

## **Congestion control and TCP/UDP connection.**

Amirmahdi Ansaripour 810198358

Soroosh Hamed 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.

```

#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> thresholds;
    std::vector<std::string> packets;
    std::vector<std::string> content;
    Socket* toRouter;
    Socket* fromRouter;
    int lastPacketSent;

```

```

#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;
};

```

```

#ifndef __ROUTER_H__
#define __ROUTER_H__

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

const int QUEUE_SIZE = 10; // infinity (10000) is the first case
const int MAX_NUM_OF_PACKETS = 1100;
const int DELAY_COEF = 100;
const int DROP_PROBABILITY = 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;
};

```

```
#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



```
1 Duration: 228 s
2 Cwnd values:
3 2
4 4
5 1
6 2
7 3
8 4
9 5
10 1
11 2
12 3
13 4
14 1
15 2
16 3
17 4
18 1
19 2
20 3
21 1
22 2
23 3
24 4
25 5
26 1
27 2
28 3
29 4
30 1
31 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.



```

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

const int INF = 10000;

using namespace std;

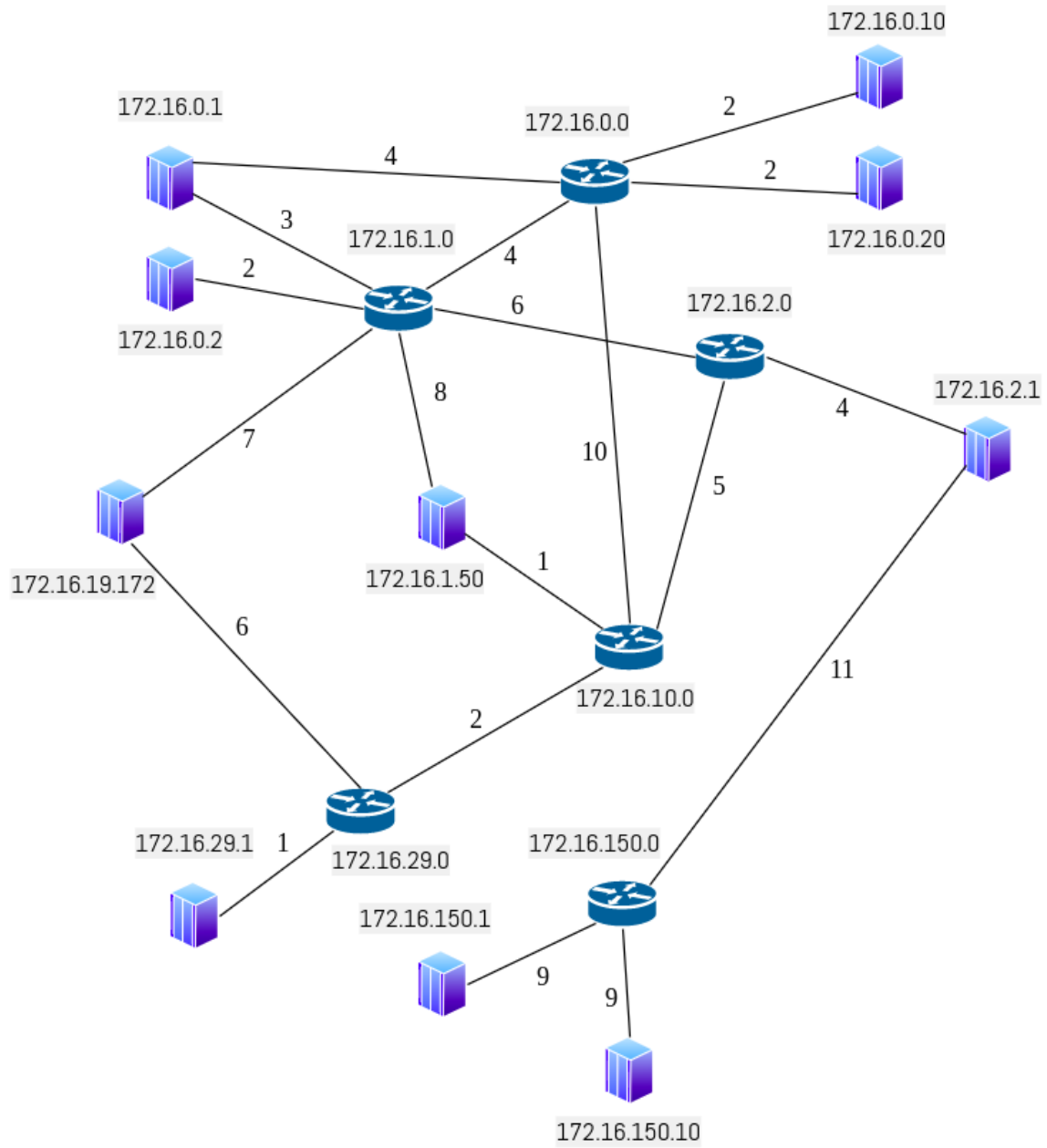
class Graph{
private:
    map<string, vector<vector<string>>> routingTables;
    map<pair<string, string>, int> edges;
    set<string> nodes;
    map<string, bool> isRouter;
    Logger logger;
public:
    Graph();
    vector<string> splitBySpace(string);
    void addHost(string);
    void addEdge(string);
    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...	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																	
50.1	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							
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.



Visual Studio Code interface showing a C++ project named "result172.16.9.13.txt" with a routing table. The Explorer sidebar shows files like sender.hpp, sender.cpp, receiver.cpp, and receiver.hpp. The main editor displays a routing table with columns for source IP, destination IP, and next hop. The table contains 24 rows of data, including source IP, destination IP, and next hop information.

Source IP	Destination IP	Next Hop
13.138.468.399556465	7547840785183738714291084250527583590948256987454	13.138.468.4066647585728506228593441483748927822517677368
7956	13.138.468.4066647585728506228593441483748927822517677368	7956
7957	7938928176423260280881849589606531848795323736816185913375521669	7957
7958	7938928176423260280881849589606531848795323736816185913375521669	7958
7959	7938928176423260280881849589606531848795323736816185913375521669	7959
7960	7938928176423260280881849589606531848795323736816185913375521669	7960
7961	7938928176423260280881849589606531848795323736816185913375521669	7961
7962	7938928176423260280881849589606531848795323736816185913375521669	7962
7963	7938928176423260280881849589606531848795323736816185913375521669	7963
7964	7938928176423260280881849589606531848795323736816185913375521669	7964
7965	7938928176423260280881849589606531848795323736816185913375521669	7965
7966	7938928176423260280881849589606531848795323736816185913375521669	7966
7967	7938928176423260280881849589606531848795323736816185913375521669	7967
7968	7938928176423260280881849589606531848795323736816185913375521669	7968
7969	7938928176423260280881849589606531848795323736816185913375521669	7969
7970	7938928176423260280881849589606531848795323736816185913375521669	7970
7971	7938928176423260280881849589606531848795323736816185913375521669	7971
7972	7938928176423260280881849589606531848795323736816185913375521669	7972
7973	7938928176423260280881849589606531848795323736816185913375521669	7973
7974	7938928176423260280881849589606531848795323736816185913375521669	7974
7975	7938928176423260280881849589606531848795323736816185913375521669	7975
7976	7938928176423260280881849589606531848795323736816185913375521669	7976
7977	7938928176423260280881849589606531848795323736816185913375521669	7977
7978	7938928176423260280881849589606531848795323736816185913375521669	7978
7979	7938928176423260280881849589606531848795323736816185913375521669	7979
7980	7938928176423260280881849589606531848795323736816185913375521669	7980
7981	7938928176423260280881849589606531848795323736816185913375521669	7981
7982	7938928176423260280881849589606531848795323736816185913375521669	7982
7983	7938928176423260280881849589606531848795323736816185913375521669	7983
7984	7938928176423260280881849589606531848795323736816185913375521669	7984

Activities Terminal Feb 4 08:36

```

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
1/1985/172.16.29.1/172.16.0.1/671
ACK671
1/2034/172.16.29.1/172.16.0.1/672
ACK672
1/1394/172.16.29.1/172.16.0.1/673
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

0/644
0/645
0/646
0/647
0/648
0/649
0/650
0/651
0/652
0/653
0/654
0/655
0/656
0/657
0/658
0/659
0/660
0/661
0/662
0/663
0/664
0/665
0/666
0/667
0/668
0/669

0/672
0/673
0/674
Queue content:
0/673
0/674
Queue content:
0/674

amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413EQ...
Queue content:
0/671
Queue content:
0/672
Queue content:
0/673
Queue content:
0/674
Queue content:
0/674

0/670
Queue content:
0/671
Queue content:
0/672
Queue content:
0/673
Queue content:
0/674

amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413EQ...
Queue content:
0/669
Queue content:
0/670
Queue content:
0/671
Queue content:
0/672
Queue content:
0/673
Queue content:
0/674

```

```
Activities Terminal Feb 4 08:36
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

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-K413E...
0/48
0/49
0/50
Queue content:
0/49
0/50
Queue content:
0/50
0/50

amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413E...
Queue content:
0/47
Queue content:
0/48
Queue content:
0/49
Queue content:
0/50
0/50

amirmahdi@amirmahdi-VivoBook-ASUSLaptop-X421EQY-K413E...
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
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 x receiver.cpp M

sender // one.txt
1 925483597445888762967287363578995741188602469855415739364949486422802496293955068028
2 794870266253348189676755957336992893824234635458006154540924209016877372727180269930
3 792925031274183733151182964363268316969407491233272699358239472530285341190133769620
4 9486937364148093931718552300016332142708943198856638524388885690117476195691551953
5 07836757408554073180473615956615951468373763739519785377856054810833889964908855334
6 35243145165602053698529822539598732463389124803873184044464631656304526356655960
7 33884663773209532220577918243354914434023750243246429566137114108450022283387592554
8 2359888776104471788733861785738288604422554488747947212304133686944149744133885668
9 637261355715280108302383643440699786389673934663784938107579116329395672989565218280
10 45553998927202815533318993081824884400973785120143440726043722579746842801708668
11 39766877106810473934609817468439916263751495769175178749805861555571198072722194846
12 8981224019853812859833875774283186796848381032126996676825137980887710911972902403
13 901712376410441028240252538741818679830943654104676987566817540836484920355661599
14 460299308534524527968111932853922935695910969650351416797072332673425425448290803
15 8646564120617234541914354493109813150442409889444206618887101189982581
16 48660331514947068728996721977033524210583566611050304855772106646505132507675174226
17 94388949260172767613937604513807906135519856732208827034429724479826579584516067532
18 838993562299943110489197106525059927356027004153005945301514166339382919100830642045
19 43900687093035912619047323733849078828011622207841439208058048780040445752401248935
20 057102416425235910083185340076929093594036226811979948751330441246159150160262
21 5927821910139360288795697248747018247795171113689571669942256506455133039313033880
22 374237272720144435500887494851344740084509801137817097461614354071050417262052380913
23 14784239157661219941596837520430560204306857225677882693943628048137843284239748986
24 85667644938171543639620927925220768927376657351478237352025973530719789202872963746
25 89402557603373631639787939132658437938857602905404337715189542952106485953471287727
26 08997570605761475781694045432756968104299473081424120402058637745035417996228268739
27 64869749810513605825083784376523205315156393243642449139265640388658710161139428520
28 683286385796209723377230498117618224141479666123407831788674975197953562150801431619
29 1601920506631892775469657814901302577091930745924198963515833980560422178865248151
30 213223514883101471746903542114409218818750252219850821598499946187684456703923596
31 3749005491254116083221964938075605607798018769554326685911148106087110866114546097
32 345455777679593093781246484933180624643643752422040282172567188751934582458867883078
33 66047118921564644285038737393682669784469139880754282886939612747450949986116318783
34 14640560655634514062920251004415867720816093082988066539598877114641549069264683697
35 03553831656305254144869071329322075745648202681452190070048059349465339234524648052
36 223471465293540972440316479954455399627077665242910565947246410148810567262328671332
37 030993557800889037315422614229403402586277311505218246493523331536183971744179945309
38 106115607193248465051691332143921540504370371604661496227023264014228406116947279157
39 5642321736052170106723131276312763106051203820701855001657300207123350321061131
40 7474251736052170106723131276312763106051203820701855001657300207123350321061131

Ln 1, Col 1 Spacex4 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 IGRP. 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 protocol drops the packets based on calculating a probability. The main drawback of RED is dropping packets.