ENSC 427: Communication Networks

Spring 2015

Final Project

Performance Analysis on VoIP over LTE network

Website: www.sfu.ca/~lchor/Ensc%20427/427Website

Group 7

Chor, Alexandrea	(Lexi)	lchor@sfu.ca
Dai, Wei	(Wei)	wda14@sfu.ca
Zhang, Chi	(Chi)	cza51@sfu.ca

Table of Contents

Acronyms

3G	**3rd Generation Telecommunication Network**
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
EBT	Enhanced Base Transceiver
EDGE	Enhanced
eNodeB	Evolved NodeB
EPC	Evolved Packet Core
GPRS	General Packet Radio Service
GTP	GPRS Tunneling Protocol
HSS	Home Subscription Station
IP	Internet Protocol
LTE	Long-Term Evolution
MME	Mobility Management Entity
PDN	Packet Data Network
PSTN	Packet-Switched Telephone Network
QoS	Quality of Service
RAN	Radio Access Network
SGW	Service Gateway
UE	User Equipment
VoIP	Voice over Internet Protocol

List of Figures

List of Tables

Abstract

Recent demand for higher data rates has quickly increased amongst mobile users. LTE provides an effective solution for this demand, as it has the ability to provide higher capacity and data rates, and reduces latency. For this reason, LTE has become the most popular wireless data communication network for mobile devices. Unfortunately, LTE only supports packet switching across an all IP network, therefore the traditional voice call solution of circuit switching is no longer available. To work around this dilemma, VoIP is introduced to enable voice calls over LTE networks. The objective of this project is to focus on the Quality of service (QoS) of VoIP over LTE networks. We will simulate multiple scenarios to measure and analyze its performance via measurement of delay, jitter and packet loss over a variety of distances and network loads.

Introduction

In recent years, we have witness the exponential growth of Smartphone's and mobile data usage. This increase in usage causes an explosive growth in internet traffic in mobile networks. According to the Cisco survey, the global mobile data traffic "is expected to grow to 24.3 exabytes per month by 2019"[1]. This is nearly a tenfold increase from 2014, making the compound annual growth rate to be 57% [1]. Voice calling over the internet has become one of the most used forms of communication when connecting internationally, either for family, friends or business calls. Since most people have easy access to the internet, Voice over internet protocol (VoIP) is a simple way to reduce the costly bills from long distance rates. However, VoIP services over mobile networks demand larger data rate requirements than the standard 3G network can provide due to the latency and low speeds. Long Term Evolution (LTE), has been developed to handle such large data rates and is quickly replacing the 3G network because of its ability to support a large number of users with minimal latency and increased speeds. Before switching to the latest technology, it is helpful to provide users with performance details such as jitter, delay and packet loss. Such analysis can be shown by simulating and evaluating VoIP applications over the LTE network. We will first explore the LTE network architecture and then simulate and compare scenarios of single and multiple users using VoIP over the LTE network to observe the effects on performance.

Background

Voice over Internet Protocol (VoIP)

Traditional voice calling uses packet-switched telephone networks (PSTN), which uses minutes to determine costs. Making international calls are more expensive which hinders long distance communication abilities. VoIP was developed to provide voice communication anywhere in the world that has access to the internet. VoIP uses internet as a backbone, as opposed to the telephone network, and compresses data packets while they are transmitted. Because of this compression, more data is able to be handled and more calls can be carried across the line. VoIP also enables multiple users to connect at the same time without much additional data, for this reason, it is a suitable way to have international business meetings. Studies have shown that around 50% of voice conversations are silent. To avoid wastage of bandwidth in data communication channels, VoIP fills these silent spaces with data which increases the efficiency [2]. A well-known example of VoIP is Skype and the number of users continue to exponentially increase, as shown in figure 1 [3].

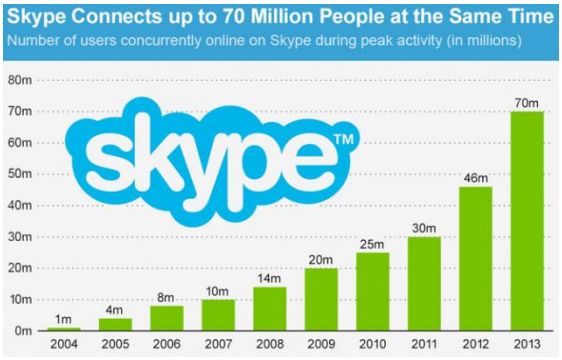

Figure 1: Skype User Statistics [3]

Long-Term Evolution (LTE)

Traditional 2G and 3G networks deliver voice over dedicated fixed bandwidth channels to the user equipment (UE). Mobile data is transferred to and from the internet using RAN timeslots either via general packet radio service (GPRS) or enhanced data rates for global evolution (EDGE). The problem with these methods arises from the data usage. As voice uses fixed bandwidth for digital voice channels, data bandwidth needs to fluctuate to accommodate for uploads and downloads. LTE uses radio waves as opposed to microwaves like 3G and WiMAX, which allows greater coverage. Another enhancement of LTE network is that it provides data rates up to 100 Mbps for downlinks and 50 Mbps for uplinks, and throughput of 50-144 Mbps [4].

LTE Architecture

The architecture of the LTE network is composed of 3 main parts. A user equipment (UE) is connected with the evolved universal terrestrial radio access network (E-UTRAN), which is then connected to the evolved packet core (EPC) then to the internet. The connections are shown below:

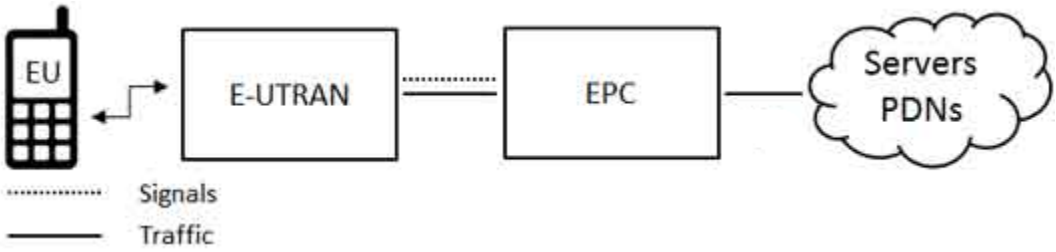

Figure 2: LTE Architecture Connection [5]

User Equipment (UE)

A UE is any device, such as a mobile phone, that is used directly by a user to communicate. It connects the evolved NodeB (eNodeB). The UE is dividing into five classes to define the performance specifications to ensure that the eNodeB can communicate correctly. Shown below are the different category parameters for UE's [6].

HEADLINE DATA RATES FOR LTE UE CATEGORIES					
	CATEGORY				
LINK	1	2	3	4	5
Downlink	10	50	100	150	300
Uplink	5	25	50	50	75

Table 1: Uplink and Downlink Data Rates for Each Category

Evolved Universal Terrestrial Radio Access Network (E-UTRAN)

The E-UTRAN connects each UE to the LTE network. It contains eNodeBs which can serve many cells in the LTE network. It sends and receives radio signals to the UEs, and performs certain tasks when a handover occurs. eNodeBs reduce the latency of all radio interface operations. They are interconnected together via the X2 interface as well as connect to the EPC by the S1 interface. The E-UTRAN architecture is shown below:

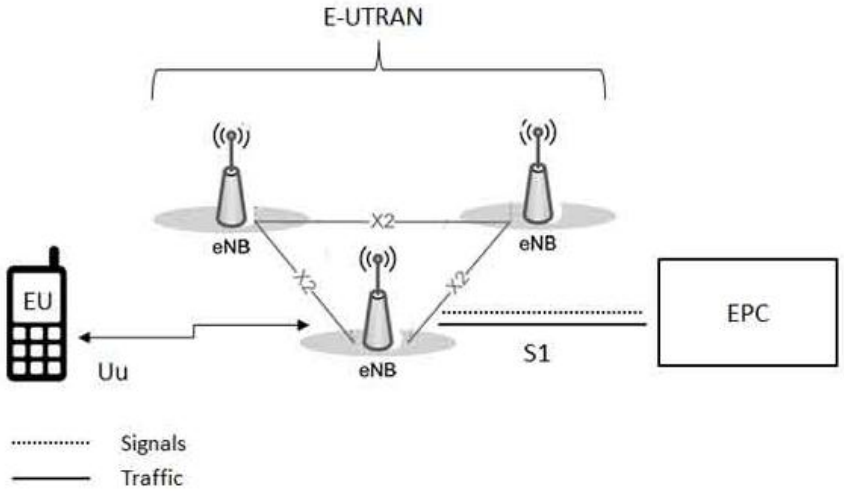

Figure 3: E-UTRAN Architecture [5]

Evolved Packet Core (EPC)

The EPC consists of several entities, amongst which are five important nodes: Serving Gateway (SGW), Packet Data Network (PDN), Home Subscriber Station (HSS), and Mobility Management Entity (MME). The SGW serves as a router which directs data packets between users and eNodeBs. It also acts as an anchor during the inter-eNodeB handover. The PDN connects the LTE network to the internet. It controls services such as packet filtering, traffic shaping and user charging policies, and is responsible for establishing, maintaining and deleting GPRS tunneling protocol (GTP) tunnels to the SGW. The HHS is a database that contains user information, such as user identification, user profile and authentication. And finally, MME manages the UE contexts such as UE identity, mobile state and security parameters. The architecture of EPC is shown below:

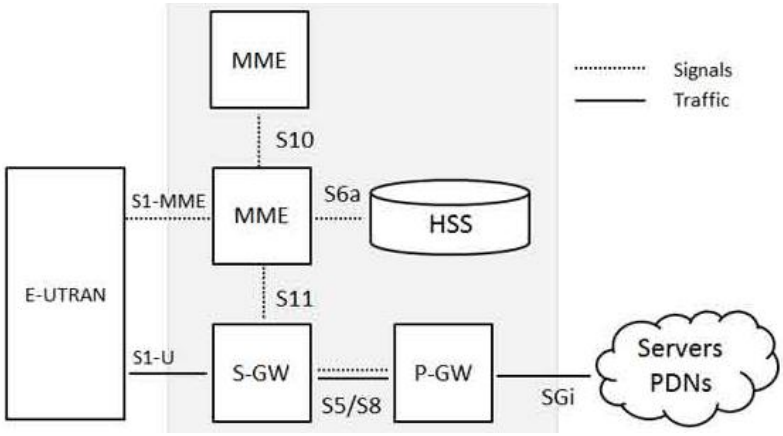

Figure 4: EPC Architecture [5]

Network Simulation

Network Topology

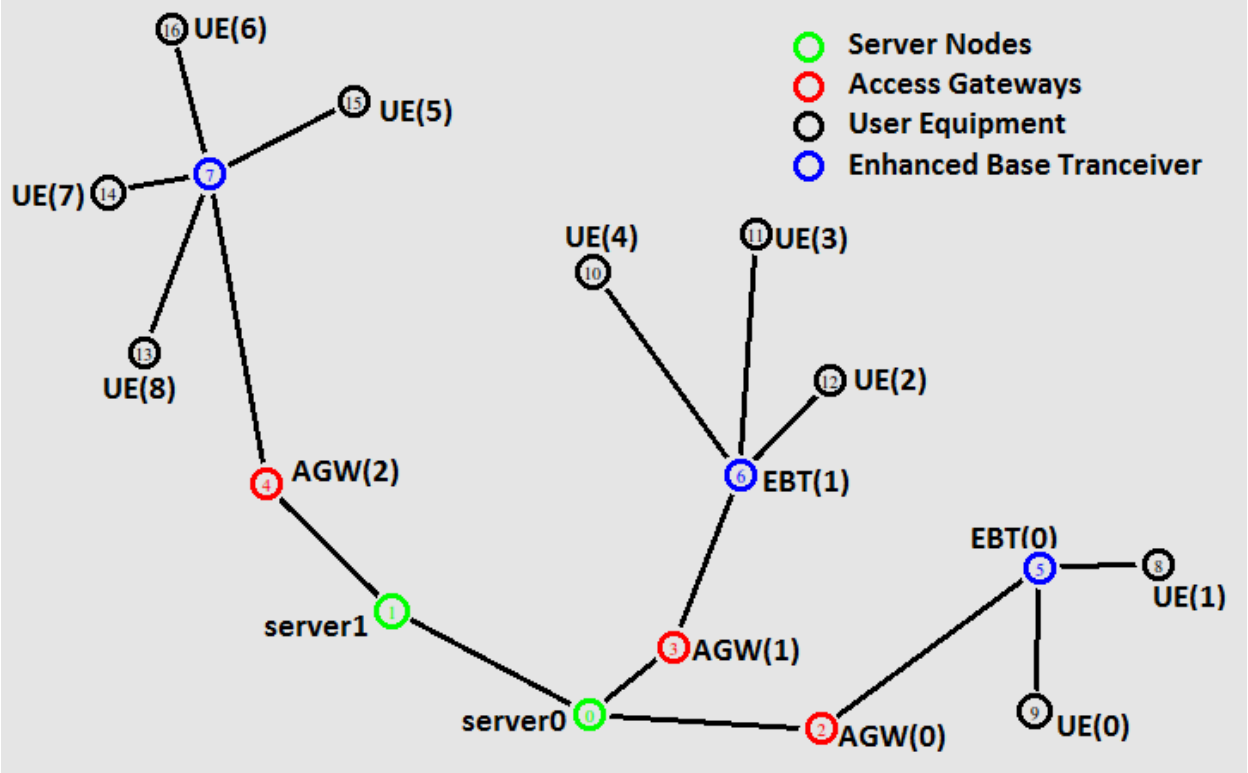

Figure 5: LTE Network Topology

The topology for the LTE network is shown in figure 5. The model consists of 2 servers, 3 access gateways (AGWs), 3 enhanced base transceivers (EBTs) and 9 user equipments (UEs). On server0, there are two AGWs. Each AGW has an EBT. We placed two users on EBT(0) and three on EBT(1). On server 1, there is one AGW and one EBT with four users. The role of each component is stated below:

Servers - manage the incoming and outgoing data and perform the necessary responses.

AGWs - facilitate the transfer of data so that the data can reach the servers.

EBTs - provide signals that mobile devices can pick up so that they can exchange data with the internet. For LTE, the specific EBT is called eNodeB.

User Equipment - any devices that can create communication between users, in our case, mobile phones.

AGW and Server setup

Since each connection is bidirectional, we use duplex links to connect each component. The AGW and server connection codes are shown below. Two servers are delayed by 100ms, which signifies a long distance connection, whereas the AGWs are delayed by 10ms.

```
#connect 2 server nodes
$ns duplex-link $server0 $server1 100Gb 100ms DropTail
```


```
#connect Access Gateway nodes to server nodes and set orietation
$ns duplex-link $AGW(0) $server0 10Gb 50ms DropTail
$ns duplex-link-op $AGW(0) $server0 orient right-up
$ns duplex-link $AGW(1) $server0 10Gb 50ms DropTail
$ns duplex-link-op $AGW(1) $server0 orient left
$ns duplex-link $AGW(2) $server1 10Gb 50ms DropTail
$ns duplex-link-op $AGW(2) $server1 orient right-down
```

ns2 LTE Connections

In order to use the LTE module on ns2, we had to install it using the steps on S. Naveen's Blog as a guide [7]. With a few modifications, we were able to successfully run the LTE module on ns2. The modified set up of S. Naveen's Blog steps can be found in Appendix A. Once this was set up, we could continue following S. Naveen's Blog guide to set up the connections between AGWs and EBTs, and the wireless connection between UEs. DLAirQueue and ULAirQueue are the downlink and uplink of the wireless connections, and the ULS1Queue and ULS1Queue are the uplink and downlink of wired connections.

```
#connect enhanced base transceiver nodes to Access Gateway nodes with separate uplinks and
downlinks
for {set i 0} {$i < 3} {incr i} {
$ns simplex-link $EBT($i) $AGW($i) 5Gb 10ms LTEQueue/ULS1Queue
$ns simplex-link $AGW($i) $EBT($i) 5Gb 10ms LTEQueue/DLS1Queue
}

#connect user ends to enhanced base transceiver nodes with separate uplinks and downlinks
for {set i 0} {$i < 2} {incr i} {
$ns simplex-link $UE($i) $EBT(0) 500Mb 2ms LTEQueue/ULAirQueue
$ns simplex-link $EBT(0) $UE($i) 1Gb 2ms LTEQueue/DLAirQueue
}
for {set i 2} {$i < 5} {incr i} {
$ns simplex-link $UE($i) $EBT(1) 500Mb 2ms LTEQueue/ULAirQueue
$ns simplex-link $EBT(1) $UE($i) 1Gb 2ms LTEQueue/DLAirQueue
}
for {set i 5} {$i < 9} {incr i} {
$ns simplex-link $UE($i) $EBT(2) 500Mb 2ms LTEQueue/ULAirQueue
$ns simplex-link $EBT(2) $UE($i) 1Gb 2ms LTEQueue/DLAirQueue
}
```

VoIP setup

To set up VoIP, we have to define the packet size and the interval between the send times. As per Cisco definitions [8], the average packet size is 480 bytes, and the interval time should be 0.03ms.

```
$cbrg($i) set packetSize_ 480
$cbrg($i) set interval_ 0.03
```

Scenarios

We will conduct two different scenarios to analyze the performance of VoIP over LTE. First we will simulate a one-to-one user scenario, then add multiple users to see how the performance is affected.

One-to-One

For the one-to-one user scenario, we will have two user pairs. The first pair will be a local VoIP call between two user nodes that belong to the same server. This event will be between UE(0) and UE(2), who are each connected to server0 via different AGWs. This connection is shown in blue in figure 6.

The second scenario will be a long distance VoIP call between the two user nodes on different servers. This event is between UE(3) and UE(5) where UE(2) is connected to server 0 and UE(5) is connected to server 1. This connection is shown in red in figure 6.

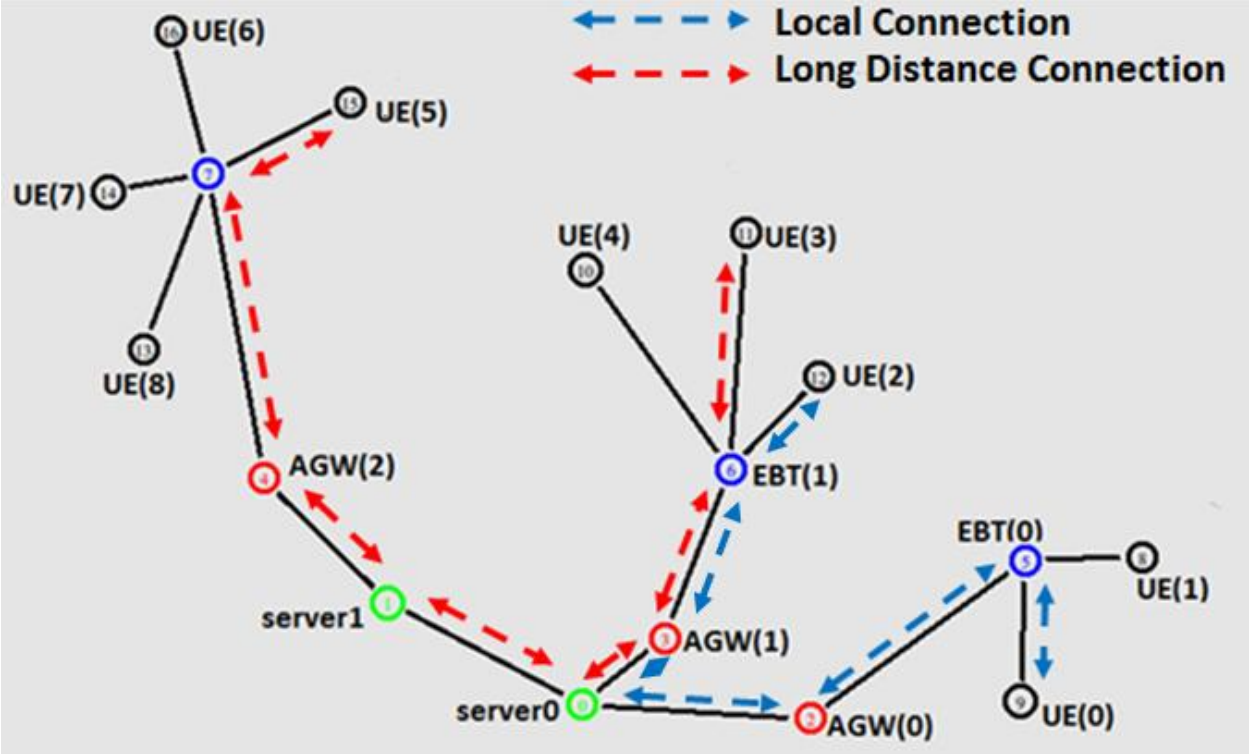

Figure 6: One-to-One Connection Topology

Multiple users

For the multiple user scenario, we will implement a group chat between 4 users: UE(1), UE(4), UE(6), and UE(7). This connection is shown in figure 7 and is a mix of local and long distance users.

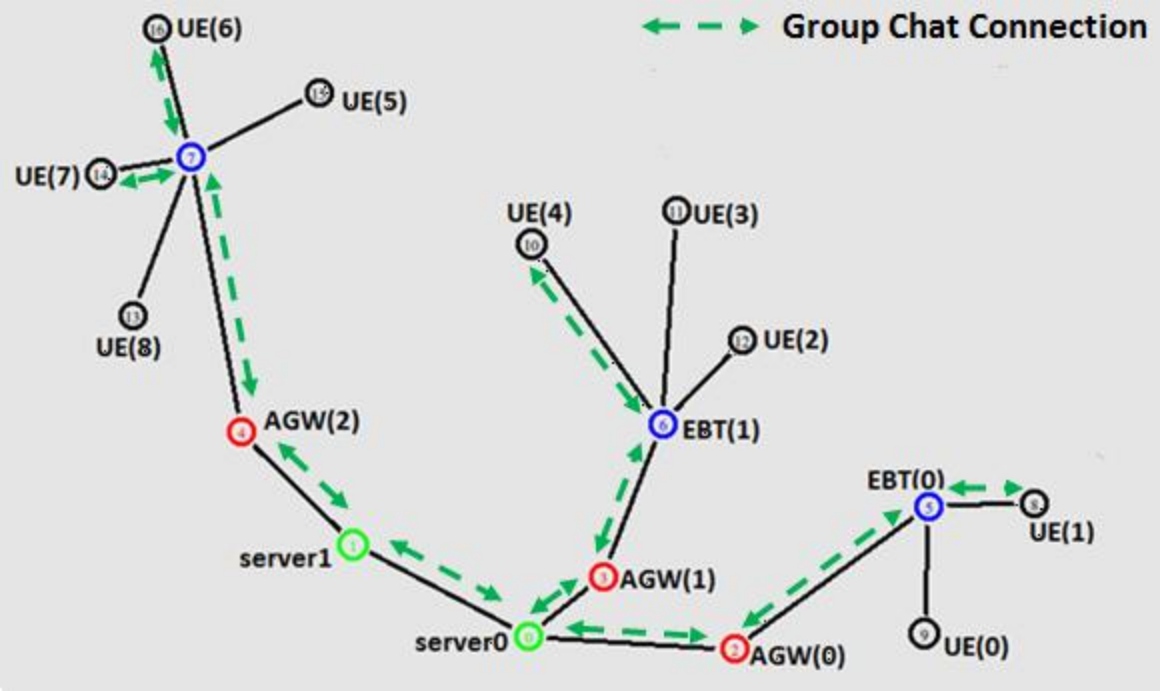

Figure 7: Group Chat Connection Topology

Parameters

VoIP QoS is determined by many factors such as jitter, throughput, end-to-end delay and packet loss. We will be comparing these factors for each VoIP usage scenario. The network quality parameters are summarized below in table 2.

Network Parameter	Good	Acceptable	Poor
End-to-End Delay (ms)	0-150	150-300	>300
Jitter (ms)	0-20	20-50	>50
Packet Loss	0-0.5%	0.5%-1.5%	>1.5%
Throughput (Mbps)	0-50	50-144	>144

Table 2: VoIP Quality Parameter Measures [9]

Jitter

When packets are sent and received, there is an end-to-end delay variation between the packets. This time difference is called jitter. Jitter can be caused by many factors such as congestion in the network, jitter buffer overload, timing drifting or route changes. Ideally we want a jitter of 0ms, but for high quality VoIP calls, a jitter of less than 50ms is acceptable [10]. Table 1 summarizes the VoIP quality parameters. Jitter is an important parameter to analyze because high jitter values can lead to poor voice quality, therefore we want to see if an LTE network can minimize this value. Some methods to lower the jitter levels include adding anti-jitter circuits, buffers and filters. Jitter is calculated with the following formula:

$$Jitter = (T4 - T3) - (T2 - T1)$$

Where T1 and T2 are the time of leaving two consecutive packets from the source node, and T3 and T4 are the time of arrival of those packets to the destination node.

End-to-End Delay

The end-to-end delay is the time it takes for a package to get from source to destination. Delay can occur because of network delays, delays at the source or destination, propagation delays and encoding/decoding delays. A delay of up to 300ms is considered acceptable. This parameter is important to analyze because we don't want any delay between the outgoing voice message and the received voice message. We will test if an increase in users affects the delay of the voice message.

Packet Loss

Packet loss is the percentage of packets that are lost during transmission from source to destination over the network. Ideally, there should be no packet loss however, a percentage of up to 1.5% is acceptable. The equation to calculate packet loss is as follows:

$$packet\ loss = \frac{packets\ sent - packets\ recieved}{packets\ sent} \times 100\%$$

Throughput

Throughput is the amount of data that can successfully be transferred from the source to destination in a given amount of time. The unit for throughput is bits per seconds.

Results

In our project, we simulated VoIP with CBR and UDP agent. All of data collections were accomplished using LossMoniter Class from NS2. The LossMonitor class is C++ class which enabled us to collect last packets arrival time, number of packets, number of loss packets and the number of bytes by accessing objects like LastPktTime_, npkts_, nlost_, and bytes_. By attaching LossMonitor class to sinks at each node, the throughput, packet loss, delay and jitter of our simulations were easy to compute from last packets arrival time, number of packets, number of loss packets and the number of bytes.

Throughput

In our case, we manipulated bytes_ to compute throughput. In our algorithm, we stored previous total bytes received in holdrate(i) and current bytes received in bwB(i), and in each iteration, thus, bwB(i) and holdrate(i) allows us to calculate data received in each sample time by subtracting holdrate(i) from bwB(i). We multiply this by 8 to convert units from bytes to bits, and divide by 2 sample times for the two intervals of time that the record has ran, and finally convert unit from bits/s to megabits/s by dividing by 1000000.

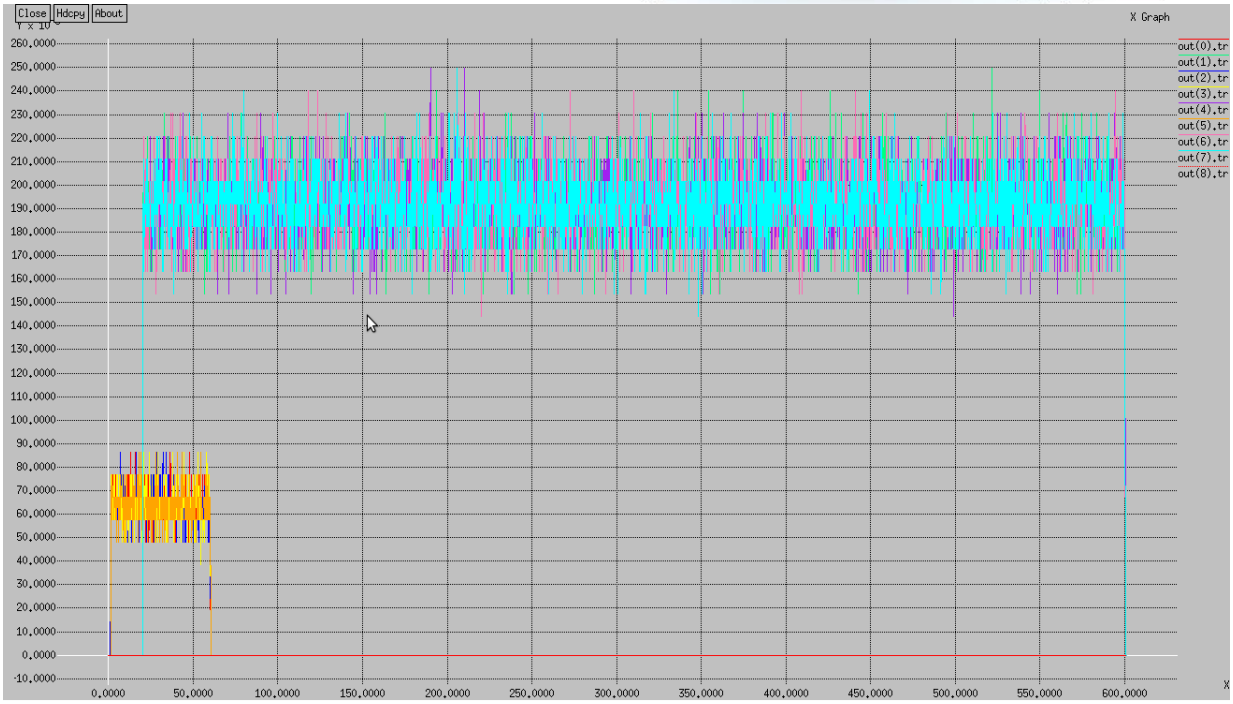

Figure 8: Throughput Simulation Results

The simulation graph shows different throughput levels on both one-to-one call and group chat. The simulation is consistent with theories as group chat has introduced 3 times the traffic than two one-to-one calls, and the overlaps of both scenarios also indicate that there are no effects from each scenario on throughput.

Packet Loss

The packet loss rate is easy to compute as we already had nlost_ which measures total amount of packet loss. Like we did in the throughput computation, we declared an intermediate variable bwN to store the number of packet loss in each sample time interval. We simply compute it by dividing the sample time by bwN , then clear it at the end of each iteration computation.

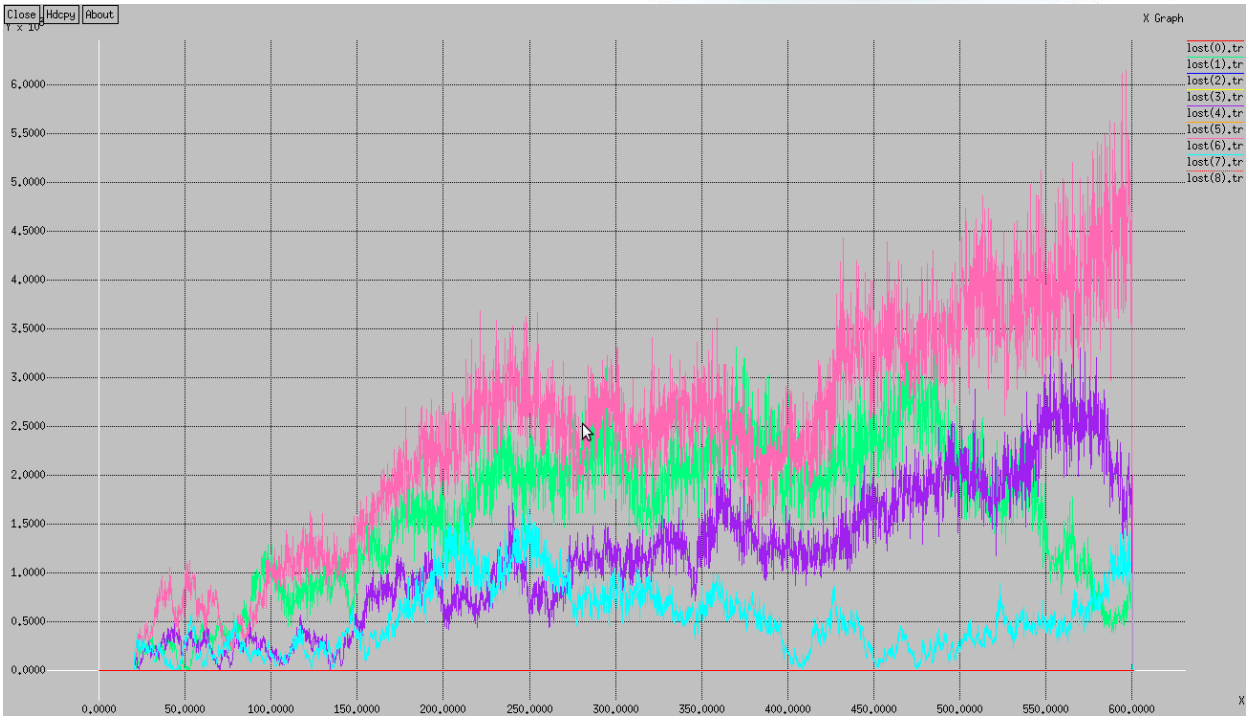

Figure 9: Packet Loss Simulation Results

From the simulation graph, a significant packet loss appeared when group chat was established. The results are consistent with theory as the group chat scenarios has introduced extreme amount of traffic into LTE network. And the overlaps of both scenarios also indicate that there are no effects from each scenario on packet loss rate.

End-to-End Delay

As we can only access those objects in LossMoniter in intervals of time, we can only compute the average delay for each packet in each time intervals. Similar to other algorithms, we used intermediate variables bwNPK, holdseq, bwLPKT and Hold time, to store current number of packets received previous number of packets received, current last packet arrival time and previous last packet arrival time. In our algorithm, we subtract last packet arrival time(holdtime) from the current last packet arrival time (bwLPKT) to obtain the time elapsed to wait for last packet to come, and divided the difference between current number of packets received (bwNPK) and the previous number of packets received (holdseq) to obtain the end to end delay. In addition, we also introduced an exceptional case where current number of packets received (bwNPK) is equal to the previous number of packets received (holdseq), in other words, there are no packet receiving after last time interval. In order to consider both cases, we used if statements in the end to end delay algorithm.

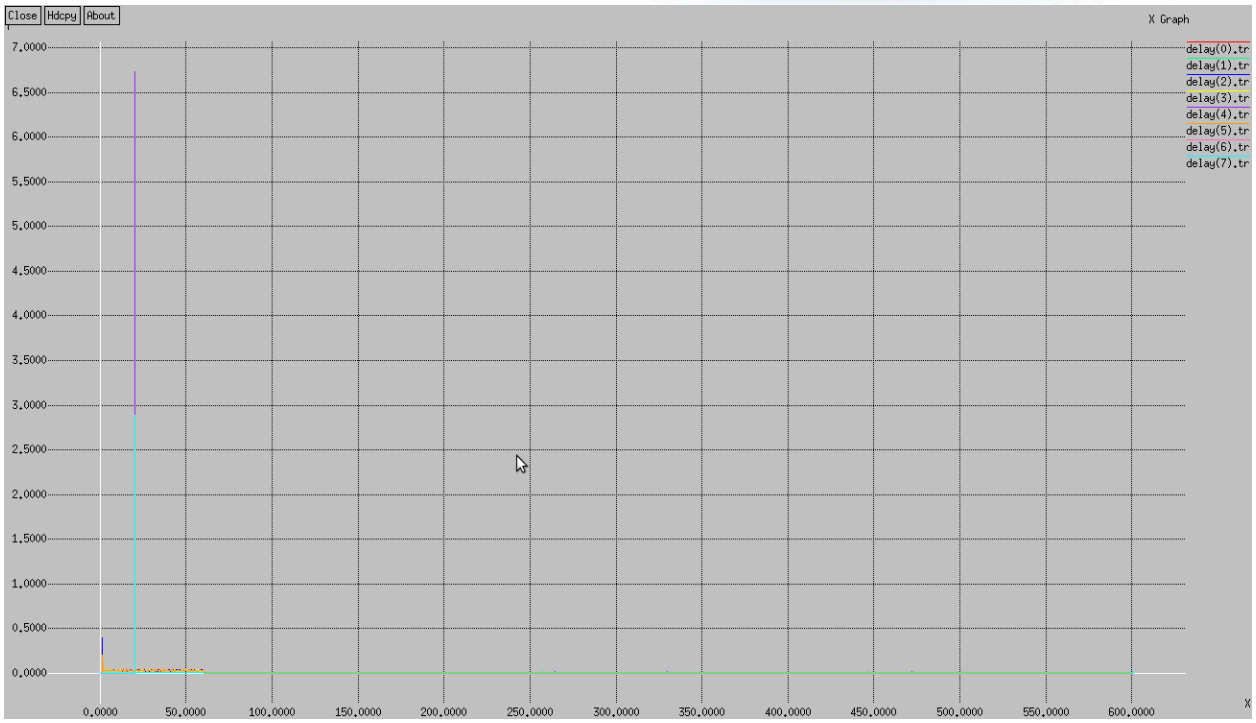

Figure 10: End-to-End Delay Simulation Results

As shown in the simulation graph, the delays were quite stable and tolerable. The spikes at beginning of each scenario are from the contention window adjustment. And the overlaps of both scenarios also indicate that there are no effects from each scenario on delay.

Jitter

As jitter is simply manipulating delay file, we used row operation in excel to calculate jitter and plot jitter graph.

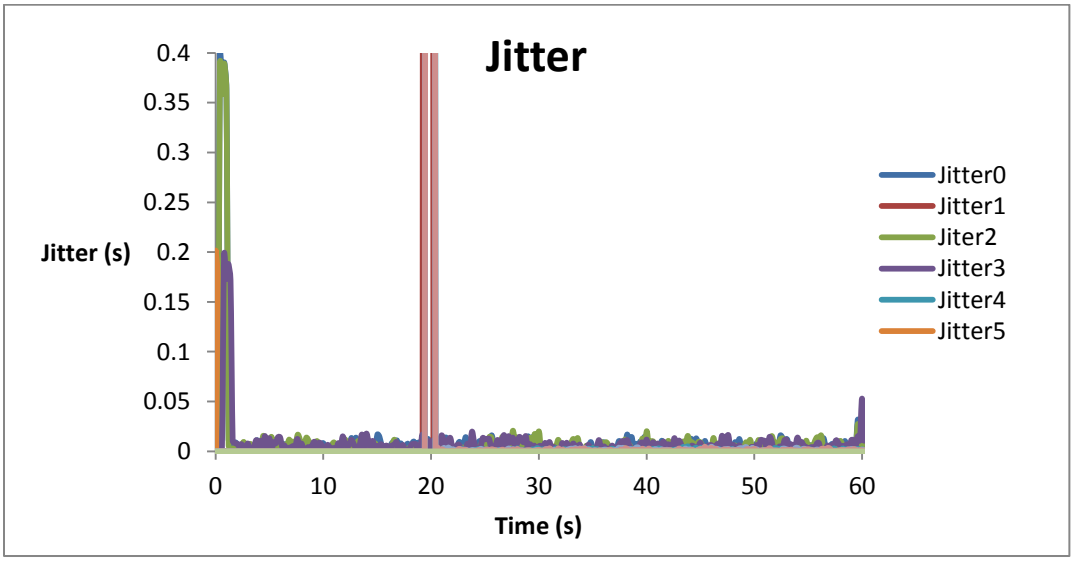

Figure 11: Jitter Simulation Results

Similar to delay graph, jitter also exhibits several spikes at beginning of each service due to contention window adjustment, and jitters at other time was stable and tolerable. And the overlaps of both scenarios also indicate that there are no effects from each scenario on packet loss rate.

Discussion

Difficulties

Since we initially wanted to implement simulation on Riverbed Modeler, we had designed complex scenarios with mobile nodes and jammers. Unfortunately, due to the licence problem of LTE modules on Riverbed Modeler, we had to switch and implement the simulation scenarios in Ns2 with less than 2 week. However, due to the nature of open sourced software, we have experienced lots of difficulties with the installation of the Ns2 LTE module. Since the LTE module we found was only compatible with NS-2.33, and NS -2.33, and was released years ago, we spent a lot of time modifying and troubleshooting the source code from NS-2.33. For example, the gcc command to build a shared library was changed from ld-shared to gcc-shared. The installation of LTE modules was very time consuming for troubleshooting problems from itself and environment.

Finally, switching from graphical simulation tool to a tcl script based tool also introduced extra work on implementation. In addition to implementation, it also brought barriers on data processing and calculation. Even though LossMonitor enabled us to collect and analyze data on each node, it does not provide an all-in-one data analyze feature like Riverbed Modeler.

Future Work

Although we have overcome difficulties on switching tools, and successfully simulated VoIP on LTE network for both one-to-one call and group chat, there are still lots of areas that can be improved in the future. First and foremost, compared to our stationary node, more realistic scenarios can be implemented with mobile nodes to represented that VoIP user are using mobile device during movements such as walking and in a vehicle. In addition, a multicast can be used to construct a group chat instead of setup each individual calls. But we do not have sufficient time to explore and implement multicasting. Finally, a hybrid transport layer protocols such as STCP and DCCP can be introduced to reach a better performance.

Conclusion

LTE network is rapidly replacing our usual mobile networks. Because of this, simulation to view performance of this phenomenon is highly attractive for a communication network project. We successfully simulated the two scenarios on ns2 using the LTE module and collected data from each scenario. According to the QoS criteria, the end to end delay and jitter fall in good range except for some spikes from beginning of each scenario due to contention of windows. However, the packet loss is at a poor region. Since packet loss doesn't meet Qos standards VOIP over LTE may not be a good service. In other words, some modification needs to be done on VoIP to make it a reliable voice solution for LTE,

and recently, an evolved version of VoIP: VoLTE has been introduced by 3GPP as standard voice solution on LTE network.

References

[1] Cisco, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017”. Available:http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white paper c11-520862.html

[2] N. Unuth, “Reasons for choosing VoIP”, 2015, Available: http://voip.about.com/od/voipbasics/a/ReasonsForVoIP_2.htm

[3] F. Richter, Aug 30 2013. “Skype Connects up to 70 Million People at the Same Time”, Available: http://www.statista.com/chart/1417/skype-usage/

[4] K. Andersson et al. “Mobile VoIP User Experience in LTE”, Local Computer Networks (LCN), 2011 IEEE 36th Conference, pp785 – 788, 2011.

[5] TutorialsPoint, “LTE Tutorial”, 2015, Available: http://www.tutorialspoint.com/lte/lte_network_architecture.htm

[6] Radio-Electronics, “LTE UE Category & Class Definitions”, Available: http://www.radio-electronics.com/info/cellulartelecomms/lte-long-term-evolution/ue-category-categories-classes.php

[7] S. Naveen, “LTE (Long Term Evaluation) Network in NS2“, Feb 19 2014, Available: http://naveenshanmugam.blogspot.ca/2014/02/lte-long-term-evaluation-network-in-ns2.html

[8] Cisco, “Cisco Catalyst 4500E Series Switches: Enabling Voice“, 2013

[9] J. Yu, I. Al-Ajarmeh. “Call Admission Control and Traffic Engineering of VoIP”. In *Digital Telecommunication,* July 2007.

[10] Y. Kabyd et al. “Performance Evaluation for VoIP by using G.711 as a Codec”. In *International Journal for Engineering Research and Technology*, Vol. 3, Issue 10, Oct 2014, pp. 758-763.

[11] G. A. Abed and M. Ismail and K. Jumari, "A Realistic Model and Simulation Parameters of LTE-Advanced Networks," Fac. Eng. & Built Env., National University of Malaysia, Selangor, Rep. ISSN:2278-1021, Aug. 2012.

[12] S. Sesia, I. Toufik, and M. Baker, "LTE - The UMTS Long Term Evolution - From Theory to Practice", Second Edition including Release 10 for LTE-Advanced, John Wiley & Sons, 2011.

[13] L. Chu, "Implementation and Application of VoIP Networks”, IEEE AIMSEC Conference, pp2139-2141, 2011.

[14] 4G LTE-Advanced Techonology Overview (n.d.) http://www.home.agilent.com/agilent/editorial.jspx?ckey=1905163&id=1905163%22&lc=eng&cc=IN.

[15] Long Term Evolution Overview. (2010, October). [Online].Available: http://www.freescale.com/files/wireless_comm/doc/white_paper/LTEPTCLOVWWP.pdf.

Appendix A

Operating System:

As NS-2 does not fully grant support for 64 bits system, a 32 bits Linux is desired, and in our case, we used Ubuntu 12.10 LTS 32bits.

NS-2.33 Installation:

As LTE modules was developed on NS-2.33, and NS -2.33 was released years ago. So some essential modification is required to install NS-2.33 on Ubuntu 12.03.

- Open terminal or Ubuntu Software center to install build-essential, tcl8.4 tcl8.4-dev tk8.4-dev, libxmu-dev, nam and xgraph.
- Download tk-8.4-lastevent.patch to /ns-allinone-2.33/tk8.4.18
- Download Download ns-allinnone-2.33 from http://sourceforge.net/projects/nsnam/files/ns-2
- Extract tar.gz in /home/xx, where xx refers to your username
- In line number 6304 from otcl-1.13/configure, change SHLIB_LD="ld -shared" to SHLIB_LD="gcc -shared"
- In line 219 from ns-2.33/tools/ranvar.cc, change return GammaRandomVariable::GammaRandomVariable(1.0 + alpha_, beta_).value() * pow (u, 1.0 / alpha_); to return GammaRandomVariable(1.0 + alpha_, beta_).value() * pow (u, 1.0 / alpha_);
- In line 65 from mac/mac-802_Ext.h, add #include<cstddef>
- From ns-2.33/mobile/nakagami.cc,
  Change if (int_m == m) {
  resultPower = ErlangRandomVariable::ErlangRandomVariable(Pr/m, int_m).value();
  } else {
  resultPower = GammaRandomVariable::GammaRandomVariable(m, Pr/m).value();
  }
  return resultPower; }
  to
  if (int_m == m) {
  resultPower = ErlangRandomVariable(Pr/m, int_m).value();
  } else {
  resultPower = GammaRandomVariable(m, Pr/m).value();
  }
  return resultPower;}
- Open Terminal , change directory to /ns-allinone-2.33, and type ./install to install NS-2.33 allinone
- Follow these prompts after installation to add environments paths to .bashrc
- Use example.tcl from NS-2 tutorials to test if NS, NAM and Xgraph is successfully installed

LTE Module:

After successfully installed NS-2.33, follow the following procedures to install LTE module on NS-2.33

- sudo cd ns-allinone-2.33/ns-2.33 && make clean && mv Makefile Makefile.org
- sudo svn checkout http://lte-model.googlecode.com/svn/trunk/ lte-model-read-only
- sudo mkdir project
- cd lte-model-read-only/
- sh checkin
- cd ../
- sudo gedit Makefile , and edit the new Makefile , lines 41, 67, 82 to actual location.

- Sudo make

Appendix B

NS2 LTE Code:

```
set ns [new Simulator -multicast on]
#Set Traces for Throughout

for { set i 0} {$i < 9} {incr i} {
#Set Traces for Throughout
set t($i) [open out($i).tr w]
#Set Traces for Packert Loss
set l($i) [open lost($i).tr w]
#Set Traces for end to end delay
set d($i) [open delay($i).tr w]
}

set f [open out.tr w]
$ns trace-all $f
set nf [open out.nam w]
$ns namtrace-all $nf

$ns color 1 Yellow
$ns color 2 Green
$ns color 3 Blue
#####LTE Topology##########
# The LTE topology for this simulation consist 2 server, 3 Access Gateway, 3 enhanced base transceiver and 9 user ends

#Create server nodes 0 and 1
set server0 [$ns node]
$server0 label "server0"
$server0 color green

set server1 [$ns node]
$server1 label "server1"
$server1 color green

#Create Access Gateway nodes

for { set i 0} {$i < 3} {incr i} {
set AGW($i) [$ns node]
```


```
$AGW($i) label "AGW($i)"
$AGW($i) color red
}

#create enhanced base transceiver nodes
for { set i 0} {$i < 3} {incr i} {
set EBT($i) [$ns node]
$EBT($i) label "EBT($i)"
$EBT($i) color blue
}

#Create User end nodes

for { set i 0} {$i < 9} {incr i} {
set UE($i) [$ns node]
$UE($i) label "UEvoice($i)"
$UE($i) color black
}
#####LTE node link##########
#connect 2 server nodes
$ns duplex-link $server0 $server1 100Gb 100ms DropTail

#connect Access Gateway nodes to server nodes and set orietation
$ns duplex-link $AGW(0) $server0 10Gb 50ms DropTail
$ns duplex-link-op $AGW(0) $server0 orient right-up
$ns duplex-link $AGW(1) $server0 10Gb 50ms DropTail
$ns duplex-link-op $AGW(1) $server0 orient left
$ns duplex-link $AGW(2) $server1 10Gb 50ms DropTail
$ns duplex-link-op $AGW(2) $server1 orient right-down

#connect enhanced base transceiver nodes to Access Gateway nodes with seperate uplinks and
downlinks
for {set i 0} {$i < 3} {incr i} {
$ns simplex-link $EBT($i) $AGW($i) 5Gb 10ms LTEQueue/ULS1Queue
$ns simplex-link $AGW($i) $EBT($i) 5Gb 10ms LTEQueue/DLS1Queue
}

#connect user ends to enhanced base transceiver nodes with seperate uplinks and downlinks
for {set i 0} {$i < 2} {incr i} {
$ns simplex-link $UE($i) $EBT(0) 500Mb 2ms LTEQueue/ULAirQueue
$ns simplex-link $EBT(0) $UE($i) 1Gb 2ms LTEQueue/DLAirQueue
}
```


```
for {set i 2} {$i < 5} {incr i} {
$ns simplex-link $UE($i) $EBT(1) 500Mb 2ms LTEQueue/ULAirQueue
$ns simplex-link $EBT(1) $UE($i) 1Gb 2ms LTEQueue/DLAirQueue
}
for {set i 5} {$i < 9} {incr i} {
$ns simplex-link $UE($i) $EBT(2) 500Mb 2ms LTEQueue/ULAirQueue
$ns simplex-link $EBT(2) $UE($i) 1Gb 2ms LTEQueue/DLAirQueue
}

#Attach UDP agents for all user ends
for {set i 0} {$i < 9} {incr i} {
set udp($i) [new Agent/UDP]
$ns attach-agent $UE($i) $udp($i)
}

#Set packet patterns
for {set i 0} {$i < 9} {incr i} {
set cbr($i) [new Application/Traffic/CBR]
$cbr($i) set packetSize_ 480
$cbr($i) set interval_ 0.03
$cbr($i) set class_ $i+1
$cbr($i) attach-agent $udp($i)
}
#Create LossMoniter agents and attach on each user ends
for {set i 0} {$i < 9} {incr i} {
set sink($i) [new Agent/LossMonitor]
$ns attach-agent $UE($i) $sink($i)
}

## UE(0) to UE(2)
$ns connect $udp(0) $sink(2)
$ns connect $udp(2) $sink(0)
## UE(3) to UE(5)
$ns connect $udp(3) $sink(5)
$ns connect $udp(5) $sink(3)

## setting up extra udp agent and attach to cbr traffic for group chatting
#Set additional CBR traffics and UDP agents for group chatting
for {set i 0} {$i < 8} {incr i} {
set cbrg($i) [new Application/Traffic/CBR]
$cbrg($i) set packetSize_ 480
$cbrg($i) set interval_ 0.03
```


```
$cbrg($i) set class_ $i+12
set udpgroup($i) [new Agent/UDP]
$cbrg($i) attach-agent $udpgroup($i)
}

$ns attach-agent $UE(1) $udpgroup(0)
$ns attach-agent $UE(1) $udpgroup(1)
$ns attach-agent $UE(4) $udpgroup(2)
$ns attach-agent $UE(4) $udpgroup(3)
$ns attach-agent $UE(6) $udpgroup(4)
$ns attach-agent $UE(6) $udpgroup(5)
$ns attach-agent $UE(7) $udpgroup(6)
$ns attach-agent $UE(7) $udpgroup(7)

$ns connect $udp(1) $sink(4)
$ns connect $udpgroup(0) $sink(6)
$ns connect $udpgroup(1) $sink(7)
$ns connect $udp(4) $sink(1)
$ns connect $udpgroup(2) $sink(6)
$ns connect $udpgroup(3) $sink(7)
$ns connect $udp(6) $sink(1)
$ns connect $udpgroup(4) $sink(4)
$ns connect $udpgroup(5) $sink(7)
$ns connect $udp(7) $sink(1)
$ns connect $udpgroup(6) $sink(4)
$ns connect $udpgroup(7) $sink(6)

# Initialize buffers to hold previous results

for {set i 0} {$i < 9} {incr i} {
set holdtime($i) 0
set holdseq($i) 0
set holdrate($i) 0
}

proc record {} {
global sink t l d holdrate holdtime holdseq

set ns [Simulator instance]
set time 0.5 ;#Set Sampling Time to 0.2 Sec

for {set i 0} {$i < 9} {incr i} {
```


```
set bwB($i) [$sink($i) set bytes_]
set bwN($i) [$sink($i) set nlost_]
set bwLPKT($i) [$sink($i) set lastPktTime_]
set bwNPK($i) [$sink($i) set npkts_]
}
set now [$ns now]
# Record data in Trace Files
for {set i 0} {$i < 9} {incr i} {
#put data rates in traces
puts $t($i) "$now [expr (($bwB($i)+$holdrate($i))*8)/(2*$time*1000000)]"
#put packet loss rate
puts $l($i) "$now [expr $bwN($i)/$time]"
#put dalays in traces
if { $bwNPK($i) > $holdseq($i) } {
puts $d($i) "$now [expr ($bwLPKT($i) - $holdtime($i))/($bwNPK($i) - $holdseq($i))]"
} else {
puts $d($i) "$now [expr ($bwNPK($i) - $holdseq($i))]"
}
}

# clear Variables
for {set i 0} {$i < 9} {incr i} {
$sink($i) set bytes_ 0
$sink($i) set nlost_ 0
}
#store intermidaites data in buffer
for {set i 0} {$i < 9} {incr i} {
set holdtime($i) $bwLPKT($i)
set holdseq($i) $bwNPK($i)
set holdrate11($i) $bwB($i)
}

$ns at [expr $now+$time] "record"
;# Schedule Record after $time interval sec
}

proc stop {} {
global ns f nf t l d
$ns flush-trace
close $f
close $nf
# Close Trace Files
```


```
for {set i 0} {$i < 9} {incr i} {
close $t($i)
close $l($i)
close $d($i)
}

#exec xgraph out(0).tr out(1).tr out(2).tr out(3).tr out(4).tr out(5).tr out(6).tr out(7).tr out(8).tr
#-geometry 800x400 &
exec xgraph lost(0).tr lost(1).tr lost(2).tr lost(3).tr lost(4).tr lost(5).tr lost(6).tr lost(7).tr lost(8).tr
-geometry 800x400 &
#exec xgraph delay(0).tr delay(1).tr delay(2).tr delay(3).tr delay(4).tr delay(5).tr delay(6).tr delay(7).tr
#delay(8).tr -geometry 800x400 &
exec nam out.nam &
exit 0
}

#single user chat
$ns at 0.0 "record"
$ns at 1 "$cbr(0) start"
$ns at 1 "$cbr(2) start"
$ns at 1 "$cbr(3) start"
$ns at 1 "$cbr(5) start"
$ns at 19 "$cbr(0) stop"
$ns at 19 "$cbr(2) stop"
$ns at 19 "$cbr(3) stop"
$ns at 19 "$cbr(5) stop"

##groupchat
$ns at 20 "$cbr(1) start"
$ns at 20 "$cbr(4) start"
$ns at 20 "$cbr(6) start"
$ns at 20 "$cbr(7) start"
$ns at 20 "$cbrg(0) start"
$ns at 20 "$cbrg(1) start"
$ns at 20 "$cbrg(2) start"
$ns at 20 "$cbrg(3) start"
$ns at 20 "$cbrg(4) start"
$ns at 20 "$cbrg(5) start"
$ns at 20 "$cbrg(6) start"
$ns at 20 "$cbrg(7) start"
$ns at 39 "$cbr(1) stop"
$ns at 39 "$cbr(4) stop"
```


```
$ns at 39 "$cbr(6) stop"
$ns at 39 "$cbr(7) stop"
$ns at 39 "$cbrg(0) stop"
$ns at 39 "$cbrg(1) stop"
$ns at 39 "$cbrg(2) stop"
$ns at 39 "$cbrg(3) stop"
$ns at 39 "$cbrg(4) stop"
$ns at 39 "$cbrg(5) stop"
$ns at 39 "$cbrg(6) stop"
$ns at 39 "$cbrg(7) stop"
$ns at 40 "stop"

$ns run
```