GROUP DETAILS

Team Name : TechGeeks

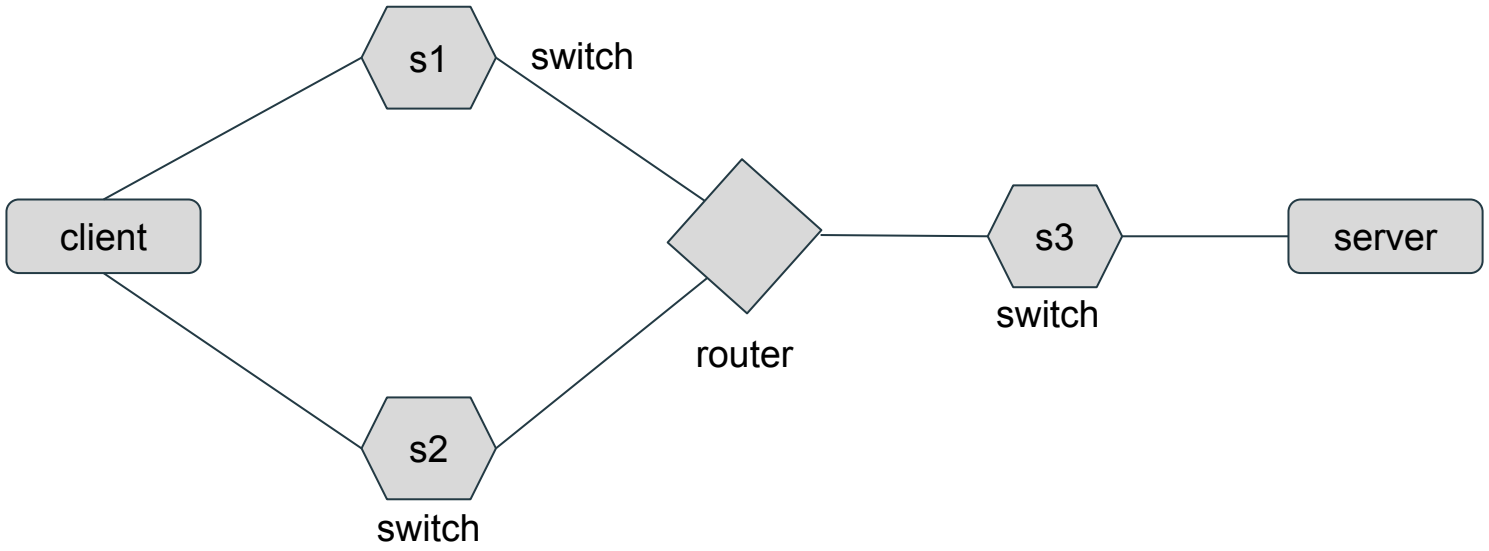
Group No: ACN08

S.No	Name	Email ID	Student ID
1	Himanshu Gupta	h20201030121@hyderabad.bits-pilani.ac.in	2020H1030121H
2	Aman Srivastav	h20201030137@hyderabad.bits-pilani.ac.in	2020H1030137H
3	Somdatta Sen	h20201030140@hyderabad.bits-pilani.ac.in	2020H1030140H

STEP - 1, Implementation of MP-QUIC

Steps followed to implement :

1. Setup MP-QUIC structure.
2. Create a topology as below :



3. Create the files client-multipath.go and server-multipath.go

4. The files are stored in location “storage-server” and are accessible by server.

5. To execute the code, follow the below steps :

- Run the python file exported from topology using command :
  - sudo python2 setup-topology.py
- Once the terminal enters mininet, open xterm of server and client :
  - xterm server client
- In xterm of server, run the below commands :
  - go build server-multipath.go
  - ./server-multipath storage-client

  Here, storage-client is the destination location where client stores its files.
- In xterm of client, run the below commands :
  - go build client-multipath.go
  - ./client-multipath storage-server/abc.txt 100.0.0.1

  Here, 100.0.0.1 is the IP address of node - server and abc.txt is the file requested by client which is stored in location “storage-server”

```
mininet@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments
/file-transfer$ sudo python2 setup-topology.py
mininet> xterm server client
```

Xterm of Client :

```
"Node: client"
Connection LocalAddress : 10.0.0.1:47586        Connection RemoteAddress : 100.0
.0.1:4242

Path ID : 3


Connection LocalAddress : 11.0.0.1:52088        Connection RemoteAddress : 100.0
.0.1:4242
session created:  100.0.0.1:4242
stream created...
Client connected
A client has connected!
Receiving filename and filesize!
Start receiving file!

Packets received on Path-1 :  4276004
Packets received on Path-3 :  4276004
Receiving: 8552008 / 8552008
Transfer took:  4m36.245492063s


File has been received, closing stream!
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer#
```

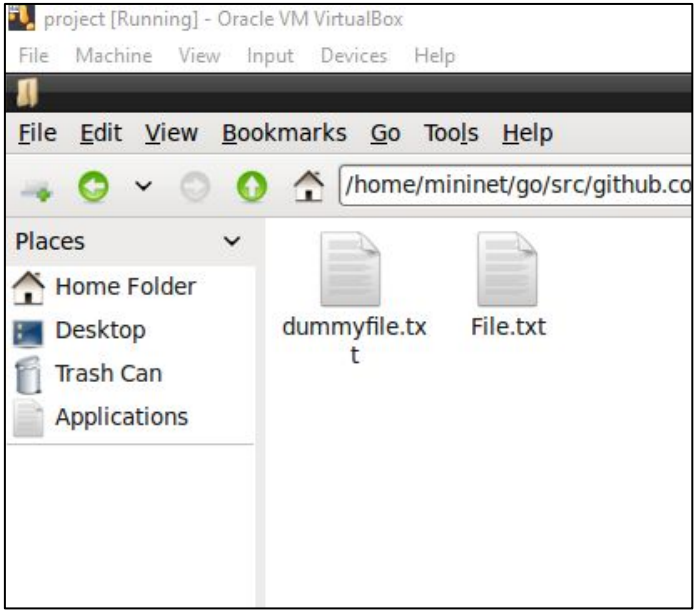
"Node: server"

```
Received: 1018880 / 8552008^C
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer# go build server-multipath.go
# github.com/lucas-clemente/quic-go/congestion
../../quic-go/congestion/olia_sender.go:5:2: imported and not used: "fmt"
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer# go build server-multipath.go
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer# ./server-multipath storage-client
Saving file to:  storage-client
Attaching to:  0.0.0.0:4242
Server started! Waiting for streams from client...
session created:  10.0.0.1:60682
stream created:  3
Connected to server, start sending the file name and file size
file size sent:  8552008
file name sent:  main
Sent: 8552008 / 8552008
Transfer took:  4m36.464983926s


Sent file completely!
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer#
```

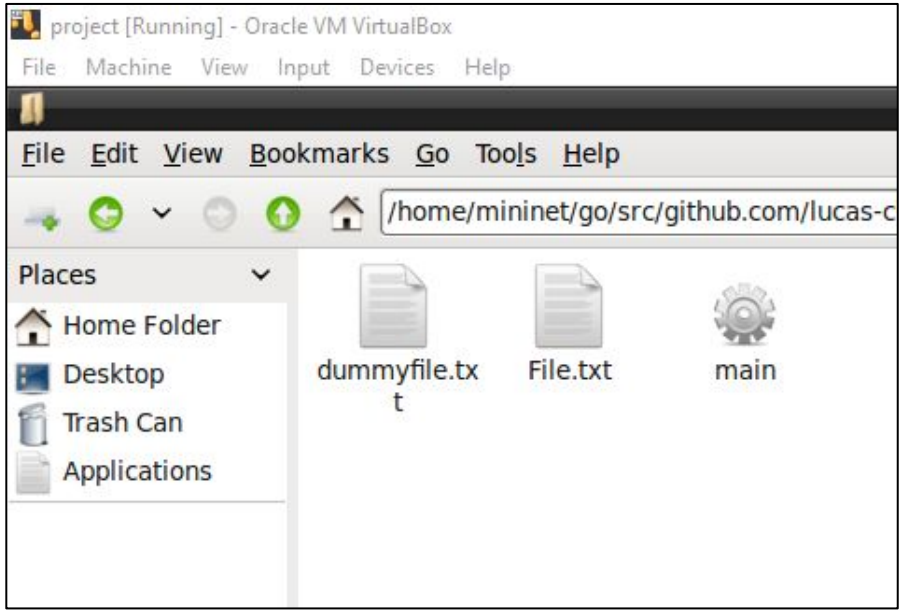
Xterm of Server

Files in location “storage-client”

Before Transfer



After Transfer



STEP - 2, Research Paper Study and Implementation

Priority-Based Stream Scheduling

- In MP-QUIC, when multiple streams share common path, there are chances of Inter-stream blocking, which may have severe consequences.
  Also, stream features such as Bandwidth,RTT Delay etc are not taken into consideration when streams are directed on paths in network.
- In Priority Based Scheduling, instead of making all streams competing for the fast path in a greedy fashion, we allocate paths for each stream by considering the match of stream and path features in the scheduling process.
- In this, streams can be prioritized by giving them priority value based on path features(bandwidth, RTT, Completion time, etc.) and then the scheduler allocates the new stream to each path with a calculated amount of data.
- This type of scheduling reduces the burst transmission of packets in congested path or paths with having low delay.

Implementation

- To implement Priority based scheduling, we have assigned a priority to each stream that is created for file transfer / transfer of all the packets created.
- The streams are scheduled on the basis of decreasing priority of streams on a common path.
- For example -
  If 3 streams are created, and we have 2 paths available in network, then, 2 streams will be redirected to 2 paths available, and 3rd stream will be sharing the path with either Stream1 / Stream2.
  In this case, the stream scheduling is done on the basis of priority. The probability of a stream being selected is calculated by dividing its priority by the priority sum of all the scheduled streams on this path.

Screenshot : Server

"Node: server"

```
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer# go build server-multipath.go
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer# ./server-multipath storage-client/
Saving file to:  storage-client/
Attaching to:  0.0.0.0:4242
Server started! Waiting for streams from client...
session created:  10.0.0.1:56613
stream created:  3
Priority of stream: %+v &{0 255 false}
Connected to server, start sending the file name and file size
stream created:  3
Priority of stream: %+v &{0 255 false}
file size sent:  13289
stream created:  3
Priority of stream: %+v &{0 255 false}
file name sent:  File.txt
Sent: 13289 / 13289
Transfer took:  133.347392ms


Sent file completely!
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer#
```

Screenshot : Client

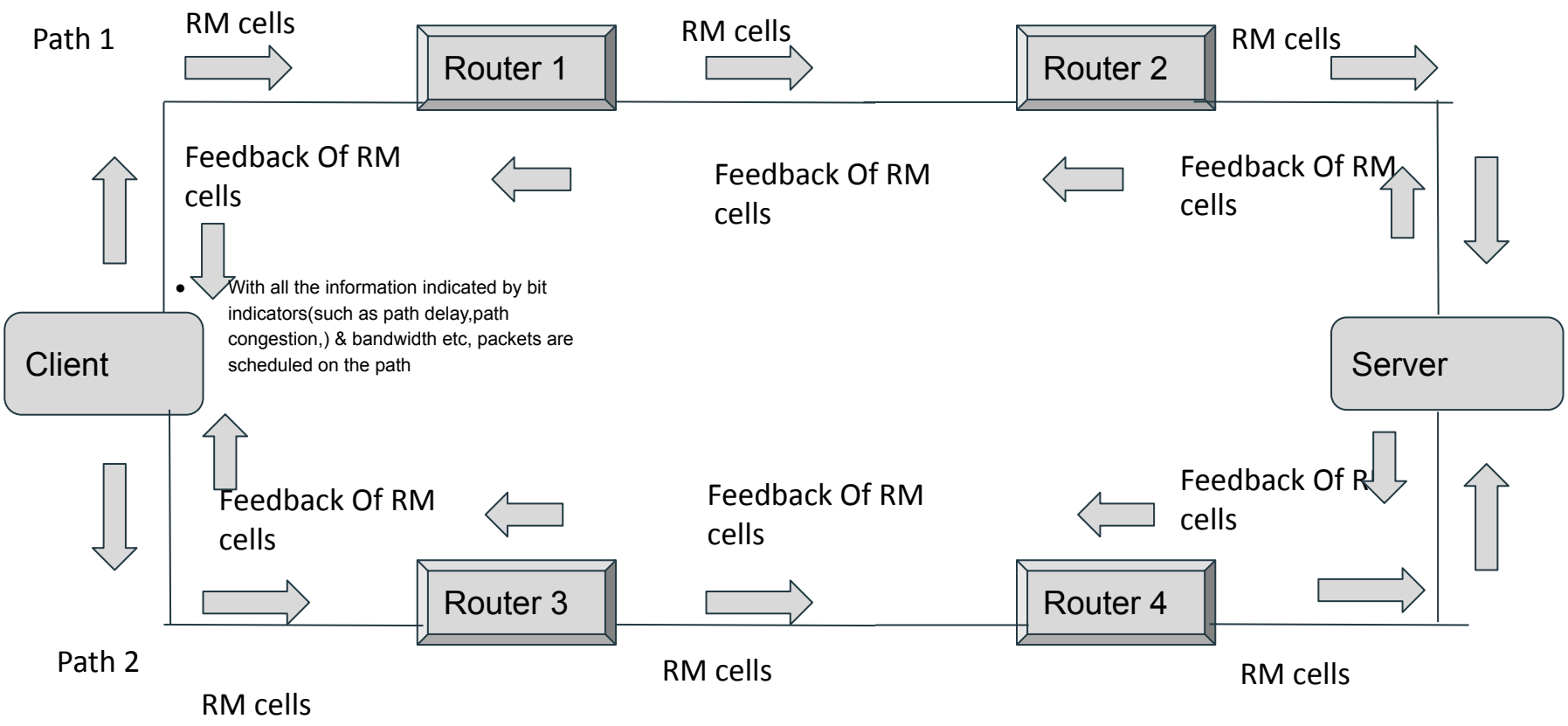
"Node: client"

```
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer# go build client-multipath.go
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer# ./client-multipath storage-server/File.txt 100.0.0.1
Server Address:  100.0.0.1:4242
Receiving File:  storage-server/File.txt
file is large, using multipath now.
Trying to connect to:  100.0.0.1:4242
session created:  100.0.0.1:4242
stream created...
Client connected
A client has connected!
Receiving filename and filesize!
Start receiving file!

Packets received on Path-1 :  6645
Packets received on Path-3 :  6644
Receiving: 13289 / 13289
Transfer took:  131.237618ms


File has been received, closing stream!
root@mininet-vm:~/go/src/github.com/lucas-clemente/multipath-quic-experiments/f
ile-transfer#
```

Step 3: Enhancement(s) / Innovation

Network Assisted Path scheduling

- The scheduling algorithm we have now, considers the path features, RTT and many things, on the sender side.
- During the initiation of connection between client and server, the RTT delay is taken and fixed length cells are sent along with handshake.
- When a destination host receives an RM cell, it will send the RM cell back to the sender with its CI and NI bits intact.
- With all the information indicated by bit indicators(such as path delay,path congestion,bandwidth etc), packets are scheduled on the path
- The data cells will be triggered after every constant time so that scheduler can schedule the data in the best path.


Path 1
RM cells
Router 1
RM cells
Router 2
RM cells
Feedback Of RM cells
Feedback Of RM cells
Feedback Of RM cells
With all the information indicated by bit indicators(such as path delay,path congestion,) & bandwidth etc, packets are scheduled on the path
Client
Server
Feedback Of RM cells
Feedback Of RM cells
Feedback Of RM cells
Router 3
Router 4
Path 2
RM cells
RM cells
RM cells

Thank You .. !!