UNIVERSITY OF MARYLAND, COLLEGE PARK

NETWORKS AND PROTOCOLS I

ENTS 640

Project Report

On

Reliable Data Transfer using User Datagram Protocol

Submitted by:

SHREYAS PANDEY

UID: 115577794

SOWMIYA IYENGAR

UID: 115681873

CONTENTS: PAGE NO.

1. Problem Discussion 3
2. Code Overview 4
3. Client 5
4. UML DIAGRAM 5
5. ALGORITHM 6
6. FLOW CHART 8
7. SERVER 12
8. UML DIAGRAM 12
9. ALGORITHM 13
10. FLOWCHART 14
11. STATE TRANSITION DIAGRAMS 16
12. NORMAL OPERATION 16
13. LOST IACK PACKET 17
14. OUTPUT 18
15. Output for different cases between client 18

and server

1. Server Output 20
2. Client Output 21
3. Timeout Error 22
4. Integrity Check Error 22
5. Packet Type Error 22
6. Graphical Measurements 23

PROBLEM DISCUSSION

User Datagram Protocol (UDP) is an unreliable protocol. It uses best effort service but still it makes no guarantee that the data will be delivered successfully. We are using UDP socket programming in order to communicate between client(transmitter) and server(receiver). Since unreliable protocol is being used, to ensure that the data is delivered, handshake communication is laid over the UDP sockets and a data transfer protocol is used. The program also measures Round Trip Time (RTT) and the overall data rate. Here we are implementing one direction transfer of data from client to server.

For establishing communication, the initial phase sends and receives the respective packets, INIT and IACK, to build a reliable connection between client and server over which the second phase, that is, the actual transfer of data takes place which uses the packets, DATA and DACK for data transfer and acknowledgement, respectively. The four different packets, INIT, IACK, DATA and DACK is recognized by their packet type.

Integrity check plays very important role by checking for bit errors to ensure reliable data transfer. Timer is added on the client side. If the acknowledgement is not received within a stipulated time the timer is incremented. However, after several failures, an error message is also displayed and the timer is reset. A timer on the server side maintains a record for the overall rate for the complete process.

CODE OVERVIEW

The code is made as flexible as possible with comments added in places to make understanding of the code better. The client and the server both check for errors and if found any, are discarded.

\*Integrity Check field:

To calculate integrity check we have divided the byte array into two separate fields, even and odd. The respective fields that is, the even field of byte array is XORed with the even byte array fields and final result is then XORed with the even byte array of integrity check field. In case of client side, the integrity check is assigned a new variable containing all 0’s and the XORed result is stored in the integrity check field. For the server side, the integrity check field is also XORed with the final result. Thus an integrity with an end result of all 0’s indicates a reliable connection. The process is same with the odd byte array.

Number of packets, length of packets, payload field, packet type and port numbers are predefined. Counters are implemented for timers and for some special purposes.

\*Timer:

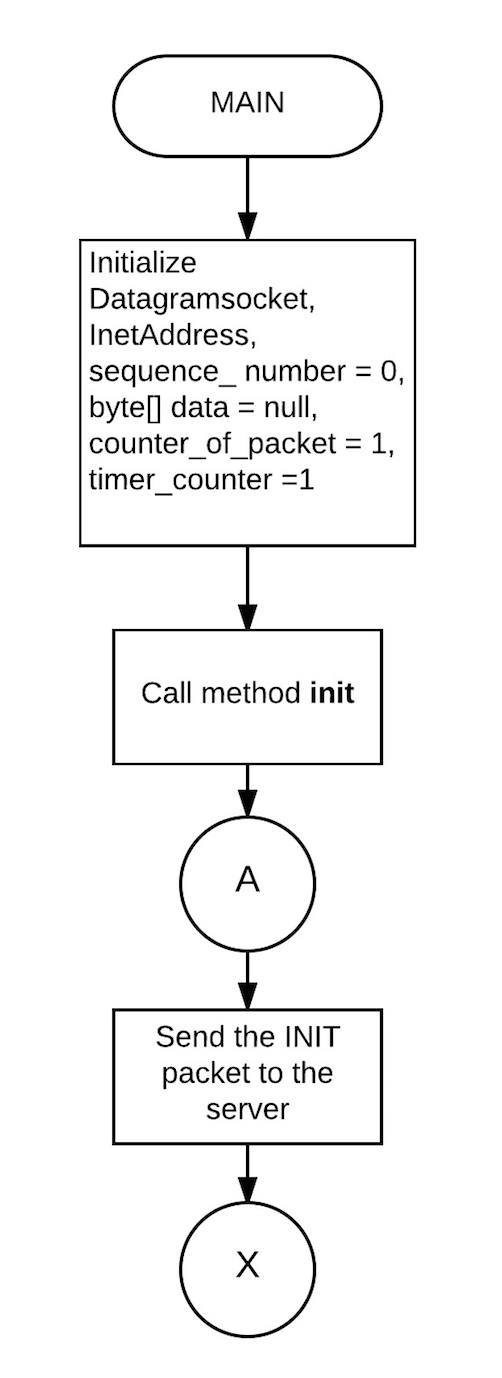
The timer should be started as soon as INIT and DATA packets are sent to the server. The initial time is assigned to be 1 second. If the timer expires before the acknowledgement is received it should be doubled and this is to be done each time the timer ceases. After the 4th time out, the client should announce about the communication failure by sending an error message. In case of DACK, if the acknowledgement is received after the time outs and retransmissions the timer is supposed to be set again to the initial values.

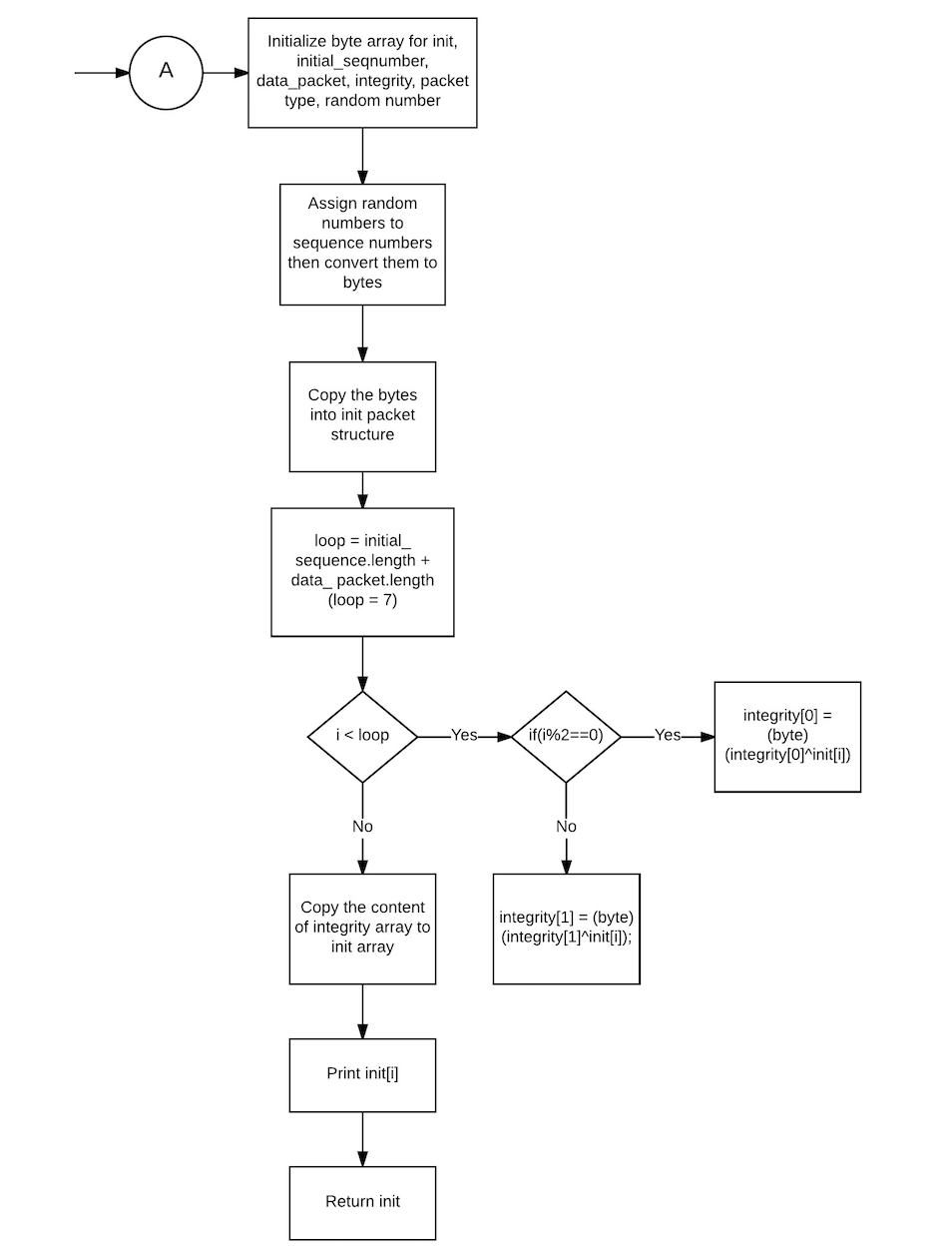
CLIENT

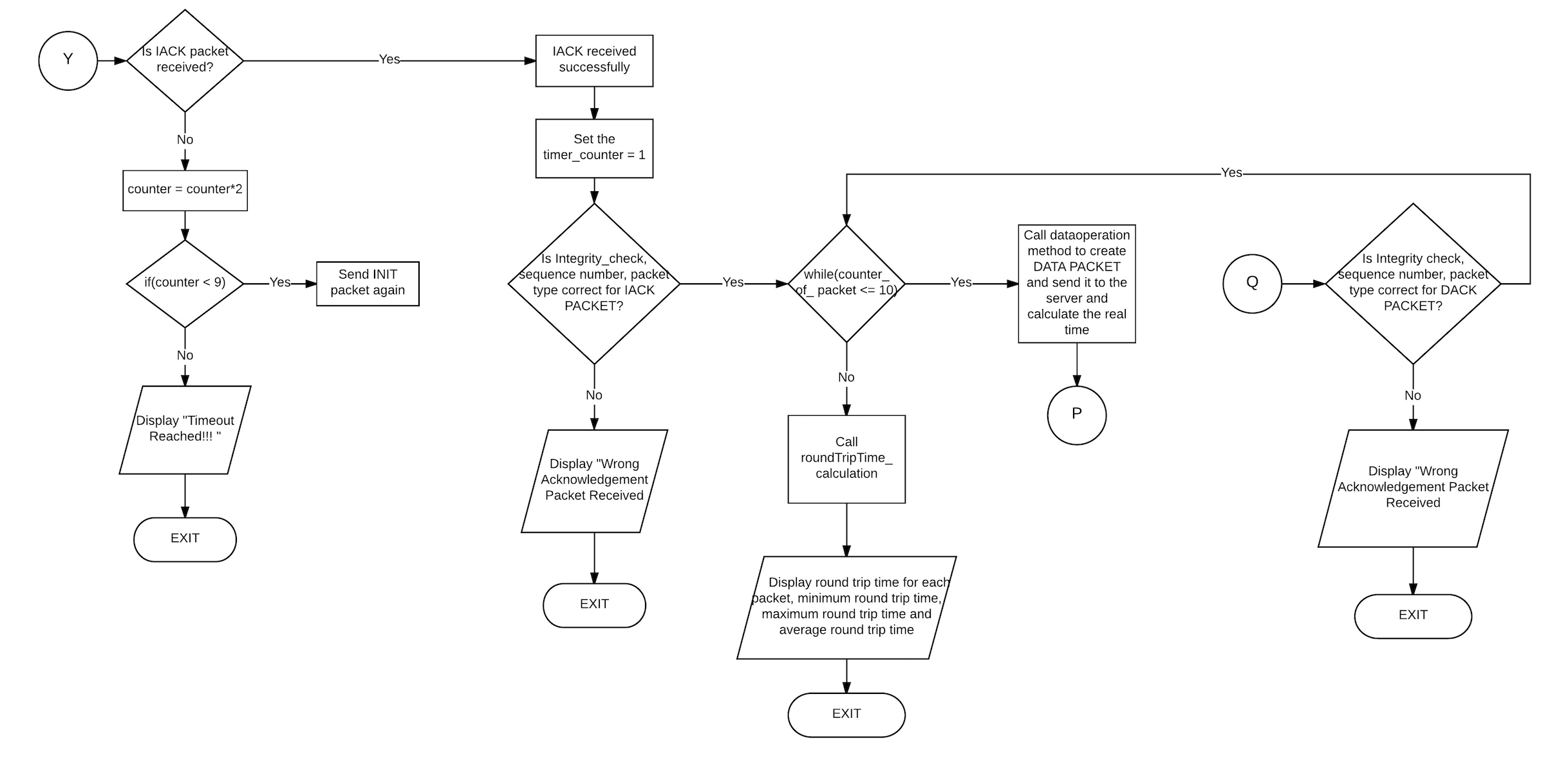
1. UML DIAGRAM:

|  |
| --- |
| Client\_UDP |
| - NUMBER\_OF\_PACKETS: int, static, final  - PORT\_NUMBER: int, static, final  - INIT\_LENGTH: int, static, final  - IACK\_LENGTH: int, static, final  - DATA\_LENGTH: int, static, final  - DACK\_LENGTH: int, static, final  - INTEGRITYCHECK\_BYTES: int, static, final  - PALOAD\_BYTES: int, static, final  - SEQUENCENUMBER\_BYTES: int, static, final  - INIT\_PACKET\_TYPE: int, static, final  - IACK\_PACKET\_TYPE: int, static, final  - DATA\_PACKET\_TYPE: int, static, final  - DACK\_PACKET\_TYPE: int, static, final  - COUNTER: int, static, final  - time\_data\_packet: long (static)  - time\_dack\_packet: long (static)  - counter\_of\_packet: int (static) |
| + main()  - integritycheck(byte[] data): boolean (static)  - byteToIntegerSequenceConversion(byte[] data): int (static)  - init\_method(): byte (static)  - dataoperation(int ackNumber): byte (static)  - rountTripTime\_calculation(Long[] time\_calculation) |

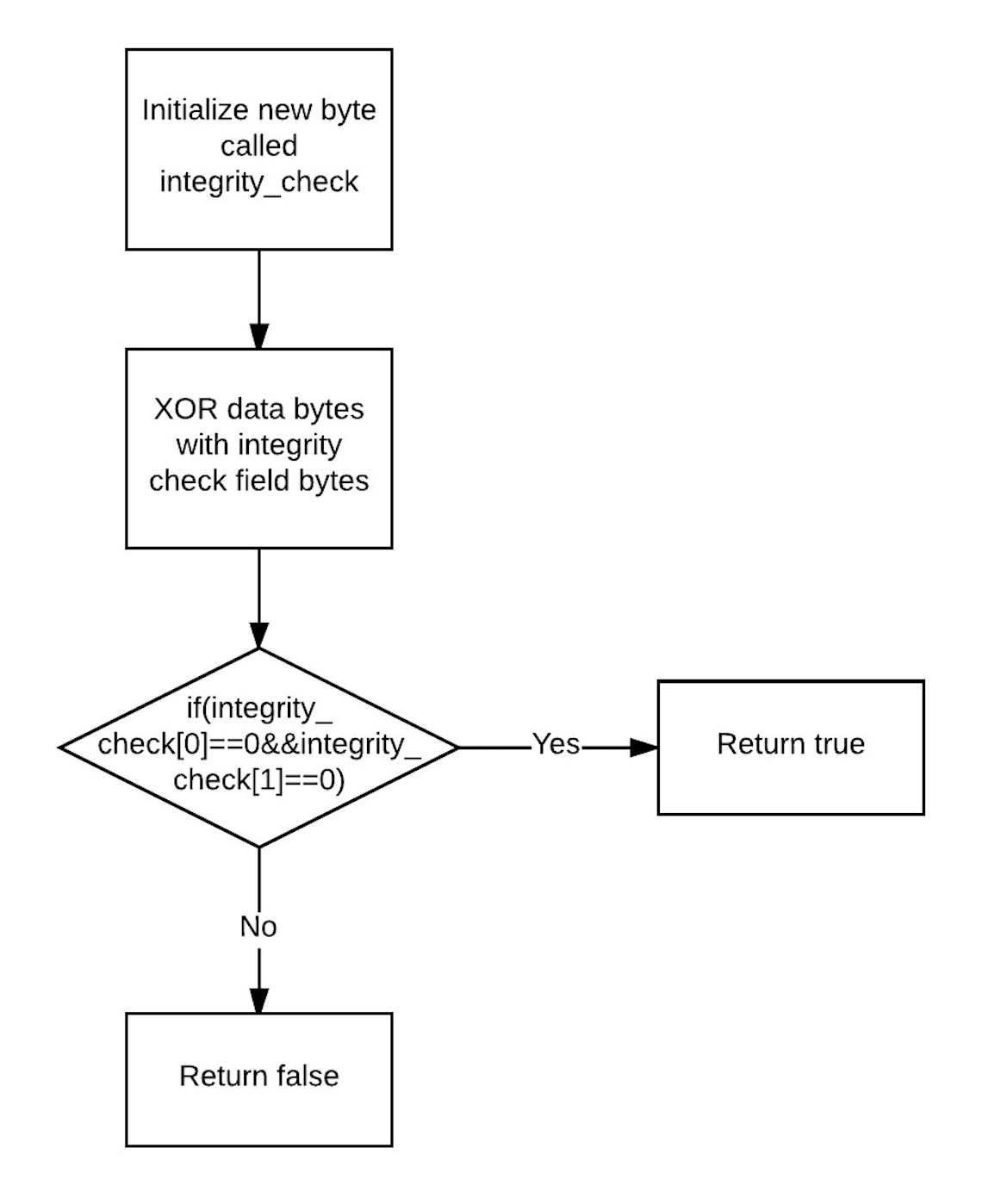
1. ALGORITHM:
2. Initialize various variables like Datagramsocket, InetAddress, sequence\_number, counter\_of\_packet and timer\_counter.
3. Call method INIT.
4. Initialize variables such as byte array, initial\_seqnumber, data\_packet, integrity, packet type, random number.
5. Assign random numbers to sequence numbers in the form of bytes.
6. Copy these bytes into the INIT packet structure.
7. Set the loop count equal to 7.
8. If i < 7, check whether i is an even number.
9. If yes then XOR even byte array of INIT packet with the even array of integrity (integrity is equal to all 0’s).
10. If no then XOR odd byte array of INIT packet with the odd array of integrity (integrity is equal to all 0’s).
11. When i becomes more than 7, copy the contents of integrity which is calculated in step f and g, into INIT packet.
12. Print the INIT packet.
13. Return INIT packet.
14. Send the INIT packet to the server.
15. Check whether IACK packet is received.
16. If the packet is retrieved successfully then set the timer\_counter = 1.
17. Check whether the received IACK packet contains the correct packet type, correct sequence number and correct integrity check.
18. If all the conditions are satisfied, start a counter to send 10 DATA packets and start creating the DATA packets and send them to the server one by one.
19. For each DATA packet sent successfully to the server, an acknowledgement packet, DACK is received.
20. Check the DACK contents for the required conditions like correct packet type, correct sequence number and correct integrity check.
21. If all these conditions are satisfied send the next DATA packet.
22. If any of the conditions fail, exit from the system.
23. When 10 DATA packets are sent successfully, calculate the round trip time for each packet, the minimum round trip time, the maximum round trip time and the average round trip.
24. Display all the calculated round trip time.
25. Exit the system.
26. If any of the conditions fail while checking the contents of IACK packet, display an error message and exit from the system.
27. FLOWCHART:

******

******

******

Integrity Check:

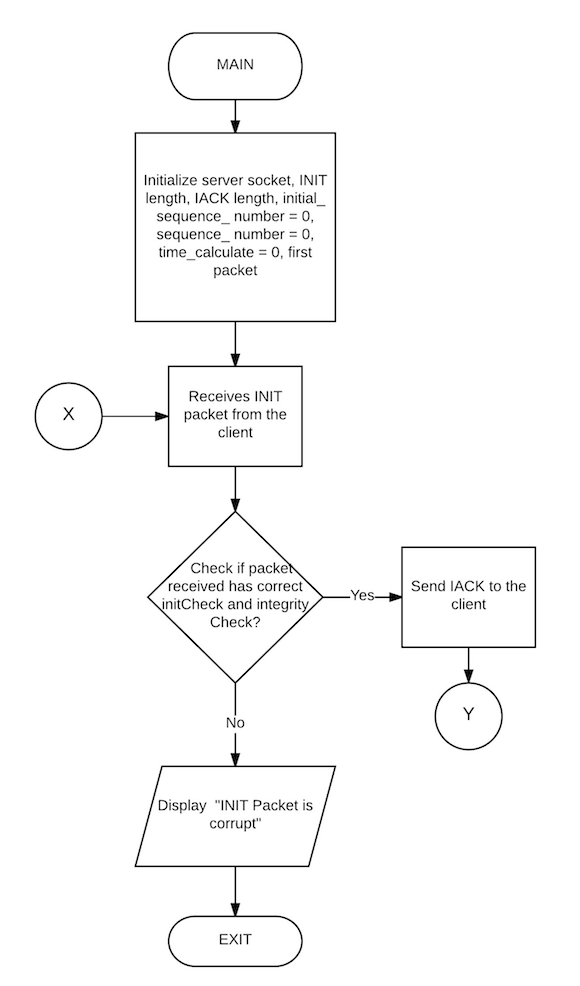
******

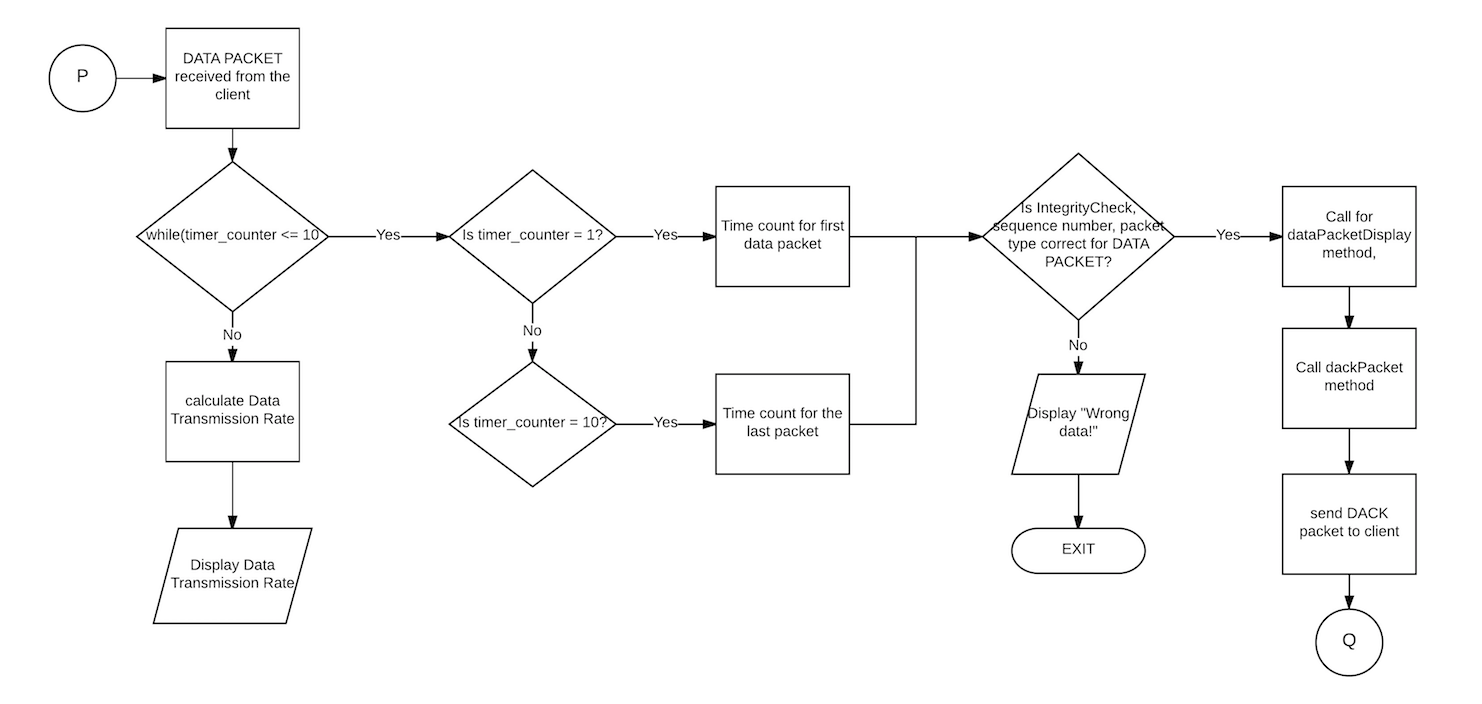
SERVER

1. UML DIAGRAM:

|  |
| --- |
| serverlastproject |
| - NUMBER\_OF\_PACKETS: int, static, final  - PORT\_NUMBER: int, static, final  - INIT\_LENGTH: int, static, final  - IACK\_LENGTH: int, static, final  - DATA\_LENGTH: int, static, final  - DACK\_LENGTH: int, static, final  - INTEGRITYCHECK\_BYTES: int, static, final  - PALOAD\_BYTES: int, static, final  - SEQUENCENUMBER\_BYTES: int, static, final  - INIT\_PACKET\_TYPE: int, static, final  - IACK\_PACKET\_TYPE: int, static, final  - DATA\_PACKET\_TYPE: int, static, final  - DACK\_PACKET\_TYPE: int, static, final |
| + main()  - byteToIntegerSequenceConversion(byte[] data): int (static)  - dataPacketDisplay(byte[] data)  - dackPacket(int sequence\_number): byte (static)  - initCheck(byte[] init): boolean (static)  - iackPacket(int initial\_sequence\_number): byte (static)  -integrityCheck(byte[] packet): boolean (static) |

1. ALGORITHM:
2. Initialize server socket, INIT length, IACK length, initial\_sequence\_number, sequence\_number, time\_calculate and first packet.
3. The INIT packet send by the client is received.
4. Check the contents of the received packet like packet type and integrity check.
5. If the contents are as required, send an acknowledgement packet, IACK, back to the client.
6. If the conditions are not satisfied, display an error message and exit from the system.
7. DATA packet is received from the client.
8. Set the timer\_counter to a maximum of 10.
9. When the timer\_counter is equal to 1, calculate the time the first data packet took to reach the server.
10. When the timer\_counter is equal to 10, calculate the time the last data packet took to reach the server.
11. After calculating the time for the received first packet, check for the required conditions like packet type, sequence number and integrity check.
12. When all the conditions are satisfied, create an acknowledgement DACK packet and send it to the client.
13. If even one of the conditions fail while verifying DATA packet, display an error and exit from the system.
14. When the timer\_counter is more than 10, calculate the DATA retransmission rate.
15. Display the DATA retransmission rate.
16. FLOW CHART:

******

******

STATE TRANSITION DIAGRAM

1. State Transition Diagram for Normal Operation

Server

Client

INIT packet

Start, INIT

Timer= 1 sec INIT Received.

Send IACK

IACK received.

Time out. Start

IACK PACKET

sending DATA

DATA packet 1

DATA PACKET 1

RECEIVED.

SEND DACK 1

RECEIVED.

DACK 1

SEND DATA

PACKET 2.

DATA packet 2

DATA PACKET 2

RECEIVED.

SEND DACK 2

RECEIVED.

DACK 2

SEND DATA

PACKET 2.

DATA packet 9

DATA PACKET 9

RECEIVED.

SEND DACK 9

RECEIVED.

DACK 9

SEND DATA

PACKET 2.

DATA packet 10

DATA PACKET 10

RECEIVED.

SEND DACK 10

DACK 10

RECEIVED.

SYSTEM EXIT

1. STATE TRANSITION DIAGRAM FOR LOST IACK PACKET

Server

Client

INIT packet

Start, INIT

Timer= 1 sec RECEIVED.

SEND IACK

IACK not received.

Timer = 2 sec

IACK PACKET LOST

INIT packet

Send INIT again

RECEIVED.

SEND IACK

IACK not received.

IACK PACKET LOST

Timer = 4 sec.

INIT packet

Send INIT again

RECEIVED.

SEND IACK

IACK not received.

IACK PACKET LOST

Timer = 8 sec.

INIT packet

Send INIT again

RECEIVED.

SEND IACK

IACK not received.

IACK PACKET LOST

Time Out.

System Exit

OUTPUT

1. Output for different cases between client and server: -
   1. Case a: -

When client and server are being operated from the same system as two different process.

Packet1 round trip time:- 40milliseconds

Packet2 round trip time:- 35milliseconds

Packet3 round trip time:- 26milliseconds

Packet4 round trip time:- 17milliseconds

Packet5 round trip time:- 16milliseconds

Packet6 round trip time:- 66milliseconds

Packet7 round trip time:- 38milliseconds

Packet8 round trip time:- 47milliseconds

Packet9 round trip time:- 132milliseconds

Packet10 round trip time:- 126milliseconds

Minimum Round Trip Time:- 16milliseconds

Maximum Round Trip Time:- 132milliseconds

Average Round Trip Time:- 54milliseconds

* 1. Case b: -

When client and server are connected via an Ethernet cable.

Packet1 round trip time:- 1042milliseconds

Packet2 round trip time:- 1028milliseconds

Packet3 round trip time:- 1017milliseconds

Packet4 round trip time:- 1013milliseconds

Packet5 round trip time:- 1015milliseconds

Packet6 round trip time:- 1009milliseconds

Packet7 round trip time:- 1009milliseconds

Packet8 round trip time:- 1012milliseconds

Packet9 round trip time:- 1011milliseconds

Packet10 round trip time:- 1012milliseconds

Minimum Round Trip Time:- 1009milliseconds

Maximum Round Trip Time:- 1042milliseconds

Average Roundd Trip Time:- 1016milliseconds

* 1. Case c: -

When client and server are connected via WiFi connection.

Packet1 round trip time:- 1041milliseconds

Packet2 round trip time:- 1046milliseconds

Packet3 round trip time:- 1023milliseconds

Packet4 round trip time:- 1018milliseconds

Packet5 round trip time:- 1023milliseconds

Packet6 round trip time:- 1016milliseconds

Packet7 round trip time:- 1002milliseconds

Packet8 round trip time:- 1004milliseconds

Packet9 round trip time:- 1013milliseconds

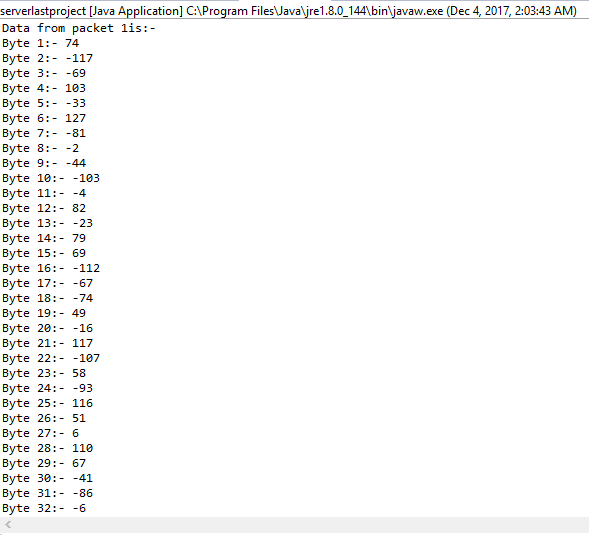
Packet10 round trip time:- 1009milliseconds

Minimum Round Trip Time:- 1002milliseconds

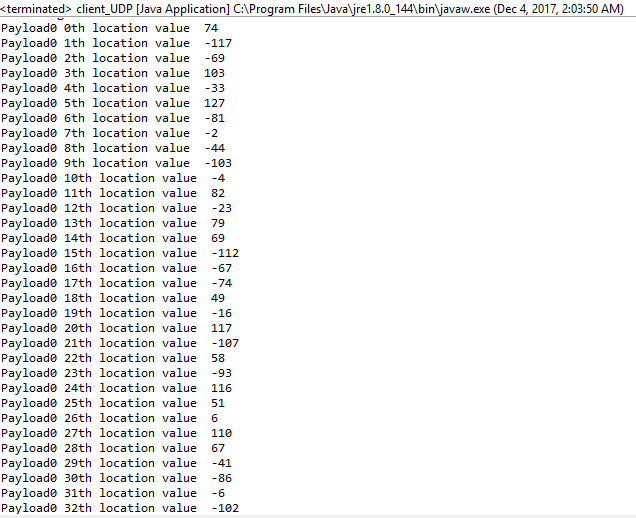
Maximum Round Trip Time:- 1046milliseconds

Average Round Trip Time:- 1019milliseconds

1. Server Output: -



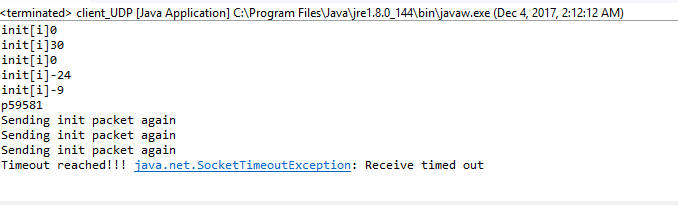
1. Client Output: -



1. Timeout Error: -

(When server is off.)

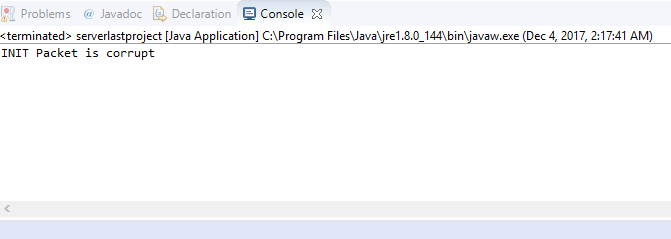
Output of client side:



1. Integrity Check Error: -

(When INIT integrity check is wrong.)

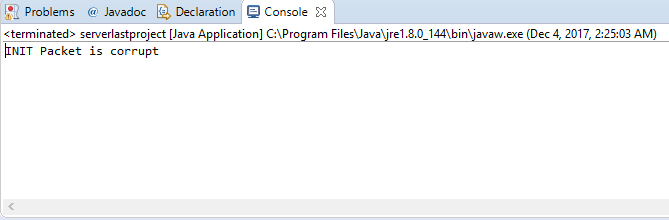
Output of server side:



1. Packet Type Error: -

(When INIT Packet is wrong.)

Output of server side:



1. Graphical Measurements: -

DATA rate is high when same computer is used and the round trip is very low as compared to using an Ethernet cable and a WiFi connection.

