Stateful Inter-Packet Signal Processing for Wireless Networking

Shangqing Zhao[†], Zhengping Luo[†], Zhuo Lu[†], Xiang Lu[‡], and Yao Liu[†]

[†] University of South Florida, Tampa FL, USA.
[‡] Institute of Information Engineering, Chinese Academy of Sciences (CAS), and University of CAS, Beijing, China.
Emails: {shangqing@mail., zhengpingluo@mail., zhuolu@}usf.edu, luxiang@iie.ac.cn, yliu@cse.usf.edu.

ABSTRACT

Traditional signal processing design (e.g., frequency offset and channel estimation) at a receiver treats each packet arrival as an independent process to facilitate decoding and interpreting packet data. In this paper, we enhance the performance of this process in the wireless network domain. We propose STAteful inter-Packet signaL procEssing (STAPLE), a framework of *stateful* signal processing residing between the physical and link layers. STAPLE transforms the signal processing procedure into a lightweight stateful process that caches in a small-sized memory table physical and link layer header fields as packet state information. The similarity of such information among packets serves as prior knowledge to further enhance the reliability of signal processing and thus improve the wireless network performance. We implement STAPLE on USRP X300-series devices with adapted configurations for 802.11a/b/g/n/ac and 802.15.4. The STAPLE prototype is of low processing complexity and does not change any wireless standard specification. Comprehensive experimental results show that the benefit from STAPLE is universal in various wireless networks.

KEYWORDS

Wireless networking; Signal processing; Estimation and decoding; Wireless link performance; Software radio system and protocol

1 INTRODUCTION

In a wireless network, signal processing is an essential procedure at a receiver during packet reception. It includes three major steps: timing/frequency synchronization, channel estimation, and equalization [1–4]. These together are generally treated as an independent process, serving as the foundation towards data decoding and higher layer protocol management [5–10].

This paper demonstrates that leveraging common protocol information across network packets can substantially improve the signal processing performance in a wireless network. Our motivation is that in a realistic network scenario, packets are not transmitted with uniformly random in-packet settings, such as date rate, packet size, source and destination addresses in the physical (PHY) or medium access control (MAC) fields. There exist common PHY and MAC header fields across different packets. For example in a WiFi network: a station only receives data from the access point (AP), indicating that the packets it cares about always have the same source address; and packets for data-intensive applications usually have the same packet size (i.e., the maximum allowable size). Such commonness of in-packet settings can serve as prior knowledge to improve the signal processing performance.

Hence, if we take signal processing out of its traditional domain and place it in a practical wireless network, there is indeed a lot of prior knowledge that can be used to boost its performance. The underlying challenge is how we are able to harness all possible common information and piece everything together to re-design signal processing with limited overhead for practical use in the wireless network domain.

In this paper, we propose the STAteful inter-Packet signaL procEssing (STAPLE) framework, which is a generic design to improve the performance of signal processing for wireless networking. Rather than designing signal processing in its traditional domain, we make signal processing stateful within the network domain. In particular, the key design in STAPLE is that a receiver maintains a small-sized state table, whose entry includes bit values of selected PHY/MAC fields from successfully decoded packets, such as MAC addresses, packet length and data rate, depending on a standard. These bit values constitute the state of a packet.

When a new packet arrives and decoding error happens, STAPLE performs state association (i.e., matching the bit values of its header fields with a known entry to restore the packet to a previous state). If the association succeeds, STAPLE considers the state of the packet is known (i.e., a part, if not all, of PHY/MAC information is known). Then, such known knowledge serves for the same purpose of "training sequence" (also known as "preamble" in many standards [11, 12]). Therefore, the restored state is combined with the original preamble in the packet to form a longer preamble. STAPLE re-performs signal processing on this longer preamble. Because more "prior" information is fetched for signal processing, STAPLE can obtain better frequency and channel estimates, which helps packet decoding and accordingly improves the network link performance.

To the best of our knowledge, there is no comprehensive system study in the literature to utilize the "prior knowledge" extracted

Corresponding authors: Zhuo Lu (University of South Florida) and Xiang Lu (CAS).

MobiCom'17, October 16–20, 2017, Snowbird, UT, USA.
© 2017 ACM. ISBN 978-1-4503-4916-1/17/10…$15.00
DOI: http://dx.doi.org/10.1145/3117811.3117846

from packet headers to improve the signal processing performance in wireless networks. We propose STAPLE as a general software radio system framework, then explore the feasibility and evaluate effectiveness of such a framework. In particular, we design and implement STAPLE with adapted configurations for 802.11a/b/g/n/ac and 802.15.4 on USRP X300 devices. The features of STAPLE are as follows: (i) lightweight processing and low overhead with a very small state table size (as experimental results suggest that it is sufficient to store 2-5 states at a station and 12-20 at the AP); (ii) a universal framework that benefits a wide range of wireless networks; (iii) no modification to any wireless standard; (iv) orthogonality to existing packet data decoding mechanisms (e.g., partial packet recovery [13–15]).

We conduct comprehensive experiments to show the benefits of STAPLE for different wireless standards. Our main contributions are summarized as follows: (i) We propose to improve signal processing within the network domain. We introduce the concept of the state of a packet as the bit values of selected PHY and MAC fields, and demonstrate that state-associating a packet to a previous state and re-performing signal processing in STAPLE improve the wireless network performance. (ii) We design, implement and configure the STAPLE prototypes for 802.11b/g/n/ac and 802.15.4. STAPLE is lightweight, effective and brings universal performance benefit to the packet reception process for wireless networks. Comprehensive experimental results show that STAPLE improves the packet delivery ratio by up to 20.8% under various conditions.

2 PRELIMINARIES

In this section, we describe preliminaries about signal processing and packet decoding in wireless networks.

2.1 Packet Specifications in Wireless Networks

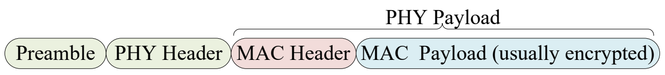

Figure 1: Generic packet structure viewed at the PHY and MAC layers.

In today's wireless networks, elements for data exchange are packets that contain both standard-specified information and user data. Figure 1 illustrates a generic packet format viewed at the PHY and MAC layers. As shown in Figure 1, a packet consists of the preamble, the PHY header, and the PHY payload that includes the MAC header and payload.

The preamble is specified in a wireless standard and known to the public. It is also called as pilot or training sequence [16] for energy detection, timing/frequency synchronization and channel estimation [17]. The PHY header usually provides specific PHY layer information for decoding the PHY payload. Depending on a wireless standard, such information may include packet length, modulation type, and data rate. The PHY header may also contain a checksum field, such as cyclic redundancy check (CRC) in 802.11b, to verify the successful decoding of the PHY header. The MAC header facilitates the functionality of coordinating multi-access of network nodes to share the wireless channel. It generally includes the source and destination MAC addresses, and packet type information, such as acknowledgement (ACK), beacon, and data packets in 802.11a/b/g/n/ac [11]. MAC payload is the actual data at the MAC layer, including high-layer protocol information and the user's data at the application layer. In today's wireless networks, this payload is usually encrypted [11, 12].

When a packet is transmitted by a sender to a receiver in a wireless network, the increase of the energy level at the receiver triggers the packet reception process.

2.2 Packet Reception in Wireless Networks

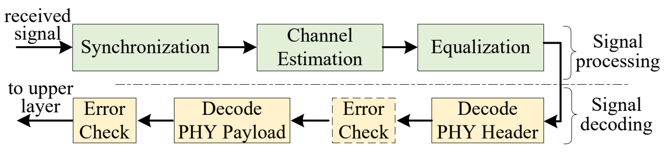

Figure 2: Packet reception at a receiver.

The packet reception process consists of two relatively independent procedures: signal processing and signal decoding, as shown in Figure 2. The signal processing procedure is used to prepare all necessary pre-decoding steps before decoding the packet data at the receiver. These steps generally happen in sequence as synchronization, channel estimation and equalization. The signal decoding procedure is to first decode the PHY header to obtain essential PHY information, which is then used to decode the PHY payload, including the MAC header and the MAC payload.

Signal Processing: Because the radio wave propagating through the wireless channel is a complicated phenomenon characterized by various environmental factors, such as the multi-path fading and the doppler effect [17], the signal processing procedure estimates and compensates those factors before the signal is decoded into data bits. Generally, signal processing contains three major steps: synchronization, channel estimation, and equalization, as shown in Figure 2. All these steps are based on the known preamble with length n in the packet, denoted by $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, where x_i is the i-th baseband symbol in the preamble.

Synchronization includes timing synchronization and frequency synchronization. Timing synchronization is triggered by energy detection, and is to find the start position of the first PHY layer symbol in a packet [18] such that the later processing on the symbols in the packet is aligned. After timing synchronization, the received signal of the preamble in the packet is represented as a vector $\mathbf{s} = [s_1, s_2, \cdots, s_n]$. Frequency synchronization is to estimate and compensate the frequency offset Δf between the transmitter and the receiver [17] from the received preamble signal $\mathbf{s}$ based on the knowledge of the preamble $\mathbf{x}$.

After frequency offset compensation, the received signal vector of the preamble $\mathbf{s} = [s_1, s_2, \cdots, s_n]$ has been compensated to a new vector $\mathbf{s'} = [s'_1, s'_2, \cdots, s'_n]$ [4], based on which channel estimation is performed to estimate the channel state information (CSI).

The maximum-likelihood (ML) algorithm that minimizes the estimation error on the flat-fading channel [17] is

$$\hat{h} = (\mathbf{x}^H \mathbf{x})^{-1} \mathbf{x}^H \mathbf{s}', \quad (1)$$

where $(\cdot)^H$ and $(\cdot)^{-1}$ denote the matrix conjugate transpose and inverse, respectively. ML channel estimation for frequency-selective channels [3], orthogonal frequency-division multiplexing (OFDM) [1] or multiple-input and multiple-output (MIMO) channels [19] are performed in a similar linear[1] process to (1).

Once the receiver obtains the estimated CSI $\hat{h}$, it performs channel equalization [20] to compensate the channel effect to support coherent demodulation. Note that OFDM has been widely adopted in today's wireless standards, such as in 802.11a/g/n/ac [11], because it significantly simplifies the channel equalization process by transforming a frequency-selective wireless channel into multiple parallel flat-fading ones.

Signal Decoding: Signal decoding follows immediately once signal processing is finished. As a packet may have different modulation and coding schemes in the PHY header and payload (e.g., in 802.11a/g/n/ac), the receiver first demodulates and decodes the PHY header and obtains the data rate information (i.e., the modulation and error-correction coding information) about the PHY payload, along with other necessary information. As shown in Figure 2, if a checksum (e.g., CRC or parity) exists in the PHY header and the decoded header does not pass the error check, the receiver has to drop the packet (when there is no other error recovery option); otherwise, the receiver uses the data rate information in the PHY header to demodulate and decode the PHY payload, obtaining the MAC header and the MAC payload. If the receiver verifies the checksum of the MAC payload, it passes the payload data to the upper layer for protocol management and data delivery.

Demodulation and error-correction decoding during the signal decoding process have been well studied in the wireless communication domain [21, 22].

3 DESIGN MOTIVATION BEHIND STAPLE

In this section, we introduce the intuition behind stateful signal processing, then use real-world measurements to validate our intuition. Finally present our basic design of the STAPLE mechanism.

3.1 Basic Intuition and Observations

Traditionally, the entire packet reception is treated as an independent process at a receiver. Recent studies have focused on enhancing the signal decoding procedure to improve wireless link performance, such as partial packet recovery [7, 9, 14, 15, 23, 24]. Little attention has been focused on re-designing the signal processing procedure for wireless packet reception, even in the signal processing community, because synchronization, channel estimation and equalization are all well explored in the literature [1, 2, 4, 16, 17, 25].

In this paper, we take a closer look at the signal processing procedure. Intuitively, such a procedure must be treated as an independent process for each packet arrival. For example, upon arrival of a packet, the receiver estimates the frequency offset and the CSI from the preamble, compensates their effects on the received signal

[1]Note that although the matrix inverse operation in (1) is nonlinear, the preamble $\mathbf{x}$ is known to the public and accordingly $(\mathbf{x}^H\mathbf{x})^{-1}\mathbf{x}^H$ in (1) can be pre-computed.

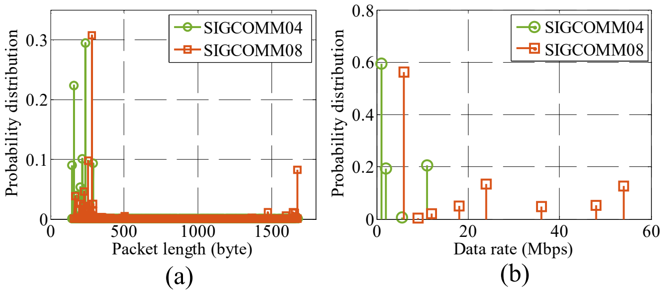

Figure 3: Distributions of (a) packet length and (b) data rate in SIGCOMM04/08 datasets.

for decoding. As the frequency offset and the CSI are time-varying and hard to predict [4], the receiver has to estimate and compensate their effects again when a new packet arrives.

It is difficult to further improve a well-established signal processing algorithm in practice. Our design intuition is that if we place signal processing in the network domain, we should be able to boost its performance if we can leverage the common information across different packets. In a wireless network, packets indeed have some common information, which can serve as prior knowledge during packet arrival. We analyze the realistic packet trace datasets SIGCOMM04 [26] and SIGCOMM08 [27] to see whether common information exists between packets in typical WiFi network usages. Figure 3(a) shows the packet length distributions in the datasets. We observe that the distribution in each dataset is extremely uneven and consists of two regions: the first region includes small packet lengths (1-250 byes), indicating control or short data packets; and the second is around the maximum allowable packet size (1400-1600 bytes), indicating packets for data-intensive applications. Similarly, uneven distributions of data rates in packets are shown in Figure 3(b). Given an AP in SIGCOMM08, we also find that on average 4.2 nodes sent 51.7 packets to the AP within one-second period. Those packets have the same PHY/MAC field values with high probability. The quantitative evaluation on each of these fields will be detailed in Section 5.

3.2 Basic Design of STAPLE

The preliminary observations show that there exists common (but maybe scattered) information among packets, which can serve as prior knowledge for signal processing during packet reception. If we look at such prior information solely within the signal processing domain, a traditional way is to perform profiling/training among all PHY/MAC fields in collected packets to build the prior distributions of such fields, and then integrate these distributions into a maximum a posteriori (MAP) framework [17] to improve signal processing.

Nonetheless, such a method faces two practical issues: (i) Profiling empirical distributions is always practically cumbersome due to its heavy dependencies on variable factors, such as physical environments, the number of users, and how they use the network, all of which cannot be easily predicted from time to time. (ii) Different data fields in packets are very likely to exhibit distinct distributions and also have correlations. Integrating these distributions into a unified framework is challenging. Moreover, the design must be lightweight and efficient to ensure low overhead and timely signal processing.

Thus, we avoid designing this method solely in the signal processing domain. Rather, we look at it from the network perspective. A core idea in network management is to maintain a network state, such as the backoff state at the MAC layer or the congestion control state in TCP. When a sender transmits a packet, we can use its header field values to form its state. Formally, we define the state of a packet as follows.

Definition 3.1. The state of a packet is a bit sequence concatenated by one or more bit strings in the PHY and MAC header fields in the packet.

All PHY/MAC header fields in a packet can be combined to represent a state. In this way, the state space of a packet might be very large in theory. In practice, we can carefully select a few fields that are very likely to have the same value across packets to form the state (as indicated in our packet trace analysis). Hence, our motivation is to introduce state management into signal processing in a network context. Figure 4 draws the sketch of STAPLE, which consists of three major steps.

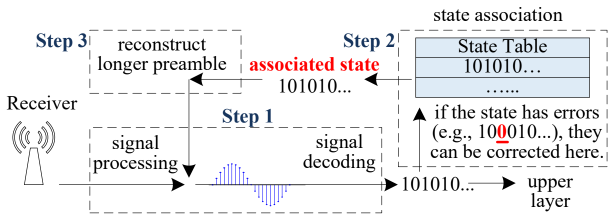

Figure 4: Sketch of STAPLE.

1) When a receiver receives a packet, it first goes through signal processing and decoding. If no error happens, the receiver can directly send the data to the MAC and higher layers, finishing the packet reception.

2) When errors happen, STAPLE tries to associate the corrupted packet with a previous state. Specifically, it compares the PHY/MAC fields in the corrupted packet with a state table that stores those fields from successfully decoded packets, which finds the closest state to the packet and recover it into that state. As shown in Step 2 of Figure 4, when the state of the packet has errors, they can be then corrected after state association; when the state contains no error and exactly matches a previous one, other parts of the packet, such as the packet payload, should have errors leading to decoding failure. In both cases, the state of the packet is associated.

3) At this point, the PHY/MAC header fields in the packet state can be considered as prior knowledge. This indicates that the information can be combined with the original preamble in the packet to form a longer preamble to re-perform the signal processing, which can yield better estimates for frequency offset and CSI (as estimation errors in signal processing generally decrease with the preamble length increasing [1, 2, 28, 29]). Then, the receiver re-performs the signal decoding using these better estimates. They can improve the performance of decoding the payload because it usually has a higher modulation/coding rate than the packet header, and accordingly requires more accurate frequency offset and CSI estimates.

STAPLE transforms prior information in a wireless network into the concept of packet state. In this way, prior information is recovered by state association, and is then used to construct a longer preamble. This new preamble is in turn used to re-do the signal processing to obtain better frequency offset and CSI estimates, which is called signal re-processing in this paper.

4 ARCHITECTURE AND COMPONENTS

In this section, we turn the basic concept of STAPLE into detailed design. We first present the STAPLE architecture, and then elaborate its key components.

4.1 STAPLE Architecture

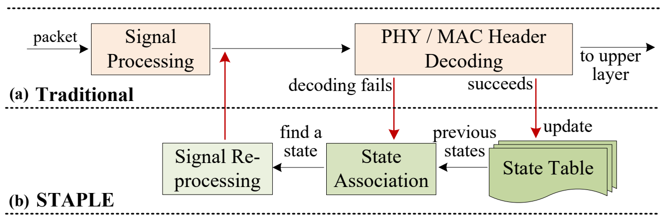

Figure 5: Architecture of STAPLE.

STAPLE can be regarded as a module (as shown in Figure 5(b)) flexibly attached to the traditional architecture of a wireless receiver (as shown in Figure 5(a)).

There are three key components in STAPLE: state association, signal re-processing, and state table management. To provide a generic framework, we decouple the design of STAPLE into two independent parts: (i) designing STAPLE components (i.e., state association and signal re-processing) and (ii) constructing the state of a packet from PHY/MAC fields. In the following, we describe the first part, which is standard-independent. The second part answers what PHY/MAC fields are used to construct the state of a packet and will be discussed based on wireless standards in Section 5.

4.2 State Association

State association is triggered when the decoding of a packet P fails. From Definition 3.1, we know that the state of a packet is the bit combination of several selected PHY/MAC fields. Thus, STAPLE can read the state of the packet $S(P)$ by combining the decoded bit values in certain PHY/MAC fields in P, as shown in Figure 6. Note that even upon failure of decoding an entire packet, the bit values in its PHY/MAC fields can still be decoded (with potential errors), and then fetched into state association for state look-up and comparison.

After obtaining $S(P)$, STAPLE associates it with a previous state in the state table, which stores a set of previous states, denoted by $\mathcal{S} = \{S_1, S_2, \cdots, S_{|\mathcal{S}|}\}$, where $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$.

As P is a corrupted packet, any bit in P can be erroneous. However, if we can find a state in $\mathcal{S}$ that resembles $S(P)$, it is very likely

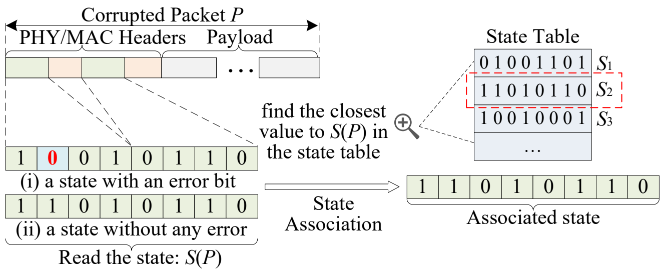

Figure 6: State association in STAPLE.

that $S(P)$ should be equal to that state, but may exhibit another value because of noise or interference during the reception of P. This is from our observations in packet trace analysis: a node is found to send packets to another node with limited variations of bit values in PHY/MAC headers, therefore exhibiting a limited set of most frequent states. If these states are all stored in the table, the state of a corrupted packet is very likely stored in the state table.

As a result, the state association component in STAPLE finds the stored state closest to $S(P)$ in the state table. The measure of "being closest" is by the Hamming distance, which is defined as the number of bit differences between two bit sequences. Accordingly, the association algorithm can be written as

$$\text{Objective}: \qquad \hat{S}(P) = \arg\min_{s \in \mathcal{S}} H(s, S(P)) \tag{2}$$

$$\text{Subject to}: \qquad H(S(P), \hat{S}(P)) < d_{\text{th}}, \tag{3}$$

where $H(\cdot,\cdot)$ denotes the Hamming distance between two bit sequences, and the objective (2) is to re-associate the packet P with a new state $\hat{S}(P) \in \mathcal{S}$ that is the closest to the original state $S(P)$. A constraint (3) also exists to make sure that the Hamming distance between $\hat{S}(P)$ and $S(P)$ must be smaller than a threshold d_{th}. This is because the real state of P does not always exist in the state table. A large value of $H(S(P), \hat{S}(P))$ indicates that (i) packet P was transmitted with a completely different state that was never stored in the state table; (ii) too many bit errors happened because of low signal quality. In both cases, STAPLE stops state association and simply drops the packet P.

It is also possible that the corrupted packet P is associated to a wrong state $\hat{S}(P)$, which can happen when the Hamming distances among some pairs of states stored in the table are very small. STAPLE makes best efforts to associate P with a previous state. When P is indeed associated to a wrong state, STAPLE is not likely to recover P by signal re-processing. But this only reduces the gain from STAPLE and does not degrade the performance as P is already corrupted. Wrong association can be mitigated by carefully choosing PHY/MAC fields to construct the state of a packet, which is detailed in Section 5. A smaller threshold d_{th} can also mitigate wrong association from happening. We will evaluate the impacts of d_{th} in experiments in Section 6.

Figure 6 shows two simple examples of how state association works: (i) when a corrupted packet P has a state with one error bit $S(P) = 10010110$, state association is performed based on (2) with threshold $d_{\text{th}} = 2$ to obtain $\hat{S}(P) = S_2 = 11010110$ because their Hamming distance $H(S(P), S_2) = 1$. This indicates that the second bit in $S(P)$ is likely an error, and is corrected from 0 to 1. Then, the state of P is associated to a new one $\hat{S}(P) = 11010110$. (ii) When P has a state without any error bit $S(P) = 11010110$, obviously $\hat{S}(P) = S(P) = S_2 = 11010110$ after state association.

When the header is coded in some standards (e.g., convolutional coding is used for 802.11g/n/ac headers), state association can be performed on the coded data and thus the Hamming distance comparison will not be affected by the coding scheme.

Note that the performance benefits of STAPLE do not come solely from error-correcting a part of the PHY/MAC header in a packet during state association. More importantly, STAPLE helps improve the performance when the payload, not the header, is decoded with errors. This is because the payload usually has a higher modulation/coding rate than the header, therefore requires more accurate frequency offset and CSI estimates that STAPLE can provide based on a longer preamble formed by the conventional preamble and the state of a packet. Overall, state association serves as the first essential step for this generic framework to improve the signal processing performance.

4.3 Signal Re-processing

Given the corrupted packet P as the input, the output of state association is the associated state $\hat{S}(P)$. Then, STAPLE performs signal re-processing, which consists of two steps as shown in Figure 7.

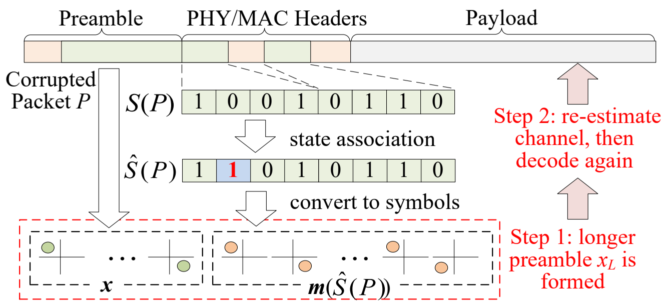

Figure 7: Signal re-processing in STAPLE.

First, STAPLE converts $\hat{S}(P)$ that represents the recovered bits from PHY/MAC fields in packet P to physical layer symbols, which is done by re-modulating $\hat{S}(P)$. If a header is interleaved or/and coded (e.g., 1/2 convolutional coding in WiFi signal), STAPLE re-interleaves or/and re-encodes, then re-modulates $\hat{S}(P)$. Denote by $\mathbf{m}(\hat{S}(P))$ the symbol vector of re-modulated symbols from $\hat{S}(P)$. The longer preamble is constructed as $\mathbf{x}_L = \left[\mathbf{x}\ \mathbf{m}(\hat{S}(P))\right]$, where $\mathbf{x}$ is the original preamble described in Section 2.2.

Second, given the newly constructed preamble $\mathbf{x}_L$, we can improve both frequency synchronization and channel estimation during signal processing, as shown in Figure 2: (i) Frequency synchronization for low complexity processing must leverage a special repetitive structure (e.g., two same consecutive symbols in 802.11 a/g/n/ac) in the preamble. The longer preamble $\mathbf{x}_L$ does not generally result in such a repetitive structure, leading to a non-convex optimization problem in [1, 2]. (ii) In contrast, channel estimation is usually a linear process (as discussed in Section 2.2). A longer preamble in general results in a more accurate CSI estimate.

As performing signal re-processing requires a tradeoff between performance and complexity, and we wish to ensure low complexity in the STAPLE implementation, we choose to perform signal re-processing only based on channel estimation, and avoid solving the non-convex optimization to re-estimate the frequency offset at the receiver. Thus, given $\mathbf{x}_L$, STAPLE re-estimates the CSI as

$$\hat{h}_L = (\mathbf{x}_L^H \mathbf{x}_L)^{-1} \mathbf{x}_L^H \mathbf{s}'_L, \quad (4)$$

where $\mathbf{s}'_L$ is the vector combined by the received symbols from the original preamble part and from the PHY/MAC fields in the received signal of packet P at the receiver.

After obtaining a better CSI estimate $\hat{h}_L$, STAPLE re-demodulates and re-decodes the corrupted packet P in order to recover it. The implementation for this step can be optimized to minimize the processing complexity for a wireless standard. We will analyze the implementation complexity for 802.11ac in Section 6.2.1.

If there are pilot signals in the payload of the packet P (e.g., pilot subcarriers in 802.11 a/g/n/ac), because more information is obtained by STAPLE, it can yield a better initial channel estimate $\hat{h}_L$, which can be based on to track the channel change using these pilots [16] in a fast fading wireless channel environment.

Although $(\mathbf{x}_L^H \mathbf{x}_L)^{-1}$ is not a linear operation in (4), in OFDM equalization for many wireless standards, $\mathbf{x}_L^H \mathbf{x}_L$ is a complex number and $(\mathbf{x}_L^H \mathbf{x}_L)^{-1}$ only incurs a division operation in the complex domain. In addition, the value of $(\mathbf{x}_L^H \mathbf{x}_L)^{-1}$ can always be pre-computed during state update and stored along with its associated state in the state table.

4.4 Table Management Policies

The state table (shown in Figure 5(b)) is used to store the states of successfully decoded packets. We design two table update policies.

1) Timestamp-based policy, which uses packet arrival time as an indicator to update the table. Each state is stored with a timestamp. Once the most recent packet is decoded successfully, STAPLE updates the state's timestamp if the state is already in the table, or otherwise replaces the oldest state with this new one.

2) Frequency-based policy, which stores a timestamp and a usage count with each state in the table. Similar to the previous policy, the timestamp is used to indicate the last update time of a state. Once a packet is decoded, STAPLE updates the state's timestamp and increments the usage count if the state is already in the table. Otherwise, STAPLE replaces the state that has the smallest count with the new state. If several states have the same count number, STAPLE chooses the oldest state for replacement. This policy also zeros out the count of a state if the state's timestamp is too old (i.e., larger than a given threshold).

When the table size is sufficiently large, the two policies should have no evident performance difference. However, if the table size is small, the downlink performance may not be affected significantly because a station, as the receiver, always receives data solely from the AP. The uplink performance of the station that transmits more data (than others) to the AP may be boosted because the state of its packets can always be stored in the AP's table. Therefore, the AP should choose a larger table size than a station in practice. The impacts of state table parameter configurations and policy setups will be evaluated in experiments in Section 6.

5 CONFIGURING THE STAPLE SYSTEM

We have designed the major components in STAPLE. Another key question left is which PHY/MAC fields constitute the state of a packet, which is standard-dependent. In this section, we configure STAPLE for 802.11a/b/g/n/ac and 802.15.4 by specifying how to construct the state of a packet in such networks.

5.1 Methodology for Configurations

The motivation to develop STAPLE is from the observation that there exist most frequently appeared values of PHY/MAC header fields in packets. Thus, we aim to select those fields exhibiting little randomness to form the state of a packet. In particular, we use Shannon entropy [30] per bit to quantitatively measure the randomness of each PHY/MAC field in packet trace data and choose those fields with low entropies. We choose the SIGCOMM04/08 datasets for 802.11b/g and our own datasets STAPLEn/STAPLEac for 802.11n/ac, respectively. These datasets can represent typical WiFi scenarios in a conference, a campus or a residential place.

STAPLEn is a dataset collected from a campus WiFi network and STAPLEac from a residential WiFi network. Unlike 802.11b/g packet capturing, it is not always possible for a third-party to correctly capture and decode all 802.11n/ac data packets because of beamforming between individual stations and the AP [31]. Hence, we place four laptops at different locations and capture packets individually in the networks. We then aggregate all packets to measure the entropy of each header field. We find that the 802.11ac network did not run in the multi-user MIMO (MU-MIMO) mode because the GroupID field in all collected packets was set to 0. Thus, our collection and measurement can reflect how PHY/MAC fields in packets are set up for routine WiFi use in a non-MU-MIMO 802.11ac network.

Note that wireless standards adopt different coding schemes for the PHY/MAC headers, which can affect the complexity of the signal re-processing component in STAPLE. We categorize a coding scheme into one of three types: plaintext (P), interleaving and coding (I/C) data, and encryption (E). Type-P indicates no coding is used for a header field (e.g., PHY/MAC headers of 802.11b); Type-I/C means that data fields in a header are interleaved and error-correction coded (e.g., MAC headers of 802.11a/g/n/ac); Type-E indicates that data fields in a header are all encrypted (e.g., MAC headers of 802.15.4). Generally, we avoid choosing Type-E fields and select Type-P or Type-I/C fields based on their entropy values for our implementation.

Constructing the state of a packet is not unique. In our STAPLE configurations for 802.11a/b/g/n/ac and 802.15.4, we target a lightweight design that balances between performance benefits and costs of storage and computation in typical wireless networks with normal usages. It is certainly feasible to add additional fields to (or remove existing fields from) our configurations to optimize STAPLE along certain direction.

Because a newer 802.11 standard usually extends and relies on previous ones, we configure STAPLE starting from 802.11b (legacy) to 802.11ac (state-of-the-art) for the sake of clear presentation. Our implementation and evaluation are focused on 802.11ac in Section 6. We also minimize the description of a wireless standard, which is well documented.

5.2 Configurations for Wireless Standards

Configurations for 802.11b: We start from 802.11b, which is a legacy WiFi standard but still backward supported in many WiFi networks. Figure 8 shows a typical structure of physical protocol data unit (PPDU) for 802.11b, including the PHY Layer Convergence Procedure (PLCP) preamble, PLCP header, and PLCP Service Data Unit (PSDU) that contains the MAC header and the MAC payload.

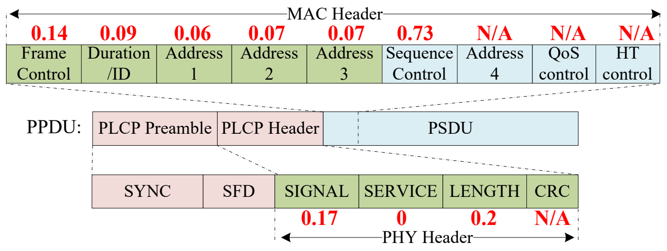

Figure 8: Typical format of an 802.11b data packet and measured entropies of field values inside PHY and MAC headers.

As shown in Figure 8, the PLCP header represents the PHY header including four fields of Type-P: SIGNAL, SERVICE, LENGTH, and CRC, where SIGNAL denotes the modulation scheme, LENGTH is the payload length, SERVICE contains three bits to represent the PHY configuration and the rest of bits are reserved and default to 0. The 802.11b MAC header at the beginning of PSDU consists of a set of fields.

According to the measured entropy in each PHY/MAC field in Figure 8, we choose SIGNAL, SERVICE, LENGTH at the PHY header and Frame Control, Duration/ID, Addr.1, Addr.2, and Addr.3 in the MAC header to form the state of a packet because of their low entropies. We also include CRC in the PHY header because CRC directly depends on other fields. Once these fields are known, we can simply calculate the CRC. Including CRC increases the length of the state of a packet, and accordingly can further improve the signal processing reliability because more data is used for signal processing. Thus, for a packet P, its state $S(P)$ is written as $S(P) =$ SIGNAL|SERVICE|LENGTH|CRC|Frame Control|Duration/ID| Addr.1|Addr.2|Addr.3 in our STAPLE configuration for 802.11b.

Configurations for 802.11a/g: 802.11a/g uses OFDM to combat multipath fading. Figure 9 shows a typical format of the PPDU including the PLCP preamble, PLCP header, PSDU and others. Different from 802.11b, all the data except for the PLCP preamble is of Type-I/C in an 802.11a/g packet.

The PLCP header includes the SIGNAL field that occupies one single OFDM symbol and the Service field that exists in the next symbol. SIGNAL contains a number of fields of Type-I/C to represent packet information like the data rate and packet length. We

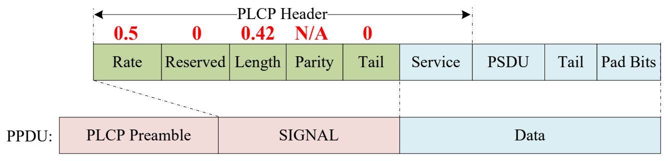

Figure 9: Typical structure of an 802.11a/g packet and measured entropies in SIGNAL.

measure the entropy of each field in SIGNAL, also shown in Figure 9. All entropies are small therefore we include the entire SIGNAL field in the state of a packet. Note that the Parity field can be directly computed given other fields, therefore we do not need to measure its entropy.

The Service field is used to synchronize the descrambler shown in Figure 9. All bits in Service are constantly set to 0. The MAC header at the beginning of PSDU follows Service. The 802.11a/g MAC header structure is exactly the same as that of 802.11b. However, Service and the MAC header together do not exactly occupy one OFDM symbol. This means that they are interleaved and coded with other data then OFDM-modulated, which can be difficult (if not impossible) for STAPLE's signal re-processing to accurately extract Service and parts of the MAC header from interleaved and coded OFDM symbols. Therefore, Service and all MAC fields can be used for state association, but not for signal re-processing in STAPLE. As we aim at a lightweight design, we avoid using any of these fields to reduce the size of the state table. Therefore, the state $S(P)$ for packet P in our STAPLE configuration for 802.11a/g is $S(P) =$ SIGNAL.

Configurations for 802.11n/ac: 802.11n and 802.11ac are the state-of-the-art WiFi standards leveraging MIMO technologies with more complicated packet formats.

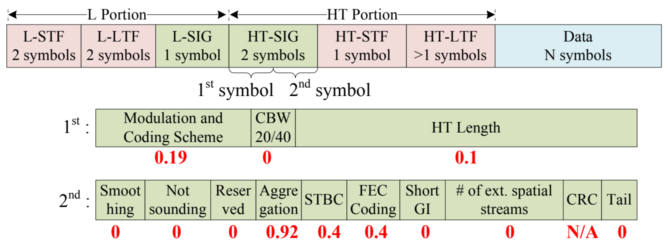

Figure 10: Typical packet format in 802.11n and measured entropies of field values inside HT-SIG.

Figure 10 depicts a typical packet structure of 802.11n which contains two portions: legacy (L) and high throughput (HT). The L portion is for backward compatibility, and the HT portion is specified to support MIMO communication. The L-SFT and L-LFT fields at the beginning of the L portion are legacy preambles. The HT-SFT and HT-LFT fields in the HT portion are preambles for MIMO training. The L-SIG of Type-I/C is exactly the same as the SIGNAL field in 802.11a/g. Hence, we use the entire L-SIG field for STAPLE. HT-SIG of Type-I/C is at the beginning of the HT portion and consists of 2 OFDM symbols with a number of fields. We measure the entropy of each field from packet trace data, as shown in Figure 10. Most fields have small entropies, therefore we include the entire HT-SIG field. Note that the single-bit Aggregation field indicates whether multiple payloads are combined together into a single packet with a unique PHY header, and it is almost random with entropy close to 1. However, we cannot exclude this bit because it is interleaved and coded with other fields in exactly one OFDM symbol. We find that the high entropy of this single bit always happens in the downlink. It does not have substantial impact on the performance but the size of the state table needs to be slightly increased at the downlink. Therefore, we still include it to represent the state of a packet. As a result, the state $S(P)$ for packet P in our configuration for 802.11n is $S(P)$ = L-SIG|HT-SIG.

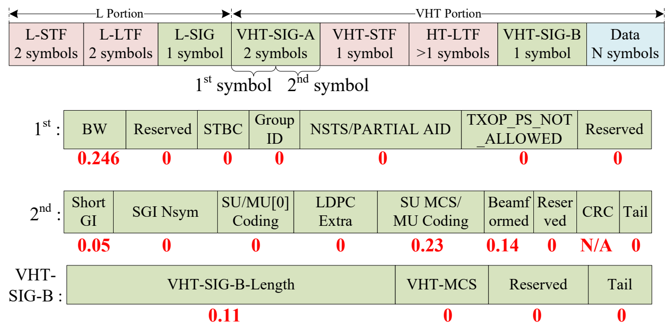

Figure 11: Typical packet format in 802.11ac and measured entropies of field values inside VHT-SIG-A and VHT-SIG-B.

Figure 11 plots the typical packet format in 802.11ac, which has a very similar structure to 802.11n: the HT portion becomes the very HT (VHT) portion. Similarly, based on the measured entropies from packet trace data shown in Figure 11, we include L-SIG and VHT-SIG to represent the state of a packet. In addition, we find the entropies of fields in the VHT-SIG-B are also small. Thus, we set the state as $S(P)$ = L-SIG|VHT-SIG|VHT-SIG-B in our configuration for 802.11ac.

Configurations for 802.15.4: IEEE 802.15.4 is the basis for ZigBee towards low-cost, low-speed ubiquitous communication. We configure STAPLE for offset-QPSK (O-QPSK) based ZigBee, whose typical packet format is shown in Figure 12.

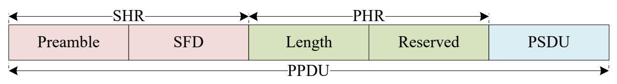

Figure 12: Typical 802.15.4 packet format.

As shown in Figure 12, the PHY header (PHR) of 802.15.4 contains only two Type-P fields: Length and Reserved, which indicate the length of the packet and reserved bits (set to all 0s), respectively. The MAC header is contained in PSDU, which is however of Type-E. To reduce the complexity, our STAPLE configuration avoids any Type-E data during signal re-processing, therefore choosing to include only Length and Reserved for the state of a packet P, i.e., $S(P)$ = Length|Reserved.

6 EXPERIMENTAL EVALUATION

In this section, we implement STAPLE with adapted configurations for 802.11b/g/n/ac and 802.15.4. We first describe our implementation and experiment setups, then focus on presenting experimental evaluation of STAPLE in 802.11ac, and finally describe experimental results on other standards.

6.1 Implementation and Setups

Platform and Implementation: STAPLE requires modifications of a conventional wireless receiver. We choose the X300 USRP with CBX daughterboards [32] as the implementation platform. We synchronize multiple USRPs using OctoClock-G [32] when performing MIMO experiments with 2 - 8 antennas.

We implement a generic software radio platform that covers the basic designs of 802.11b/g/n/ac and 802.15.4 transceivers. We adopt BPSK, QPSK, and 16QAM for 802.11b/g/n/ac, and O-QPSK for 802.15.4. For 802.11n/ac, we implement the Alamouti code [33] based MIMO transmission schemes. In 802.11g/n/ac implementations, the header and data payload of a packet use 1/2 and 3/4 convolutional coding schemes, respectively. We also implement a CSMA/CA scheme with uniform random backoff (i.e., the contention window for random backoff is always fixed [34, 35]) at the MAC layer for all standards. Note that it is challenging and time-consuming to implement a comprehensive USRP-based system fully compatible with all 802.11 PHY/MAC standards. To balance the efficiency of fast prototyping and the cost of high-fidelity evaluation, our implementation is a proof of concept one with a subset of fundamental STAPLE-related PHY functionalities, serving the purpose of evaluating the benefits of STAPLE brought to practical wireless scenarios with different system setups and conditions. As mentioned in Section 4.3, our STAPLE implementation does not improve frequency offset estimation but improves the channel estimation in signal re-processing.

Also note that we implement a subset of modulation and coding schemes. However, our experiments generate packets according to packet trace data. This means the PHY field that denotes the data rate in some generated packets may indicate a higher rate scheme that we do not implement. In this case, we choose 16QAM as the scheme to modulate and demodulate. This does not give an advantage to STAPLE during performance evaluation, since we still allow all possible values in the data rate field of a packet, which affect state association and state table update of STAPLE in practice.

Experimental setups: We conduct experiments in a realistic indoor environment shown in Figure 13 as the scenario reflects how STAPLE can help improve the wireless link performance in an office environment with typical WiFi usages. Network nodes are placed at different locations for performance evaluation with limited transmit power. In our network experiments, packets are generated with PHY/MAC field values according to the SIGCOMM04/08 and STAPLEn/ac datasets.

In STAPLE, the default threshold for state association (2) is set to 12. The default size of the state table for a station in the downlink

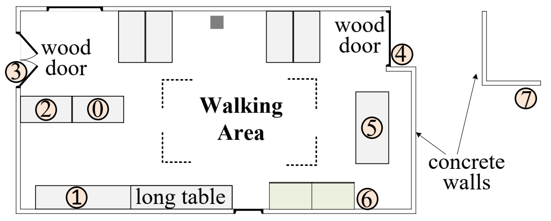

Figure 13: Environment for experiments.

is set to 5; and that for the AP in the uplink is 15. The default table update policy is frequency-based. In experiments, we change table setups to evaluate the impact of STAPLE parameters on the performance. We use the construction of the packet state for each standard given in Section 5.

Performance Metrics: Packet dropping due to decoding errors only happens at the PHY layer and STAPLE is a method to improve the PHY layer performance. Thus, it is natural to use the performance metric at the PHY layer to directly see the benefits of STAPLE without involving other potential affecting factors at higher layers. In experiments, we measure the packet delivery ratio (PDR) with and without STAPLE, where PDR is defined as the ratio of the number of successfully decoded PHY packets at a node to the total number of PHY packets transmitted to the node. We measure the benefit of STAPLE by using the performance gain ratio, which is defined as

$$\text{Performance gain ratio} = \frac{\text{PDR with STAPLE}}{\text{PDR with traditional design}} - 1.$$

6.2 STAPLE for 802.11ac Networks

We focus on performance evaluation on 802.11ac because it is a state-of-the-art WiFi standard. We consider two types of scenarios in experiments: single-link evaluation, in which a two-antenna transmitter sends packets to a receiver with 2 – 8 antennas; (ii) network evaluation, in which we set up two clock-synchronized USRPs as a single 802.11ac AP, and six USRPs as stations communicating with the AP.

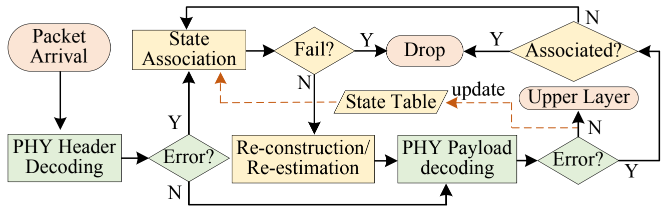

Figure 14: Implementation of 802.11ac.

6.2.1 *Implementation Complexity for 802.11ac.*

Figure 14 shows the flowchart of our STAPLE implementation for 802.11ac. the extra storage due to STAPLE is mainly the memory used for the state table, whose size is 5 states for stations and 15 states for the AP as default values. In terms of extra processing complexity, if errors are detected in the PHY layer, extra procedures including state association, preamble reconstruction, and CSI re-estimation are introduced by STAPLE. All these procedures are linear, which will not incur much overhead. If errors occur in the data payload, re-decoding the payload is needed. The re-decoding complexity is not linear and depends on the decoding algorithm that is usually polynomial. However, thanks to the PHY payload check, this case does not quite often happen in today's WiFi networks.

Table 1: Complexity of STAPLE for 802.11ac.

Situation	**Extra processing**
1) No error in a packet	none
2) Error first detected in PHY header	state association, preamble reconstruction and CSI re-estimation
3) Error not detected in header but payload	extra processing in 2), and re-decoding of PHY payload

6.2.2 *Single-Link Performance Evaluation.*

In our single-link performance evaluation, we fix the modulation scheme for packets and measure the link performance between two nodes to evaluate how STAPLE helps improve the single-link performance.

Varying locations: We first compare the single-link performance between a traditional receiver and STAPLE at different locations. We fix the transmitter at location 1 as shown in Figure 13, and place the receiver at location 0, 3, 4, or 5, which represents the short-distance line of sight (S-LoS), short-distance Non-LoS (S-NLoS), long-distance NLoS (L-NLoS), or long-distance LoS (L-LoS) channel condition, respectively. The transmitter and receiver are both equipped with 2 antennas.

Figure 15 depicts the performance gain ratios at different locations under BPSK, QPSK, and 16QAM. We can observe that the performance of STAPLE is uniformly better than the performance of traditional signal reception for most cases. There is no performance improvement at locations 0 and 5 (both LoS cases) for the BPSK modulation because the packet delivery ratio is 100%. We also find that at location 4 (L-NLoS), STAPLE substantially improves the packet delivery ratio for 16QAM from 80% to 95% (a performance gain ratio of 17.6% shown in Figure 15). This is because communication with higher data rate is less error-tolerant, thus demanding more accurate CSI estimation, which STAPLE provides. We can conclude from Figure 15 that STAPLE improves the performance in NLoS cases more than that in LoS cases.

Varying packet lengths: We then evaluate the effect of packet length on the performance gain of STAPLE. During this experiment, the transmitter always sends packets with a fixed length. Table 2 shows the packet delivery ratios on a 2×4 MIMO link from location 1 to location 5 with QPSK modulation for varying packet lengths. It is noted from Table 2 that transmitting shorter packets results in better packet delivery ratios. For example, traditional signal reception and STAPLE achieve almost 100% packet delivery ratio when the packet length is no larger than 100 bytes. As the packet length increases, STAPLE gradually outperforms traditional signal reception, and improves the packet delivery ratio

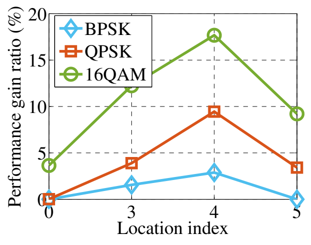

Figure 15: Performance gain ratios under different locations.

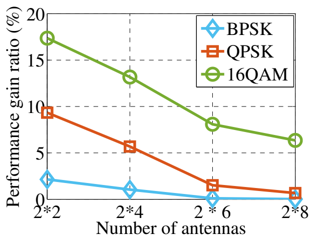

Figure 16: Performance gain ratios in single-link with 2×2–8 MIMO.

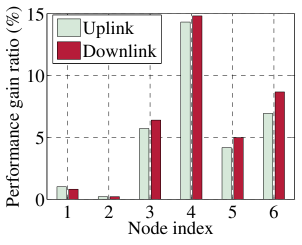

Figure 17: Performance gain ratios of STAPLE for 802.11ac (uplink and downlink).

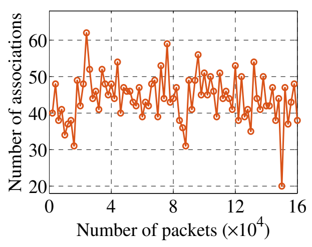

Figure 18: Number of state associations for every 2000 packets.

Table 2: Packet delivery ratios with different packet lengths (in bytes) on 2×4 MIMO.

	10	100	500	1000	1500
Traditional	99.9%	99.9%	98.3%	93.5%	91.5%
STAPLE	100%	100%	99.9%	99.7%	98.3%

Table 3: Performance gain ratios from STAPLE under noisy MIMO channels at location 7.

	BPSK	QPSK	16QAM
2 × 4	20.2%	N/A	N/A
2 × 6	16.6%	20.4%	N/A
2 × 8	11.4%	18.9%	20.8%

from 91.5% to 98.3% (a performance gain ratio of 7.4%) when the length is 1500 bytes.

Varying numbers of antennas: We also measure the performance benefits from STAPLE for 2×2–8 MIMO links. Figure 16 plots the performance gain ratios of STAPLE on 2×2, 2×4, 2×6, and 2×8 MIMO links under different modulation schemes. The transmitter and the receiver are placed at locations 1 and 4, respectively, as shown in Figure 13. We can observe from Figure 16 that STAPLE generally improves the performance more as the number of antennas decreases. For example, under 16QAM, STAPLE achieves a 17.6% performance gain ratio in the 2×2 scenario and a 6.0% ratio for the 2×8 scenario. Similarly, the performance gain ratio for QPSK reduces from 9.4% to only 0.6% when the number of receive antennas increases from 2 to 8. This is due to more link reliability from more antennas.

We then move the receiver to location 7 in Figure 13 to test the performance in an L-NLoS MIMO channel. Table 3 shows the performance gain ratios of STAPLE on 2×4, 2×6, and 2×8 MIMO links. We find that location 7 is beyond the limit of the 2×4 MIMO link under QPSK and 16QAM, and beyond the limit of the 2×6 MIMO link under 16QAM. In these cases, the receiver cannot reliably decode any packet with or without STAPLE. In other cases, we always observe that STAPLE substantially improves the link reliability, with the maximum gain ratio achieved at 20.8% for the 2×8 MIMO link under 16QAM. Consequently, we conclude that STAPLE brings more benefits in noisy channel environments.

6.2.3 *Network Performance Evaluation.*

Next, we evaluate the performance gain of STAPLE in an 802.11ac network environment. In our testing network, the AP and stations are equipped with 4 and 2 antennas, respectively. In the single-link evaluation, we always fix the modulation scheme for a packet to test the performance of a single link. In our network performance evaluation, all the values in the PHY/MAC fields of a packet (e.g. packet size and modulation scheme) are generated according to 802.11ac packet trace data. This creates a much more dynamic environment especially in uplink scenarios. We place the AP at location 0, and also place other stations, named nodes 1–6, at locations 1–6, respectively, as illustrated in Figure 13.

Overall performance: Figure 17 shows the overall performance gain ratios under uplink and downlink scenarios. Similar to the single-link case, STAPLE always improves the packet delivery ratio. We find in Figure 17 that there are very slight performance gains for nodes 1 and 2. This is because these two nodes have very good LoS channels to the AP and corrupted packet at these nodes are mostly due to collisions in the network. STAPLE is not a mechanism to resolve packet collisions. If there is no collision, the single-link packet delivery ratios for nodes 1 and 2 are close to 100%. Thus, there is little room for STAPLE to improve the performance.

As Figure 17 illustrates, the maximum performance gain ratios for the uplink and downlink are 14.3% and 14.8%, respectively, both observed at node 4, which is the furthest away from the AP in the network. This shows the capability of STAPLE to boost the link performance under noisy conditions.

We also demonstrate the dynamics of state associations inside STAPLE at the AP. We compute the number of successful state associations for every 2000 packets received. Figure 18 plots this number as a function of the total number of received packets at the AP. It is seen that most of the time, STAPLE needs to perform signal re-processing on 30-60 packets for every 2000 packets, and packet corruption events are uniformly distributed over time in our testing environment.

Impacts of threshold in state association: In state association (2), a pre-set threshold d_{th} is needed to associate a corrupted packet with a previous state. We test the impact of different thresholds on

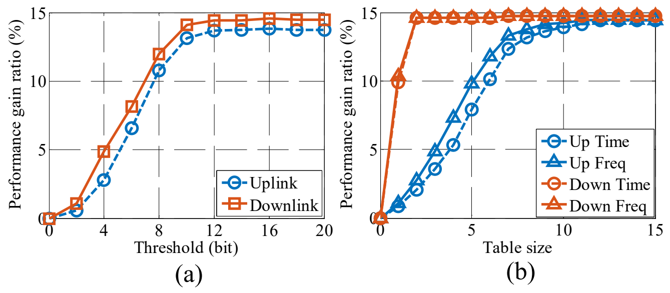

Figure 19: The impacts of (a) threshold in state association and (b) state table size.

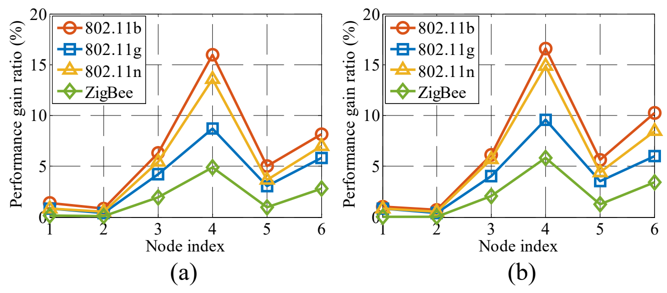

Figure 20: Network-level performance gain ratios: (a) uplink and (b) downlink.

the performance gain ratio of STAPLE at node 4. Figure 19(a) illustrates the uplink and downlink performance gain ratios with different thresholds. We can observe that the performance gain ratio of STAPLE first increases then remains approximately the same as the threshold d_{th} keeps increasing. This means that a larger d_{th} is desirable to maximize the performance gain from STAPLE in practice. But a larger d_{th} may indicate that STAPLE tries to recover heavily corrupted packets that may be not recoverable with signal re-processing, incurring more wrong associations and processing overhead. Therefore, the optimal value of d_{th} for STAPLE to balance the performance gain and complexity is 12-20 bits, as observed in Figure 19(a).

Impacts of state table setups: We also evaluate the impacts of the state table size and update policy on the performance gain of STAPLE at node 4 in the network.

Figure 19(b) shows the comparison of performance gain ratios of timestamp-based and frequency-based table update policies for different table sizes. From Figure 19(b), the frequency-based policy exhibits slightly better performance than the timestamp-based policy. The difference is negligible for both uplink and downlink when the table size is large.

We also see from Figure 19(b) that the state table size has more impacts on the performance benefits of STAPLE. For the downlink, as a station only cares about the packets from the AP, a small table maintaining 2-5 states is sufficient to increase the performance gain ratio to approximately 15%. For the uplink, the AP requires a larger table size because it has to communicate with a number of nodes. Figure 19(b) shows that a table with the size of 12-15 states at the AP is sufficient for the testing network to achieve a nearly 15% performance gain ratio.

6.3 STAPLE for Other Standards

During our experiments with 802.11ac, we find that STAPLE always improves the packet delivery by up to 20.8% and 15% in single-link and network evaluations, respectively. In the following, we show the benefits of STAPLE brought to 802.11b/g/n and 802.15.4.

802.11b/g/n network setups: We use the previous network scenario and default STAPLE parameters to set up the 802.11b/g/n testing network with the AP placed at location 0, and nodes 1-6 placed at locations 1-6, respectively, as shown in Figure 13.

802.15.4 network setups: The network topology is the same as the one in 802.11 scenarios: the network coordinator is placed at location 0, and nodes 1-6 placed at locations 1-6, respectively. Nodes 1-6 are communicating with the coordinator in the network. We obtain two ZigBee datasets [36] and [37]. The packets in [36] have 7 different lengths and packets in [37] always have the same length. We generate data packets according to [36]. The state table size is set to 5 for all nodes.

Experimental Results: Figure 20 shows the performance gain ratios of STAPLE in the uplink and downlink scenarios, respectively. Similar to Figure 17, nodes 1 and 2 gain slight benefits from STAPLE because of their good LoS channels to the AP; and node 4 has the highest performance gain ratios among all nodes.

It is observed at node 4 that the highest performance gain ratio (15.9% in uplink or 16.6% in downlink) is achieved under 802.11b and the lowest (4.9% in uplink or 5.8% in downlink) is under 802.15.4. This is because the packet state sizes in 802.11b and 802.15.4 are the largest and smallest in our STAPLE configurations (in Section 5.2), respectively. A large packet state size means that STAPLE can construct a longer preamble for signal re-processing, therefore generally indicating better performance gain. The experimental results from Figure 20 also demonstrate that STAPLE can benefit a wide range of wireless networks.

6.4 Discussions and Limitations

6.4.1 Discussions. In our performance evaluation, the downlink performance is found always slightly better than the uplink performance. This is because all nodes in the network transmit packets with limited power, leading to a minor hidden terminal problem. For example, as shown in Figure 13, with the AP being placed at location 0, node 1 (at location 1) has a small probability of not accurately sensing the transmission from node 4 (at location 4), and vice versa. This only results in minor asymmetric uplink and downlink performance, and does not affect the evaluation of the performance gains from STAPLE.

During our experiments, the size of the state table in STAPLE is chosen and evaluated empirically according to the distributions of PHY/MAC header fields in real-world 802.11 datasets. Therefore, the choice of 2-5 states for stations and 12-20 for the AP should be practically suitable for a normal WiFi network environment. The table size may have to be increased in a network with a large number of users using data-intensive applications. An under-designed table size may only reduce the performance gain of STAPLE, but does not degrade the network performance. In addition, the advantage of STAPLE is that the downlink requires much smaller table size than the uplink, which significantly benefits network users.

For 802.11ac, our packet collection and evaluation are based on the non-MU-MIMO mode. The MU-MIMO mode does not affect the uplink and is for the downlink only [11]. In addition, MU-MIMO related PHY fields (e.g., GroupID) usually remain constant if a user does not change his/her location dramatically. Thus, a low-overhead STAPLE design is expected to be still effective when used for the MU-MIMO mode in 802.11ac.

6.4.2 Limitations. In this paper, configurations for STAPLE are based on packet trace analysis for typical wireless network scenarios (e.g., a typical residential WiFi network). When a network has a large number of users, the size of the state table should be increased at least for the uplink in order to accommodate more users' packet information. As a result, configurations for STAPLE may need to be adapted accordingly for the scenarios with a large number of users and intensive network usages.

STAPLE can be considered as a way to push the performance limit of signal processing for wireless networking. Experimental results show that overall, STAPLE achieves moderate performance improvements in different evaluation scenarios. Thus, it is important to minimize the complexity of the STAPLE implementation and configuration for a practical scenario to avoid excessive energy overhead, which is proportional to the processing complexity.

STAPLE is designed as a general software radio architecture to improve the signal processing performance on a wireless link. STAPLE is shown effective to improve the link performance under long-distance or severe channel fading scenarios. However, STAPLE cannot resolve any packet corruption due to collisions in wireless networks. This may limit its use in the scenarios with congested network traffic under good channel conditions, where the wireless packet collisions are the dominate performance bottleneck.

7 RELATED WORK

In this section, we discuss existing works related to the research efforts in this paper.

Wireless signal processing: Timing/frequency synchronization and channel estimation have been well explored in the signal processing and wireless communication communities under a wide range of algorithmic settings and channel conditions [1, 2, 4, 25, 29, 38, 39], thereby establishing a relatively mature research area. Essential signal processing procedures for traditional packet reception remain generally unchanged. STAPLE takes signal processing out of the traditional domain and places it in the network domain, then demonstrates a new perspective of re-designing the signal processing procedure to improve the performance during packet reception at a wireless receiver.

Partial packet recovery: STAPLE is related to partial packet recovery (PPR) that attempts to recover a corrupted packet using a number of approaches, such as automatic repeat request (ARQ), forward error correction (FEC) [6–10, 14, 24, 40–42]. In particular, hybrid-ARQ (HARQ) [7, 9, 24] has become a widely-used technique to combine the information from multiple packets to improve the wireless link performance. STAPLE differs from H-ARQ in the following aspects: (i) H-ARQ focuses on repairing corrupted packets with the cooperation from the transmitter. STAPLE is designed as a new architecture of signal processing only residing at the receiver without any cooperation at the receiver; (ii) H-ARQ combines the retransmitted packet (including the redundant code) with the originally corrupted packet for error correction to help decoding. On the contrary, STAPLE does not directly touch upon signal decoding, but leverages low-entropy packet header information to construct a longer preamble to improve the signal processing accuracy, which in turn enhances the decoding performance.

To the best of our knowledge, STAPLE is designed as a new way to provide better signal processing performance, which is orthogonal to H-ARQ and other existing partial packet recovery mechanisms. Therefore, STAPLE can be integrated with these mechanisms to further improve the wireless network performance.

Link quality improvement: Research efforts [21, 22, 42–45] have been devoted to improving the wireless link quality. For example, [22, 43] focus on resolving the problem of packets collisions. Channel prediction is proposed in [46] to facilitate rate selection in 802.11 networks. An architecture is designed in [47] to flexibly support multiple radio frequency link chains. In addition, a few recent efforts focus on improving the MU-MIMO performance from various aspects [19, 48–50]. STAPLE is also related to [21] that leverages protocol signatures to improve the packet decoding performance based on modifications to the 802.11 PHY layer.

In contrast, STAPLE focuses on a different domain to improve the signal processing performance during packet reception and does not modify any standard. Hence, STAPLE is also complementary to these efforts and is broadly applicable to wireless networks.

8 CONCLUSION

This paper presents STAPLE, which is a lightweight, efficient mechanism to improve the signal processing performance for wireless networking. The core idea of STAPLE is to collect common header information to form the state of a packet, and use state association to recover a corrupted packet back to a previous state for constructing a longer preamble to further enhance the reliability of signal processing. We implement STAPLE on USRP X300 devices with adapted configurations for 802.11a/b/g/n/ac and 802.15.4. Experimental results show that STAPLE improves the wireless network performance under various conditions.

ACKNOWLEDGEMENT

The authors would like to thank the shepherd Dr. Bo Chen and the anonymous reviewers for the insightful comments and feedback. The work at University of South Florida in this paper was supported in part by NSF under grants CNS-1527144, CNS-1553304, and CNS-1701394, and ARO under grant W911NF-14-1-0324.

REFERENCES

[1] J.-J. Van de Beek, M. Sandell, and P. O. Borjesson, "ML estimation of time and frequency offset in OFDM systems," *IEEE Trans. Signal Process.*, vol. 45, pp. 1800–1805, 1997.

[2] P. H. Moose, "A technique for orthogonal frequency division multiplexing frequency offset correction," *IEEE Trans. Commun.*, vol. 42, pp. 2908–2914, 1994.

[3] L. M. Davis, I. B. Collings, and P. Hoeher, "Joint MAP equalization and channel estimation for frequency-selective and frequency-flat fast-fading channels," *IEEE Trans. Commun.*, vol. 49, pp. 2106–2114, 2001.

[4] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE trans. commun.*, vol. 45, pp. 1613–1621, 1997.

[5] M.-A. Badiu, G. E. Kirkelund, C. N. Manchón, E. Riegler, and B. H. Fleury, "Message-passing algorithms for channel estimation and decoding using approximate inference," in *Prof. of IEEE ISIT*, 2012.

[6] G. Angelopoulos, M. Medard, and A. Chandrakasan, "Harnessing partial packets in wireless networks: Throughput and energy benefits," *IEEE Trans. Wireless Commun.*, 2016.

[7] P. Larsson, L. K. Rasmussen, and M. Skoglund, "Throughput analysis of hybrid-AQR–a matrix exponential distribution approach," *IEEE Trans. Commun.*, vol. 64, pp. 416–428, 2016.

[8] Y. Wang, W. Li, and S. Lu, "The capability of error correction for burst-noise channels using error estimating code," in *Prof. of IEEE SECON*, 2016.

[9] D. Jia, Z. Fei, J. Yuan, S. Tian, and J. Kuang, "A hybrid EF/DF protocol with rateless coded network code for two-way relay channels," *IEEE Trans.Commun.*, vol. 64, pp. 3133–3147, 2016.

[10] M. S. Mohammadi, Q. Zhang, and E. Dutkiewicz, "Reading damaged scripts: partial packet recovery based on compressive sensing for efficient random linear coded transmission," *IEEE Trans. Commun.*, vol. 64, pp. 3296–3310, 2016.

[11] "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11*, 2013.

[12] "Low-rate wireless personal area networks (LR-WPANs)," *IEEE Std 802.15.4*, 2011.

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

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

[15] M.-H. Lu, P. Steenkiste, and T. Chen, "Design, implementation and evaluation of an efficient opportunistic retransmission protocol," in *Prof. of ACM MobiCom*, 2009.

[16] S. Noh, M. D. Zoltowski, Y. Sung, and D. J. Love, "Pilot beam pattern design for channel estimation in massive MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 8, pp. 787–801, 2014.

[17] A. Goldsmith, *Wireless communications.* Cambridge university press, 2005.

[18] D. Tse and P. Viswanath, *Fundamentals of wireless communication.* Cambridge university press, 2005.

[19] X. Xie, E. Chai, X. Zhang, K. Sundaresan, A. Khojastepour, and S. Rangarajan, "Hekaton: Efficient and practical large-scale MIMO," in *Prof. of ACM MobiCom*, 2015.

[20] Y. Chen, X. Liu, Y. Cui, J. Zou, and S. Yang, "A multiwinding transformer cell-to-cell active equalization method for lithium-ion batteries with reduced number of driving circuits," *IEEE Trans. Power Electron.*, vol. 31, pp. 4916–4929, 2016.

[21] J. Huang, Y. Wang, and G. Xing, "LEAD: leveraging protocol signatures for improving wireless link performance," in *Prof. of ACM MobiSys*, 2013.

[22] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *Prof. of ACM SIGCOMM*, 2008.

[23] P. Larsson, L. K. Rasmussen, and M. Skoglund, "Throughput analysis of ARQ schemes in Gaussian block fading channels," *IEEE Trans. Commun.*, vol. 62, pp. 2569–2588, 2014.

[24] A. Chelli, E. Zedini, M.-S. Alouini, J. R. Barry, and M. Pätzold, "Performance and delay analysis of hybrid ARQ with incremental redundancy over double Rayleigh fading channels," *IEEE Trans. Wireless Commun.*, vol. 13, pp. 6245–6258, 2014.

[25] M.-A. Badiu, C. N. Manchón, and B. H. Fleury, "Message-passing receiver architecture with reduced-complexity channel estimation," *IEEE Commun. Lett.*, vol. 17, pp. 1404–1407, 2013.

[26] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, J. Zahorjan, and E. Lazowska, "CRAWDAD dataset uw/sigcomm2004 (v. 2006-10-17)," Downloaded from http://crawdad.org/uw/sigcomm2004/20061017, Oct. 2006.

[27] A. Schulman, D. Levin, and N. Spring, "CRAWDAD dataset umd/sigcomm2008 (v. 2009-03-02)," Downloaded from http://crawdad.org/umd/sigcomm2008/20090302, Mar. 2009.

[28] O. Besson and P. Stoica, "On parameter estimation of MIMO flat-fading channels with frequency offsets," *IEEE Trans. Signal Process.*, vol. 51, pp. 602–613, 2003.

[29] Q. Li, K. C. Teh, and K. H. Li, "Low-complexity channel estimation and turbo equalisation for high frequency channels," *IET Commun.*, vol. 7, pp. 980–987, 2013.

[30] T. M. Cover and J. A. Thomas, *Elements of information theory.* John Wiley & Sons, 2012.

[31] "Monitoring 802.11n and 802.11ac networks," http://www.tamos.com/htmlhelp/commwifi/monitoring_802_11n_networks.htm.

[32] M. Ettus, "USRP user's and developer's guide," *Ettus Research LLC*, 2005.

[33] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 16, pp. 1451–1458, 1998.

[34] S. Sen, R. R. Choudhury, and S. Nelakuditi, "CSMA/CN: carrier sense multiple access with collision notification," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 544–556, 2012.

[35] S. Sen, R. Roy Choudhury, and S. Nelakuditi, "No time to countdown: Migrating backoff to the frequency domain," in *Prof. of ACM MobiCom*, 2011.

[36] S. Fu and Y. Zhang, "CRAWDAD dataset due/packet-delivery (v. 2015-04-01)," Downloaded from http://crawdad.org/due/packet-delivery/20150401/packet-metadata, Apr. 2015, traceset: packet-metadata.

[37] A. Iqbal, K. Shahzad, S. A. Khayam, and Y. Cho, "CRAWDAD dataset niit/bit_errors (v. 2008-07-08)," Downloaded from http://crawdad.org/niit/bit_errors/20080708/802.15.4, Jul. 2008, traceset: 802.15.4.

[38] G. Liu, L. Zeng, H. Li, L. Xu, and Z. Wang, "Adaptive interpolation for pilot-aided channel estimator in OFDM system," *IEEE Trans. Broadcast.*, vol. 60, pp. 486–498, 2014.

[39] A. Movahedian and M. McGuire, "On the capacity of iteratively estimated channels using LMMSE estimators," *IEEE Trans. Veh. Technol.*, vol. 64, pp. 97–107, 2015.

[40] J. Xie, W. Hu, and Z. Zhang, "Revisiting partial packet recovery in 802.11 wireless LANs," in *Prof. of ACM MobiSys*, 2011.

[41] Y. Xie, Z. Li, M. Li, and K. Jamieson, "Augmenting wide-band 802.11 transmissions via unequal packet bit protection," in *Prof. of IEEE INFOCOM*, 2016.

[42] J. Huang, G. Xing, J. Niu, and S. Lin, "CodeRepair: PHY-layer partial packet recovery without the pain," in *Prof. of IEEE INFOCOM*, 2015.

[43] L. Kong and X. Liu, "mZig: Enabling multi-packet reception in ZigBee," in *Prof. of ACM MobiCom*, 2015.

[44] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "piStream: Physical layer informed adaptive video streaming over LTE," in *Prof. of ACM MobiCom*, 2015.

[45] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive Wi-Fi: Bringing low power to Wi-Fi transmissions," in *Prof. of NSDI*, 2016.

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

[47] B. Chen, V. Yenamandra, and K. Srinivasan, "Flexradio: Fully flexible radios and networks," in *Prof. of NSDI*, 2015.

[48] A. B. Flores, S. Quadri, and E. W. Knightly, "A scalable multi-user uplink for Wi-Fi," in *Prof. of NSDI*, 2016.

[49] C. Shepard, A. Javed, and L. Zhong, "Control channel design for many-antenna MU-MIMO," in *Prof. of ACM MobiCom*, 2015.

[50] E. Hamed, H. Rahul, M. A. Abdelghany, and D. Katabi, "Real-time distributed MIMO systems," in *Proc. of ACM SIGCOMM*, 2016, pp. 412–425.