Advanced Communication Systems: CS 258

Project 2 Tutorial

Main Topic: Addressing Social Cloud Issues

Sub Topic: Simulate Cloud in Mininet and Scale for Optimized Content Replication and Distribution

Team members:
Anumeha Shah
Neha Rajkumar
Shruti Sharma

Table of Contents

1. What is a Geo-distributed cloud?

A geo-distributed cloud infrastructure is a combination of different cloud sites which are dispersed across geographical locations and owned by different providers. Each cloud site resides in one data center and contains a collection of interconnected and virtualized servers.

Geo Distributed clouds provide following services for efficient scaling of social media applications

- Automatic on demand provisioning of servers provides infinite on demand cloud resources.
- Infinite capacity to meet ever increasing demand for storage and bandwidth.
- It can serve millions of requests to the media application very efficiently.
- Low latency in streaming video contents.
- Geo Distributed cloud can provide efficient services to the users in proximity.
- Less management effort.
- Low operational cost.

2. Challenges in Implementing the Geo distributed Clouds with Social Media Applications

- How to migrate and store the contents efficiently in between different cloud sites to serve user's demand cost effectively and with low latency.

- A user from India may demand to view a video which is stored in cloud sites in America. If the cloud site in America serves the requests, it may cause some propagation delay which is directly proportional to length. They may also be not limited to only one user. In this scenario the best approach is to replicate and migrate the content to some cloud site in India or near India.

- How to design an algorithm which can store the content and also distribute the contents dynamically based on demand and associated cost.

3. Proposed Solutions

Firstly, anticipate future demands of the media contents in a social application based on user's social relationships and the correlation or association among the media contents.

Secondly, one shot optimal content migration and request distribution strategy algorithm formulated to serve the anticipated content requests.

Thirdly, a delta (t) step look-ahead mechanism is proposed to formulate offline optimality algorithm based on one-shot optimization results, which can be used towards the online algorithm for optimal content distribution and migration of content with reduced costs.

4. Project Design

In the existing solution a network was created using Mininet as a network simulator. The network consists of two switches, 8 hosts which are controlled by a remote controller, where hosts h1,h2,h3,h4 are under switch1 and h5,h6,h7,h8 are under switch2. The simulator used here is Mininet. The Amazon AWS account was created to store data files for the network. Files were stored in the cloud to demonstrate the advantage of combining cloud computing in social networks. Buckets were created in the cloud to further arrange the data for each host. The Mininet was then connected to the Amazon cloud to access the files. This removed the overhead of each host storing the files in the network.

Through Mininet the files could be uploaded and downloaded to and from the cloud. The provision for audio streaming was implemented to check how an audio file can be transferred between hosts. Traffic is generated between the different hosts to monitor the performance characteristics like delay and bandwidth. The performance of the hosts is monitored by sending output to the multiple files. The average ping time in between hosts and the number of requests are calculated and stored in a log file. The streaming cost for each host is also calculated here.

The system also allows checking the access times of all the files under each host. So a file with the least access time which is not in demand was deleted for further optimization. In the system if a particular file is required by a host, then the file is obtained from another host which is closer to them. So by simulating the network it was found that the cost was reduced while getting a file from nearby host rather than from the cloud or a farther host.

5. Development Environment

- Mininet for simulating the networks
- Amazon S3(Simple Storage Service) for simulating the Social Cloud
- Amazon CLI : Amazon Command Line interface to integrate Mininet with Cloud
- Lynx : To simulate browser in mininet
- Netcat : To transfer files between hosts in the mininet network
- Mpg123 : To play the audio files shared by a friend.
- ImageMagicK : For displaying the graph plotted by mininet.
- Matplotlib : Is a python 2D plotting library which is used to display the performance of the system.

5.1Amazon S3 (Simple Storage Service)

- Simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web.
- It gives any developer access to the same highly scalable, reliable, secure, inexpensive infrastructure that Amazon uses to run its own web sites.
- To sign up for an AWS account go to http://aws.amazon.com, and then click Sign Up.

5.2 Amazon CLI (Command Line Interface)

- The AWS Command Line Interface is a unified tool to manage your AWS services.
- With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

- We have integrated Amazon CLI with Mininet to simulate the Social Cloud scenario.
- Download Amazon CLI from the link, unzip and run the install command.

```
$ wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip
$ unzip awscli-bundle.zip
$ sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
```

- AWS was configured using $ aws configure command

5.3 Lynx browser for Mininet

- Text-based Web Browser which is highly configurable.
- Install lynx : apt-get install lynx-cur

5.4Netcat for file transfer

- Netcat is a computer networking service for reading from and writing to network connections using TCP or UDP.
- Netcat is designed to be a dependable back-end that can be used directly or easily driven by other programs and scripts.

5.5 Mpg123 for music streaming

- mpg123 is a free and open-source audio player.
- As the name suggests, it supports MPEG audio formats, including MP2, andMP3.
- It is a console application, meaning that it has no graphical user interface.

5.6 ImageMagicK for displaying the graph plotted in mininet

- ImageMagick is a software suite to create, edit, compose, or convert bitmap images.
- The functionality of ImageMagick is typically utilized from the command line
- Run this command in mininet and install ImageMagicK:sudo apt-get install imagemagick

5.7 Matplotlib for plotting graph at runtime in mininet

- Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- Matplotlib can be used in python scripts. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.
- It provides an object-oriented API for embedding plots into applications. It is a procedural "pylab" interface designed to closely relate to that of MATLAB.
- In this project we are using it to display the performance analysis of the entire system by plotting the number of hosts versus the cost of fetching files from each host.

6. Performance evaluation results

- Simulate the cloud in the Mininet and store content in the cloud.
- Allow the host to get the content from the cloud using the AWS Command Line.
- Generating traffic in between different hosts.
- Monitoring the various hosts by sending output to the multiple files.
- Calculating the average ping time in between different hosts depending on the number of requests by the user.
- Calculating the streaming cost for each host from host h1
- Storing the calculated cost to result.txt for further actions.

7. Test Cases

- The number of requests was given as 5 and the log file, result.txt was analyzed.
- ***Request time in between h1 and h2 is 1.99 and cost of streaming data is 9.95 USD***
- ***Request time in between h1 and h3 is 1.9 and cost of streaming data is 9.5 USD***
- ***Request time in between h1 and h4 is 2.37 and cost of streaming data is 11.85 USD***
- ***Request time in between h1 and h5 is 67.8 and cost of streaming data is 339.0 USD***
- ***Request time in between h1 and h6 is 67.5 and cost of streaming data is 337.5 USD***
- ***Request time in between h1 and h7 is 77.0 and cost of streaming data is 385.0USD***
- ***Request time in between h1 and h8 is 65.0 and cost of streaming data is 325.0 USD***
- The cost of streaming data between h1 and h2 was found to be 9.95 USD while the cost of streaming data between h1 and h8 was found to be 325 USD. Thus it was clearly shown that the cost can be reduced by getting a file from a nearby host rather than by a farther one.

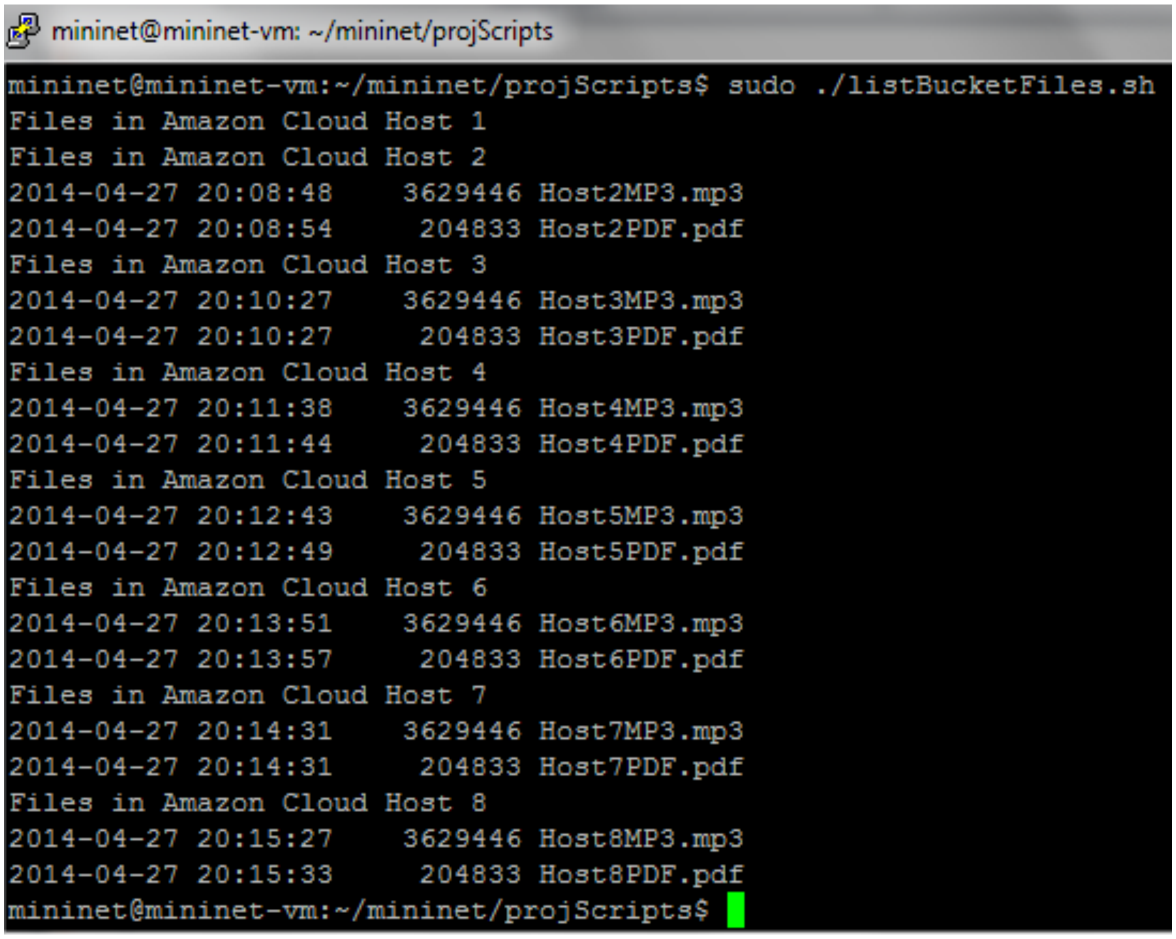
8. Description of the Enhanced Solution

- Simulate the cloud in the Mininet and store content in the cloud.
- Simulate hosts to make requests to the content stored.
- Optimize the content distribution and replication algorithm.
- Test the implementation for traffic load and delay.

9. Demo Steps

i. Listing All files currently present in amazon s3

- First we will list the files that are present in all the buckets
- Example CLI : aws s3 ls s3://shrutihost1 : Will list the number of files in Host1 on the Cloud.
- We will execute the shell script : ./lishBucketFiles.sh to display the list of files in each of the hosts.

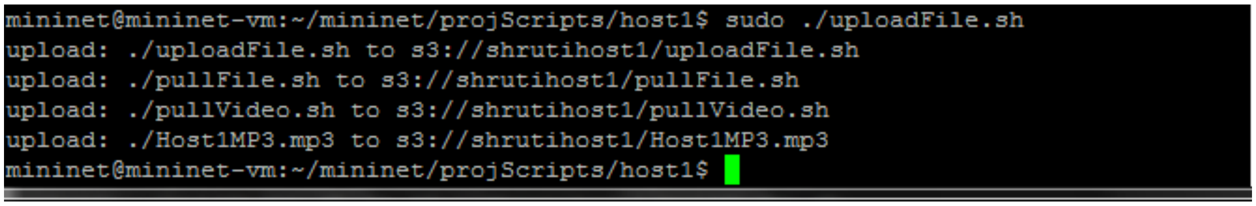


```
mininet@mininet-vm: ~/mininet/projScripts
mininet@mininet-vm:~/mininet/projScripts$ sudo ./listBucketFiles.sh
Files in Amazon Cloud Host 1
Files in Amazon Cloud Host 2
2014-04-27 20:08:48    3629446 Host2MP3.mp3
2014-04-27 20:08:54     204833 Host2PDF.pdf
Files in Amazon Cloud Host 3
2014-04-27 20:10:27    3629446 Host3MP3.mp3
2014-04-27 20:10:27     204833 Host3PDF.pdf
Files in Amazon Cloud Host 4
2014-04-27 20:11:38    3629446 Host4MP3.mp3
2014-04-27 20:11:44     204833 Host4PDF.pdf
Files in Amazon Cloud Host 5
2014-04-27 20:12:43    3629446 Host5MP3.mp3
2014-04-27 20:12:49     204833 Host5PDF.pdf
Files in Amazon Cloud Host 6
2014-04-27 20:13:51    3629446 Host6MP3.mp3
2014-04-27 20:13:57     204833 Host6PDF.pdf
Files in Amazon Cloud Host 7
2014-04-27 20:14:31    3629446 Host7MP3.mp3
2014-04-27 20:14:31     204833 Host7PDF.pdf
Files in Amazon Cloud Host 8
2014-04-27 20:15:27    3629446 Host8MP3.mp3
2014-04-27 20:15:33     204833 Host8PDF.pdf
mininet@mininet-vm:~/mininet/projScripts$
```

Fig 1: Listing all the files in mininet hosts

ii. Uploading files to amazon s3

- We can perform recursive uploads and downloads of multiple files in a single command.
- Every host has a file uploadFile.sh which will recursively upload all the files present in the host to the cloud

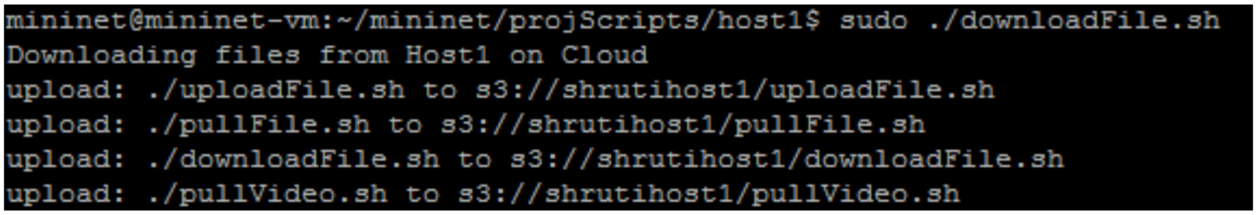


```
mininet@mininet-vm:~/mininet/projScripts/host1$ sudo ./uploadFile.sh
upload: ./uploadFile.sh to s3://shrutihost1/uploadFile.sh
upload: ./pullFile.sh to s3://shrutihost1/pullFile.sh
upload: ./pullVideo.sh to s3://shrutihost1/pullVideo.sh
upload: ./Host1MP3.mp3 to s3://shrutihost1/Host1MP3.mp3
mininet@mininet-vm:~/mininet/projScripts/host1$
```

Fig 2: Uploading files to Amazon S3

iii. Downloading files recursively from amazon S3

- A sync command makes it easy to synchronize the contents of a local folder with a copy in an S3 bucket.
- Example : aws s3 sync . s3://shrutihost1;
- This command will sync the current folder with all the files in the bucket named shrutihost1



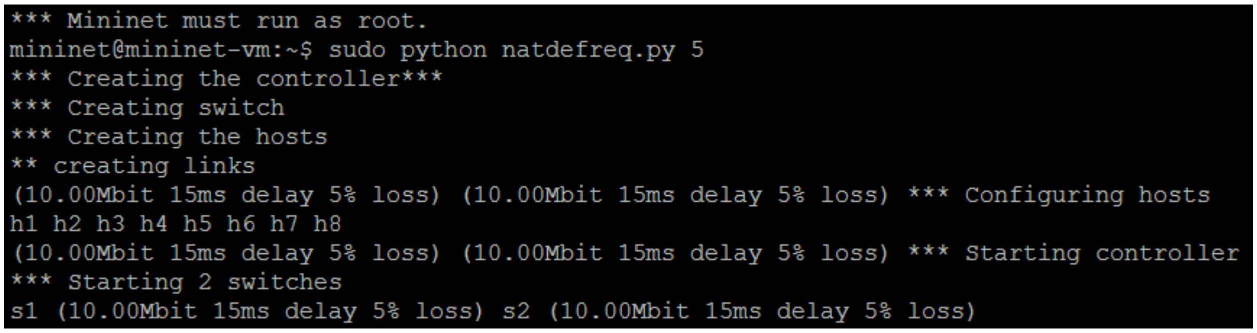
```
mininet@mininet-vm:~/mininet/projScripts/host1$ sudo ./downloadFile.sh
Downloading files from Host1 on Cloud
upload: ./uploadFile.sh to s3://shrutihost1/uploadFile.sh
upload: ./pullFile.sh to s3://shrutihost1/pullFile.sh
upload: ./downloadFile.sh to s3://shrutihost1/downloadFile.sh
upload: ./pullVideo.sh to s3://shrutihost1/pullVideo.sh
```

Fig 3: Downloading files from Amazon S3 recursively

iv. Creating complex network in mininet and enabling hosts to connect to outside world

- NAT (Network address translation) needs to be configured for each host in the network.
- This will ensure that each of the hosts can communicate with the cloud and also with other hosts on the network.
- How did we achieve this?

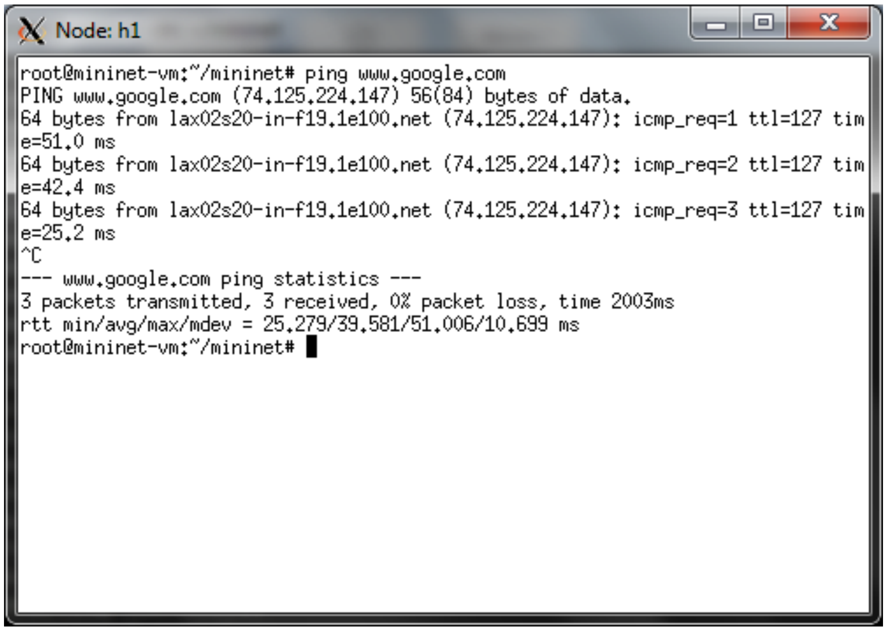
  We have written a python script natdefreq.py : It creates a default gateway for all the hosts as root-eth0, and then uses iptables to allow natting to work.



```
*** Mininet must run as root.
mininet@mininet-vm:~$ sudo python natdefreq.py 5
*** Creating the controller***
*** Creating switch
*** Creating the hosts
** creating links
(10.00Mbit 15ms delay 5% loss) (10.00Mbit 15ms delay 5% loss) *** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
(10.00Mbit 15ms delay 5% loss) (10.00Mbit 15ms delay 5% loss) *** Starting controller
*** Starting 2 switches
s1 (10.00Mbit 15ms delay 5% loss) s2 (10.00Mbit 15ms delay 5% loss)
```

Fig 4: Specifying Bandwidth, delay and creating a complex network in mininet

- Example : Host1 can ping www.google.com



```
root@mininet-vm:~/mininet# ping www.google.com
PING www.google.com (74.125.224.147) 56(84) bytes of data.
64 bytes from lax02s20-in-f19.1e100.net (74.125.224.147): icmp_req=1 ttl=127 tim
e=51.0 ms
64 bytes from lax02s20-in-f19.1e100.net (74.125.224.147): icmp_req=2 ttl=127 tim
e=42.4 ms
64 bytes from lax02s20-in-f19.1e100.net (74.125.224.147): icmp_req=3 ttl=127 tim
e=25.2 ms
^C
--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 25.279/39.581/51.006/10.699 ms
root@mininet-vm:~/mininet# 
```

Fig 5: Displaying each host is able to connect to the outside network by using NAT functionality

- Network that we have created is in the below diagram.

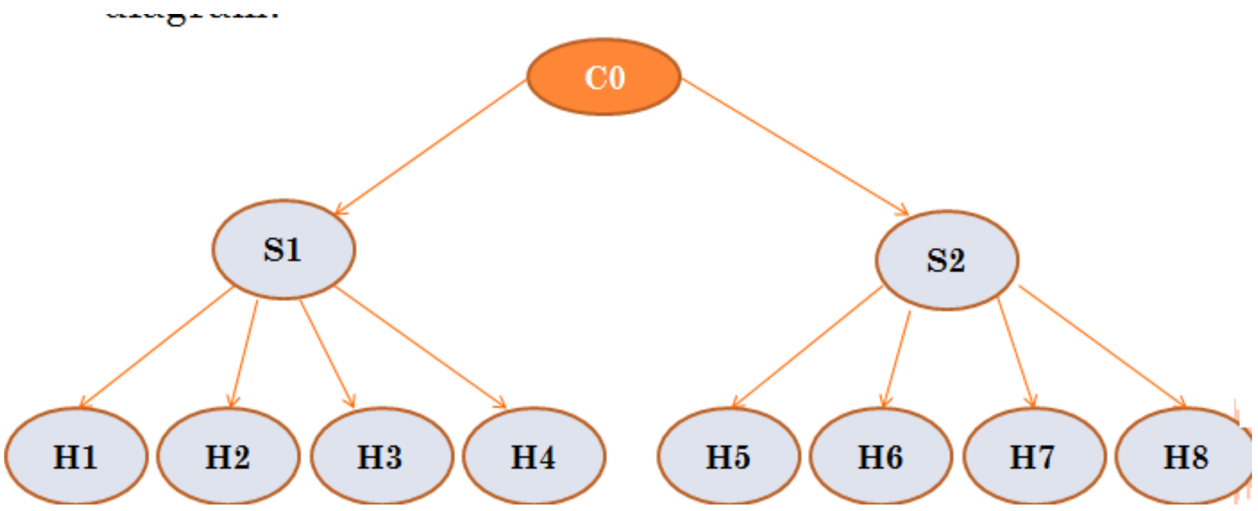
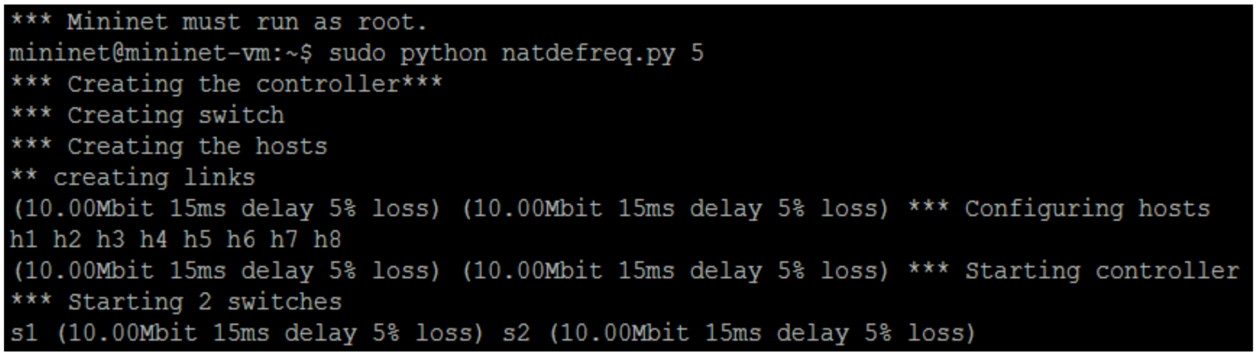


Fig 6: Diagrammatic representation of the network created in mininet

v. Setting delay, bandwidth and packet loss

- In order to simulate network conditions, we have set the Delay, bandwidth and packet loss while creating the network.
- This has been done in the natdefreq.py file.
- Delay is 15ms, Packet loss is 5%

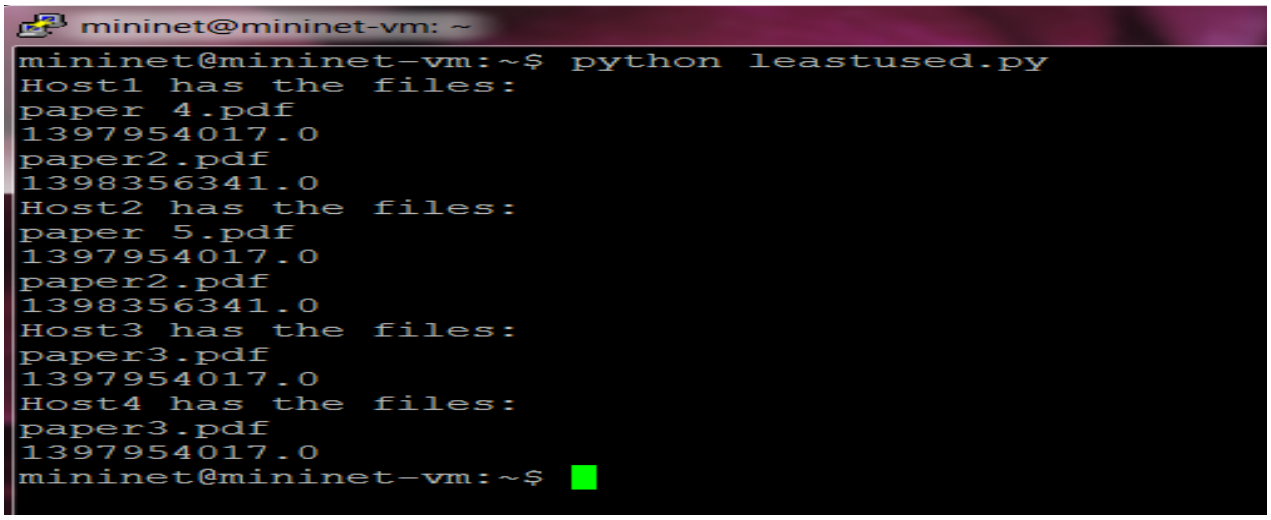


```
*** Mininet must run as root.
mininet@mininet-vm:~$ sudo python natdefreq.py 5
*** Creating the controller***
*** Creating switch
*** Creating the hosts
** creating links
(10.00Mbit 15ms delay 5% loss) (10.00Mbit 15ms delay 5% loss) *** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
(10.00Mbit 15ms delay 5% loss) (10.00Mbit 15ms delay 5% loss) *** Starting controller
*** Starting 2 switches
s1 (10.00Mbit 15ms delay 5% loss) s2 (10.00Mbit 15ms delay 5% loss)
```

Fig 7: Setting of packet loss, delay and bandwidth in network creation

vi. Optimizing disk space : Find the files present in each host and access time

- We have written an algorithm to find the files present in each host and the time when each of these files was accessed.
- The leastused.py will list out all the files with .txt,.pdf,.mp3 formats.
- We will be using this to determine which of the files have not been used/accessed and can be removed from the hosts.
- The file with the largest time since accessed can be deleted.



```
mininet@mininet-vm: ~
mininet@mininet-vm:~$ python leastused.py
Host1 has the files:
paper 4.pdf
1397954017.0
paper2.pdf
1398356341.0
Host2 has the files:
paper 5.pdf
1397954017.0
paper2.pdf
1398356341.0
Host3 has the files:
paper3.pdf
1397954017.0
Host4 has the files:
paper3.pdf
1397954017.0
mininet@mininet-vm:~$
```

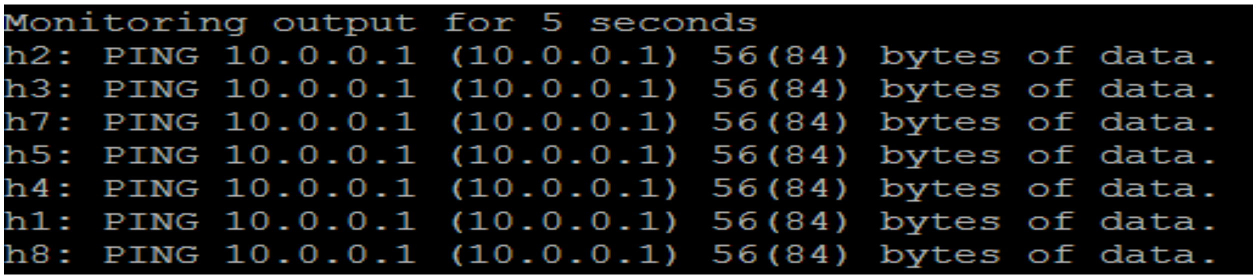
Fig 8: Optimizing the disk space by listing out the files present in each host

vii. Generating Traffic by implementing multiple pings to different hosts

- We have created a multiple ping functionality to generate traffic and calculate the cost to reach each host.
- Traffic is generated between host h1 and all the other hosts. We are also calculating the average ping time in between all the hosts and storing that value in a file for further analysis
- Time is very important performance measuring factor and we are using time as one of the cost calculation factor to determine whether a file should be replicated from a server or not
- Example h1 needs a file abc.pdf. It will ping all the hosts in the network h2, h3, h4 etc. to find the minimum cost to fetch the file.

viii. Monitoring the output for certain duration

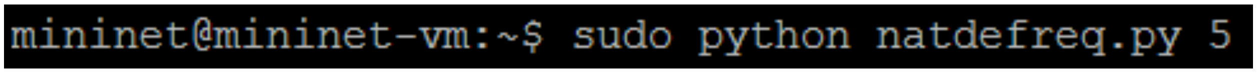
- We will observe the ping across hosts for certain duration of time.



```
Monitoring output for 5 seconds
h2: PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
h3: PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
h7: PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
h5: PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
h4: PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
h1: PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
h8: PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
```

Fig 9: Result of ping monitoring for 5 seconds

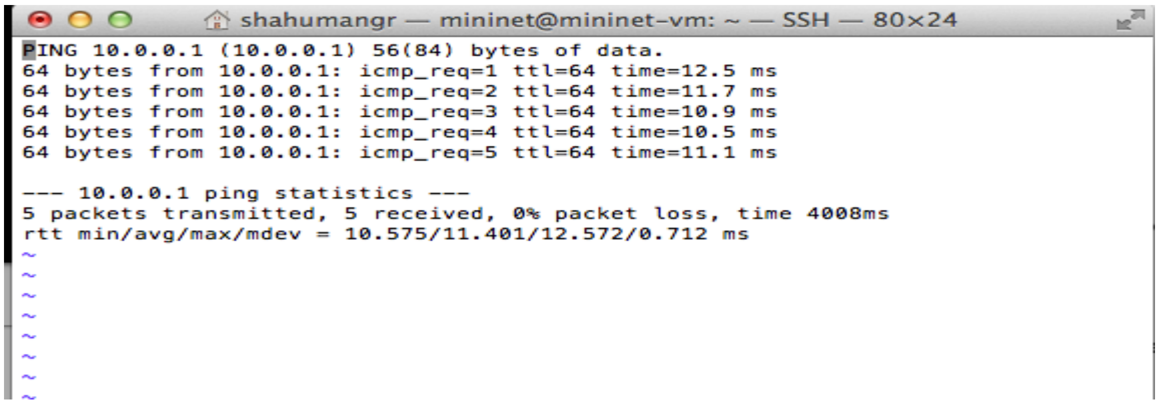
- We have also provided a functionality to set the number of times the host should ping other hosts



```
mininet@mininet-vm:~$ sudo python natdefreq.py 5
```

- In the above example 5 denotes the number of requests which is the input by the user

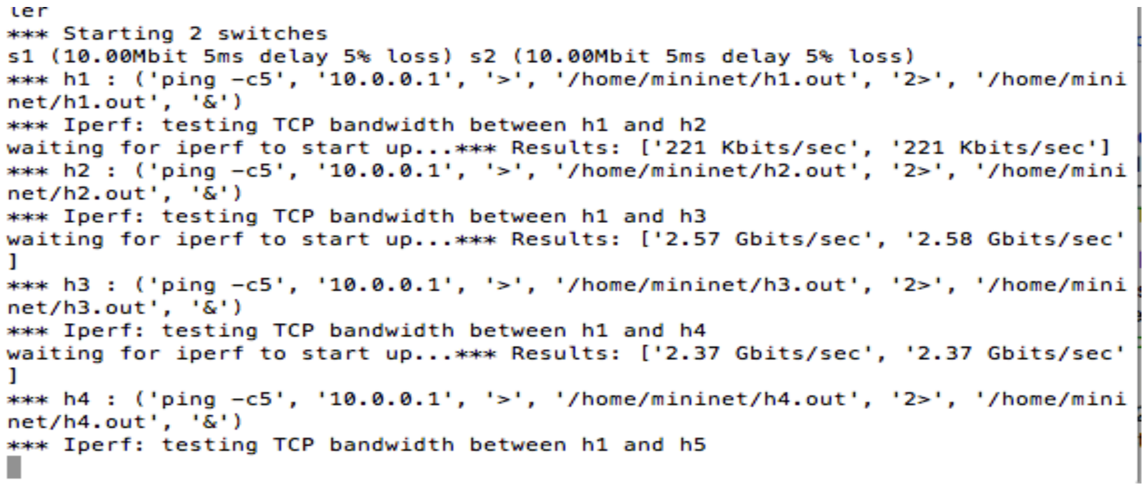
ix. We are storing ping outputs for each host to hostname. Out file for analysis.



```
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_req=1 ttl=64 time=12.5 ms
64 bytes from 10.0.0.1: icmp_req=2 ttl=64 time=11.7 ms
64 bytes from 10.0.0.1: icmp_req=3 ttl=64 time=10.9 ms
64 bytes from 10.0.0.1: icmp_req=4 ttl=64 time=10.5 ms
64 bytes from 10.0.0.1: icmp_req=5 ttl=64 time=11.1 ms

--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 10.575/11.401/12.572/0.712 ms
~
~
~
~
~
~
~
~
```

Fig10: Storing ping output for each host in a file.

x. Testing bandwidth in between multiple host as another performance measurement parameter



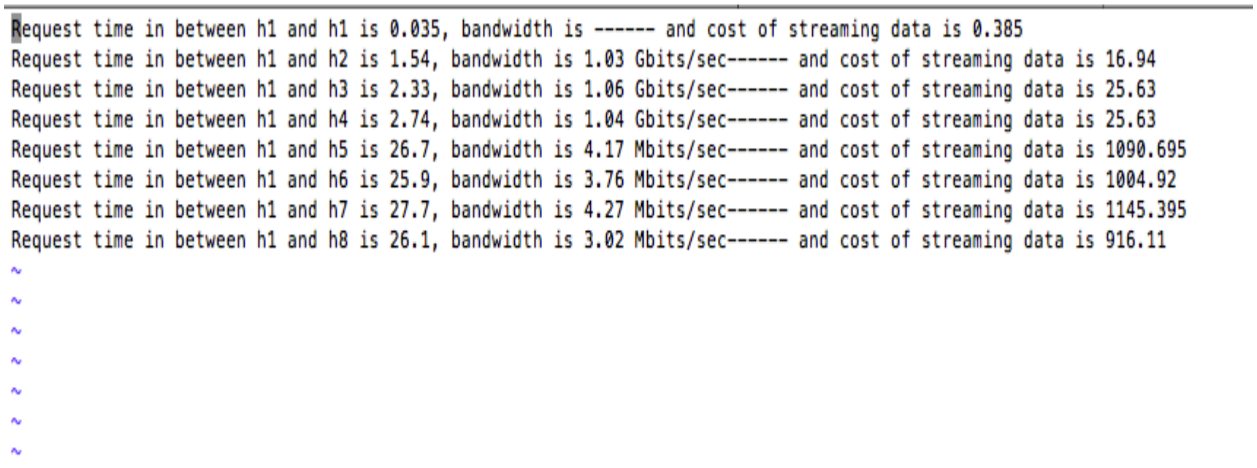
```
ler
*** Starting 2 switches
s1 (10.00Mbit 5ms delay 5% loss) s2 (10.00Mbit 5ms delay 5% loss)
*** h1 : ('ping -c5', '10.0.0.1', '>', '/home/mininet/h1.out', '2>', '/home/mini
net/h1.out', '&')
*** Iperf: testing TCP bandwidth between h1 and h2
waiting for iperf to start up...*** Results: ['221 Kbits/sec', '221 Kbits/sec']
*** h2 : ('ping -c5', '10.0.0.1', '>', '/home/mininet/h2.out', '2>', '/home/mini
net/h2.out', '&')
*** Iperf: testing TCP bandwidth between h1 and h3
waiting for iperf to start up...*** Results: ['2.57 Gbits/sec', '2.58 Gbits/sec'
]
*** h3 : ('ping -c5', '10.0.0.1', '>', '/home/mininet/h3.out', '2>', '/home/mini
net/h3.out', '&')
*** Iperf: testing TCP bandwidth between h1 and h4
waiting for iperf to start up...*** Results: ['2.37 Gbits/sec', '2.37 Gbits/sec'
]
*** h4 : ('ping -c5', '10.0.0.1', '>', '/home/mininet/h4.out', '2>', '/home/mini
net/h4.out', '&')
*** Iperf: testing TCP bandwidth between h1 and h5
```

Fig 11: Testing bandwidth in between multiple host

xi. Calculating cost for streaming a file to one host from the other entire host.

- Cost calculation of streaming one file to one host from all the other hosts depends upon following factor:
    - **No of requests**
    - **Time**
    - **Bandwidth**
    - **Delay**
    - **Packet loss**
    - **Storage cost**
- **Cost = No of requests * (time + bandwidth + delay+ packet loss+ storage)**

We are also outputting the ping time in between hosts, bandwidth test and total streaming cost to a file for further analysis and performance measurement



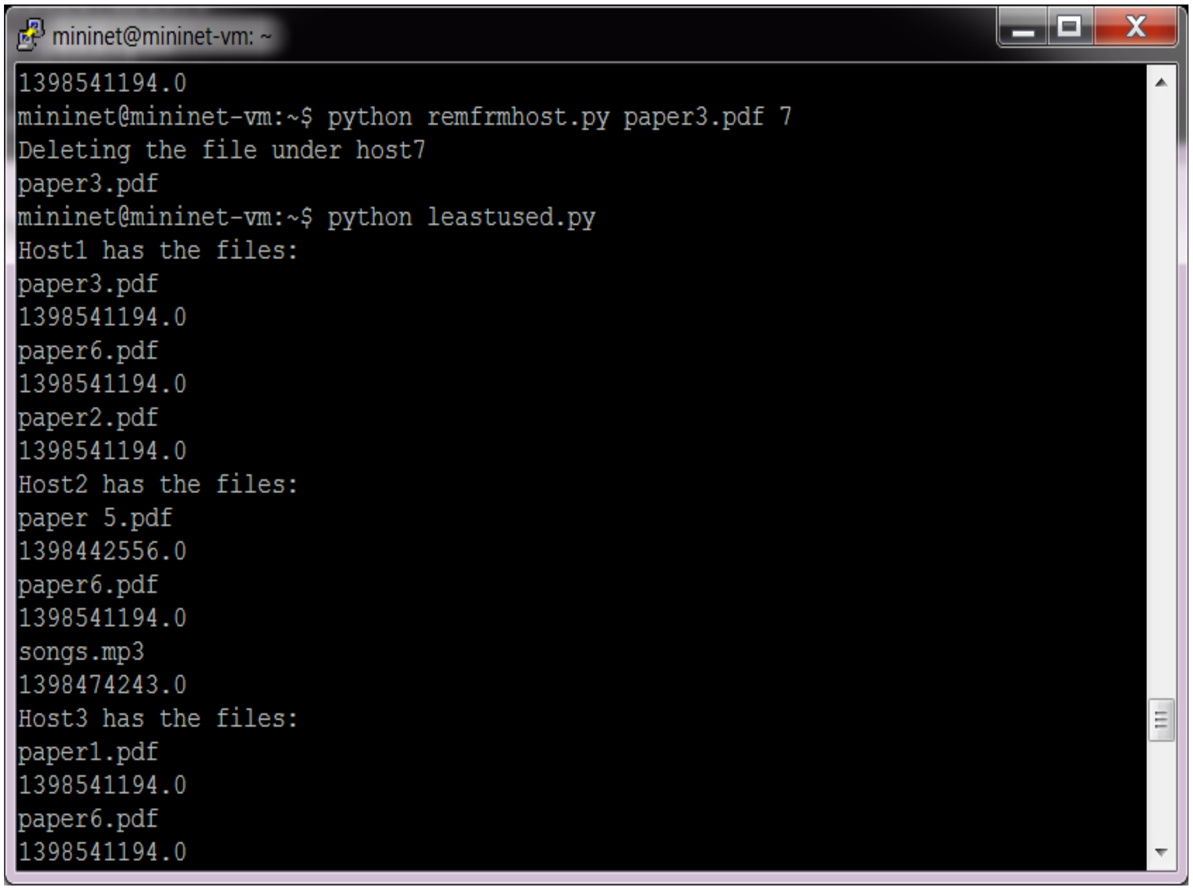
```
Request time in between h1 and h1 is 0.035, bandwidth is ------ and cost of streaming data is 0.385
Request time in between h1 and h2 is 1.54, bandwidth is 1.03 Gbits/sec------ and cost of streaming data is 16.94
Request time in between h1 and h3 is 2.33, bandwidth is 1.06 Gbits/sec------ and cost of streaming data is 25.63
Request time in between h1 and h4 is 2.74, bandwidth is 1.04 Gbits/sec------ and cost of streaming data is 25.63
Request time in between h1 and h5 is 26.7, bandwidth is 4.17 Mbits/sec------ and cost of streaming data is 1090.695
Request time in between h1 and h6 is 25.9, bandwidth is 3.76 Mbits/sec------ and cost of streaming data is 1004.92
Request time in between h1 and h7 is 27.7, bandwidth is 4.27 Mbits/sec------ and cost of streaming data is 1145.395
Request time in between h1 and h8 is 26.1, bandwidth is 3.02 Mbits/sec------ and cost of streaming data is 916.11
~
~
~
~
~
~
~
```

Fig 12: Request time, bandwidth measurement and cost calculation output to a file for future analysis and performance measurement.

Here From this output file we can deduce that the cost of streaming from far located servers/hosts h5, h6. h7, h8 are very high

xii. Removing the least used file

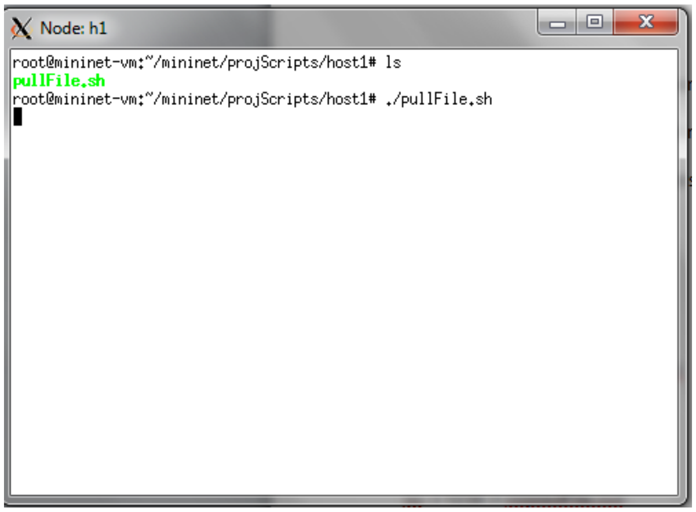
-
- The least used file in a host was listed using the leastused.py file.
- The file can be removed by running remfrmhost.py



```
1398541194.0
mininet@mininet-vm:~$ python remfrmhost.py paper3.pdf 7
Deleting the file under host7
paper3.pdf
mininet@mininet-vm:~$ python leastused.py
Host1 has the files:
paper3.pdf
1398541194.0
paper6.pdf
1398541194.0
paper2.pdf
1398541194.0
Host2 has the files:
paper 5.pdf
1398442556.0
paper6.pdf
1398541194.0
songs.mp3
1398474243.0
Host3 has the files:
paper1.pdf
1398541194.0
paper6.pdf
1398541194.0
```

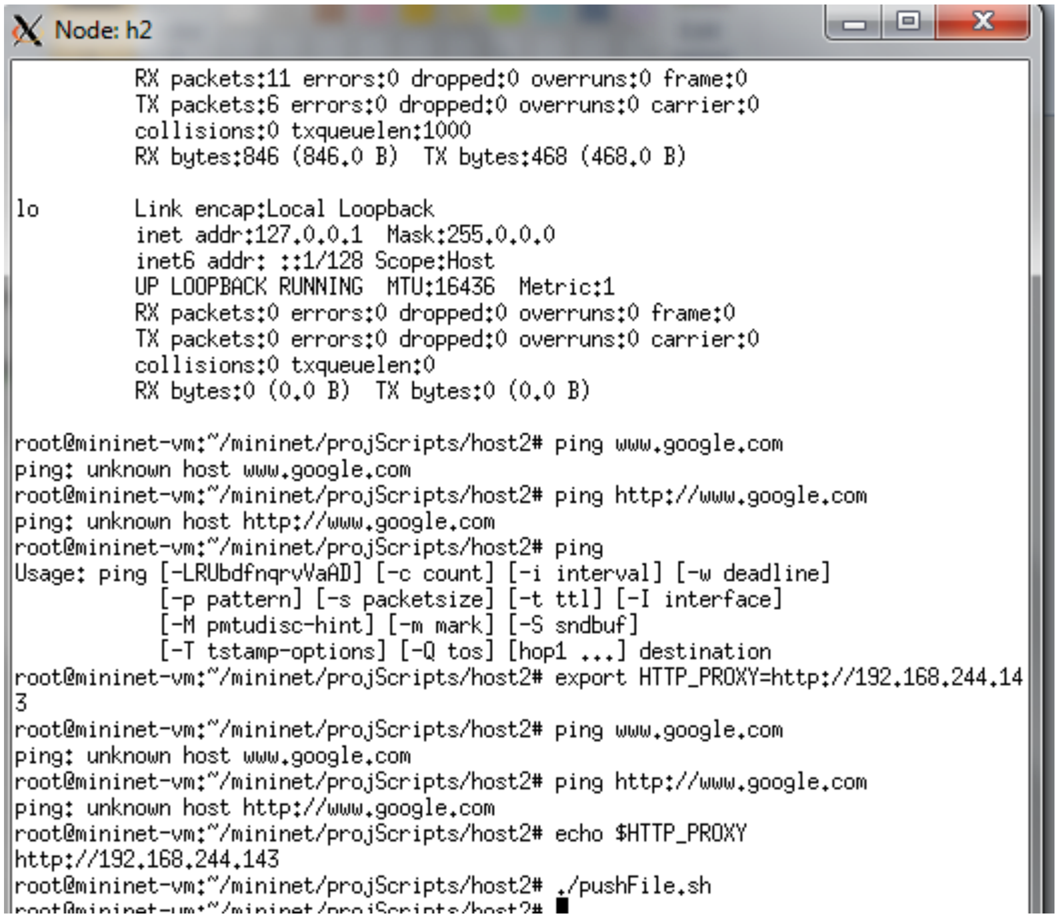
Fig 13: The least used file can be deleted from each of the hosts

xiii. Transferring files between hosts



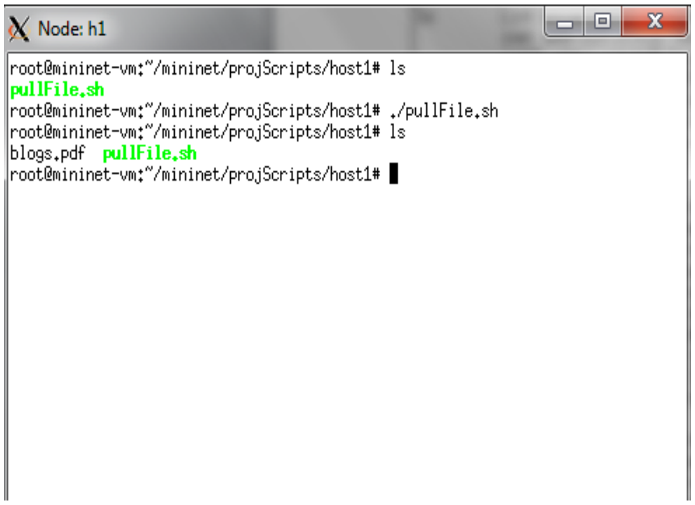
Node: h1

```
root@mininet-vm:~/mininet/projScripts/host1# ls
pullFile.sh
root@mininet-vm:~/mininet/projScripts/host1# ./pullFile.sh
```



Node: h2

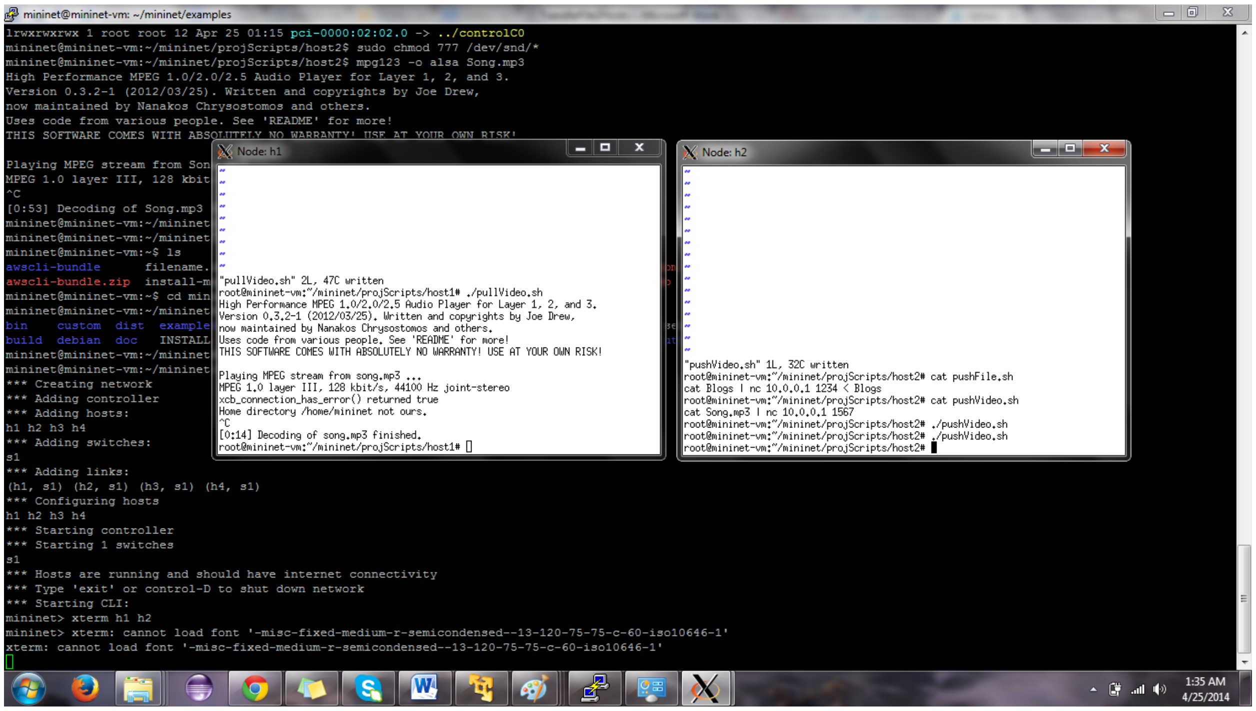
```
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:846 (846.0 B)  TX bytes:468 (468.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@mininet-vm:~/mininet/projScripts/host2# ping www.google.com
ping: unknown host www.google.com
root@mininet-vm:~/mininet/projScripts/host2# ping http://www.google.com
ping: unknown host http://www.google.com
root@mininet-vm:~/mininet/projScripts/host2# ping
Usage: ping [-LRUbdfnqrvVaAD] [-c count] [-i interval] [-w deadline]
            [-p pattern] [-s packetsize] [-t ttl] [-I interface]
            [-M pmtudisc-hint] [-m mark] [-S sndbuf]
            [-T tstamp-options] [-Q tos] [hop1 ...] destination
root@mininet-vm:~/mininet/projScripts/host2# export HTTP_PROXY=http://192.168.244.14
3
root@mininet-vm:~/mininet/projScripts/host2# ping www.google.com
ping: unknown host www.google.com
root@mininet-vm:~/mininet/projScripts/host2# ping http://www.google.com
ping: unknown host http://www.google.com
root@mininet-vm:~/mininet/projScripts/host2# echo $HTTP_PROXY
http://192.168.244.143
root@mininet-vm:~/mininet/projScripts/host2# ./pushFile.sh
root@mininet-vm:~/mininet/projScripts/host2#
```



```
root@mininet-vm:~/mininet/projScripts/host1# ls
pullFile.sh
root@mininet-vm:~/mininet/projScripts/host1# ./pullFile.sh
root@mininet-vm:~/mininet/projScripts/host1# ls
blogs.pdf  pullFile.sh
root@mininet-vm:~/mininet/projScripts/host1# 
```

Fig 14: File transfer between Host1 and Host2

xiv. Streaming audio files between hosts using Mpg123



```
lrwxrwxrwx 1 root root 12 Apr 25 01:15 pci-0000:02:02.0 -> ../controlC0
mininet@mininet-vm:~/mininet/projScripts/host2$ sudo chmod 777 /dev/snd/*
mininet@mininet-vm:~/mininet/projScripts/host2$ mpg123 -o alsa Song.mp3
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2, and 3.
Version 0.3.2-1 (2012/03/25). Written and copyrights by Joe Drew,
now maintained by Nanakos Chrysostomos and others.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!

Playing MPEG stream from Son
MPEG 1.0 layer III, 128 kbit
^C
[0:53] Decoding of Song.mp3
mininet@mininet-vm:~/mininet
mininet@mininet-vm:~/mininet
mininet@mininet-vm:~$ ls
awscli-bundle      filename.
awscli-bundle.zip  install-m
mininet@mininet-vm:~$ cd min
mininet@mininet-vm:~/mininet
bin    custom  dist  example
build  debian  doc   INSTALL
mininet@mininet-vm:~/mininet
mininet@mininet-vm:~/mininet
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 1 switches
s1
*** Hosts are running and should have internet connectivity
*** Type 'exit' or control-D to shut down network
*** Starting CLI:
mininet> xterm h1 h2
mininet> xterm: cannot load font '-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10646-1'
xterm: cannot load font '-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso10646-1'
```

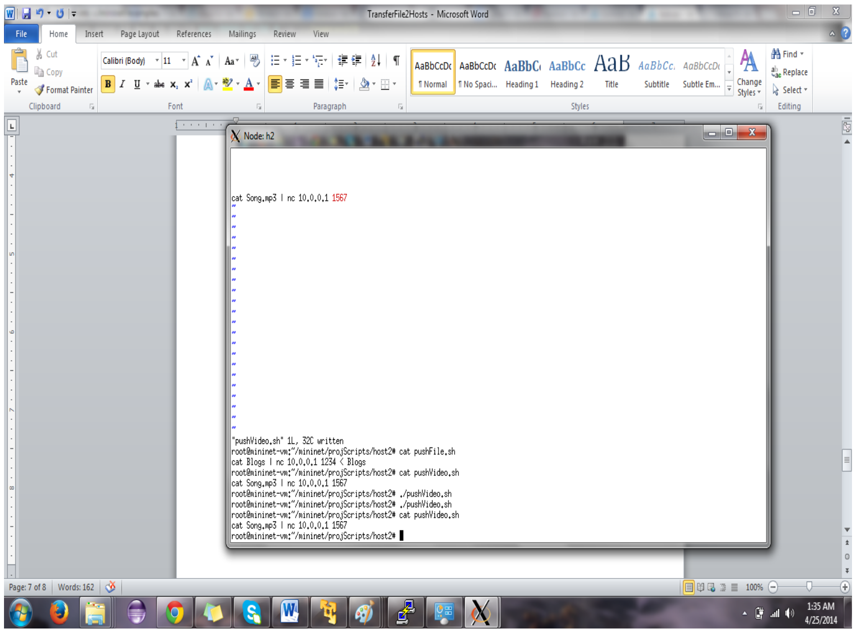
```
~
~
~
~
~
~
~
~
~
"pullVideo.sh" 2L, 47C written
root@mininet-vm:~/mininet/projScripts/host1# ./pullVideo.sh
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2, and 3.
Version 0.3.2-1 (2012/03/25). Written and copyrights by Joe Drew,
now maintained by Nanakos Chrysostomos and others.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!

Playing MPEG stream from song.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
xcb_connection_has_error() returned true
Home directory /home/mininet not ours.
^C
[0:14] Decoding of song.mp3 finished.
root@mininet-vm:~/mininet/projScripts/host1# 
```

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"pushVideo.sh" 1L, 32C written
root@mininet-vm:~/mininet/projScripts/host2# cat pushFile.sh
cat Blogs | nc 10.0.0.1 1234 < Blogs
root@mininet-vm:~/mininet/projScripts/host2# cat pushVideo.sh
cat Song.mp3 | nc 10.0.0.1 1567
root@mininet-vm:~/mininet/projScripts/host2# ./pushVideo.sh
root@mininet-vm:~/mininet/projScripts/host2# ./pushVideo.sh
root@mininet-vm:~/mininet/projScripts/host2# 
```

Fig 15.a : Audio files streaming between two hosts

- Host 2 is the server sending the audio mp3 file. Song.mp3 should be present in this
- Host1 is the client who is going to play this file

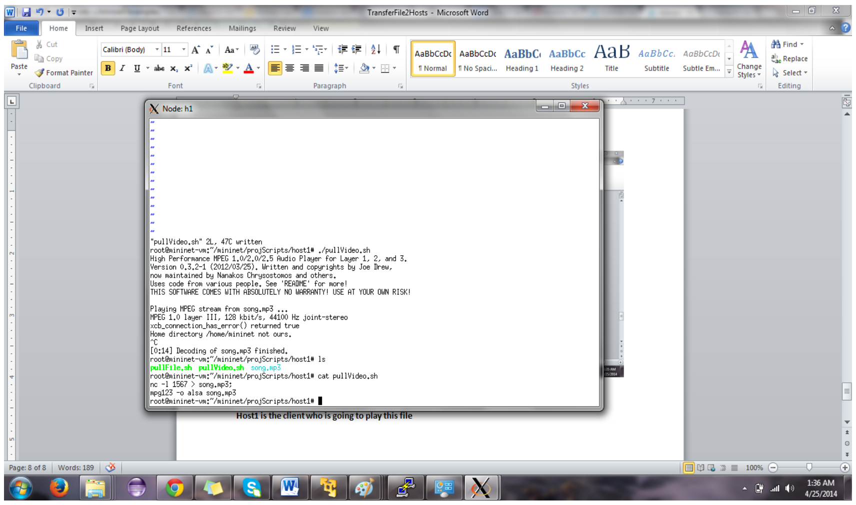
Host 2:



```
cat Song.mp3 | nc 10.0.0.1 1567
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"pushVideo.sh" 1L, 32C written
root@mininet-vm:~/mininet/projScripts/host2# cat pushFile.sh
cat Blogs | nc 10.0.0.1 1234 < Blogs
root@mininet-vm:~/mininet/projScripts/host2# cat pushVideo.sh
cat Song.mp3 | nc 10.0.0.1 1567
root@mininet-vm:~/mininet/projScripts/host2# ./pushVideo.sh
root@mininet-vm:~/mininet/projScripts/host2# ./pushVideo.sh
root@mininet-vm:~/mininet/projScripts/host2# cat pushVideo.sh
cat Song.mp3 | nc 10.0.0.1 1567
root@mininet-vm:~/mininet/projScripts/host2#
```

Fig 15.b : Audio files streaming between two hosts : Host2 is the server

Host 1:



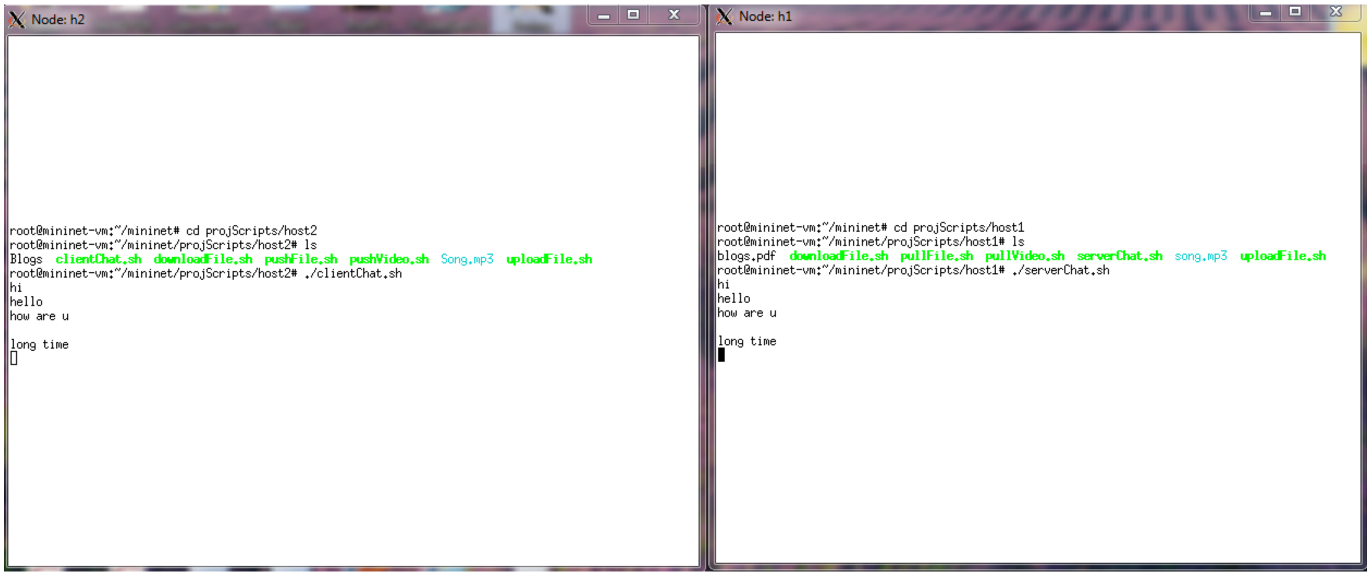
```
~
~
~
~
~
~
~
~
~
~
~
~
~
"pullVideo.sh" 2L, 47C written
root@mininet-vm:~/mininet/projScripts/host1# ./pullVideo.sh
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2, and 3.
Version 0.3.2-1 (2012/03/25). Written and copyrights by Joe Drew,
now maintained by Nanakos Chrysostomos and others.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!

Playing MPEG stream from song.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
xcb_connection_has_error() returned true
Home directory /home/mininet not ours.
^C
[0:14] Decoding of song.mp3 finished.
root@mininet-vm:~/mininet/projScripts/host1# ls
pullFile.sh  pullVideo.sh  song.mp3
root@mininet-vm:~/mininet/projScripts/host1# cat pullVideo.sh
nc -l 1567 > song.mp3;
mpg123 -o alsa song.mp3
root@mininet-vm:~/mininet/projScripts/host1#
```

Host1 is the client who is going to play this file

Fig 15.c : Audio files streaming between two hosts : Host1 is client

xv. Internet chat integration with mininet

- In order to make it easier for hosts to request files from other hosts we have integrated the chat functionality with each of these hosts.
- There are 2 scripts which will help us to achieve this : serverChat.sh and clientChat.sh
- The host which is acting as a client and is wishing to chat with another host acting like a server and having the files to store should execute clientChat.sh
- The server will always be listening on a predefined port and this server will be running the serverChat.sh.
- Using these to files each of the hosts can communicate with each other in real time.
- Depicted in the diagram below is a screenshot of the real time communication between host1 and host2.



```
root@mininet-vm:~/mininet# cd projScripts/host2
root@mininet-vm:~/mininet/projScripts/host2# ls
Blogs  clientChat.sh  downloadFile.sh  pushFile.sh  pushVideo.sh  Song.mp3  uploadFile.sh
root@mininet-vm:~/mininet/projScripts/host2# ./clientChat.sh
hi
hello
how are u

long time
```

```
root@mininet-vm:~/mininet# cd projScripts/host1
root@mininet-vm:~/mininet/projScripts/host1# ls
blogs.pdf  downloadFile.sh  pullFile.sh  pullVideo.sh  serverChat.sh  song.mp3  uploadFile.sh
root@mininet-vm:~/mininet/projScripts/host1# ./serverChat.sh
hi
hello
how are u

long time
```

Fig 16: Live chat between two hosts on the mininet network

xvi. Plotting of Performance graph for the entire system

- The matplotlib, a python 2D plotting library is used for plotting the graph. It provides an object-oriented API for embedding plots into applications.
- It is a procedural "pylab" interface designed to closely relate to that of MATLAB.
- So here the graph is displayed which shows the relation between the total streaming costs and the required host.

The X- axis denotes the hosts: 1 to 8 , while the Y axis denotes the Total Streaming Cost.

A screenshot of the graph is shown below:

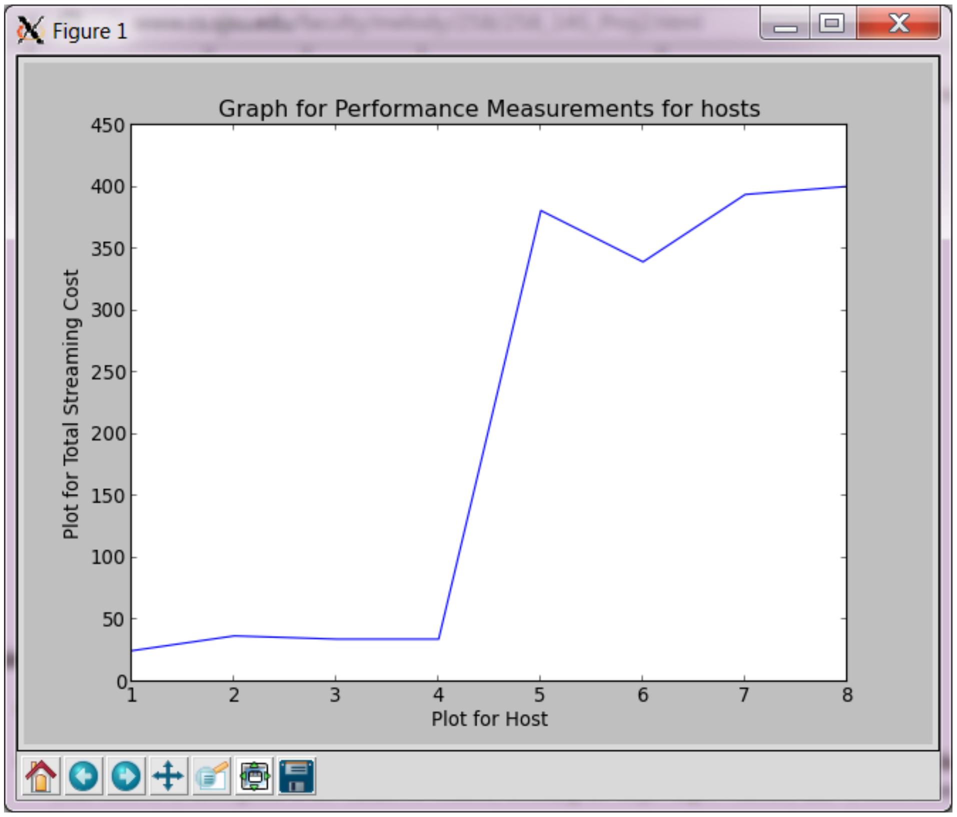


Fig 17: Performance analysis of the system

- As seen in the graph above it is seen that the streaming cost from a nearby host is smaller compared to that of a farther host. So the experiment clearly proves that it is a feasible option to get a file from a nearby host rather than from a farther one.
- In the project the graph is stored in plotgraph.png file for analysis. The software imagemagick was used to display the graph. Imagemagick is a software used to create,edit or convert bitmap images.

10. Comparison with the existing simulator system

Feature	IEEE Paper	Our Implementation
Collect statistics over a certain duration of time	Supported	Supported
Replicate data based on statistics collected	Supported	Supported
Data stored in multiple clouds across countries	Supported	Not supported
In case of decreased demands, remove data	Supported	Supported
Cost calculation	Supported	Supported
Plotting graph based on hosts and cost in real time	Not supported	Supported
Live chat between the hosts for requesting files	Not supported	Supported
Listening to audio files on another hosts system	Not supported	Supported
File transfer from Host to host on a network and deleting listing all files with a host	Not Supported	Supported

11. References

[1] Yu Wu, Chuan Wu, BoLi, Linquan Zhang, Zongpeng Li, and Francis C. M. Lau: Scaling Social Media applications into Geo Distributed Clouds, ‘IEEE/ACM TRANSACTIONS ON NETWORKING’.

[2]http://www.mininet.org

[3] https://aws.amazon.com/cli/

[4]:http://comments.gmane.org/gmane.network.routing.openflow.general/2632

[5]: http://pages.cs.wisc.edu/~agember/cs640/s14/project3

[6]: https://mailman.stanford.edu/pipermail/mininet-discuss/2012-July/000979.html

[7]: http://groups.geni.net/geni/wiki/GeniTmix

[8]: http://www.brianlinkletter.com/mininet-test-drive/

[9]:http://nmap.org/book/inst-linux.html#inst-rpm

[10]:http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-set-up.html