set opt(chan) Channel/WirelessChannel

set opt(prop) Propagation/TwoRayGround

set opt(netif) Phy/WirelessPhy

set opt(mac) Mac/802\_11

set opt(ifq) Queue/DropTail/PriQueue

set opt(ll) LL ;#Link Layer

set opt(ant) Antenna/OmniAntenna

set opt(x) 500 ;# X dimension of the topography

set opt(y) 500 ;# Y dimension of the topography

set opt(ifqlen) 50 ;# max packet in ifq

set opt(nn) 100

set opt(connections) 50

set opt(stop) 50

set opt(dataRate) [expr 1.0\*256\*8] ;#packet size=256 bytes

set opt(adhocRouting) AODV

set ns\_ [new Simulator]

set topo [new Topography]

set opt(fn) "wireless"

set tracefd [open $opt(fn).tr w]

set namtrace [open $opt(fn).nam w]

$ns\_ trace-all $tracefd

$ns\_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# declare finish program

proc finish {} {

global ns\_ tracefd namtrace

$ns\_ flush-trace

close $tracefd

close $namtrace

#exec nam $namtrace

exit 0

}

# define topology

$topo load\_flatgrid $opt(x) $opt(y)

# Create God(Generate Operations Director): stores table of shortest no of hops from 1 node to another

set god\_ [create-god $opt(nn)]

# define how node should be created

#global node setting

$ns\_ node-config -adhocRouting $opt(adhocRouting) \

-llType $opt(ll) \

-macType $opt(mac) \

-ifqType $opt(ifq) \

-ifqLen $opt(ifqlen) \

-antType $opt(ant) \

-propType $opt(prop) \

-phyType $opt(netif) \

-channelType $opt(chan) \

-topoInstance $topo \

-agentTrace ON \

-movementTrace ON \

-routerTrace ON \

-macTrace ON

# Create the specified number of nodes [$opt(nn)] and "attach" them to the channel.

for {set i 0} {$i < $opt(nn) } {incr i} {

set node\_($i) [$ns\_ node]

$node\_($i) random-motion 1 ;# disable random motion

}

for {set i 0} {$i < $opt(nn) } {incr i} {

$node\_($i) set X\_ [expr rand()\*500]

$node\_($i) set Y\_ [expr rand()\*500]

$node\_($i) set Z\_ 0

}

for {set i 0} {$i < $opt(nn)} {incr i} {

$ns\_ initial\_node\_pos $node\_($i) 20

}

for {set i 0} {$i < $opt(connections)} {incr i} {

#Setup a UDP connection

set udp\_($i) [new Agent/UDP]

$ns\_ attach-agent $node\_($i) $udp\_($i)

set null\_($i) [new Agent/Null]

$ns\_ attach-agent $node\_([expr $i+2]) $null\_($i)

$ns\_ connect $udp\_($i) $null\_($i)

#Setup a CBR over UDP connection

set cbr\_($i) [new Application/Traffic/CBR]

$cbr\_($i) attach-agent $udp\_($i)

$cbr\_($i) set type\_ CBR

$cbr\_($i) set packet\_size\_ 256

$cbr\_($i) set rate\_ $opt(dataRate)

$cbr\_($i) set random\_ false

$ns\_ at 0.0 "$cbr\_($i) start"

$ns\_ at $opt(stop) "$cbr\_($i) stop"

}

# random motion

for {set j 0} {$j < 10} {incr j} {

for {set i 0} {$i < $opt(nn)} {incr i} {

set xx\_ [expr rand()\*$opt(x)]

set yy\_ [expr rand()\*$opt(y)]

set rng\_time [expr rand()\*$opt(stop)]

$ns\_ at $rng\_time "$node\_($i) setdest $xx\_ $yy\_ 15.0" ;

}

}

# Tell nodes when the simulation ends

for {set i 0} {$i < $opt(nn) } {incr i} {

$ns\_ at $opt(stop) "$node\_($i) reset";

}

$ns\_ at $opt(stop) "finish"

$ns\_ run

**1.Running exp7.tcl file:**

Mareena@DESKTOP-I8DD1RQ:~/CodeWork$ ns exp7.tcl

When configured, ns found the right version of tclsh in /usr/bin/tclsh8.6

but it doesn't seem to be there anymore, so ns will fall back on running the first tclsh in your path. The wrong version of tclsh may break the test suites. Reconfigure and rebuild ns if this is a problem.

num\_nodes is set 100

warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl

INITIALIZE THE LIST xListHead

channel.cc:sendUp - Calc highestAntennaZ\_ and distCST\_

highestAntennaZ\_ = 1.5, distCST\_ = 550.0

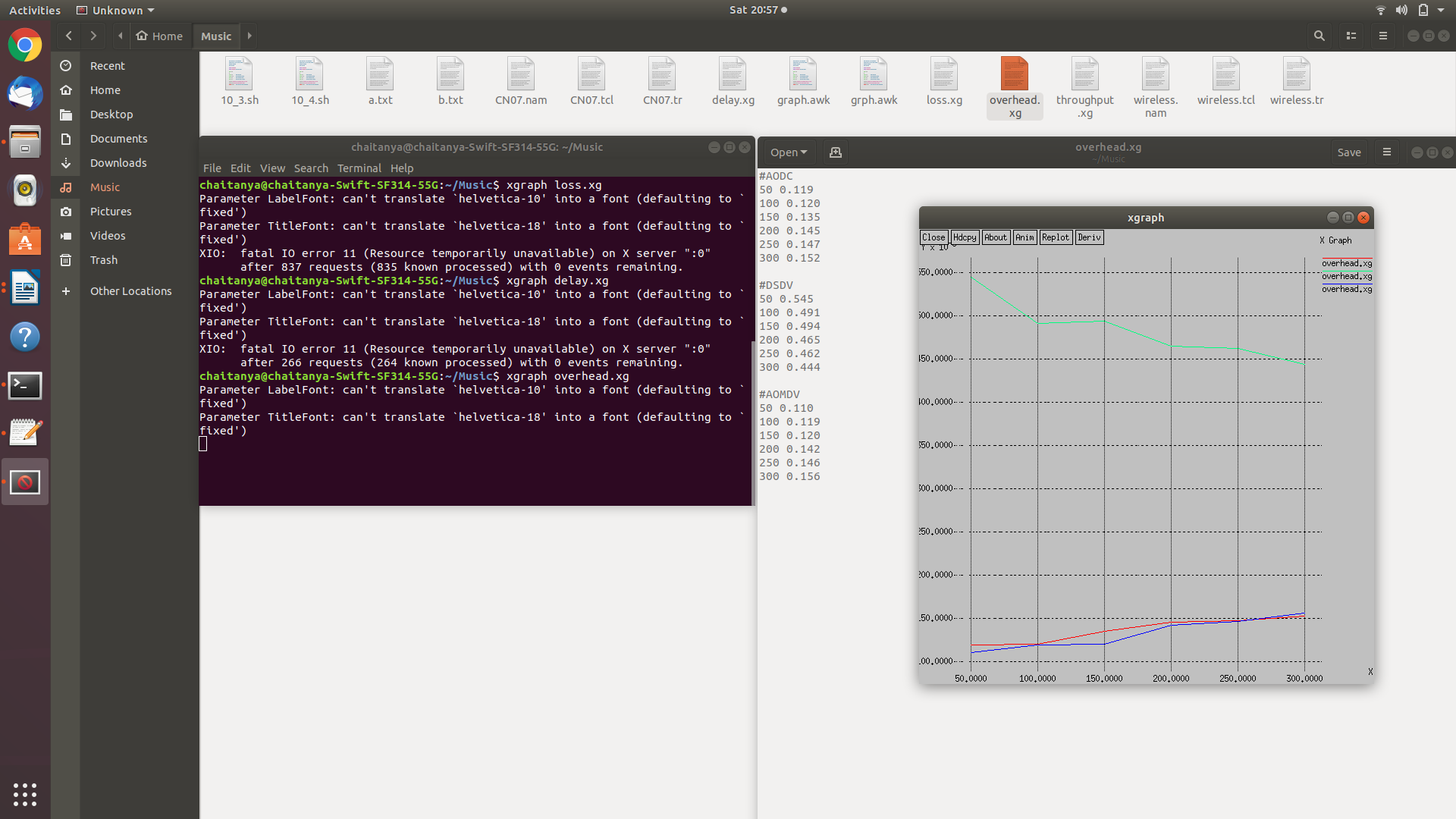
SORTING LISTS ...DONE!

**2. Outputs after running above tcl file:**

2 files are generated as output:

exp7.tf file i.e., Trace File

exp7.nam i.e., NS simulation file

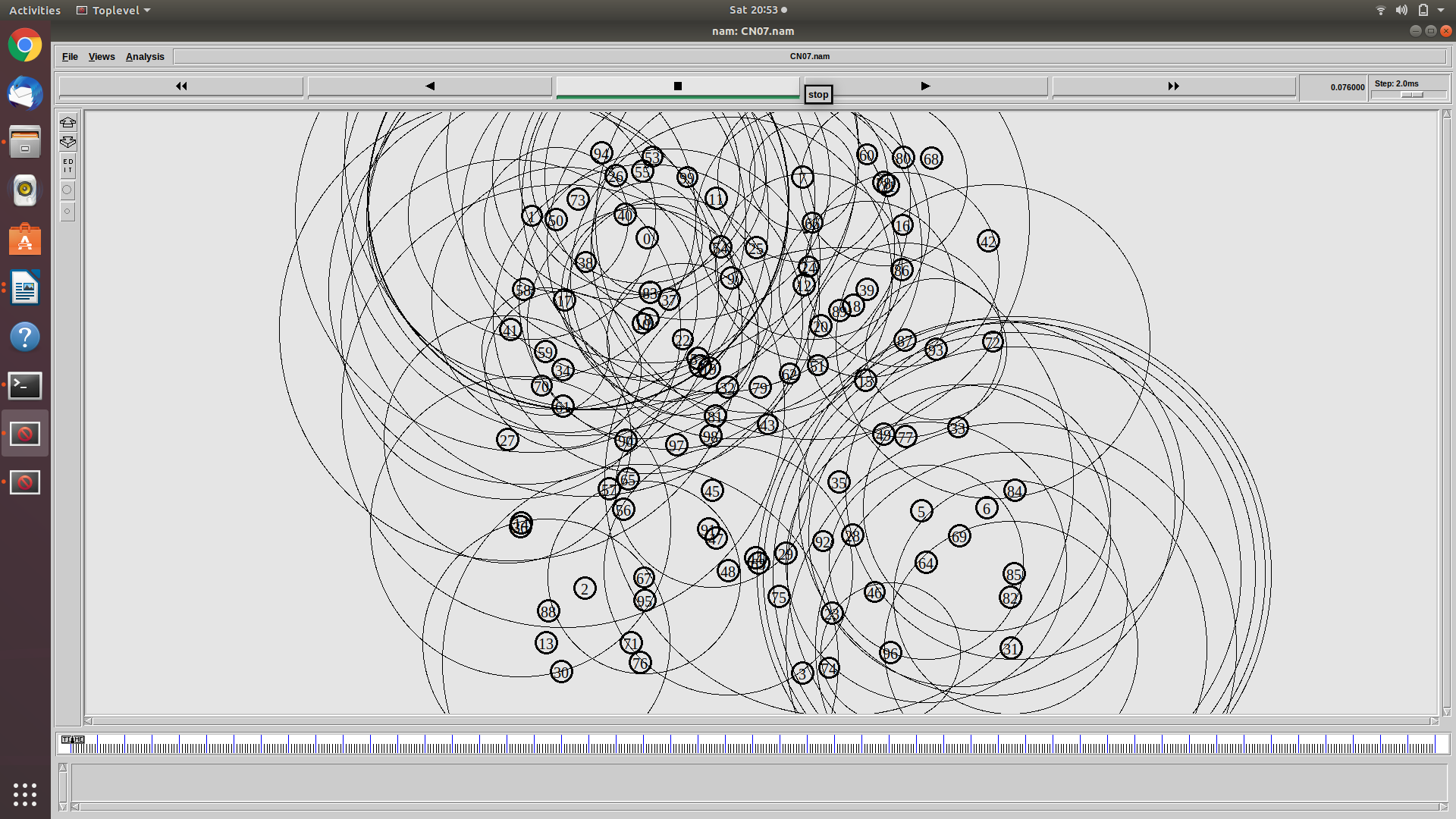


exp7.tr

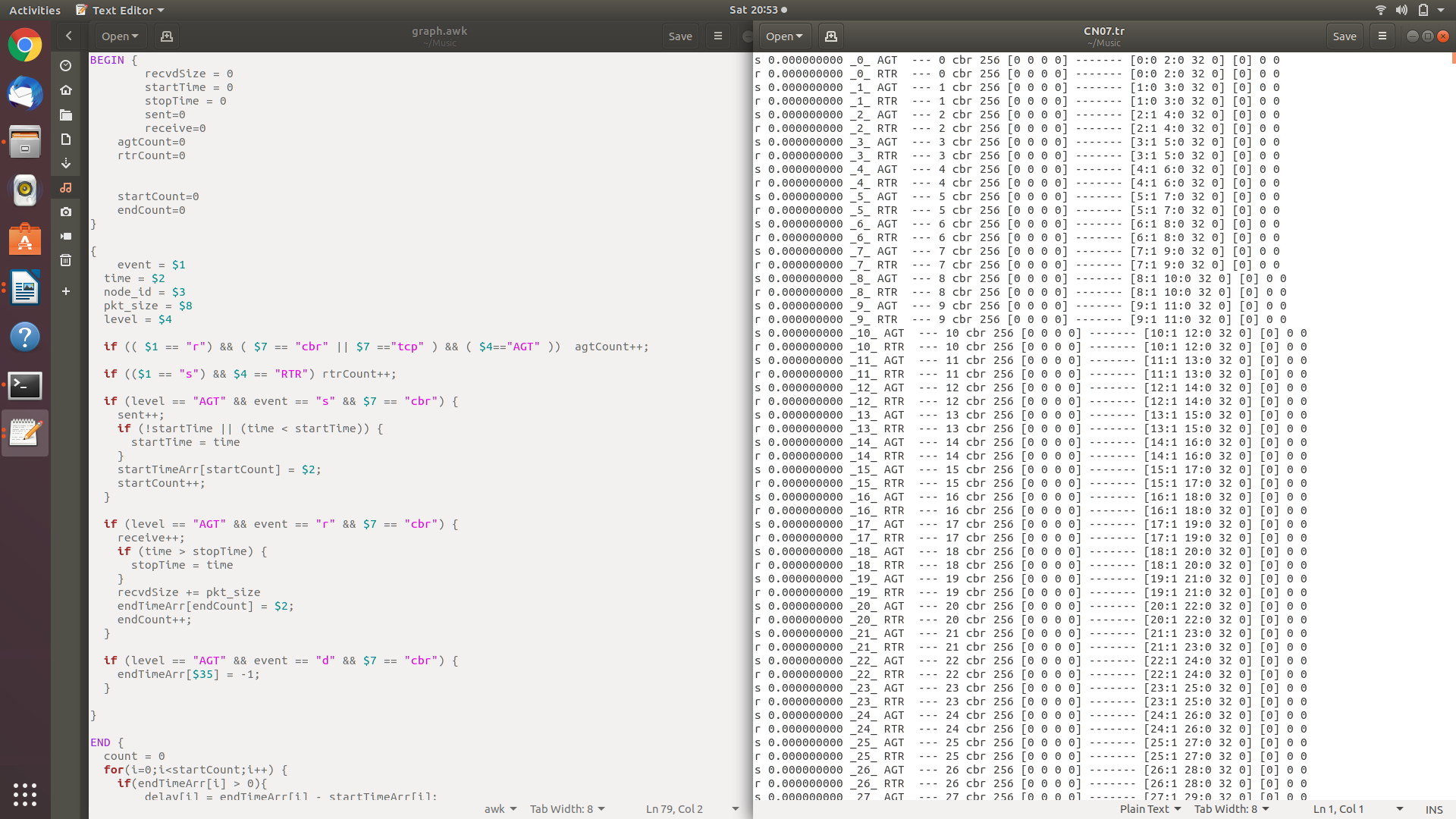
exp7.tcl

exp7.nam

**3. NS simulation (100 nodes ) :**



**4.Trace File and awk file Content :**



**5.AWK command:**

AWK Scripts are very good in processing the data from the log (trace files) which we get from NS2 also if you want to process the trace file manually, they are useful.

Mareena@DESKTOP-I8DD1RQ:~/CodeWork$ awk -f graph.awk exp7.tr

Sent 1900

Received 1594

Dropped 306

PDR 83.89

Average Throughput[kbps] = 97.59 StartTime=1.00 StopTime = 37.06

Normalized Load 0.102

Average End-to-End Delay = 5236.52 ms

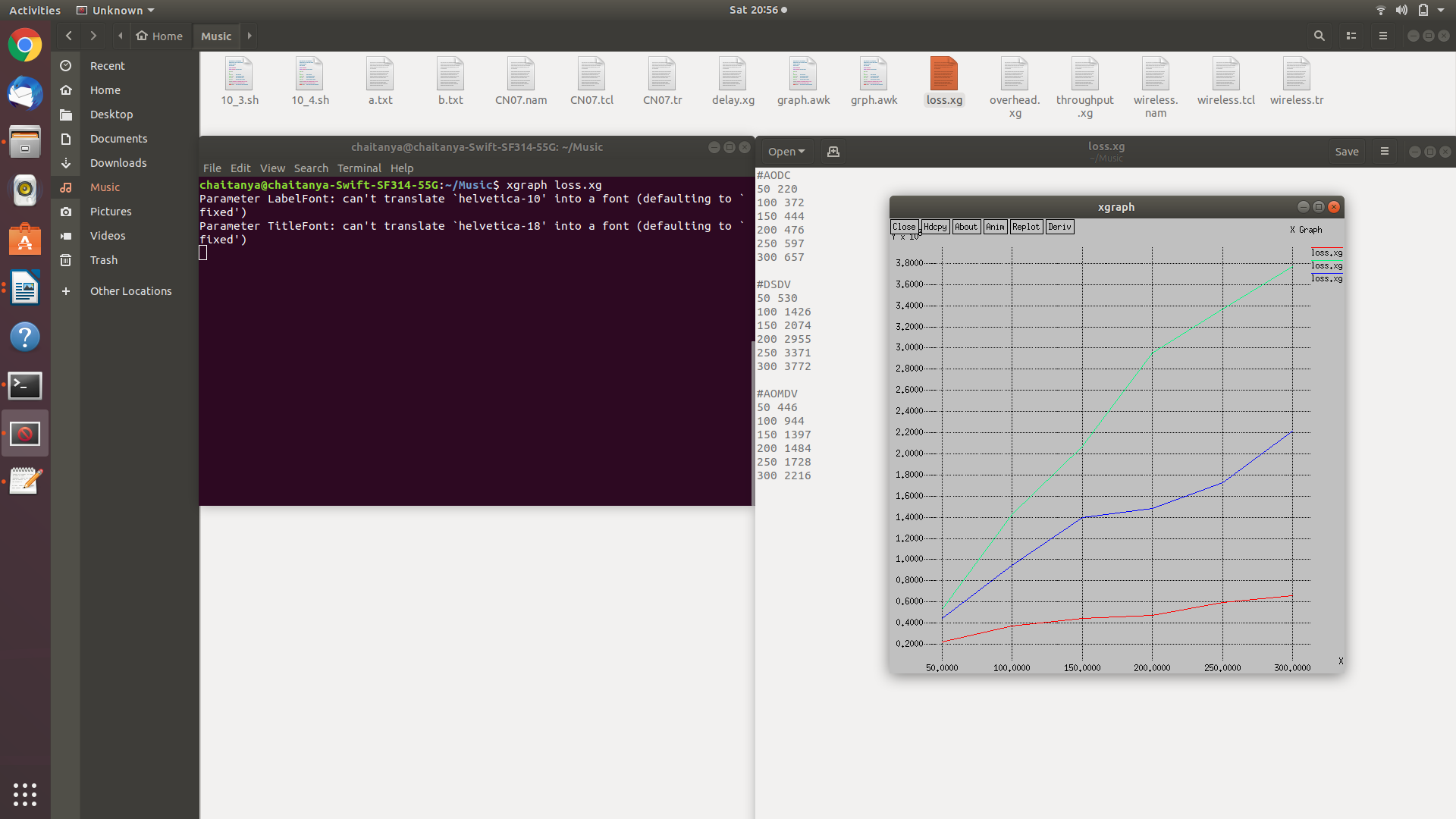
**6.Packet Loss Graph:**

**Packet loss** reflects the number of packets lost per 100 packets sent by a host

Mareena@DESKTOP-I8DD1RQ:~/CodeWork$ xgraph loss.xg

Parameter LabelFont: can't translate `helvetica-10' into a font (defaulting to `fixed')

Parameter TitleFont: can't translate `helvetica-18' into a font (defaulting to `fixed')



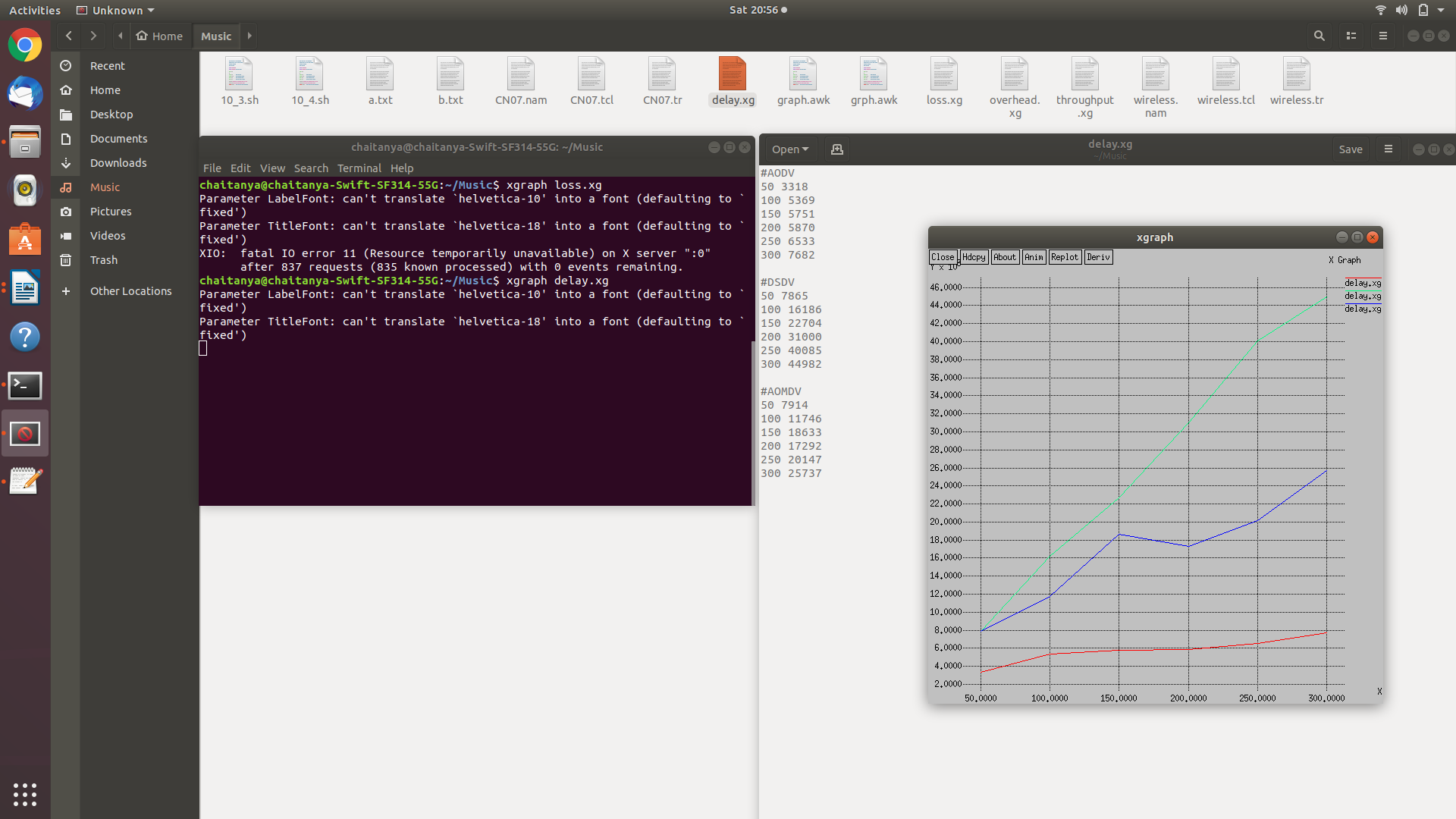
**7.Delay Graph:**

The **delay** of a **network** specifies how long it takes for a bit of data to travel across the **network** from one communication endpoint to another

Mareena@DESKTOP-I8DD1RQ:~/CodeWork$ xgraph delay.xg

Parameter LabelFont: can't translate `helvetica-10' into a font (defaulting to `fixed')

Parameter TitleFont: can't translate `helvetica-18' into a font (defaulting to `fixed')



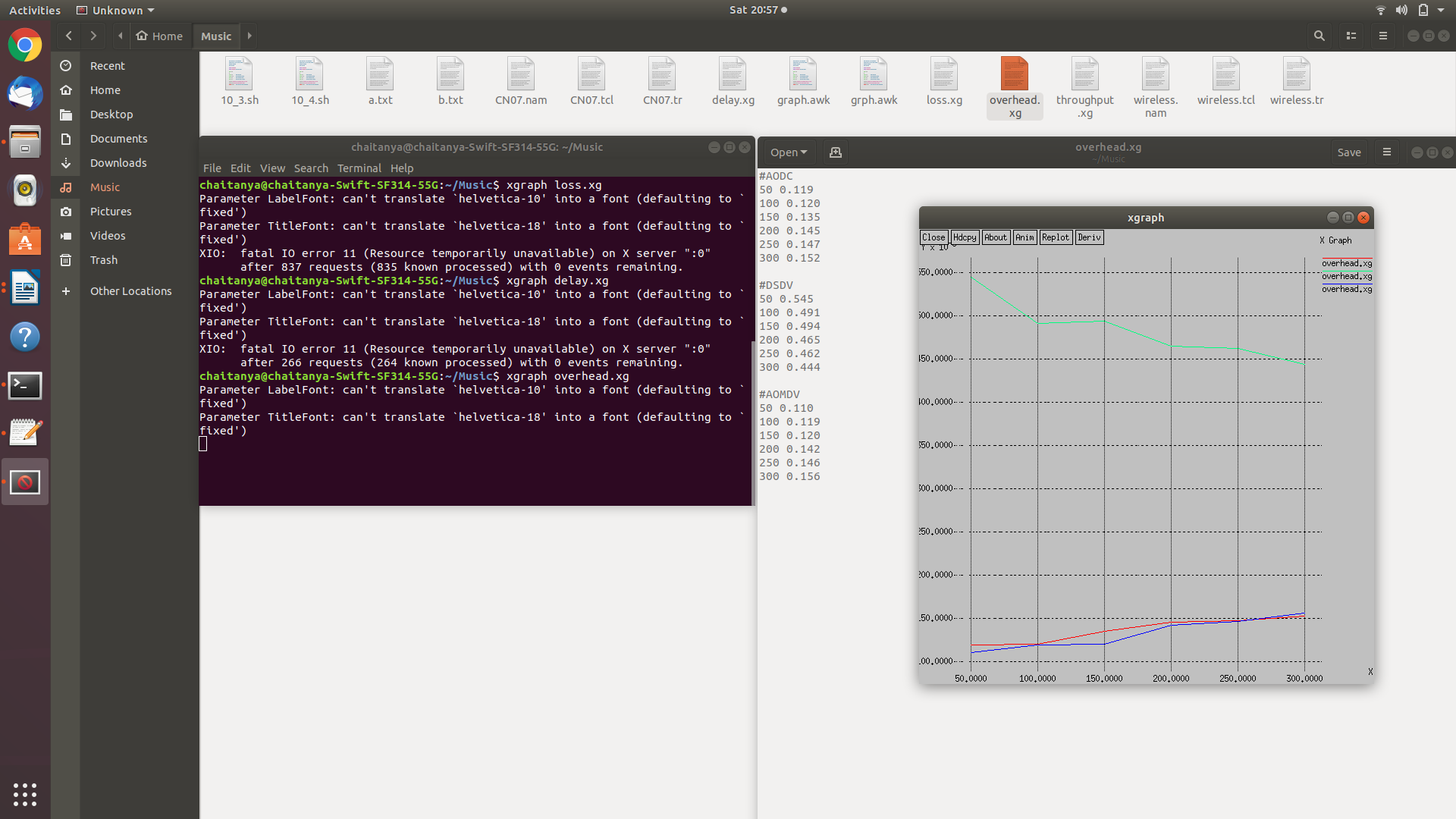
**8.Overhead Graph:**

The time it takes to transmit data on a **packet**-switched network

Mareena@DESKTOP-I8DD1RQ:~/CodeWork$ xgraph overhead.xg

Parameter LabelFont: can't translate `helvetica-10' into a font (defaulting to `fixed')

Parameter TitleFont: can't translate `helvetica-18' into a font (defaulting to `fixed')



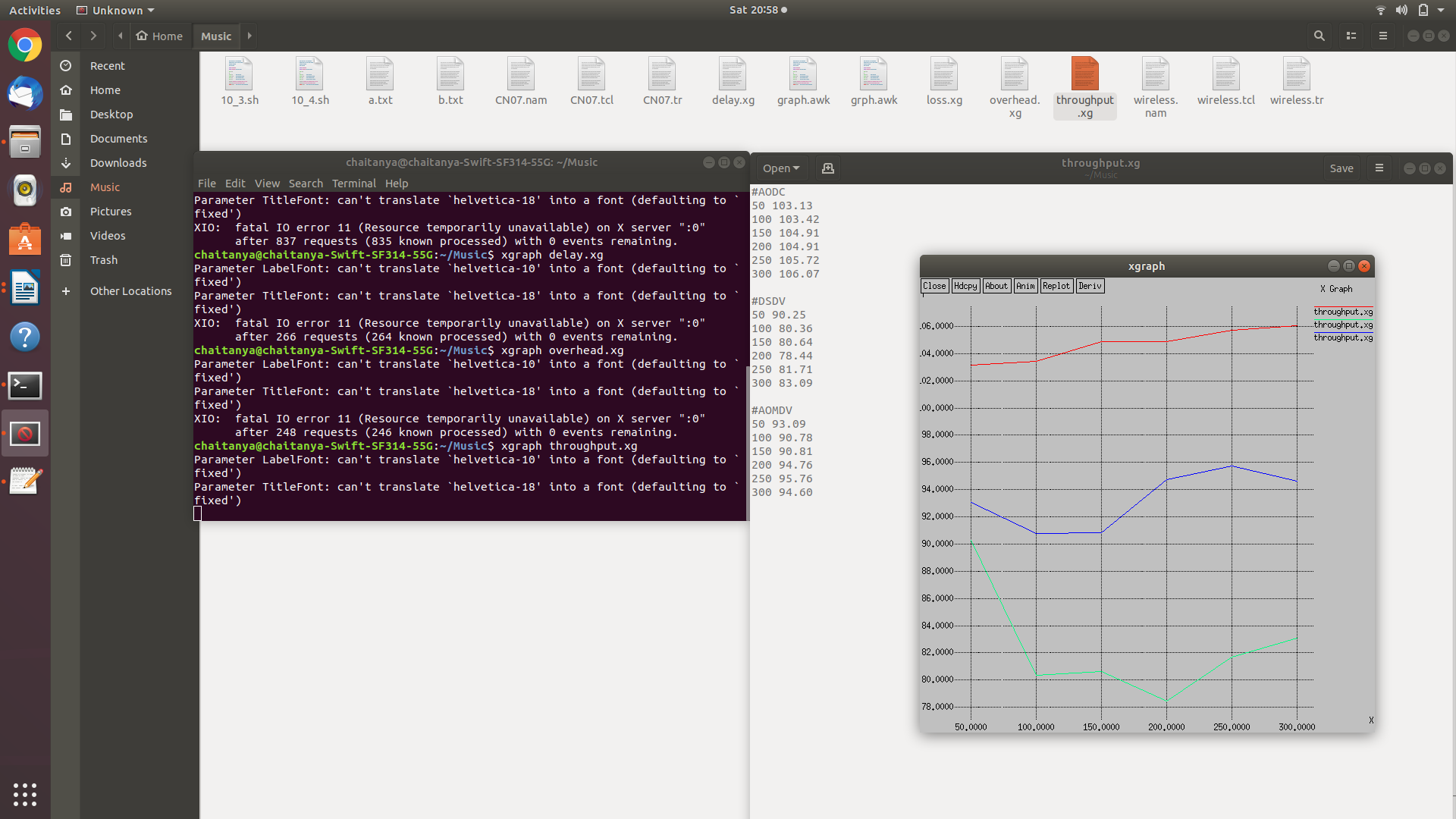
**9.Thoughput Graph:**

**Throughput** is defined as the quantity of data being sent/received by unit of time

Mareena@DESKTOP-I8DD1RQ:~/CodeWork$ xgraph throughput.xg

Parameter LabelFont: can't translate `helvetica-10' into a font (defaulting to `fixed')

Parameter TitleFont: can't translate `helvetica-18' into a font (defaulting to `fixed')



**(All the data passed to draw graphs is sorted out with the help of awk script from Trace File which was generated after running ns filename command in step 2)**

**Conclusion:**

We learnt to analyze Different Routing protocols on **xgraph** as above on the basis of loss packets, Throughput, Delay and Overhead.

Postlabs

Q. Compare different types of routing protocols

A:

Routing Information Protocol (RIP):

Routing Information Protocol or RIP is one of the first routing protocols to be created. RIP is used in both **Local Area Networks** (LANs) and **Wide Area Networks**(WANs), and also runs on the Application layer of the OSI model. There are multiple versions of RIP including **RIPv1** and **RIPv2**. The original version or RIPv1 determines network paths based on the IP destination and the hop count of the journey.

Interior Gateway Protocol (IGRP):

Interior Gateway Protocol or IGRP is a distance vector protocol produced by Cisco. IGRP was designed to build on the foundations laid down on RIP to function more effectively within larger networks and **removed the 15-hop cap**that was placed on RIP. IGRP uses metrics such as bandwidth, delay, reliability, and load to compare the viability of routes within the network. However, only bandwidth and delay are used under IGRP’s default settings.

Open Shortest Path First (OSPF):

Open Shortest Path First or OSPF protocol is a link-state IGP that was tailor-made for IP networks using the **Shortest Path First** (**SPF**) **algorithm**. The SPF algorithm is used to calculate the shortest path spanning-tree to ensure efficient transmission of packets. OSPF routers maintain databases detailing information about the surrounding topology of the network. This database is filled with data taken from **Link State Advertisements (LSAs)** sent by other routers. LSAs are packets that the detail information about how many resources a given path would take.

Exterior Gateway Protocol (EGP):

Exterior Gateway Protocol or EGP is a protocol that is used to exchange data between gateway hosts that neighbor each other within autonomous systems. In other words, EGP provides a forum for routers to share information across different domains. The most high-profile example of an EGP is the internet itself. The routing table of the EGP protocol includes known routers, route costs, and addresses of neighboring devices. EGP was widely-used by larger organizations but has since been replaced by BGP.

Border Gateway Protocol (BGP):

**Border Gateway Protocol or BGP** is the routing protocol of the internet that is classified as a distance path vector protocol. BGP was **designed to replace EGP** with a decentralized approach to routing. The BGP Best Path Selection Algorithm is used to select the best routes for packet transfers. If you don’t have any custom settings then BGP will select routes with the shortest path to the destination