THE UNIVERSITY OF ALABAMA® | College of Engineering

VoIP: Voice over Internet Protocol for Small Single-board Computers

Final Report of Term Project

Students:
Trupeshkumar R. Patel
trpatel2@crimson.ua.edu
David Coleman
dmcoleman1@crimson.ua.edu
Xiaoming Guo
xguo29@crimson.ua.edu

Supervisor:
Dr. Xiaoyan Hong
hxy@cs.ua.edu

Department of Computer Science

May 4, 2022

VoIP: Voice over Internet Protocol for Small Single-board Computers

(Final Report)

Trupesh R. Patel*, David Coleman*, Xiaoming Guo*
*Department of Computer Science
College of Engineering
University of Alabama
Tuscaloosa, AL 35487
{$trpatel2$, $dmcoleman1$, $xguo29$}@crimson.ua.edu

***Abstract*— Asterisk PBX is a software implementation of Private Branch Exchange (PBX), a system used to handle telecommunication services, allowing VoIP (Voice over Internet Protocol) services on computers, including less powerful devices such as Raspberry Pi. However, using smaller single-board devices come with the limitation of computing power and storage capacity. As a result of these limitations are dropping packets, cutting off the callers while handling a more significant number of calls simultaneously. In this project, the authors provides the proof-of-concept by analyining the packets delays, network jitters, end-to-end network latency, and bandwidth between an Asterisk PBX system (which is install on Raspberry Pi 4) and IP clients that are using VoIP as service. This analyis includes internet through wired, wireless, cellular wireless connection.**

I. Introduction & Background

Voice always has multiple functionalities among the information flowing at the edge of the IoT network. It carries valuable content, reflects the conditions of the environment, and can be used to command other entities through acoustic actuators and phone calls.

Implementing voice systems at the edge of an IoT network typically faces challenges, namely that edge devices are always constrained by computing power, bandwidth contention, and energy consumption. Therefore, it is not feasible to implement a complete TCP/IP stack on each node and give all nodes the ability to connect to remote entities outside the local network.

IP-PBX (IP-based Private Branch Exchange) provides a comprehensive solution to address the aforementioned issues. As the prototype of IP PBX, the design of the traditional PBX system is to serve a private organization, in which both the geographic area and the communication connection are limited to a specific scope. Interconnections between internal phones are without cost, while only central office lines provide connections to the public switched telephone network (PSTN). This scheme meets our expectations for edge IoT communication – internal communication does not occupy egress bandwidth, and some switcher servers still reserve the communication egress to the outside. Leveraging VoIP technology, IP-PBX has ported the PBX scheme to the Internet, replacing telephone lines with packet-switching networks. The IP-based paradigm offers better scalability and lowers the cost same as the Internet brings to other domains.

Several mature implementations of the IP-PBX paradigm are available, including 3CX and Asterisk PBX. Asterisk is an open-source software package that can run all the PBX functions, usually on a Linux operating system platform. It contains the functions of PBX and some other additional features. Along with the essential telephony services, voicemail services, conference calling, interactive voice response, and call queuing are also provided by Asterisk. It also provides multi-party calling, display caller ID (display calling number). To interact with digital telephone equipment and analog telephone equipment, Asterisk needs the support of PCI hardware, the most famous of which is provided by the Digium platform.

From the architecture perspective, Asterisk serves as a middleware function, connecting the underlying telephony technology and the upper-level telephony applications. Both PBX and IVR (Interactive Voice Response) functionalities are integrated within Asterisk. Using compatible PCI hardware, Asterisk supports traditional telephone lines, including TDM (Time Division Multiplexing), TI/El PRI/PRA&RBS (Robbed Bit Signal) mode, analog telephone line/analog telephone (POTS), ISDN (Integrated Services Digital Network) and BRI (Basic Rate) and PRI (Primary Rate). Also, due to the PCI hardware support feature, Raspberry Pi can be used to implement the Asterisk instance.

By using small single-board computers such as Raspberry Pi, any types of businesses can deploy the cost affordable IP based PBX system. Raspberry Pi is small and light weight device, and runs on low power consumsption. So, having it for single tasks like VoIP with multiple functionallity provided by Asterisk PBX is very easy and economical. However, every since this device are smaller and lighter, it can only handle limited number of tasks. So, this paper will analyze the boudrise of Raspberry Pi by packets delays, network jitters, end-to-end network latency, and bandwidth through wired, wireless, cellular wireless connection.

This paper is organized as following convention. The next Section II, experiment setup information on this project. Section III shows detailed decription on each experments.

II. Software & Hardware Setup

A. Hardware Setup

1) Raspberry Pi 4 from CanaKit (32 GB EVO+, 4GB RAM)
2) Cell Phone (iPhone/Android)
3) Desktop/Laptop (MacBook/Windows/Linux)

B. Software Setup

1) Raspberry Pi OS with desktop:
   - System: 32-bit
   - Kernel version: 5.15
   - Debian version: 11 (bullseye)
2) Asterisk:
   - Version: 18.11.1
   - Releases: Long Term Support (LTS)
   - Build: From source http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-18-current.tar.gz
3) ZoiPer
   - Version: 5
   - Use: non-commercial
   - For: Mac, Windows, Linux, iOS, Android
4) MicroSIP
   - Version: 3.20.7
   - Use: non-commercial
   - For: Windows
5) Telephone
   - Version: 1.5.2
   - Use: non-commercial
   - For: Mac
6) Wireshark
   - Version: 3.6.3
   - For: Mac, Windows, Linux

III. Experiment

A. Basic SIP Analysis

The Session Initiation Protocol (SIP) is an application layer control protocol for establishing, changing and terminating multimedia sessions, where the sessions can be IP telephony, multimedia sessions or multimedia conferences. SIP is the core protocol of many IP-based PBX applications, including Asterisk.

To demonstrate the main components of the SIP communication, a simplified scenario is implemented, as shown in Figure Figure 1. Two softphone applications are running on the same computer, and an IP-based PBX application (Asterisk), running on a Raspberry Pi 4, is serving as the PBX server.

The SIP process starts with registration. All SIP terminals as User Agents should register with the registration server to inform their location, session capability, and other information.

Usually, when the SIP terminal (User Agent) is powered on or configured to perform a registration operation, a registration request message (REGISTER) is sent to the registration server, carrying all the information that needs to be registered. After receiving the registration request message, the registration

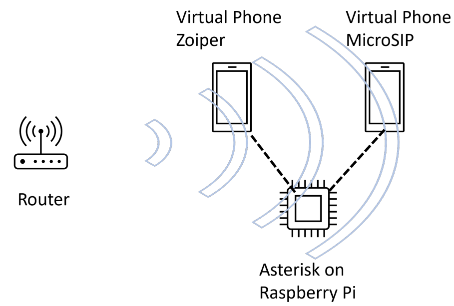

Fig. 1: Simplified Scenario

server sends a response message to the terminal to inform it that the request message has been received. If the registration is successful, it will send a "200 OK" message to the terminal. However, the most common response is a "401 Unauthorized" message because authorization is required in most of the implantations for security purposes.

To trace and see all types of packets tramissions, author used Wireshark open-souce software. Figure 2 is a flow sequence of the one single call from Zoiper5 user (IP: 192.168.60.106:63771) to MicroSIP user (IP:192.168.60.106:53493). In this, senario their are actual two connections, first from Zoiper5 user to Asterisk (shown in left-side of Figure 2) and second Asterisk to MicroSIP (shown in right-side of Figure 2).

The SIP protocol adopts the Client-Server pattern, in which the calls are established between User Agents through the proxy server.

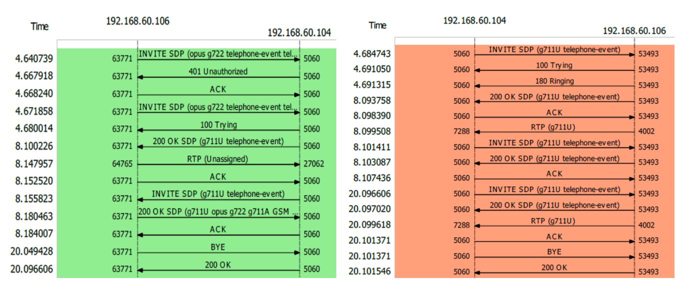

Fig. 2: SIP packet trasmission from Zoiper to Asterisk(left-side), Asterisk to MicroSIP (right-side)

IV. Obstacles

A. Local Network Issues

An initial problem found when working with PBX software was configuring connections for the device hosting it and for the devices that are connected with it (e.g. smart phones and computers). As IP-PBX uses IP networks for its functionality, a single local network was used for testing the functionality of the Raspberry Pi, which the Pi itself and all machines running VoIP software were connected to, Eduroam. Eduroam is the primary wireless network used within the University of

Alabama campus, with access points in each building, allowing devices across buildings to stay on one local area network.

This provided a few benefits to the experiment, most prominently allowing for VoIP calls to be made from greater distances (i.e. from a different building) while simplifying the process of connecting devices to the Raspberry Pi. However, an issue found with this was that Eduroam was not initially allowing the Pis to connect. An additional script was required to sign into the network as students. After the script was ran and the Pis rebooted, they could freely join the network afterward.

```bash
#!/bin/bash
ifconfig wlan0 down
ifconfig eth0 down
killall wpa_supplicant
echo "

allow-hotplug wlan0
iface wlan0 inet manual
        wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

iface wlan0 inet dhcp

" >> /etc/network/interfaces
read -p 'Username(email-address): ' username
read -sp 'Password: ' password
echo "

network={
        ssid=\"eduroam\"
        key_mgmt=WPA-EAP
        eap=PEAP
        identity=\"$username\"
        password=\"$password\"
        phase2=\"auth=MSCHAPV2\"
}

" >> /etc/wpa_supplicant/wpa_supplicant.conf

echo "Testing conntion with \"eduroam\""

timeout 20 wpa_supplicant -i wlan0 -c /etc/wpa_supplicant/wpa_supplicant.conf > test_config_eduroam.txt

test=`grep "authentication completed successfully" test_config_eduroam.txt | wc -l`

if [ $test -ge 1 ]
then
        echo "\"eduroam\" Successfully connected!!!"
else
        echo "\"eduroam\" Conection Failed :("
fi

ifconfig wlan0 up
ifconfig eth0 up

/etc/init.d/dhcpcd restart

reboot
```

Fig. 3: Script used to join Eduroam network

Another problem more central to the use of IP-PBX software is the complication of connecting remote devices to the Raspberry Pi. Within Eduroam, VoIP software only required the domain of the Raspberry Pi within the local area network to register on the IP-PBX software the Pi was running. However, if the device was on a different network, it proved incapable of finding the Pi as easily. There are a few possible explanations for this, such as there being a firewall blocking most traffic outside of Eduroam or NAT changing the IP of the Pi from outside Eduroam. Without the ability to make calls from different networks, making long distance VoIP calls proved to be quite limited, and the available tests for the Raspberry Pi were limited. While this issue may have been resolved with further research, it was ultimately decided that having all devices connect to Eduroam still allowed for enough trials to analyze the Pis functionality in running IP-PBX software.

B. Complexity of IP-PBX

IP-PBX proved to be quite complex in setup and configuration. Asterisk PBX, a mature open-source implementation, was utilized. Initial compilation of Asterisk showed that there were many options on modules to be used, many of which were extraneous. After setup was done, the PBX needed to be further configured to allow for distinct users to be registered, to denote what should be done if a call was made to a certain number, and to dictate what kind of channel to use for SIP. Determining what configuration files needed to be used and changed and in what ways was a major obstacle in beginning the testing of the Raspberry Pi.

```
pi@raspberrypi:~ $ ls /etc/asterisk/
asterisk.conf     extconfig.conf    indications.conf
cdr.conf          extensions.conf   logger.conf
cli.conf          func_odbc.conf    manager.conf
codecs.conf       http.conf         modules.conf
confbridge.conf   iaxprov.conf      musiconhold.conf

pjsip.conf        res_parking.conf       sip.conf
queuerules.conf   res_snmp.conf          sip_notify.conf
queues.conf       res_stun_monitor.conf  udptl.conf
res_fax.conf      rtp.conf               users.conf
res_odbc.conf     say.conf               voicemail.conf
```

Fig. 4: The configuration files for Asterisk PBX. Note that the majority of edits needed to be made to sip.conf and extensions.conf

C. Abundance of Third Party Software

Many pieces of third party software needed to be utilized to conduct these tests, each adding more setup time and complication to the experiments. Asterisk PBX needed to be downloaded onto the Raspberry Pi to allow the Pi to provide IP-PBX services. Zoiper was needed to make VoIP calls and register devices on the Pi for IP-PBX services, with MicroSIP and Telephone needed to make additional calls, as Zoiper limits the number of free users. Wireshark was required to gather more information on the packets sent to and from the Pi while hosting calls to analyze its performance. Additionally, other services, such as SIP.US (used for SIP trunking) and Clumsy (used for network emulation) were considered for use until they were deemed unnecessary. The number of extraneous services utilized exacerbated the complexity of performing these experiments.

V. Future Work

While this may act as a basic foundation for proving the efficacy of the Raspberry Pi as a cheap means to implement IP-PBX software, there are several means of expanding upon this work.

A. Further Testing

More experiments can be performed to determine the limitations that Raspberry Pis can prove to have when running IP-PBX software. A major test would be stressing the Pi by seeing how many calls it can handle in situations where bandwidth is plentiful, with the CPU and RAM of the Pi being the major bottlenecks at that point. This same test could be performed using different models of the Pi to see how the difference in CPU and RAM can affect VoIP calls. Analyzing this data could provide a way to determine the best board to purchase based on how many calls are needed to be handled.

A limitation of data for outside network calls restricts current analysis. Future studies could attempt to have separate Raspberry Pis running PBX software on different networks, while allowing VoIP calls to be made between them. Data could then be collected based on test calls between these IP-PBX instances. This would prove useful in analyzing the effectiveness of the Pi in making outside network calls.

B. Horizontal Scaling

A powerful characteristic of cheap computers such as a Raspberry Pi is their suitability for horizontal scaling. If an institution grows large enough that one Pi cannot handle the all of the institution's VoIP calls, there are two methods of scaling up the memory and processing power available for IP-PBX services. The first is to scale the system vertically; replace the Pi with a single, more powerful computer with more memory, and either sell the Pi or reuse it for some other purpose. The second is to scale the system horizontally; connect more Raspberry Pis with the original and have them share the workload of the VoIP calls. As Raspberry Pis are inexpensive, this allows for a cheaper alternative to buying a more powerful computer. Further works can delve into whether this is a possible technique to implement with IP-PBX software, and whether or not horizontal scaling provides many benefits for IP-PBX services.

VI. Conclusion

A single Raspberry Pi with 4GB of RAM is quite capable of running IP-PBX software without any noticeable issues. However, configuring the software to allow for VoIP calls can be quite complicated, and requires a great deal of outside instruction to fully understand. As well, a vital component of using the Raspberry Pi for these purposes is considering the network involved. If the Pi or VoIP devices are in areas with weak signal strength, jitter can grow rapidly, packet loss can begin to noticeably occur, connections can be dropped abruptly, and delays can become much more noticeable. If these effects occur, either due to weak signal strength or due to other possible issues, call quality can suffer dramatically. Consideration is required to install a usable and useful IP-PBX system for VoIP calls.