Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets

Francesco Gringoli
CNIT/University of Brescia
Italy
francesco.gringoli@unibs.it

Matthias Schulz
TU Darmstadt
Germany
mschulz@seemoo.tu-darmstadt.de

Jakob Link
TU Darmstadt
Germany
jlink@seemoo.tu-darmstadt.de

Matthias Hollick
TU Darmstadt
Germany
matthias.hollick@seemoo.tu-darmstadt.de

ABSTRACT

Modern wireless transmission systems heavily benefit from knowing the channel response. The evaluation of Channel State Information (CSI) during the reception of a frame preamble is fundamental to properly equalizing the rest of the transmission at the receiver side. Reporting this state information back to the transmitter facilitates mechanisms such as beamforming and MIMO, thus boosting the network performance. While these features are an integral part of standards such as 802.11ac, accessing CSI data on commercial devices is either not possible, limited to outdated chipsets or very inflexible. This hinders the research and development of innovative CSI-dependent techniques including localization, object tracking, and interference evaluation. To help researchers and practitioners, we introduce the *nexmon CSI Extractor Tool*. It allows per-frame CSI extraction for up to four spatial streams using up to four receive chains on modern Broadcom and Cypress Wi-Fi chips with up to 80 MHz bandwidth in both the 2.4 and 5 GHz bands. The tool supports devices ranging from the low-cost Raspberry Pi platform, over mobile platforms such as Nexus smartphones to state-of-the-art Wi-Fi APs. We release all tools and Wi-Fi firmware patches as extensible open source project. It includes our user-friendly smartphone application to demonstrate the CSI extraction capabilities in form of a waterfall diagram.

WiNTECH '19, October 25, 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6931-2/19/10…$15.00
https://doi.org/10.1145/3349623.3355477

ACM Reference Format:
Francesco Gringoli, Matthias Schulz, Jakob Link, and Matthias Hollick. 2019. Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets. In *13th International Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization (WiNTECH '19), October 25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3349623.3355477

1 INTRODUCTION

The availability of tools for measuring Channel State Information (CSI) is considered the key-enabler for the development of new opportunistic sensing techniques [1]. CSI represents the frequency response of a communication channel, that heavily depends on the environment. By analyzing CSI of received Wi-Fi frames, even minor modifications of the surroundings can be spotted with higher chance than by considering the Received Signal Strength Indicator (RSSI) [2]. For each pair of transmit and receive antennas, CSI provides the receiver with a set of values $a_n \cdot \exp(j\theta_n)$ reporting attenuation a_n and phase shift θ_n for each carrier n, while RSSI provides only the overall received energy.

Some factors make Wi-Fi particularly suited for opportunistically using frames originating at a static transmitter to gather knowledge of physical variations in ones proximity. First, Wi-Fi is a universally adopted standard—CISCO predicts that by 2022 there will be nearly 549 million public Wi-Fi hotspots[1]. Second, Wi-Fi hardware is produced at a low price per device. Third, Wi-Fi supports Orthogonal Frequency-Division Multiplexing (OFDM) that is the basis for easily measuring a channel's frequency response per-frame. In the first place this information is required for equalization at the receiver and, additionally, builds the base for both implicit and explicit channel sounding used to configure beamforming [3]. Despite the benefits access to CSI would

[1]From "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper"

offer to users, chip manufacturers keep CSI as a "private" feature. Only few devices using the outdated communication standards 802.11g and 802.11n are able to dump CSI with various limitations. This creates an implementation gap between the capabilities modern Wi-Fi chips can offer with respect to large bandwidths and number of antennas and the desires of researchers to exploit such capabilities to collect more detailed CSI measurements.

In 2018, we demonstrated in [4] for the first time how to collect 80 MHz CSI data on a commercial device (a Nexus 5 smartphone). At the time, however, we did not provide any measurement about the quality of the data: we could not even infer it from the manufacturer since the chipset does not officially include CSI extration among its features. As this could be an issue for many applications that require high quality data, we report in this work the tests that we ran to validate the CSI extraction feature that we discovered. In addition, in this paper we overcome the main limit of our previous work: more specifically the system is now a real CSI extraction tool compatible with multiple chipsets manufactured by Broadcom and Cypress, that can be configured everywhere with the same syntax and export data in a unified format. Apart from smartphones (including Nexus 5 and now 6P), we added support for the wide-spread, low-cost Raspberry Pi 3B+ and 4B devices backing high-density CSI extraction testbeds; and for the rt-ac86u access point from Asus that differently from the other devices comes with four antennas. With respect to existing tools, ours has the following outstanding features: First, it is compatible with Very-High-Throughput (VHT) encodings defined by 802.11ac and can work up to 80 MHz bandwidth or 242 subcarriers. Second, it supports various transmit-receive antenna combinations (up to 4×4 MIMO). Third, it can export the entire CSI data without the need of suppressing selected carriers. Fourth, it offers flexible capture filters to selectively collect CSI data from chosen transmitters. Fifth, the whole patch code is available for custom extensions at https://nexmon.org/csi.

We believe this tool will provide researchers and practitioners with an unrivalled facility for experimentally verifying algorithms with state-of-the-art Wi-Fi devices. In particular, it will increase the precision of localization techniques, either based on Angle-of-Arrival or on Fine-Timing-Measurements. Here, larger bandwidths allow to resolve multipath propagation issues [6]. It will contribute to the development of more accurate techniques for human and object tracking—including gestures where spectral resolution plays an important role [7]. And it will provide powerful primitives for the verification of physical layer security algorithms [8]. It can also help circumventing inferior security mechanisms by enhancing physical layer sniffers [9].

The paper is organized as follows: We review known CSI extraction tools in Section 2. In Section 3, we demonstrate that the data we extract with our tool is really a CSI. In Section 4, we explain how our tool works and we present results in Section 5 before closing the paper with Section 6.

2 RELATED WORK

Even though CSI is measured by Wi-Fi receivers as a standard feature, access to it is not common on off-the-shelf devices. University of Washington provides the Linux 802.11n CSI Tool enabling CSI extraction on Intel Wi-Fi Link 5300 radios [10]. The toolkit uses a customized proprietary Intel firmware and the open-source iwlwifi wireless driver. CSI is reported as 30 subcarrier groups taken over the 56 or 114 subcarriers of 20 or 40 MHz channels, each holding signed 8-bit real and imaginary parts. Non-grouped CSI can be obtained using the Atheros CSI Tool maintained by the Wireless And Networked Distributed Sensing (WANDS) research group [11]. It is build on top of the open-source Linux kernel driver ath9k and provides CSI access on Qualcomm Atheros 802.11n Wi-Fi chipsets including the AR9580, AR9590, AR9344, and QCA9558. Real and imaginary parts are 10-bit values and reported for each subcarrier on 20 and 40 MHz channels. Both tools can operate in the 2.4 and 5 GHz band. However, none supports 80 MHz wide channels introduced with the newer 802.11ac standard.

Surprisingly, new Qualcomm Atheros 802.11ac cards powered by the `ath10k` driver do not support CSI collection, neither officially nor with alternative firmwares like those provided by Candela Technologies that has access to Qualcomm's proprietary sources. Still these chipsets include a built-in spectral analysis feature that can report Fast Fourier Transform (FFT) samples computed by the baseband circuitry [12]. Those samples are not related to (or triggered by) any specific frame so that the tool is more oriented on the evaluation of spectral interference rather than sensing a communication channel.

Our current work on Broadcom and Cypress Wi-Fi chips first started on BCM4318 SoftMAC devices using the old 802.11g standard and was mainly used for refining Time-of-Arrival measurements [13]. In 2018, we created the first CSI extractor for smartphone Wi-Fi chips (BCM4339 on the Nexus 5) and used it for the implementation of a practical physical layer covert channel [4].

Recenty, Quantenna Communications, a division of On Semiconductor that manufacturers Wi-Fi chipsets for high-end/enterprise Access Points, released a solution for exporting CSI data from 11ac APs serving stations. This solution, however, is only available to chipset customers and system integrators and there is no report about its performance [14].

As an alternative to off-the-shelf devices, Software Defined Radios (SDRs) like the Wireless Open Access Research Platform (WARP) [15] and the Universal Software Radio

Peripheral (USRP) [16] can be used for CSI measurements. Some of their front-ends support up to 200 MHz bandwidth. However, also their prices are roughly 200 times higher than the Raspberry Pi's we support with our framework.

In Table 1, we report the features of the aforementioned tools together with those introduced by us in this work. We do not include SDR solutions as our focus is on off-the-shelf devices. In column *Device* we report what are the typical architectures hosting the *Supported Chipset*. Column *Supp. Std.* reports the highest type of physical layer for which the tool can report CSI. VHT stands for Very-High-Throughput and is typically used on 802.11ac devices (5GHz only), while High-Throughput (HT) is found in 802.11n devices (both 2.4GHz and 5GHz). Both are OFDM-based which was introduced with 802.11a (5GHz) and back-ported to 2.4GHz with 802.11g. Our nexmon CSI Extractor is downwards compatible and supports 802.11a/g/n/ac transmissions in both frequency bands using the proper nexmon patches. 802.11b is not supported as it is based on Direct Sequence Spread Spectrum (DSSS). The *number of subcarriers (NSC)* generally depends on the maximum bandwidth, except for the Linux 802.11n CSI Tool which is limited to 30 subcarrier groups. All tools provide complex numbers with inphase and quadrature components that are either represented as signed integers or floating-point numbers.

3 WI-FI "CRIME SCENE INVESTIGATION"

CSI extraction resembles "Crime Scene Investigation" for 802.11. It allows to understand what happens on the channel between the transmitter of a frame and the receiver. CSI is computed by analysing how the preamble with known content is modified during transmission. As a result, we obtain a set of complex numbers in the form $a_n \cdot \exp(j\theta_n)$ where n is the index of the carrier and the total number of carriers depends on the modulation as we already reported in Table 1. While for devices where CSI support is claimed from the manufacturer the data is trusted "as is", in the case of an unsupported platform like Broadcom or Cypress, finding some data that *resembles* CSI requires deeper investigation. Before explaining our tool, we first report our test procedure to verify that the extracted data is actually CSI.

To this end, we illustrate the amplitude of the data that we collected with our tool when transmitting an OFDM/11g frame from a single antenna device connected to a four antenna receiver using a splitter and some cables in Figure 1. Due to the absence of a wireless channel, one would expect a flat behaviour and the same amplitudes over all subcarriers. However, this is not the case. We further notice different amplitudes at the four receive chains. This is due to the bending of cables: slightly shaking the cables results in changes in the extracted data, which demonstrates the sensibility of the measurement to the environment—as expected from a real CSI. We also notice that the subcarriers in the middle (carrier #0) are zero. This is expected as in all Wi-Fi modulations the carrier is always suppressed. Finally we notice a general tendency to decay when moving away from carrier #0: also this behaviour is expected because of analogue channel filtering. At least qualitatively, the data we collect with our tool verifies the typical properties of a CSI. We now provide a quantitative (even though naïve) demonstration using a couple of experimental tests.

3.1 Noise test

We first ran an experiment to verify the mapping between the extracted tones and the corresponding frequencies. In a controlled environment we transmitted packets on channel 14 in saturation and we collected all the corresponding CSI values at a receiver with our tool. At the same time, a narrowband noise generator performs periodic sweeps in the range of $[-4MHz, 4MHz]$ with respect to the channel center. In

Table 1: Features of different CSI extraction tools, including maximum bandwidth (BW), number of spatial streams (NSS), number of receive chains (NRX), number of subcarriers (NSC), and resolution (Res.) of numbers (i = signed integer, f = signed float mantissa).

Tool	Open Source	Device	Supp. Chipset	max. BW	NSS×NRX	Supp. Std.	NSC	Res.
nexmon CSI Extractor	yes	Router, PCIE e.g. Asus RT-AC86U	BCM43{65, 66}	80 MHz	4×4	VHT/11ac	242	12 bit (f)
	yes	Smartphone, IoT e.g. Nexus 5/6P, RPi3B+/4B	BCM43{39, 58, 455}	80 MHz	1×1	VHT/11ac	242	14 bit (i) or 10 bit (f)
Linux 802.11n CSI Tool	no	PCI	IWL5300	40MHz	3x3	HT/11n	30	8 bits (i)
Atheros CSI Tool	yes	Router, PCIE	AR9580, AR9590 AR9344, QCA9558	40MHz	3x3	HT/11n	114	10 bits (i)
OpenFWWF CSI Tool	no	Router, PCI e.g. Linksys WRT54GL	BCM4318	20MHz	1x1	OFDM/11g	52	12 bits (i)

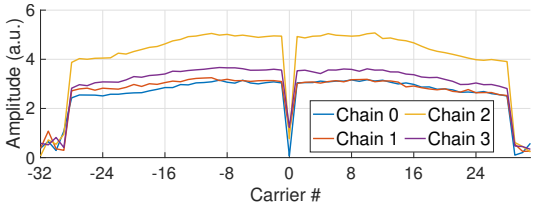


Figure 1: CSI extracted from a 1x4 setup (1 tx antenna, 4 rx antennas) with a 1-to-4 splitter and equally long cables. Different amplitudes likely due to splitter characteristics and not perfectly straight cables.

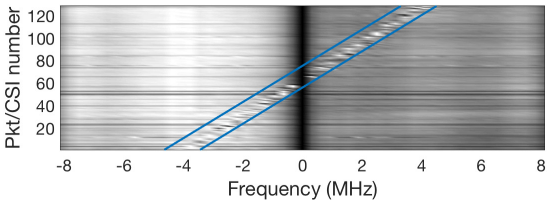


Figure 2: Experiment with narrow band noise: the noise corridor is emphasized with the two solid lines and perfectly follows the configured sweep.

Figure 2, we report the captured CSI data corresponding to one period/sweep of the noise generator. The narrow-band noise appears where it should be. We emphasize it with two solid, blue lines. It demonstrates a perfect match between the noise frequency and the captured CSI over time (vertical axis represents number of received packets).

3.2 MUSIC test

We then performed a second experiment for truly assessing the quality of the data and the continuity of the CSI over adjacent frequency windows. We repeated the large band experiment from [6] for discriminating two paths very close in space [17] by means of the MUSIC algorithm. In this case, we completely cover the 2.4 GHz band by sweeping through channels [1..13] to increase the spatial resolution: a single 20 MHZ CSI, in fact, could not distinguish paths differing less than 15 meters. We send 20 MHz packets with transmitter and receiver connected through a pair of cables of different length ($\Delta = 10\,m$) with the help of a splitter and a mixer.

In Figure 3, we show in the top graph the combination of CSI data from 13 channels. As adjacent channels partially overlap, we simply take fewer tones from each. The amplitude clearly shows a very low frequency ripple due to the small relative delay between the two paths.

After applying the MUSIC algorithm to the complex CSI data, we obtain the curve marked with triangles in the bottom sub-figure where the second peak to the right is exactly located in $\Delta = 10\,m$. A similar experiment with $\Delta = 50\,m$ confirms the result (other curve marked with stars).

4 NEXMON CSI EXTRACTOR TOOL

In this section, we describe the internal working of the Broadcom chipset as well as the required modifications in order to collect CSI on the fly for each packet and deliver this information to userspace. For more details about the chipset we refer readers to [5, 18].

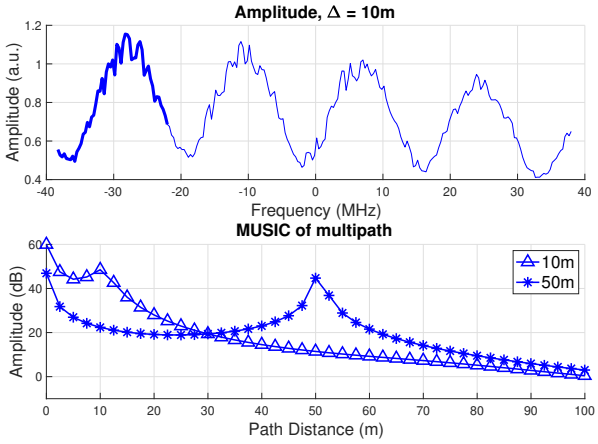


Figure 3: Results from adjacent channels. Top: channels overlap at borders, thicker line is data from channel 1. Bottom: MUSIC diagram over the top spectral data ($\Delta = 10m$) and for another setup ($\Delta = 50m$).

4.1 Hardware description

Our tools support FullMAC chipsets manufactured by Broadcom or Cypress. The latest versions of these chipsets are 802.11ac compatible and—from a CSI extraction point of view—mainly differ in the number of transmit and receive chains resp. spatial diversity. The underlying design is very similar to the architecture shown in Figure 4. FullMAC means that the processing of wireless frames is offloaded to a dedicated subsystem that acts as a bridge between the Ethernet and Wi-Fi domains. The main host executing a Linux kernel, simply handles Ethernet frames addressed to/coming from a Wi-Fi node. The interfaces to the Wi-Fi chip are Linux kernel modules. Depending on the platform, different modules are used: On the Asus router it is *dhd.ko*, on the Raspberry Pi family it is *brcmfmac.ko* and on Android smartphones it is *bcmdhd.ko*. Being part of the Linux kernel both brcmfmac and bcmdhd are open source. Nevertheless, none of them has access to CSI information.

The main components of the Wi-Fi chipset are:

- **FullMAC ARM CPU**: This dedicated unit executes the *firmware* that is responsible for non-real time operations. The code offloads the majority of Wi-Fi functionalities, from frame preparation/parsing to handling association. Part of the code resides in ROM but can be temporarily changed by adding new functions in RAM according to a mechanism used by Broadcom when releasing official patches. The firmware communicates with the main host through various interfaces such as PCI Express or SDIO. And it has direct access to the D11 MAC core for handling received and providing transmission frames. The data transfer rate is enhanced by using Direct Memory Access (DMA) controllers. To control the firmware, the Wi-Fi kernel module can send ioctls to the firmware. The firmware can also directly communicate with the components of the *physical layer* and the *radio front-end*.
- **D11 MAC core**: This microcontroller executes the *ucode* resp. microcode and implements a Programmable State Machine (PSM) that manages real-time operations like frame aggregation, retransmissions, acknowledging and scheduling. The PSM code is a never-ending loop that reacts on hardware conditions and executes handlers that configure hardware registers and can change the PSM state. Its code resides in a writable *ucode memory*, and it keeps variables and states in the *shared memory*. The D11 core communicates with the FullMAC ARM CPU through FIFO queues and read/write operations in the *shared memory* and it directly accesses the underlying radio front-end.
- **Radio front-end**: This is the physical interface. It manages frame detection and decoding, frame encoding and transmission, and all other low level activities like channel energy detection. It performs operations in the base-band and is responsible for up/down-converting and amplifying the radio-frequency signal.

The PHY block in the RADIO contains several tables that are used for configuring the device or temporarily stores quantities that depend on received frames. Some of them contain the CSI data. They change after the reception of a new preamble, and are constant until the next preamble is processed. Even though they can be accessed from the FullMAC ARM CPU, this is not practical as the ARM unit is asynchronous with the actual stream of transmitted and received frames. Tables can also be accessed from the D11 MAC core. Being responsible for actual real-time frame transmission and reception, this core can collect CSI data as soon as a frame's preamble is received.

We next discuss the journey of a received packet from the air to the Linux host. Subsequently, we report how we deployed the CSI collection feature.

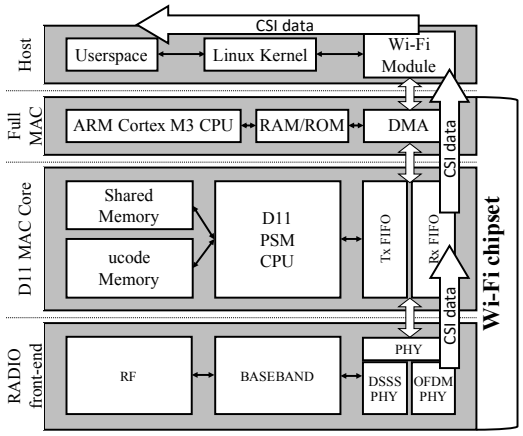


Figure 4: Broadcom FullMAC device high-level view. D11 core pushes CSI data from PHY tables to the ARM CPU that embeds it into UDP datagrams.

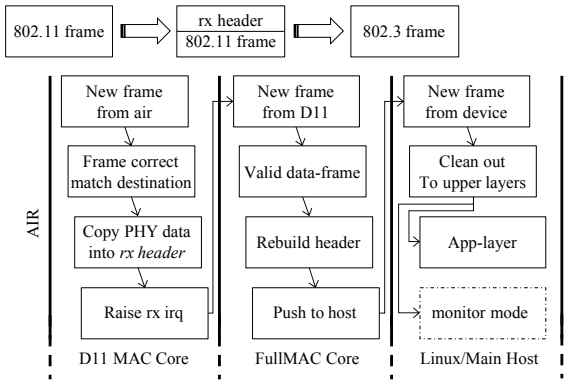


Figure 5: Original frame processing chain.

4.2 Receive path

Receiving packets is a distributed activity involving several operations. We show the journey of a packet in Figure 5 and we describe the main actions in the following.

- **Actions performed by the D11 MAC Core**: Once the main loop of the ucode observes that a new preamble was detected by the hardware, the D11 core reacts immediately and executes some handlers that decide to either abort or go ahead with the reception. In the latter case, the state machine is configured for waking up at the end of the reception. Once the reception terminates—if the frame is deemed correct—the ucode executes some handlers for extracting details on the received frame from PHY registers. This includes the bandwidth, number of spatial streams, energy detected on each antenna etc. It creates a *receive header* and next *pushes* the frame to the receive FIFO by raising an IRQ.

- **Actions performed by the FullMAC Core**: After receiving the IRQ request from the underlying D11 core, the FullMAC CPU performs frame type dependent actions. In case of a data frame, it performs sanity checks to verify the frame can be accepted (like checking it comes from a connected peer such as the AP). If accepted, it performs a cleanup by removing the receive header and converting the 802.11 Wi-Fi header into an 802.3 Ethernet header. The resulting frames are eventually pushed to the host.
- **Actions performed on the host**: The host receives the frame through the Wi-Fi device kernel module that, after additional sanity checks, delivers it to some application through the standard socket library.

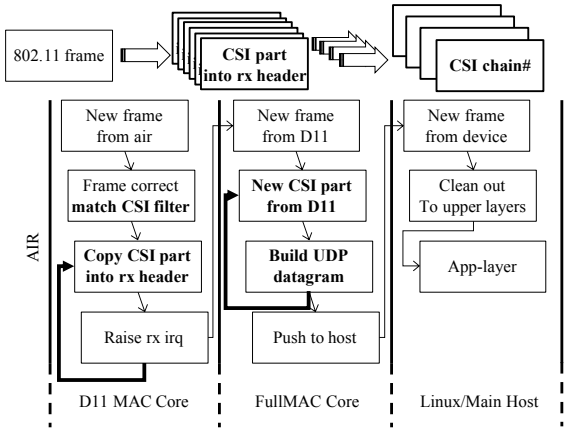


Figure 6: Boldface: modifications for receiving CSI.

4.3 Modified CSI-aware receive path

Figure 6 shows the journey of CSI data: it starts from the radio, from which the D11 MAC CPU extracts CSI data split in multiple parts, and it terminates at the user space where some application is either saving to storage or displaying the recollected CSI data. We emphasize in boldface the parts of the receive path that we modified.

CSI extraction is triggered in the ucode by the reception of a frame that matches a configurable filter, i.e., a specific MAC address or frame control. When a condition triggers, the modified ucode sets up the PSM to wait until the reception completes, and then starts collecting data from the CSI tables. We cannot add the CSI to the received frame payload, hence, we modified the ucode for embedding CSI data inside the receive header. As the CSI data does not fit within a single receive header, we first execute a *default* push operation that transfers the frame into the receive FIFOs. We then break the CSI data into multiple *CSI parts* and push additional receive headers (together with empty frames) until all CSI parts are delivered. Into the CSI part we also embed counters for letting the next block (in the ARM FullMAC Core) understand which part it is handling. Because of length limits, this loop can iterate up to 256 times, like in the case of a 4x4 reception at 80MHz for a total of 4096 complex tones of 4 bytes each. As this can be a lengthy process, the ucode switches the radio into *deaf mode* to avoid that the current CSI could be corrupted by the arrival of a new frame.

In the ARM FullMAC Core, we receive each of these CSI parts separately. On the CSI part that starts a new CSI data, we allocate space for a new UDP datagram where we copy the consecutive CSI parts. After the reception of the last part, we prepend Ethernet, IP and UDP headers and push the datagram to the host. There, it will be processed by the Wi-Fi driver and passed on into user space where it can be received by any application with UDP sockets. We generate separate datagrams for each spatial stream/receive chain to keep size below the Maximum Transmission Unit (MTU).

As an example, we can execute *tcpdump* to save all UDP-CSI datagrams to a pcap file that we can later process offline using another application such as *Matlab*. Or we can execute an application that listens on the used UDP port and collects and processes the CSI datagrams on-the-fly. To demonstrate, we provide an Android app that creates a waterfall diagram. We further describe it in subsection 5.2.

4.4 Implementation details

Apart from understanding the chip internals, the main challenge behind the CSI extractor tool was actually *modifying* the existing firmware. While Broadcom shares the code of the Wi-Fi module running in the host, it keeps both firmware and ucode proprietary and distributes them as binary blobs. Still, modifying the behaviour of the Wi-Fi chip is possible, with the help of two tools that allow firmware modifications. The first one is OpenFWWF [19]. As an open ucode implementation for Broadcom chips, it provides a basic understanding of the ucode [20] that was gained by disassembling, analyzing, modifying and reassembling ucodes of SoftMAC Wi-Fi chips. The nexmon firmware patching framework [21] focuses mainly on the modification of binary ARM firmwares by writing patches as C code. Yet, it also incorporates the b43 (dis-)assembler used by the OpenFWWF project for modifying the ucode. Using nexmon, we can add new ioctl handlers for controlling the firmware by the host and for preprocessing CSI frames coming from the D11 core before forwarding them to the host. With OpenFWWF we can better understand the code-flow in the ucode, spot existing functionalities and add new assembly instructions for customizing the behaviour of the chip directly within the nexmon framework.

5 RESULTS

We report here results that we collected on some of the platforms supported by our tool. On all devices, we configure channels and capture filters with the `nexutil` command.

5.1 CSI on Asus RT-AC86U

Asus RT-AC86U is a powerful router powered by a BRCM4366 chipset (802.11ac, Wave2). It can receive up to $NSS = 4$ transmitted spatial streams at each of the $NRX = 4$ antennas (4x4 configuration). Each tone of the CSI is extracted from a 4 byte data structure where I and Q values are floating-point numbers, with 12-bit mantissa and 6-bit exponent plus sign.

In Figure 7, we illustrate CSI data collected in two different configurations. In the first (top) we inject a frame from another Asus router with a single spatial stream ($NSS = 1$). In this case, the router transmits the same data Space Time Block Coded (STBC) over four antennas. At the receiver, this appears as a sort of multipath effect, and, in fact, we can see the ripples in the CSI amplitudes at antenna 0 and 1 of the CSI receiver (we omit the other two receive antennas). In the second case (bottom) we inject with four spatial streams ($NSS = 4$). Because of the different preamble, now the receiver can isolate the four tx streams at each of the four receive antennas. As we see for receive antenna 0 and 1, there is no ripple in the CSI from transmit antenna 0 (we omit the remaining 14 CSIs).

Finally, in Table 2, we show the maximum number of 80 MHz CSI that can be collected per second from the Asus router. We extrapolated this value by experimentally measuring the time it takes for extracting the actual data, to which we add the theoretical time for transmitting the triggering frame (UDP without payload encoded with MCS9). We neglect the channel access time and we also omit the reply frame and RTS/CTS exchange. We then report in the table the inverse of the total time. We first note that the table is not symmetrical: though counter-intuitive, it is due to the time for transmitting the frame that increases with higher NSS. We also note that the performance indicator for the three cases in the bottom-right corner decreases faster than expected. This is due to a delay that must be introduced in the ucode to avoid problems with excessive FIFO saturation.

Table 2: Number of extracted $80MHz$ CSI per second .

NRX \ NSS	1	2	3	4
1	8223	5112	3668	2927
2	5219	2988	2080	1616
3	3837	2116	1454	**427**
4	3034	1638	**427**	**168**

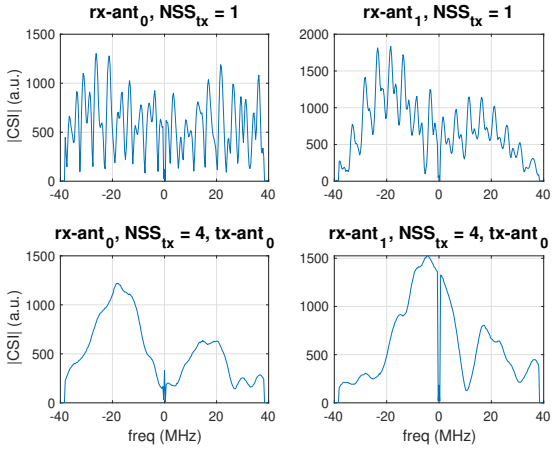


Figure 7: CSI for different configurations: top, single stream received at two antennas; bottom, 4x4 setup.

5.2 CSI Demo App: Live CSI in Your Pocket

To showcase the capabilities of our CSI extraction platform, we implemented a user-friendly smartphone application. It is subdivided into *Receiver*, *Transmit*, and *Settings* screens. On startup the custom firmware required to capture CSI is installed and reset during exit, thus simplifying the interaction with the system. Note that installing the patched firmware requires root permissions.

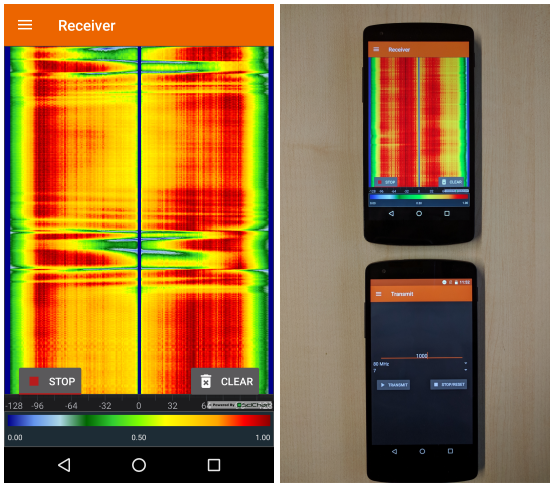
The Settings screen offers options to toggle monitor mode and tune the Wi-Fi chip to a specific channel as well as choosing bandwidth of 20, 40, or 80 MHz, utilizing `nexutil` to communicate with the chip. The transmit screen has inputs for the number of frames to send, transmission bandwidth, and Modulation Coding Scheme (MCS), as well as buttons for controlling the transmission, again using `nexutil` to pass events and information to the Wi-Fi chip.

The Receiver screen shows the live CSI captured by our tool as a waterfall diagram. A dynamic heatmap plot shows the magnitude of the extracted CSI values. In Figure 8a, we show an 80 MHz capture of 802.11ac frames. X-axis shows the 256 OFDM subcarriers, y-axis represents the time, with some 1,000 CSI frames displayed at once. The top and middle of the plot show fluctuations in CSI due to moving the sending phone. The remainder of the plot shows nearly static CSI due to stationary sender and receiver.

The app uses a background thread listening on a UDP socket to receive the CSI datagrams generated by the Wi-Fi chip. On each reception, the app updates the heatmap creating the waterfall effect using the SciChart [22] library. Figure 8b shows a setup with two Nexus 5 devices.

6 SUMMARY & CONCLUSIONS

The knowledge of CSI is indispensable to further optimize and innovate wireless communication technology. Despite



(a) Receiver screen (b) App on two Nexus 5

Figure 8: Screenshots of the waterfall diagram representing CSI measurements over time on the *Receiver* screen and the application in action on two Nexus 5 smartphones, receiver (top) and transmitter (bottom).

the fact that CSI-enhanced mechanisms have been recently proposed, they are typically validated with special purpose SDR-platforms and outside of recent regular standards such as 802.11ac. In this work, we address the lack of a flexible and powerful CSI extraction tool for modern Wi-Fi chipsets. We design and implement a platform facilitating CSI extraction on multiple platforms covering a large scenario space. Our Platform supports sub 50 EUR Raspberry Pis, which facilitate cheaply capturing CSI at scale, Android-based smartphones such as Nexus 5 and Nexus 6P that allow CSI extraction in highly mobile scenarios and state-of-the-art Wi-Fi access points supporting up to four spatial streams and up to 80MHz of spectral bandwidth per stream. Additionally, we provide a user-friendly smartphone application to demonstrate the CSI extraction capabilities. We release the aforementioned tools to the community: interested readers can download them from https://nexmon.org/csi.

7 ACKNOWLEDGMENTS

This work has been performed in the context of the DFG CRC 1053 MAKI and the LOEWE centre emergenCITY.

REFERENCES

[1] Yongsen Ma, Gang Zhou, and Shuangquan Wang. Wifi sensing with channel state information: A survey. *ACM Comput. Surv.*, 52(3):46:1–46:36, June 2019.

[2] Zheng Yang, Zimu Zhou, and Yunhao Liu. From rssi to csi: Indoor localization via channel response. *ACM Comput. Surv.*, 46(2):25:1–25:32, December 2013.

[3] Ieee standard for information technology– telecommunications and information exchange between systems– local and metropolitan area networks– specific requirements. part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–425, Dec 2016.

[4] M. Schulz, J. Link, F. Gringoli, and M. Hollick. Shadow wi-fi: Teaching smartphones to transmit raw signals and to extract csi to implement practical covert channels over wi-fi. In *Proceedings of Mobisys*, Jun. 2018.

[5] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. Nexmon: Build your own wi-fi testbeds with low-level mac and phy-access using firmware patches on off-the-shelf mobile devices. In *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization*, WiNTECH '17, pages 59–66, New York, NY, USA, 2017. ACM.

[6] J. Xiong, K. Sundaresan, and K. Jamieson. Tonetrack: Leveraging frequency-agile radios for time-based indoor wireless localization. In *Proceedings of Mobicom*, 2015.

[7] Ouyang Zhang and Kannan Srinivasan. Mudra: User-friendly fine-grained gesture recognition using wifi signals. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, pages 83–96, New York, NY, USA, 2016. ACM.

[8] Yuexing Peng, Peng Wang, Wei Xiang, and Yonghui Li. Secret key generation based on estimated channel state information for tdd-ofdm systems over fading channels. *IEEE Transactions on Wireless Communications*, 16(8):5176–5186, 2017.

[9] Xu Zhang and Edward Knightly. Csisnoop : Inferring channel state information in multi-user mimo wlans. *IEEE/ACM Transactions on Networking*, 27(1):231–244, 2018.

[10] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Tool release: Gathering 802.11n traces with channel state information. *SIGCOMM Comput. Commun. Rev.*, 41(1):53–53, January 2011.

[11] Y. Xie, Z. Li, and M. Li. Precise power delay profiling with commodity wifi. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, page 53–64, New York, NY, USA, 2015. ACM.

[12] QCA Atheros. ath10k spectral scan.

[13] F. Ricciato, S. Sciancalepore, F. Gringoli, N. Facchi, and G. Boggia. Position and velocity estimation of a non-cooperative source from asynchronous packet arrival time measurements. *IEEE Transactions on Mobile Computing*, 17(9):2166–2179, Sep. 2018.

[14] Quantenna. Quantenna and aerial deliver advanced motion detection utilizing existing wi-fi infrastructures, 2018.

[15] Rice University. Warp: Wireless open access research platform, 2019.

[16] Ettus Research. The universal software radio peripheral usrp software defined radio device, 2019.

[17] X. Li and K. Pahlavan. Super-resolution toa estimation with diversity for indoor geolocation. *IEEE Trans. on Wireless Comms.*, 3(1), 2004.

[18] M. Schulz, D. Wegemer, and M. Hollick. The nexmon firmware analysis and modification framework: Empowering researchers to enhance wi-fi devices. *Elsevier Computer Communications*, 129:269–285, Sept. 18.

[19] Open Firmware for WiFi Networks. http://www.ing.unibs.it/openfwwf/.

[20] F. Gringoli, P. Serrano, I. Ucar, N. Facchi, and A. Azcorra. Experimental qoe evaluation of multicast video delivery over ieee 802.11aa wlans. *IEEE Trans. on Mobile Computing*, 2018.

[21] NexMon Project. https://github.com/seemoo-lab/nexmon/.

[22] SciChart: high performance realtime charts. https://www.scichart.com/.