Congestion control and TCP/UDP connection.

Amirmahdi Ansaripour 810198358
Soroosh Hamedi 810198503

In this section, we will discuss the TCP/UDP connection. There are 4 main modules in this part, Socket, Sender, Receiver, and Router.

Socket: instead of placing the methods related to socket connections in each module, it is better to separate it as a class. The class does not have any complex method, they are all gathered in one file.

Sender: As its name suggests, this module sends packets to the router which will then send the packets to the receiver. The key points about sender are how it handles its congestion window size. The slow start algorithm is used, in which the window is doubled each time the ACK of the last packet sent is received. Timeout is set to default value of 1; however, it can be changed by the DELAYCOEF coefficient. If the last ACK is not obtained before time-out, the threshold will be equal to half of the current window, and the window will start from 1 again.

Router: This module resembles a router in a real TCP connection, which contains a limited-sized queue, and some sockets to connect to different hosts or other routers. The queue size has been set to 10, and the packets which arrive when the queue is full are dropped.

Receiver: The receiver side of the connection has many responsibilities. The first one is to figure out whether the current packet has arrived in order or not. The second one is to send back the acknowledgement packets, and the last duty is to reconstruct the 1MB file which has been sent by the sender in the form of thousands of packets.

```cpp
#include <math.h>
#include <fstream>
#include "../socket/socket.hpp"


const std::string dest = "172.16.0.1";
const int MAXNUMOFPACKETS = 1100;
const int INF = 10000;
const int NEWLINE = 127;
const int DELAYCOEF = 1;     // increase leads to long timeouts

class Sender{
public:
    Sender(std::string, int, int);
    void run();
    int extractAck(std::string);
    void sendPacketsBasedCwd();
    void updateCWND();
    void handleTimeout();
    void findHeader(std::string);
    void recordData();
    void splitIntoPackets();
    void makePackets();
private:
    clock_t start;
    clock_t end;
    std::vector<int> cwnds;
    std::vector<int> threasholds;
    std::vector<std::string> packets;
    std::vector<std::string> content;
    Socket* toRouter;
    Socket* fromRouter;
    int lastPacketSent;
```

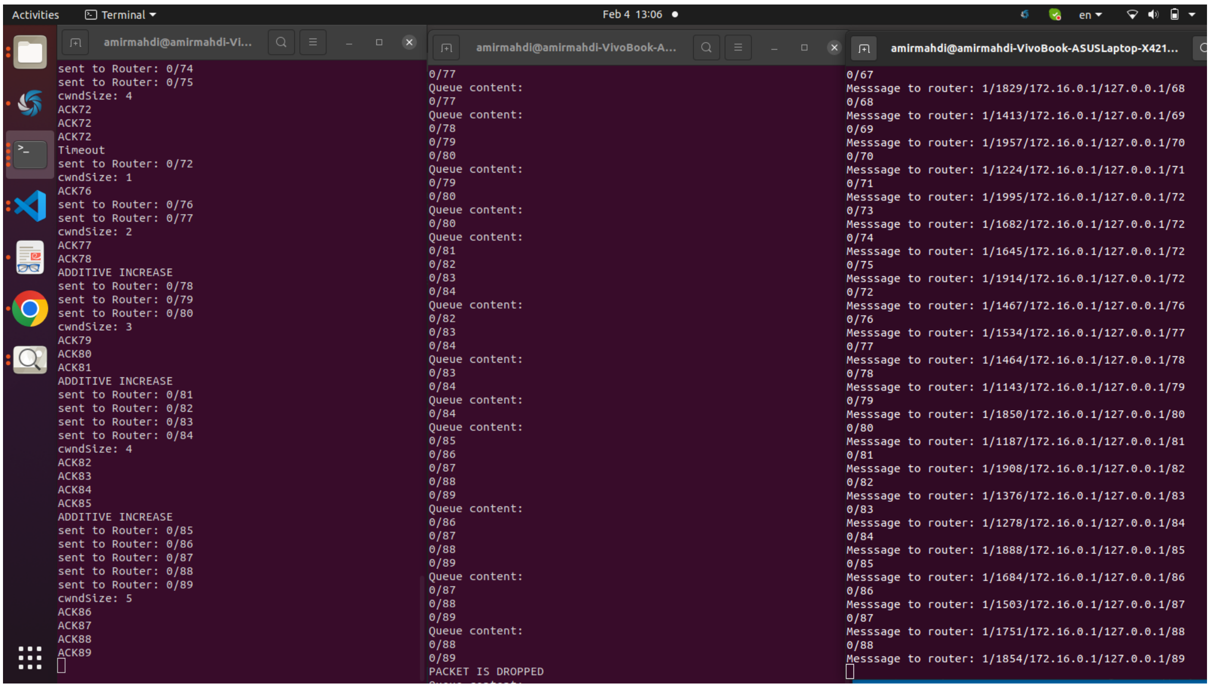
```cpp
#include "../socket/socket.hpp"
#include <map>
#include <fstream>
#include <string>

const int MAX_PACKET_IND = 1110;
const int MAXNUMOFPACKETS = 1100;
const std::string dest = "127.0.0.1";

class Receiver{
public:
    Receiver(std::string, int, int);
    void run();
    int searchFirstLost();
    void sendAck(std::string);
    bool handlePacket(std::string);
    void reconstructFile();
    std::string makeAckMessage(int);
private:
    std::vector<bool>Acks;
    Socket* toRouter;
    Socket* fromRouter;
    std::string port;
    std::vector<std::string> content;
    int indexHeader1, indexHeader2, indexHeader3, indexHeader4;
};
```

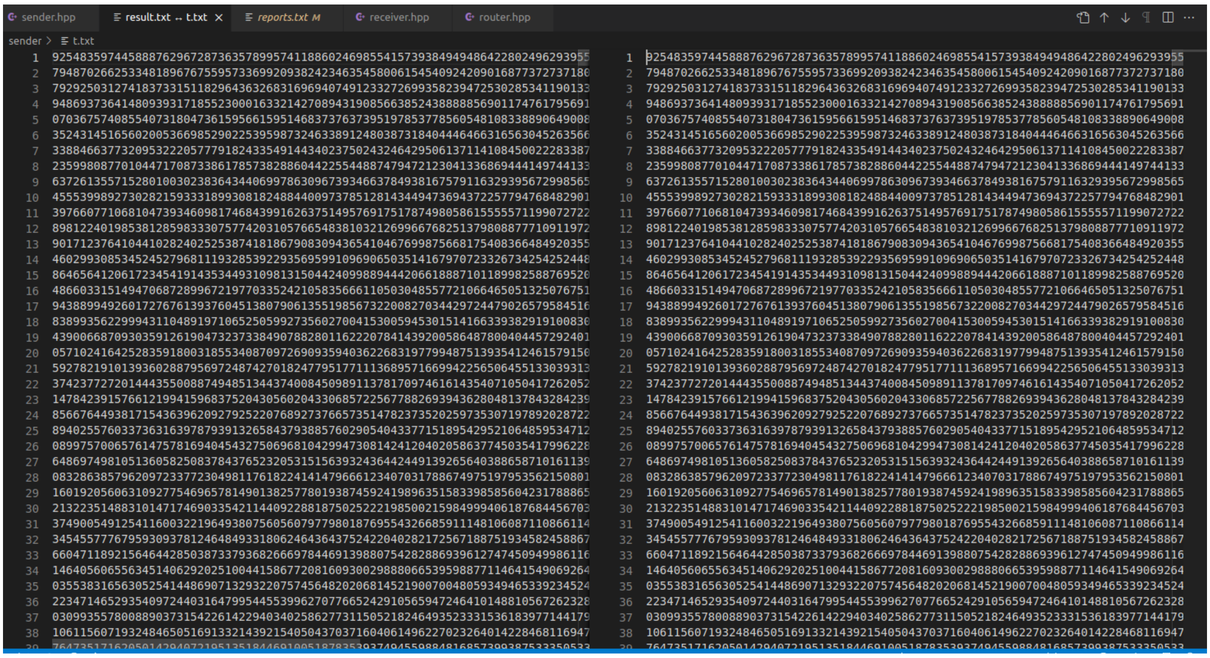
```cpp
#ifndef __ROUTER_H__
#define __ROUTER_H__

#include"../socket/socket.hpp"
#include <map>

const int QUEUESIZE = 10;    // infinity (10000) is the first case
const int MAXNUMOFPACKETS = 1100;
const int DELAYCOEF = 100;
const int DROPPROBABILITY = 10; // 0 if no need to drop

class Router{
private:
    std::string port;
    Socket* toSender;
    Socket* fromSender;
    Socket* toReceiver;
    Socket* fromReceiver;
    void findHeader(std::string);
    void decideOnDropped();
public:
    Router(std::string, int, int, int, int);
    std::map<int, bool> droppedPackets;
    void run();
    std::vector<int> dropped;
    std::vector<std::string> queue;
    clock_t lastPacketSent;
    void showQueueContent();
    int numOfSents;
    int indexHeader1, indexHeader2;
};
```

```cpp
#define STDIN 0

class Socket{
public:
    Socket(int);
    void setup();
    int send(std::string);
    std::string receive();
    int getFd();
    int fd;
private:
    int socketPort;
    struct sockaddr_in bus;
    char buffer[255];
};
```

How to run:
./sender.out 127.0.0.1 8001 8002
./router.out 172.16.0.0 8002 8001 8003 8004
./receiver.out 172.16.0.1 8004 8003

check reports.txt in sender folder


Activities
Terminal
Feb 4 13:06
amirmahdi@amirmahdi-Vi...
sent to Router: 0/74
sent to Router: 0/75
cwndSize: 4
ACK72
ACK72
ACK72
Timeout
sent to Router: 0/72
cwndSize: 1
ACK76
sent to Router: 0/76
sent to Router: 0/77
cwndSize: 2
ACK77
ACK78
ADDITIVE INCREASE
sent to Router: 0/78
sent to Router: 0/79
sent to Router: 0/80
cwndSize: 3
ACK79
ACK80
ACK81
ADDITIVE INCREASE
sent to Router: 0/81
sent to Router: 0/82
sent to Router: 0/83
sent to Router: 0/84
cwndSize: 4
ACK82
ACK83
ACK84
ACK85
ADDITIVE INCREASE
sent to Router: 0/85
sent to Router: 0/86
sent to Router: 0/87
sent to Router: 0/88
sent to Router: 0/89
cwndSize: 5
ACK86
ACK87
ACK88
ACK89
amirmahdi@amirmahdi-VivoBook-A...
0/77
Queue content:
0/77
Queue content:
0/78
0/79
0/80
Queue content:
0/79
0/80
Queue content:
0/80
Queue content:
0/81
0/82
0/83
0/84
Queue content:
0/82
0/83
0/84
Queue content:
0/83
0/84
Queue content:
0/84
Queue content:
0/85
0/86
0/87
0/88
0/89
Queue content:
0/86
0/87
0/88
0/89
Queue content:
0/87
0/88
0/89
Queue content:
0/88
0/89
PACKET IS DROPPED
amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421...
0/67
Messsage to router: 1/1829/172.16.0.1/127.0.0.1/68
0/68
Messsage to router: 1/1413/172.16.0.1/127.0.0.1/69
0/69
Messsage to router: 1/1957/172.16.0.1/127.0.0.1/70
0/70
Messsage to router: 1/1224/172.16.0.1/127.0.0.1/71
0/71
Messsage to router: 1/1995/172.16.0.1/127.0.0.1/72
0/73
Messsage to router: 1/1682/172.16.0.1/127.0.0.1/72
0/74
Messsage to router: 1/1645/172.16.0.1/127.0.0.1/72
0/75
Messsage to router: 1/1914/172.16.0.1/127.0.0.1/72
0/72
Messsage to router: 1/1467/172.16.0.1/127.0.0.1/76
0/76
Messsage to router: 1/1534/172.16.0.1/127.0.0.1/77
0/77
Messsage to router: 1/1464/172.16.0.1/127.0.0.1/78
0/78
Messsage to router: 1/1143/172.16.0.1/127.0.0.1/79
0/79
Messsage to router: 1/1850/172.16.0.1/127.0.0.1/80
0/80
Messsage to router: 1/1187/172.16.0.1/127.0.0.1/81
0/81
Messsage to router: 1/1908/172.16.0.1/127.0.0.1/82
0/82
Messsage to router: 1/1376/172.16.0.1/127.0.0.1/83
0/83
Messsage to router: 1/1278/172.16.0.1/127.0.0.1/84
0/84
Messsage to router: 1/1888/172.16.0.1/127.0.0.1/85
0/85
Messsage to router: 1/1684/172.16.0.1/127.0.0.1/86
0/86
Messsage to router: 1/1503/172.16.0.1/127.0.0.1/87
0/87
Messsage to router: 1/1751/172.16.0.1/127.0.0.1/88
0/88
Messsage to router: 1/1854/172.16.0.1/127.0.0.1/89

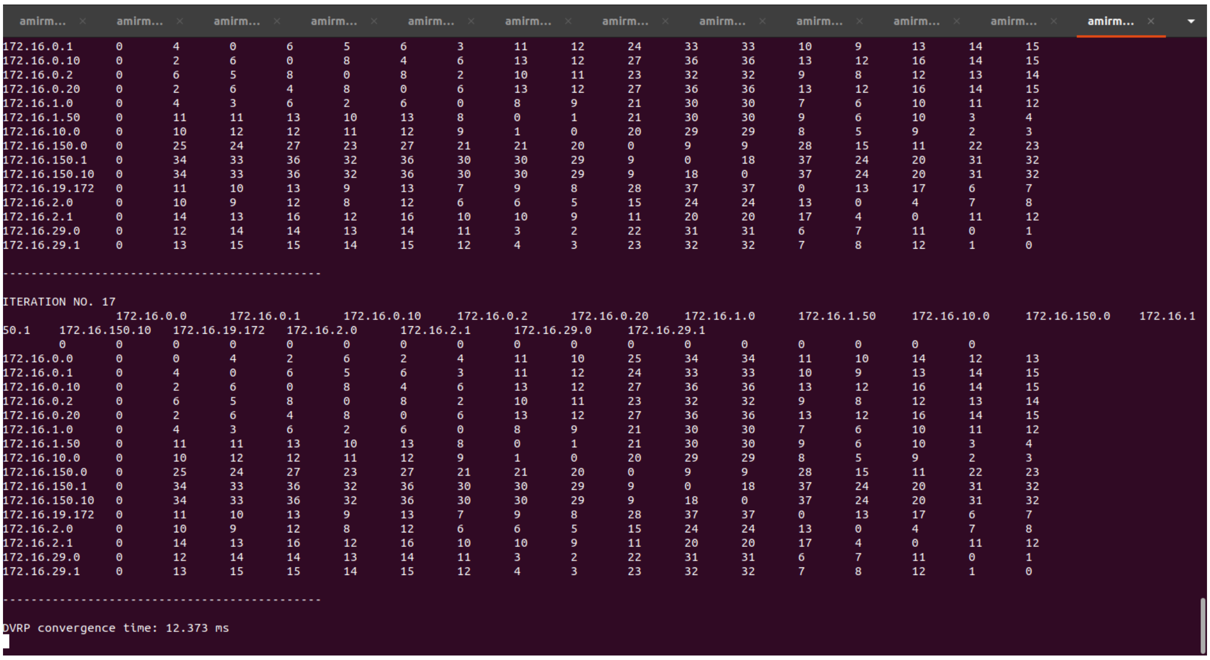

sender.hpp
result.txt ↔ t.txt
reports.txt M
receiver.hpp
router.hpp
sender > t.txt

```
Duration: 228 s
Cwnd values:
2
4
1
2
3
4
5
1
2
3
4
1
2
3
4
1
2
3
1
2
3
4
5
1
2
3
4
1
2
```

The congestion control part)

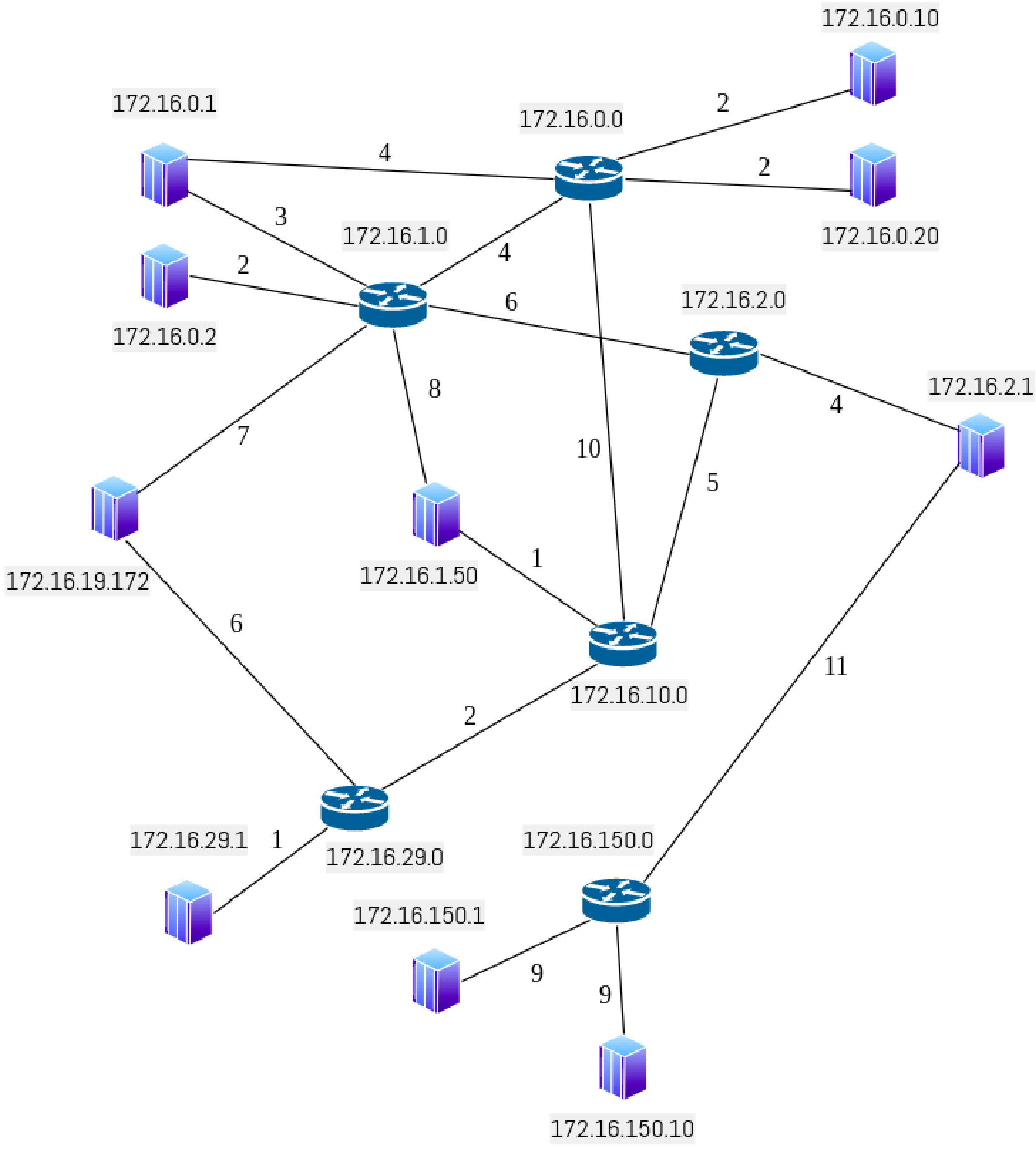
This part of the assignment is fairly related to the previous section explained. A new class name graph has been added to resemble a network. This module is tested by some instructions, for example, its edges can be updated or removed. Moreover, the DVRP routing algorithm is run on the graph, and the routing table of each node can be seen through entering the show_table <host port> command.

```cpp
#include "logger.hpp"
#include <experimental/filesystem>

const int INF = 10000;

using namespace std;

class Graph{
private:
    map<string, vector<vector<string>>> routingTables;
    map<pair<string, string>, int> edges;
    set<string> nodes;
    map<string, bool> isRouter;
    Logger logger;
public:
    Graph();
    vector<string> splitBySpace(string);
    void addHost(string);
    void addEdge(string);
    void removeLink(string);
    void bellmanFord(string);
    void showTable(string);
    // void printData();
    void DVRP();
    void printIteration(map<pair<string, string>, int>, int);
    void writeInFiles(string, map<string, int>, map<string, string>);
};

#endif
```

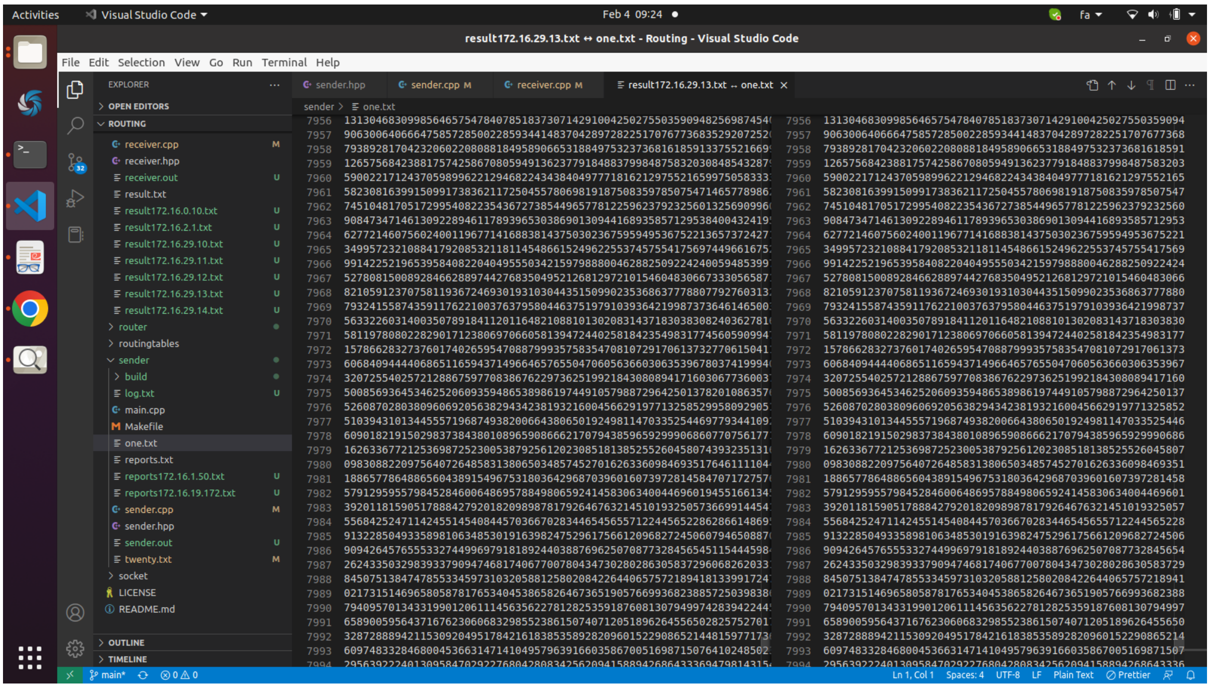


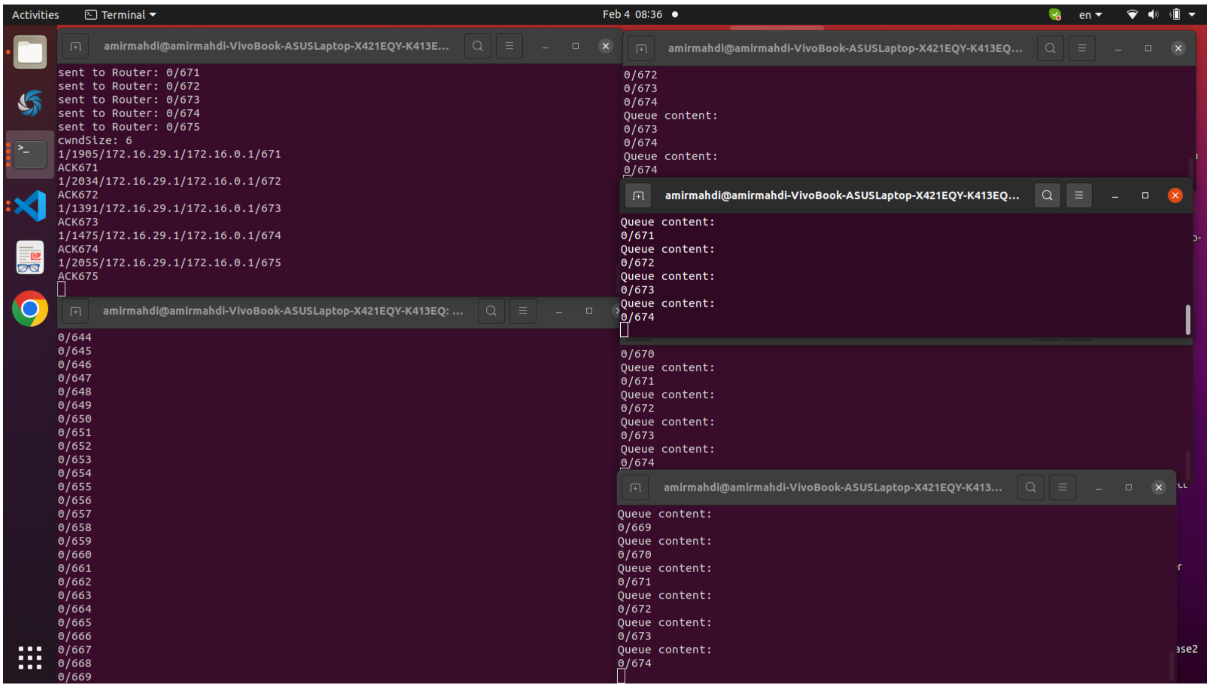
```
amirm...  amirm...  amirm...  amirm...  amirm...  amirm...  amirm...  amirm...  amirm...  amirm...  amirm...  amirm...
172.16.0.1      0   4   0   6   5   6   3   11  12  24  33  33  10  9   13  14  15
172.16.0.10     0   2   6   0   8   4   6   13  12  27  36  36  13  12  16  14  15
172.16.0.2      0   6   5   8   0   8   2   10  11  23  32  32  9   8   12  13  14
172.16.0.20     0   2   6   4   8   0   6   13  12  27  36  36  13  12  16  14  15
172.16.1.0      0   4   3   6   2   6   0   8   9   21  30  30  7   6   10  11  12
172.16.1.50     0   11  11  13  10  13  8   0   1   21  30  30  9   6   10  3   4
172.16.10.0     0   10  12  12  11  12  9   1   0   20  29  29  8   5   9   2   3
172.16.150.0    0   25  24  27  23  27  21  21  20  0   9   9   28  15  11  22  23
172.16.150.1    0   34  33  36  32  36  30  30  29  9   0   18  37  24  20  31  32
172.16.150.10   0   34  33  36  32  36  30  30  29  9   18  0   37  24  20  31  32
172.16.19.172   0   11  10  13  9   13  7   9   8   28  37  37  0   13  17  6   7
172.16.2.0      0   10  9   12  8   12  6   6   5   15  24  24  13  0   4   7   8
172.16.2.1      0   14  13  16  12  16  10  10  9   11  20  20  17  4   0   11  12
172.16.29.0     0   12  14  14  13  14  11  3   2   22  31  31  6   7   11  0   1
172.16.29.1     0   13  15  15  14  15  12  4   3   23  32  32  7   8   12  1   0

--------------------------------------------

ITERATION NO. 17
                172.16.0.0      172.16.0.1      172.16.0.10     172.16.0.2      172.16.0.20     172.16.1.0      172.16.1.50     172.16.10.0     172.16.150.0    172.16.1
50.1    172.16.150.10   172.16.19.172   172.16.2.0      172.16.2.1      172.16.29.0     172.16.29.1
        0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
172.16.0.0      0   0   4   2   6   2   4   11  10  25  34  34  11  10  14  12  13
172.16.0.1      0   4   0   6   5   6   3   11  12  24  33  33  10  9   13  14  15
172.16.0.10     0   2   6   0   8   4   6   13  12  27  36  36  13  12  16  14  15
172.16.0.2      0   6   5   8   0   8   2   10  11  23  32  32  9   8   12  13  14
172.16.0.20     0   2   6   4   8   0   6   13  12  27  36  36  13  12  16  14  15
172.16.1.0      0   4   3   6   2   6   0   8   9   21  30  30  7   6   10  11  12
172.16.1.50     0   11  11  13  10  13  8   0   1   21  30  30  9   6   10  3   4
172.16.10.0     0   10  12  12  11  12  9   1   0   20  29  29  8   5   9   2   3
172.16.150.0    0   25  24  27  23  27  21  21  20  0   9   9   28  15  11  22  23
172.16.150.1    0   34  33  36  32  36  30  30  29  9   0   18  37  24  20  31  32
172.16.150.10   0   34  33  36  32  36  30  30  29  9   18  0   37  24  20  31  32
172.16.19.172   0   11  10  13  9   13  7   9   8   28  37  37  0   13  17  6   7
172.16.2.0      0   10  9   12  8   12  6   6   5   15  24  24  13  0   4   7   8
172.16.2.1      0   14  13  16  12  16  10  10  9   11  20  20  17  4   0   11  12
172.16.29.0     0   12  14  14  13  14  11  3   2   22  31  31  6   7   11  0   1
172.16.29.1     0   13  15  15  14  15  12  4   3   23  32  32  7   8   12  1   0

--------------------------------------------

DVRP convergence time: 12.373 ms
```

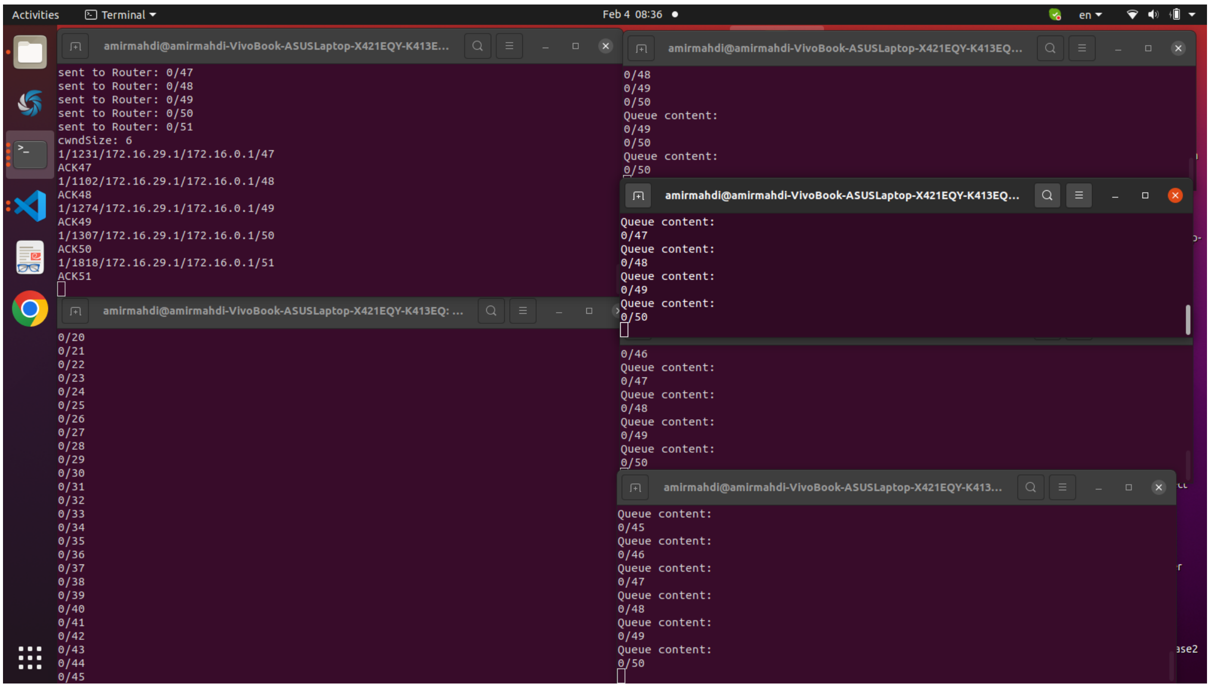
When we have implemented both the graph, which can detect the optimal path to each node, and the receiver, router, and sender modules, we are able to simulate the following network. Some connections like the ones described in the first topic should be established in this network. In these connections, the receiver nodes should rewrite the files which are sent to them through packets, and routers should handle the busy situation in which plenty of packets are obtained each second.

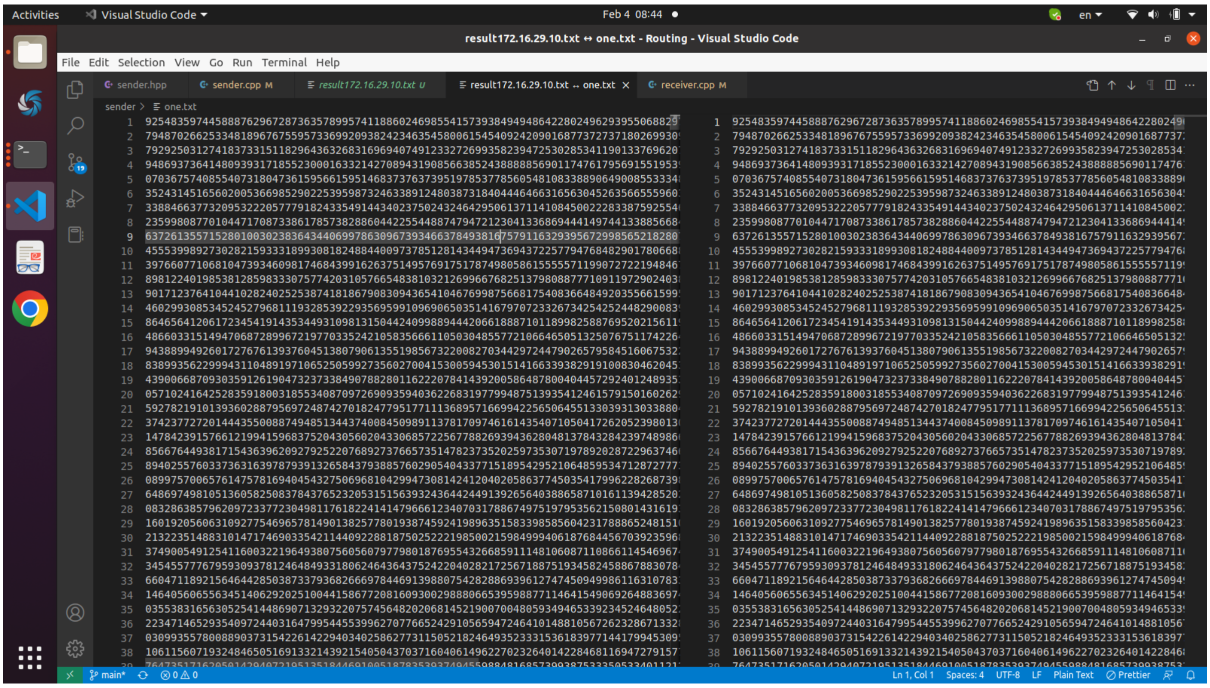


172.16.0.10
172.16.0.1
172.16.0.0
2
2
4
3
172.16.0.20
172.16.1.0
4
2
6
172.16.2.0
172.16.0.2
8
172.16.2.1
4
7
10
5
1
172.16.19.172
172.16.1.50
6
11
172.16.10.0
2
172.16.29.1
1
172.16.29.0
172.16.150.0
172.16.150.1
9
9
172.16.150.10


Activities
Visual Studio Code
Feb 4 09:24
result172.16.29.13.txt ↔ one.txt - Routing - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
ROUTING
receiver.cpp
receiver.hpp
receiver.out
result.txt
result172.16.0.10.txt
result172.16.2.1.txt
result172.16.29.10.txt
result172.16.29.11.txt
result172.16.29.12.txt
result172.16.29.13.txt
result172.16.29.14.txt
router
routingtables
sender
build
log.txt
main.cpp
Makefile
one.txt
reports.txt
reports172.16.1.50.txt
reports172.16.19.172.txt
sender.cpp
sender.hpp
sender.out
twenty.txt
socket
LICENSE
README.md
OUTLINE
TIMELINE
sender.hpp
sender.cpp
receiver.cpp
result172.16.29.13.txt ↔ one.txt
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text Prettier


Activities
Terminal
Feb 4 08:36
amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413E...
sent to Router: 0/671
sent to Router: 0/672
sent to Router: 0/673
sent to Router: 0/674
sent to Router: 0/675
cwndSize: 6
1/1905/172.16.29.1/172.16.0.1/671
ACK671
1/2034/172.16.29.1/172.16.0.1/672
ACK672
1/1391/172.16.29.1/172.16.0.1/673
ACK673
1/1475/172.16.29.1/172.16.0.1/674
ACK674
1/2055/172.16.29.1/172.16.0.1/675
ACK675
Queue content:
0/671
Queue content:
0/672
Queue content:
0/673
Queue content:
0/674


Activities
Terminal
Feb 4 08:36
en
amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413E...
sent to Router: 0/47
sent to Router: 0/48
sent to Router: 0/49
sent to Router: 0/50
sent to Router: 0/51
cwndSize: 6
1/1231/172.16.29.1/172.16.0.1/47
ACK47
1/1102/172.16.29.1/172.16.0.1/48
ACK48
1/1274/172.16.29.1/172.16.0.1/49
ACK49
1/1307/172.16.29.1/172.16.0.1/50
ACK50
1/1818/172.16.29.1/172.16.0.1/51
ACK51
amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413EQ: ...
0/20
0/21
0/22
0/23
0/24
0/25
0/26
0/27
0/28
0/29
0/30
0/31
0/32
0/33
0/34
0/35
0/36
0/37
0/38
0/39
0/40
0/41
0/42
0/43
0/44
0/45
amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413EQ...
0/48
0/49
0/50
Queue content:
0/49
0/50
Queue content:
0/50
amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413EQ...
Queue content:
0/47
Queue content:
0/48
Queue content:
0/49
Queue content:
0/50
0/46
Queue content:
0/47
Queue content:
0/48
Queue content:
0/49
Queue content:
0/50
amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413...
Queue content:
0/45
Queue content:
0/46
Queue content:
0/47
Queue content:
0/48
Queue content:
0/49
Queue content:
0/50


Activities
Visual Studio Code
Feb 4 08:44
en
result172.16.29.10.txt ↔ one.txt - Routing - Visual Studio Code
File Edit Selection View Go Run Terminal Help
sender.hpp
sender.cpp M
result172.16.29.10.txt U
result172.16.29.10.txt ↔ one.txt
receiver.cpp M
sender > one.txt
main*
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text Prettier

Project Questions)

2.3:
TCP is reliable and has the ability to resend the lost packets. On the other hand, UDP is connectionless and works based on the best-effort policy. As a result, it is faster than TCP and is used in multimedia programs.

2: In go-back-N, the sender sends all the packets with indexes from the lost packet to the current one. In selective-N, however, only the lost packet is retransmitted. It seems that selective-N is more complex, but through using the data structures of C++, especially map, lost packets can be easily identified. Therefore, we have used selective-N.

3.1.3: One way is to cluster the hosts near to each other, so that a link failure will cause changes in subnets only. I mean, less nodes are involved in the process of updating their tables.
In the distance vector method, the whole routing table is sent to hosts, so it takes a lot of bandwidth in comparison to IVRP. Another result of this form of routing is long-time convergence.

3.2.2 In the ECN method no packets are dropped, and the router sends a signal to sender convincing the sender to reduce its window size. On the contrary, RED protocole drops the packets based on calculating a probability. The main drawback of RED is dropping packets.