Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Summer, Year:2023), B.Sc. in CSE (Day)

Lab Report NO #04
Course Title: Computer Networking Lab
Course Code: CSE 312 Section: 212-D3

Lab Experiment Name: Implementation of TCP Congestion Control Mechanism: TCP Tahoe and TCP Reno.

Student Details

	Name	ID
1.	Sajid Rahman Rifan	212902017

Lab Date : 20/12/2023
Submission Date : 27/12/2023
Course Teacher's Name : Md. Noyan Ali

Lab Report Status

Marks: …… **Signature:.....**
Comments:..... **Date:.....**

1. TITLE OF THE LAB REPORT EXPERIMENT:

Implementations of Flow and Congestion Controls could be demonstrated, using the same programming code of JAVA.

2. OBJECTIVES/AIM

- Implement a simple sliding window protocol for flow control on the sender side.
- Sender will not send more data than the window size.
- Receiver will acknowledge received data and update the window.
- Implement a basic congestion control mechanism, like a simple congestion window.
- Adjust the sending rate based on congestion window size.

3. PROCEDURE

The implementation is based on the provided Java code for TCP Congestion Control. The code simulates a basic data transmission scenario and incorporates both Flow Control and Congestion Control mechanisms. The simulation is conducted in a controlled environment, allowing us to observe the behavior of the system.

Environment:

- Operating System: Windows/Linux/macOS
- Java Development Kit (JDK): Version 8 or higher
- Text Editor or Integrated Development Environment (IDE)

Implementation Details:

1.Flow Control:

Flow Control in TCP is implemented through the concept of a sliding window. The cwnd (congestion window) variable represents the size of the window. In the code, the program implements Slow Start and Congestion Avoidance phases to control the flow of data based on the congestion window size.

2.Congestion Control:

Congestion Control is simulated by introducing randomness in acknowledgment reception and congestion detection. The code handles congestion through Timeout and Three Duplicate Acknowledgment mechanisms. The ssthresh variable represents the slow start threshold.

4. IMPLEMENTATION

```
package pkg456;

import java.util.*;

public class
TCPFlowAndCongestionControl {

   private int cwnd;          // Congestion
window size
   private int ssthresh;      // Slow start
threshold
   private int rtt;           // Round-trip time
   private boolean congestion; //
Congestion flag

   public
TCPFlowAndCongestionControl(int
init_ssthresh) {
      cwnd = 1;
      ssthresh = init_ssthresh;
      congestion = false; rtt =
      0;
   }

   public void run(int dataLength) {
      System.out.println("Connected to the
Server... ...");
      System.out.println("Your data is
started to be sent ... ");

      int dataSeqNum = 0;

      while (dataSeqNum < dataLength) {
         this.rtt++;
         System.out.println();
         System.out.println("Data sending
in RTT number " + this.rtt);

System.out.println("−−−−−−−−−−−−−−−−
−−−−−−−−−−−−");

         // Flow Control
         flowControl();

         System.out.println("Data from " +
(dataSeqNum + 1) + " − " + (dataSeqNum +
cwnd) + " is being sent now... ...\n\n");

         // Congestion Control
         if (!receiveAcknowledgment()) {
            congestion = true;
            System.out.println("... but wait!
Congestion has been detected!");
            if (timeout()) {
               handleTimeoutCongestion();
            } else {
handle3DupAckCongestion();
      Random ack = new
Random();
      return ack.nextBoolean();
   }

   private boolean timeout() {


 }
         } else {
            congestion = false;
         }

         dataSeqNum += cwnd;
      }

      System.out.println("\n\nYour
data sending is completed. No
more data to send."
            + "\nCongestion Control
mechanism concludes.\nIt took " +
this.rtt + " transmission rounds to
send the whole data.");
   }

   private void flowControl() {
      System.out.println("previous
cwnd size: " + cwnd);
      System.out.println("updated
ssthresh value: " + ssthresh);

      if (!congestion) {
         if (cwnd < ssthresh) {
            // Slow Start Phase
            // Exponentially increase
of cwnd
            cwnd = cwnd * 2;
            System.out.println("...SS
phase running...");
         } else if (cwnd >= ssthresh)
{
            // Congestion Avoidance
Phase
            // Linearly increase of
cwnd
            cwnd = cwnd + 1;
            System.out.println("...CA
phase running...");
         }
      }

      System.out.println("updated
cwnd size: " + cwnd);
   }

   private boolean
receiveAcknowledgment() {
detected using timeout.

      Random rttRandom = new
Random();
      return rttRandom.nextBoolean();
   }

   private void
handleTimeoutCongestion() {
      System.out.println("\n\nTimeout
occurred. Handling Timeout based
congestion: cwnd value will become 1.");

      ssthresh = cwnd / 2;
      if (ssthresh == 0) ssthresh = 1; //
Making ssthresh 1 if it comes as zero.
      cwnd = 1;

      retransmitPacket();
   }

   private void
handle3DupAckCongestion() {
      System.out.println("\n\nHandling
Triple Dup Ack based congestion: cwnd
value will be halved.");
      ssthresh = cwnd / 2;
      if (ssthresh == 0) ssthresh = 1; //
Making ssthresh 1 if it comes as zero.
      cwnd = ssthresh;

      retransmitPacket();
   }

   private void retransmitPacket() {
      congestion = false;
      System.out.println("\nRetransmitting
the lost packet now after handling.\n");
   }
   public static void main(String[] args) {
      Scanner scn = new
Scanner(System.in);
      System.out.println("Please input the
initial ssthresh value: ");
      int ssthresh = scn.nextInt();

      System.out.println("Enter the length
of your data: ");
      int dataLength = scn.nextInt();

      TCPFlowAndCongestionControl
simulation = new
TCPFlowAndCongestionControl(ssthresh)
;
      simulation.run(dataLength);
   }
}
```

5. TEST RESULT / OUTPUT

```
run:
Please input the initial ssthresh value:
10
Enter the length of your data:
20
Connected to the Server... ...
Your data is started to be sent ...

Data sending in RTT number 1
------------------------------
previous cwnd size: 1
updated ssthresh value: 10
...SS phase running...
updated cwnd size: 2
Data from 1 - 2 is being sent now... ...


... but wait! Congestion has been detected!


Timeout occurred. Handling Timeout based congestion: cwnd value will become 1.

Retransmitting the lost packet now after handling.


Data sending in RTT number 2
------------------------------
previous cwnd size: 1
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 2
Data from 2 - 3 is being sent now... ...


... but wait! Congestion has been detected!


Handling Triple Dup Ack based congestion: cwnd value will be halved.

Retransmitting the lost packet now after handling.


Data sending in RTT number 3
------------------------------
previous cwnd size: 1
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 2
Data from 3 - 4 is being sent now... ...


... but wait! Congestion has been detected!


Handling Triple Dup Ack based congestion: cwnd value will be halved.

Retransmitting the lost packet now after handling.


Data sending in RTT number 4
------------------------------
previous cwnd size: 1
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 2
Data from 4 - 5 is being sent now... ...


... but wait! Congestion has been detected!


Handling Triple Dup Ack based congestion: cwnd value will be halved.

Retransmitting the lost packet now after handling.


Data sending in RTT number 5
------------------------------
previous cwnd size: 1
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 2
Data from 5 - 6 is being sent now... ...


... but wait! Congestion has been detected!


Timeout occurred. Handling Timeout based congestion: cwnd value will become 1.

Retransmitting the lost packet now after handling.


Data sending in RTT number 6
------------------------------
previous cwnd size: 1
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 2
Data from 6 - 7 is being sent now... ...


Data sending in RTT number 7
------------------------------
previous cwnd size: 2
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 3
Data from 8 - 10 is being sent now... ...


... but wait! Congestion has been detected!


Timeout occurred. Handling Timeout based congestion: cwnd value will become 1.

Retransmitting the lost packet now after handling.


Data sending in RTT number 8
------------------------------
previous cwnd size: 1
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 2
Data from 9 - 10 is being sent now... ...


... but wait! Congestion has been detected!


Timeout occurred. Handling Timeout based congestion: cwnd value will become 1.

Retransmitting the lost packet now after handling.


Data sending in RTT number 9
------------------------------
previous cwnd size: 1
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 2
Data from 10 - 11 is being sent now... ...


Data sending in RTT number 10
------------------------------
previous cwnd size: 2
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 3
Data from 12 - 14 is being sent now... ...


Data sending in RTT number 11
------------------------------
previous cwnd size: 3
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 4
Data from 15 - 18 is being sent now... ...


Data sending in RTT number 12
------------------------------
previous cwnd size: 4
updated ssthresh value: 1
...CA phase running...
updated cwnd size: 5
Data from 19 - 23 is being sent now... ...


... but wait! Congestion has been detected!


Timeout occurred. Handling Timeout based congestion: cwnd value will become 1.

Retransmitting the lost packet now after handling.


Data sending in RTT number 13
------------------------------
previous cwnd size: 1
updated ssthresh value: 2
...SS phase running...
updated cwnd size: 2
Data from 20 - 21 is being sent now... ...



Your data sending is completed. No more data to send.
Congestion Control mechanism concludes.
It took 13 transmission rounds to send the whole data.
BUILD SUCCESSFUL (total time: 15 seconds)
```

6. ANALYSIS AND DISCUSSION

Sliding Window Protocol:

- The sliding window protocol is a classic approach for flow control.
- It ensures that the sender doesn't overwhelm the receiver by limiting the number of unacknowledged packets.
- The implementation uses a simple window size (WINDOW_SIZE), allowing the sender to send a specified number of packets before waiting for acknowledgments.

Sender and Receiver Interaction:

- The sender and receiver interact through a sequence number mechanism.
- The sender sends data with a sequence number, and the receiver acknowledges the data, updating the expected sequence number.
- Continuous Operation:
- Both sender and receiver operate in an infinite loop, simulating continuous communication.