**High-Level Design Document for Wifi-Sniffer**

**Revision 0.2**

**9/22/2019**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Contents** | **Author(s)** |
| 0.1 | August 11, 2019 | Initial version | Gal Zur and Matan Grynbaum-Nachmias |
| 0.2 | September 22, 2019 | Corrected version | Gal Zur and Matan Grynbaum-Nachmias |

**Table of Contents**

[Revision History](#_gjdgxs) **2**

[Abbreviation](#_17dp8vu) 5

[Terms](#_26in1rg) 6

[**Overview**](#_35nkun2) **7**

General Diagram of the Wi-Fi Sniffer 7

Controlling the WMP using the HPC 8

Communication Protocol 9

File Transfer 11

Wi-Fi Monitoring Program 12

HPC Class Diagram 13

[**System and Architecture Requirements**](#_2v2yqqxvg96h) **15**

[System Requirements](#_44sinio) 15

[Architectural Requirements](#_1y810tw) 15

[General](#_4i7ojhp) 15

[Constraints](#_2xcytpi) 15

**Open Questions** 16

Alternative Solutions 16

**List of Tables**

[Table ‎0-1: Reference Documents](#_1t3h5sf) 5

[Table ‎1-1: List of Abbreviations](#_3rdcrjn) 5

[Table ‎1-2: List of Terms](#_lnxbz9) 6

Table 2: Control and Execution Communication Protocol 9

**List of Figures**

Basic Wi-Fi Sniffer Diagram 7

Control and Execution Diagram 8

Communication Protocol Sequence Diagram 10

File Transferring in the Wi-Fi Sniffer Diagram 11

HPC Class Diagram 13

**Reference Documents**

|  |  |
| --- | --- |
| **Item #** | **Name** |
| **1** | Wi-Fi sniffer – functional requirements |

**Table ‎0-1: Reference Documents**

## Abbreviation

|  |  |
| --- | --- |
| ***Abbreviation*** | ***Description*** |
| TBD | To Be Decided - where shown item still has not been decided |
| RPi | Raspberry Pi 3 |
| HPC | Host PC |
| Network Segment | The network sniffed by the sniffer, including the AP and the wifi devices. |
| HTCPC | Host TCP Client - The TCP client program in the HPC |
| RTCPS | Raspberry TCP Server - The TCP server program in the RPi |
| WMP | Wifi Monitoring Program - The program that monitors the wifi traffic in the Network Segment |
| Pipe | The pipe channel connecting the RTCPS to the WMP |
| Init file | The file containing all of the initializing settings for the RPi. |
| Filter Data | The file containing the info on how to filter the packets coming in after the initial filter. |
| HUDPC | Host UDP Client - The UDP client program in the HPC |
| HUDPS | Host UDP Server - The UDP server program in the HPC |
| RUDPC | Raspberry UDP Client - The UDP client program in the RPi |
| RUDPS | Raspberry UDP Server - The UDP server program in the RPi |
| WS | Work Station - the Wi-Fi devices that the Wi-Fi Sniffer listens to. |

**Table ‎1-1: List of Abbreviations**

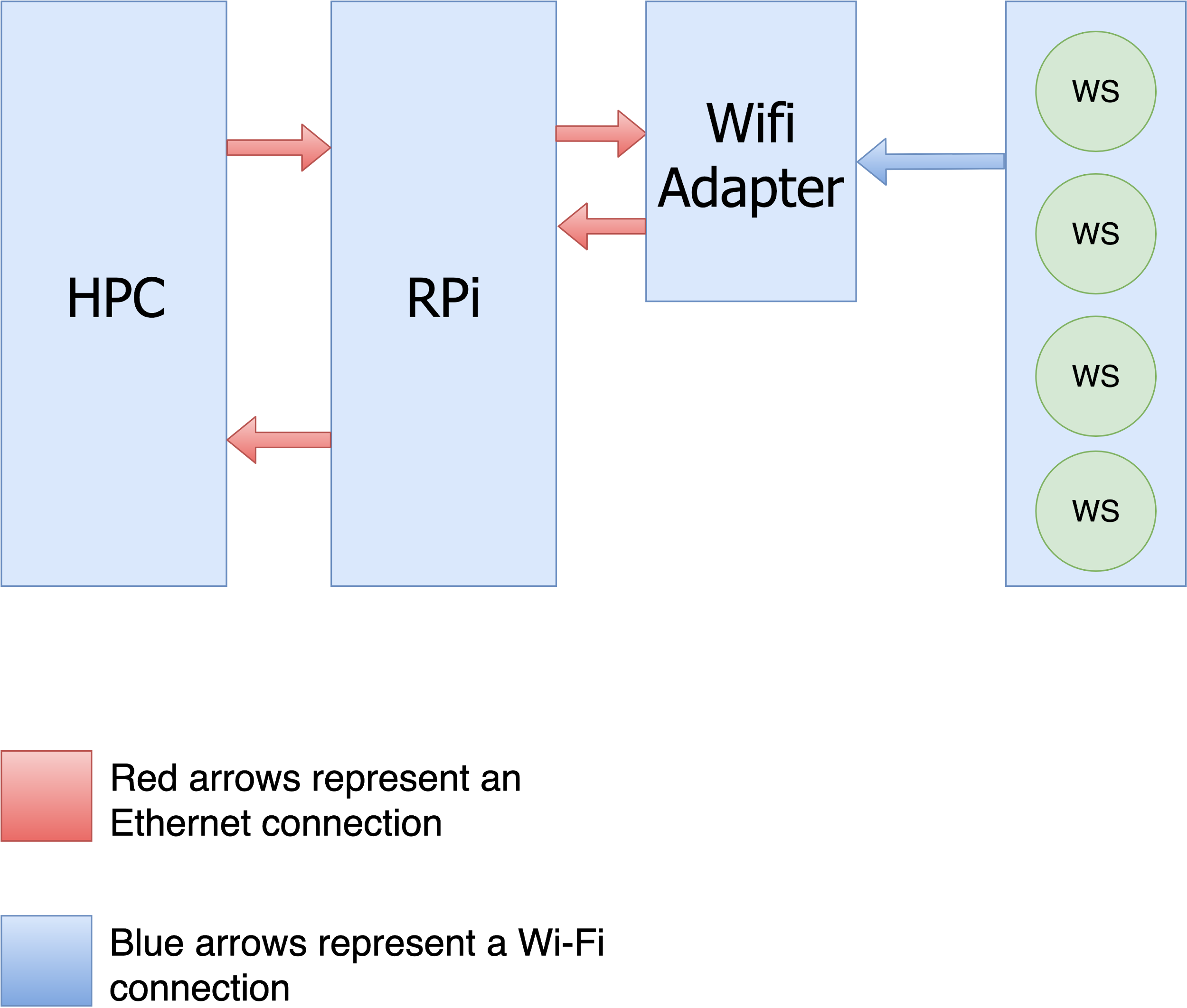
## Terms

|  |  |
| --- | --- |
| ***Term*** | ***Description*** |
| TCP | Transmission Control Protocol – reliable, ordered and error-checked data protocol – meant to transfer critical data |
| UDP | User Datagram Protocol - meant to transfer data where it is not crucial that all of the data will reach the target computer. |
| AP | Access Point – a device which opens a Wi-Fi network and allows devices to connect to it and allow for data transfer to\from the device |
| Wi-Fi device | Endpoint - any device which opens a Wi-Fi network and allows devices to connect to it and allows for data transfer to\from the device |
| Host-PC | the PC that the system will be connected to - a PC with display, keyboard etc. |
| Wireshark | Program - de-facto standard of analysis of network data - open-source |
| Raspberry-Pi | A System-On-Module - basically a low-performance fully-capable PC |
| Socket | Programmatic connection between two SW elements (running on the same or different systems) that allows data transfer |
| JSON | JSON is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types |
| PCAP | Packet capture - an API for capturing packets from the network. Files with a PCAP extension contain the packet data, Wireshark can read this type of files. |
| Dissector | A protocol parser. Wireshark already contains a lot of dissectors, but if needed they can be adjusted and created. |

**Table ‎1-2: List of Terms**

# Overview

* 1. **General Diagram of the Wi-Fi Sniffer**

[](https://www.draw.io/?page-id=TJWygpqnf_4eo9sZrp_q&scale=auto#G12w2H4HaG4K1pZ1bz-a34ynacH8uMRrBC)

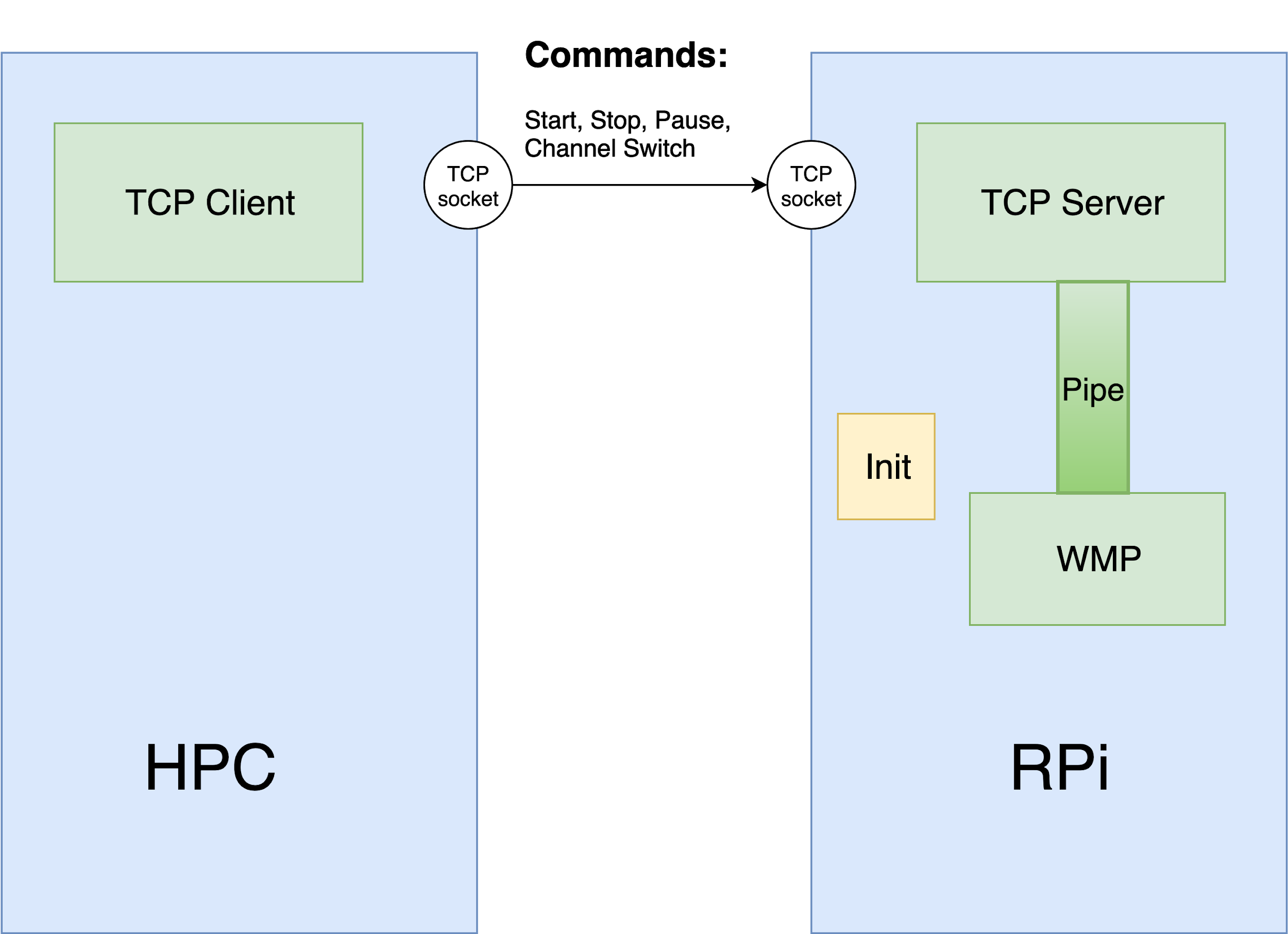
This system performs Wi-Fi Sniffing of Wi-Fi devices by the HPC using the WMP in the RPi. The RPi is connected via a physical connection to the Wi-Fi adapter. Each arrow in the diagram represents information sent from one device to another.

HPC’s and the RPi’s IP is a static one so the system will work flawlessly even if it’s on a different network.

Since the RPi will need to perform multiple actions at once it will use a few processes, one for each server/client and one for the WMP.

Each one of the next couple of diagrams will explain one of the subsystems.

* 1. **Controlling the WMP using HPC**

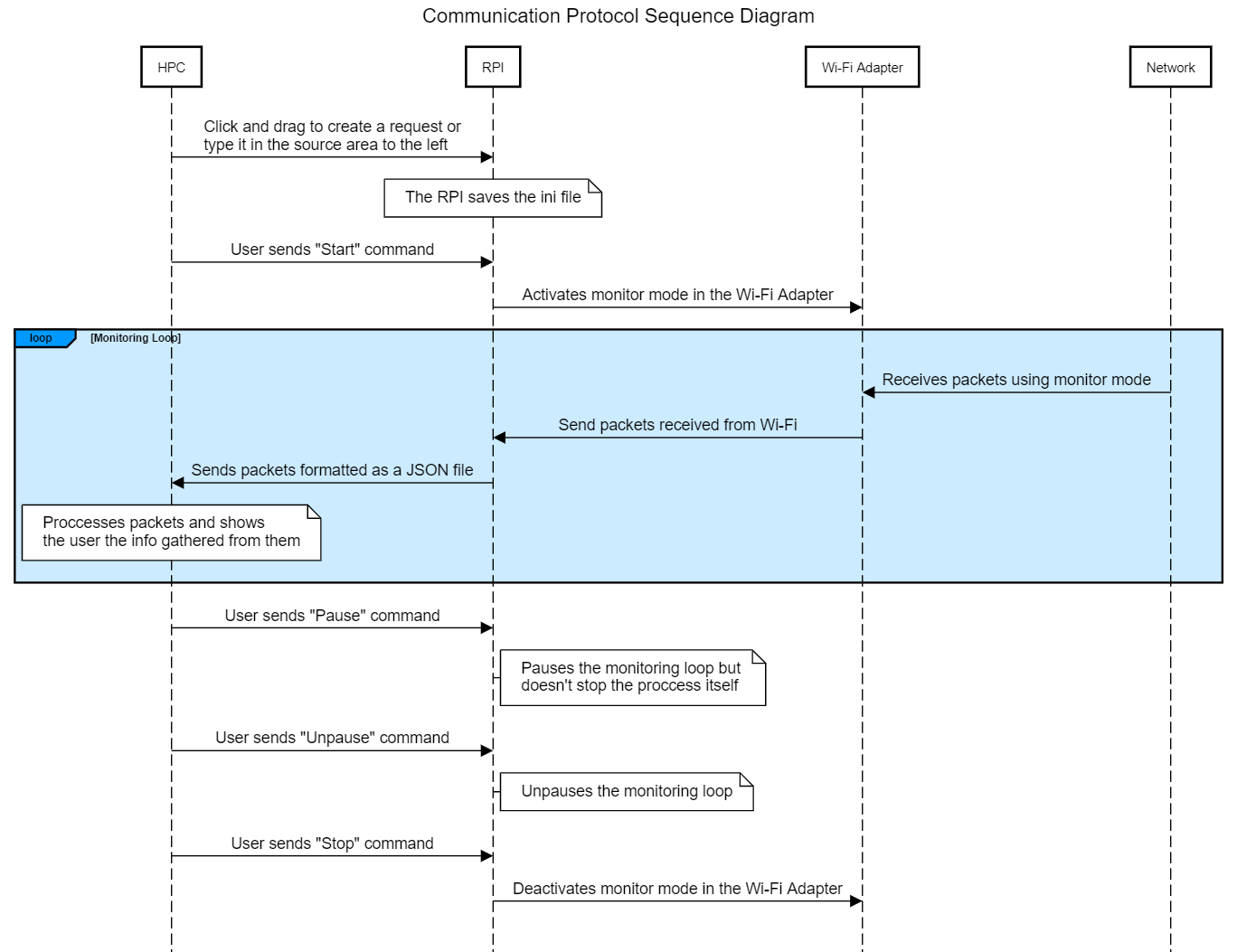


The HPC runs Wireshark and controls the WMP by passing the commands using TCP, formatted as a String. The HTCPC sends the command, that was entered by the user (from the list of commands mentioned above) to the RTCPS. The RTCPS uses the Pipe to execute the command requested by the user.

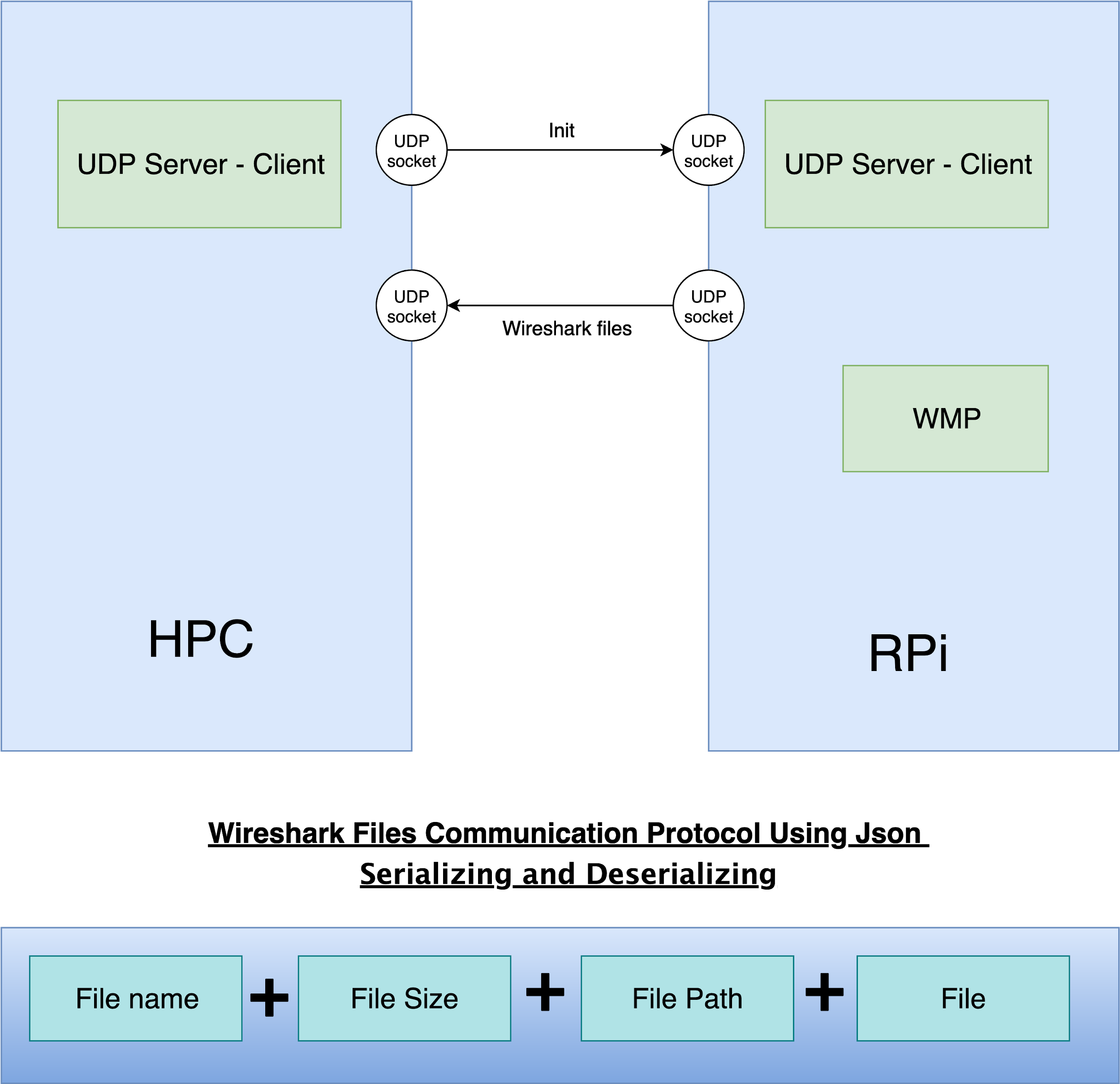
**2.2.2 Communication Protocol**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Command | Sender | Receiver | Description (Goal) | The Role of the Pipe |
| Start | HPC | RPi | Reading the Init file then activating of the WMP as a new process and creation of the Pipe. |  |
| Stop | HPC | RPi | Stopping the WMP and ending the process of the WMP. | Sends the Stop command through the Pipe then ending the process. |
| Pause | HPC | RPi | Stopping the WMP loop. | Sends the Pause command through the Pipe - by doing so, stopping the monitoring loop. |
| Unpause | HPC | RPi | Starts the WMP loop again | Sends the Unpause command through the Pipe - by doing so, restarting the monitoring loop. |
| Switch Channels | HPC | RPi | TBD | TBD |

**Table 2: Control and Execution Communication Protocol**



**2.3 File Transfer**

****

File transferring in the Wi-Fi Sniffer is done using servers and clients. At first, the Init file is transferred to the RPi using the HUDPC and the RUDPS, then when the WMP sniffs data, using the Wi-Fi adapter, it sends the Wireshark files over to the HPC using the RUDPC and the HUDPS. The Wireshark files will be sent in a JSON format in the way mentioned above so the Wireshark plugin will be able to read it properly and in an easy fashion.

**2.4 Wi-Fi Monitoring Program**

The WMP is the program in charge of the interaction with the Wi-Fi Adapter. It has a few main roles. The first role is to change the Wi-Fi Adapter’s mode to monitor mode, monitor mode allows the Wi-Fi Adapter to receive every packet from any network nearby. This causes a problem of overflowing the system with packets the user never requested, that’s why the WMP also acts as an initial filter, it checks for the correct network, IP etc. The final role of the WMP is to format the files as JSON so they can be sent through to the HPC.

**2.5 HPC Class Diagram**

# 

The user will control the program using a user interface - supposedly a separate application, yet more research needs to be done. At first, the user will have to initialize the Sniffing Settings that will be divided into Init and Filter Data. Executing the start command will send via pipe the Init and the commands to the HUDPC and to the HTCPC respectively (see article 2.3 and 2.2).

The JSON files sent from the RPi to the HUDPS are not filtered completely, thus a filter program is required. Due to the need to easily access the files, the program will be written in Python. Its inputs are the filter data and the JSON files. Then it filters the files so only the wanted packets, according to the filter data, will remain. In the end, the program converts the filtered JSON files into PCAP files and saves them in a dedicated files directory.

When it's done, the user will be able to read the packets using Wireshark. If needed for internal purpose, Wireshark could be ran with a written dissector. Because it is intended for internal use, it could be implemented in Lua as an external script - running it will require starting Wireshark with the script using Shell. The dissector could be written in C, but Lua offers easier implementation. With that said, if needed the code could be then converted to C.

# System and Architecture Requirements

## System Requirements

* Raspberry Pi 3 b+ / Raspberry Pi 4
* Wi-Fi Adapter (specific type unknown)

## Architectural Requirements

### General

### Constraints

* + - 1. ***Coding Language***

The servers, clients, filter program and WMP will be programmed using Python and the Wireshark dissector will be programmed using Lua.

* + - 1. ***Operating*** ***System***

HPC’s operating system still needs to be discussed by reviewing the implications of each solution. As for the RPi, the operating system used will be Kali Linux since this Linux version is built to do this kind of assignments and has many tools that can help with it.

1. ***Open Questions***

* We ran into some trouble when trying to test Wi-Fi sniffing using our RPi, thus we could write too much info about how it works and details about the library we would be using.
* The user interface controlling the sniffing process is still not decided - whether it will be a separate application and if it will be a GUI.
  1. ***Alternative Solutions***

Another implementation is having the Filter Data comprised in the Init file. That means the RPi will completely filter the packets sniffed, and the software components in the HPC will only need to convert the JSON files into PCAP.

To do so, RPi will need to run T-Shark with the sniffing filters requested. This solution is problematic because the RPi will have to withstand a great amount of load.