Kamailio EsiNet
Multimedia Networking

Illinois Institute of Technology
Real-Time Communication Laboratory

Professor Carol Davids
Jason Cheng (A20439400), Rishon Dass (A20443042), Lina Kang,
Joshua Prila (A20436748)

December 2, 2022

Abstract

The report demonstrates how Kamailio servers can be used to implement NG911 services. It also unpacks the importance and rigor pertaining to NG911 services.Through various testing processes Team A was unsuccessful in rerouting the call to the SBC, while Team B was successful. The report shows the different approach each team took and the challenges that came with them.

Table of Contents

Introduction

This project aims to configure Kamailio as a proxy server. The Kamailio server can check for calls intended for emergency services (911) and route these calls to a Session Border Controller (SBC). This SBC will then route the call to its final destination at the Public Safety Access Point (PSAP) where it can be answered by emergency operators in the EsiNet.

In this experiment, an Emergency Call Routing Function (ECRF) will not be used and instead the Kamailio proxy server is trusted to handle routing by itself. Future iterations should note the location of callers to more effectively assist emergency responders and increase protection from DDOS/DOS attacks.

The SBC is held in the Real-Time Communication (RTC) Lab on the Mies campus of the Illinois Institute of Technology (IIT or Illinois Tech) in Chicago, Illinois. The PSAP used in place is monitored at Illinois Tech's Rice campus in Wheaton, Illinois.

Team A is made up of Jason Cheng, Lina Kang, and Joshua Prila. Team B has Rishon Dass and Amrutya LNU.

Kamailio Setup

For the Kamailio setup, group A and group B worked in parallel, opting for a slightly different approach in structure. The Kamailio configuration script was similar for both teams, but debugging and database control were handled differently.

Team A began preliminary research with Ubuntu manuals and Kamailio documentation. Team A installed Kamailio 5.7 on Linux Ubuntu 20.04. This is dual booted from a Windows machine, and used partitioned hard drive space. Team A chose to use MySQL as the database to hold user data. This was chosen because Kamailio's latest versions come with functions that will run MySQL for you, speeding up database configuration. Figure 1 depicts the original state of code for Team A. Team A iterated on these changes several times. First, the second "if" statement was changed from "sip@proxy1" to "sip@10.15.98.8" as the RTC Lab opened up that IP for Team A to use. Team A fixed an error in "t_relay" where the arguments of the function were not strings by changing them to strings. Team A's MySQL database also went through iterations to solve issues. The first database issue was a permissions issue, even with "sudo" commands. This issue was solved by Team A doing a complete reinstall of Kamailio and MySQL and then changing database permissions for the folders that hold these.

Team B also utilized resources to begin scripting. After watching tutorials and reading documentation. The Kamailio server was initially attempted to be installed on a windows 11 operating system. Upon several failures Team B decided to try on Mac OS Ventura and still had no success. Team B decided to use a VM Running Ubuntu 22.04.1 LTS on a Windows 11 host machine to download Kamailio 5.5 and MariaDB 10.6. Team B initially tried to run Kamailio version 5.7, which was the latest version. However, there were issues with setting up Ubuntu to successfully retrieve and install the files. Team B also opted to go with mariaDB instead of MySQL simply because mariaDB was an open source fork of MySQL, so they're very similar in functionality.

Lab Setup

The lab setup for Team A was to have Kamailio on one of the team members' machines, have XLITE on another team members machine, and have the second XLITE on the machine that was provided in the lab. The machine we had Kamailio on was set to the IP address of 10.15.98.8. We had two XLITE users, 2005 and 2004, that were both set to the domain of 10.15.98.8. All of the machines we used in this lab had to be connected to the 10.15.98.xx network through ethernet. These users were registered into the Kamailio MySQL user database using the command “kamctl add 200x 200x200x”. The parameters denote the username and passwords of the user. Once we added the users into the Kamailio MySQL database we were able to register both XLITES into the Kamailio.

Team B had a similar setup to Team A, however, the machine running the Kamailio server(Ubuntu VM) was set to IP address 10.15.98.88 and the host machine was set to 10.15.98.89 respectively. The setup to the database was also done using the same commands as Team A, but went with a different user login using 01 and 02 for the usernames and passwords. Once the users were successfully added we are ready to call and test the routing functions built into the Kamailio configuration.

Testing

Team A went through rigorous testing to view Wireshark traces and attempt to view Kamailio script responses to test calls. Figure 3, Figure 4, Figure 5, and Figure 6 show the XLITE users 2004 and 2005, account information.. Once both the XLITE users were registered to the Kamailio, we started testing our project. The first test consisted of doing vanilla calls back and forth between the 2005 and 2004 agents. The calls went through without any issue. Attached to the document are traces we took when we made the vanilla calls. Figure 7 shows the trace of the vanilla calls from Team A’s Kamailio server perspective

Our next test was to see if our script was able to route the 911 call to the SBC and then reroute it to the PSAP. In this test we were unsuccessful in doing so. We called 911 on our 2005 user and was unable to reach the SBC. On the XLITE machine (Figure 10) it stated “Call Failed: Media negotiation Failed.” At this point we were unsure if the problem was in the script or the SBC server. Figures 8 and 9 highlight two of the 911 tests from Team A’s Kamailio server perspective. Note that the INVITE SIP frames are still visibly being received.

Our next approach was to contact the SBC directly. We dialed 911@10.15.98.100 directly on our XLITE machine with the 2005 user. Figure 11 shows the XLITE machine sends out another error message stating “Call Failed: Not relaying.” Our next approach was to ping the SBC on the 10.15.98.xx network to see if the server was online. We were able to ping the SBC using the command “ping 10.15.98.100.” After this, Team A proceeded to attempt several reinstalls of Kamailio with versions 5.3 and 5.5. These had similar results to the original attempt of 5.7, but the debugger run of Kamailio was able to run successfully.

Our conclusion in our testing was that there was something wrong with our script and its relationship to the Operating System that was blocking our communication to the SBC. We began to run some more tests after changing the script. Our script was still not working and we decided to take a trace of the vanilla call to see if the trace would help us in figuring out what was wrong with our script. The vanilla calls stopped working and gave us the error message "Call Failed: Not Found." After some more testing we realized that every time we restarted the Kamailio we needed to re-register our XLITE users. We re-registered the users and were able to make the vanilla calls again. 911 calls were still not working and we still could not get through to the SBC. Additionally, Team A tried changing "$hu(To)" to "$tu" from collaborating assistance from Team B, who successfully routed an emergency call. Team A changed "t_relay" to "t_relay_to_udp" also after suggestions from Team B. These did not solve the problems Team A were facing. Unusually, the INVITE requests are still seen by the Kamailio server computer, and the vanilla calls did not work without the Kamailio server running. This leads us to believe that the Kamailio configuration was not integrated correctly in the given OS.

Team B's testing process was similar. The vanilla calls between user agents were tested with two different UAs on the same computer. This was done primarily because the other teammate wasn't able to get xlite running. Additionally, the test was simply to see if the regular kamailio server was functioning properly before we edited the configuration file. Seen in Figure 12 and Figure 13 we were able to successfully connect to the server and register the users using the software. We were also able to make a call between each other successfully. Then we started editing the configuration file and the code we added can be seen in Figure 2. That is the only change we made in the entire configuration file. After we tried rerouting the 911 call it was successful as seen in Figure 15.

Illustrations, Tables, and Figures

Figure 1: Old Team A Code

```
    # RTC LAB EsiNet Routing
    # Check for invite emergency call, then redirect to SBC
    # SBC send to PSAP
    if(method=="INVITE") # invite?
    {
        if($hdr(To)=="911@proxy1") # emergency call?
        {
            xlog("Emergency call detected: Routing to SBC!");
            #route to SBC:
            insert_hf("10.15.98.100; lr\r\n", "Route: header")
            # insert over append so that this is the first route header displayed

            if (!t_relay(10.15.98.100,5060)) #t_relay(host, port) returns negative number if failure # Alternatively, run with
send(host,port);
            { #t_relay_to_xxx exists, xxx replaced by tcp or udp
                # run if t_relay fails
                xlog("Failure in route to SBC");
                sl_reply_error();
                break;
            }
            else
            {
                # Redirect to PSAP here
                # PSAP Address?
                xlog("Successful route to SBC");
                #send("sip.911@proxy1")

            }
        }
    }
    # RTC Lab End
```

Figure 2: Latest Team B Code

```
543     if(is_method("INVITE")){
544     
545     xlog("SIP Request: method [$rm] from [$fu] to [$tu]\n");
546     
547     if($tu == "sip:911@10.15.98.88"){
548     
549         xlog("EMERGENCY: PPL ARE DYING!!!!!!!!!\n");
550     
551         if(t_relay_to_udp("10.15.98.100","5060")){
552             xlog("YOU DID IT!!!!!!");
553     
554     
555         }else{
556             xlog("It don't work, something is broken");
557         }
558     }
559     
560     
561     }
```

Figure 3: Account properties for 2005

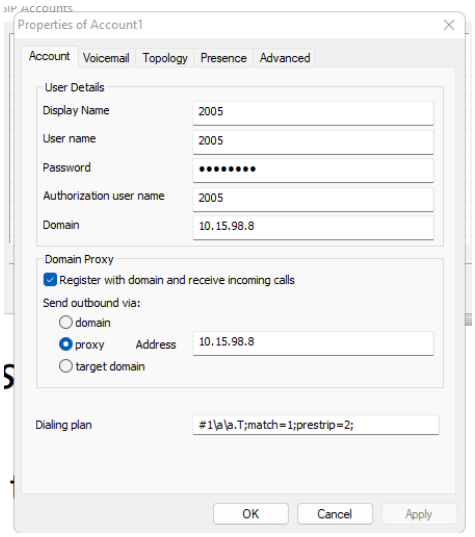


Figure 4: Account properties for 2004

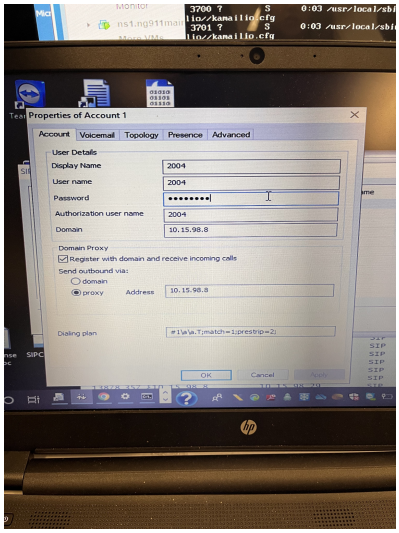


Figure 5: User 2005 registered

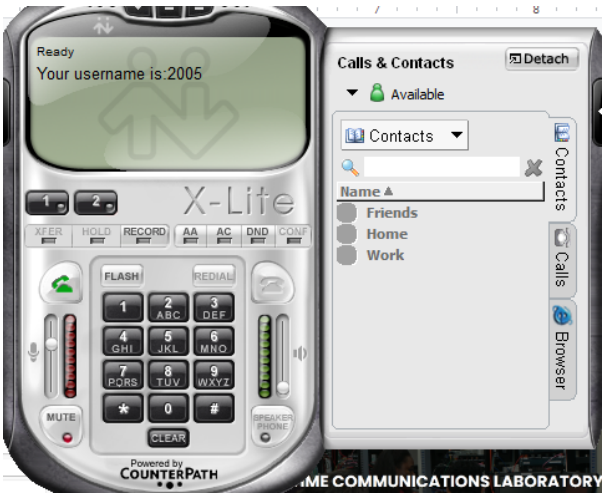


Figure 6: User 2004 registered

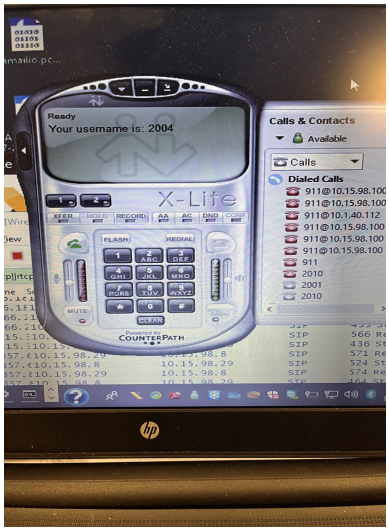


Figure 7: Vanilla Call trace (Team A)

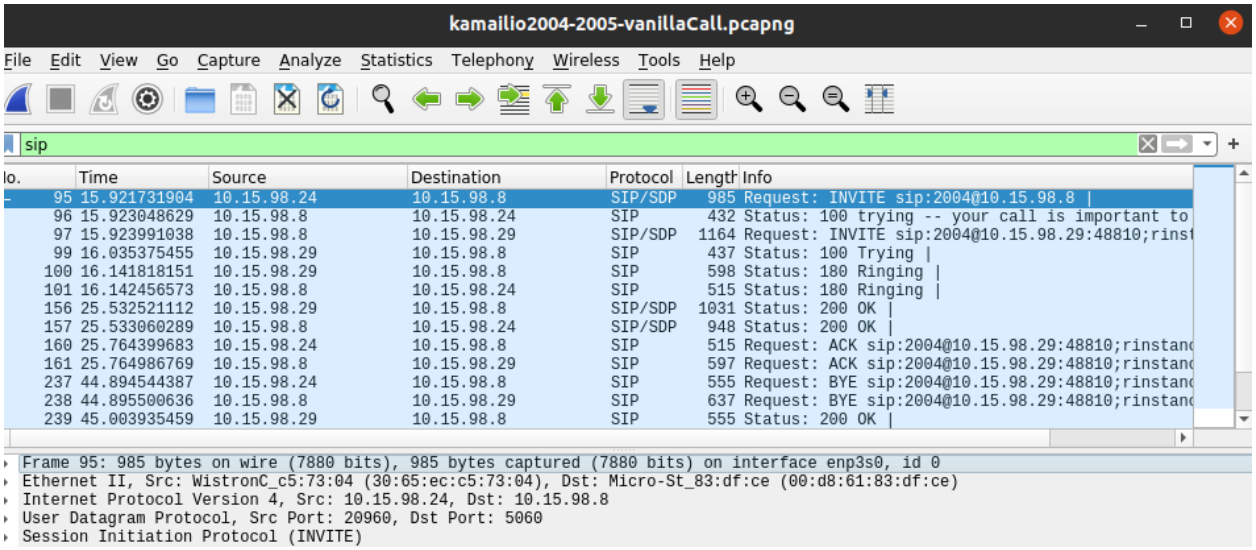


Figure 8: 911 Call Wireshark trace

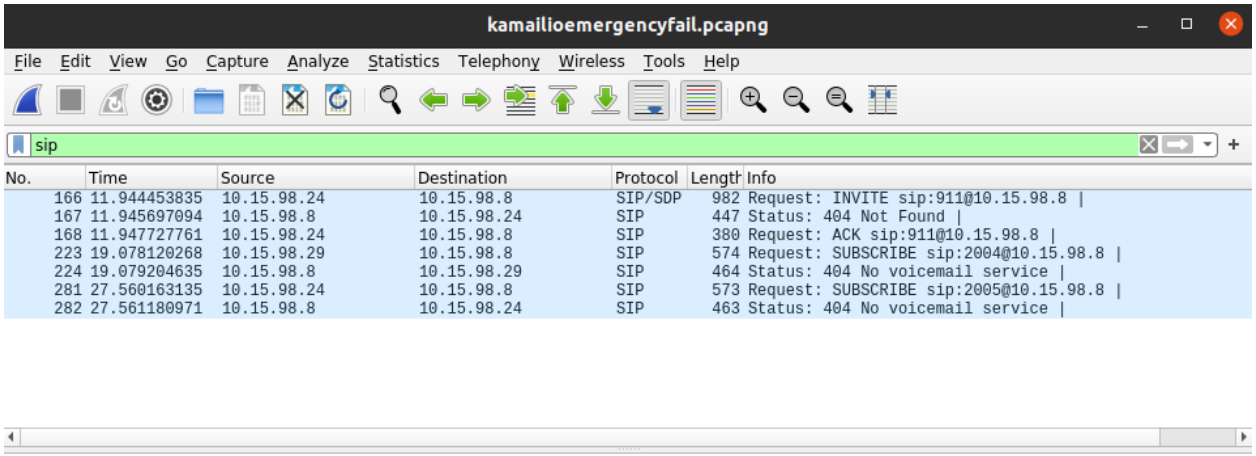


Figure 9: Second 911 Call Wireshark trace

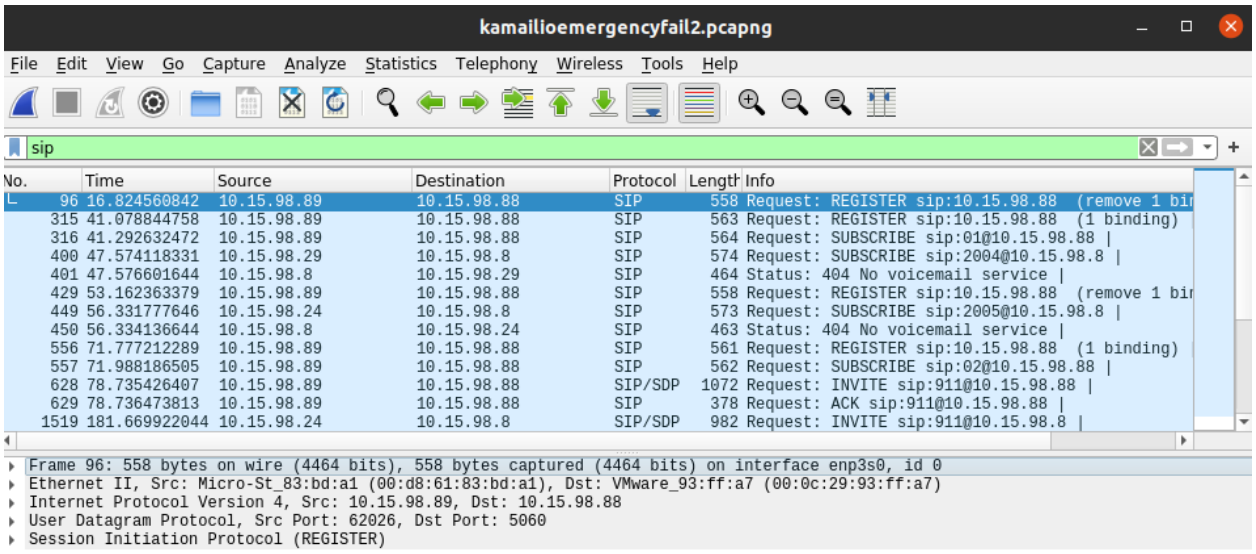


Figure 10: First 911 Call

Figure 11: Call straight to SBC

Figure 12: Team B UA1

Figure 13: Team B UA2

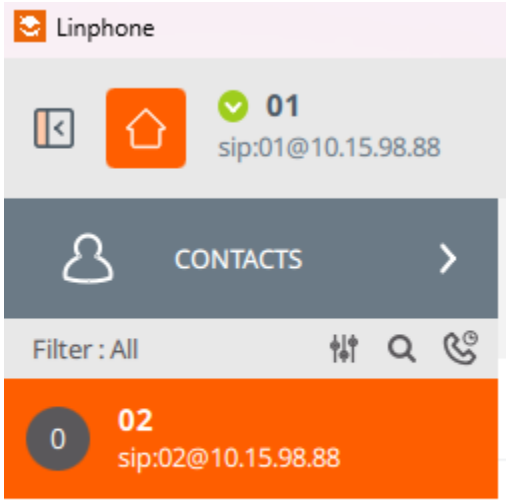
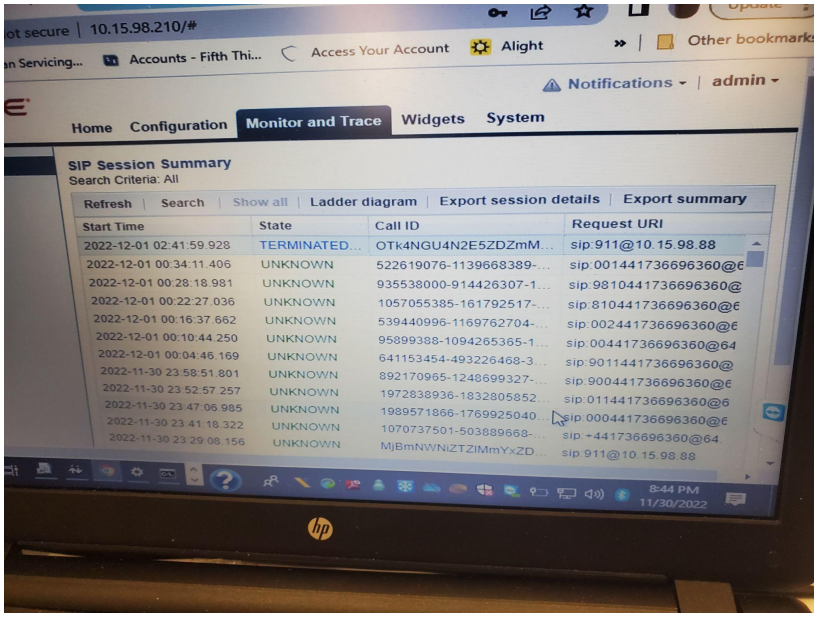


Figure 14: Team B successful reroute

Figure 15: Team B SBC log



Conclusion

A successful route to the SBC will allow the SBC to view the headers of the incoming call, and act as a final gate that blocks anything that may be heading the incorrect direction to the EsiNet. This Esinet represents a step towards Next Generation 911 Services (Stylized as NG911). Understanding NG911 starts by realizing that current emergency services run on outdated analog systems. Modern apps like food delivery, texting friends, and navigating the road take advantage of new IP-based telecommunications (e.g., VoiP) that older 911 systems are incapable of taking advantage of. NG911 seeks to replace the old analog 911 systems currently in place in the United States, as well as SOS systems set in place commonly around the world. Upgrading to NG911 will create an emergency calling system that is both stronger and faster than what we have today. This includes seamless voice, video, photo, and text communication to emergency services, as well as the ability to reduce the impact of call overload, and increase security of emergency responders. Implementing systems like the one tested here can save lives.

References

[1] C. Davids, V. K. Gurbani, S. Loreto and R. Subramanyan, "Next Generation 911: Where Are We? What Have We Learned? What Lies Ahead?," in IEEE Communications Magazine, vol. 55, no. 1, pp. 130-131, January 2017, doi: 10.1109/MCOM.2017.7823350.

[2] “Guidelines for generating a State NG911 Plan” in 911.gov, 2018; https://www.911.gov/assets/Guidelines_for_Developing_a_State_NG911_Plan.pdf

[3] NENA i3 Standard for Next Generation 9-1-1, NENA; January 2021;

[4] Vidal, Soto, Banchs, et. al. Multimedia Networking Technologies, Protocols, and Architectures;

[5]Nick, Post author By, et al. “Kamailio 101 – Part 1 – Introduction.” Nick vs Networking, 17 May 2021, https://nickvsnetworking.com/kamailio-introduction/.

[6]“Core Cookbook.” *Core Cookbook - Kamailio Wiki Documentation*, https://www.kamailio.org/wikidocs/cookbooks/5.5.x/core/.