Assignment 3

Asterisk and VoIP

991451344

Timothy Pang

Executive Summary:

In this Assignment, we discover Asterisk and VoIP and how they are used. We discover what the asterisk modules are, the Asterisk Gateway interface, and the Asterisk Extensions Language and how each one of the are used in Asterisk and its dial plan. We investigate the use of trunking and the stress test tool SIPP.

Lab Objective:

- Deploy VoIP “phones”
- Build a basic VoIP exchange service off an Asterisk PBX node
- Interconnect multiple nodes via PBX “trunking”.
- Design and implement a call plan to allow local and trunked calls between PBX sites
- Analyze SIP traffic at the session layer as well as headers and payloads.
- Analyze at least two media traffic types.
- Explore SIP security with SIPP.

1.

Dial Plan:

SIP.conf:

```
 GNU nano 4.8                                        sip.conf                                         Modified
[general]
allowguest=no                         ;disable unauthenticated call
allowoverlap=no                       ;disables overlap
srvlookup=no                          ;disables dns
bindport=5060                         ;port address
transport=udp                         ;transport medium
tcpenable=no                          ;no tcp
tcpbindaddr=0.0.0.0                    ;ip addr to bind socket to all
alwaysauthreject=yes                   ;any incoming invite to be rejected, rejects all with same response
session-timers=refuse                  ;do not run session timers in any case
localhost=192.168.86.0/255.255.255.0    ;local host of the server
nat=force_rport,comedia
insecure=invite
allow=ulaw                           ;used in US aka G.711


[voip](!)
context=internal                  ;defined in extensions.conf
type=friend                       ;Accepts calls requiring only authorization mathes rather than IP
host=dynamic                      ;device will register with asterisk
disallow=all
allow=ulaw

[7001](voip)                    ;Account
secret=123                      ;Password to login to account

[7002](voip)
secret=456

[7003](voip)
secret=789
```

Figure 1.

Created a general phone dial plan that is used just for calling between the two or whichever softphone is on the same NAT, then created the accounts for it with the password inside.

Extensions.conf:

```
  GNU nano 4.8                                          extensions.conf
[internal]

exten => 7001,1,Answer()            ;Answer first
exten => 7001,2,Dial(SIP/7001)      ;Way to call / User
exten => 7001,3,Hangup()            ;Hangup

exten => 7002,1,Answer()
exten => 7002,2,Dial(SIP/7002)
exten => 7002,3,Hangup()

exten => 7003,1,Answer()
exten => 7003,2,Dial(SIP/7003)
exten => 7003,3,Hangup()
```

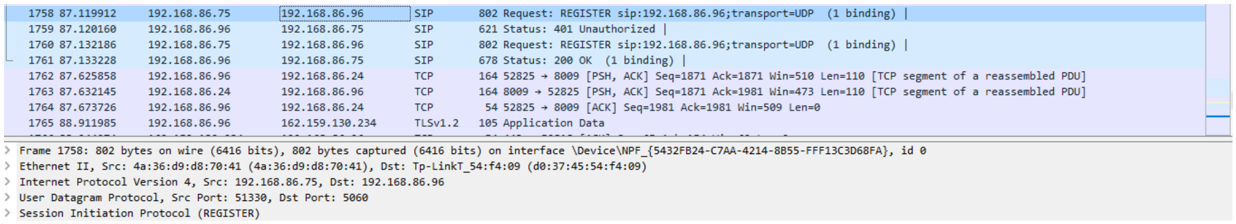
Figure 2.

In the Extensions.conf program, it shows the way the account / user works in steps.

2.

Wireshark Packets:

Sip caller register:

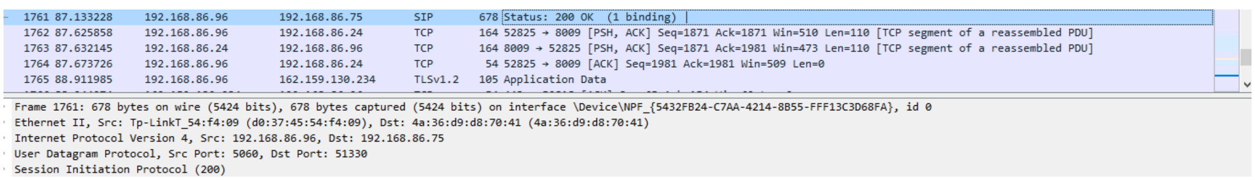


1758	87.119912	192.168.86.75	192.168.86.96	SIP	802 Request: REGISTER sip:192.168.86.96;transport=UDP (1 binding) |
1759	87.120160	192.168.86.96	192.168.86.75	SIP	621 Status: 401 Unauthorized |
1760	87.132186	192.168.86.75	192.168.86.96	SIP	802 Request: REGISTER sip:192.168.86.96;transport=UDP (1 binding) |
1761	87.133228	192.168.86.96	192.168.86.75	SIP	678 Status: 200 OK (1 binding) |
1762	87.625858	192.168.86.96	192.168.86.24	TCP	164 52825 → 8009 [PSH, ACK] Seq=1871 Ack=1871 Win=510 Len=110 [TCP segment of a reassembled PDU]
1763	87.632145	192.168.86.24	192.168.86.96	TCP	164 8009 → 52825 [PSH, ACK] Seq=1871 Ack=1981 Win=473 Len=110 [TCP segment of a reassembled PDU]
1764	87.673726	192.168.86.96	192.168.86.24	TCP	54 52825 → 8009 [ACK] Seq=1981 Ack=1981 Win=509 Len=0
1765	88.911985	192.168.86.96	162.159.130.234	TLSv1.2	105 Application Data

Frame 1758: 802 bytes on wire (6416 bits), 802 bytes captured (6416 bits) on interface \Device\NPF_{5432FB24-C7AA-4214-8B55-FFF13C3D68FA}, id 0
Ethernet II, Src: 4a:36:d9:d8:70:41 (4a:36:d9:d8:70:41), Dst: Tp-LinkT_54:f4:09 (d0:37:45:54:f4:09)
Internet Protocol Version 4, Src: 192.168.86.75, Dst: 192.168.86.96
User Datagram Protocol, Src Port: 51330, Dst Port: 5060
Session Initiation Protocol (REGISTER)

Figure 3.

SIP Authentication:

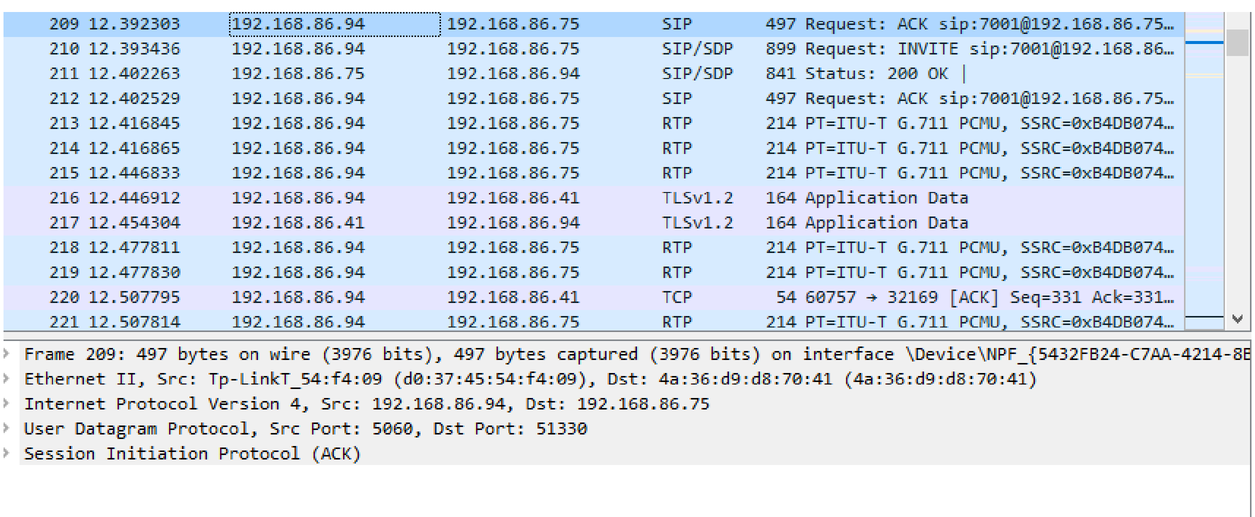


1761	87.133228	192.168.86.96	192.168.86.75	SIP	678 Status: 200 OK (1 binding) |
1762	87.625858	192.168.86.96	192.168.86.24	TCP	164 52825 → 8009 [PSH, ACK] Seq=1871 Ack=1871 Win=510 Len=110 [TCP segment of a reassembled PDU]
1763	87.632145	192.168.86.24	192.168.86.96	TCP	164 8009 → 52825 [PSH, ACK] Seq=1871 Ack=1981 Win=473 Len=110 [TCP segment of a reassembled PDU]
1764	87.673726	192.168.86.96	192.168.86.24	TCP	54 52825 → 8009 [ACK] Seq=1981 Ack=1981 Win=509 Len=0
1765	88.911985	192.168.86.96	162.159.130.234	TLSv1.2	105 Application Data

Frame 1761: 678 bytes on wire (5424 bits), 678 bytes captured (5424 bits) on interface \Device\NPF_{5432FB24-C7AA-4214-8B55-FFF13C3D68FA}, id 0
Ethernet II, Src: Tp-LinkT_54:f4:09 (d0:37:45:54:f4:09), Dst: 4a:36:d9:d8:70:41 (4a:36:d9:d8:70:41)
Internet Protocol Version 4, Src: 192.168.86.96, Dst: 192.168.86.75
User Datagram Protocol, Src Port: 5060, Dst Port: 51330
Session Initiation Protocol (200)

Figure 4.

Request Ack from user 7002 to 7001:

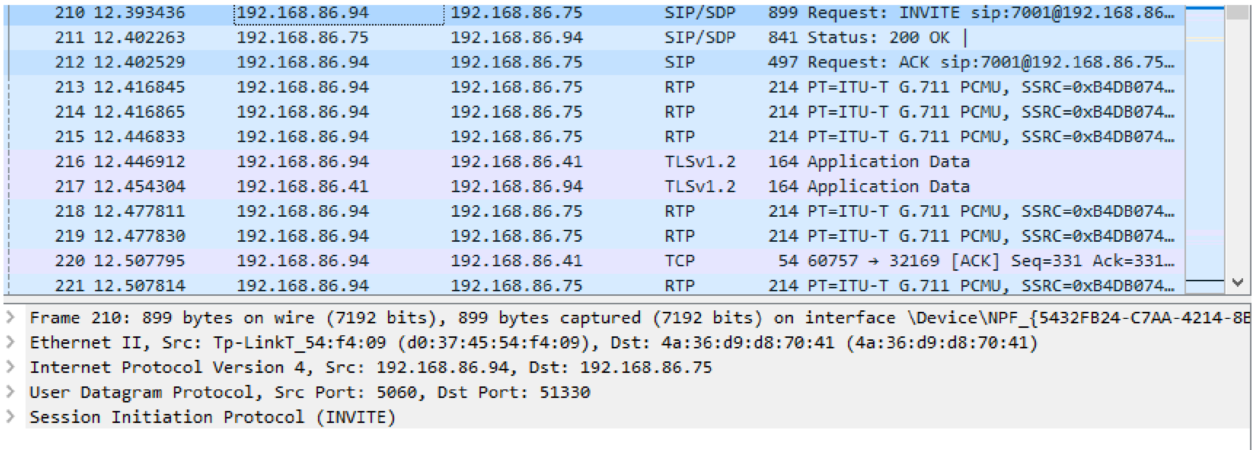


209	12.392303	192.168.86.94	192.168.86.75	SIP	497 Request: ACK sip:7001@192.168.86.75…
210	12.393436	192.168.86.94	192.168.86.75	SIP/SDP	899 Request: INVITE sip:7001@192.168.86…
211	12.402263	192.168.86.75	192.168.86.94	SIP/SDP	841 Status: 200 OK |
212	12.402529	192.168.86.94	192.168.86.75	SIP	497 Request: ACK sip:7001@192.168.86.75…
213	12.416845	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
214	12.416865	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
215	12.446833	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
216	12.446912	192.168.86.94	192.168.86.41	TLSv1.2	164 Application Data
217	12.454304	192.168.86.41	192.168.86.94	TLSv1.2	164 Application Data
218	12.477811	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
219	12.477830	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
220	12.507795	192.168.86.94	192.168.86.41	TCP	54 60757 → 32169 [ACK] Seq=331 Ack=331…
221	12.507814	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…

Frame 209: 497 bytes on wire (3976 bits), 497 bytes captured (3976 bits) on interface \Device\NPF_{5432FB24-C7AA-4214-8E
Ethernet II, Src: Tp-LinkT_54:f4:09 (d0:37:45:54:f4:09), Dst: 4a:36:d9:d8:70:41 (4a:36:d9:d8:70:41)
Internet Protocol Version 4, Src: 192.168.86.94, Dst: 192.168.86.75
User Datagram Protocol, Src Port: 5060, Dst Port: 51330
Session Initiation Protocol (ACK)

Figure 5.

Request: Invite:

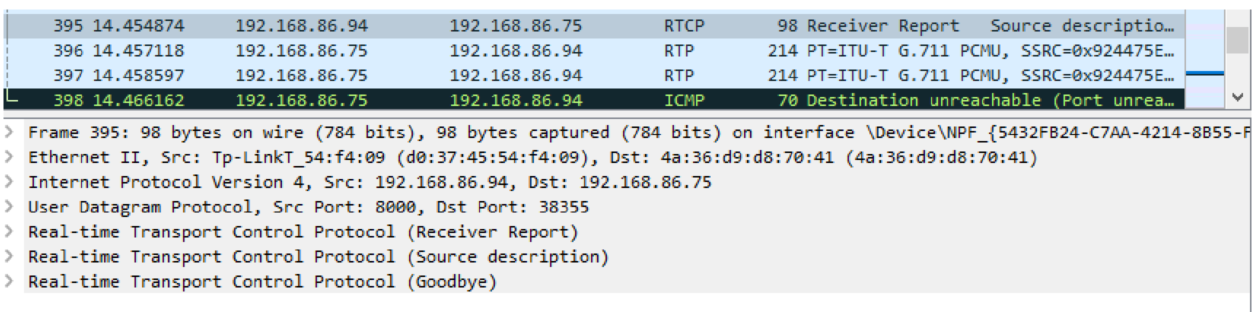


210	12.393436	192.168.86.94	192.168.86.75	SIP/SDP	899 Request: INVITE sip:7001@192.168.86…
211	12.402263	192.168.86.75	192.168.86.94	SIP/SDP	841 Status: 200 OK |
212	12.402529	192.168.86.94	192.168.86.75	SIP	497 Request: ACK sip:7001@192.168.86.75…
213	12.416845	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
214	12.416865	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
215	12.446833	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
216	12.446912	192.168.86.94	192.168.86.41	TLSv1.2	164 Application Data
217	12.454304	192.168.86.41	192.168.86.94	TLSv1.2	164 Application Data
218	12.477811	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
219	12.477830	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…
220	12.507795	192.168.86.94	192.168.86.41	TCP	54 60757 → 32169 [ACK] Seq=331 Ack=331…
221	12.507814	192.168.86.94	192.168.86.75	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0xB4DB074…

```
> Frame 210: 899 bytes on wire (7192 bits), 899 bytes captured (7192 bits) on interface \Device\NPF_{5432FB24-C7AA-4214-8B
> Ethernet II, Src: Tp-LinkT_54:f4:09 (d0:37:45:54:f4:09), Dst: 4a:36:d9:d8:70:41 (4a:36:d9:d8:70:41)
> Internet Protocol Version 4, Src: 192.168.86.94, Dst: 192.168.86.75
> User Datagram Protocol, Src Port: 5060, Dst Port: 51330
> Session Initiation Protocol (INVITE)
```

Figure 6.

Receiver Report:



395	14.454874	192.168.86.94	192.168.86.75	RTCP	98 Receiver Report Source descriptio…
396	14.457118	192.168.86.75	192.168.86.94	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0x924475E…
397	14.458597	192.168.86.75	192.168.86.94	RTP	214 PT=ITU-T G.711 PCMU, SSRC=0x924475E…
398	14.466162	192.168.86.75	192.168.86.94	ICMP	70 Destination unreachable (Port unrea…

```
> Frame 395: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface \Device\NPF_{5432FB24-C7AA-4214-8B55-F
> Ethernet II, Src: Tp-LinkT_54:f4:09 (d0:37:45:54:f4:09), Dst: 4a:36:d9:d8:70:41 (4a:36:d9:d8:70:41)
> Internet Protocol Version 4, Src: 192.168.86.94, Dst: 192.168.86.75
> User Datagram Protocol, Src Port: 8000, Dst Port: 38355
> Real-time Transport Control Protocol (Receiver Report)
> Real-time Transport Control Protocol (Source description)
> Real-time Transport Control Protocol (Goodbye)
```

Figure 7.

SDP Payload:

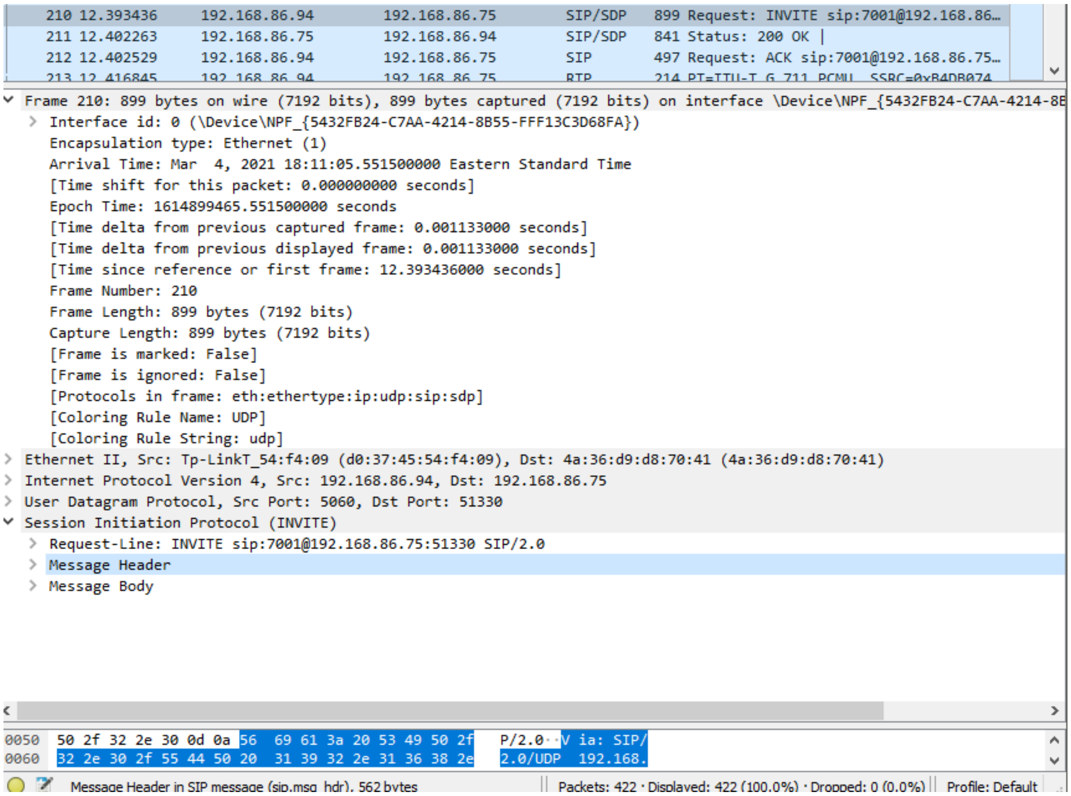


Figure 8.

Inside the SDP Packet:

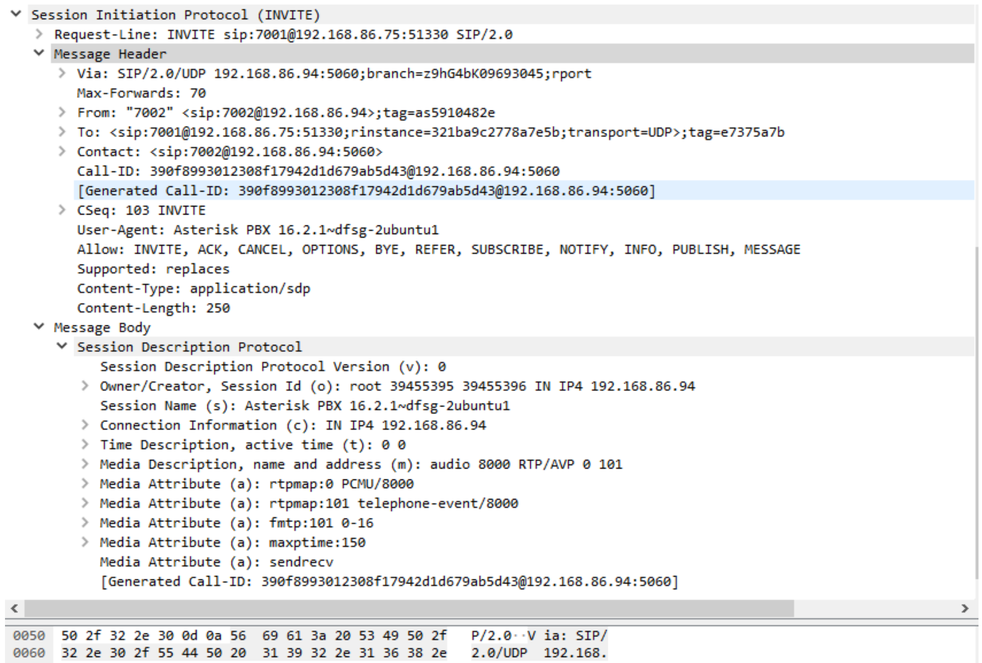


Figure 9.

- Contains a message Header and body.
- Inside the header contains who it is from as well as the contact information of the system
- Inside the body it contains the owner and information as well as the session name and connection information

RTP header:

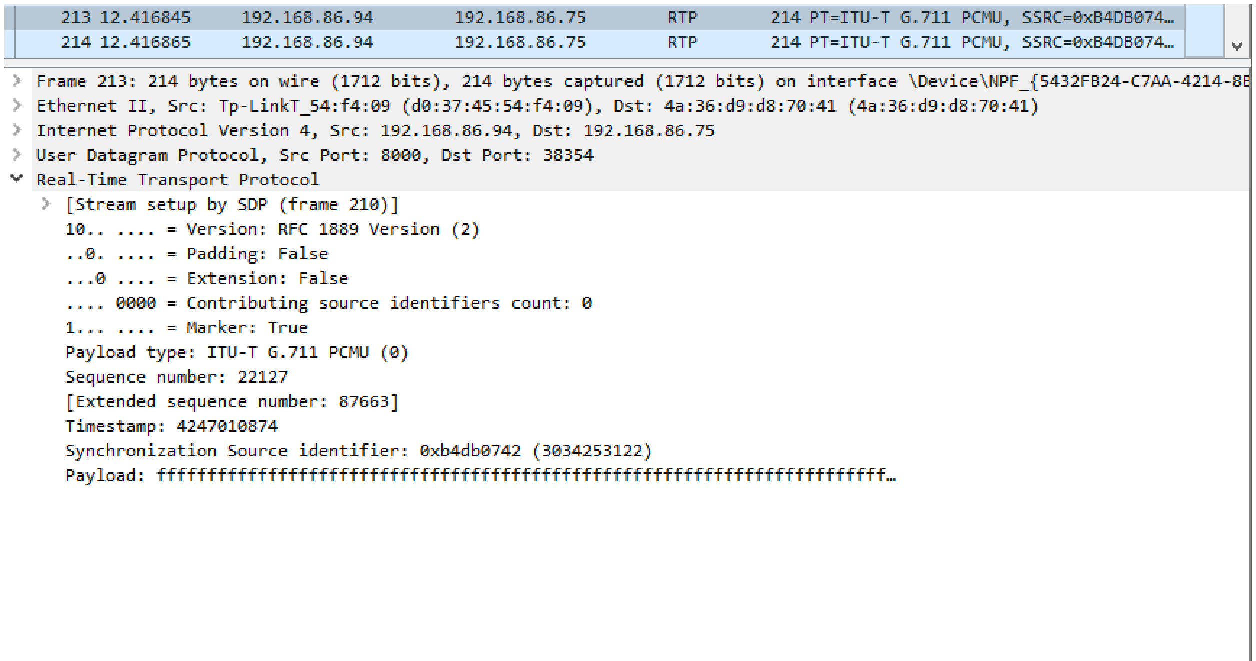


Figure 10.

The RTP header shows the payload type, it is sending a broadcast message to find whoever owns the IP address of the IP phone they are trying to reach. It includes the payload type, sequence number, and the actual payload.

5 Key characteristics of RTP:

- Has the ability to reconstruct timing.
- Loss detection
- Security
- Content delivery
- Identification of encoding schemes

3.

Conference calling:

I could not use conference call, because of a showstopper. However, looking into how sip and conference calling works I discovered that when a conference call starts, compared to a normal call. A normal call begins with invite, trying, ringing, ok and ack. With a conference call because the call is already happening so when you add in another call, the function of ringing is not implemented.

In Wireshark, I believe when for example a user calls a buyer and would like to conference call with an accountant to finalize a deal, the buyer is put on hold and the user makes another call to the accountant, with the accountant on the phone, with the accountant on the phone, when the user presses the conference call or add button the accountant is placed on hold, and when the buyer is added then both parties are then taken off hold and put in a 3 way call together.

The streams in Wireshark are separate for each call, however there is an option to merge the two streams together to visualize how it works. The streams start off with the first call to the buyer which includes the invite, trying, ringing, ok and ack. Then it puts the buyer on hold then continues with ringing the accountant, in Wireshark the stream separates the calls so they each have the same steps, when merged they also include the two invites to join the conference as well as the acknowledgement of their participation.

4.

Trunking:

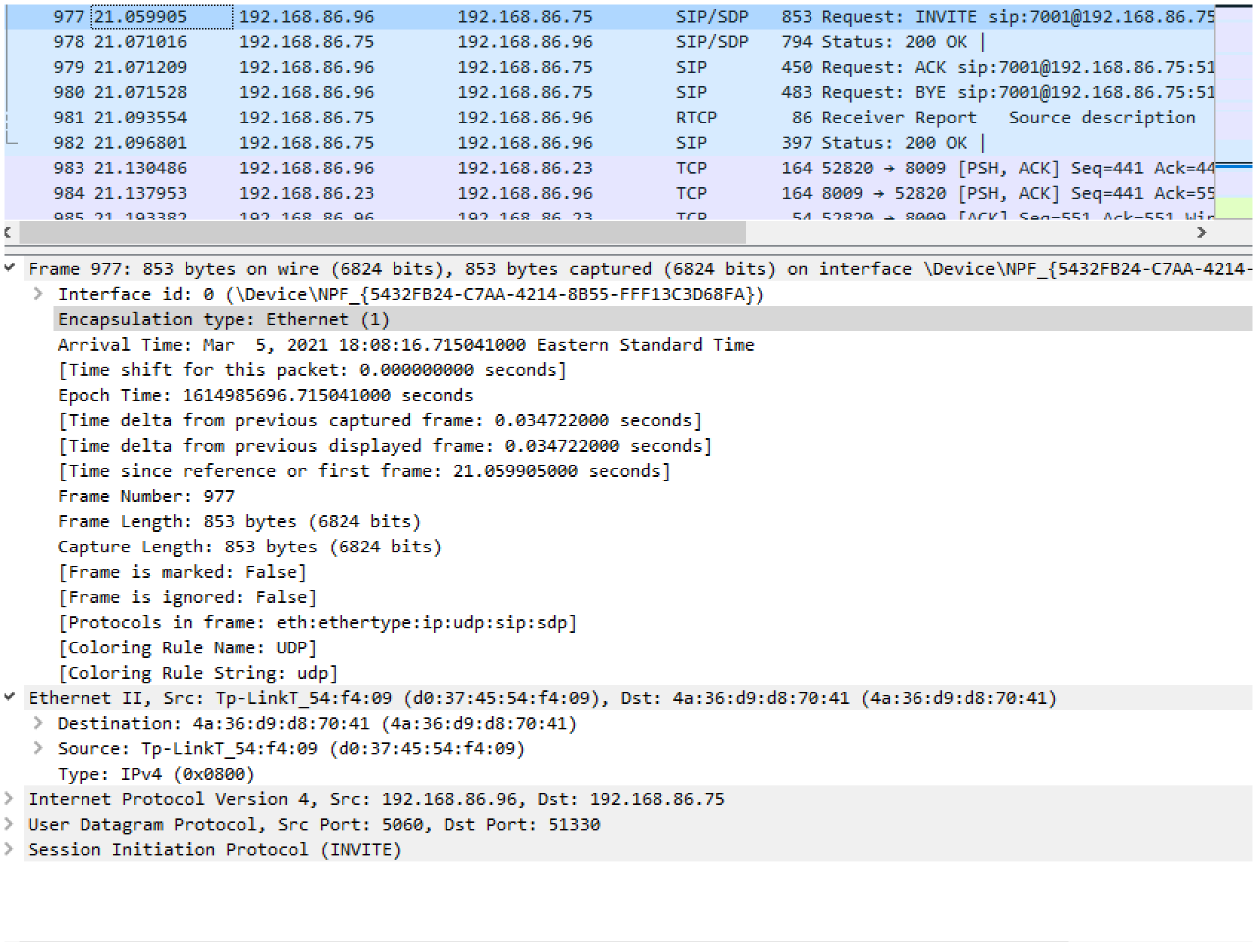


Figure 11.

In the trunk, canreinvite is set to “no”, with this set the RTP messages are shown less. Trying it with canreinvite set to “yes” the RTP messages appear more frequently when the user answers.

Trunking:

Trunking is used for sharing a connection between 2 entities, it is to connect 2 systems together. It is a method of sending voice or video over the internet. It allows for local, long distance and international calls. It is a way to call outside to a HW IP phone from a computer because they are connected on the same IP NAT.

Call Flow:

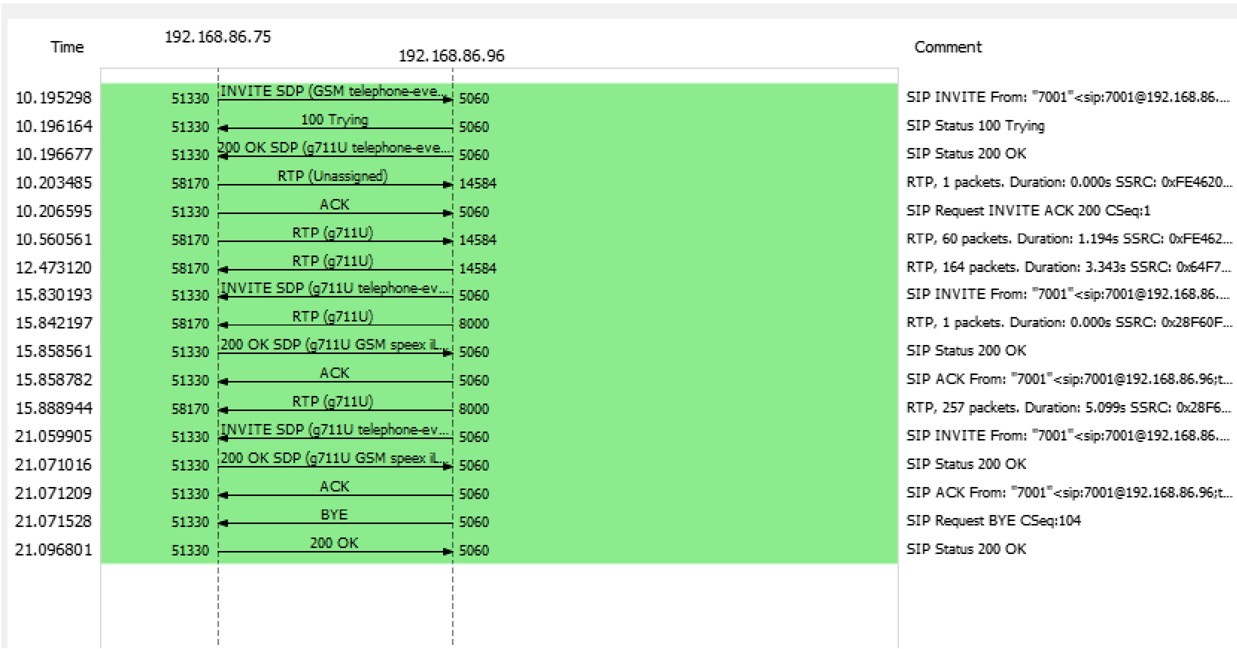


Figure 12.

In the call flow, both servers are not kept in because they are already connected via the dial. When the two parties are connected in the call the servers have already acknowledged each other / discovered each other so that they do not need the servers to find the other caller.

The canreinvite when set to “no” will not allow any RTP messages to be exchanged directly through endpoints, however when it is set to “yes” the RTP messages can be exchanged through the endpoints. When setting the canreinvite option to yes will send invites even when the call is established, whereas when set to no, asterisk will stop sending the invite once the call is established. Even when canreinvite is set to no, it only disables asterisk sending the invites not all the invites.

5.

For G.711:

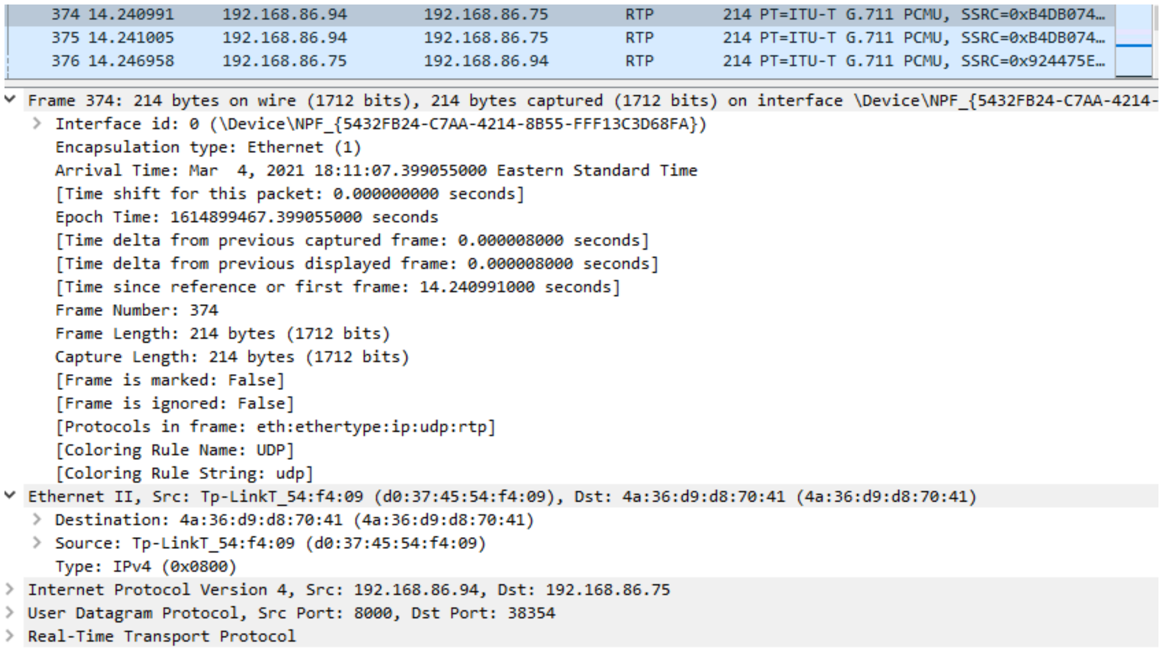


Figure 13.

The Figure above is the G.711 at the beginning.

Name:	**Value:**
PPS:	14.7 ms
Packet Size:	214 bytes
IPG:	14 ms
Jitter:	11.84 ms
Bandwidth:	1.6 bps

For GSM:

```
335 6.004217   192.168.86.96   192.168.86.75   RTP   87 PT=GSM 06.10, SSRC=0x3A2EA01A, Seq=7747, Time=205628053, Mark
336 6.019938   192.168.86.96   192.168.86.75   RTP   87 PT=GSM 06.10, SSRC=0x3A2EA01A, Seq=7748, Time=205628213
337 6.051933   192.168.86.96   192.168.86.75   RTP   87 PT=GSM 06.10, SSRC=0x3A2EA01A, Seq=7749, Time=205628373

Frame 335: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface \Device\NPF_{5432FB24-C7AA-4214-8B55-FFF13C3D68FA}, id 0
  Interface id: 0 (\Device\NPF_{5432FB24-C7AA-4214-8B55-FFF13C3D68FA})
  Encapsulation type: Ethernet (1)
  Arrival Time: Mar  5, 2021 16:02:59.032858000 Eastern Standard Time
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1614978179.032858000 seconds
  [Time delta from previous captured frame: 0.016995000 seconds]
  [Time delta from previous displayed frame: 0.016995000 seconds]
  [Time since reference or first frame: 6.004217000 seconds]
  Frame Number: 335
  Frame Length: 87 bytes (696 bits)
  Capture Length: 87 bytes (696 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:udp:rtp]
  [Coloring Rule Name: UDP]
  [Coloring Rule String: udp]
Ethernet II, Src: Tp-LinkT_54:f4:09 (d0:37:45:54:f4:09), Dst: 4a:36:d9:d8:70:41 (4a:36:d9:d8:70:41)
  Destination: 4a:36:d9:d8:70:41 (4a:36:d9:d8:70:41)
  Source: Tp-LinkT_54:f4:09 (d0:37:45:54:f4:09)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.86.96, Dst: 192.168.86.75
User Datagram Protocol, Src Port: 8000, Dst Port: 58170
Real-Time Transport Protocol
```

Figure 14.

The Figure above is GSM at the beginning.

Name:	**Value:**
PPS:	0.6 sec
Packet Size:	87 bytes
IPG:	0.016 ms
Jitter:	12.58 ms
Bandwidth:	0.58 bps

6.

Linktropy device:

The Linktropy part of the Assignment was not possible due to a showstopper. However, because I could not sign out a device, I researched instead what it did for the VoIP calling. The maximum MOS score for the codec G 7.11 is 4.4, which according to the table means that if a number has a MOS score over 4.34, the call quality is the best, which G 7.11 codecs maximum is the best call quality. As the monitoring continues the latency will decrease slowly over time, however with the G 729 codec, because it is a compressed codec, the voice quality can quickly degrade as the latency increases. The G 7.11 codec maintains better quality with more latent connections.

7.

SIPP in Linux:

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
10.0(10000 ms)/1.000s   5061     929.19 s          831  192.168.86.99:5060(UDP)

  0 new calls during 5.436 s period      5436 ms scheduler resolution
  10 calls (limit 10)                    Peak was 10 calls, after 1 s
  0 Running, 12 Paused, 7 Woken up
  85 dead call msg (discarded)           50 out-of-call msg (discarded)
  3 open sockets

                                 Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->         831       91        6
         100 <----------         815       0         0         0
         180 <----------         0         0         0         0
         183 <----------         0         0         0         0
         200 <----------  E-RTD1 815       6         0         0
         ACK ---------->         815       6
       Pause [    10.0s]         815                           4
         BYE ---------->         811       114       3
         200 <----------         807       0         0         1

------ [+|-|*|/]: Adjust rate ---- [q]: Soft exit ---- [p]: Pause traffic -----
```

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
10.0(10000 ms)/1.000s   5061     932.21 s          831  192.168.86.99:5060(UDP)

  0 new calls during 3.016 s period      3016 ms scheduler resolution
  10 calls (limit 10)                    Peak was 10 calls, after 1 s
  0 Running, 12 Paused, 6 Woken up
  85 dead call msg (discarded)           51 out-of-call msg (discarded)
  3 open sockets

                                 Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->         831       91        6
         100 <----------         815       0         0         0
         180 <----------         0         0         0         0
         183 <----------         0         0         0         0
         200 <----------  E-RTD1 815       6         0         0
         ACK ---------->         815       6
       Pause [    10.0s]         815                           4
         BYE ---------->         811       114       3
         200 <----------         807       0         0         1

------------------------------ Test Terminated --------------------------------


----------------------------- Statistics Screen ------- [1-9]: Change Screen --
  Start Time             | 2021-03-13   15:39:55.362727 1615667995.362727
  Last Reset Time        | 2021-03-13   15:55:24.560428 1615668924.560428
  Current Time           | 2021-03-13   15:55:30.663895 1615668930.663895
-------------------------+---------------------------+--------------------------
  Counter Name           | Periodic value            | Cumulative value
-------------------------+---------------------------+--------------------------
  Elapsed Time           | 00:00:06:103000           | 00:15:35:301000
  Call Rate              |    0.000 cps              |    0.888 cps
-------------------------+---------------------------+--------------------------
  Incoming call created  |        0                  |        0
  OutGoing call created  |        0                  |      831
  Total Call created     |                           |      831
  Current Call           |       10                  |
-------------------------+---------------------------+--------------------------
  Successful call        |        0                  |      807
  Failed call            |        0                  |       14
-------------------------+---------------------------+--------------------------
  Response Time 1        | 00:00:00:000000           | 00:00:00:057000
  Call Length            | 00:00:00:000000           | 00:00:10:690000
------------------------------ Test Terminated --------------------------------
```

Figure 15.

Table:

INVITE	831 messages sent
100	815 received messages
200	815 received messages
ACK	815 messages sent
BYE	811 messages sent
200	807 received messages

Results:

During the SIPp stress test, it executes 10 concurrent calls, each lasting 10s to extensions created to test account of 1002. The test included RTP as well, each of the calls out all came back which means there are no dropped calls, and the quality of the voice is ok. There were a couple calls that were timeout or not found, and only some of the calls were retransmitted.

According to the diagram, out of all those calls 14 of the calls failed, however 807 were successful. In Figure 15. It shows that there are 85 dead call messages that have been discarded as well as 51 out-of-call messages discarded. The table within the test shows how the VoIP calls are made with RTP added within.

8.

Asterisk Modules:

Asterisk uses modules, those modules are basically loadable components within the program that provides a specific functionality, as well as a resource that allows connection to an external technology.

Asterisk Extensions Language (AEL):

AEL is a specialized language intended only for Asterisk dial plans, it is a merger of 4 different ‘languages’/’syntaxes’, such as: the AEL syntax itself, Expression syntax, variable reference syntax, and extension language syntax.

Asterisk Gateway Interface (AGI):

AGI is a software interface for asterisk that allows external, user written programs to be launched from the asterisk dial plan.