ENSC 427: COMMUNICATION NETWORKS

SPRING 2014

FINAL PROJECT

VoIP Performance of City-Wide Wi-Fi and LTE

www.sfu.ca/~tly/webpage.html

Ou, Cheng Jie 301144355 <jou@sfu.ca>

Yang, Tian Lin 301107652 <tly@sfu.ca>

Chen, Yawen 301122305 <yca137@sfu.ca>

TEAM 5

Table of contents

List of Figures

List of Tables

Abstract

Voice calling over the internet (VoIP) has become more accessible to consumers in recent years thanks to the increasing amount of voice call applications and the increasing capabilities of the internet. With the decreasing costs of wireless networks, consumers can now access the internet almost anywhere. While network providers offer various options for our daily network usage, our project focuses on two most popular choices: Municipal Wi-Fi and LTE. Municipal Wi-Fi is a large wireless access area consisting of many Wi-Fi hotspots. Customers can access the Internet anywhere within this region through thousands of these wireless hotspots. LTE stands for Long-Term Evolution. It is currently the most popular wireless data communication technology for mobile devices, and may be accessed anywhere within the network providers' range of services. The goal of this project is to examine the delay, jitter, and packet loss of the two technologies while voice calling in order to compare the advantages and disadvantages of each technology. We plan to measure performance by creating more realistic simulation scenarios by using various network loads as well as by varying the distance from the signal source.

Introduction

Making voice calls over the internet has become an everyday norm for many people. From contacting love ones to attending international meetings, voice calling has liberated many from the costly phone bills incurred from lengthy and long distance calls. As of Q1 of 2013 there are over 150 million full service VoIP subscribers, replacing conventional telephone calls, and in 2010 there were over 660 million subscribers to Skype (the most popular internet telephony service) [11]. The improvement of wireless network technologies over the years has granted us access to the internet from almost anytime anywhere which allows us to make VoIP calls on the go. As VoIP technologies attracting more and more users every year, people will undoubtedly question which type of wireless internet technology is better for using it? To answer this question, our report hopes to bring forth ns2 simulations of two of the most widely used wireless network technologies. In particular we will be running simulations of both LTE and Wi-Fi network, and then compare their performances to each other, in hopes to discover which one of the two technologies is better suited for making VoIP calls.

1. City-Wide Wi-Fi

City-Wide WI-FI is a network consisting of many wireless internet hotspots (access points), that are spread throughout a city in order to offer its citizens the freedom to connect to the Internet, wherever they are. WI-FI uses microwaves in order to allow devices to exchange data with the internet, and follows the IEEE 802.11 standard. WI-FI has become a staple in many people's homes and businesses, and is the most widely used method for connecting to the internet when lingering in a very small area.

1.1 Wi-Fi Topology

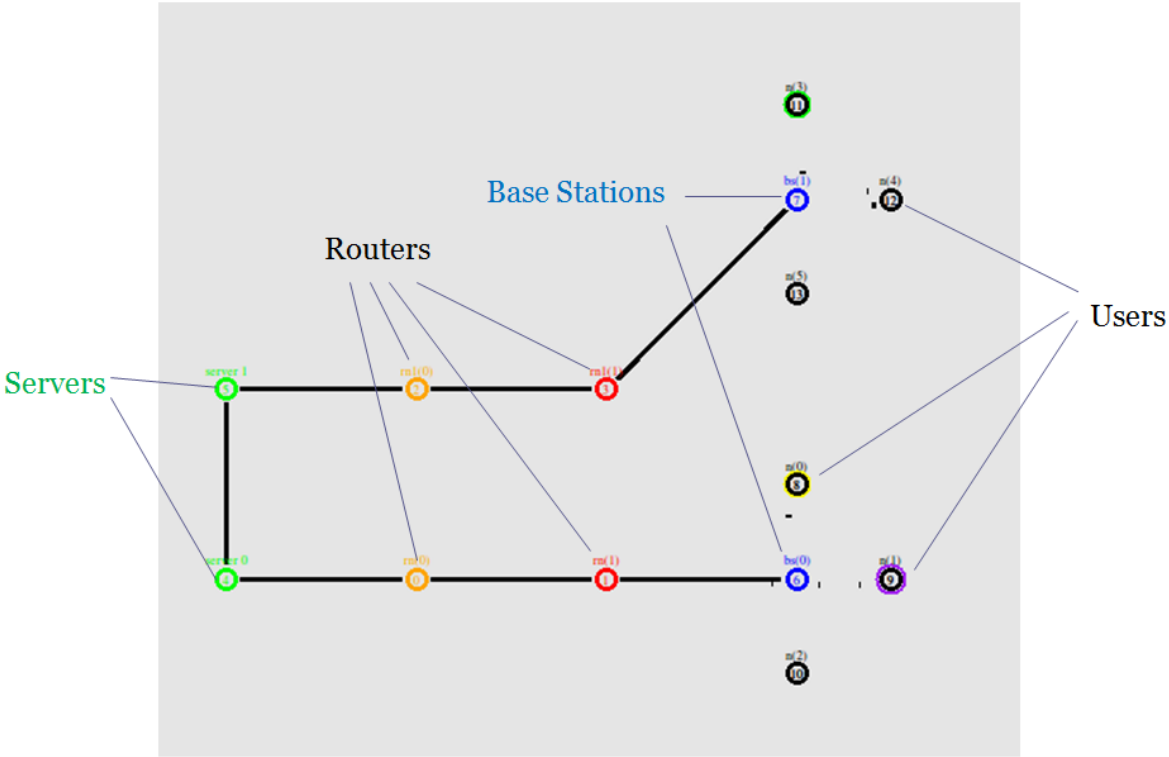

Figure 1: Wi-Fi Topology

Servers - Servers are the backbone of the internet. They provide services for a network and in our case, allow our users to access the internet. When a VoIP call is performed, the data is sent to the server who then facilitates what to do with the data in order for two or more users to be able to chat over the internet.

Routers - Routers are essentially checkpoints for our data. The data we send go through a pathway of routers in order to access the servers. If a certain router goes down then the pathway will be updated so as to not use the broken router.
In essence, routers make up the path our data travels along as well as facilitates the paths our data must follow.

Base Station - A base station is what our mobile devices connect to in order to send data to servers. Base stations can be commonly found in homes and businesses and are often referred to as “routers” as well. Base stations are the devices that create the WI-FI microwaves for devices to detect and connect to.

In our WI-FI simulation, we picked G.728 (16 Kbps) codec and bitrates. The Voice payload size is 60 Bytes and the frequency of the packets sent are 33.3 PPS(Packets Per Second) [13].

1.1.1 Wired Connections

All wired connections were made with duplex links as followed:

```
$ns duplex-link $sn(0) $rn(0) 5Gb 2ms DropTail
$ns duplex-link $rn(0) $rn(1) 5Gb 2ms DropTail
$ns duplex-link $rn(1) $bs(0) 500Mb 2ms DropTail
$ns duplex-link $sn(1) $sn(0) 5Gb 100ms DropTail
$ns duplex-link $sn(1) $rn1(0) 5Gb 2ms DropTail
$ns duplex-link $rn1(0) $rn1(1) 5Gb 2ms DropTail
$ns duplex-link $rn1(1) $bs(1) 500Mb 2ms DropTail
```

The delay between two servers sn(0) and sn(1) were set to 100ms in order to simulate long distance voice calling. The bandwidth of all wired links were set relatively large, this is so that there will not be any bottlenecking in the wired parts of the simulation. This allows us to focus our attention on the wireless transmissions.

1.1.2 Wireless Connections

Wireless connections were configured using IEEE 802.11g parameters as followed:

```
set opt(Wi-Fi_bw) 54Mb       ;# link BW on Wi-Fi net
Phy/WirelessPhy set Pt_ 0.25622777   ;#transmit power
Phy/WirelessPhy set L_ 1.0    ;#System loss factor
Phy/WirelessPhy set bandwidth_ 1   ;#opt(Wi-Fi_bw)
Phy/WirelessPhy set freq_ 2.472e9   ;#channel-13. 2.472GHz
Phy/WirelessPhy set CPThresh_ 10.0   ;#reception of simultaneous packets
Phy/WirelessPhy set CSThresh_ 5.011872e-12 ;#carrier sensing threshold
Phy/WirelessPhy set RXThresh_ 5.82587e-09 ;#reception threshold
Mac/802_11 set dataRate_ $opt(Wi-Fi_bw)
Mac/802_11 set basicRate_ 24Mb    ;#for broadcast packets
```

Data Rates (Transmit Rates), Basic Rates, and Frequency band parameters were taken from the tables below:

Operational Mode	Supported Transmit Rates (Mbps)	Supported Basic Rates (Mbps)*	Slot Time	Protection Mechanism (CTS-to-Self)
11b only	1, 2, 5.5, 11	1 (default), 2	20 us	Not Supported
11g only	6, 9, 12, 18, 24, 36, 48, 54	6, 12,2 4(default)	9 us	Not Supported
11g - wifi	1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54	1, 2, 5.5, 11, 6, 12, 24(default)	9 us	Not Supported
11bg	1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54	1, 2 (default), 5.5, 11	9 or 20 us (dynamic)	Implemented

Table 1: 802.11 Wireless LAN standard operational parameters [12]

802.11 protocol suite

	802.11a	802.11b	802.11g
Year	1999	1999	2003
Products since	2001	1999	2003
Typical range	~15 m indoor ~100 m outdoor	~30 m indoor ~200 m outdoor	~30 m indoor ~200 m outdoor
Bandwidth	54 Mbps	11 Mbps	54 Mbps
Physical layer	OFDM	DSSS	OFDM
Frequency band	5 GHz unlicensed	2.4 GHz unlicensed	2.4 GHz unlicensed
Backward compatibility	None	802.11	802.11b

Table 2: 802.11 protocol suite [9]

CSThresh_ (carrier sensing threshold) and RXThresh_ (reception threshold) were adjusted accordingly to adjust the range of reception to around 50 meters.

2. Long-Term Evolution (LTE)

LTE is a standard for wireless phones and mobile devices to connect to the internet. LTE offers much higher speeds than conventional 3G technologies and is commonly referred to as 4G LTE. LTE waves span a wide area, usually throughout a city, and allow users to access the internet when they're on the move. Many phones and devices now come equipped with the ability to use LTE and so this technology has experienced massive adoption in the past couple of years.

2.1 LTE Topology

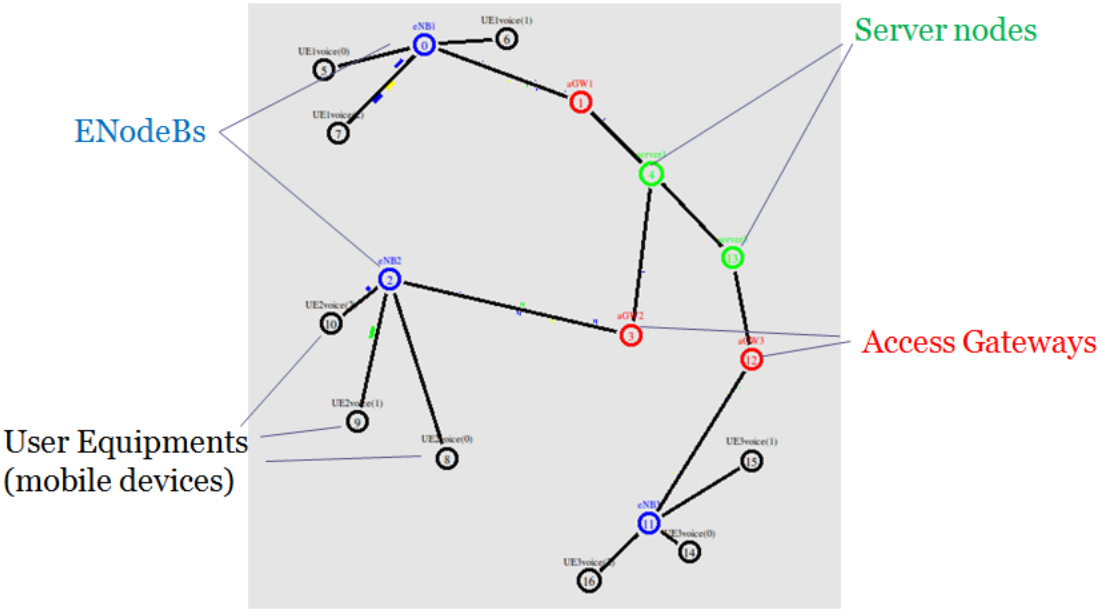

Figure 2: LTE Topology

Servers - Servers for LTE performs the same functions as servers for WI-FI. They manage all the data that gets sent to them and performs the necessary actions and replies.

Access Gateway (AGW) - Access gateways are somewhat analogous to routers for WI-FI. AGW facilitates the transfer of data so that they can reach the servers. Furthermore, AGWs also provide broadband services and perform core network functions.

eNodeB - eNodeBs are enhanced base transceiver systems that provides the LTE air interface. Essentially eNodeBs provide the signals that mobile devices, such as cellphones, pick up so that they may exchange data with the internet.

2.1.1 aGW to Server and Server to Server connections

aGW to Server connections were made using duplex links as followed:

$ns duplex-link $aGW1 $server1 10Gb 10ms DropTail
$ns duplex-link $aGW2 $server1 10Gb 10ms DropTail
$ns duplex-link $aGW3 $server2 10Gb 10ms DropTail
$ns duplex-link $server1 $server3 100Gb 100ms DropTail

The 100ms delay between servers simulates long distance connection.

2.1.2 LTE connections

We installed the ns2 LTE module using procedures found on S. Naveen's blog [6] (See detailed description in appendix A). Using the installed module, we were able to connect the aGWs to EnodeBs with two simplex-links with LTEQueues :

$ns simplex-link $eNB $aGW2 5Gb 10ms LTEQueue/ULS1Queue
$ns simplex-link $aGW $eNB2 5Gb 10ms LTEQueue/DLS1Queue

Where ULS1Queue is for uploading and DLS1Queue is for downloading.

UE nodes (User Equipment) were connected to EnodeBs using simplex links with ULAirQuene and DLAirQueue to simulate the wireless connections:

$ns simplex-link $UE $eNB 500Mb 2ms LTEQueue/ULAirQueue
$ns simplex-link $eNB $UE 1Gb 2ms LTEQueue/DLAirQueue

3. The Simulation

Voice over internet protocol means sending voice data traffic over IP-based network. In our simulations, we implemented VoIP with UDP protocol. We attached UDP agents and sinks to all user devices for sending and receiving voice data.

3.1 One to One Voice Call

For both of our LTE and Wi-Fi topologies, we first initiated long distance voice calls by exchanging voice data between two user nodes belonging to two different server branches. For LTE, we connected UE1(1) to UE3(1) and for Wi-Fi, n(0) to n(3) and n(2) to n(5). We also implemented a local voice call in our LTE simulation by exchanging data between UE1(0) and UE2(0) (both belongs to the same server branch). The simulation for One to One Voice Calls starts at 1.0 seconds and ends at 14.0 seconds.

3.2 Group Chat

In both topologies, we initiated group chat simulations between 15.0 seconds to 30.0 seconds. For the LTE simulation, we exchanged voice data between UE1(2), UE2(1), UE2(2) and UE3(0). For Wi-Fi, data was exchanged between n(0), n(1), n(3) and n(4).

4. Data Collection

All of the data from the simulations were collected using the LossMonitor class included in ns2. The LossMonitor class can be attached to a sink to retrieve last packets arrival time, number of packets, number of loss packets and the number of bytes received for that specific sink. All the data retrieved from LossMonitor were stored into these variables: LastPktTime_(last packet time), npkts_(number of packets), nlost_(number of packets lost), and bytes_ (number of bytes received) respectively. Using these variables, we were able to compute the throughput, packet loss, delay and jitter of our simulations.

4.1 Throughput

Throughput is the rate of successful packet delivery. Bytes_ will be incremented whenever the sink's LossMonitor detects that a packet has been received successfully from its sender. In order to calculate the throughput in our simulations, we created several bw variables in the record procedure and set them equal to bytes_. We set the sample time to 0.2 so all the values will be updated every 0.2 seconds. We also created holdrate variables and set them equal to bytes_ but update them one iteration of record later than bw variables. Thus, bw (the current number of bytes received) and holdrate (the previous number of bytes received) allows us to calculate the throughput by adding them together, multiplying by 8 (to convert from bytes to bits), dividing by 2 (for the two intervals of time that record has ran), dividing by 1000000 (convert from bits to mega bits), and lastly by dividing by the current time so that we may get bits/s. Each iteration of record places this calculated value into an output trace file along with the current runtime. Following is an sample code for our throughput calculation:

```
# Record Bit Rate in Trace Files
puts $t11 "$now [expr (($bwB11+$holdrate11)*8)/(2*$time*1000000)]"
```

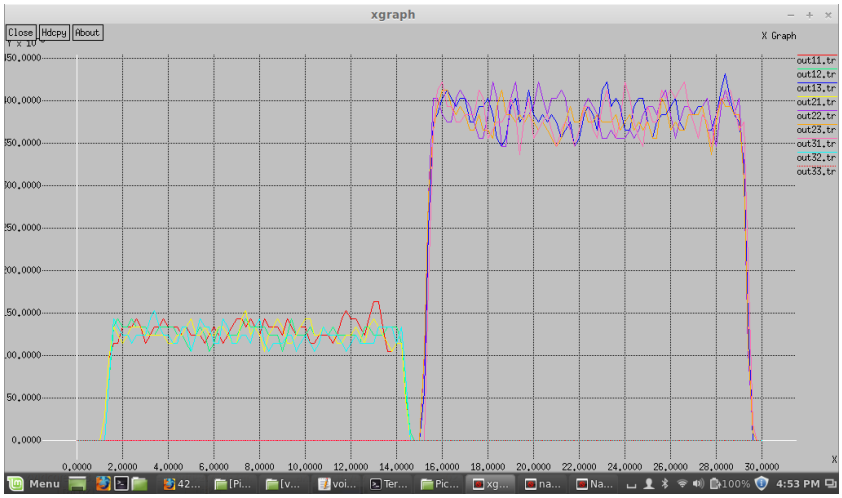

Figure 3: LTE Throughput

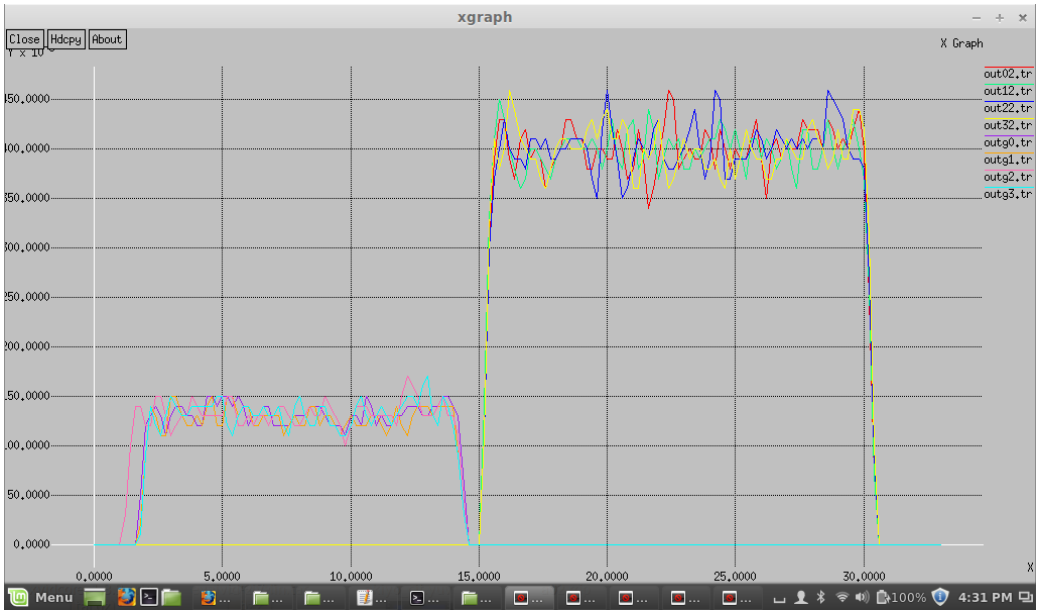

Figure 4: Wi-Fi Throughput

The two graphs show throughput levels similar to plateaus with different heights. This is consistent with the simulation because at the beginning there are only single voice calls (local and/or international). At 15 seconds however, a significantly higher plateau occurs. This plateau is evidently due to the increase in packets of data being transmitted from the group chats, which consists of many nodes sending and receiving to and from each other.

4.2 Packet Loss Rate

Packet loss occurs when sent packets of data fail to reach their intended locations. In our case packet loss happens when the data from one node does not reach the node it was trying to send to. Just as in the calculations for throughput, we placed the number of loss packets into a *bw* variable and then divided by the current runtime to get the number of packets lost per second. Once again, this calculation will be performed each time that *record* runs and the calculated value will be output to a trace file along with the current runtime. The following is an example of Packet Loss Rate function:

```
# Record Packet Loss Rate in File
puts $l11 "$now [expr $bwN11/$time]"
```

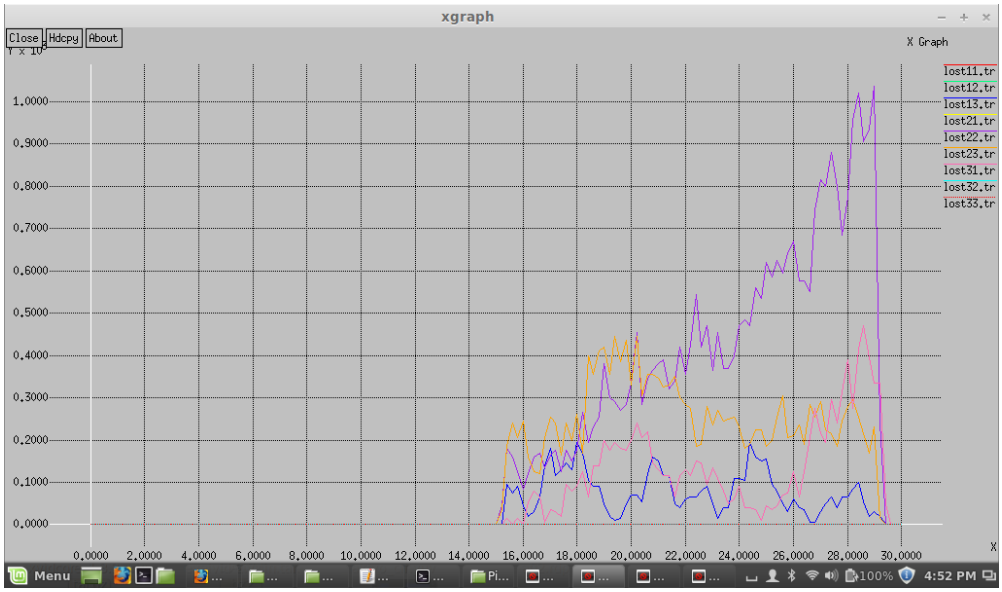

Figure 5: LTE Packet Loss Rate

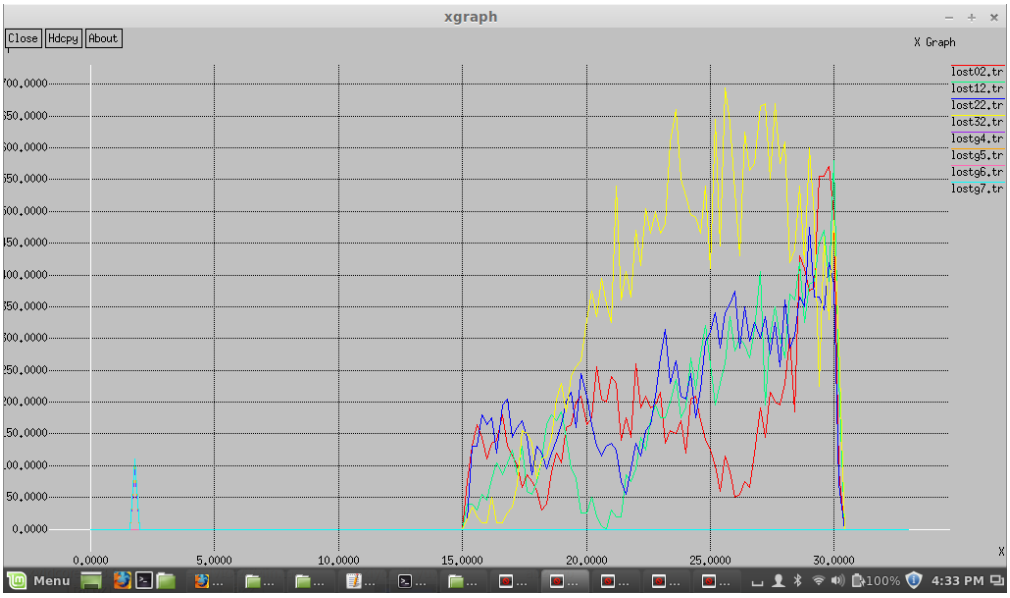

Figure 6: Wi-Fi Packet Loss Rate

From the two graphs above it can be seen that significant packet loss occurs when the group chat sessions are commencing. These results are consistent with the simulation because when group chat sessions are occurring there is significantly more stress on both the wired and wireless links to transfer more packets. For packet loss rate, LTE triumphs due to the fact that it has much fewer packets lost when compared to WI-FI.

4.3 End to End Delay

End to End Delay is the amount of time it takes for a packet to transmit from its source to destination. Since our *record* function runs in intervals of time, we are only able to calculate an average delay for each packet inside that interval of time. This is done by an if statement and corresponding calculations. If the current number of packets received (bwNPK) is greater than the last number of packets received (holdseq) then subtract the most current last packet arrival time (bwLPKT) with the previous iteration of *record*'s most current last packet arrival time (holdtime). Then divide this value by the current number of packets received (bwNPK) minus the previous number of packets received (holdseq). This calculation will give us an average delay of each packet received since the last *record* procedure has ran. If the current number of packets received is less than or equal to the previous number of packets received ("less than" should never happen), it simply means no packet has been received since the last interval, and the output is the difference of the two (should be zero). Sample calculation code is as follows:

```
# Record Packet Delay in File
if { $bwNPK11 > $holdseq11 } {
    puts $d11 "$now [expr ($bwLPKT11 - $holdtime11)/($bwNPK11 - $holdseq11)]"
} else {
    puts $d11 "$now [expr ($bwNPK11 - $holdseq11)]"
}
```

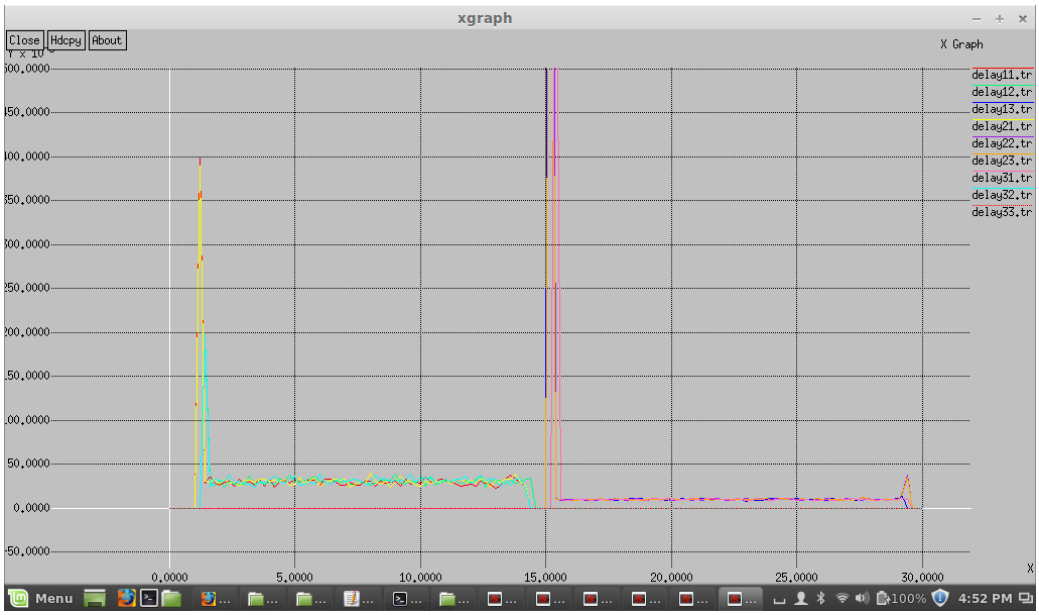

Figure 7: LTE Delay

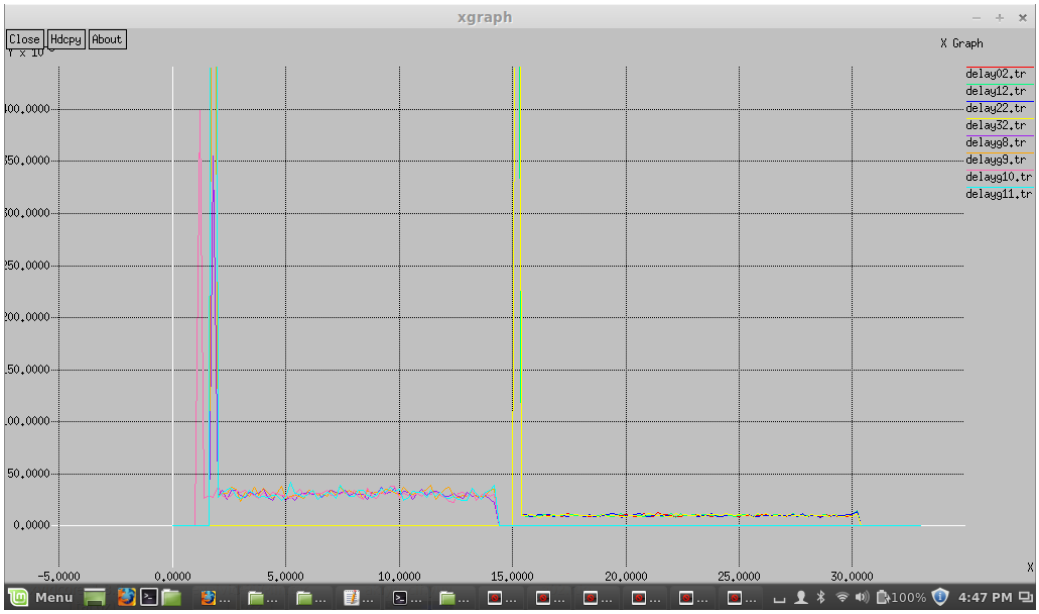

Figure 8: Wi-Fi Delay

As can be seen from both Delay graphs, the delays for WI-FI and LTE are quite similar when it comes to VoIP calls. The significant spikes in delays are due to contention window adjustments. All in all, both technologies display similar performances in delays during packet deliver. All in all, both technologies display similar performances in delays during packet deliver with LTE having slightly smaller delay spikes than WiFi.

4.4 Jitter

Jitter is the difference in the delays of the packets received. In order to calculate the jitter of each sink we simply imported the delay trace files we previously obtained into Microsoft Excel and calculated the difference of each row in the column of delay times.

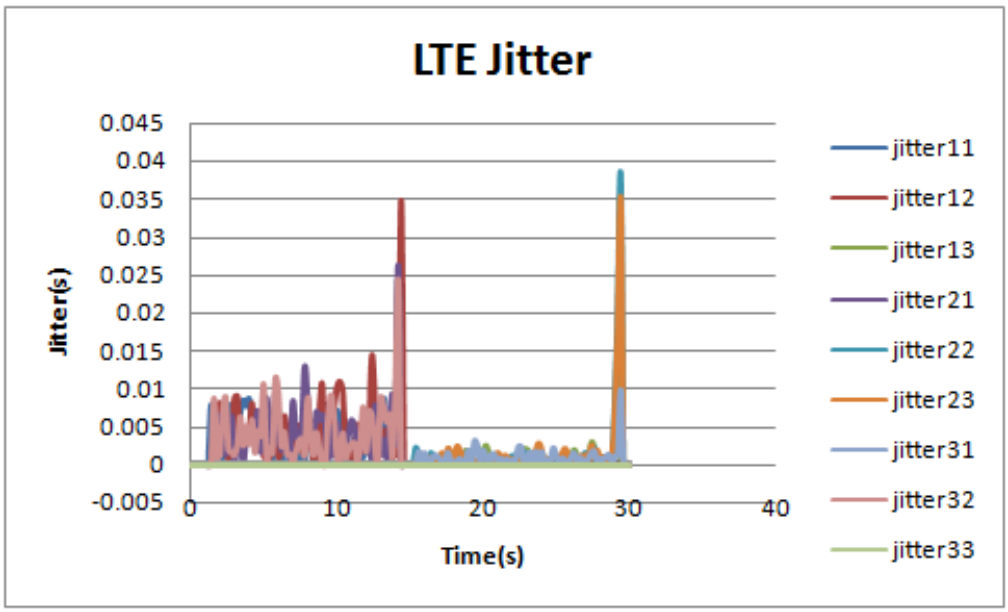

Figure 9: LTE Jitter

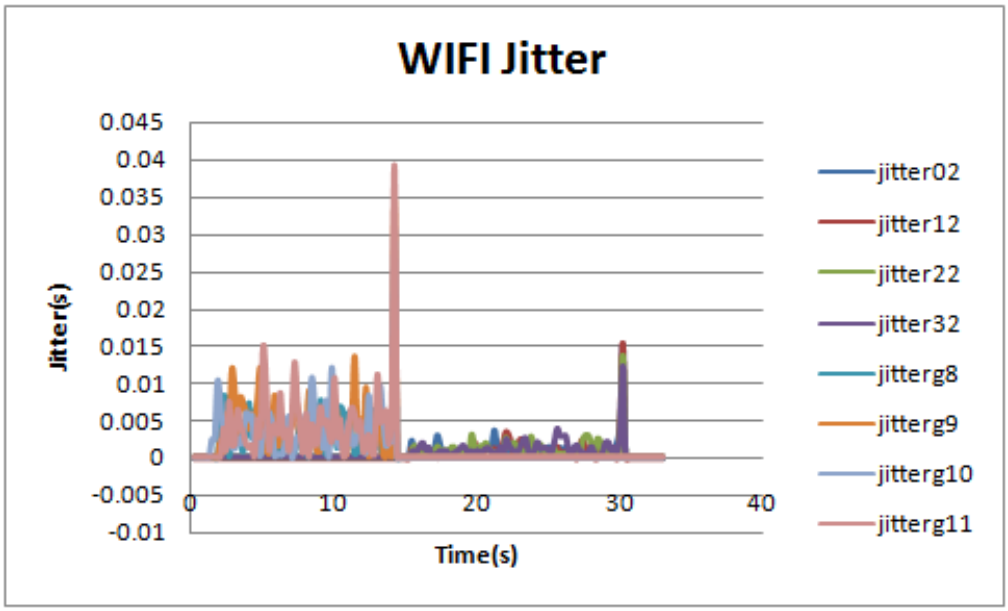

Figure 10: Wi-Fi Jitter

As can be seen from both Delay graphs, the delays for WIFI and LTE are quite similar when it comes to VoIP calls. The significant spikes in delays at the beginning of the voice calls are due to contention window adjustments. All in all, both technologies display similar performances in delays during packet deliver.

5. Discussion

5.1 Difficulties

For this project many difficulties and obstacles occurred before we were finally able to collect the correct data for our simulations. First and foremost, in order to have a LTE simulation we needed a LTE module to implement into ns2. The only LTE module we could find was compatible with version 2.33 of ns2 only, thus in order to use the LTE module we first needed to install ns2.33. After successfully installing ns2.33 we began to implement the LTE module, which consequently resulted in many errors when making and configuring ns2. After many attempts and searches through forums we were finally able to iron out the errors and have ns2 running successfully. The WI-FI simulations did not require a module to be implemented into ns2.

5.1.1 Topology

In order to implement LTE and WI-FI in ns2 we first needed to understand their topologies. This involved understanding the jobs and functions of each individual system within the two topologies before creating them in ns2. In particular we needed to know the transfer speeds and other parameters of base stations, routers, eNodeBs, AGWs and servers, as well as their individual functions in the transferring of data packets.

5.1.2 Data Calculation and Graphing

Although a quick Google search will result in methods to calculate throughput, packet loss, delay and jitter, it was difficult to implement those methods into ns2 for our simulations. Furthermore, after we concluded that the data in the trace files were indeed correct, graphing them all on xgraph required a new level of knowledge and understanding of xgraph.

5.1.3 Wi-Fi Hierarchy

For the WI-FI simulations we needed to use domains and clusters in order for ns2 to correctly understand and simulate our WI-FI topology. Understanding how domains and clusters work in ns2 required much research and time.

5.2 Desired Improvements

Although the essences of WI-FI and LTE have been simulated, there is still much that can be improved for our project.

Firstly, a better WI-FI topology should be constructed because our current topology only uses 4 routers and 2 servers. A more realistic simulation will have many more routers making up the path for packets to travel.

Second, movement of our wireless nodes should be implemented. With our current setup, all the nodes representing mobiles devices are stationary; in a real world simulation the user nodes should be moving in order to better represent consumers using their mobile devices for communication.

Lastly, the multicast function in ns2 should be used for group chatting instead of our current individual UDP setup. Our current setup utilizes many UDPs and attaches them to every single user node in order to send packets of data. With the multicast function in ns2 there will be no need for such redundancy, but much more time will have to be invested into the project in order to understand and be able to use the multicasting functionality of ns2.

5.3 Future Work

For future work individuals may choose to implement the 802.11ac standard for WI-FI instead of our 802.11g standard. This will undoubtedly result in a simulation more concurrent with today's technologies. Individuals may also implement larger traffic to represent high-definition video calling by users.

Conclusion

For our report we have brought forth ns2 simulations of VoIP calling using LTE and WIFi, currently the two most widely used network technologies. Using the capabilities of ns2 we have successfully simulated and collected data from both the LTE and Wi-Fi topologies. From the data, it can be seen from their established graphs that the throughput, delay and jitter for both technologies are very similar. Small performance differences in the two technologies can be seen from the packet loss and delay graphs, which shows LTE as the prevailing technology. As a result, it is fair to conclude that LTE provides a more fluid experience when performing voice calls over the internet. Although LTE has prevailed as the better technology for voice calling, in essence both Wi-Fi and LTE do a very thorough job, and using either one for everyday VoIP calls will suffice.

Reference:

[1] G. A. Abed, M. Ismail, and K. Jumari, "A Realistic Model and Simulation Parameters of LTE-Advanced Networks," Faculty of Engineering and Built Environment, National University of Malaysia, Selangor, Rep. ISSN:2278-1021, Aug. 2012. Available: www.researchgate.net/publication/256871810_A_Realistic_Model_and_Simulation_Parameters_of_LTE-Advanced_Networks/file/72e7e524063701459f.pdf+&cd=1&hl=en&ct=clnk&gl=ca

[2] A. Leon-Garcia and I. Widjaja, "Multimedia Information and Networking," in *Communication Networks Fundamental Concepts and Key Architectures*, 1st Ed. Sydney, Australia: McGraw-Hill Education, 2000, Ch.12, pp. 753–804.

[3] H. Wong, L. Kondi, A. Luthra, and S. Ci, "4G Wireless Communications and Networking," in *4G Wireless Video Communications*, 1st Ed. Mississauga, CA: John Wiley and Sons, 2009, Ch.4, pp. 97-133.

[4] K. Fall, K. Varadhan. (2011, November 5). *The Manual* (*2nd ed.*)[Online]. Available: http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf

[5] A. Ezreik and A. Gheryani, "Design and simulation of wireless networks using ns-2," in *Proc. International Conference on Computer Science and Information Technology*, Singapore, pp.1–5, Apr. 2012. Available: http:psrcentre.org/images/extraimages/412630.pdf [Mar. 6, 2014].

[6] S. Naveen. "LTE (Long Term Evaluation) Network in NS2." available: http://naveenshanmugam.blogspot.ca/2014/02/lte-long-term-evaluation-network-in-ns2.html [Mar. 6, 2014].

[7] Google, "Simulating VOIP over UDP," Available: https://sites.google.com/site/networksprojectwiki/bit10/compnetworks/voip-performance-over-udp-and-sctp-in-ns2/simulating-voip/voip-over-udp [Mar. 20, 2014].

[8] T. Haukaas, "Rate Adaptive Video Streaming over Wireless Networks." Dep. of Telematics, Norwegian University of Science and Technology, Trondheim, Jun. 2007. pp.98-99. Available:http://folk.uio.no/paalee/referencing_publications/ref-admctrl-haukaas-thesis-2007.pdf [Mar. 20, 2014].

[9] J. Naoum-Sawaya, B. Ghaddar, S. Khawam, H. Safa, H. Artail, and Z. Dawy, "Adaptive Approach for QoS Support in IEEE 802.11e Wireless LAN," in IEEE International Conference on Wireless and Mobile Computing , Networking and Communications (WiMob 2005), Montreal, Canada, August 2005.

[10] Google, "How to measure the throughput, packet drop rate, and end-to-end delay for UDP-based application over wirelessnetworks ," Available: http://hpds.ee.ncku.edu.tw/~smallko/ns2/wireless-udp-1.htm [Mar. 12, 2014]

[11] Point Topic Ltd., "VoIP Statistics – Market Analysis Q1 2013," Available: http://point-topic.com/wp-content/uploads/2013/02/Point-Topic-Global-VoIP-Statistics-Q1-2013.pdf [Mar. 12, 2014].

[12] Moonblink, “Differences in 802.11b and 802.11g,” Available: http://www.moonblinkWi-Fi.com/differences_in_80211b_and_802.cfm[Mar. 12, 2014].
[13] CISCO, “Voice Over IP - Per Call Bandwidth Consumption,” Available: http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html [Mar. 12, 2014].

Appendix

Patching LTE Module into NS2

Step 1: Download the LTE patch, tk-8.4-lastevent.patch
Step 2: Place the LTE patch into ns-allinone-2.33/tk8.4.18/
Step 3: Using terminal navigate to the tk8.4.18 folder, *cd ns-allinone-2.33/tk8.4.18/*
Step 4: In terminal type *patch -p0 < tk-8.4-lastevent.patch*. If permission is denied, type *sudo patch -p0 < tk-8.4-lastevent.patch* and enter your password. After the patching has finished type return to the ns-allinone-2.33 folder, *cd ../*
Step 5: In terminal type *./install*
Step 6: After the install has finished type *cd ns-2.33/ && mv ns ns233 && make clean && mv Makefile Makefile.org*
Step 7: Next type *svn checkout http://lte-model.googlecode.com/svn/trunk/lte-model-read-only*. Make sure that subversion repository is installed in your computer.
Step 8: Next type *mkdir project* followed by *cd lte-model-read-only/*
Step 9: Next *sh checkin* and *cd ../*
Step 10: Now edit the new Makefile, lines 41, 67, 82 to the actual location of ns-allinone-2.33/ns-2.33/ on your computer.
Step 11: Finally type *make*
Note: for errors that may surface during installation or patching, please refer to http://www.linuxquestions.org/questions/ubuntu-63/how-to-installing-lte-module-patch-in-ns2-33-a-857930/

NS2 Code for Wi-Fi:

```
# Define options
set val(chan) Channel/WirelessChannel ;# channel type
set val(prop) Propagation/TwoRayGround ;# radio-propagation model
set val(netif) Phy/WirelessPhy ;# network interface type
set val(mac) Mac/802_11 ;# MAC type
set val(ifq) Queue/DropTail/PriQueue ;# interface queue type
set val(ll) LL ;# link layer type
set val(ant) Antenna/OmniAntenna ;# antenna model
set val(ifqlen) 50 ;# max packet in ifq
set val(nn) 6 ;# number of mobilenodes
set val(rp) DSDV ;# routing protocol
set val(x) 1000 ;# X dimension of topography
set val(y) 1000 ;# Y dimension of topography
set val(stop) 33 ;# time of simulation end

#WIFI 802.11g settings
set opt(wifi_bw) 54Mb                   ;# link BW on wifi net
Phy/WirelessPhy set Pt_ 0.25622777          ;#transmit power
Phy/WirelessPhy set L_ 1.0                  ;#System loss factor
Phy/WirelessPhy set bandwidth_ 1            ;#opt(wifi_bw)
Phy/WirelessPhy set freq_ 2.472e9           ;#channel-13. 2.472GHz
Phy/WirelessPhy set CPThresh_ 10.0          ;#reception of simultaneous packets
Phy/WirelessPhy set CSThresh_ 5.011872e-12 ;#carrier sensing threshold
Phy/WirelessPhy set RXThresh_ 5.82587e-09  ;#reception threshold
Mac/802_11 set dataRate_ $opt(wifi_bw)
Mac/802_11 set basicRate_ 24Mb              ;#for broadcast packets

# *** Jitter ****
set j0 [open jitter01.tr w]
set j1 [open jitter02.tr w]
set j2 [open jitter03.tr w]
set j3 [open jitter04.tr w]
set jg0 [open jitterg01.tr w]
set jg1 [open jitterg02.tr w]
set jg2 [open jitterg03.tr w]
set jg3 [open jitterg04.tr w]

# *** Throughput Trace ***
set f0 [open out02.tr w]
set f1 [open out12.tr w]
set f2 [open out22.tr w]
set f3 [open out32.tr w]
set g0 [open outg0.tr w]
```

```
set g1 [open outg1.tr w]
set g2 [open outg2.tr w]
set g3 [open outg3.tr w]

# *** Packet Loss Trace ***
set f4 [open lost02.tr w]
set f5 [open lost12.tr w]
set f6 [open lost22.tr w]
set f7 [open lost32.tr w]
set g4 [open lostg4.tr w]
set g5 [open lostg5.tr w]
set g6 [open lostg6.tr w]
set g7 [open lostg7.tr w]

# *** Packet Delay Trace ***
set f8 [open delay02.tr w]
set f9 [open delay12.tr w]
set f10 [open delay22.tr w]
set f11 [open delay32.tr w]
set g8 [open delayg8.tr w]
set g9 [open delayg9.tr w]
set g10 [open delayg10.tr w]
set g11 [open delayg11.tr w]

# Initialize Flags
set previous 0
set previous1 0
set previous2 0
set previous3 0

set previousg0 0
set previousg1 0
set previousg2 0
set previousg3 0

set delaynow 0
set delaynow1 0
set delaynow2 0
set delaynow3 0

set delaynowg0 0
set delaynowg1 0
set delaynowg2 0
set delaynowg3 0
```

```
set holdtime 0
set holdseq 0

set holdtime1 0
set holdseq1 0

set holdtime2 0
set holdseq2 0

set holdtime3 0
set holdseq3 0

set holdtimeg0 0
set holdseqg0 0

set holdtimeg1 0
set holdseqg1 0

set holdtimeg2 0
set holdseqg2 0

set holdtimeg3 0
set holdseqg3 0

set holdrate1 0
set holdrate2 0
set holdrate3 0
set holdrate4 0

set holdrateg0 0
set holdrateg1 0
set holdrateg2 0
set holdrateg3 0

set ns [new Simulator]
set tracefd [open voip_wifi.tr w]
set namtrace [open voip_wifi.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
# set up topography object
set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)
```

```
create-god [expr $val(nn) + 8]

# configure the nodes
$ns node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-wiredRouting ON \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON

$ns node-config -addressType hierarchical

AddrParams set domain_num_ 4                    ;# number of domains
lappend cluster_num 1 2 1 2                     ;# number of clusters in each domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 2 1 4 2 1 4                 ;# number of nodes in each cluster
AddrParams set nodes_num_ $eilastlevel ;# of each domain

$ns color 1 Yellow
$ns color 2 Green
$ns color 3 Blue
$ns color 4 Purple

##router nodes
set rn(0) [$ns node {0.0.0}]
$rn(0) label "rn(0)"
$rn(0) set X_ 100.0
$rn(0) set Y_ 100.0
$rn(0) set Z_ 0.0
$rn(0) color orange
set rn(1) [$ns node {1.0.0}]
$rn(1) label "rn(1)"
$rn(1) set X_ 200.0
```

```
$rn(1) set Y_ 100.0
$rn(1) set Z_ 0.0
$rn(1) color red

set rn1(0) [$ns node {2.0.0}]
$rn1(0) label "rn1(0)"
$rn1(0) set X_ 100.0
$rn1(0) set Y_ 200.0
$rn1(0) set Z_ 0.0
$rn1(0) color orange
set rn1(1) [$ns node {3.0.0}]
$rn1(1) label "rn1(1)"
$rn1(1) set X_ 200.0
$rn1(1) set Y_ 200.0
$rn1(1) set Z_ 0.0
$rn1(1) color red

##server nodes
set sn_adr {0.0.1 2.0.1}
for {set i 0} {$i < 2} { incr i} {
set sn($i) [$ns node [lindex $sn_adr $i]]
$sn($i) label "server $i"
$sn($i) color green
}

$sn(0) set X_ 0.0
$sn(0) set Y_ 100
$sn(0) set Z_ 0.0

$sn(1) set X_ 0.0
$sn(1) set Y_ 200.0
$sn(1) set Z_ 0.0

##base station
set bs(0) [$ns node {1.1.0}]
$bs(0) label "bs(0)"
$bs(0) random-motion 0
$bs(0) set X_ 300.0
$bs(0) set Y_ 100.0
$bs(0) set Z_ 0.0
$bs(0) color blue
set bs(1) [$ns node {3.1.0}]
$bs(1) label "bs(1)"
$bs(1) random-motion 0
```

```
$bs(1) set X_ 300.0
$bs(1) set Y_ 300.0
$bs(1) set Z_ 0.0
$bs(1) color blue

##wired links
$ns duplex-link $sn(0) $rn(0) 5Gb 2ms DropTail
$ns duplex-link $rn(0) $rn(1) 5Gb 2ms DropTail
$ns duplex-link $rn(1) $bs(0) 500Mb 2ms DropTail

$ns duplex-link $sn(1) $sn(0) 5Gb 100ms DropTail

$ns duplex-link $sn(1) $rn1(0) 5Gb 2ms DropTail
$ns duplex-link $rn1(0) $rn1(1) 5Gb 2ms DropTail
$ns duplex-link $rn1(1) $bs(1) 500Mb 2ms DropTail

##$ns node-config -wiredRouting OFF

##mobile nodes
set adr {1.1.1 1.1.2 1.1.3 3.1.1 3.1.2 3.1.3}
set n(0) [$ns node [lindex $adr 0]]
set n(1) [$ns node [lindex $adr 1]]
set n(2) [$ns node [lindex $adr 2]]
set n(3) [$ns node [lindex $adr 3]]
set n(4) [$ns node [lindex $adr 4]]
set n(5) [$ns node [lindex $adr 5]]
for {set i 0} {$i < 6} { incr i} {
$n($i) label "n($i)"
}

$n(0) base-station [AddrParams addr2id [$bs(0) node-addr]]
$n(1) base-station [AddrParams addr2id [$bs(0) node-addr]]
$n(2) base-station [AddrParams addr2id [$bs(0) node-addr]]
$n(3) base-station [AddrParams addr2id [$bs(1) node-addr]]
$n(4) base-station [AddrParams addr2id [$bs(1) node-addr]]
$n(5) base-station [AddrParams addr2id [$bs(1) node-addr]]

# Provide initial location of mobilenodes
$n(0) set X_ 300.0
$n(0) set Y_ 150.0
$n(0) set Z_ 0.0

$n(1) set X_ 350.0
$n(1) set Y_ 100.0
```

```
$n(1) set Z_ 0.0

$n(2) set X_ 300.0
$n(2) set Y_ 50.0
$n(2) set Z_ 0.0

$n(3) set X_ 300.0
$n(3) set Y_ 350.0
$n(3) set Z_ 0.0

$n(4) set X_ 350.0
$n(4) set Y_ 300.0
$n(4) set Z_ 0.0

$n(5) set X_ 300.0
$n(5) set Y_ 250.0
$n(5) set Z_ 0.0

# Set a UDP connection between n(0) and n(3)
set udp1(1) [new Agent/UDP]
set udp1(2) [new Agent/UDP]
$udp1(1) set class_ 0
$udp1(2) set class_ 1
$udp1(1) set fid_ 1
$udp1(2) set fid_ 2
set sink11 [new Agent/LossMonitor]
set sink12 [new Agent/LossMonitor]
$ns attach-agent $n(0) $udp1(1)
$ns attach-agent $n(3) $sink11
$ns connect $udp1(1) $sink11
$ns attach-agent $n(3) $udp1(2)
$ns attach-agent $n(0) $sink12
$ns connect $udp1(2) $sink12
set cbr1(1) [new Application/Traffic/CBR]
$cbr1(1) set packetSize_ 480
$cbr1(1) set interval_ 0.03
$cbr1(1) set class_ 0
$cbr1(1) attach-agent $udp1(1)
set cbr1(2) [new Application/Traffic/CBR]
$cbr1(2) set packetSize_ 480
$cbr1(2) set interval_ 0.03
$cbr1(2) set class_ 1
$cbr1(2) attach-agent $udp1(2)
```

```
$ns at 1.0 "$cbr1(1) start"
$ns at 1.0 "$cbr1(2) start"
$ns at 14.0 "$cbr1(1) stop"
$ns at 14.0 "$cbr1(2) stop"

#Set up UDP connection between n(2) and n(5)
set udp2(1) [new Agent/UDP]
set udp2(2) [new Agent/UDP]
$udp2(1) set class_ 0
$udp2(2) set class_ 1
$udp2(1) set fid_ 1
$udp2(2) set fid_ 2
set sink21 [new Agent/LossMonitor]
set sink22 [new Agent/LossMonitor]
$ns attach-agent $n(2) $udp2(1)
$ns attach-agent $n(5) $sink21
$ns connect $udp2(1) $sink21
$ns attach-agent $n(5) $udp2(2)
$ns attach-agent $n(2) $sink22
$ns connect $udp2(2) $sink22
set cbr2(1) [new Application/Traffic/CBR]
$cbr2(1) set packetSize_ 480
$cbr2(1) set interval_ 0.03
$cbr2(1) set class_ 0
$cbr2(1) attach-agent $udp2(1)
set cbr2(2) [new Application/Traffic/CBR]
$cbr2(2) set packetSize_ 480
$cbr2(2) set interval_ 0.03
$cbr2(2) set class_ 1
$cbr2(2) attach-agent $udp2(2)

$ns at 1.0 "$cbr2(1) start"
$ns at 1.0 "$cbr2(2) start"
$ns at 14.0 "$cbr2(1) stop"
$ns at 14.0 "$cbr2(2) stop"

####group chat stuff starts here##########
set sinkGC0 [new Agent/LossMonitor]
set sinkGC1 [new Agent/LossMonitor]
set sinkGC2 [new Agent/LossMonitor]
set sinkGC3 [new Agent/LossMonitor]

$ns attach-agent $n(0) $sinkGC0
$ns attach-agent $n(1) $sinkGC1
```

```
$ns attach-agent $n(3) $sinkGC2
$ns attach-agent $n(4) $sinkGC3

for {set i 0} {$i < 12} {incr i} {
        set udpGC($i) [new Agent/UDP]
}

$ns attach-agent $n(0) $udpGC(0)
$ns attach-agent $n(0) $udpGC(1)
$ns attach-agent $n(0) $udpGC(2)
$ns connect $udpGC(0) $sinkGC1
$ns connect $udpGC(1) $sinkGC2
$ns connect $udpGC(2) $sinkGC3

$ns attach-agent $n(1) $udpGC(3)
$ns attach-agent $n(1) $udpGC(4)
$ns attach-agent $n(1) $udpGC(5)
$ns connect $udpGC(3) $sinkGC0
$ns connect $udpGC(4) $sinkGC2
$ns connect $udpGC(5) $sinkGC3

$ns attach-agent $n(3) $udpGC(6)
$ns attach-agent $n(3) $udpGC(7)
$ns attach-agent $n(3) $udpGC(8)
$ns connect $udpGC(6) $sinkGC0
$ns connect $udpGC(7) $sinkGC1
$ns connect $udpGC(8) $sinkGC3

$ns attach-agent $n(4) $udpGC(9)
$ns attach-agent $n(4) $udpGC(10)
$ns attach-agent $n(4) $udpGC(11)
$ns connect $udpGC(9) $sinkGC0
$ns connect $udpGC(10) $sinkGC1
$ns connect $udpGC(11) $sinkGC2

for {set i 0} {$i < 12} {incr i} {
        set cbrGC($i) [new Application/Traffic/CBR]
        $cbrGC($i) set packetSize_ 480
        $cbrGC($i) set interval_ 0.03
        $cbrGC($i) set class_ $i
        $cbrGC($i) attach-agent $udpGC($i)
        $ns at 15.0 "$cbrGC($i) start"
        $ns at 30.0 "$cbrGC($i) stop"
}
```

```
$ns at 0.0 "record"

proc record {} {
        global sink11 sink12 sink21 sink22 sinkGC0 sinkGC1 sinkGC2 sinkGC3 f0 f1 f2 f3 f4 f5 f6
f7 holdtime holdseq holdtime1 holdseq1 holdtime2 holdseq2 holdtime3 holdseq3 f8 f9 f10 f11
holdrate1 holdrate2 holdrate3 holdrate4 g0 g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 holdtimeg0 holdtimeg1
holdtimeg2 holdtimeg3 holdseqg0 holdseqg1 holdseqg2 holdseqg3 holdrateg0 holdrateg1 holdrateg2
holdrateg3 j0 j1 j2 j3 jg0 jg1 jg2 jg3 previous previous1 previous2 previous3 previousg0 previousg1
previousg2 previousg3 delaynow delaynow1 delaynow2 delaynow3 delaynowg0 delaynowg1
delaynowg2 delaynowg3

        set ns [Simulator instance]

    set time 0.2 ;#Set Sampling Time to 0.9 Sec

        set bw0 [$sinkGC0 set bytes_]
        set bw1 [$sinkGC1 set bytes_]
        set bw2 [$sinkGC2 set bytes_]
        set bw3 [$sinkGC3 set bytes_]

    set bwg0 [$sink11 set bytes_]
        set bwg1 [$sink12 set bytes_]
        set bwg2 [$sink21 set bytes_]
        set bwg3 [$sink22 set bytes_]

        set bw4 [$sinkGC0 set nlost_]
        set bw5 [$sinkGC1 set nlost_]
        set bw6 [$sinkGC2 set nlost_]
        set bw7 [$sinkGC3 set nlost_]

    set bwg4 [$sink11 set nlost_]
        set bwg5 [$sink12 set nlost_]
        set bwg6 [$sink21 set nlost_]
        set bwg7 [$sink22 set nlost_]

        set bw8 [$sinkGC0 set lastPktTime_]
        set bw9 [$sinkGC0 set npkts_]

        set bw10 [$sinkGC1 set lastPktTime_]
        set bw11 [$sinkGC1 set npkts_]

        set bw12 [$sinkGC2 set lastPktTime_]
        set bw13 [$sinkGC2 set npkts_]
```

```
        set bw14 [$sinkGC3 set lastPktTime_]
        set bw15 [$sinkGC3 set npkts_]

set bwg8 [$sink11 set lastPktTime_]
        set bwg9 [$sink11 set npkts_]

        set bwg10 [$sink12 set lastPktTime_]
        set bwg11 [$sink12 set npkts_]

        set bwg12 [$sink21 set lastPktTime_]
        set bwg13 [$sink21 set npkts_]

        set bwg14 [$sink22 set lastPktTime_]
        set bwg15 [$sink22 set npkts_]

if { $bw9 > $holdseq } {
set delaynow [expr ($bw8 - $holdtime)/($bw9 - $holdseq)]
} else {
                set delaynow [expr ($bw9 - $holdseq)]
}

if { $bwg9 > $holdseqg0 } {
set delaynowg0 [expr ($bwg8 - $holdtimeg0)/($bwg9 - $holdseqg0)]
} else { set delaynowg0 [expr ($bwg9 - $holdseqg0)]
}

if { $bw11 > $holdseq1 } {
set delaynow1 [expr ($bw10 - $holdtime1)/($bw11 - $holdseq1)]
} else {
                set delaynow1 [expr ($bw11 - $holdseq1)]
}

if { $bwg11 > $holdseqg1 } {
set delaynowg1 [expr ($bwg10 - $holdtimeg1)/($bwg11 - $holdseqg1)]
} else { set delaynowg1 [expr ($bwg11 - $holdseqg1)]
}

if { $bw13 > $holdseq2 } {
set delaynow2 [expr ($bw12 - $holdtime2)/($bw13 - $holdseq2)]
} else {
                set delaynow2 [expr ($bw13 - $holdseq2)]
}
```

```
if { $bwg13 > $holdseqg2 } {
set delaynowg2 [expr ($bwg12 - $holdtimeg2)/($bwg13 - $holdseqg2)]
} else { set delaynowg2 [expr ($bwg13 - $holdseqg2)]
}

if { $bw15 > $holdseq3 } {
set delaynow3 [expr ($bw14 - $holdtime3)/($bw15 - $holdseq3)]
} else {
                set delaynow3 [expr ($bw15 - $holdseq3)]
}

if { $bwg15 > $holdseqg3 } {
set delaynowg3 [expr ($bwg14 - $holdtimeg3)/($bwg15 - $holdseqg3)]
} else {
        set delaynowg3 [expr ($bwg15 - $holdseqg3)]
}

set now [$ns now]

      # Record Bit Rate in Trace Files
      puts $f0 "$now [expr (($bw0+$holdrate1)*8)/(2*$time*1000000)]"
      puts $f1 "$now [expr (($bw1+$holdrate2)*8)/(2*$time*1000000)]"
      puts $f2 "$now [expr (($bw2+$holdrate3)*8)/(2*$time*1000000)]"
      puts $f3 "$now [expr (($bw3+$holdrate4)*8)/(2*$time*1000000)]"
puts $g0 "$now [expr (($bwg0+$holdrateg0)*8)/(2*$time*1000000)]"
      puts $g1 "$now [expr (($bwg1+$holdrateg1)*8)/(2*$time*1000000)]"
      puts $g2 "$now [expr (($bwg2+$holdrateg2)*8)/(2*$time*1000000)]"
      puts $g3 "$now [expr (($bwg3+$holdrateg3)*8)/(2*$time*1000000)]"

      # Record Packet Loss Rate in File
      puts $f4 "$now [expr $bw4/$time]"
      puts $f5 "$now [expr $bw5/$time]"
      puts $f6 "$now [expr $bw6/$time]"
      puts $f7 "$now [expr $bw7/$time]"
puts $g4 "$now [expr $bwg4/$time]"
      puts $g5 "$now [expr $bwg5/$time]"
      puts $g6 "$now [expr $bwg6/$time]"
      puts $g7 "$now [expr $bwg7/$time]"

      # Record Packet Delay in File
      if { $bw9 > $holdseq } {
                puts $f8 "$now [expr ($bw8 - $holdtime)/($bw9 - $holdseq)]"
      } else {
                puts $f8 "$now [expr ($bw9 - $holdseq)]"
```

```
	}

	if { $bw11 > $holdseq1 } {
		puts $f9 "$now [expr ($bw10 - $holdtime1)/($bw11 - $holdseq1)]"
	} else {
		puts $f9 "$now [expr ($bw11 - $holdseq1)]"
	}

	if { $bw13 > $holdseq2 } {
		puts $f10 "$now [expr ($bw12 - $holdtime2)/($bw13 - $holdseq2)]"
	} else {
		puts $f10 "$now [expr ($bw13 - $holdseq2)]"
	}

	if { $bw15 > $holdseq3 } {
		puts $f11 "$now [expr ($bw14 - $holdtime3)/($bw15 - $holdseq3)]"
	} else {
		puts $f11 "$now [expr ($bw15 - $holdseq3)]"
	}
if { $bwg9 > $holdseqg0 } {
		puts $g8 "$now [expr ($bwg8 - $holdtimeg0)/($bwg9 - $holdseqg0)]"
	} else {
		puts $g8 "$now [expr ($bwg9 - $holdseqg0)]"
	}

	if { $bwg11 > $holdseqg1 } {
		puts $g9 "$now [expr ($bwg10 - $holdtimeg1)/($bwg11 - $holdseqg1)]"
	} else {
		puts $g9 "$now [expr ($bwg11 - $holdseqg1)]"
	}

	if { $bwg13 > $holdseqg2 } {
		puts $g10 "$now [expr ($bwg12 - $holdtimeg2)/($bwg13 - $holdseqg2)]"
	} else {
		puts $g10 "$now [expr ($bwg13 - $holdseqg2)]"
	}

	if { $bwg15 > $holdseqg3 } {
		puts $g11 "$now [expr ($bwg14 - $holdtimeg3)/($bwg15 - $holdseqg3)]"
	} else {
		puts $g11 "$now [expr ($bwg15 - $holdseqg3)]"
	}

	# Record Jitter in Trace Files
```

```
	puts $j0 "$now [expr (($delaynow - $previous ) + ($delaynowg0 - $previousg0))]"
puts $j1 "$now [expr (($delaynow1 - $previous1 ) + ($delaynowg1 - $previousg1))]"
puts $j2 "$now [expr (($delaynow2 - $previous2 ) + ($delaynowg2 - $previousg2))]"
puts $j3 "$now [expr (($delaynow3 - $previous3 ) + ($delaynowg3 - $previousg3))]"

	# Reset Variables
	$sinkGC0 set bytes_ 0
	$sinkGC1 set bytes_ 0
	$sinkGC2 set bytes_ 0
	$sinkGC3 set bytes_ 0

	$sinkGC0 set nlost_ 0
	$sinkGC1 set nlost_ 0
	$sinkGC2 set nlost_ 0
	$sinkGC3 set nlost_ 0

$sink11 set bytes_ 0
	$sink12 set bytes_ 0
	$sink21 set bytes_ 0
	$sink22 set bytes_ 0

	$sink11 set nlost_ 0
	$sink12 set nlost_ 0
	$sink21 set nlost_ 0
	$sink22 set nlost_ 0

	set holdtime $bw8
	set holdseq $bw9
	set holdtime1 $bw10
	set holdseq1 $bw11
	set holdtime2 $bw12
	set holdseq2 $bw13
	set holdtime3 $bw14
	set holdseq3 $bw15

set holdtimeg0 $bwg8
	set holdseqg0 $bwg9
set holdtimeg1 $bwg10
	set holdseqg1 $bwg11
set holdtimeg2 $bwg12
	set holdseqg2 $bwg13
set holdtimeg3 $bwg14
	set holdseqg3 $bwg15
```

```
            set    holdrate1 $bw0
            set    holdrate2 $bw1
            set    holdrate3 $bw2
            set    holdrate4 $bw3

        set    holdrateg0 $bwg0
            set    holdrateg1 $bwg1
            set    holdrateg2 $bwg2
            set    holdrateg3 $bwg3

#### international group chat#####
            if { $bw9 > $holdseq } {
        set previous [expr ($bw8 - $holdtime)/($bw9 - $holdseq)]
        } else { set previous [expr ($bw9 - $holdseq)]
        }

        if { $bw11 > $holdseq1 } {
        set previous1 [expr ($bw10 - $holdtime1)/($bw11 - $holdseq1)]
        } else { set previous1 [expr ($bw11 - $holdseq1)]
        }

        if { $bw13 > $holdseq2 } {
        set previous2 [expr ($bw12 - $holdtime2)/($bw13 - $holdseq2)]
        } else { set previous2 [expr ($bw13 - $holdseq2)]
        }

        if { $bw15 > $holdseq3 } {
        set previous3 [expr ($bw14 - $holdtime3)/($bw15 - $holdseq3)]
        } else { set previous3 [expr ($bw15 - $holdseq3)]
        }

##### international chat#####
        if { $bwg9 > $holdseqg0 } {
        set previousg0 [expr ($bwg8 - $holdtimeg0)/($bwg9 - $holdseqg0)]
        } else { set previousg0 [expr ($bwg9 - $holdseqg0)]
        }

        if { $bwg11 > $holdseqg1 } {
        set previousg1 [expr ($bwg10 - $holdtimeg1)/($bwg11 - $holdseqg1)]
        } else { set previousg1 [expr ($bwg11 - $holdseqg1)]
        }

        if { $bwg13 > $holdseqg2 } {
        set previousg2 [expr ($bwg12 - $holdtimeg2)/($bwg13 - $holdseqg2)]
```

```
        } else { set previousg2 [expr ($bwg13 - $holdseqg2)]
        }

        if { $bwg15 > $holdseqg3 } {
        set previousg3 [expr ($bwg14 - $holdtimeg3)/($bwg15 - $holdseqg3)]
        } else { set previousg3 [expr ($bwg15 - $holdseqg3)]
        }

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

#defining heads
#Color change while moving from one group to another
$ns at 1.0 "$n(0) delete-mark n(0)"
$ns at 1.0 "$n(0) add-mark n(0) yellow circle"

$ns at 1.0 "$n(3) delete-mark n(3)"
$ns at 1.0 "$n(3) add-mark n(3) green circle"

$ns at 15.0 "$n(1) delete-mark n(1)"
$ns at 15.0 "$n(1) add-mark n(1) purple circle"

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 20 defines the node size for nam
$ns initial_node_pos $n($i) 10
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
$ns at $val(stop) "$n($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 35.00 "puts \"end simulation\" ; $ns halt"
proc stop {} {
global ns tracefd namtrace f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 g0 g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11
j0 j1 j2 j3 jg0 jg1 jg2 jg3
$ns flush-trace
close $tracefd
close $namtrace
# Close Trace Files
```

```
close $f0
close $f1
close $f2
close $f3
close $f4
close $f5
close $f6
close $f7
close $f8
close $f9
close $f10
close $f11
close $g0
close $g1
close $g2
close $g3
close $g4
close $g5
close $g6
close $g7
close $g8
close $g9
close $g10
close $g11
close $j0
close $j1
close $j2
close $j3
close $jg0
close $jg1
close $jg2
close $jg3
#exec xgraph jitter01.tr jitter02.tr jitter03.tr jitter04.tr -geometry 800x400 &
#exec xgraph out02.tr out12.tr out22.tr out32.tr -geometry 800x400 &
#exec xgraph outg0.tr outg1.tr outg2.tr outg3.tr -geometry 800x400 &
exec xgraph out02.tr out12.tr out22.tr out32.tr outg0.tr outg1.tr outg2.tr outg3.tr -geometry 800x400 &
#exec xgraph lost02.tr lost12.tr lost22.tr lost32.tr -geometry 800x400 &
#exec xgraph lostg4.tr lostg5.tr lostg6.tr lostg7.tr -geometry 800x400 &
exec xgraph lost02.tr lost12.tr lost22.tr lost32.tr lostg4.tr lostg5.tr lostg6.tr lostg7.tr -geometry 800x400
&
#exec xgraph delay02.tr delay12.tr delay22.tr delay32.tr -geometry 800x400 &
#exec xgraph delayg8.tr delayg9.tr delayg10.tr delayg11.tr -geometry 800x400 &
exec xgraph delay02.tr delay12.tr delay22.tr delay32.tr delayg8.tr delayg9.tr delayg10.tr delayg11.tr
-geometry 800x400 &
```

```
exec nam voip_wifi.nam &
exit 0
}

$ns run
```

NS2 Code for LTE:

```
# *** Throughput Trace ***
set t11 [open out11.tr w]
set t12 [open out12.tr w]
set t13 [open out13.tr w]

set t21 [open out21.tr w]
set t22 [open out22.tr w]
set t23 [open out23.tr w]

set t31 [open out31.tr w]
set t32 [open out32.tr w]
set t33 [open out33.tr w]

# *** Packet Loss Trace ***
set l11 [open lost11.tr w]
set l12 [open lost12.tr w]
set l13 [open lost13.tr w]

set l21 [open lost21.tr w]
set l22 [open lost22.tr w]
set l23 [open lost23.tr w]

set l31 [open lost31.tr w]
set l32 [open lost32.tr w]
set l33 [open lost33.tr w]

# *** Packet Delay Trace ***
set d11 [open delay11.tr w]
set d12 [open delay12.tr w]
set d13 [open delay13.tr w]

set d21 [open delay21.tr w]
set d22 [open delay22.tr w]
set d23 [open delay23.tr w]

set d31 [open delay31.tr w]
set d32 [open delay32.tr w]
set d33 [open delay33.tr w]

# Initialize Flags
set holdtime11 0
set holdseq11 0
```

```
set holdtime12 0
set holdseq12 0

set holdtime13 0
set holdseq13 0

set holdtime21 0
set holdseq21 0

set holdtime22 0
set holdseq22 0

set holdtime23 0
set holdseq23 0

set holdtime31 0
set holdseq31 0

set holdtime32 0
set holdseq32 0

set holdtime33 0
set holdseq33 0

set holdrate11 0
set holdrate12 0
set holdrate13 0

set holdrate21 0
set holdrate22 0
set holdrate23 0

set holdrate31 0
set holdrate32 0
set holdrate33 0

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

#assign the number of nodes for topology 1
set number 3
```

```
$ns color 1 Yellow
$ns color 2 Green
$ns color 3 Blue

#####LTE 1&2 nodes#########
set eNB1 [$ns node]
$eNB1 label "eNB1"
$eNB1 color blue

set aGW1 [$ns node]
$aGW1 label "aGW1"
$aGW1 color red

set eNB2 [$ns node]
$eNB2 label "eNB2"
$eNB2 color blue

set aGW2 [$ns node]
$aGW2 label "aGW2"
$aGW2 color red

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

#LTE1
for { set i 0} {$i < $number} {incr i} {

      set UE1($i) [$ns node]
      $UE1($i) label "UE1voice($i)"
            $UE1($i) color black
}

for {set i 0} {$i < $number} {incr i} {
      $ns simplex-link $UE1($i) $eNB1 500Mb 2ms LTEQueue/ULAirQueue
      $ns simplex-link $eNB1 $UE1($i) 1Gb 2ms LTEQueue/DLAirQueue
}

#LTE 1 links
$ns simplex-link $eNB1 $aGW1 5Gb 10ms LTEQueue/ULS1Queue
$ns simplex-link $aGW1 $eNB1 5Gb 10ms LTEQueue/DLS1Queue
$ns duplex-link $aGW1 $server1 10Gb 50ms DropTail
$ns duplex-link-op $aGW1 $server1 orient right-up
```

```
#LTE2
for { set i 0} {$i < $number} {incr i} {

        set UE2($i) [$ns node]
        $UE2($i) label "UE2voice($i)"
              $UE2($i) color black
}

for {set i 0} {$i < $number} {incr i} {
        $ns simplex-link $UE2($i) $eNB2 500Mb 2ms LTEQueue/ULAirQueue
        $ns simplex-link $eNB2 $UE2($i) 1Gb 2ms LTEQueue/DLAirQueue
}

#LTE 2 links
$ns duplex-link $aGW2 $server1 10Gb 50ms DropTail
$ns simplex-link $eNB2 $aGW2 5Gb 10ms LTEQueue/ULS1Queue
$ns simplex-link $aGW2 $eNB2 5Gb 10ms LTEQueue/DLS1Queue
$ns duplex-link-op $aGW2 $server1 orient left


######LTE 3 nodes#######
set eNB3 [$ns node]
$eNB3 label "eNB3"
$eNB3 color blue

set aGW3 [$ns node]
$aGW3 label "aGW3"
$aGW3 color red

set server3 [$ns node]
$server3 label "server3"
$server3 color green

#LTE3
for { set i 0} {$i < $number} {incr i} {

        set UE3($i) [$ns node]
        $UE3($i) label "UE3voice($i)"
              $UE3($i) color black
}

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

```
        $ns simplex-link $UE3($i) $eNB3 500Mb 2ms LTEQueue/ULAirQueue
        $ns simplex-link $eNB3 $UE3($i) 1Gb 2ms LTEQueue/DLAirQueue
}

#LTE 3 links
$ns simplex-link $eNB3 $aGW3 5Gb 10ms LTEQueue/ULS1Queue
$ns simplex-link $aGW3 $eNB3 5Gb 10ms LTEQueue/DLS1Queue
$ns duplex-link $aGW3 $server3 10Gb 50ms DropTail
$ns duplex-link-op $aGW3 $server3 orient right-down

#Server 1 link Server 2
$ns duplex-link $server1 $server3 100Gb 100ms DropTail

#UDP agents for all UEs
for {set i 0} {$i < $number} {incr i} {
        #LTE 1
        set udp1($i) [new Agent/UDP]
        $ns attach-agent $UE1($i) $udp1($i)
        #LTE 2
        set udp2($i) [new Agent/UDP]
        $ns attach-agent $UE2($i) $udp2($i)
        #LTE 3
        set udp3($i) [new Agent/UDP]
        $ns attach-agent $UE3($i) $udp3($i)
}

#LTE 1 packet colors
$udp1(0) set fid_ 1
$udp1(1) set fid_ 2
$udp1(2) set fid_ 3
#LTE 2 packet colors
$udp2(0) set fid_ 1
$udp2(1) set fid_ 2
$udp2(2) set fid_ 3
#LTE 3 packet colors
$udp3(0) set fid_ 1
$udp3(1) set fid_ 2
$udp3(2) set fid_ 3

#LTE groupchat packet colors

#LTE 1 packets & sinks
for {set i 0} {$i < $number} {incr i} {
        set cbr1($i) [new Application/Traffic/CBR]
```

```
        $cbr1($i) set packetSize_ 480
            $cbr1($i) set interval_ 0.03
        $cbr1($i) set class_ $i+1
        $cbr1($i) attach-agent $udp1($i)
}
set sinkUE11 [new Agent/LossMonitor]
set sinkUE12 [new Agent/LossMonitor]
set sinkUE13 [new Agent/LossMonitor]

$ns attach-agent $UE1(0) $sinkUE11
$ns attach-agent $UE1(1) $sinkUE12
$ns attach-agent $UE1(2) $sinkUE13


#LTE 2 packets & sinks
for {set i 0} {$i < $number} {incr i} {
        set cbr2($i) [new Application/Traffic/CBR]
        $cbr2($i) set packetSize_ 480
            $cbr2($i) set interval_ 0.03
        $cbr2($i) set class_ $i+4
        $cbr2($i) attach-agent $udp2($i)
}
set sinkUE21 [new Agent/LossMonitor]
set sinkUE22 [new Agent/LossMonitor]
set sinkUE23 [new Agent/LossMonitor]

$ns attach-agent $UE2(0) $sinkUE21
$ns attach-agent $UE2(1) $sinkUE22
$ns attach-agent $UE2(2) $sinkUE23

#LTE 3 packets & sinks
for {set i 0} {$i < $number} {incr i} {
        set cbr3($i) [new Application/Traffic/CBR]
        $cbr3($i) set packetSize_ 480
            $cbr3($i) set interval_ 0.03
        $cbr3($i) set class_ $i+8
        $cbr3($i) attach-agent $udp3($i)
}
set sinkUE31 [new Agent/LossMonitor]
set sinkUE32 [new Agent/LossMonitor]
set sinkUE33 [new Agent/LossMonitor]

$ns attach-agent $UE3(0) $sinkUE31
$ns attach-agent $UE3(1) $sinkUE32
```

```
$ns attach-agent $UE3(2) $sinkUE33

## UE1voice(0) to UE2voice(0)
$ns connect $udp1(0) $sinkUE21
$ns connect $udp2(0) $sinkUE11

## UE1voice(1) to UE3voice(1)
$ns connect $udp1(1) $sinkUE32
$ns connect $udp3(1) $sinkUE12

## UE2voice(0) to UE3voice(2) off
##$ns connect $udp3(0) $sinkUE3(2)
##$ns connect $udp3(2) $sinkUE3(0)

## setting up extra udp agent and attach to cbr traffic for group chatting
set udpgroup(0) [new Agent/UDP]
set udpgroup(1) [new Agent/UDP]
set udpgroup(2) [new Agent/UDP]
set udpgroup(3) [new Agent/UDP]
set udpgroup(4) [new Agent/UDP]
set udpgroup(5) [new Agent/UDP]
set udpgroup(6) [new Agent/UDP]
set udpgroup(7) [new Agent/UDP]

#LTE groupchat packet colors
$udpgroup(0) set fid_ 3
$udpgroup(1) set fid_ 3
$udpgroup(2) set fid_ 2
$udpgroup(3) set fid_ 2
$udpgroup(4) set fid_ 3
$udpgroup(5) set fid_ 3
$udpgroup(6) set fid_ 1
$udpgroup(7) set fid_ 1

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
}

$cbrg(0) attach-agent $udpgroup(0)
$cbrg(1) attach-agent $udpgroup(1)
$cbrg(2) attach-agent $udpgroup(2)
```

```
$cbrg(3) attach-agent $udpgroup(3)
$cbrg(4) attach-agent $udpgroup(4)
$cbrg(5) attach-agent $udpgroup(5)
$cbrg(6) attach-agent $udpgroup(6)
$cbrg(7) attach-agent $udpgroup(7)

$ns attach-agent $UE1(2) $udpgroup(0)
$ns attach-agent $UE1(2) $udpgroup(1)
$ns attach-agent $UE2(1) $udpgroup(2)
$ns attach-agent $UE2(1) $udpgroup(3)
$ns attach-agent $UE2(2) $udpgroup(4)
$ns attach-agent $UE2(2) $udpgroup(5)
$ns attach-agent $UE3(0) $udpgroup(6)
$ns attach-agent $UE3(0) $udpgroup(7)

## UE1voice(2) groupchat with UE2voice(1), UE2voice(2) & UE3voice(0)
$ns connect $udp1(2) $sinkUE22
$ns connect $udpgroup(0) $sinkUE23
$ns connect $udpgroup(1) $sinkUE31
$ns connect $udp2(1) $sinkUE13
$ns connect $udpgroup(2) $sinkUE23
$ns connect $udpgroup(3) $sinkUE31
$ns connect $udp2(2) $sinkUE13
$ns connect $udpgroup(4) $sinkUE22
$ns connect $udpgroup(5) $sinkUE31
$ns connect $udp3(0) $sinkUE13
$ns connect $udpgroup(6) $sinkUE22
$ns connect $udpgroup(7) $sinkUE23

$ns at 0.0 "record"

$ns at 1 "$cbr1(0) start"
$ns at 1 "$cbr2(0) start"
$ns at 1 "$cbr3(1) start"
$ns at 1 "$cbr1(1) start"
##$ns at 0.1 "$cbr3(0) start"
##$ns at 0.1 "$cbr3(2) start"
$ns at 14 "$cbr1(0) stop"
$ns at 14 "$cbr2(0) stop"
$ns at 14 "$cbr3(1) stop"
$ns at 14 "$cbr1(1) stop"
##$ns at 14 "$cbr3(0) stop"
##$ns at 14 "$cbr3(2) stop"
```

```
##groupchat

$ns at 15 "$cbr1(2) start"
$ns at 15 "$cbr2(1) start"
$ns at 15 "$cbr2(2) start"
$ns at 15 "$cbr3(0) start"

$ns at 15 "$cbrg(0) start"
$ns at 15 "$cbrg(1) start"
$ns at 15 "$cbrg(2) start"
$ns at 15 "$cbrg(3) start"
$ns at 15 "$cbrg(4) start"
$ns at 15 "$cbrg(5) start"
$ns at 15 "$cbrg(6) start"
$ns at 15 "$cbrg(7) start"

$ns at 29 "$cbr1(2) stop"
$ns at 29 "$cbr2(1) stop"
$ns at 29 "$cbr2(2) stop"
$ns at 29 "$cbr3(0) stop"

$ns at 29 "$cbrg(0) stop"
$ns at 29 "$cbrg(1) stop"
$ns at 29 "$cbrg(2) stop"
$ns at 29 "$cbrg(3) stop"
$ns at 29 "$cbrg(4) stop"
$ns at 29 "$cbrg(5) stop"
$ns at 29 "$cbrg(6) stop"
$ns at 29 "$cbrg(7) stop"
$ns at 30 "stop"

proc record {} {
        global sinkUE11 sinkUE12 sinkUE13 sinkUE21 sinkUE22 sinkUE23 sinkUE31 sinkUE32
sinkUE33 t11 t12 t13 t21 t22 t23 t31 t32 t33 l11 l12 l13 l21 l22 l23 l31 l32 l33 d11 d12 d13 d21 d22
d23 d31 d32 d33 holdrate11 holdrate12 holdrate13 holdrate21 holdrate22 holdrate23 holdrate31
holdrate32 holdrate33 holdtime11 holdtime12 holdtime13 holdtime21 holdtime22 holdtime23
holdtime31 holdtime32 holdtime33 holdseq11 holdseq12 holdseq13 holdseq21 holdseq22 holdseq23
holdseq31 holdseq32 holdseq33

    set ns [Simulator instance]

    set time 0.2 ;#Set Sampling Time to 0.9 Sec

        set bwB11 [$sinkUE11 set bytes_]
```

```
        set bwB12 [$sinkUE12 set bytes_]
        set bwB13 [$sinkUE13 set bytes_]

set bwB21 [$sinkUE21 set bytes_]
        set bwB22 [$sinkUE22 set bytes_]
        set bwB23 [$sinkUE23 set bytes_]

set bwB31 [$sinkUE31 set bytes_]
        set bwB32 [$sinkUE32 set bytes_]
        set bwB33 [$sinkUE33 set bytes_]

        set bwN11 [$sinkUE11 set nlost_]
        set bwN12 [$sinkUE12 set nlost_]
        set bwN13 [$sinkUE13 set nlost_]

set bwN21 [$sinkUE21 set nlost_]
        set bwN22 [$sinkUE22 set nlost_]
        set bwN23 [$sinkUE23 set nlost_]

set bwN31 [$sinkUE31 set nlost_]
        set bwN32 [$sinkUE32 set nlost_]
        set bwN33 [$sinkUE33 set nlost_]

        set bwLPKT11 [$sinkUE11 set lastPktTime_]
        set bwNPK11 [$sinkUE11 set npkts_]

        set bwLPKT12 [$sinkUE12 set lastPktTime_]
        set bwNPK12 [$sinkUE12 set npkts_]

        set bwLPKT13 [$sinkUE13 set lastPktTime_]
        set bwNPK13 [$sinkUE13 set npkts_]

set bwLPKT21 [$sinkUE21 set lastPktTime_]
        set bwNPK21 [$sinkUE21 set npkts_]

        set bwLPKT22 [$sinkUE22 set lastPktTime_]
        set bwNPK22 [$sinkUE22 set npkts_]

        set bwLPKT23 [$sinkUE23 set lastPktTime_]
        set bwNPK23 [$sinkUE23 set npkts_]

set bwLPKT31 [$sinkUE31 set lastPktTime_]
        set bwNPK31 [$sinkUE31 set npkts_]
```

```
	set bwLPKT32 [$sinkUE32 set lastPktTime_]
	set bwNPK32 [$sinkUE32 set npkts_]

	set bwLPKT33 [$sinkUE33 set lastPktTime_]
	set bwNPK33 [$sinkUE33 set npkts_]

set now [$ns now]

	# Record Bit Rate in Trace Files
	puts $t11 "$now [expr (($bwB11+$holdrate11)*8)/(2*$time*1000000)]"
	puts $t12 "$now [expr (($bwB12+$holdrate12)*8)/(2*$time*1000000)]"
	puts $t13 "$now [expr (($bwB13+$holdrate13)*8)/(2*$time*1000000)]"

puts $t21 "$now [expr (($bwB21+$holdrate21)*8)/(2*$time*1000000)]"
	puts $t22 "$now [expr (($bwB22+$holdrate22)*8)/(2*$time*1000000)]"
	puts $t23 "$now [expr (($bwB23+$holdrate23)*8)/(2*$time*1000000)]"

	puts $t31 "$now [expr (($bwB31+$holdrate31)*8)/(2*$time*1000000)]"
	puts $t32 "$now [expr (($bwB32+$holdrate32)*8)/(2*$time*1000000)]"
	puts $t33 "$now [expr (($bwB33+$holdrate33)*8)/(2*$time*1000000)]"

	# Record Packet Loss Rate in File
	puts $l11 "$now [expr $bwN11/$time]"
puts $l12 "$now [expr $bwN12/$time]"
puts $l13 "$now [expr $bwN13/$time]"

puts $l21 "$now [expr $bwN21/$time]"
puts $l22 "$now [expr $bwN22/$time]"
puts $l23 "$now [expr $bwN23/$time]"

puts $l31 "$now [expr $bwN31/$time]"
puts $l32 "$now [expr $bwN32/$time]"
puts $l33 "$now [expr $bwN33/$time]"

	# Record Packet Delay in File
	if { $bwNPK11 > $holdseq11 } {
		puts $d11 "$now [expr ($bwLPKT11 - $holdtime11)/($bwNPK11 - $holdseq11)]"
	} else {
		puts $d11 "$now [expr ($bwNPK11 - $holdseq11)]"
	}

if { $bwNPK12 > $holdseq12 } {
		puts $d12 "$now [expr ($bwLPKT12 - $holdtime12)/($bwNPK12 - $holdseq12)]"
	} else {
```

```
                puts $d12 "$now [expr ($bwNPK12 - $holdseq12)]"
        }

if { $bwNPK13 > $holdseq13 } {
                puts $d13 "$now [expr ($bwLPKT13 - $holdtime13)/($bwNPK13 - $holdseq13)]"
        } else {
                puts $d13 "$now [expr ($bwNPK13 - $holdseq13)]"
        }

if { $bwNPK21 > $holdseq21 } {
                puts $d21 "$now [expr ($bwLPKT21 - $holdtime21)/($bwNPK21 - $holdseq21)]"
        } else {
                puts $d21 "$now [expr ($bwNPK21 - $holdseq21)]"
        }

if { $bwNPK22 > $holdseq22 } {
                puts $d22 "$now [expr ($bwLPKT22 - $holdtime22)/($bwNPK22 - $holdseq22)]"
        } else {
                puts $d22 "$now [expr ($bwNPK22 - $holdseq22)]"
        }

if { $bwNPK23 > $holdseq23 } {
                puts $d23 "$now [expr ($bwLPKT23 - $holdtime23)/($bwNPK23 - $holdseq23)]"
        } else {
                puts $d23 "$now [expr ($bwNPK23 - $holdseq23)]"
        }

if { $bwNPK31 > $holdseq31 } {
                puts $d31 "$now [expr ($bwLPKT31 - $holdtime31)/($bwNPK31 - $holdseq31)]"
        } else {
                puts $d31 "$now [expr ($bwNPK31 - $holdseq31)]"
        }

if { $bwNPK32 > $holdseq32 } {
                puts $d32 "$now [expr ($bwLPKT32 - $holdtime32)/($bwNPK32 - $holdseq32)]"
        } else {
                puts $d32 "$now [expr ($bwNPK32 - $holdseq32)]"
        }

if { $bwNPK33 > $holdseq33 } {
                puts $d33 "$now [expr ($bwLPKT33 - $holdtime33)/($bwNPK33 - $holdseq33)]"
        } else {
                puts $d33 "$now [expr ($bwNPK33 - $holdseq33)]"
        }
```

```
        # Reset Variables
        $sinkUE11 set bytes_ 0
        $sinkUE12 set bytes_ 0
        $sinkUE13 set bytes_ 0

$sinkUE21 set bytes_ 0
        $sinkUE22 set bytes_ 0
        $sinkUE23 set bytes_ 0

$sinkUE31 set bytes_ 0
        $sinkUE32 set bytes_ 0
        $sinkUE33 set bytes_ 0

        $sinkUE11 set nlost_ 0
        $sinkUE12 set nlost_ 0
        $sinkUE13 set nlost_ 0

$sinkUE21 set nlost_ 0
        $sinkUE22 set nlost_ 0
        $sinkUE23 set nlost_ 0

$sinkUE31 set nlost_ 0
        $sinkUE32 set nlost_ 0
        $sinkUE33 set nlost_ 0

        set holdtime11 $bwLPKT11
        set holdseq11 $bwNPK11
        set holdtime12 $bwLPKT12
        set holdseq12 $bwNPK12
        set holdtime13 $bwLPKT13
        set holdseq13 $bwNPK13

        set holdtime21 $bwLPKT21
        set holdseq21 $bwNPK21
        set holdtime22 $bwLPKT22
        set holdseq22 $bwNPK22
        set holdtime23 $bwLPKT23
        set holdseq23 $bwNPK23

        set holdtime31 $bwLPKT31
        set holdseq31 $bwNPK31
        set holdtime32 $bwLPKT32
        set holdseq32 $bwNPK32
```

```
            set holdtime33 $bwLPKT33
            set holdseq33 $bwNPK33

            set    holdrate11 $bwB11
            set    holdrate12 $bwB12
            set    holdrate13 $bwB13

        set    holdrate21 $bwB21
            set    holdrate22 $bwB22
            set    holdrate23 $bwB23

        set    holdrate31 $bwB31
            set    holdrate32 $bwB32
            set    holdrate33 $bwB33

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

# Halting simulation
$ns at 35.00 "puts \"end simulation\" ; $ns halt"

proc stop {} {
        global ns f nf t11 t12 t13 t21 t22 t23 t31 t32 t33 l11 l12 l13 l21 l22 l23 l31 l32 l33 d11 d12 d13
d21 d22 d23 d31 d32 d33
        $ns flush-trace
        close $f
        close $nf
        # Close Trace Files
        close $t11
        close $t12
        close $t13

        close $t21
        close $t22
        close $t23

        close $t31
        close $t32
        close $t33

        close $l11
        close $l12
        close $l13
```

```
        close $l21
        close $l22
        close $l23

        close $l31
        close $l32
        close $l33

        close $d11
        close $d12
        close $d13

        close $d21
        close $d22
        close $d23

        close $d31
        close $d32
        close $d33


        exec xgraph out11.tr out12.tr out13.tr out21.tr out22.tr out23.tr out31.tr out32.tr out33.tr
-geometry 800x400 &

        exec xgraph lost11.tr lost12.tr lost13.tr lost21.tr lost22.tr lost23.tr lost31.tr lost32.tr lost33.tr
-geometry 800x400 &

        exec xgraph delay11.tr delay12.tr delay13.tr delay21.tr delay22.tr delay23.tr delay31.tr delay32.tr
delay33.tr -geometry 800x400 &

        exec nam out.nam &
        exit 0
}

$ns run
```