregator then combines the soft information of the duplicates to boost reliability. Of importance are two challenges: 1) how to deserialize the scrambled soft information from different streams to enable correct SSIC and 2) the construct of an SSIC dispatching and aggregation framework compatible with commercial network interface cards (NICs) and TCP/IP networks. To address the challenges, we put forth: 1) a soft descrambling (SD) method to minimize the bit-error rate (BER) and packet-error rate (PER) at the SSIC’s output and 2) an SSIC networking architecture readily deployable over today’s TCP/IP networks without specialized NICs. For concept proving and experimentation, we realized an SSIC system over two Wi-Fi’s physical paths in such a way that all legacy TCP/IP applications can enjoy the reliability brought forth by SSIC without modification. Experiments over our testbed corroborate the effectiveness of SSIC in lowering the packet delivery failure rate and the possibility of SSIC in providing 99.99% reliable packet delivery for short-range communication.

Index Terms—Multistream transmission, soft information combining, ultrareliable communications, Wi-Fi 7, wireless networks.

I. INTRODUCTION

Wireless networks are error prone and may fail to deliver packets to their destinations due to environmental interference and noise [1], [2]. Improving the reliability of wireless networks is essential for the support of mobile industrial, medical, audiovisual, and Internet of Things (IoT) applications with stringent quality-of-service requirements [3], [4], [5], [6], [7].

Many industrial IoT (IIoT) applications call for highly reliable communications [8]. A way to ensure reliability is via “time diversity”: if a message fails to be delivered, retransmit it. For a wireless network, however, the poor channel condition may persist for a while. For example, the wireless network may be busy due to access by many devices, the wireless path to the device may be blocked, or the device may move to a location far away from the access point (AP). Time diversity does not address the reliability problem in the above scenarios well.

“Parallel redundancy protocol (PRP)” [9] is an IEC 62439-3 industry standard originally targeted for Ethernet. PRP realizes network redundancy by dispatching duplicate packets over two independent networks. The same concept can be applied to wireless networks. For example, Cena et al. [10] investigated using PRP to enhance the reliability of Wi-Fi networks, whereby each Wi-Fi device is armed with two radio transceivers operating over different channels and bands.

The upcoming IEEE 802.11be ETH (Wi-Fi 7) standard aims to improve Wi-Fi reliability via multilink operation [11]. While the detailed formal standard will only be released in 2024, proponents in [12] have already begun to discuss the possibility of integrating IEEE 802.1CB, a standard akin to PRP, into the IEEE 802.11 standard. IEEE 802.1CB is also called a frame-replication-and-elimination scheme for reliability (FRER). With IEEE 802.1CB, a device can send duplicate frames via separate paths over a multihop network [13].

In the aforementioned duplicate schemes, duplicates do not undergo joint processing, foregoing a possible way to boost reliability further. Consider a setup in which a source sends two copies of a packet to a destination via two paths, as shown in Fig. 1. There are three possibilities as far as the reception at the destination is concerned.
1) **Case 1:** Both packets are decoded successfully, passing the cyclic redundancy check (CRC).

2) **Case 2:** Only one of the packets is decoded, passing the CRC.

3) **Case 3:** Both packets fail to be decoded and do not pass the CRC.

PRP-enabled Wi-Fi and the Wi-Fi 7 proposal work well in case 2, improving reliability over nonduplicate schemes. However, they do not help in case 3. Note that failing CRC does not mean having no information on the packet. Consider a packet of 1500 Bytes (the maximum transmission unit of Ethernet). Mis-decoding just one of the 8192 bits still leads to a CRC failure. As observed in [14], the percentage of partially correct packets can be up to 55% in 802.11 networks. Modern channel decoding methods often yield soft information on the source bits that give the log-likelihood ratio (LLR) of the probabilities of the bits being zero and one.

This article puts forth a source soft-information combining (SSIC) mechanism to exploit the soft information from the decoders. With SSIC, even if both packet copies fail to be decoded (case 3), combining the soft information of the two copies may lead to a successful decoding. In case 3 above, if the decoders of the two packets can expose the soft source information for joint signal processing, then there is a chance that the packet can be decoded. However, two challenges need to be addressed before SSIC can be deployed on today’s wireless networks over the TCP/IP networking platform.

**Challenge 1:** Directly combining the LLRs from the different streams does not work for common wireless systems, such as Wi-Fi and WiMAX. For systems, such as Wi-Fi and WiMAX, the source, before performing channel coding, scrambles the source bits by masking them (XORing them) with a random binary sequence to avoid long consecutive sequences of 0 or 1 s. Within a network interface controller (NIC), different random masking sequences are used to XOR successive packets. Also, for the duplicates of the same packet, two NICs at the source may mask the packet with different random sequences. The receiver needs to first decode the mask of each packet copy in order to obtain the unscrambled source bits. For the example in Fig. 1, the two packets may be scrambled with different masking sequences by the two NICs at the source. The LLRs of the source bits output from the two-channel decoders must first be descrambled with their respective scrambling masks before they can be combined. Conventionally, a decoder first converts the LLRs of the whole packets to 0 or 1 binary values, then hard decodes the masking sequence from the preamble part of the binary values. The masking sequence is then used to descramble the 0 or 1 values of the subsequent payload (Section II-B gives more details on the conventional descrambling). This conventional descrambling is not compatible with what we want to do because the channel decoder is essentially performing hard decoding, and soft information is lost. Although one may opt to hard decode the masking sequence only but not the source bits, as will be shown later in this article, the hard-decoded masking sequence is often erroneous, leading to wrong descrambling of the LLRs, and therefore the wrong combination of the LLRs of the two packets.

**Our Solution:** We devise a new descrambling method, referred to as soft descrambling (SD). First, SD allows the scrambled LLRs of the source bits to be descrambled without being converted to binary values. Hence, SD maintains the soft information required by SSIC. Second, SD also yields the “soft” masking sequence. Descrambling the LLRs using a soft masking sequence (as opposed to a hard masking sequence) may lead to better bit-error-rate (BER) and packet error rate (PER) performance in SSIC.

**Challenge 2:** Design of an SSIC dispatcher at the source that dispatches the packets over different streams and an SSIC aggregator that combines the packets from different streams into one stream in a way that is compatible with commercial NICs and TCP/IP networking. Ideally, SSIC should be deployable over today’s TCP/IP networks without the need for specialized NICs, and all unmodified TCP/IP applications should run over the SSIC-enabled network in a transparent manner. Also, other than the need to have multiple physical network paths, it would be desirable not to introduce additional hardware equipment into the system. That is, the overall SSIC networking solution should be compatible with the installed base of legacy networking equipment.

**Our Solution:** We design a TCP/IP-compatible networking called the SSIC networking architecture in which the SSIC dispatcher, SSIC aggregator, and SD are all implemented as middleware software running between the medium access control (MAC) layer and IP layer (i.e., a layer 2.5 solution). The middleware creates a virtual connection (VC) between the two end nodes over which the multiple physical paths are deployed. The middleware at the source encapulates an IP packet into a VC frame, and duplicates of the VC frame are sent over the multiple physical paths. At the receiver, the middleware that serves as the SSIC aggregator performs SD, SSIC, and deduplication (in case multiple copies of the same packet are decoded) before forwarding a single packet copy to the IP layer. Conceptually, with this solution, the IP layers at the two ends run over a “virtual network” created by the middleware. That is, SSIC networking allows multiple streams to be grouped and exposed as a single virtual link to the application for TCP/IP communication without the need to modify legacy TCP/IP applications.

We note that heterogeneous networking is also possible with the SSIC networking architecture. For example, the two physical paths can be over different types of physical networks (e.g., Wi-Fi and WiMAX).

To validate the performance of SSIC in a real environment, we set up an SSIC network over Wi-Fi. Experimental results show that SSIC significantly improves the packet delivery success rate in a noisy and lossy wireless environment. Specifically, with two streams over two paths, the SSIC network: 1) decreases the PER and packet loss rate (PLR) by more than fourfold compared with a single-stream network and 2) provides 99.99% reliable packet delivery for short-range communication. Potentially, SSIC could be incorporated into Wi-Fi 7 to improve the performance of its multilink mode.

The remainder of this article is organized as follows. Section II gives a quick overview of SSIC, SD for SSIS, and SSIC networking. Section III delves into the details and
Fig. 2. Architecture of SSIC wherein multiple copies of the source information bits are dispatched either over multiple networks of the same kind or multiple networks of different kinds, and wherein the soft bit information from different streams are aggregated for soft combining.

II. OVERVIEW

This section gives an overview of the three key technologies for our system: SSIC, SD for SSIS, and SSIC networking. Table 1 lists the notations used in this article.

A. SSIC

Fig. 2 shows the architecture of SSIC. The source generates \( M \) information bits \( x_0, x_1, \ldots, x_{M-1} \) and passes them to a dispatcher. The dispatcher sends \( N \) copies of information bits over multiple homogeneous networks or multiple heterogeneous networks. The \( N \) copies may be subject to different physical-layer signal processing, including the possible use of different modulations and different channel codes. However, even when the decoder of a particular copy \( n \), \( n \in \{0, 1, \ldots, N-1\} \), cannot decode the source bits \( x_0, x_2, \ldots, x_{M-1} \), the decoder often has the soft information of the source bits \( x_0^{(n)}, x_1^{(n)}, \ldots, x_{M-1}^{(n)} \), where \( x_m^{(n)} = \log(P(x_m^n = 0)/P(x_m^n = 1)) \).

The decoders expose their soft source information to an aggregator. To the extent that the noise corrupting the source information \( x_0, x_1, \ldots, x_{M-1} \) of different streams is independent, \(^1\) the aggregator can perform SSIC as follows:

\[
\tilde{x}_m^N = \frac{1}{N} \sum_{n=0}^{N-1} x_m^{(n)}. \tag{1}
\]

The aggregator decides \( x_m = 0 \) if \( \tilde{x}_m^N \geq 0 \); and \( x_m = 1 \) otherwise. The aggregator may successfully decode \( x_0, x_1, \ldots, x_{M-1} \) even when none of the \( N \) decoders succeed in decoding \( x_0, x_1, \ldots, x_{M-1} \) from their individual \( x_0^{(n)}, x_1^{(n)}, \ldots, x_{M-1}^{(n)} \).

In general, if decoder \( n \) succeeds in decoding \( x_0, x_2, \ldots, x_{M-1} \), as indicated by its CRC check, it informs the aggregator that there is no need to perform SSIC, and it simply just exposes \( x_0, x_2, \ldots, x_{M-1} \) to the aggregator.

\(^1\)This assumption holds in real settings when deploying SSIC over multiple streams running on different NICs and frequency bands. In this case, the thermal electrical noises at the receivers of different streams and the interferences from other Wi-Fi signals on different frequency bands of different streams are independent.

B. SD for SSIC

In many situations, the source information \( x_0, x_1, \ldots, x_{M-1} \) is scrambled to avoid long consecutive sequences of 0 and or 1 (see https://en.wikipedia.org/wiki/scrambler). For example, in the Wi-Fi/WMAX processing chain, as shown in Fig. 3, the channel-decoded bits at the receiver need to be descrambled in a way to allow the SSIC in (1) to be performed correctly.

More specifically, for a stream \( n \), what is being transmitted is

\[
y_m^{(n)} = \tilde{x}_m^{(n)} \oplus z_m^{(n)}, \quad m = 0, 1, \ldots, M - 1 \tag{2}
\]

where the scrambling sequence \( z_m^{(n)} \in \{0, 1\} \), \( m = 0, 1, \ldots, M - 1 \), is unique to each stream. As a result, what is obtained from the decoder \( n \) is \( \tilde{y}_m^{(n)}, \tilde{y}_1^{(n)}, \ldots, \tilde{y}_{M-1}^{(n)} \), where \( \tilde{y}_m^{(n)} = \log(P(y_m^n = 0)/P(y_m^n = 1)) \). To perform SSIC as in (1), we need to first descramble \( \tilde{y}_0^{(n)}, \tilde{y}_1^{(n)}, \ldots, \tilde{y}_{M-1}^{(n)} \) to \( \tilde{x}_0^{(n)}, \tilde{x}_1^{(n)}, \ldots, \tilde{x}_{M-1}^{(n)} \).

Without loss of generality, we explain our idea using a specific scrambler, the 7-bit and 2-tap linear-feedback shift register (LFSR) scrambler for Wi-Fi [15] shown in Fig. 4.

---

**TABLE I**

**Table of Notations**

| Symbol | Description |
|--------|-------------|
| \( M \) | Number of information bits |
| \( N \) | Number of streams |
| \( L \) | Number of pilot bits |
| \( S \) | Knowledge of the scrambler structure |
| \( S_m \) | Indexes of the subset of the registers whose XOR produce \( y_m \) |
| \( m \) | Index of a information bit |
| \( n \) | Index of a stream |
| \( x_m \) | The \( m \)-th information bit |
| \( \tilde{x}_m \) | Estimated soft information of the \( m \)-th received information bit |
| \( y_m^{(n)} \) | The \( m \)-th scrambled bit |
| \( y_m \) | The \( m \)-th received scrambled bit |
| \( \tilde{y}_m \) | Estimated soft information of the \( m \)-th received scrambled bit |
| \( z_m \) | Scrambling sequence used for the \( m \)-th information bit |
| \( z_m^{(n)} \) | Scrambling sequence used in the \( n \)-th stream for the \( m \)-th information bit |
| \( \{r_0, r_1, \ldots, r_6\} \) | Initial seed bits randomly set in the scrambler for each new transmission of a packet. |

---

Fig. 3. Wi-Fi transceiver’s physical (PHY)-layer processing chain.
the remainder of this article, unless otherwise stated, we focus on an arbitrary decoder out of the $N$ decoders and its associated scrambler and descrambler, and drop the superscript $(n)$ in our notations.

Let $r_{m,j}$ denote the $j$th scrambler’s register bit for the generation of $z_m$, where $0 \leq j \leq 6$. The scrambling bit $z_m$ is generated by XORing the first and the fourth registers’ bits as follows. Specifically, $\{z_0, z_1, \ldots, z_M\}$ are prepared to the data bits $\{x_0, x_1, \ldots, x_M\}$. The $L$ pilot bits are all set to 0. Thus, according to the scrambling structure in Fig. 4

$$y_m = z_m \oplus x_m = r_{m,0} \oplus r_{m,3} \oplus x_m.$$  

Many systems, including Wi-Fi, set the initial seed bits $\{r_0, r_1, \ldots, r_6\}$ randomly for each new transmission of a packet.

At the transmitter side, the seed bits are encoded into the transmitted sequence of bits as follows. Specifically, $L$ pilot bits $\{x_{-L}, x_{-(L-1)}, \ldots, x_{-1}\}$ are prepended to the data bits $\{x_0, x_1, \ldots, x_M\}$. The $L$ pilot bits are all set to 0. Thus, according to the scrambling structure in Fig. 4

$$y_{-L} = z_{-L} = r_{-L,0} \oplus r_{-L,3} = r_0 \oplus r_3$$
$$y_{-(L-1)} = z_{-(L-1)} = r_{-(L-1),0} \oplus r_{-(L-1),3} = r_1 \oplus r_4$$
$$\ldots$$
$$y_{-(L-6)} = z_{-(L-6)} = r_{-(L-6),0} \oplus r_{-(L-6),3} = r_2 \oplus r_5 \oplus r_6$$
$$\ldots$$
$$y_{-1} = h_{-1}(r_0, r_1, \ldots, r_6)$$

where $h_{-1}(r_0, r_1, \ldots, r_6)$ is an XOR of a subset of the seed bits $\{r_0, r_1, \ldots, r_6\}$ that depends on the size of $L$. The scrambled sequence, $y_m$ for $m = 0, \ldots, M-1$, depends on the data sequence according to (4). Furthermore, in general, $z_m = h_m(r_0, r_1, \ldots, r_6)$ is the XOR of a subset of seed bits $\{r_0, r_1, \ldots, r_6\}$.

For illustration, let us first review the conventional descrambling method of Wi-Fi. In the standard Wi-Fi implementation $L = 7$ (i.e., there are 7 seed bits). All the received bits are first hard-decoded into 0 and 1 before the descrambling operation. That is, the Wi-Fi channel decoder outputs the estimated value of $y_m : \hat{y}_m = 0$ or 1, $m = -L, -(L-1), \ldots, M-1$. Suppose that there is no error in the preamble so that $\hat{y}_m = y_m, -L \leq m \leq -1$. Then

$$\hat{y}_{-7} = y_{-7} = z_{-7} = r_0 \oplus r_3 = r_{0,1}$$
$$\hat{y}_{-6} = y_{-6} = z_{-6} = r_1 \oplus r_4 = r_{0,1}$$
$$\ldots$$
$$\hat{y}_{-1} = y_{-1} = z_{-1} = r_2 \oplus r_5 \oplus r_6 = r_{0,6}.$$  

Thus, as can be seen from the above, after their estimations, $\{\hat{y}_{-7}, \hat{y}_{-6}, \ldots, \hat{y}_{-1}\} = \{z_{-7}, z_{-6}, \ldots, z_{-1}\}$ can be preloaded into the seven registers in the structure in Fig. 4, to produce the subsequent $z_m, m = 0, 1, \ldots, M - 1$. Wi-Fi then descrambles the scrambled source bits $\hat{x}_m, m = 0, 1, \ldots, M - 1$, by

$$\hat{x}_m = \hat{y}_m \oplus z_m, 0 \leq m \leq M - 1.$$  

Note that the descrambler’s structure is the same as the scrambler’s structure of Fig. 4 except that now $x_m$ becomes $\hat{y}_m$, $y_m$ becomes $\hat{x}_m$ (i.e., for the descrambler, $\hat{y}_m$ is the input and $\hat{x}_m$ is the output), with $z_m$ remaining the same. We refer to this conventional descrambling as hard descrambling (HD) since this method uses the hard-bit $\hat{y}_m, -7 \leq m \leq -1$ to derive the hard-bit $\hat{x}_m$ and the hard-bit $\hat{x}_m, 0 \leq m \leq M - 1$. However, HD has two problems.

1) The channel decoder outputs $\{\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{M-1}\}$ rather than $\{\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{M-1}\}$. Thus, HD does not yield the soft information $\hat{x}_m, 0 \leq m \leq M - 1$ and hence cannot be used in our proposed SSIC architecture.

2) If the decoder fails to decode $y_m$ correctly from some $m$, $-7 \leq m \leq -1$, then many of the $z_m, 0 \leq m \leq M - 1$, and therefore many of the $x_m, 0 \leq m \leq M - 1$, will be incorrect.

We propose an SD technique to circumvent the above problems. Our SD can: 1) derive the soft information $\hat{x}_m$ needed for SSIC and 2) improve the decoding of $x_m$. Suppose that $y_m, -7 \leq m \leq M - 1$ is corrupted by noise. Two variants of SD described in the following are possible (see Section III-B on the details).

The first variant uses the maximum a posteriori (MAP) estimation to estimate the scrambler’s seed bits $\{r_0, r_1, \ldots, r_6\}$ by

$$\arg \max_{r_0, r_1, \ldots, r_6} P(r_0, r_1, \ldots, r_6|\hat{y}_{-L}, \hat{y}_{-(L-1)}, \ldots, \hat{y}_{-1}, S)$$  

where $\hat{y}_m = \log(P[y_m = 0]/P[y_m = 1]), -L \leq m \leq -1$, is the soft information produced by the channel decoder, and $S$ stands for the knowledge of the scrambler structure. Also, note that the MAP estimation is the same as the maximum-likelihood (ML) estimation, $\arg \max_{x_0, x_1, \ldots, x_M} P(\hat{y}_{-L}, \hat{y}_{-(L-1)}, \ldots, \hat{y}_{-1}|x_0, x_1, \ldots, x_M, S)$ to the extent that the 127 nonzero bit patterns of $\{r_0, r_1, \ldots, r_6\}$ are equally likely.

Upon deriving $\{\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_6\}$, the hard-bit $\hat{y}_m$ can be determined. Then, according to the value of $z_m, \hat{x}_m$ is obtained by simply changing the sign of $\hat{y}_m$

$$\hat{x}_m = \begin{cases} \hat{y}_m, & \text{if } z_m = 0 \\ -\hat{y}_m, & \text{if } z_m = 1. \end{cases}$$  

The second variant first calculates the probability of $x_m$, denoted by $P(x_m)$, as follows:

$$P(x_m = 0) = P(y_m = 0)P(z_m = 0) + P(y_m = 1)P(z_m = 1)$$
$$P(x_m = 1) = P(y_m = 1)P(z_m = 0) + P(y_m = 0)P(z_m = 1)$$  

where $P(y_m = 0) = \log^{-1}\hat{y}_m/(1 + \log^{-1}\hat{y}_m)$ and $P(y_m = 1) = 1/(1 + \log^{-1}\hat{y}_m)$. The derivation of $P(z_m)$ is as follows.
First we note that $z_m$ is the XOR of a subset of $\{r_0, r_1, \ldots, r_6\}$. Let us denote the indexes of the registers in this subset by $S_m$. Then, $P(z_m)$ is given by

$$P(z_m = 0) \propto \sum_{\{r_0, r_1, \ldots, r_6\} \in S_m} P(r_0, r_1, \ldots, r_6) \langle \bar{y}_{-L}, \ldots, \bar{y}_{-1}, S \rangle$$

$$P(z_m = 1) \propto \sum_{\{r_0, r_1, \ldots, r_6\} \in S_m} P(r_0, r_1, \ldots, r_6) \langle \bar{y}_{-L}, \ldots, \bar{y}_{-1}, S \rangle.$$

(11)

Once $P(z_m)$ is obtained, we use (10) to obtain $P(x_m = 0)$ and $P(x_m = 1)$. And, $\tilde{x}_m$ is calculated by

$$\tilde{x}_m = \log \frac{P(x_m = 0)}{P(x_m = 1)}.$$  

(12)

We have two remarks for these two variants.

1) $L$ in the two variants can be larger than 7 if desired. As will be shown in Section IV-A, larger $L$ has better decoding performance because larger $L$ imparts a greater degree of redundancy to the seed bits $\{r_0, r_1, \ldots, r_6\}$ through $y_m, m = -L, \ldots, -1$ to provide more protection for them.

2) For the derivation of $\tilde{x}_m$, the second variant uses the soft information of $z_m$ rather than hard-bit $z_m$ as in the first variant. As will be shown in Section IV-A, the second variant, with a better estimate for $\tilde{x}_m$, outperforms the first variant.

C. SSIC Networking

SSIC networking overlays a virtual circuit network (VC) network on the Internet. In the VC network, a VC ID (VCI) is used to identify a source-destination pair. The SSIC network supports TCP/IP networking among SSIC nodes equipped with multiple NICs. For each SSIC node, all of its MAC addresses, one for each NIC, are associated with a VCI and a single IP address.

Fig. 5 illustrates the main idea. Nodes A and B communicate over two separate physical wireless paths. These two paths can involve disjoint wireless sections. Particularly, Fig. 5 shows two possible scenarios. In Fig. 5(a), the two paths are in the same physical wireless network and terminate at the same end points at node A and at node B. In Fig. 5(b), the two paths are over different physical networks. In this scenario, node A is a mobile device with two NICs to the two networks. Node B is a server within a core network. We will investigate the performance of SSIC in these two scenarios in Sections IV-B1 and IV-B2.

In both two scenarios, a unique VCI, VCI_{A,B}, is assigned to the VC frames between node A and node B. The IP addresses $I_A$ and $I_B$ within the same subnet are assigned to node A and node B, respectively. $P_A^1$ and $P_A^2$, node A’s MAC addresses, are associated with VCI_{A,B} and $I_A$; $P_B^1$ and $P_B^2$, node B’s MAC addresses, are associated with VCI_{A,B} and $I_B$. This setup will also be used in our experiments in Section IV-B.

III. DETAILED SYSTEM DESIGN

This section delves into the details and nuances of the system design. Section III-A describes how to acquire the soft information needed for SSIC. Section III-B presents two variants of SD variants. Section III-C details the networking issues related to the deployment of SSIC, including the middleware design and the VC frame format. For concreteness, throughout

\[ \text{https://github.com/Leo-Cheung-CUHK/Soft-Source-Information-Combing-SSIC} \]
this section, we assume that SSIC is deployed over Wi-Fi networks, although SSIC is a general technique that can be deployed over other networks or heterogeneous networks consisting of networks of different types.

A. Soft Information Acquisition

Soft information, such as $\hat{x}_m$ or $\tilde{y}_m$ in (9) can be obtained using a soft-output channel decoder. Take Wi-Fi for example, in the 802.11 standard [15], the low-density parity-check (LDPC) decoder is used to decode LDPC codes, and the hard-output Viterbi decoder is used to decode convolutional codes. For the LDPC coding scheme, the LDPC decoder can readily provide the soft output. For the convolutional coding scheme, we need to replace the conventional hard-output Viterbi decoder with a soft-output decoder, e.g., the soft-output Viterbi algorithm (SOVA) decoder [17] or the MAP-based BCJR (named after its inventors: Bahl, Cocke, Jelinek, and Raviv) decoder [18]. In this work, we assume the use of the LDPC coding scheme and the LDPC decoder.

Given that descrambling is needed in Wi-Fi, the steps for the acquisition of $\hat{x}_m$ in Wi-Fi are as follows.

1) The output of the LDPC decoder $\tilde{y}_m$ is forwarded to the middleware software.

2) Then, the SD in the middleware performs descrambling to obtain $\tilde{x}_m$.

3) If there are several copies of the $\tilde{x}_m$ : $\tilde{x}_m^n, n = 1, 2, 3, \ldots, N$, the middleware then performs SSIC as in (1).

In terms of the local transfer of $\tilde{y}_m$ from the hardware to the middleware, a simple one-bit field inside the packet informs the middleware whether this packet is hard or soft. Specifically, if it is a hard packet, the packet is a sequence of binary values; otherwise, the packet is a sequence of LLR floating-point values. Meanwhile, a proper quantization method can reduce the storage requirements for LLR values. Much research [19] has been done on the number of bits per floating-point value is needed to reach a certain LLR combining performance. Hence, rather than repeating the investigation here, we adopt a proven quantization method (see Table III in Section IV-B).

B. SD Methodology

One naïve method to derive $\tilde{x}_m$ from $\tilde{y}_m$ from the LDPC decoder is as follows.

1) The first seven soft bits $\{\tilde{y}_{-7}, \tilde{y}_{-6}, \ldots, \tilde{y}_{-1}\}$ are hard decoded into $\{\hat{y}_{-7}, \hat{y}_{-6}, \ldots, \hat{y}_{-1}\}$. These seven hard bits $\{\hat{y}_{-7}, \hat{y}_{-6}, \ldots, \hat{y}_{-1}\}$ are then preloaded into the seven registers in the structure in Fig. 4.

2) The scrambler’s seeds $\{z_{-7}, z_{-6}, \ldots, z_1\}$ are obtained using (6). With the seeds, all the other $z_m, m = 0, \ldots, M - 1$ is derived using (3).

3) The soft bits $\tilde{x}_m$ are then obtained by directly changing the sign of $\tilde{y}_m$ as in (9) based on the $z_m$ obtained in 2).

Although this method is simple, it is, in fact, the same as the conventional HD as far as the decoding of $\{r_0, r_1, \ldots, r_6\}$ is concerned; just that the $\tilde{x}_m$ obtained is still soft information—i.e., this is a hard $\{r_0, r_1, \ldots, r_6\}$ soft $\tilde{x}_m$ method. In particular, the system will fail to obtain the correct descrambled $\tilde{x}_m$ if the hard $\{r_0, r_1, \ldots, r_6\}$ is mis-decoded. As will be shown in Section IV-A, this method is far from optimal when deployed with SSIC. We refer to this method as SSIC naïve since it can still output soft information $\tilde{x}_m$.

Next, we present two advanced variants of SD to obtain better estimated $\tilde{x}_m$. For both variants, redundancy has been introduced to $\{r_0, r_1, \ldots, r_6\}$ so that more than seven pilot bits are used to transmit information related to $\{r_0, r_1, \ldots, r_6\}$.

First Variant [Hard-r-Soft-x (HRSX)]: The first method is based on the observation that the correct decoding of $\{r_0, r_1, \ldots, r_6\}$ is critical. If $\{r_0, r_1, \ldots, r_6\}$ is mis-decoded, the soft $\tilde{x}_m$ obtained in the naïve method will not only be useless, but may actually cause harm when $\tilde{x}_m$ is combined with other streams using SSIC because the signs of $\tilde{x}_m$ for many bits will be reversed. For higher protection of $\{r_0, r_1, \ldots, r_6\}$, we could introduce redundancy. In particular, instead of using seven pilot bits for $\{r_0, r_1, \ldots, r_6\}$, we could use $L \geq 7$ bits to encode $\{r_0, r_1, \ldots, r_6\}$. Then, at the receiver, we use the $L$ bits to decode $\{r_0, r_1, \ldots, r_6\}$ so as to reduce the error rate. We note that this redundancy is in addition to the PHY-layer redundancy of the convolutional code. The extra redundancy is added in view of the fact that, for SSIC, it is crucial that $\{r_0, r_1, \ldots, r_6\}$ from each stream is decoded correctly so that the sign of $\tilde{x}_m$ is not reversed. In this way, aligned combination of $\tilde{x}_m$ from different streams can be performed.

Recall from Section II-B that, with $L$ pilot bits, the first variant of SD aims to derive $\{r_0, r_1, \ldots, r_6\}$ using the MAP/ML estimation arg max$_{r_0, r_1, \ldots, r_6}$ $P(r_0, r_1, \ldots, r_6|\tilde{y}_{-L}, \tilde{y}_{-(L-1)}, \ldots, \tilde{y}_{-1}, S)$ as written in (8). Let us examine the a posteriori probability here. First, let us define $\tilde{y} = (y_{-L}, y_{-(L-1)}, \ldots, y_{-1})$ and $\tilde{r} = (r_0, r_1, \ldots, r_6)$. For $L \geq 7$, there is a one-to-one mapping from $\tilde{r}$ to $\tilde{y}$ (i.e., not all bit patterns of $\tilde{y}$ are possible; only 127 of them are possible, each corresponding to a particular $\tilde{r}$). We can write $\tilde{y}$ as a function of $\tilde{r}$ as $\tilde{y} = f(\tilde{r})$. In fact, for a given $\tilde{r}$, $y_{m}, -L \leq m \leq -1$, are all determined. Thus, we can also write $y_m$ as a function of $\tilde{r}$ as $y_m(\tilde{r})$. Then, the a posteriori probability $P(r_0, r_1, \ldots, r_6|\tilde{y}_{-L}, \tilde{y}_{-(L-1)}, \ldots, \tilde{y}_{-1}, S)$ can be written as follows:

$$P(\tilde{r}|\tilde{y}_{-L}, \tilde{y}_{-(L-1)}, \ldots, \tilde{y}_{-1}, S) = P(y_{-L}(\tilde{r})y_{-(L-1)}(\tilde{r})\cdots y_{-1}(\tilde{r})|\tilde{y}_{-L}, \tilde{y}_{-(L-1)}, \ldots, \tilde{y}_{-1}, S)$$

$$= \sum_{\tilde{r} \in \tilde{R}} P(y_{-L}(\tilde{r})y_{-(L-1)}(\tilde{r})\cdots y_{-1}(\tilde{r})|\tilde{y}_{-1})$$

$$= \sum_{\tilde{r} \in \tilde{R}} P(y_{-L}(\tilde{r})y_{-(L-1)}(\tilde{r})\cdots y_{-1}(\tilde{r})|\tilde{y}_{-1})$$

where $\tilde{R}$ is the set of the 127 nonzero bit patterns for $\{r_0, r_1, \ldots, r_6\}$. Define

$$f(\tilde{r}) = f_{-L}(\tilde{r}) \cdots f_{-1}(\tilde{r})$$

where

$$f_{m}(\tilde{r}) = P(y_m(\tilde{r})|\tilde{y}_m), -L \leq m \leq -1.$$
and

$$\arg \max \{ P(\vec{r}|y_{-L}, y_{-(L-1)}, \ldots, y_{-1}, S) \} = \arg \max \{ f(\vec{r}) \}. \quad (17)$$

Deriving $y_m(\vec{r}), -L \leq m \leq -1$: We note that $y_m(\vec{r})$ is the XOR of a subset of the seed bits $\{r_0, r_1, \ldots, r_6\}$ that depends on the scrambler structure. The assembly of the equations can be written in matrix form as follows:

$$[y_{-L}, y_{-(L-1)}, \ldots, y_{-1}]^T = A[r_0 \ r_1 \ \ldots \ r_6]^T \quad (18)$$

where

$$A = \begin{bmatrix}
ad_{-L,0} & ad_{-L,1} & \cdots & ad_{-L,6} \\
ad_{-(L-1),0} & ad_{-(L-1),1} & \cdots & ad_{-(L-1),6} \\
\vdots & \vdots & \ddots & \vdots \\
ad_{-1,0} & ad_{-1,1} & \cdots & ad_{-1,6}
\end{bmatrix}$$

and $a_{m,j} = 1$ or $0, -L \leq m \leq -1, 0 \leq j \leq 6$. Note that the summation operation in (18) is a mod-2 operation. Thus, for each $y_m$ we have

$$y_m(\vec{r}) = a_{m,0}r_0 + a_{m,1}r_1 + \ldots + a_{m,L-1}r_6. \quad (19)$$

We can devise an algorithm to determine the value of $a_{m,j}$, which depends on the scrambler structure, as follows (in the following, $S_m$ is the indexes of the subset of the registers whose XOR produce $y_m$, i.e., $y_m = \sum_{r \in S_m} f(r) \pmod{2}$).

1) Initialize Seven Sets: $S_{-L-7} = \{0\}, S_{-L-6} = \{1\}$, $S_{-L-5} = \{2\}, S_{-L-4} = \{3\}, S_{-L-3} = \{4\}, S_{-L-2} = \{5\}$, and $S_{-L-1} = \{6\}$.

2) For each $m, -L \leq m \leq -1$, we further define a corresponding set $S_m$ as follows:

$$S_m = S_{m-7} \cup S_{m-4} - S_{m-7} \cap S_{m-4} \quad (20)$$

where the set operations in (20) realize the XOR operation that produce the $y_m$ (i.e., $y_m$ given that the pilot bits $x_m$ are 0) in the scrambler structure of Fig. 4.

3) Given a specific $m, -L \leq m \leq -1$ and $j, 0 \leq j \leq 6$, $a_{m,j}$ is determined by

$$a_{m,j} = \begin{cases} 1, & \text{if } j \in S_m \\ 0, & \text{if } j \notin S_m \end{cases} \quad (21)$$

After determining $a_{m,j}$, we can use (19) to list the $L$ equations of $y_m$ and $\{r_0, r_1, \ldots, r_6\}$. Note that the derivation of $a_{m,j}$, which is independent of the values of $\{r_0, r_1, \ldots, r_6\}$, only needs to run once during initialization.

Deriving $f_m(\vec{r}), -L \leq m \leq -1$, and $f(\vec{r})$: From (15), we have

$$f_m(\vec{r}) = P(y_m(\vec{r})|\vec{y}_m) = \begin{cases} \log^{-1} y_m, & \text{if } y_m(\vec{r}) = 0 \\ \log^{-1} y_m, & \text{if } y_m(\vec{r}) = 1 \\ \log^{-1} y_m, & \text{if } y_m(\vec{r}) = 1 \end{cases} \quad (22)$$

where $\vec{y}_m = \log(P(y_m = 0)/P(y_m = 1))$ is the output of the LDPC decoder as discussed in Section III-A. The $L$ values of $f_m(\vec{r}), -L \leq m \leq -1$ obtained by (22) can then be substituted into (14) to get $f(\vec{r})$.

With $f(\vec{r})$, the first variant is as follows.

1) From (17), we obtain the maximum a posteriori $\vec{r}^\ast = \arg \max_{\vec{r}} f(\vec{r})$.

2) We load $\vec{r}^\ast$ as the initial values to the scrambler’s registers, and $z_m, m \geq 0$, can then be obtained as the successive outputs form the scrambler. Then, according to the value of $z_m$, $\vec{x}_m$ is obtained by simply changing the sign of $\vec{y}_m$ using (9).

We refer to this variant as the HRSX since it hard decodes the scrambling bit $z_m$ (i.e., we hard decode $\{r_0, r_1, \ldots, r_6\}$ which gives rise to hard $z_m$) to derive the soft information $\vec{x}_m$. In the meantime, it is worthwhile to note that the naive method is also a kind of HRSX, but without the “redundant encoding” of $\vec{x}_m$ in $y_{-L}, y_{-(L-1)}, \ldots, y_{-1}$.

Second Variant [Soft-r-Soft-x (SRSX)]: The second variant uses the estimated probability of $z_m$ to improve the estimation. Note that for the estimated probability of $x_m, P(x_m)$, we have

$$P(x_m = 0) = P(y_m = 0)P(z_m = 0) + P(y_m = 1)P(z_m = 1)$$

$$P(x_m = 1) = P(y_m = 1)P(z_m = 0) + P(y_m = 0)P(z_m = 1). \quad (23)$$

We have (23) because the noises incurred in obtaining the probabilities $P(y_m)$ and $P(z_m)$ are also independent. Once we obtain the above $P(x_m)$ for $x_m$, we can also obtain $\vec{x}_m$.

The following describes the steps of this variant.

1) $f(\vec{r})$ for $-L \leq m \leq -1$ is derived as in HRSX.

2) Note that $z_m$ is the XOR of a subset of the possible values for $\vec{r}$ and has a period of 127. Let $R_m = \{r_0, r_1, \ldots, r_6 : \sum_{r \in S_m} f(r) \pmod{2} = 0\}$ where the summation here is the mod 2 summation. The $P(z_m)$ for $-L \leq m \leq -1$ is given by

$$P(z_m = 0) = \frac{\sum_{r \in R_m} f(\vec{r})}{\sum_{r \in R} f(\vec{r})}$$

$$P(z_m = 1) = 1 - P(z_m = 0). \quad (24)$$

3) Once $P(z_m)$ is derived, we obtain $P(x_m)$ using (23). Then, $\vec{x}_m$ is given by

$$\vec{x}_m = \log(P(x_m = 0)/P(x_m = 1)). \quad (25)$$

We refer to the second variant as the SRSX since it soft decodes the scrambling bit $z_m$ to derive $\vec{x}_m$.

In conclusion, all variants of SD, including the SSIC naïve (HRSX without redundancy), HRSX, and SRSX, can be deployed in the SSIC network. Specifically, given $N, N \geq 1$ streams in total, after deriving $\vec{x}_m$ by using any variant of SD, the information bit $x_m$ is finally decoded by

$$x_m = \begin{cases} 0, & \text{if } \vec{x}_m = \sum_{n=0}^{N-1} z_n^{(n)} \geq 0 \\ 1, & \text{if } \vec{x}_m^N = \sum_{n=0}^{N-1} z_n^{(n)} < 0. \end{cases} \quad (26)$$

C. Detailed Design of SSIC Networking

This section details the SSIC networking framework, including the middleware design and the VC frame format. As illustrated in Fig. 6, a TUN device [16] is first created along with the middleware on each SSID node. The IP address of each node’s TUN device is set to the VC IP address. The user application, on the other hand, can use any type of socket...
Fig. 6. Detailed working mechanism of the middleware.

Fig. 7. Layer-2 VC frame format: a VC header is introduced in front of the layer-2 payload.

(e.g., TCP socket, UDP socket, or raw socket) to use the SSIC network service. A packet generated from the user application is sent to the network stack by the socket. Thanks to the TUN devices in the operation system (OS), if the destination IP address is located within the SSIC network, this packet is then forwarded to the TUN device by the network stack according to the system’s routing table.

The VC frame format is shown in Fig. 7, where a VC header is added in between the MAC header and a copy of the IP packet. The VC header includes a VCI to identify the source-destination pair, a VC sequence number (VCS) to identify and match the copies of an IP packet, CRC bits, and parity bits. In particular, the parity bits are to provide extra-strong protection to the header, making sure that the header can be decoded to hard bits correctly (by checking the CRC bits) even if the receiver’s VC frame fails the layer-2 CRC. With the VC being identified by the decoded header, at the destination end, SSIC can be performed over the VC frames with the same VC header to decode the IP packet.

At the source end, an IP packet with $I_A$ as source IP and $I_B$ as destination IP is generated by the user application. Then the IP packet is forwarded to TUN and read by the middleware. Upon getting the IP packet from TUN, the middleware encapsulates the IP packet into a VC frame. Specifically, the SSIC middleware generates a VC header and prepends the VC header to the IP packet. Then, the SSIC middleware duplicates the IP packet prepended with the VC header, and further prepends different MAC headers to them. At this time, all the VC frames are constructed. In general, with $N$ NICs, the middleware generates $N$ VC frames with different MAC headers for $N$ NICs. After that, the middleware sends these VC frames to their corresponding NICs using the layer-2 raw sockets.

Remark 1: At the source end, the middleware generates multiple VC frames carrying the same IP packet and the same VC header. However, VC frames destined for different NICs have different MAC headers.

At the destination end, the middleware receives either a hard or a soft VC frame carrying an IP packet from any one of the NICs. The MAC layer returns the hard VC frame if it passes the layer-2 CRC; or the soft VC frame if it fails the layer-2 CRC. Upon receiving a VC frame, the middleware performs the following steps.

1) If the VC frame is a hard frame, the middleware directly goes to the final step.
2) If the VC frame is a soft frame, the middleware first performs SD if the scrambling process exists in the lower layer. Then, it hard-decodes the VCI, VCS, and CRC from the VC header and checks its CRC status. If the frame passes CRC, the middleware detects a valid VC frame and goes to the next step; otherwise, the VC frame is discarded.
3) The middleware checks whether it has received the same VC frame that passes the layer-2 CRC before. If yes, the latest received VC frame is also discarded as a duplicate; otherwise, it goes to the next step.
4) The middleware checks whether it has received a soft VC frame with the same VCI and VCS as that of the latest VC frame. If yes, the middleware collects all these VC frames and performs SSIC; otherwise, the latest soft VC frame is stored locally in the middleware.
5) The middleware extracts the IP packet from the VC frame and writes it to the TUN device. From there, the IP packet goes to the user application. The reception of the IP packet is then complete.

Remark 2: On each SSIC node, the MAC layer returns soft VC frame when the VC frame fails the layer-2 CRC so that SD and SSIC can be performed.

IV. SIMULATION AND EXPERIMENTAL VALIDATIONS

Section IV-A presents simulation results to validate the theoretical performance of SD, Section IV-B presents experimental results over an SSIC network in a real environment.

A. Theoretical Performance of SD

To validate the theoretical performance of SD, we performed simulations in MATLAB in which the wireless local area network format waveform was generated and passed through the AWGN channel. The detailed settings are listed in Table II. In this simulation, four streams with different settings were used. Specifically, stream $n, 1 \leq n \leq 4$, has a signal-to-noise ratio (SNR) of $(0.5(n - 1) + p)$ dB at the index of $p, 8 \leq p \leq 16$. That is, among all these four streams, stream 1 experiences the lowest SNR throughout the experiment, whereas stream 4 experiences the largest SNR. For
TABLE II
SIMULATION SETTINGS

| Payload Size       | 1500 Bytes  |
|--------------------|-------------|
| Modulation and Channel Coding scheme | 64 QAM, 2/3 Code Rate (LDPC) |
| Number of streams  | 4           |
| Stream 1 SNR range | 5dB to 18dB |
| Stream 2 SNR range | 8.5dB to 18.5dB |
| Stream 3 SNR range | 9dB to 19dB  |
| Stream 4 SNR range | 9.5dB to 19.5dB |

Fig. 8. BER of the single-stream case with different $L$: (a) seed-bit decoding performance of HD, HRSX, and SRSX; and (b) payload decoding performance of HD, HRSX, and SRSX is investigated.

For example, stream 1 has an SNR of 8 dB at index 8, and stream 4 has an SNR of 17.5 dB at index 16.

The simulation results are shown in Figs. 8 and 9. Fig. 8 shows the BER of the single-stream case (results of stream 1, which is the worst case among the single-stream cases), whereas Fig. 9 shows the BER and PER performance for both the SSIC case and a single-stream case (results of stream 4, which is the best case among the single-stream cases). We can see the following.

1) Fig. 8(a) shows that both HRSX and SRSX significantly improve the decoding of seed bits $\{r_0, r_1, \ldots, r_6\}$. With redundancy introduced $\{r_0, r_1, \ldots, r_6\}$, the larger $L$ the better the performance of HRSX and SRSX compared with HD.

2) Fig. 8(b) shows that with a better estimation of $\{r_0, r_1, \ldots, r_6\}$, both HRSX and SRSX improve the BER of the packet in the single-stream case. However, HRSX and SRSX have almost the same BER performance.

3) Fig. 9 shows that SSIC with either HRSX or SRSX descrambling has better BER and PER performance than SSIC with SSIC naïve. In particular, when $L = 16$, SSIC with SRSX already has 1-dB BER gain over SSIC with SSIC naïve, as shown in Fig. 9(a).
Fig. 10. PPDU format of legacy IEEE 802.11 in which 16 zeros bits already exist in the SERVICE field.

4) Fig. 9 shows, in general, SSIC with SRSX outperforms SSIC with HRSX. With a larger $L$, the performance gap between HRSX and SRSX becomes smaller. In particular, HRSX approximates SRSX when $L = 127$, as shown in Fig. 9(b) and (d).

In short, with more than one stream’s soft information, SSIC with SD significantly improves communication reliability. What’s more, SRSX is preferred when SD is needed in SSIC since SRSX maintains good performance even when $L$ is small. For example, when $L = 16$, as shown in Fig. 9, SSIC with SRSX can already obtain good BER and PER performance and a further increase in $L$ is not necessary. Note that the legacy IEEE 802.11 PPDU already has 16 zeros bits in the SERVICE field as pilot bits, as shown in Fig. 10. Overall, SRSX, $L = 16$, is the best choice when deploying SSIC over Wi-Fi networks since it does not require any further pilot bits beyond those given by the standard.

Remark 3: Since the legacy 802.11 PPDU already has 16 zeros bits in the SERVICE field as pilot bits, SRSX with $L = 16$ is preferred when deploying SSIC in the Wi-Fi network because it maintains good performance when $L = 16$ and does not require extra pilot bits and hence incurs zero additional overhead with respect to the standard.

Let us look at another simulation setup with all four streams having the same SNR. We focus on the case with $L = 16$. The BER and PER results are shown in Fig. 11(a) and (b), respectively. We can see from these results that the improvement by SSIC is more obvious than in Fig. 9(a) and (c), with more than 2-dB BER and PER gains. In the next section, we investigate the performance of SSIC with SRSX in a real Wi-Fi network.

B. SSIC Network Performance

To validate the overall performance of SSIC in a real environment, we set up an SSIC network in our laboratory. Two general PCs are set up as shown in Fig. 12: 1) personal computer (PC) A, a laptop as shown in Fig. 12(a), serves as node A in Fig. 5, and it is an SSIC transmitter with two commercial-used Wi-Fi USB sticks (Two 802.11n Wi-Fi adapters provided by DCOM Technology CO., LTD3 were used) and 2) PC B, a PC as shown in Fig. 12(b), serves as node B in Fig. 5, and it is an SSIC receiver equipped with two SDR devices—Universal Software Radio Peripheral (USRPs) [20].

Here, each USRP attached to PC B simulates an NIC that is capable of outputting the output of $\hat{y}_m$—there is no such commercial NIC yet. Furthermore, we disable the 802.11 retransmission because the latency between PC and the middleware, written in python, runs on both nodes. Furthermore, the Bose–Chaudhuri–Hocquenghem (BCH) coding scheme is used to protect the VC header. The python modules that the middleware needs are listed in Table III.

Meanwhile, as suggested in Remark 3, the SRSX with $L = 16$ is used for SD purposes.

We investigated the performance of SSIC networking over two noisy channels on the 2.4-GHz ISM band (one on WLAN

---

3https://www.globalsources.com/Wireless-USB/wifi-adapter-for-window-7-1184974073p.htm
always at the same place. This setup is as in Fig. 5(a), whereas PC B together with its attached two USRPs were located at four different places (location “L1,” “L2,” “L3,” and “L4,” as indicated in Fig. 13) for the different rounds of the experiment, whereas PC B attached with USRPs was fixed at one place all the time.

Once set up, a UDP client running on PC A sends UDP packets to a UDP server running on PC B. In particular, each UDP packet is duplicated and encapsulated into two VC packets by the middleware on PC A (detailed in Section III-C). These two VC packets are then separately transmitted by the two NICs. In the meantime, the two USRPs on PC B receive the packets sent from PC A over the two streams.

Two experiments with different device-placement profiles are performed. In each experiment, we compare SSIC with two single-stream cases—one in which only stream 1 is used and the other in which only stream 2 is used—and one multistream case with duplicate transmissions over both streams. In the duplicate-transmission case, referred to as DUP in the rest of this article, packets are processed in a first-come-first-serve (FCFS) manner, and duplicated packets that arrive later, if any, are dropped (as in PRP).

1. **Experiment 1**: In this experiment, PC A was located in four different places (location “L1,” “L2,” “L3,” and “L4,” as indicated in Fig. 13) for the different rounds of the experiment, whereas PC B together with its attached two USRPs were always at the same place. This setup is as in Fig. 5(a), simulating the communication of two static devices. The details of the experiment, e.g., SNR and distance between PC A and PC B, for each round can be found in Table IV.

We first investigated the VC-header decoding performance since the middleware relies on the VC header to identify a packet even when the packet does not pass the layer-2 CRC. To allow this investigation, the middleware on PC B collects the layer-2 CRC status and the VC-header CRC status for each packet. These statuses are shown in Table IV. We can see that the failure rate of the VC-header CRC is 10× lower than that of layer-2 CRC. The VC header is well protected in the SSIC communication.

We next investigated the packet-deducing performance of SSIC. Note that for Wi-Fi, only after the presence of a packet is detected will the packet decoding process kicks in. Hence, for a more thorough investigation, the middleware at PC B gathers statistics on PLR—the packet miss-detection rate, PER—the probability of decoding failure for detected packets, and failure rate, $\text{FR} = 1 - (1 - \text{PLR})(1 - \text{PER})$, of the aforementioned four cases. Fig. 14 shows the PLR, PER, and FR in round 4 of the experiment. This is the case when the distance between PC A and B is the largest in experiment 1. The full experimental results for all four rounds can be found in Table V.

To better explain these results, we also plot the signal amplitudes of some received packets in round 4 in Fig. 15. In particular, Fig. 15(a) and (b) show some of the packets received by stream 1 and stream 2, respectively. For easy reference, we index the six shown packets by 1 to 6 in Fig. 15.

Any two packets with the same index have the same VCS and hence are used for SSIC. We explain the results in the following.

1. **In terms of PLR, SSIC, and DUP share the same performance and significantly outperform the single-stream cases, as shown in Fig. 14(a). This result is intuitive, because for both, a packet is not detected (lost) only if it is neither detected in stream 1 nor stream 2. We explain this result through Fig. 15. For example, although packet 3 is totally distorted by interference and cannot be detected in stream 1 [as circled in red in Fig. 15(a)], it is well received in stream 2 [as circled in green in Fig. 15(b)]. With more than a stream, multistream transmission, such as SSIC and DUP have much lower PLR.

2. While DUP and SSIC have the same low PLR, in terms of PER among the detected packets, DUP does not have an improvement over the single-stream case. SSIC, thanks to the signal combination from the two streams, has a much lower PER, as shown in Fig 14(b). We further explain this result through Fig. 15. For example, packet 4, 5, and 6 are not decoded successfully in either stream 1 or stream 2, as circled in yellow in Fig. 15(a) and (b). However, by combining two copies of soft information, SSIC successfully decodes these packets.

Overall, SSIC networking has the lowest FR in the noisy environment and provides good packet delivery performance, as shown in Fig. 14(c). Specifically, with a 3-m nonline-of-sight (NLOS) distance between PC A and PC B, the FR of SSIC is 6× lower than that of single-stream cases and is more than...
TABLE IV
VC CRC FAILURE RATE VERSUS LAYER-2 CRC FAILURE RATE IN EXPERIMENT 1

| Round No. | PC A Location | Layer-2 CRC failure rate (stream 1) | VC Header CRC failure rate (stream 1) | Layer-3 CRC failure rate (stream 2) | VC Header CRC failure rate (stream 2) | Distance Between PC A and PC B | SNR |
|-----------|---------------|--------------------------------------|--------------------------------------|-------------------------------------|--------------------------------------|--------------------------------|-----|
| 1         | L1            | 0.0017                               | 2.04e-04                             | 0.0013                              | 3.4264e-04                          | 1M LOS                           | 50 dB |
| 2         | L2            | 0.0017                               | 6.1782e-04                           | 0.0093                              | 5.8680e-04                          | 2M LOS                           | 45 dB |
| 3         | L3            | 0.0107                               | 0.0014                               | 0.0097                              | 6.90276e-04                         | 3M NLOS                          | 35 dB |
| 4         | L4            | 0.0095                               | 0.0014                               | 0.0206                              | 0.0024                              | 4M NLOS                          | 30 dB |

Fig. 14. PLR, PER, and FR of four cases (two single-stream cases, DUP and SSIC) are compared in experiment 1.

2× lower than that of DUP. What’s more, as highlighted in Table V, SSIC provides 99.99% reliable transmission when the distance between two end devices is below 3 m.

2) Experiment 2: In this experiment, PC A moved from location “L5” to location “L10,” as illustrated in Fig. 16. The two USRPs of PC B were located at different places. This setup is as in Fig. 5(b), simulating the case when one end device is moving while communicating with one or two APs simultaneously. The details of receiving SNR information for different locations can be found in Table VI.

The experimental results, including PLR and PER, are shown in Fig. 17 and described as follows.

1) In both single-stream cases, PLR fluctuates, and PER increases as the PC A moves away from its receiver. In particular, the variant of PER is large. For example, when only using stream 2, PER at location “L5” is around 6× larger than that at location “L10,” as shown in Fig. 17(a).

2) SSIC and DUP reduce both PLR and PER, as shown in Fig. 17(a). For example, SSIC and DUP enable PC A to have a PLR of around 0.01 from location “L5” to location “L10,” lowering PLR up to 4× than that of single-stream cases. Particularly, SSIC achieves the lowest PER among all the four cases, as shown in Fig. 17(b).

Experiment 2 shows that SSIC networking stabilizes both PLR and PER even though the performance of single streams...
Table V
PLR, PER, AND FR OF FOUR CASES IN EXPERIMENT 1

| Round No. | Stream 1 | Stream 2 | DUP | SSIC | Stream 1 | Stream 2 | DUP | SSIC | Stream 1 | Stream 2 | DUP | SSIC |
|-----------|----------|----------|-----|------|----------|----------|-----|------|----------|----------|-----|------|
| 1         | 0.0017   | 0.0013   | 0   | 0    | 0.0034   | 0.0047   | 0   | 0    | 0.0051   | 0.0060   | 0   | 0    |
| 2         | 0.0017   | 0.0093   | 0   | 0    | 0.0100   | 0.0090   | 0.0033 | 0    | 0.0117   | 0.0182   | 0   | 0.0003 |
| 3         | 0.0107   | 0.0097   | 0   | 0    | 0.0189   | 0.0143   | 0.0044 | 0.0001 | 0.0294   | 0.0238   | 0   | 0.004 |
| 4         | 0.0095   | 0.0206   | 0.0001 | 0.0001 | 0.0335   | 0.0208   | 0.0016 | 0.0006 | 0.0427   | 0.0410   | 0   | 0.0017 |

Fig. 16. Experiment 2, PC A was moving from location “L5” to location “L10,” and two USRPs of PC B were located at different places.

fluctuates due to the movement of the end device, achieving the lowest overall FR, as shown in Fig. 17(c). Furthermore, Fig. 17 indicates that if the PLR (rather than the PER) is the limit, then the improvement of SSIC over DUP is not much. In other words, if the packet detection process could be improved, then the advantage of SSIC over DUP would be more obvious. This observation offers us a direction for future work. For example, a cooperated packet-detection process over different streams can be designed so that once one stream has detected a packet, this stream would notify other streams to get ready for their receptions, reducing the PLR as much as possible.

V. RELATED WORK

Miu et al. [23] proposed combining receptions from multiple transmitters to recover faulty packets. However, the method in [23] does not use the soft information of the layer-2 payload as we do. Specifically, the transmitter in [23] divides the layer-2 payload into multiple blocks. When the transmitters receive conflicting blocks, [23] attempts to resolve the conflict by trying all block combinations and checking whether any resulting combinations produce a packet that passes the layer-2 checksum test. Without the capability of combining faulty packets carrying the same content, [23] cannot help when conflicting blocks are all wrong. As shown in Fig. 18, in a simulation setup with four streams having the same SNR, our SSIC methods significantly outperform the method in [23].

The method in [24] uses the soft information to find chunks that are likely to be wrong in a packet and retransmit only those chunks rather than the entire packet. However, this design is not compatible with existing NICs since legacy NICs are not allowed to transmit only a particular part of a packet.

Moreover, [24] does not discuss the combining of faulty packets carrying the same content and does not have a multistream transmission mechanism.

The closest work to ours is [25], which proposed an architecture for WLAN to allow the physical layer to convey its soft source bits to the higher layers. A receiver can then combine the source bits from multiple transmitters to correct faulty bits in a corrupted packet. However, [25] did not take care of
is compatible with commercial NICs and TCP/IP networks. the PHY-layer signal without validating whether the design implementation realized on USRPs in [25] only processed networking design to resolve Challenge 2 in Section I. The soft information from different streams leads to decoding masks before they can be combined. Directly combining descrambled at the receiver side with their respective scram-

The WLAN system are scrambled randomly and need to be

packet that fail to be decoded (i.e., that fail the CRC), we could combine their LLRs using SSIC.

VI. DISCUSSION

A. SSIC in Single-Link Mode

SSIC can also be deployed in the single-link mode. For example, SSIC can be incorporated into the current Wi-Fi PHY-retransmission scheme to further improve both PER and FR. Instead of discarding the copies of the same packet that fail to be decoded (i.e., that fail the CRC), we could combine their LLRs using SSIC.

Challenge 1 remains: directly combining the LLRs from different rounds of retransmission is not viable. We have validated through experiments that even a single NIC uses different scrambling sequences for the first transmission and the later retransmissions of the same packet. Hence, SD of SSID as proposed in this article is still needed so that the soft information from different rounds of transmissions can be combined.

B. SSIC for Low-Latency Communication

Here, we discuss the possibility of reducing communication latency using the SSIC paradigm. SSIC provides extra opportunities to decode source information from several corrupted copies. When deploying SSIC over Wi-Fi with multiple streams and with the original Wi-Fi MAC retransmission scheme over each of the multiple streams, we can expect SSIC to significantly reduce the number of retransmissions. Consider the worst case (case 3) in Fig. 1, where two copies of the same packet are sent over two streams. SSIC provides an extra chance to decode the packet from these copies; hence further retransmissions from each stream are not needed anymore. In order to reduce the number of retransmissions to achieve low latency, the retransmission mechanisms of the different streams will need to work collaboratively. Specifically, once a packet is decoded, ACK needs to be sent to all the streams, notifying them of the successful reception.

C. Scalability of SSIC Middleware

Here, we discuss the scalability of the SSIC middleware design when \( N \) is large.

1) In the transmitting chain, the computation time cost lies in constructing and transmitting VC frames for the \( N \)-link transmissions. A possible way to reduce computation time is as follows. After prepending the VC header to the IP packet, the middleware uses multiple threads or multiple processes to construct \( N \) VC frames and sends them to their corresponding NICs in parallel. In this way, the time from receiving an IP packet to the completion of transmissions of all \( N \) VC frames can be much reduced.

2) In the receiving chain, the computation time cost lies in SD and SSIC. Recall from Section III-B that SD has two variants: a) HRSX and b) SRSX. For both HRSX and SRSX, the overall computation complexity of the \( N \) streams is \( O(N) \), with each stream having the complexity of \( O(M) \), the same as for HD. Note that the SD processing of different streams is independent. Thus, if different threads or processes are used to process different streams, the time complexity is \( O(M) \). The overall computation complexity of SSIC is also \( O(M\log N) \). However, if the information of all \( N \) streams are available together, then we can reduce the computation time to \( O(M\log N) \), assuming \( N \) is a power of 2. Toward this end, in the first step, we can form \( N/2 \) pairs and perform \( N/2 \) pairwise SSIC in parallel using multiple threads. After that, we have \( N/2 \) streams, and we then perform \( N/4 \) pairwise SSIC in parallel, and so on and so forth. For further speedup, both SD and SSIC could be implemented in hardware rather than software.

VII. CONCLUSION

Existing duplicate-transmission standards, such as PRP in IEEE 802.11 or IEEE 802.1CB in the upcoming IEEE 802.11 be, lack a joint processing mechanism for duplicates, foregoing an effective way to boost reliability. This article puts forth an SSIC framework that combines the soft information of the duplicates to effect highly reliable communication. The SSIC framework contains two salient components.

1) A soft descrambler (SD) that minimizes the bit-error rate (BER) and PER at the SSIC’s output—SD yields the “soft” masking sequence required todescramble the

| PC A Location | Receiver | USRP 1 | USRP 2 |
|---------------|----------|-------|-------|
| L5            | 31.15 dB | 17.94 dB |
| L6            | 24.14 dB | 23.83 dB |
| L7            | 23.29 dB | 26.12 dB |
| L8            | 24.96 dB | 25.76 dB |
| L9            | 23.93 dB | 26.53 dB |
| L10           | 18.85 dB | 31.72 dB |

Fig. 18. BER comparisons among five different combining methods (method in [23], method in [25], SSIC naïve, SSIC with HRSX, and SSIC with SRSX) when \( \text{L} = 16 \).
soft information properly to minimize BER and PER of SSIC.

2) An SSIC networking architecture readily deployable over today’s TCP/IP networks without specialized NICs—The architecture allows multiple streams to be grouped and exposed as one single virtual link to the application for TCP/IP communication without the need to modify legacy TCP/IP applications.

We set up an SSIC network testbed over Wi-Fi. Experimental results indicate that with two streams over two paths in a noisy and lossy wireless environment, the SSIC network can: 1) decrease the PER and PLR by more than fourfold compared with a single-stream network and 2) provide 99.99% reliable packet delivery for short-range communication.

We believe that SSIC can be easily incorporated into Wi-Fi 7 to enhance the performance of its multilink mode. Furthermore, SSIC can be also deployed in a heterogeneous networking setup in which the different paths are established over wireless networks of different types (e.g., Wi-Fi and WiMAX).

ACKNOWLEDGMENT

The authors acknowledge the invaluable assistance of Prof. He (Henry) Chen and thank Dr. Jiaxin Liang for his constructive suggestions on the prototype development.

REFERENCES

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, “Link-level measurements from an 802.11b mesh network,” in Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun., 2004, pp. 121–132.

[2] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan, “Measurement-based characterization of 802.11 in a hotspot setting,” in Proc. ACM SIGCOMM Workshop Exp. Approaches Wireless Netw. Design Anal., 2005, pp. 5–10.

[3] B. Gokalpandhi, M. Tavares, D. Sandradzija, I. Seskar, and H. Gacanin, “Reliable low-latency Wi-Fi mesh networks,” IEEE Internet Things J., vol. 9, no. 6, pp. 4533–4553, Mar. 2022.

[4] T. R. Wanasinghe, R. G. Gosine, L. A. James, G. K. Mann, O. De Silva, and P. J. Warrian, “The Internet of Things in the oil and gas industry: A systematic review,” IEEE Internet Things J., vol. 7, no. 9, pp. 8654–8673, Sep. 2020.

[5] P. Park, P. Di Marco, J. Nah, and C. Fischione, “Wireless avionics intracommunications: A survey of benefits, challenges, and solutions,” IEEE Internet Things J., vol. 8, no. 10, pp. 7745–7767, May 2021.

[6] S. Savazzi, V. Rampa, and U. Spagnolini, “Wireless cloud networks for the factory of things: Connectivity modeling and layout design,” IEEE Internet Things J., vol. 1, no. 2, pp. 180–195, Apr. 2014.

[7] E. M. Ar-Reyouchi, K. Ghomdu, D. Ar-Reyouchi, S. Rattal, J. Zaitoun, A. Samet, and A. Elmazria, “Protocol wireless medical sensor networks in IoT for the efficiency of Healthcare,” IEEE Internet Things J., vol. 9, no. 13, pp. 10693–10704, Jul. 2022.

[8] Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor, and B. Vucetic, “High-reliability and low-latency wireless communication for Internet of Things: Challenges, fundamentals, and enabling technologies,” IEEE Internet Things J., vol. 6, no. 5, pp. 7946–7970, Oct. 2019.

[9] Industrial Communication Networks—High Availability Automation Networks. Part 3: Parallel Redundancy Protocol (PRP) and High-Availability Seamless Redundancy (HSR), IEC Standard 62439-3:2021, 2021.

[10] G. Cena, S. Scanzio, and A. Valenzano, “Experimental evaluation of seamless redundancy applied to industrial Wi-Fi networks,” IEEE Trans. Ind. Informat., vol. 13, no. 2, pp. 856–865, Apr. 2017.

[11] E. Khorov, I. Levitsky, and I. F. Akyildiz, “Current status and directions of IEEE 802.11be, the future Wi-Fi 7,” IEEE Access, vol. 8, pp. 88664–88688, 2020.

[12] “Improving WLAN reliability.” IEEE, Piscataway, NJ, USA, document IEEE 802.11i-11-19-1223-00-00be-improving-wlan-reliability-joint-tsn-11be-session.pdf, 2019. Accessed: Mar. 28, 2022. [Online]. Available: https://mentor.ieee.org/802.11/dct/19/11-19-1223-00-00be-improving-wlan-reliability-joint-tsn-11be-session.pdf

[13] D. Cavalcanti and G. Venkatesan, IEEE 802.1 Plenary: 802.1 TSN Over 802.11 with Updates From Developments in 802.11be, Intel, Mountain View, CA, USA, 2020.

[14] K. C.-J. Lin, N. Kushman, and D. Katabi, “ZipTs: Harnessing partial packets in 802.11 networks,” in Proc. 14th ACM Int. Conf. Mobile Comput. Netw., 2008, pp. 351–362.

[15] IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11n-2012 (Revision of IEEE Std 802.11n-2012), 2016.

[16] “Tun/tap interface tutorial.” Accessed: Mar. 28, 2022. [Online]. Available: http://backreference.org/2010/03/26/tun.tap-interface-tutorial/

[17] J. Hagenauer and P. Hoeher, “A Viterbi algorithm with soft-decision outputs and its applications,” in Proc. IEEE Global Telecommun. Conf. Exhibit. Commun. Technol., 1989, pp. 1680–1686.

[18] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate (corresp.),” IEEE Trans. Inf. Theory, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[19] T. Volkhausen, K. Schinköthe, and H. Karl, “Quantization techniques for accurate soft message combining,” in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), 2012, pp. 575–580.

[20] “USRP X310 device.” Accessed: Mar. 28, 2022. [Online]. Available: https://www.ettus.com/all-products/x310-kit/

[21] IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11n, 2007.

[22] “Linux tun/tap wrapper for Python.” Accessed: Aug. 26, 2022. [Online]. Available: https://stny.io/advisor/python/python-python

[23] A. Miu, H. Balakrishnan, and C. E. Koksal, “Improving loss resilience with multi-radio diversity in wireless networks,” in Proc. 11th Ann. Int. Conf. Mobile Comput. Netw., 2005, pp. 16–26.

[24] K. Jamieson and H. Balakrishnan, “PPR: Partial packet recovery for wireless networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 409–420, 2007.

[25] G. R. Woo, P. Krohadpour, D. Shen, and D. Katabi, “Beyond the bits: Cooperative packet recovery using physical layer information,” in Proc. 13th Annu. ACM Int. Conf. Mobile Comput. Netw., 2007, pp. 147–158.