

Green University of Bangladesh

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering Semester: (Spring, Year:2025), B.Sc. in CSE (Day)**

**Lab Report NO : 04**

**Course Title: Computer Networking Lab Course Code: CSE 312 Section: 223-D1**

**Lab Experiment Name:** Implementation of Flow Control and Congestion Control Mechanisms in TCP.

**Student Details**

|  |  |  |
| --- | --- | --- |
| **Name** | | **ID** |
| **1.** | **Avijit Datta Gupta** | **223902048** |

**Lab Date : 05-03-2025**

**Submission Date : 13-05-2025**

**Course Teacher’s Name : Md. Sabbir Hosen Mamun**

**Lab Report Status**

**Marks: ………………………………… Comments:..............................................**

**Signature:.....................**

**Date:..............................**

### **1. Title of the Lab Report Experiment:**

Implementation of Flow Control and Congestion Control Mechanisms in TCP.

## ****Objective:****

To understand and implement the concepts of TCP flow control and congestion control mechanisms using a simulated environment. The lab focuses on how these mechanisms manage data transmission efficiently and prevent network/resource overload.

## ****Tools and Technologies:****

* Programming Language: Python/C (simulated environment)
* Network Simulator or Sockets
* Operating System: Windows/Linux

## ****Part 1: Flow Control Implementation (rwnd)****

### **Description:**

TCP flow control uses a sliding window protocol, where the **receiver advertises a receive window (rwnd)** to the sender, indicating how much data it can receive without being overwhelmed.

### **Implementation Logic:**

* rwnd is initialized based on receiver buffer size.
* Sender maintains variables:
  + LastByteSent
  + LastByteAcked
* Transmission condition:  
  LastByteSent − LastByteAcked ≤ rwnd
* Sender pauses if rwnd = 0 and resumes only upon receiving an updated rwnd > 0.

## ****Part 2: Congestion Control Implementation (cwnd)****

### **Description:**

Congestion control ensures the network is not flooded with too much data. The sender controls the sending rate using a **congestion window (cwnd)**.

### **Mechanism:**

* Starts with cwnd = 1 MSS (slow start).
* cwnd increases exponentially until it hits a **threshold (ssthresh)**.
* After threshold, grows linearly (congestion avoidance).
* On congestion detection:
  + If **timeout**: reset cwnd = 1, ssthresh = cwnd / 2.
  + If **triple duplicate ACKs**: enter **Fast Recovery**.

## ****Experiment Results:****

| **Event** | **rwnd** | **cwnd** | **ssthresh** | **Action Taken** |
| --- | --- | --- | --- | --- |
| Start | 5000 | 1 MSS | 16 MSS | Slow Start |
| Receiver slows | 0 | - | - | Sender pauses |
| Receiver resumes | 3000 | 1 MSS | 16 MSS | Sender resumes |
| Network congested | - | 18 MSS | 9 MSS | cwnd reduced |
| Triple ACK | - | 9 MSS | 9 MSS | Fast Recovery |

## ****Questions and Answers:****

### **1**. **What is the purpose of the receive window (rwnd) in TCP?**

The rwnd ensures the sender does not send more data than the receiver can process, preventing receiver buffer overflow.

### **2.** **What happens when the rwnd value reaches zero and is advertised to the sender?**

The sender stops transmitting further data, effectively pausing until a non-zero rwnd is advertised again.

### **3.** **How does the sender resume transmission after receiving an rwnd value of zero?**

The sender resumes only after receiving a TCP segment (window update) from the receiver with a non-zero rwnd.

### **4.** **What is the role of the sliding window protocol in TCP flow control?**

It allows for continuous data transmission while maintaining control over how much unacknowledged data can be in transit, based on rwnd.

### **5.** **Explain the equation: LastByteSent − LastByteAcked ≤ rwnd.**

This ensures the sender does not send more data than the receiver's advertised buffer space, maintaining flow control.

### **6.** **What is Silly Window Syndrome (SWS), and how can it be avoided?**

SWS is a problem where small segments are exchanged, reducing efficiency. Avoided using:

* Nagle's algorithm (sender side)
* Clark’s solution (receiver side)

### **7.** **What is the role of the congestion window (cwnd) in TCP?**

cwnd limits the amount of data the sender can send into the network before receiving ACKs, controlling congestion.

### **8.** **How does TCP differentiate between flow control and congestion control?**

* **Flow control** is **receiver-side** (based on rwnd)
* **Congestion control** is **network-side** (based on cwnd)

### **9.** **What is the significance of the slow start threshold (ssthresh)?**

It defines the boundary between exponential growth (slow start) and linear growth (congestion avoidance) of cwnd.

### **10.** **What are the three phases of TCP congestion control, and how does cwnd behave in each?**

1. **Slow Start**: cwnd doubles each RTT.
2. **Congestion Avoidance**: cwnd increases linearly.
3. **Fast Recovery**: cwnd is halved and grows linearly after triple duplicate ACKs.

### **11.** **Compare the behavior of TCP Tahoe and TCP Reno upon detecting congestion.**

* **Tahoe**: On congestion, resets cwnd = 1 regardless of cause.
* **Reno**: On triple duplicate ACKs, uses **Fast Recovery**; only timeout triggers reset to 1.

### **12.** **How does TCP detect network congestion (i.e., timeout vs. triple duplicate ACKs)?**

* **Timeout**: Implies packet loss, likely due to severe congestion.
* **Triple Duplicate ACKs**: Indicates a segment is lost, but others are arriving, so less severe.

### **13.** **What is AIMD (Additive Increase, Multiplicative Decrease) and how does it relate to TCP congestion control?**

AIMD strategy:

* Additively increases cwnd in congestion avoidance.
* Multiplicatively decreases cwnd on detecting congestion.  
  Ensures fair and stable bandwidth sharing.

### **14.** **What is the purpose of the Fast Recovery phase in TCP Reno, and why is it not used in Tahoe?**

**Fast Recovery** allows Reno to avoid returning to slow start after triple ACKs, improving throughput. Tahoe lacks this mechanism and always reverts to slow start.

### **15**. **What would happen if TCP ignored congestion control and only used flow control?**

Without congestion control, multiple flows could overwhelm the network, leading to **packet loss**, **delays**, and **unfair bandwidth usage**, degrading overall network performance.

## ****Conclusion:****

This lab demonstrates how **flow control** and **congestion control** together ensure **reliable and efficient** data transmission in TCP. Flow control prevents receiver overload, while congestion control avoids network collapse. Their interplay, guided by window sizes (rwnd and cwnd), dynamic algorithms like AIMD, and protocols like Fast Recovery, makes TCP resilient and scalable.