Chapter 1

Software-Defined Radio (SDR)-based Real-Time WLANs for Industrial Wireless Sensing and Control

Zelin Yun, Natong Lin, Shengli Zhou, and Song Han

University of Connecticut, United States

*Corresponding Author: Song Han; song.han@uconn.edu

Abstract: In this chapter, we review three different 802.11-based WiFi solutions that address the urgent need for a real-time high-speed wireless communication protocol to support time- and mission-critical wireless sensing and control systems. First, an RT-WiFi protocol design based on a time division multiple access (TDMA)-based data link layer scheduler is presented to provide deterministic timing guarantee on packet delivery by operating on top of commercial off-the-shelf (COTS) devices. Second, a software-defined radio (SDR)-based solution called SRT-WiFi is implemented on FPGA-based SDR platform, providing full-stack configurability to align with the evolving IEEE 801.11 standard. Finally, we discuss an ongoing effort to explore the implementation of the 802.11a/g/n/ac physical layers on the GNU Radio based SDR platforms, which includes both

single-user (SU) and multi-user (MU) MIMO transmissions. The effectiveness of the three solutions are validated through both real-world testbed deployment and extensive simulation-based experiments.

Keywords: Software-defined radio, RT-WiFi, full-stack configurability, FPGA, GNU Radio, SU/MU-MIMO

1.1. Introduction

Applying real-time wireless technologies in industrial control systems has been gaining popularity in recent years. Pervasively deployed sensors and actuators based on high-throughput wireless standards allow for increased network throughput, improved system mobility and reduced maintenance costs. A key step of designing a wireless control system is to choose the most appropriate wireless protocol based on its desired control specifications. Some protocols based on low-data-rate physical layers (PHYs), like 802.15.4, focus on real-time packet delivery and reliable performance but are only suitable for low-speed control applications. Comparatively, IEEE 802.11 standard (WiFi) is designed for high-speed wireless local area networks (WLANs) [Tramarin et al., 2019].

Table 1.1 gives an overview of the evolution of IEEE 802.11 standards, which was first released in 1997 and designed for WLAN usage as part of the IEEE 802 family. After the first widely used version IEEE 802.11b (WiFi 1) in 1999, the 802.11 working group (WG) released the version of IEEE 802.11a (WiFi 2) and IEEE 802.11g (WiFi 3) supporting orthogonal frequency division multiplexing (OFDM). From IEEE 802.11n (WiFi 4), the single-user MIMO (SU-MIMO) is supported with multiple optional beamforming transmissions. In IEEE 802.11ac (WiFi 5), the multi-user MIMO (MU-MIMO) is added and

Table 1.1: An overview of 802.11 standard evolution

Protocol		PHY Name	Max.Rate [Mbit/s]	Channel bandwidth [MHz]	Band [GHz]	Name
802.11	1997	DSSS	2	22	2.4	(Wi-Fi 1)
802.11b	1999	HR/DSSS	11	22	2.4	(Wi-Fi 2)
802.11a	1999	OFDM	54	20	5	-
802.11g	2003	ERP-OFDM	54	20	2.4	(Wi-Fi 3)
802.11n	2009	HT-MIMO	600	20/40	2.4/5	Wi-Fi 4
802.11ac	2013	VHT-MIMO	3466.8	20/40/80/160	5	Wi-Fi 5
802.11ax	2019	HE-OFDM	10530	20/40@2.4GHz 20/40/80/160@5GHz	2.4/5	Wi-Fi 6
802.11be	2024	EHT-OFDM	46120	20/40@2.4GHz 20/40/80/160@5GHz 80/160/320@6GHz	2.4/5/6	Wi-Fi 7

the compressed channel feedback is the only method for MU-MIMO beamforming. IEEE 802.11ax (WiFi 6) supports orthogonal frequency-division multiple access (OFDMA), and the latest IEEE 802.11be standard (WiFi 7) has a maximum data rate of 46 Gbps.

Notably, the non-deterministic communication performance of standard 802.11 makes it incapable of mission- and safety-critical applications that require high determinism and reliability. To address this issue, a systematic solution named RT-WiFi [Wei et al., 2013, Leng et al., 2014, Wei et al., 2018] was proposed to provide real-time data delivery for a range of wireless control systems. RT-WiFi is a TDMA-based data link layer (DLL) protocol built on IEEE 802.11 a/g PHY, providing deterministic timing guarantees for packet delivery with a configurable sampling rate of up to 6 kHz. RT-WiFi was implemented on AR9285, a commercial-off-the-shelf (COTS) 802.11 network interface card (NIC). This allows running existing applications on top of RT-WiFi with minimum modifications thus offering the advantage of much shortened development periods; however, it comes with the trade-off of limited flexibility in terms of

radio technologies. For instance, the Atheros AR9285 is limited to compatibility with IEEE 802.11a/g, whereas many real-time wireless protocols are based on varied radio technologies and thus require different hardware platforms. It is thus a significant challenge to develop a uniform communication platform that integrates various real-time wireless technologies to maximize the existing hardware investments and software development efforts.

These limitations motivate us to develop a SDR-based configurable real-time wireless platform. This platform is programmable at both PHY and DLL layers to meet the diverse requirements of industrial control systems, including those with multiple operational modes. SRT-WiFi [Yun et al., 2022] is a SDR-based solution for RT-WiFi to serve this purpose. Its design and implementation leverage an advanced SDR platform (Xilinx Zynq-7000 and Analog Device AD9364), with radio functions programmed on an FPGA. SRT-WiFi can operate in hard real-time because its radio functions are executed by logic blocks in the FPGA running at oscillator-driven speeds, and thus support the essential functions needed for high-speed real-time communications and provide an open-source platform to accommodate the evolving IEEE 802.11 standards.

While SRT-WiFi provides real-time and reliable wireless communications for industrial control applications, its current version only supports IEEE 802.11a/g PHYs and SISO communications. Our ultimate goal is to develop SRT-WiFi into a full-blown SDR-based real-time wireless platform and support newer standards of WiFi, e.g., IEEE 802.11n/ac/ax, to enable both SU/MU MIMO and OFDMA. As the first step towards this goal, we extend SRT-WiFi on GNU Radio, a widely used open-source SDR platform [GNU Radio Foundation, 2007]. With GNU Radio and USRP, we can implement and evaluate

Table 1.2: Pros and cons of the three solutions.

	Pros	Cons
RT-WiFi [Wei et al., 2013]	• Timing guarantee on packet delivery • Flexible DLL configuration • Seamless integration with existing hardware	• Needs significant effort and hardware expertise to manage and upgrade COTS devices
SRT-WiFi [Yun et al., 2022]	• Full-stack configurability • Precise time synchronization and real-time communication • Efficient queue management	• Complexity of implementation and long developing period • Only support IEEE 802.11a/g
GR-WiFi	• Support multiple standards, including IEEE 802.11a/g/n/ac	• Not able to perform on real-time testbed yet

the PHYs of newer 802.11 standards with a much shorter development period when compared to developing those PHYs directly on FPGA-based SDR platform. For simplicity of presentation, we call this GNU Radio-based implementation GR-WiFi to differentiate it from the SRT-WiFi system developed on FPGA-based SDR platform. Once GR-WiFi is fully developed and tested on GNU Radio, it will be ported on the FPGA-based SDR platform to make it full-blown and support hard real-time performance. In GR-WiFi, we have successfully implemented the PHYs of IEEE 802.11a/g/n/ac standards supporting the Legacy OFDM (Legacy), high-throughput (HT) and very-high-throughput (VHT) PHY formats with SISO and 2×2 SU-MIMO and MU-MIMO. Both FPGA-based SRT-WiFi and GNU Radio-based GR-WiFi implementations, once mature, will be made public to the wireless communities to support a broad range of R&D activities. Table 1.2 summarizes the strengths and limitations of the three solutions reviewed in this chapter.

1.2. RT-WiFi based on IEEE 802.11a/g

An RT-WiFi network consists of three primary components: RT-WiFi stations (STAs), which are devices equipped with 802.11-compatible hardware and the RT-WiFi protocol stack; RT-WiFi Access Points (APs), which function as intermediaries to support message exchange between the network manager and RT-WiFi STAs; and the network manager, a software module that configures the network, coordinates communication between APs and STAs, and adjusts the communication schedule when necessary. An RT-WiFi AP and its associated STAs are defined as a cluster. An RT-WiFi network with multiple APs is called a multi-cluster RT-WiFi network [Leng et al., 2019] (see Fig. 1.1). In industrial practice, the placement of the RT-WiFi APs will be done through careful site survey, resulting in each AP and its associated STAs forming a star topology. In a multi-cluster RT-WiFi network, each AP is managed by an AP network manager responsible for its cluster. Also, a central network manager supervises all AP network managers and coordinates packet transmissions among different clusters. Since RT-WiFi is a TDMA-based communication protocol, the local clocks of all STAs and APs are synchronized [Wei et al., 2013].

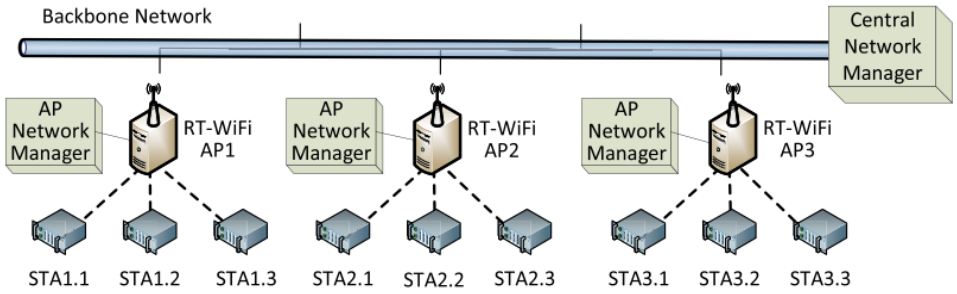

Figure 1.1: Overview of an RT-WiFi network with three clusters.

1.2.1. RT-WiFi Protocol Design

The RT-WiFi protocol stack is the most essential building block of the RT-WiFi network. It enables real-time and high-speed data transmissions and customizable DLL configurations for diverse applications. The RT-WiFi protocol design takes into consideration the requirements of different types of control applications, enabling control designers to choose the communication behavior that fits their applications the best. At the same time, RT-WiFi design minimizes the modification on the original WiFi protocol so that it can be transparent to both the upper layer software stack and underlying hardware to provide the most compatibility and usability.

The architecture of RT-WiFi protocol is shown in Fig. 1.2. At the very bottom, RT-WiFi utilizes IEEE 802.11 PHY, which is sufficiently fast for most wireless control systems. Control application users can easily implement the RT-WiFi DLL on COTS IEEE 802.11 hardware to support high-speed and real-time data transmissions. Above the IEEE 802.11 PHY layer is a TDMA-based DLL, which is the core of the RT-WiFi protocol. Combined with the centralized channel and time management schemes imposed by the RT-WiFi network manager, this DLL ensures collision-free and deterministic communications. Additionally, it offers a flexible abstraction for the upper layers, allowing seamless support for standard UDP/TCP-based applications.

The RT-WiFi DLL comprises three main components: a timer that ensures global synchronization across all RT-WiFi nodes and initiates timing events; a link scheduler that manages media access and executes scheduled events at designated time points; and a flexible channel access controller that dynamically configures hardware parameters to execute timing events based on the target application's behavior.

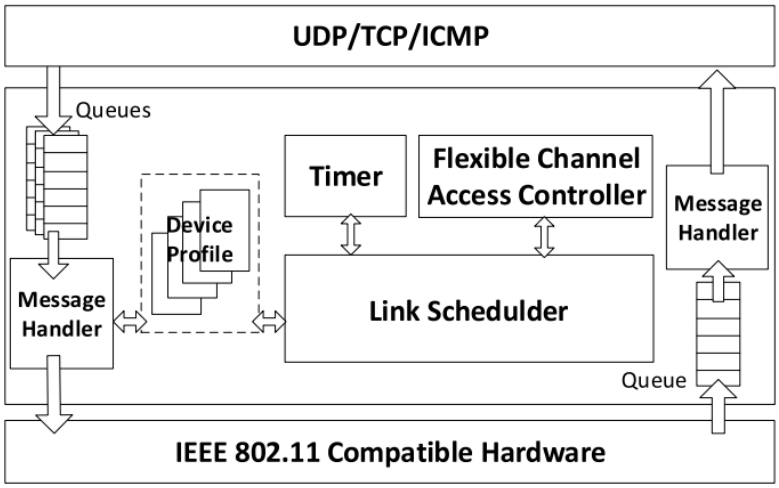

Figure 1.2: System architecture of the RT-WiFi protocol.

1.2.2. Performance Evaluation

We set up a small-scale testbed with one AP and three STAs to evaluate the performance of the RT-WiFi DLL design (see Fig. 1.3). Each device utilized the Atheros AR9285 NIC operating on the 802.11g protocol, though they were powered by CPUs with varying computing capabilities. In the experiments, six pairs of UDP sockets were established between the STAs and the AP, with data published every 4 ms for each socket with a fixed payload of 460 bytes. The comparison results between WiFi and RT-WiFi are summarized in Table 1.3. It shows that the average latency variation in a WiFi network is up to 90 times greater than in an RT-WiFi network, with the maximum delay exceeding 30 times that of RT-WiFi. In contrast, RT-WiFi supports a sampling rate of up to 6 kHz, and less than 0.01% of packets have latency greater than 1 ms, meeting the requirements of most industrial control systems [Wei et al., 2013].

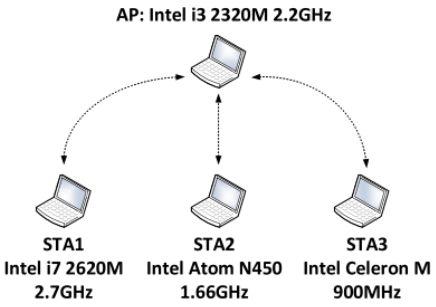

Figure 1.3: Testbed setup for RT-WiFi performance evaluation.

Table 1.3: Comparison of delay between RT-WiFi and regular WiFi networks.

Link	Max Delay(μs)		Mean Delay(μs)		Standard Deviation(μs)	
	RT-WiFi	Wi-Fi	RT-WiFi	Wi-Fi	RT-WiFi	Wi-Fi
STA1 $\rightarrow$ AP	3865	100078	176	401	25.86	1491.69
STA2 $\rightarrow$ AP	4193	81499	171	348	27.62	1000.60
STA3 $\rightarrow$ AP	3861	75298	174	429	25.16	1221.72
AP $\rightarrow$ STA1	1197	78089	184	788	16.86	2861.42
AP $\rightarrow$ STA2	1342	78923	189	790	15.19	2806.56
AP $\rightarrow$ STA3	2186	77860	189	799	19.03	2855.89

1.3. SRT-WiFi based on IEEE 802.11a/g

The RT-WiFi protocol design offers several advantages, including deterministic timing guarantee on packet delivery, flexible data link layer configuration, and seamless integration with existing hardware. However, it has some limitations, such as limited flexibility in terms of radio technologies and no support for frequently updated wireless protocols. To address these limitations, SRT-WiFi [Yun et al., 2022] introduces an SDR-based configurable real-time solution. In contrast to RT-WiFi, which relies on COTS hardware, the SDR platform offers programmability at both the PHY and DLL levels. This flexibility allows it to accommodate the needs of various industrial control systems with multiple operational modes.

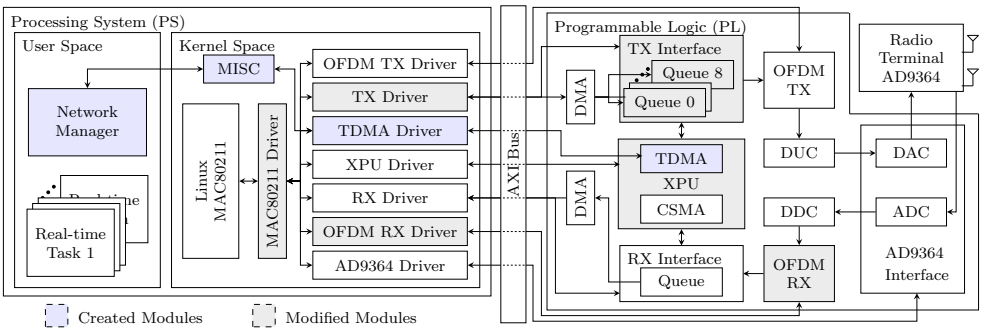

Figure 1.4: Overview of the SRT-WiFi system architecture. It highlights the created and modified modules in SRT-WiFi based on the Openwifi architecture.

SRT-WiFi is built upon Openwifi [Jiao et al., 2020], a SoftMAC IEEE 802.11 design compatible with the Linux MAC80211 subsystem. As shown in Fig. 1.4, the Openwifi system has two major components: the processing system (PS) and the programmable logic (PL). The PS handles the majority of the MAC layer and all higher layers. The PL is an FPGA-based embedded system responsible for the real-time portion of the MAC and PHY layers. Both PS and PL are implemented on the Zynq-7000 SoC, which includes an FPGA for the PL and an ARM processor for the PS. Data exchange between PL and PS occurs through the Advanced eXtensible Interface (AXI) bus, supporting direct memory access as well as register reading and writing. Additionally, the PL is connected to an AD9364 radio terminal from Analog Devices for signal transceiving.

Fig. 1.4 shows three main modules of PL on the right side: the TX interface (TXI), the XPU (application-specific processing unit), and the RX interface (RXI). The TXI module manages packet transmission, while the RXI module handles packet reception. The XPU module controls channel access by using IEEE 802.11 distributed coordination function (DCF) [IEEE 802.11 Working Group, 2021]. Leveraging concurrent processing ability in FPGA, the radio

terminal can operate its transmitter (TX) and receiver (RX) modules at the same time. Besides, PL modules equipped with registers allow configurations of operation modes and parameters.

In PS, a Linux OS is operating on an ARM processor. As the platform adopts SoftMAC architecture, most MAC functionalities are integrated into the Linux kernel (MAC80211 subsystem [Mur, 2011]), excluding the real-time MAC and PHY that are being implemented in PL. Between the Linux MAC80211 subsystem and the wireless adapter (PL), the MAC80211 driver is in place to facilitate communication. Sub-drivers (depicted on the left side of Fig. 1.4) ensure data communications between the MAC80211 driver and PL. MAC80211 driver interacts with PL by calling APIs provided by sub-drivers. Additionally, TX and RX drivers manage the transmission and reception of data packets between the PS and PL, respectively, using DMA.

Building on top of Openwifi, SRT-WiFi aims to achieve several key objectives: enabling precise network-wide time synchronization and facilitating multi-cluster real-time communications with effective rate adaptation at run time. The design details of the modified PL and PS components of SRT-WiFi are presented below.

1.3.1. Programmable Logic (PL) in SRT-WiFi

The PL component of SRT-WiFi is designed to 1) achieve real-time transmissions with high-precision time synchronization, 2) enhance queue management efficiency, and 3) measure precise link reception SNR as a reference for rate adaptation. We now describe how to achieve these functions in SRT-WiFi PL.

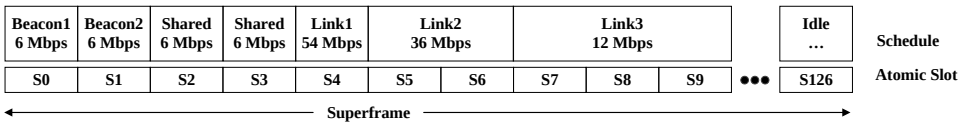

Figure 1.5: The timing diagram of an example superframe in multi-cluster SRT-WiFi with 127 time slots.

1.3.1.1. TDMA Block Design in SRT-WiFi PL

To improve real-time performance, a TDMA block is designed in XPU to supplement the CSMA block. SRT-WiFi can seamlessly switch between TDMA and CSMA modes during runtime.

The TDMA mode in SRT-WiFi is designed to transmit and receive frames at designated times, coordinating communication between APs and STAs to prevent collisions. With this objective in mind, all transmissions follow a schedule that includes the transmitting times of the links with a specified time duration called superframe. A superframe consists of consecutive time slots, with each slot specifying the transmission state(TX, RX, or Idle) and the corresponding sender or receiver. At run time, the superframe is continuously generated to schedule the transmissions. Each time slot of the superframe has an atomic slot as the basic time unit. The length of time slots varies along with the rate to support rate adaptation, as a lower rate requires more time, namely more atomic slots, to transmit the same packet. In SRT-WiFi, superframe lengths, time slots, and atomic slots are fully customized. In most cases, application's requirements decide superframe length and selected data rate in the PHY link configure time slot and atomic slot lengths.

Fig. 1.5 illustrates a superframe example in an SRT-WiFi network. The superframe has 127 atomic slots, with Slot0 and Slot1 allocated to AP1 and AP2 for sending beacons. Shared slots shown in Slot2 and Slot3 are available for

any links that are used for the association process. The remaining atomic slots are either used for specific link communications or left idle. In the example, each link has the same MTU but operates at different rates, requiring varied time slot lengths and different numbers of atomic slots to transmit.

In the TDMA block, a register page is implemented for the TDMA driver can store and maintain schedule information. A scheduler timer located in the TDMA block will trigger the transmissions based on schedule. Each time slot attached a link assignment at the beginning will be retrieved by the TDMA block. The TXI module, which has the queue ready for each link, will send a frame to the next module as soon as it detects a loaded queue. The OFDM TX module then processes the frame by modulating the bit stream into the digital signal stream and handing it over to the DAC interface for final signal emission through the radio terminal's antenna

1.3.1.2. TDMA Time Synchronization Design

SRT-WiFi has an accurate time synchronization feature among the devices in the network. The SRT-WiFi network contains multiple clusters with an AP and several STAs, which may share the same channel. STAs and AP within a cluster running at the same channel need to be synchronized to avoid potential collisions. To solve this, a novel synchronization method is implemented at the PHY layer on the SDR device. In the scenario that APs run on the same channel, one AP is selected as the main AP (MAP), and the rest are the subordinate APs (SAPs). The setting assumes that all SAPs can receive signals from the MAP, which serves as the provider of the reference clock. The SAPs synchronize with the MAP, while all STAs synchronize with their respective APs. For instance, as depicted in Fig. 1.1, AP2 serves as the MAP while AP1

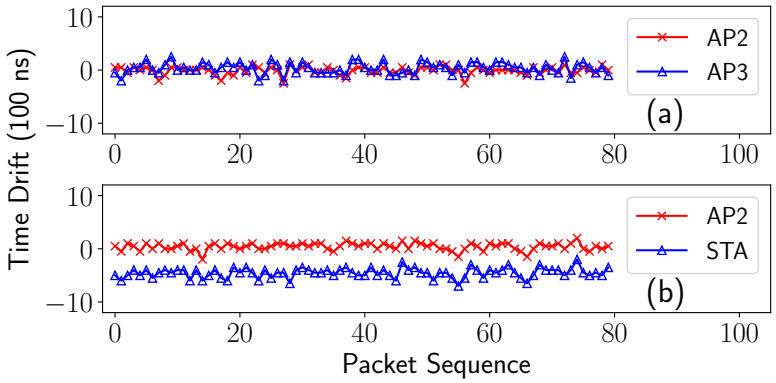

Figure 1.6: Synchronization performance of APs and STAs in a multi-cluster SRT-WiFi network.

and AP3 are SAPs synchronizing to AP2. This synchronization mechanism prevents devices from relying on the timer in a non-real-time operating system, leveraging instead the timer in hard real-time PL. To achieve this, a scheduler timer with nanosecond precision is added into the TDMA block to trigger transmissions.

In the PL layer, OFDM RX module performes PHY demodulation, and demodulated status and symbols are passed to RXI and XPU. In the TDMA block, a synchronization function is added to synchronize time with a specific AP by utilizing the demodulated result. The synchronization procedure has the following steps.

The synchronization function waits for the MAC packet header from the OFDM RX module and checks the packet's content. If it's a beacon packet, the function continually waits for the service set ID (SSID) in the following packet payload. Upon reading the SSID, the synchronization function compares it against the target SSID provided by the TDMA driver through the registers. If two SSIDs match, the buffered time is updated to the schedule

timer; If not, the synchronization function waits for the next packet, and the schedule timer continues to run as usual without any update. Notably, this synchronization method is also compatible with other protocol versions like IEEE 802.11n/ac/ax.

Using this synchronization mechanism, our experimental results demonstrate that the synchronization time drift of the SRT-WiFi devices can be maintained within 0.2 μs, which outperforms the COTS hardware. Fig. 1.6 presents the results of two experiments from [Yun et al., 2022]. In the first experiment, two SAPs, AP2 and AP3, synchronize with a MAP AP1, and their synchronization time error is measured, as shown in Figure. 1.6 (a). The maximum error observed is 0.2 μs. In the second experiment, AP2, acting as an SAP, synchronizes with the MAP AP1, while a STA synchronizes with AP2. The maximal synchronization error of the STA in multi-cluster SRT-WiFi networks is measured, which is within 1 μs, as depicted in Fig. 1.6 (b). This improvement on time synchronization accuracy can help reduce guard time and support shorter time slot lengths, thereby improving the sampling rates.

1.3.1.3. Queue Management

In SRT-WiFi, packets from the PS are pushed into queues before transmission. Unlike COTS hardware-based RT-WiFi, where the queue number is fixed and cannot be changed, SRT-WiFi provides greater flexibility in queue configuration. For instance, AR9285 used in RT-WiFi [Wei et al., 2013] has only 8 queues. In SRT-WiFi real-time transmissions, each link has its own queue stack to guarantee the desired timing and throughput performance. However, if the number of STAs exceeds the number of available queues in the AP, packets from different links need to share a queue, potentially causing timing

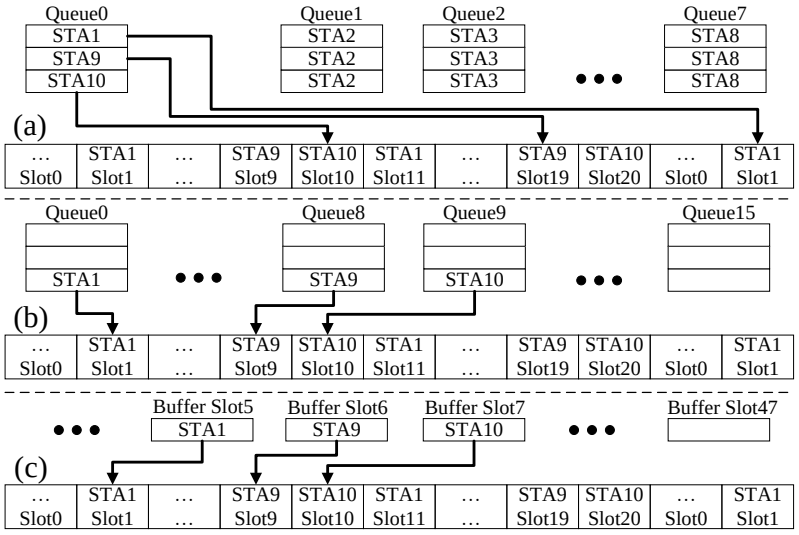

Figure 1.7: Queue management issues in real-time networks with shared queues.

violations.

Fig. 1.7(a) shows a device with 10 links but has 8 available queues, causing STA9 and STA10 to share a queue with others. When multiple packets from different links enter the same queue, they may wait until the packet at the queue head being sent, even if their assigned time slots in the superframe occur earlier. AP will encounter this time violation issue when managing real-time transmissions for multiple STAs.

In SRT-WiFi, the TDMA block initiates packet transmission. Queues can be assigned to different links, and the packets from different links are put into the corresponding queues, as shown in Fig. 1.7(b). The TDMA block's schedule determines which queue to trigger for each slot. This flexibility is a feature of SDR-based systems, as the queues can be configured in software rather than hard programmed, as in COTS hardware. Given sufficient FPGA resources, the supported queues can be extended to accommodate any number.

To address the time violation issue when the available AP queues are less

Table 1.4: Assigned and dynamic queue management maximum and average packet delay (slot number) with 16 links.

Number of Queues	8	10	12	14	16
Assigned Maximum Delay (slot)	2816	2358	1707	1125	82
Dynamic Maximum Delay (slot)	591	162	106	104	103
Assigned Average Delay (slot)	336	236	159	87	16
Dynamic Average Delay (slot)	271	42	16	16	16

than supported STAs, a dynamic buffer is designed in SRT-WiFi as shown in Fig. 1.7(c). A dynamic buffer contains a series of slots, each of which stores one packet at most. When a packet arrives, the TXI pushes it into an unused buffer slot. Next, based on the schedule, the TDMA module checks the link information located at the beginning of each time slot and scans the whole buffer to check if a packet for that link is present. If it finds one, it triggers the transmission from the corresponding buffer slot. Since each buffer slot holds only one packet, more buffer slots can be implemented with the same FPGA resources.

Table 1.4 compares the performance of assigned and dynamic queue management using 16 links and a variable number of queues. AP periodically generates a packet that requires only one atomic slot for transmission for each link. In assigned queue management, the packet is pushed into this pre-assigned and handled according to the schedule. While in dynamic queue management, the packet is pushed into an available queue when it arrives. All transmissions follow a randomly generated schedule where the throughput of each link is guaranteed and the length of superframe is fixed. During the scheduled transmission, packet delay is recorded, and no packets are dropped due to delay. The results show that even a small disparity between the number of queues

and links in the assigned queue management can lead to a significantly high maximum and average delay. On the other hand, the dynamic queue management method can manage more links with the same number of queues and maintain a lower max. and avg. packet delays. However, it does not completely eliminate delays since all queues are shared.

1.3.1.4. Link Quality Measurement

The rate adaptation function in SRT-WiFi requires precise SNR measurement. There are two practical methods developed to meet the SNR requirement. Both methods leverage the long training field (LTF) in the Legacy physical layer protocol data unit (PPDU) preamble of the 802.11 PHY signal. The first method involves calculating the auto-correlation [Lee and Messerschmitt, 2012] of the LTF. The LTF contains a half symbol followed by two repeated symbols, which correspond to 160 samples at a 20 MHz sampling rate. So, the LTF shows the same pattern in every 64 samples [IEEE 802.11 Working Group, 2021]. We utilize 128 consecutive samples of the LTF to calculate the auto-correlation, denoted as ρ, and the SNR value (in dB) can then be determined as follows:

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{\rho}{1 - \rho} \right) \tag{1.1}$$

where we assume that $\rho < 1$. Two 64 repeated samples are used, but not the first 32 samples because of transient effects that can occur at the start of a transmission in the sender's hardware.

In the second method, the LTF and a piece of background noise before the data symbol are buffered after the packet arrival. The power of the background noise can be measured before packet arrival, and the power of LTF signal

includes noise power plus the signal power. Then, the SNR (dB) is computed in this way:

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{P_{\mathrm{LTF}} - P_{\mathrm{noise}}}{P_{\mathrm{noise}}} \right) \tag{1.2}$$

where P_{LTF} is the signal power of LTF and P_{noise} is the power of the background noise signal before the packet. We assume that P_{LTF} is larger than P_{noise}.

Both SNR measurement methods are integrated into the OFDM RX module of SRT-WiFi. An SNR value is calculated each time a packet is received. If the received packet contains a source address, the computed SNR value is stored along with the source address. The MAC80211 driver forwards the SNR information to the TDMA driver, who manages the scheduling in the system and utilizes the SNR data for scheduling decisions. The device manager on each device interacts with the TDMA driver to access the SNR information and forwards the SNR information to the central network manager. The central network manager uses this SNR data to determine the appropriate data rate for each link based on the quality of the wireless connections and creates the network schedule

1.3.2. Processing System (PS) in SRT-WiFi

The two main components of the PS design in SRT-WiFi are the drivers and the network manager. The drivers act as the interface between the PL and Linux (see Fig. 1.4), serving two primary functions: 1) configuring parameters in the PL modules to support various working modes and functions and 2) managing the packet exchange between the PL and the OS. Each PL module has a corresponding driver connected to the kernel because each PL module contains a register page used to set or read status. For example, TXI uses a

register to determine if a packet requires an ACK and another register to report the packet delivery status. On the kernel side, sub-drivers handle configuration tasks within the PL component and interact with the OS by encapsulating read and write functions into APIs for the MAC80211 driver. The TDMA block in the XPU also has a register page containing three parts. The first part is for schedule allocation, including superframe and atomic slot length. The second part is used for PL synchronization by acquiring AP'SSID for stations to synchronize with. The third part is a mode switch for toggling between defaulted CSMA and customized TDMA modes. Since the TDMA mode's functions are incompatible with the MAC80211 subsystem, configuring the TDMA block through MAC80211 is challenging. Therefore, the TDMA driver is implemented as a miscellaneous character driver (MISC), providing basic read/write functions for user space. In user space, the network manager configures the TDMA block by calling the APIs of the TDMA driver so that it can adjust the schedule, set parameters, and switch modes as needed.

In SRT-WiFi, there are three types of network managers forming a hierarchical structure: the central network manager (CNM) managing all network resources, cluster managers (CM) operating on the APs, and device managers (DM) operating on the STAs. During the process of joining an SRT-WiFi network, the CNM starts first, awaiting TCP connections from CMs to distribute schedules to the links. Following this, CMs initiate the cluster networks, with slave APs synchronizing with the master AP on the same channel, and then await STA connections. To simplify the synchronization and the joining process, beacon and shared slots remain fixed throughout system operation, and this information is broadcasted among all devices. Once a STA powers on, it scans the channels, synchronizes with the designated AP, and joins the network.

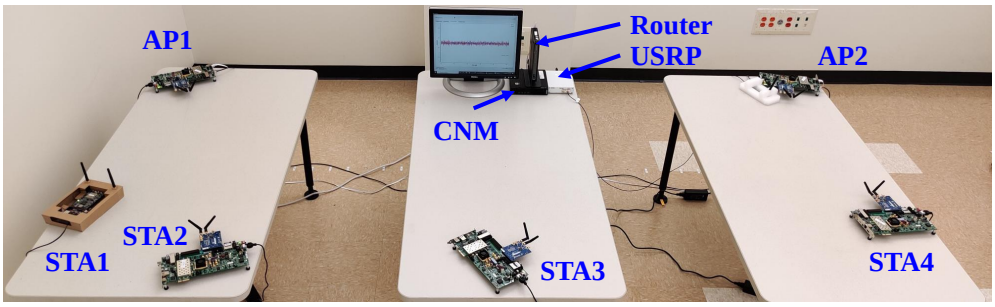

Figure 1.8: An overview of the multi-cluster SRT-WiFi testbeds.

Once the network is joined, the DM on the STA establishes a TCP connection with the CM on the AP to receive and update the schedule. Until the schedule is received, the STA uses shared slots to complete the joining process. Unlike assigned slots, shared slots are contention-based, where each sender first undergoes a random backoff, similar to CSMA mode, and then senses the channel. During operation, DMs and CMs on each device monitor link qualities and interference. This channel information is collected by the CNM, which then determines and updates schedules and data rates for the DMs and CMs to adapt to current channel conditions, ensuring stable transmissions.

A unique aspect of SRT-WiFi network management is its capability to dynamically adjust slot lengths within the schedule to support rate adaptation during real-time transmission. While the maximum transmission unit (MTU) for an individual link remains fixed, the data rate may vary depending on interference levels. A lower data rate requires more time to transmit a packet of the same length, which can exceed the time slot boundary and lead to collisions. SRT-WiFi addresses this issue with dynamic slot lengths. In the schedule, an atomic slot (AS) is defined as the shortest slot length that can support transmitting an MTU-sized packet at the highest rate. When transmitting at a lower rate, a packet can use multiple consecutive atomic slots without

preemption. Therefore, by selecting different rates during runtime, the packet transmission can occupy varying atomic slots.

1.3.3. Performance Evaluation

A testbed of multi-cluster SRT-WiFi is implemented to perform a comprehensive evaluation. This network configuration setting includes two APs, AP1 and AP2 for Cluster1 and Cluster2, operating on a single channel, with each AP having two STAs connected, STA1 and STA2 in Cluster1, STA3 and STA4 in Cluster2. Fig. 1.8 provides a testbed overview with a total of four links. CNM and APs are connected to a router in the testbed, forming a backbone network, and stations connect to their APs accordingly. Also, a USRP device is employed to create interference via a noise antenna positioned near AP2.

Due to the page limit, this chapter presents a single experiment to showcase the rate adaptation function in SRT-WiFi. In this experiment, interference is introduced to the testbed. Each device measures its reception SNR and reports it to the CNM. The CNM then constructs the schedule and selects appropriate data rate. In the experiment, We add the interference on the AP side and let the station both send UDP packets to the AP and measure the PDR and SNR. The level of interference is not fixed but varied every 0.5 seconds, meaning that in the first half of each second, the interference rises to a set level while in the next half of that second, the interference shuts down so that the interference varies fast.

Fig. 1.9 (b) shows the measured SNR of the channel and Fig. 1.9 (a) provides a closer look at a portion of the measured SNR to illustrate the interference variations. The SNR values decrease from 27 to 12 dB and rise gradually.

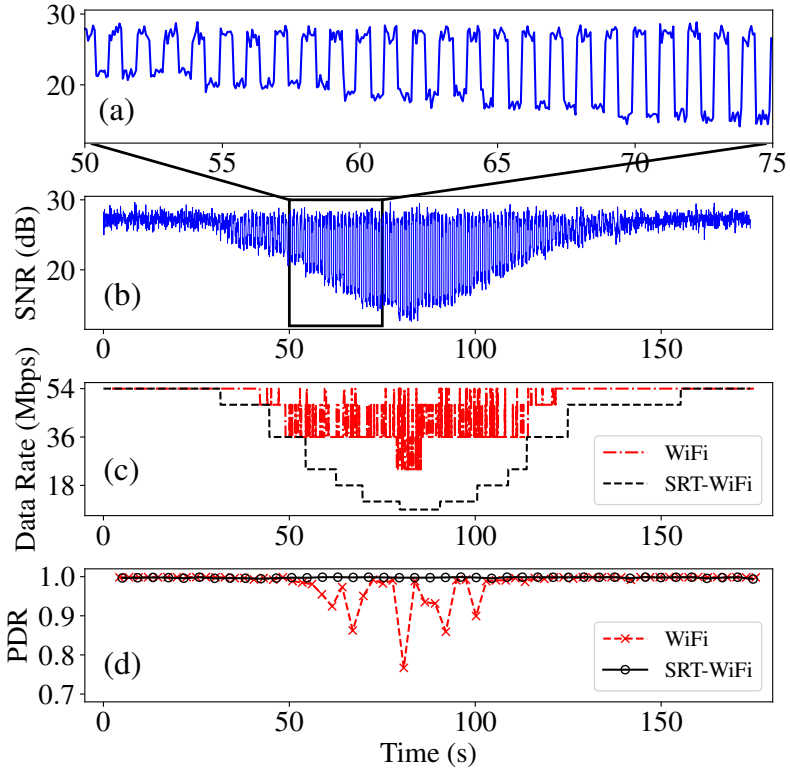

Figure 1.9: Data rate and throughput comparison between SRT-WiFi and regular WiFi in the presence of interference.

Fig. 1.9 (c) presents the data rates for both SRT-WiFi and standard WiFi, the latter of which uses the Minstrel algorithm [Xia et al., 2013] for rate adaptation based on transmission history. The corresponding PDR is shown in Fig. 1.9 (d). Observing The experiment result reaveals that standard WiFi cannot maintain stable transmissions when the SNR value falls below 20 dB. In contrast, SRT-WiFi employing a rate adaptation method can provide stable transmissions under different SNR conditions. The CNM buffers the measured SNR values over a time window and adjusts the data rate based on the lowest SNR value in the buffer; once a lower SNR is detected, the data rate is immediately reduced and does not increase until all buffered SNR values exceed the threshold for

a higher rate. This method wastes some resources when the channel condition is good, but it ensures stable transmissions. The performance of rate adaptation in SRT-WiFi under interference is shown in Fig. 1.9 (c), characterized by a stepped pattern without rapid changes. Fig. 1.9 (d) illustrates the PDR of SRT-WiFi during the test, demonstrating stable performance thanks to the rate adaptation mechanism.

1.4. GR-WiFi based on 802.11a/g/n/ac

SRT-WiFi currently provides real-time, reliable wireless communication for industrial control applications but is limited to IEEE 802.11a/g PHYs and SISO communications. Our long-term goal is to enhance SRT-WiFi to support newer WiFi standards such as IEEE 802.11n/ac/ax. As the first step towards this goal, we extend SRT-WiFi on GNU Radio, a popular open-source SDR platform, and introduced GR-WiFi, a GNU Radio-based open-source platform for IEEE 802.11 research. Note that there is already a GNU Radio implementation for IEEE 802.11/a/g/p available as referenced in [Bloessl et al., 2013]. In this work, we implement PHYs of 802.11a/g/n/ac standards on GR-WiFi, which can support the Legacy OFDM (Legacy), high-throughput (HT) and very-high-throughput (VHT) PHY formats with SISO and up to 2×2 SU-MIMO and MU-MIMO. Fig. 1.7 summarizes the three supported PHY formats (Legacy, HT and VHT) in GR-WiFi and detail reference can be found at IEEE 802.11 standards [IEEE 802.11 Working Group, 2021]. In the following sections, we describe the design of the packet transmission and reception functions in GR-WiFi and then present their implementation details on GNU Radio.

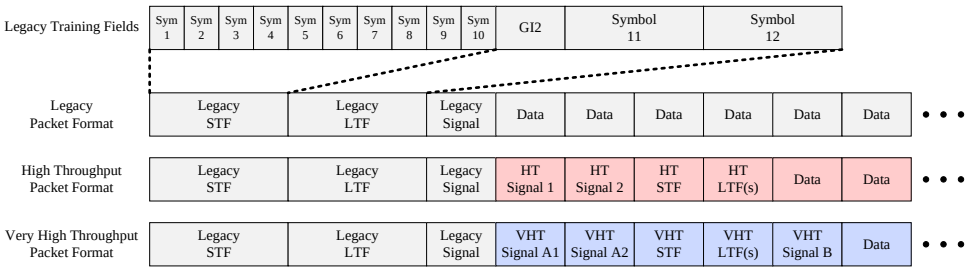

Figure 1.10: IEEE 802.11a/g/n/ac physical layer packet formats.

1.4.1. Packet Transmission Design

The packet transmission function in GR-WiFi follows these steps to generate the packets of different supported formats in 802.11a/g/n/ac standards.

In Legacy format, beginning with the preparation of training field, transmitter converts the given orthogonal frequency division multiplexing (OFDM) training symbol from frequency to time domain waveform by applying Inverse Fast Fourier Transform (IFFT), scales the waveform amplitude with a tone scaling factor and inserts the guard interval (GI). In the Legacy signal field, 24-bit Legacy signal bits, which contain information on packet length and modulation and coding scheme (MCS) go through binary convolutional coding (BCC) and interleave process. The interleaved bits are then modulated with binary phase shift keying (BPSK) and converted to a scaled time domain waveform. The data payload that is distributed into data symbols will go through the same steps as Legacy signal symbols but with an additional bit of scrambling before the BCC.

The HT format's packet transmission is more complex due to the MIMO and beamforming support. First, the signal is generated for multiple spatial streams (SSs) for MIMO transmissions. Each SS has the same packet format as shown in Fig. 1.10. Before each SS is modulated and emitted into the air

through antennas, a cyclic shift is applied to each SS to prevent constructive interference and unintentional beamforming, which happen when the same signal is transmitted through different transmit chains. After applying cyclic shift, the modulation and scaling steps are the same as the Legacy packet. The Legacy signal field in HT packet has the rate set to lowest, and the length is computed to cover the duration of the following HT portion transmission. The HT signal field is the same as Legacy signal field but occupies two symbols. The HT portion starts from the HT training field. Besides the cyclic shift, there is also the spatial mapping for each sub-carrier to apply the required phase for beamforming, called the Q matrix in the standard. HT-STF and each of the HT-LTFs have one symbol. The number of the HT-LTFs is determined by the SS number to provide sufficient channel information for the receiver to estimate the channel(s). In HT data symbols, after coding, stream parsing is performed to separate the coded bits into multiple spatial streams for MIMO Interleave is also applied for each spatial stream but with different phase rotations as specified in the standard. Cyclic shift, waveform scaling, and GI insertion are necessary steps before the packet is ready to be transmitted through the air. The signal generation for the VHT format is similar to HT's, but it supports 256 QAM, up to 8 spatial streams, and a 160 MHz bandwidth.

1.4.2. Packet Reception Design

To design the packet reception in GR-WiFi, we begin by addressing the packet reception trigger. As shown in Fig. 1.10 from sample 10 to 170, the STF has 10 repeated symbols last for 0.8 μs s with 16 samples. We utilize an auto-correlation method that leverages the 10 times repetition of the STF symbol.

This approach is robust against multipath propagation effects CFO distortion during the reception. The auto-correlation output forms a plateau. Once the output passes the threshold, we measure the length of the continuous plateau to detect the STF and initiate the reception process.

Once packet reception is triggered, the next steps involve packet synchronization and fine tune the timing. To fine-tune the timing, we find the maximum auto-correlation within a specified time window and locate two shoulder indices at 80% of the maximum value on both the left and right sides. As illustrated in Fig. 1.10, the Long Training Field (LTF) repeats 2.5 times. The correlation reaches its peak at the start of the LTF Guard Interval 2 (GI2) and drops at the end of the LTF GI2, making the center of this correlation correspond to the middle of LTF GI2.

With the correct timing, we estimate channel and CFO. The CFO of the following samples will be compensated. For each data symbol, it will be converted to frequency domain and then be compensated with channel. The recovered data symbols are demodulated using QAM constellations and then decoded. Currently, the proposed receiver only supports the binary convolutional coding (BCC) with a soft-input Viterbi decoder.

1.4.3. Implementation and Evaluation

We now present the implementation details of GR-WiFi on the GNU Radio. As shown in Fig. 1.11, we implement both SISO and 2×2 MIMO receivers. The 2×2 MIMO can also handle SISO packet reception. However, we design a seperate SISO block because each port in this block processes samples in parallel. Using a MIMO receiver to decode SISO input will keep the second port

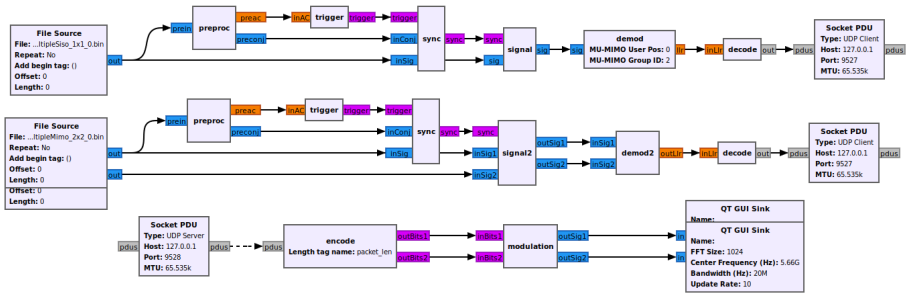

Figure 1.11: GNU Radio implementations of IEEE 802.11a/g/n/ac SISO and 2×2 MIMO receivers and transmitter.

running without processing any samples. For the transmission, the transmitter only generates packet samples, so the waste is negligible.

1.4.3.1. Key Blocks in GR-WiFi implementation

The GR-WiFi implementation on GNU Radio includes the following key blocks:

Pre-Processing: The pre-processing block is a hierarchical block to compute the auto-correlation of the input samples. The values obtained during the auto-correlation computation are reused to compute the coarse CFO. The average blocks use a sliding window with a maximum length to prevent repeated computations of samples and avoid the accumulation of floating-point errors.

Trigger: The trigger block takes the continuous auto-correlation samples as the input and detects the plateau of auto-correlation. Once the plateau length meets a threshold, it generates output flag to trigger the synchronization block. The trigger block also has a state machine to avoid multiple triggers for one packet to avoid wasting computation resources.

Synchronization: The synchronization block is triggered to use auto-correlation of LTF and identify the start of LTF, with the index being passed to the next

block. At the same time, the synchronization block compensates the coarse CFO from the pre-processing block for the LTF samples, re-estimates the CFO with two LTF symbols, and computes the accurate CFO value. This accurate CFO value is passed tp next block using a tag.

Signal: The signal block is triggered by the synchronization block to get the timing of the packet and the estimated CFO. It first compensates the CFO for LTF and Legacy Signal to estimate the Legacy channel and demodulate and decode the Legacy Signal. If the Legacy Signal is correctly decoded, this packet is identified as at least a Legacy packet with the maximum possible timing length corresponding to the Legacy length. The signal block computes the symbol and sample number according to the MCS and packet length. The following input samples within the sample number will be compensated with CFO and passed to the next block. That means the signal block chops the input sample stream and only keeps useful packet samples for further processing. For MIMO receivers, the Signal2 block is used, with the primary difference being that Signal2 also chops samples from the second sample stream to match the length of the first stream and compensates for the CFO of the second stream.

Demodulation: The demodulation block converts the OFDM symbols to QAM constellations and disassembles them into soft bits. A state machine is used to determine packet format. It first updates the Legacy channel and checks whether legacy MCS is the lowest. The lowest MCS leads to further demodulation and decodes on following two symbols to check the HT Signal and VHT Signal A, which will decide the following demodulation. If the Non-Legacy checking fails, the packet is demodulated as Legacy. If it is either HT or VHT, the channel is re-estimated, and packet is then demodulated accordingly. For the OFDM part, the channel is compensated after FFT, and pilots are used to

correct the residual CFO. The QAM constellations are disassembled into soft bits and deinterleaved. Some steps are simplified to speed up the processing, such as deinterleaving using a pre-defined lookup table for each symbol but not following the method given in the standards. The demodulation block outputs the soft bits to the next decoding block.

The difference between the demodulation block and demodulation-2 block lies in their design purposes and functionalities. The demodulation block is intended for SISO and MU-MIMO receivers. It can perform channel sounding and receive MU-MIMO packets with group number and position in the group to estimate the corresponding channel and demodulate accordingly. On the other hand, the demodulation-2 block is designed for the AP side in MU-MIMO, which can handle two stream inputs simultaneously. To simplify processing, channel sounding function is removed. However, we consider adding full functionality to all blocks in future developments.

Decoding: The decoding block processes a bit stream input and outputs a message stream, which is the required input type for the socket PDU block. It takes the input soft bits with a specified trellis length, decodes the packet, and checks the cyclic redundancy check 32 (CRC32). The Viterbi decoder performs a forward update for the whole packet and then traces back from the very end, which is the 6-bit-zero tail. This approach is used because GNU Radio accumulates some samples and then provides them to the blocks so that the proposed receiver does not aim for real-time performance in the communication stack. If the packet is correct, decoding block passes the packet to the Python MAC layer through a UDP message for further customized processing. In case of a null data packet (NDP) used for channel sounding that has no bits to decode, the decoder simply packages the channel information into a UDP message and

sends it to the MAC layer.

1.4.3.2. Performance Evaluation

We now present the performance evaluation of GR-WiFi through simulations and real-world testbed experiments.

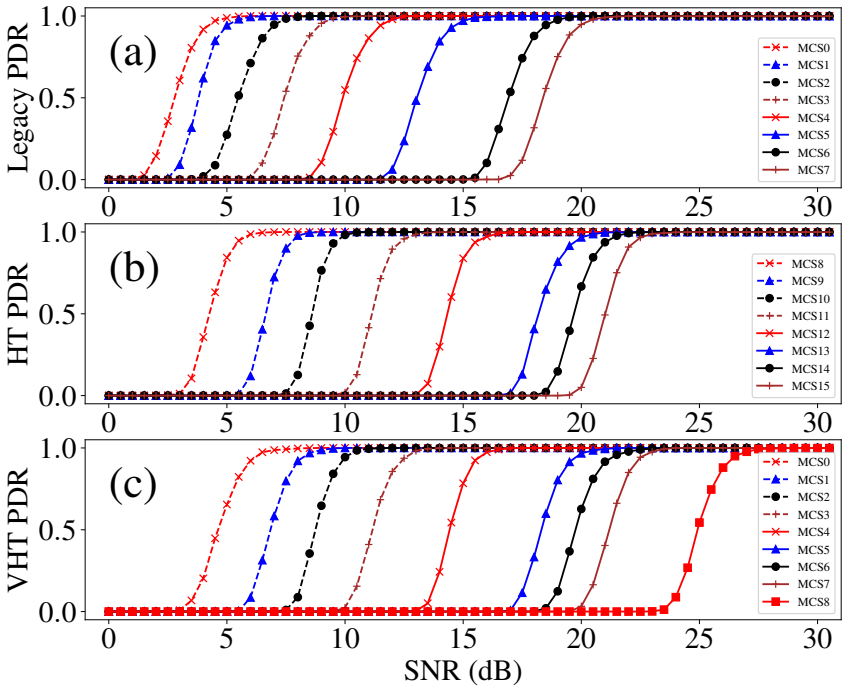

Figure 1.12: GR-WiFi packet delivery ratio against signal-to-noise ratio.(a)Legacy format packet, (b)HT format packet, and (c)VHT packet.

We generate simulation signal samples and have the receiver demodulate. The receiver successfully receives the packet under low SNR conditions, indicating its enhanced performance in handling lower-quality signals. A higher SNR value means a better wireless link that can support higher MCS, leading to higher data rates. Fig. 1.12 presents the packet delivery ratio (PDR) of GR-

WiFi for three different format packets under different SNR conditions with different MCS. The figure reveals that a VHT format may require a higher SNR than the other two formats to achieve the same PDR. Fig. 1.13 presents the PDR of GR-WiFi VHT and HT for SU-MIMO transmissions. With the additive white Gaussian noise (AWGN) channel simulation, the performance is similar to SISO. However, the performance drops in real-world multi-path propagation conditions.

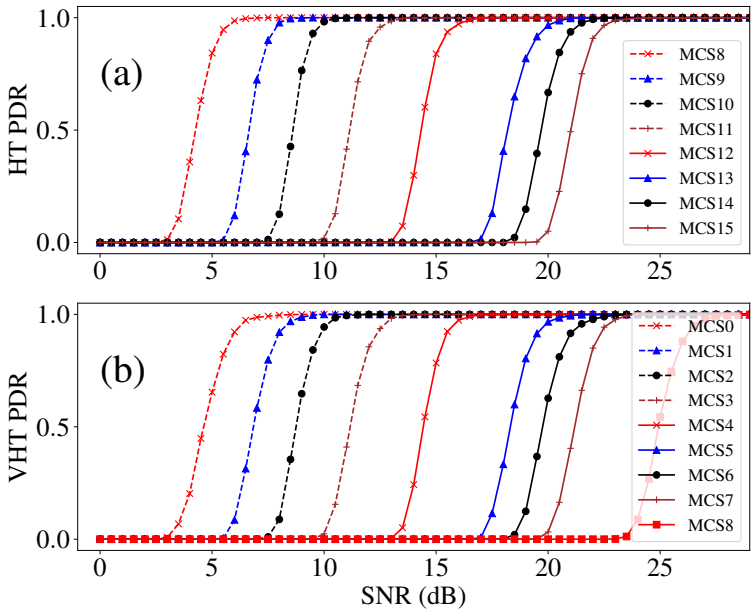

Figure 1.13: (a) GR-WiFi HT SU-MIMO packet delivery ratio against SNR. (b) GR-WiFi VHT SU-MIMO packet delivery ratio against SNR.

Fig. 1.14 shows our MU-MIMO testbed, which aims to demonstrate the simultaneous transmissions between AP and two stations. Equipment deployed for the testbed includes one USRP B210 with 2×2 TRX antenna array and two USRP B200 with 1×1 TRX antennas. One complete transmission involves the following steps: the AP first broadcasts NDP packets to have all stations'

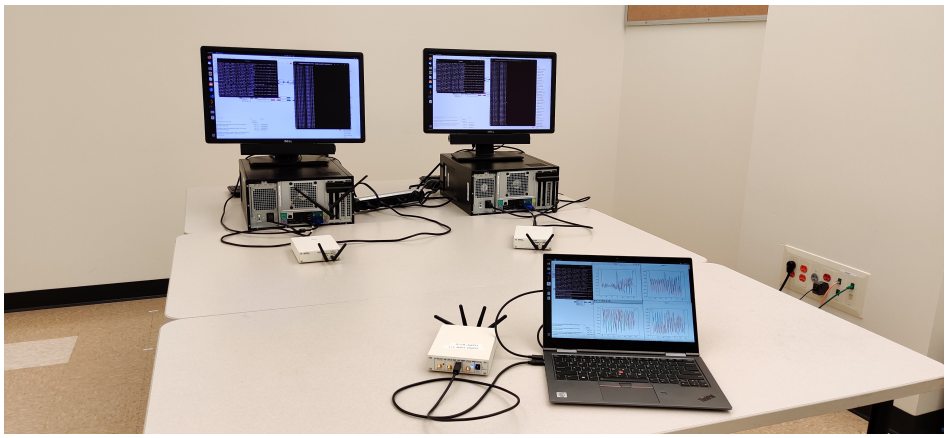

Figure 1.14: GR-WiFi MU-MIMO testbed with 1 AP (laptop) and two STAs (desktops) where real channel response is shown at the AP side.

attention. Stations receive and capture the two LTFs in the NDP packets. Each station sends its two received LTFs back to AP. AP gathers all the LTFs information from stations and calculates the steering matrix accordingly. Based on the steering matrix, AP can then transmit packets biased in a particular direction based on where the station is. In our testbed, we demonstrate the successful reception in both stations.

The current implementation of GR-WiFi has certain limitations, primarily due to its inability to fully connect to a real-world WiFi network. This is because the receiver cannot return the acknowledgment (ACK) to the sender within the designated ACK timeout period. Nonetheless, it can operate as a passive receiver to intercept packets transmitted over the air. Despite its limitations, GR-WiFi is a crucial step in exploring new protocols, and its successful deployment on GNU Radio offers a valuable reference for future implementations on SRT-WiFi.

1.5. Conclusion and Future Work

In this chapter, we review three different WiFi implementations for supporting high-speed and real-time industrial control applications. The RT-WiFi solution is based on COTS hardware. SRT-WiFi is implemented on an advanced SDR platform where the radio functions are programmed on FPGA to support hard real-time performance. GR-WiFi is implemented on GNU Radio-based SDR platform, supporting a much shorter development period. Extensive experiments have been conducted on both SRT-WiFi and GR-WiFi for functional validation and performance evaluation.

As future work, we will port the implementations of 802.11n/ac PHYs from GR-WiFi to SRT-WiFi to make it full-blown and support hard real-time performance. We will add newer standards to SRT-WiFi, such as IEEE 802.11ax. Both FPGA-based SRT-WiFi and GNU Radio-based GR-WiFi implementations, once mature, will be made public to the wireless communities to support a broad range of research and development (R&D) activities.

Bibliography

Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. Decoding IEEE 802.11a/g/p OFDM in Software Using GNU Radio. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom'13, page 159–162, New York, NY, USA, 2013.

GNU Radio Foundation. GNU-Radio, 2007. URL https://www.gnuradio.org/.

IEEE 802.11 Working Group. 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 and Physical Layer Specifications. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–7524, 2021.

Xianjun Jiao, Wei Liu, Michael Mehari, Muhammad Aslam, and Ingrid Moerman. Openwifi: a free and open-source IEEE 802.11 SDR implementation on SoC. In *IEEE 91st Vehicular Technology Conference*, pages 1–2, 2020.

Edward A Lee and David G Messerschmitt. *Digital communication*. Springer Science & Business Media, 2012.

Quan Leng, Yi-Hung Wei, Song Han, Aloysius K Mok, Wenlong Zhang, and Masayoshi Tomizuka. Improving control performance by minimizing jitter in rt-wifi networks. In *2014 IEEE Real-Time Systems Symposium*, pages 63–73. IEEE, 2014.

Quan Leng, Wei-Ju Chen, Pei-Chi Huang, Yi-Hung Wei, Aloysius K Mok, and Song Han. Network management of multicluster rt-wifi networks. *ACM Transactions on Sensor Networks (TOSN)*, 15(1):1–26, 2019.

Daniel Camps Mur. Linux Wi-Fi open source drivers-mac80211, ath9k/ath5k, 2011. http://campsmur.cat/files/mac80211_intro.pdf.

Federico Tramarin, Aloysius K. Mok, and Song Han. Real-Time and Reliable Industrial Control Over Wireless LANs: Algorithms, Protocols, and Future Directions. *Proceedings of the IEEE*, 107(6):1027–1052, 2019.

Yi-Hung Wei, Quan Leng, Song Han, Aloysius K Mok, Wenlong Zhang, and Masayoshi Tomizuka. RT-WiFi: Real-time High-Speed Communication Protocol for Wireless Cyber-Physical Control Applications. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 140–149. IEEE, 2013.

Yi-Hung Wei, Quan Leng, Wei-Ju Chen, Aloysius K Mok, and Song Han. Schedule adaptation for ensuring reliability in rt-wifi-based networked embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(5):1–23, 2018.

Dong Xia, Jonathan Hart, and Qiang Fu. Evaluation of the Minstrel rate adaptation algorithm in IEEE 802.11 g WLANs. In *2013 IEEE International Conference on Communications (ICC)*, pages 2223–2228. IEEE, 2013.

Zelin Yun, Peng Wu, Shengli Zhou, Aloysius K. Mok, Mark Nixon, and Song Han. RT-WiFi on Software-Defined Radio: Design and Implementation. In *2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 254–266, 2022.