Multichannel-Sniffing-System for Real-World Analysing of Wi-Fi-Packets

Kristof Friess
Applied Computer Science
University of Applied Sciences Erfurt
Erfurt, Germany
kristof.friess@fh-erfurt.de

***Abstract*—Wireless technologies like Wi-Fi send their data using multiple channels. To analyze an environment and all Wi-Fi packets inside, a sniffing system is needed, which can sniff on all used channels of the wireless technology at the same time. This allows catching most packets on each channel. In this paper, a way to build up a multi-channel-sniffing-system (MCSS) is described. The test system uses several single board computers (SBC) with an external Wi-Fi adapter (USB), 19 SBCs are sniffing nodes (SFN) and one SBC as sending node (SN). The sniffing SBCs are placed in a cycle around the sender so that every node has the same chance to receive the simulated packets from the SN. For the control of all 20 SBCs, a self-developed software is used, which connects from the host to the clients and is used for configuring the experiments. The configuration is sent to each client and will initiate their start, so that their times are also synchronized, for this all clients are synchronised using a time server.**

***Index Terms*—sniffing, multichannel, wifi, sbc, node.js**

I. Motivation

There is growing market for worn computer systems like smartphones, smart rings and smartwatches, in short wearables, which are changing the interaction between human and computer. It also changes the interaction between human, computer and the environment. There are a lot of use cases where a human can use the computer as an assistant in filtering data, storing context information or analyzing sports activities. Next to them, there are a lot of use cases coming up, where not the device itself helps the human to become smarter, but where the environment, based on the knowledge about the human, acts smartly - smart environments. They can be used to make the environment also smart as the computer system is. Using the connectivity technologies of the wearables, in most cases Wi-Fi or Bluetooth, a recognition system can be deployed. However, the application field of human recognition and re-recognition is already here. There are many situations where video surveillance detect movements of humans and an algorithm identifies a security thread, for examples in industry, banks, private buildings, and country borders. Also at huge events, video surveillance is used for monitoring individuals and groups to detect panic situations, timely. But all those systems bring no additional benefit, apart from the security, to the human, who is recognized. Also, the detection using video analysis and live images suffers from people that are overlapping and very small, because of the distance of cameras. When the overlap increases (more people), or people get smaller (because of the distance) in the image, then negative-positive results will be much higher [1], [2]. To re-recognize an object, in this case, a human, features of the human (for example clothing) are needed. The required characteristics used for re-recognition can only be found at the human himself [2], [3]. So if the human changes his or her clothes, he can no longer be identified as the same human being as before.

Because of the challenges and the issue that the user can not get any benefit from the recognition, a new way has to be found. This way can be the detection using wearable computers. In view of the current types of wearables, most of them have a wireless connectivity for data input and output. While researching the wearable devices (more then 75 wearables like smartphones, smartwatches, sport-tracker, head-up-displays were checked) currently on the market, it was detected that most systems use Wi-Fi and Bluetooth/Bluetooth LE. So if it is possible using the wireless data output characteristics of wearable devices to identify and recognize the same human being again later, a smart environment could act based on this information.

To use the potential of recognition based on Wi-Fi, a system should be build, which can be used to detect worn devices by receiving their Wi-Fi packets, as many as possible. In the normal case, a simple Wi-Fi sniffing-system will do this job. Just one Wi-Fi module is needed, which will sniff all the channels using channel-hopping. Using a hopping frequency of 250ms for 13 channels means that a specified channel is not listened for three seconds. Therefor, it is better to setup an system which will be sniffing on each channel at the same time. But this is really expensive, because of the amount of Wi-Fi modules. Especially for the use case smart environment, more then one sniffing system is needed. So the question is, how should the system be designed to reduce costs? Is it enough to use a single Wi-Fi module jumping around the channels, or should several Wi-Fi modules be used, one for each channel? How many packets are dropped with a single module sniffing on all channels (drop rate)? To answer these question, a comparison between channel hopping sniffing and static channel sniffing of a defined technology is needed. This is what this paper is about, it describes a system, which can be used to analyze channel traffic and sniff using different

configurations. So that an experiment, for example, 2.4GHz, can be set up to sniff on each communication channel with a single Wi-Fi module and also on all channels using a Wi-Fi module which is jumping between the channels (channel hopping). So a detection of the drop rate in a real environment is possible.

II. Introduction

For the analysis of user behavior with portable computers (wearables), it was determined that the behavior should be based on the probe requests sent by the devices to discover access points in the environment. Normally, the user wants comfort, therefore they leave Wi-Fi on and tell the device to auto-connect to known access points – so probe requests are important. For this investigation, it is necessary to capture as many packets as possible while listening to the wireless network such as 2.4GHz and 5GHz Wi-Fi. In wireless communication, the packets can be sent in different frequencies (channels). Therefore, it is necessary to capture all packets at the same time, to listen on all channels and to evaluate the data traffic.

By definition, in the 2.4 GHz range, there are up to 14 channels (table I) and in the 5 GHz there are up to 23 channels (table II) possible as a transfer way of data [4, p. 317ff] [5] [6] [7, p. 2239,p. 2403] [8, p. 269]. Despite the high number of possible channels for the wireless network, not all channels are always released for use, for individual countries have different regulations. In Europe, an use of 13 channels in the 2.4 GHz and 19 channels in the 5 GHz is defined. However, the 19 channels in the 5 GHz are not fully usable, 3 channels (120, 124 and 128) are reserved for the weather radar data, so if these channels are in use, normal communications have to release them [5]. The consequence of this is that for this sniffing experiment, which is conducted in Germany, only 13 channels in the 2.4 GHz range and 16 channels in the 5 GHz range are available and must be considered.

TABLE I
2.4 GHz frequency range for Europe, USA and Japan [4, p. 317ff] [5] [6] [7, p. 2239,p. 2403] [8, p. 269]

Channel	Frequency	Europa	USA	Japan
1	2412 MHz	x	x	x
2	2417 MHz	x	x	x
3	2422 MHz	x	x	x
4	2427 MHz	x	x	x
5	2432 MHz	x	x	x
6	2437 MHz	x	x	x
7	2442 MHz	x	x	x
8	2447 MHz	x	x	x
9	2452 MHz	x	x	x
10	2457 MHz	x	x	x
11	2462 MHz	x	x	x
12	2467 MHz	x	-	x
13	2472 MHz	x	-	x
14	2484 MHz	-	-	x (11b)

Building a sniffer that can capture all the data at the same time is a very costly affair, because of the number of Wi-Fi modules for each channel. Especially if a high number of such

TABLE II
5 GHz frequency range for Europe, USA and Japan [4, p. 317ff] [5] [6] [7, p. 2239,p. 2403] [8, p. 269]

Channel	Frequency	USA	EU	Japan
36	5,180 GHz	x	x	x
40	5,200 GHz	x	x	x
44	5,220 GHz	x	x	x
48	5,240 GHz	x	x	x
52	5,260 GHz	x	x	-
56	5,280 GHz	x	x	-
60	5,300 GHz	x	x	-
64	5,320 GHz	x	x	-
100	5,500 GHz	-	x	-
104	5,520 GHz	-	x	-
108	5,540 GHz	-	x	-
112	5,560 GHz	-	x	-
116	5,580 GHz	-	x	-
120	5,600 GHz	-	(x)	-
124	5,620 GHz	-	(x)	-
128	5,600 GHz	-	(x)	-
132	5,660 GHz	-	x	-
136	5,680 GHz	-	x	-
140	5,700 GHz	-	x	-
147	5,735 GHz	x	-	-
151	5,755 GHz	x	-	-
155	5,775 GHz	x	-	-
167	5,835 GHz	x	-	-

a sniffer is needed, for example, to cover a larger territory for the motivation scenario to create a smart environment. For this reason, the goal has been defined to find a way that allows listening on all channels with less hardware, that means to decrease the number of Wi-Fi modules, decrease the costs. For this, the use of channel hopping is apparent, because with only one Wi-Fi receiver several channels can be sniffed. But the number of packets lost, because of channel switching, is not tracked. Therefore, it is important to identify how many Wi-Fi receivers with channel hopping are needed, instead of sniffing each single channel. For this review, a system has to be developed which enables simultaneous listening on all 13 / 16 channels in the 2.4 / 5 GHz frequency range, and additionally listening using channel hopping. Only then, an actual comparison can be made. Because with the static listening channels, for example, the 13 channels in the 2.4 GHz, it is possible to identify the noise floor / the number of collisions in a real environment and then compare it with the error rate of the channel hopping.

III. Hardware

In the preliminary investigation for the setup, it was identified that later on at least 13 channels in the 2.4 GHz and on 16 channels in the 5 GHz (according to European / German law) must be sniffed. This corresponds up to 29 Wi-Fi adapters for simultaneous listening on both frequency bands. Due to the cost of hardware, it was initially defined that only one frequency band should be analyzed at a time.

In addition to listening on all channels of a frequency band, a comparison with other methods (e.g. channel hopping) should be made, this will require more Wi-Fi adapters. That is, if a comparison to the channel hopping (at least 1 system hops

between the channels) should take place, at least one more Wi-Fi antenna is needed. In addition, another two antennas are needed to test if there is an increase in the number of packets received if more than one antenna hops on the channels. It was defined that up to three antennas should be used for channel hopping. So finally there are 19 Wi-Fi adapters. Another adapter serves as a transmitter and sends packets, so it is obvious how many packets should have been received. Thus in addition to the traffic in a real environment, also a statement on the loss rate of packets on a channel can be made.

So, there is the following basic requirement for the system:

- 19x Sniffing Modules for 2.4 GHz and 5 GHz
- 1x Transmitter Module for 2.4 GHz and 5 GHz
- Wi-Fi Monitor Mode
- Wi-Fi Antenna should be changeable and omni-directional
- Cost efficiency

With all the predefinitions, the basics are done. Now, a system should be build which can handle all. So the first question is, is there any system which can handle the 20 Wi-Fi adapter. On considering the systems available on the market, it has been noticed, that a similar construction like a MIMO - Multi Input Multi Output System as in antenna experiments for more stable and/or stronger networks is needed [9] [10] [11].

In further research, a required setup for 20 antennas could be found. In the paper by Pradeep Reddy, Hari Sharma and Dominic Paulraj on the system architecture of a multi-channel Wi-Fi sniffer [12], an interesting setup for the solution is described. The authors rely on a system of several single-board-computers (SBC) and a server host for analyzing the data. This setup served as a template for the construction of this sniffing system.

The basic idea is that the currently planned sniffing system will not only be used for Wi-Fi, but also for other wireless connectivities such as Bluetooth. Therefore, the single-board computer (SBC) should be modular and have USB ports, so that simply via USB adapter the respective wireless module can be connected. In addition, it was necessary that the SBC has an Ethernet network card, sufficient memory capacity, and computing power. Since it was a cheap solution, the decision was made on a Raspberry Pi 3 Model B (specification table III). This decision was reinforced by the successful use of a sniffing experiment 'SenseFlow ' [13].

The Raspberry Pi 3 comes from scratch with a wireless module. Unfortunately, this is a fairly cheap chipset, which can not be put into monitor mode. Therefore, the integrated Wi-Fi module is not useful for the sniffing experiments. Also, this module is internally on the SBC and is thus not free from interference by the board. Thus, it was decided to use a Wi-Fi adapter for the experiments. The Wi-Fi adapter should be connected to the Raspberry Pi using USB. For the adapter, the antenna should be external, so that if necessary, other antennas could be attached to the system. For the sniffing itself, it is important that the drivers work under Linux and the Wi-Fi module can be put into monitor mode. With the good experience with TP-Link modules and the set requirements, the adapter "Archer T2UH AC60 High Gain Dual Band USB WLAN Adapter" (specification table IV) was chosen.

TABLE III
HARDWARE DETAILS RASPBERRY PI 3 MODEL B

Chip	Broadcom BCM2387 Quad-Core ARM Cortex-A53 64 Bit
CPU Frequency	1200 MHz
Memory (RAM)	1024 MB LPDDR2
Disk-Storage (microSD-Slot)	SD-Card 16 GB (max. 64) GB
Ethernet	10/100 MBit
USB	4x 2.0
HDMI	rev 1.3 & 1.4 Composite RCA
Wi-Fi	Ja
Bluetooth	4.1 (Bluetooth Classic and LE)
Power	Micro-USB 5V 2.5A

TABLE IV
HARDWARE DETAILS WI-FI ADAPTER ARCHER T2UH

Wi-Fi standards	IEEE802.11a/b/g/n/ac
Interface	USB 2.0
Antenna Type	omni-directional, removable, RP-SMA
Antenna Gain	3dBi
Chipset	MediaTek MT7610U
Vendor ID	0x148f
Product ID	0x761a

After acquiring the module, it was found that the modules can monitor without problems, but they could not transmit packets to 2.4 GHz or 5 GHz. The driver is unfortunately not suitable for this. For this reason, another Wi-Fi adapter has been chosen to send packets at 2.4 GHz and 5 GHz. In 2.4 GHz it could be identified that the adapter (was available without buying a new one) "TL-WN722N 150mbit/s High-Gain WLAN-USB-Adapter" (specification table V) by TP-Link reliably sends and for 5 GHz, the "ANEWKODI WLAN Stick 600Mbit/s" (specification table VI) could be identified as a functioning adapter.

TABLE V
HARDWARE DETAILS WI-FI ADAPTER TL-WN722N

Wi-Fi standards	IEEE802.11b/g/n
Interface	USB 2.0
Antenna Type	omni-directional, removable
Antenna Gain	4dBi
Chipset	Atheros AR9002U
Vendor ID	0x0cf3
Product ID	0x9271

Since the single board computers with the Wi-Fi adapters are to serve only as receiver/transmitter, a host system is needed, which is used to control the SBCs and the configurations. This host can be any machine with sufficient processing power. In this setup, a Mac Mini is used, because of the availability and the power (specification table VII). To connect the 20 Raspberry Pis and the host system, an ethernet network with a 24-port 10/100/1000 LANCOM switch is used.

TABLE VI
HARDWARE DETAILS WI-FI ADAPTER ANEWKODI 600MBIT/S

Wi-Fi standards	IEEE802.11a/b/g/n/ac
Interface	USB 2.0
Antenna Type	omni-directional, removable
Antenna Gain	5dBi
Chipset	Realtek RTL8811AU
Vendor ID	0x0bda
Product ID	0xa811

TABLE VII
HARDWARE DETAILS HOST-SYSTEM MAC MINI

Model	Mac mini (Mid 2010, Macmini4,1)
Betriebssystem	maxOS Sierra 10.12.5
Prozesser	Intel Core 2 Duo 2.4 GHz
Arbeitsspeicher	4096 MB 1067 MHz DDR3
GPU	NVIDIDA GeForce 320M 256 MB
Festplatte	TOSHIBA 320 GB 5400 RPM
LAN/Netzwerk	0xa811

For best results in sniffing, it was considered which installation of the hardware is the best. In the first step, all receiver SBCs were placed side by side and the transmitter SBC in front of them. However, it quickly became apparent that this is not the optimal setup. Because the distances (illustration 1) of the individual receivers from the transmitter are clearly different, this can influence the results. For this reason, it was decided to form a ring structure of the system. All SBCs were at the same angle and same distance from the transmitter, so all sniffing SBCs have the same precondition to receive the packets from the SN (Figure 2). Depending on how the Raspberry Pi and the Wi-Fi antenna are positioned, a distance from the transmitter antenna from 210mm to 320mm is possible.

IV. SOFTWARE

As the hardware setup shows, there are 20 SBCs and one host system which need to be installed and configured. For the host system (Mac Mini), the delivered operation system OSX 10.12.5 is used. The Raspberry Pis came without any operation system (OS) on it. So, all 20 SD-Cards must be burned

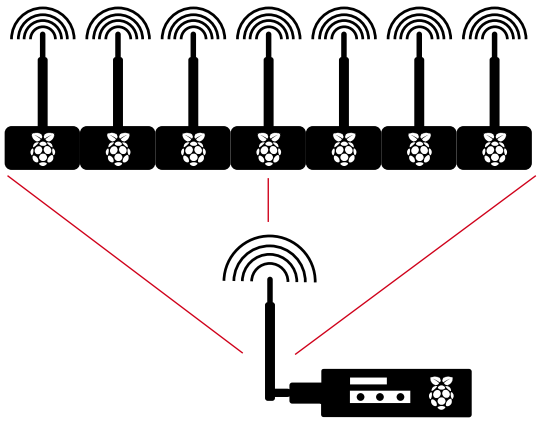

Fig. 1. SBCs setup side by side

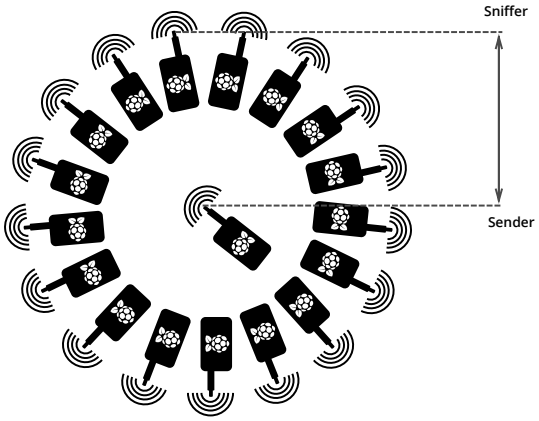

Fig. 2. SBCs setup as ring

with an OS, here Raspbian [14] is used, a distribution of Debian Jessi. For this, the OS image from raspberrypi.org was downloaded and then copied to the plugged in SD-Card using the "dd" command line tool of OSX. As all Raspberry Pis need the same installation, everything was initially installed and configured on one SD-Card. Subsequently, a copy of the first card was transferred to the other SBCs.

Next to the OS, some required packages were installed. The *Wi-Fi adapter driver* to get the adapters running, *tcpdump* used for Wi-Fi sniffing, *mkd3 v6* used for sending Wi-Fi packets, *git* using for updating the control software and *node.js* for the control software.

Wi-Fi adapter driver - Getting Wi-Fi adapters up and running on linux is not always easy. The adapter TL-WN722N could easily be installed via the package manager *firmware-atheros* of Debian and successfully put into operation. By contrast, the ANEWKODI and Archer T2UH were more challenging. Here, the correct driver had to be identified, built and tested.

tcpdump - The tool tcpdump allows eavesdropping with a Wi-Fi antenna on the network. The tool offers several options from selective output to saving data in a PCAP file [15]. This is used in the experiments, all captured packets are stored in a PCAP file and this file is sent to the host system.

mkd3 v6 - For simulating/sending Wi-Fi packets, the software mkd3 from Kali Tools [16] was used. We resorted to a community-enhanced version of the v6, which already includes implementations for 5 GHz. However, this implementation was not enough and even extensions were made for correct channel selection and outputs. The output of this tool is written into a text file, which is also sent to the host system for analyzing, later.

git - The Version Control System (VCS) git [17] is a tool for tracking and managing changes in the implementation of the software. But in this software architecture, git us used as an simple software update sharing system for the Raspberry Pi clients. They are directly connected to the repository of the client software, so they will load the current version at each reboot. Thereby, an update can be done by simply "restarting"

the SBC.

node.js - Node.js is a JavaScript runtime implementation on the current Chrome's V8 JavaScript engine [18] and it is executes server-side JavaScript code [19]. In this project, it is the base language for the server and client software implementation.

As described, Node.js has been used for the control software for all systems. This is necessary because of the 20 SBCs, which have to start and stop nearly at the same time. The control software is divided into server and client.

A. Server side software

On the Mac mini, our host system and thus the interface between configuration and execution (host is located in the middle, see Figure 3), the server software is programmed in Node.js. This software provides a web server with a web page for management by the user and an io-WebSocket server for fast data exchange between browser/clients and server. This website can be used with any currently available web browser (here Firefox is used). On the website itself, there are three areas - clients, scheduled tasks, past tasks. In the clients area, all clients known to the server (connected or not) are visible. In addition to the status via the connection status, the clients have the information about the UUID, the IP address and the current configuration for the next task. With the configuration, the next task can be planned and set. This means that it is possible to define the respective settings for each client, after which a new task (job) can be created. This task is then saved with all clients and their settings with a start and stop time. Using the WebSocket, which exists between client and server, the clients now receive the information for the next job. From this point on, the client is responsible for starting the job as soon as the start time is reached. As explained, the clients are connected to the server via the network, which establishes a socket connection using TCP. The socket connectivity is provided by socket-io [20]. Using the socket connection, information (states, IP, configurations, ...) can be quickly exchanged between the clients and the server. However, the start tasks command is not sent using the exchange socket, the clients and the server organize themselves when starting/stopping tasks beyond the start time (date and time). This ensures that the execution of the tasks starts simultaneously on all systems without any relevant time delay. The prerequisite for this is a Network Time Protocol (NTP) server for providing the time to all connected systems.

B. Client side software

Every single Raspberry Pi (SBC) runs the client software programmed in Node.js. The software starts up with the operating system and automatically searches in the network for the host server using Apple Bonjour [21]. If the client detects the host server, it will connect directly to it using the WebSocket interface. The server then transmits the current jobs with their configurations. If a user changes the tasks/jobs in the web interface or creates additional ones, the server sends the connected clients an update directly, but each client only receives its settings. It does not know the configuration of the other clients. At runtime, each client checks every full minute to see if there are scheduled tasks that need to be started. If this is the case, the configuration is loaded and the respective execution is carried out by means of bash scripts. When a job completes successfully, the client submits the recorded data to the host server and removes the job from the jobs list. Now the client remains in passive mode until a task needs to be started again.

V. Practical usage

The set up outlined above was used in several measuring experiments (2.4GHz and 5GHz) during 2017. It was used to identify the best channel hopping frequency, sniffing environments to check how many traffic there is and how many collisions and also to check how many channel hopping Wi-Fi modules are needed to get nearly the same results like sniffing on all channels at the same time.

The last explained experiment and the result is shown here, as an example. It focused on the detection of the drop rate of packets using channel hopping sniffers vs. sniffing on each static channel in the frequency band of 2.4 GHz. The setup was quite simple, there are 13 SBCs (European standard) configured for each channel from 1 to 13 and 4 channel hopping SBCs. The channel hopping modules jump across the channels from 1 to 13 by jumping to the next non-overlapping one – 1,7,13,2,8,3,9,4,10,5,11,6,12. The hopping sniffers stay 250ms on each channel before jumping to the next one. To get a better result instead of still sniffing packets in the environment, one SBC (the one who is in the mid of the setup in Figure 2) is used for sending packets, around 2 - 4 packages per second, randomized to all channels

The experiment itself runs five times with a duration of one hour. But for the analysis, only four runs can be used. Because in the last run, some of the SBCs unexpectedly crashed, currently without any idea why. The four runs were analyzed separately by the following attributes *captured beacons created by the sender*, *captured beacons created by the sender for the own channel*, *missed beacons created by the sender for the own channel*. Then they are combined to an average and this is used to calculate the accuracy and packet drop rate. In table VIII, static.1 to static.13, represent the static sniffing results. The static.all is a combination of all statics without duplicates, to get an idea how high the drop rate in the environment is. hopping.1 to hopping.4 is a combination of one to four channel hopping modules without duplicates. So, it is possible to see, how many channel hopping modules are needed to get nearly the same result as the static sniffing one.

As table VIII shows, the drop rate of the static sniffing Wi-Fi adapters is between 1.70309% and 0.16892%, the combination all static modules has a drop rate of 0.02939%. The drop rate of the hopping modules in combination moves from 1.02380% to 0.02519%. This means that a combination of four channel hopping sniffers will capture a bit better than sniffing on all channels. So, a sniffer should have 13 modules sniffing on 13

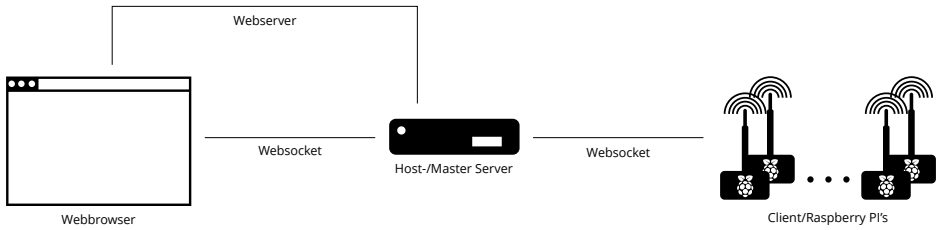

Fig. 3. Communication between User(Browser), Server and Client

TABLE VIII
RESULT OF THE HOPPING VS. SNIFFING EXPERIMENT FOCUSED ON THE SELF GENERATED BEACONS

Name	Captured Beacons	Missed Beacons	Drop Rate
static.1	1049.75	2.25	0.21434%
static.2	917.5	2.25	0.24523%
static.3	888	1.5	0.16892%
static.4	912.5	5.75	0.63014%
static.5	876.25	8.5	0.97004%
static.6	874.5	8.75	1.00057%
static.7	889.5	1.5	0.16863%
static.8	862.25	12.25	1.42070%
static.9	969.25	3	0.30952%
static.10	926.5	2.75	0.29682%
static.11	894.5	1.5	0.16769%
static.12	892	12	1.34529%
static.13	880.75	15	1.70309%
static.all	11910.25	3.5	0.02939%
hopping.1	11910.25	121.94	1.02380%
hopping.2	11910.25	19.25	0.16163%
hopping.3	11910.25	6.5	0.05457%
hopping.4	11910.25	3	0.02519%

channels or just 4 modules hopping between the channels to get the same result.

VI. PERSPECTIVE

The goal, to build a system which can be used to sniff on all channels of a Wi-Fi system next to up to three channel hopping sniffers got achieved using the set up outlined above. The system is implemented with 20 SBCs. 19 of them are used for sniffing and one is used for sending Wi-Fi packets. All nodes are controlled with a host system and a self developed software. Several experiments are executed to get more knowledge about the system, doing some fine tuning and getting better results. One of the next steps is a stable analyzing of both Wi-Fi bands over a longer time. Currently, we observe that the SBCs crash after 8-24 hours, therefore some experiments delivered no result. At the moment, it is not clear why this happened, perhaps the Raspberry Pis should be replaced with other SBCs like Tinker Board, Arduino or Orange Pi. If the system is stable, the setup is used to get started with the idea of a smart environment by sniffing the wearables of the people inside.

REFERENCES

[1] E. Corvee, S. Bak, and F. Bremond, "People detection and re-identification for multi surveillance cameras," in *VISAPP - International Conference on Computer Vision Theory and Applications -2012*, Rome, Italy, Feb 2012.

[2] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[3] F. Jahan, M. K. Islam, and J.-H. Baek, "Person Detection, Re-identification and Tracking Using Spatio-Color-based Model for Non-Overlapping Multi-Camera Surveillance Systems," *Smart Computing Review*, vol. 2, no. 1, p. 42, Feb 2012.

[4] M. Sauter, *Grundkurs Mobile Kommunikationssysteme: LTE-Advanced, UMTS, HSPA, GSM, GPRS, Wireless LAN und Bluetooth*, 6, Ed. Wiesbaden: Springer Vieweg, 2015.

[5] P. Schnabel. WLAN-Frequenzen und -Kanäle. [Online]. Available: http://www.elektronik-kompendium.de/sites/net/1712061.htm

[6] C. J. Hou, "IEEE Std 802.11ac Deployment in Japan," March 2013. [Online]. Available: https://wifiamateur.blogspot.de/2013/04/80211ac-device-deployment-in-japan.html

[7] "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE 3 Park Avenue New York, NY 10016-5997 USA, 2016.

[8] J. Beale, G. Ramirez, and A. Orebaugh, *Jay Beale's Open Source Security Series: Wireshark & Ethereal Network Protocol Analyzer Toolkit*. 800 Hingham Street Rockland, MA 02370: Syngress Publishing, Inc., 2006, no. 9781597490733.

[9] P. Schnabel. MIMO - Multiple Input Multiple Output. [Online]. Available: https://www.elektronik-kompendium.de/sites/net/1004251.htm

[10] brian wang, "A 96-Antenna System for Next Generation of Wireless." [Online]. Available: https://www.nextbigfuture.com/2014/01/a-96-antenna-system-for-next-generation.html

[11] Argos, "Practical Many-Antenna MU-MIMO." [Online]. Available: http://argos.rice.edu

[12] P. Reddy, H. Sharma, and D. Paulraj, "Multi Channel Wi-Fi Sniffer," in *4th International Conference on Wireless Communications, Networking and Mobile Computing*, 2008.

[13] K. Li, C. Yuen, and S. Kanhere, "SenseFlow: An Experimental Study of People Tracking," in *Proceedings of the 6th ACM Workshop on Real World Wireless Sensor Networks*, ser. RealWSN '15. New York, NY, USA: ACM, 2015, pp. 31–34, http://doi.acm.org/10.1145/2820990.2820994. [Online]. Available: http://doi.acm.org/10.1145/2820990.2820994

[14] Raspberry Pi Foundation, "Raspbian," 2017. [Online]. Available: https://www.raspberrypi.org/downloads/raspbian/

[15] "tcpdump(8) - Linux man page." [Online]. Available: https://linux.die.net/man/8/tcpdump

[16] KALITOOLS, "mdk3 Penetration Testing Tools," Februar 2017. [Online]. Available: https://tools.kali.org/wireless-attacks/mdk3

[17] "1.3 Getting Started - Git Basics." [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

[18] "Chrome V8 Introduction." [Online]. Available: https://developers.google.com/v8/

[19] Node.js. [Online]. Available: https://nodejs.org/en/

[20] "socket.io." [Online]. Available: https://socket.io

[21] Apple, "Bonjour for Developers." [Online]. Available: https://developer.apple.com/bonjour/