CS 331: Computer Networks Assignment #2 SubmissionDate: Before 08-03-2025, 11:59pm

---

Assignment Guidelines:

Teamwork: The assignment must be completed in pairs of two students. Only one team member (Member 1) must submit the assignment on behalf of the group.

Protocol Assignment Scheme:

Each team will work on **three congestion control protocols** based on their **team number** using the following formula:

1. **Protocol 1**→ `(team_number % 10)`
2. **Protocol 2**→ `((team_number + 3) % 10)`
3. **Protocol 3**→ `((team_number + 6) % 10)`

List of available protocols:

Protocol No.	Protocol
0	Reno
1	CUBIC
2	BBR
3	BIC
4	Vegas
5	Westwood
6	HighSpeed
7	H-TCP
8	Scalable
9	Yeah

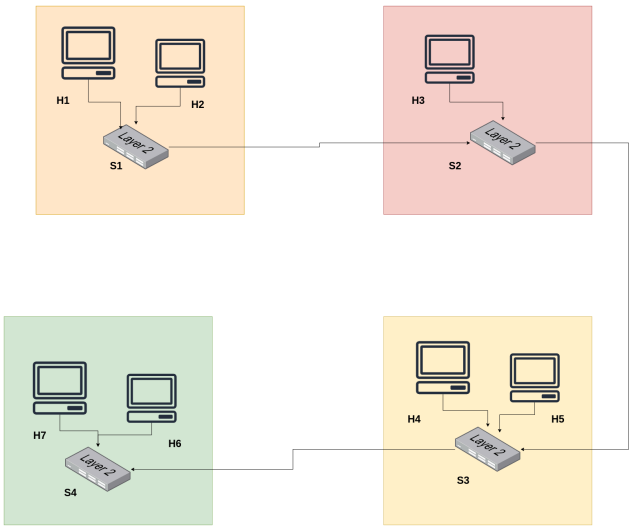
Submission File Naming: Report should be named as:TeamID-RollNumber1_RollNumber2.pdf For example, For TeamID 1, and roll numbers 23210122 and 23210023, the file should be named: 1-23210122_23210023.pdf and the submission must be made by student with roll number 23210122 only.

GitHub Repository: Include a link to your GitHub repository containing all the programs and scripts used for the assignment. A common repository may be provided to you at later stages to upload all your project files.

Note: Wherever possible, try to make a relevant plot and tables that help compare and contrast the results in a lucid manner.

Task-1: Comparison of congestion control protocols [50 Points]

Create a mininet topology as shown in the diagram below. H1 to H7 are seven hosts connected to switches S1,S2,S3,S4. Run a TCP Server at H7. H1- H6 are the TCP clients that connect with the server (H7) and send data packets.



Note: Here is a quick walkthrough of mininet (mininet) and to create mininet topology refer to these examples (example link).

To analyze TCP congestion control, you need to implement an **iperf3 client-server setup** that generates **TCP traffic** for analysis. The traffic should be generated using the following command:
iperf3 -c <server_ip> -p <server_port> -b 10M -P 10 -t 150 -C <congestion_control_scheme>
For detailed guidance on setting up **iperf3 client-server** communication and configuring congestion control schemes, refer to the **official iPerf documentation**: https://iperf.fr/

Implementation of the mininet topology and TCP client-server programs. The TCP client-server program should run with the required configuration. The program should take an argument called option which helps run the below experiments. Example, for --option=a will run the part a of the program. Other configurable parameters like congestion control scheme, link loss should also be applied as per the specified option.

a. Run the client on H1 and the server on H7. Measure the below parameters and summarize the observations for the three congestion schemes as applicable.
   1. Throughput over time (with valid Wireshark I/O graphs)
   2. Goodput
   3. Packet loss rate
   4. Maximum window size achieved (with valid Wireshark I/O graphs).

b. Run the clients on H1, H3 and H4 in a staggered manner(H1 starts at T=0s and runs for 150s, H3 at T=15s and runs for T=120s, H4 at T=30s and runs for 90s) and the server on H7. Measure the parameters listed in part (a) and explain the observations, for the 3 congestion schemes for all the three flows.

c. Configure the links with the following bandwidths:
   I. Link S1-S2: 100Mbps
   II. Links S2-S3: 50Mbps
   III. Links S3-S4: 100Mbps

Measure the performance parameters listed in part (a) and explain the observations in the following conditions:

1. Link S2-S4 is active with client on H3 and server on H7.
2. Link S1-S4 is active with:
   a. Running client on H1 and H2 and server on H7
   b. Running client on H1 and H3 and server on H7
   c. Running client on H1, H3 and H4 and server on H7

d. Configure the link loss parameter of the link S2-S3 to 1% and 5% and repeat part (c).

Task-2 : Implementation and mitigation of SYN flood attack [100 Points]

Perform this task in a virtual/isolated environment. You can implement any client-server program that generates tcp packets. Preferably use two VM's or two host machines.

A. Implementation of SYN flood attack

Modify the receiver's (server) Linux kernel in any way you want to implement the TCP/IP SYN flood attack. Focus on tuning the following Linux kernel parameters to optimize the attack:

- **net.ipv4.tcp_max_syn_backlog** – Increases the maximum number of pending SYN requests in the backlog.
- **net.ipv4.tcp_syncookies** – Disables SYN cookies (set to 0) to prevent the system from mitigating the attack.
- **net.ipv4.tcp_synack_retries** – Reduces the number of SYN-ACK retries, making it easier to exhaust resources.

Using a stopwatch perform the following scenario:

1. Start capturing the packets at client side using tcpdump
2. Start legitimate traffic
3. After 20 seconds start the attack
4. After 100 seconds stop the attack
5. After 20 seconds stop the legitimate traffic
6. Stop the tcpdump on the client and save the file.

(Note: **Legitimate Traffic**: Normal network traffic generated by any client program, **Attack Traffic(SYN flood)**: Excessive **TCP SYN requests** aimed at exhausting server resources.)

Process the output PCAP file and calculate the connection duration for each TCP connection recorded in the file. The connection duration is defined as the time difference between the first **SYN** packet and the **ACK** following a **FIN-ACK**, or between the first **SYN** and the first **RESET** packet for that connection. If a connection does not end with an **ACK** after a **FIN-ACK**, assign it a default duration of **100 seconds**. To uniquely identify a TCP connection, use the tuple: **(Source IP, Destination IP, Source Port, Destination Port)**. Plot the connection duration vs. connection start time and mark the start and end of the attack using vertical lines or arrows. You can use Wireshark to verify the successful execution of the attack. Attach screenshots displaying the packet status. (Hint: Use **tshark** to extract details of unique connections.)

B. Apply SYN flood attack mitigation

Explore and implement any of the SYN flood mitigation strategies. After implementing the mitigation technique, perform the same experiment and plot a graph of **connection duration vs. connection start time** and compare it with the graph from the initial experiment (without mitigation). Use Wireshark to show the successful implementation, attach the screenshots showing the packet status.

Task-3: Analyze the effect of Nagle's algorithm on TCP/IP performance. [50 Points]

You will transmit a **4 KB file** over a TCP connection for a duration of **~2 minutes** with a transfer rate of **40 bytes/second**. Perform the following **four combinations** by enabling/disabling **Nagle's Algorithm** and **Delayed-ACK** on both the **client** and **server sides**:

1. Nagle's Algorithm enabled, Delayed-ACK enabled.
2. Nagle's Algorithm enabled, Delayed-ACK disabled.
3. Nagle's Algorithm disabled, Delayed-ACK enabled.
4. Nagle's Algorithm disabled, Delayed-ACK disabled.

For each configuration measure the following performance metrics and compare. Explain your observations:

1. Throughput
2. Goodput
3. Packet loss rate
4. Maximum packet size achieved.