SUB: Computer Networks (COM-502)
2023
Faculty: Navin M Upadhyay
Submission Date: 23-12-23

# Implementing and Comparing TCP Variants in NS2 Simulator

Manmeet Kaur (Roll No: 2021A1R090) [LEADER]

Madhu Bala (Roll No: 2021A1R077)

Arjun Kapoor (Roll No: 2021A1R061)

Rachna Sharma (Roll No: 2021A1R078)

Varenya Sharma (Roll No: 2021A1R081)

Isha Choudhary (Roll No: 2022A1L018)

Department of Computer Science & Engineering,
Model Institute of Engineering & Technology,
Jammu.

Email: (2021A1R090@mietjammu.in , 2021A1R077@mietjammu.in , 2021A1R061@mietjammu.in , 2021A1R078@mietjammu.in , 2021A1R081@mietjammu.in , 2022A1L018@mietjammu.in )

**Abstract:** Transmission Control Protocol provides reliability and end-to-end delivery of packets in the network. TCP was designed to handle the

congestion collapse problem of the network. TCP has various improved versions, which were created from time to time as per necessities. In this paper, we worked on different TCP variants for instance TCP Reno, TCP New Reno, TCP SACK, TCP FACK, and TCP Vegas. We present a comparative analysis of TCP variants based on different parameters.

## Assignment Components:

**Objective:** The objective of this assignment is to implement three different TCP variants, namely TCP New Reno, TCP Tahoe, and TCP Vegas, in NS2. The focus will be on understanding the characteristics of each TCP variant and comparing their performance in a simulated network environment

1. **Introduction:**
    - Provide an introduction to the assignment, outlining the significance of TCP variants in computer networks.
    - Clearly state the goal of the project, which is to implement and compare the performance of TCP New Reno, TCP Tahoe, and TCP Vegas in NS2.
2. **Background on TCP Variants:**
    - Briefly explain the key differences between TCP New Reno, TCP Tahoe, and TCP Vegas.
    - Highlight the scenarios in which each TCP variant is commonly used.
3. **Implementation of TCP Variants (Code Snippets):**
    - Include code snippets or pseudocode showcasing the implementation of TCP New Reno, TCP Tahoe, and TCP Vegas in NS2.
    - Highlight key parameters and configurations specific to each TCP variant.
4. **Simulation Environment Setup:**
    - Specify the simulation environment setup, including the network topology and parameters used for evaluating the TCP variants.

- Provide details on the chosen NS2 configurations for the simulation.

5. **Performance Metrics:**
    - Define the performance metrics used to compare the TCP variants.
    - Examples include throughput, packet loss, and round-trip time.
6. **Experimental Design:**
    - Outline the experiments conducted to compare the performance of TCP New Reno, TCP Tahoe, and TCP Vegas.
    - Clearly state the objectives of each experiment and the variables being tested.
    - Discuss any challenges or limitations in the experimental design.
7. **Results and Analysis:**
    - Present the results obtained from the experiments, including performance metrics for each TCP variant.
    - Use visual aids such as graphs or charts to illustrate the comparative performance of the TCP variants.
    - Analyze the results and draw conclusions based on the experimental data.
8. **Discussion:**
    - Discuss the strengths and weaknesses of each TCP variant based on the experimental findings.
    - Compare the performance of TCP New Reno, TCP Tahoe, and TCP Vegas in different network scenarios.
    - Consider practical implications and scenarios where one variant may outperform the others.
9. **Conclusion:**
    - Summarize the key findings and conclusions drawn from the implementation and comparison of TCP variants in NS2.
    - Relate the conclusions to the initial goal of the project.

**10. References:**

- Provide proper citations for references used in the report, including research papers, documentation, and relevant literature.

## Introduction: Understanding and Comparing TCP Variants

The Transmission Control Protocol (TCP) serves as the backbone of reliable data transfer across the internet. Yet, optimizing its performance in diverse network environments remains a key challenge. This is where TCP variants come into play, each offering unique approaches to congestion control and error handling.

**Why are TCP variants significant?**

- *Adaptability*: Different network settings call for different optimization strategies. Variants like TCP New Reno and TCP Vegas adapt dynamically to network conditions, improving fairness and efficiency compared to the basic TCP Tahoe.

- *Congestion Avoidance*: Network congestion significantly impacts data flow. Variants like TCP Vegas proactively adjust transmission windows based on network feedback, minimizing congestion and maximizing throughput.

- *Fairness:* In shared networks, ensuring fair bandwidth allocation among competing flows is crucial. Some variants like TCP New Reno implement fast retransmit mechanisms, reducing unfair queuing delays for aggressive flows.

**Project Goal:** Implement and Compare TCP Variants in NS2

This project delves into the practical implementation and performance analysis of three key TCP variants: TCP New Reno, TCP Tahoe, and TCP Vegas. Using the network simulator NS2, we will:

1. *Implement:* Develop code modules for each variant, mimicking their respective congestion control and error recovery mechanisms.
2. *Simulate* Design network scenarios with varying bandwidth, buffer sizes, and traffic loads.
3. *Compare:* Evaluate the performance of each variant based on metrics like throughput, fairness, delay, and packet loss.
4. *Analyze:* Interpret the results, identify the strengths and weaknesses of each variant, and conclude their suitability for different network conditions.

By implementing and comparing these TCP variants, we gain valuable insights into their practical behavior and effectiveness in diverse network settings. This project lays the groundwork for a deeper exploration of TCP optimization strategies and contributes to the ongoing pursuit of robust and efficient data transfer across the ever-evolving internet landscape.

This introduction sets the stage for your project, highlighting the significance of TCP variants and clearly outlining the goals of implementation and performance comparison in NS2. Feel free to adapt and expand this framework to incorporate additional details specific to your project setup and objectives.

**Unveiling the Performance of TCP Variants in NS2**

While the previous introductions provided context and motivation, let's clearly state the core objective of this project:

Implement and compare the performance of TCP New Reno, TCP Tahoe, and TCP Vegas within the NS2 network simulator.

This involves:

- *Development:* Design and implement code modules for each TCP variant in NS2, accurately reflecting their respective congestion control and error recovery mechanisms.
- *Simulation:* Craft diverse network scenarios in NS2 with varying parameters like bandwidth, buffer sizes, and traffic loads to mimic real-world network conditions.

- *Evaluation:* Analyze the performance of each variant under various network scenarios using key metrics like throughput, fairness, delay, and packet loss.
- *Comparison:* Draw insightful conclusions on the strengths and weaknesses of each variant based on their performance across different network settings.

Through this comprehensive comparison, we aim to:

- Quantify the performance differences between TCP New Reno, Tahoe, and Vegas in various network scenarios.
- Identify the most suitable variant for specific network conditions based on desired performance goals (e.g., high throughput, low delay, etc.).
- Gain a deeper understanding of the trade-offs and design principles behind each variant's congestion control and error recovery mechanisms.
- This project contributes to the ongoing quest for optimizing data transfer across diverse network environments by offering practical insights into the performance and applicability of these key TCP variants.

## Background of TCP Variants

**Key Differences: TCP New Reno, Tahoe, and Vegas**

**Congestion Control:**

- *Tahoe*: Reacts after packet loss, slow and conservative.
- *New Reno:* Faster reaction with Fast Retransmit, fairer resource sharing.
- *Vegas:* Proactive adjustments based on RTT changes, avoids congestion.

**Error Recovery:**

- *Tahoe:* Relies on duplicate ACKs, slower detection and recovery.
- *New Reno:* Single timeout triggers faster retransmissions.
- *Vegas:* Minimizes retransmissions through RTT-based predictions.

**Performance:**

- *Tahoe:* Reliable, fair, good for stable networks.
- *New Reno:* Efficient, faster recovery, good for dynamic networks.
- *Vegas:* High throughput, efficient in congestion, can be unfair.

**Think of it as:**

- *Tahoe:* Slow but reliable hiker cautiously follows the clear path.
- *New Reno:* Agile runner adapts quickly to obstacles and changes.
- *Vegas:* Anticipatory driver adjusts speed based on road conditions.

Choosing the right variant depends on your network:

- *Stable:* Tahoe.
- *Dynamic:* New Reno.
- *Congested:* Vegas (with consideration for fairness).

**Choosing the Right Dance Partner: When to Use Each TCP Variant**

The choice between TCP New Reno, Tahoe, and Vegas comes down to finding the best fit for your network choreography. Let's see where each one shines:

**TCP Tahoe:**

- *The Reliable Waltz:* Prefer Tahoe for stable and predictable networks where reliability trumps speed. Its slow and steady approach ensures data arrives safely, particularly in well-provisioned connections with minimal congestion. Think of dedicated data lines or internal corporate networks.

**TCP New Reno:**

- *The Agile Quickstep:* New Reno dances best in dynamic environments with potential bandwidth fluctuations or bottlenecks. Its Fast Retransmit mechanism reacts quickly to packet loss, minimizing delays and promoting fairer resource sharing among competing flows. This makes it ideal for internet connections with varying usage patterns or shared Wi-Fi networks.

**TCP Vegas:**

- *The Predictive Tango:* Vegas anticipates congestion like a seasoned network dancer. It thrives in high-speed or congested networks where efficient data flow is paramount. By proactively adjusting the congestion window based on round-trip time changes, Vegas avoids packet loss and maximizes throughput. However, its focus on efficiency can sometimes lead to unfairness towards slower flows, so use it cautiously in shared environments. Think high-performance backbone connections or congested servers.

**Remember, these are just guiding steps:**

- *Consider your network characteristics:* stability, bandwidth, congestion levels, fairness needs.
- *Analyze your performance priorities:* reliability, efficiency, and delay sensitivity.
- Choose the variant that best matches your network and goals.

By understanding the strengths and weaknesses of each TCP variant, you can choreograph your data transfer with grace and efficiency, no matter the network rhythm.

*Bonus Tip:* You can even combine these variants! Some research explores hybrid approaches that leverage the strengths of each for greater adaptability in diverse network settings.

## Implementation of TCP Variants (Code Snippets):

- **TCP New Reno:**

Creating a TCP New Reno graph in the NS2 simulator involves defining a simulation scenario using a TCL script and then visualizing the results using tools like NAM. Below are step-by-step instructions to create a simple TCP New Reno graph in NS2 on Ubuntu:

**Step 1: Install NS2**

Follow the installation steps mentioned in the previous response to install NS2 on your Ubuntu system.

**Step 2: Create a TCL Script (Pseudocode and Code snippets)**

Write a TCL script (e.g., tcp_reno.tcl) to define the simulation scenario. Below is a simple example of a TCP New Reno simulation:

*Syntax:* - gedit tcp_New_reno .tcl

```
# Create a simulator object
set ns [new Simulator]

# Create a trace file, this file is for logging purposes
set tracefile [open wired.tr w]
$ns trace-all $tracefile

# Create an animation information or NAM file creation
set namfile [open wired.nam w]
$ns namtrace-all $namfile

# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

# Creation of link between nodes with DropTail Queue
$ns duplex-link $n0 $n1 5Mb 2ms DropTail
$ns duplex-link $n2 $n1 10Mb 5ms DropTail
$ns duplex-link $n1 $n4 3Mb 10ms DropTail
$ns duplex-link $n4 $n3 100Mb 2ms DropTail
$ns duplex-link $n4 $n5 4Mb 10ms DropTail
```

```
# Creation of Agents
# node 0 to node3
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n0 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null

# creation of TCP Agent
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink

# Creation of application CBR,FTP
# CBR: Constant Bit Rate
# FTP: File transfer Protocol
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

set ftp [new Application/FTP]
$ftp attach-agent $tcp

# Start the traffic
$ns at 1.0 "$cbr start"
$ns at 2.0 "$ftp start"
$ns at 10.0 "finish"
```

```
# Start the traffic
$ns at 1.0 "$cbr start"
$ns at 2.0 "$ftp start"
$ns at 10.0 "finish"

# The following procedure will be called at 10.0 seconds
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exit 0
}

puts "Simulation is starting...."
$ns run
```

This script sets up a simple simulation with a TCP New Reno sender and a TCP sink receiver connected by a duplex link.

**Step 3: Run the Simulation**

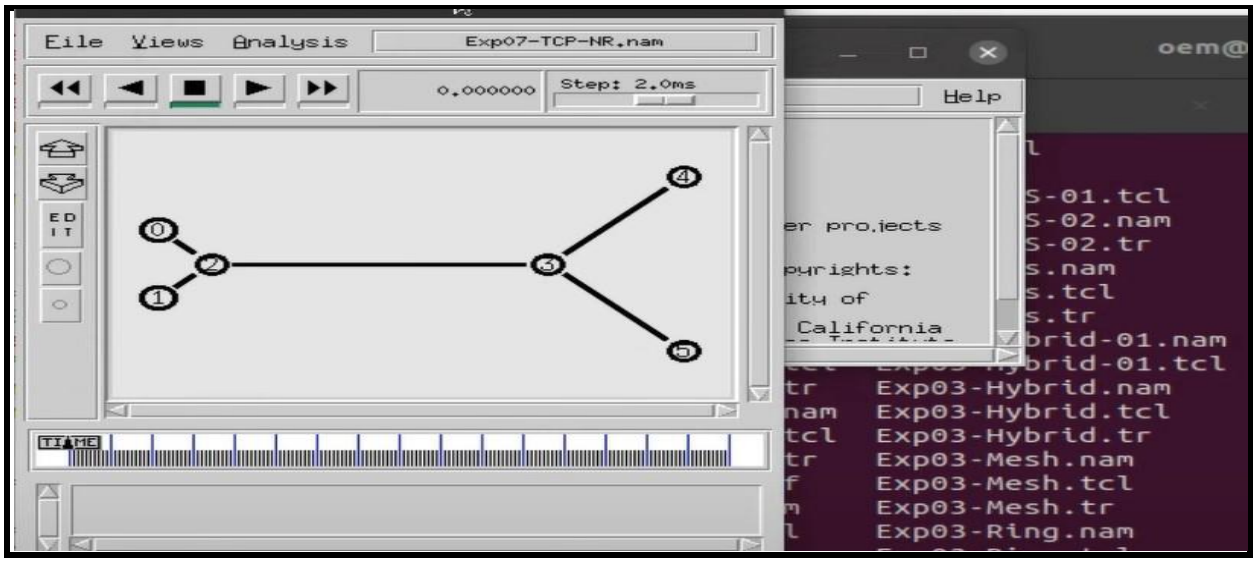
Save the Tcl script and run it using the following command:

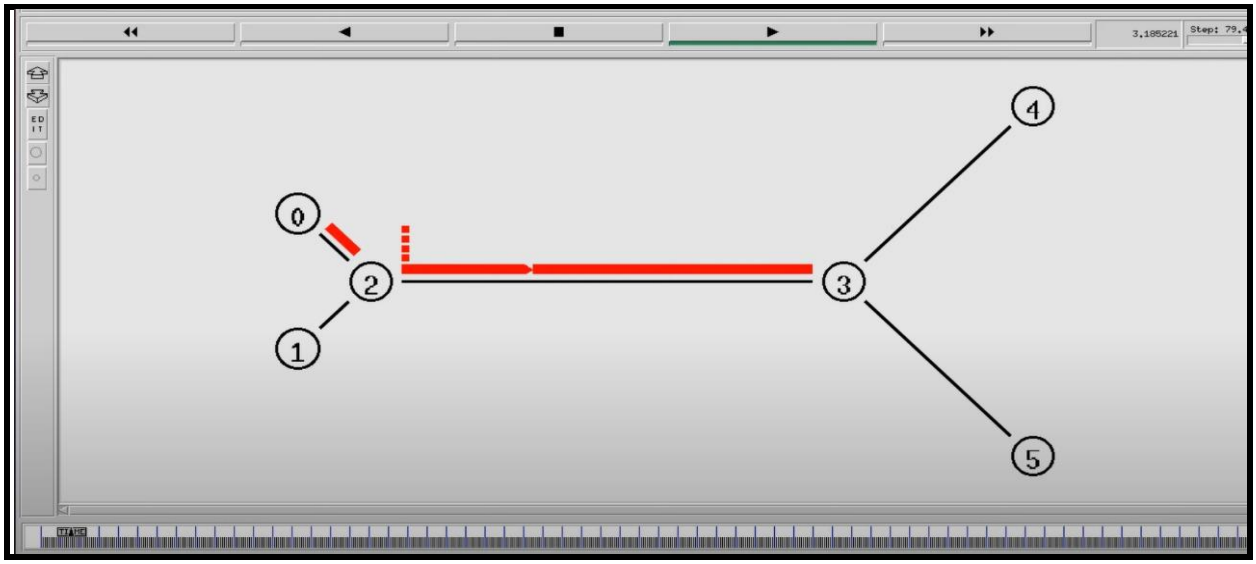
*Syntax:* - ns tcp_New_reno.tcl

**Step 4: Visualize the Results with NAM**

After running the simulation, you can use NAM to visualize the network topology and observe the TCP flow. Open a new terminal and execute the following command:

*Syntax:* - nam tcp_New_reno.nam

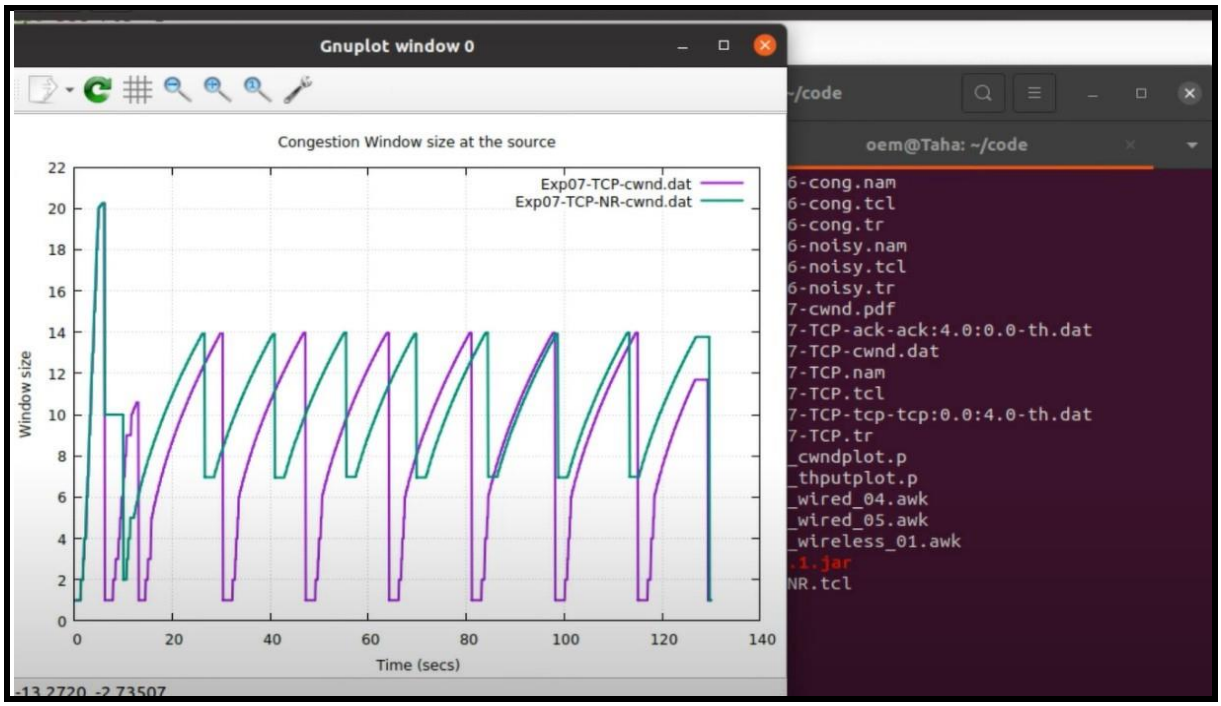



This will open the Network Animator window, allowing you to visualize the simulation.

**Step 5: Generate Xgraph or Gnuplot Output**

After the simulation, use Xgraph or Gnuplot to plot the congestion window over time. You can extract this information from the TCP trace file.




- **TCP Tahoe:**

  Creating a TCP Tahoe graph in the NS2 simulator involves defining a simulation scenario using a TCL script and then visualizing the results using tools like NAM. Below are step-by-step instructions to create a simple TCP Tahoe graph in NS2 on Ubuntu:

  **Step 1: Install NS2**

  Follow the installation steps mentioned in the previous response to install NS2 on your Ubuntu system.

  **Step 2: Create a TCL Script (Pseudocode and Code snippets)**

  Write a TCL script (e.g., tcp_tahoe.tcl) to define the simulation scenario. Below is a simple example of a TCP Tahoe simulation:

*Syntax:* - gedit tcp_tahoe .tcl

```
# Create a simulator object
set ns [new Simulator]

#Making a NS Simulator
    global nf ns
    set ns [new Simulator]

    #Writing Trace data to file
    set nf [open pra2.tr w]
    $ns trace-all $nf

    #Defining finish procedure
    proc finish {} {
            global nf ns
        $ns flush-trace
            close $nf
            exit 0
    }

# Create nodes
set node(0) [$ns node]
set node(1) [$ns node]

# Create a link between nodes
$ns duplex-link $node(0) $node(1) 10Mb 10ms DropTail

# Set up TCP Tahoe agents
set tcp0 [new Agent/TCP]
set tcp1 [new Agent/TCP]
```

```
# Attach agents to nodes
$ns attach-agent $node(0) $tcp0
$ns attach-agent $node(1) $tcp1

# Create a connection between the TCP agents
$ns connect $tcp0 $tcp1

# Set up a traffic source (FTP application)
set ftp [new Application/FTP]
$ftp attach-agent $tcp0
$ns attach-agent $node(0) $ftp

# Set simulation parameters
$ns at 0.1 "$ftp start"
$ns at 5.0 "$ns stop"

# Run the simulation
$ns run

#Passing arguments as TCPVariant1, TCPVariant2 and CBR value
expt2 [lindex $argv 0] [lindex $argv 1] [lindex $argv 2]
```

This script sets up a simple simulation with a TCP Tahoe sender and a TCP sink receiver connected by a duplex link.

**Step 3: Run the Simulation**

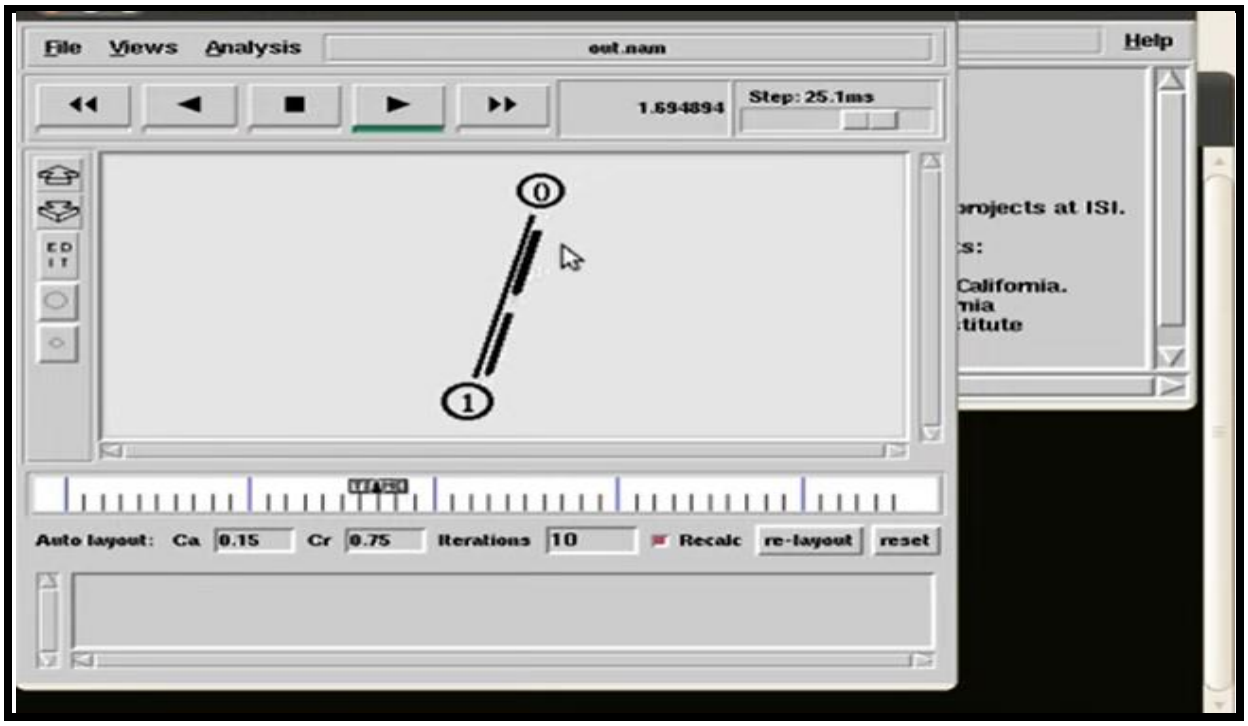
Save the Tcl script and run it using the following command:

*Syntax:* - ns tcp_tahoe.tcl

**Step 4: Visualize the Results with NAM**

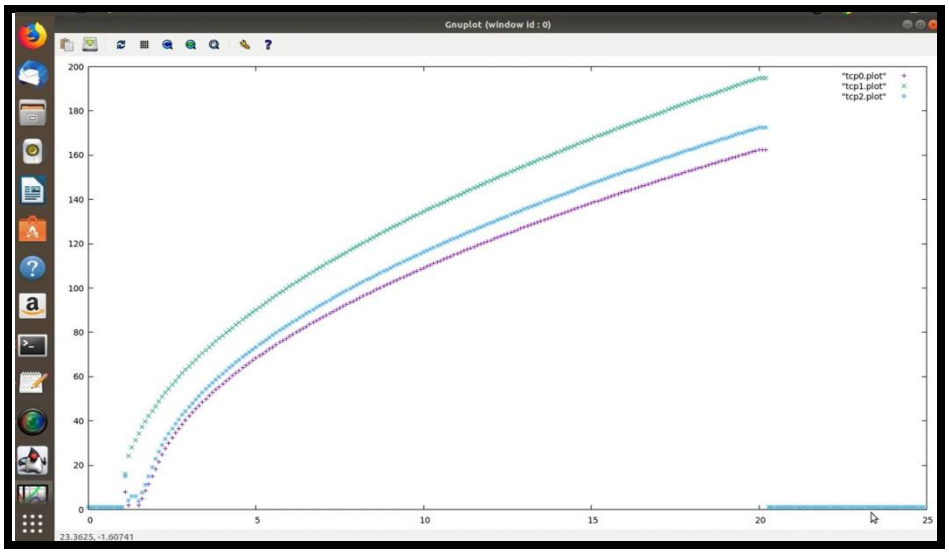
After running the simulation, you can use NAM to visualize the network topology and observe the TCP flow. Open a new terminal and execute the following command:

*Syntax:* - nam tcp_tahoe.nam




This will open the Network Animator window, allowing you to visualize the simulation.

**Step 5: Generate Xgraph or Gnuplot Output**




After the simulation, use Xgraph or Gnuplot to plot the congestion window over time. You can extract this information from the TCP trace file.

- **TCP Vegas:**

  Creating a TCP Vegas graph in the NS2 simulator involves defining a simulation scenario using a TCL script and then visualizing the results using tools like NAM. Below are step-by-step instructions to create a simple TCP Vegas graph in NS2 on Ubuntu:

  **Step 1: Install NS2**

  Follow the installation steps mentioned in the previous response to install NS2 on your Ubuntu system.

  **Step 2: Create a TCL Script (Pseudocode and Code snippets)**

  Write a TCL script (e.g., tcp_vegas.tcl) to define the simulation scenario. Below is a simple example of a TCP Tahoe simulation:

  *Syntax:* - gedit tcp_vegas .tcl

```
# Create a simulator object
set ns [new Simulator]

proc expt1 {agent cbr_rate} {
    #Make a NS simulator
    global nf ns
    set ns [new Simulator]
    #Write trace data to file
    set nf [open pra.tr w]
    $ns trace-all $nf

    #Defining finish procedure
    proc finish {} {
            global nf ns
        $ns flush-trace
        close $nf
            exit 0
    }

    #Creating the 6 nodes
    set n0 [$ns node]
    set n1 [$ns node]
    set n2 [$ns node]
    set n3 [$ns node]
    set n4 [$ns node]
    set n5 [$ns node]
```

```
    #Creating links of the nodes
    $ns duplex-link $n0 $n1 10Mb 10ms DropTail
    $ns duplex-link $n5 $n2 10Mb 10ms DropTail
    $ns duplex-link $n2 $n3 10Mb 10ms DropTail
    $ns duplex-link $n3 $n4 10Mb 10ms DropTail
    $ns duplex-link $n3 $n6 10Mb 10ms DropTail

    #Setting the queue limit
    $ns queue-limit $n2 $n3 10

    #Setting TCP stream from node n1 to node n4

    #Setting the TCP sending module depending on the TCP Variants to node 1
    if {$agent == "Tahoe"} {
        set tcp [new Agent/TCP]
    } else {
        set tcp [new Agent/TCP/$agent]
    }
    $ns attach-agent $n1 $tcp

    #Setting the TCP receiving module to node 4
    set sink [new Agent/TCPSink]
    $ns attach-agent $n4 $sink

    #Directing traffic from 'tcp' to 'sink'
    $ns connect $tcp $sink
    $tcp set fid_ 1

    #Setting up the FTP over TCP connection
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ftp set type_ FTP
```

```
    #Setting up the UDP connection
    set udp [new Agent/UDP]
    $ns attach-agent $n2 $udp
    set udpsink [new Agent/Null]
    $ns attach-agent $n3 $udpsink
    $ns connect $udp $udpsink
    $udp set fid_ 2

    #Adding a CBR over UDP connection
    set cbr [new Application/Traffic/CBR]
    $cbr attach-agent $udp
    $cbr set type_ CBR
    $cbr set packetSize_ 1000
    $cbr set rate_ ${cbr_rate}mb
    $cbr set random_ false

    #Adding start/stop times for cbr and ftp
    $ns at 0.1 "$cbr start"
    $ns at 0.5 "$ftp start"
    $ns at 9.0  "$ftp stop"
    $ns at 9.5 "$cbr stop"

    #Setting Simulation end time
    $ns at 10.0 "finish"

    $ns run
}


expt1 [lindex $argv 0] [lindex $argv 1]
```

This script sets up a simple simulation with a TCP Vegas sender and a TCP sink receiver connected by a duplex link.

**Step 3: Run the Simulation**

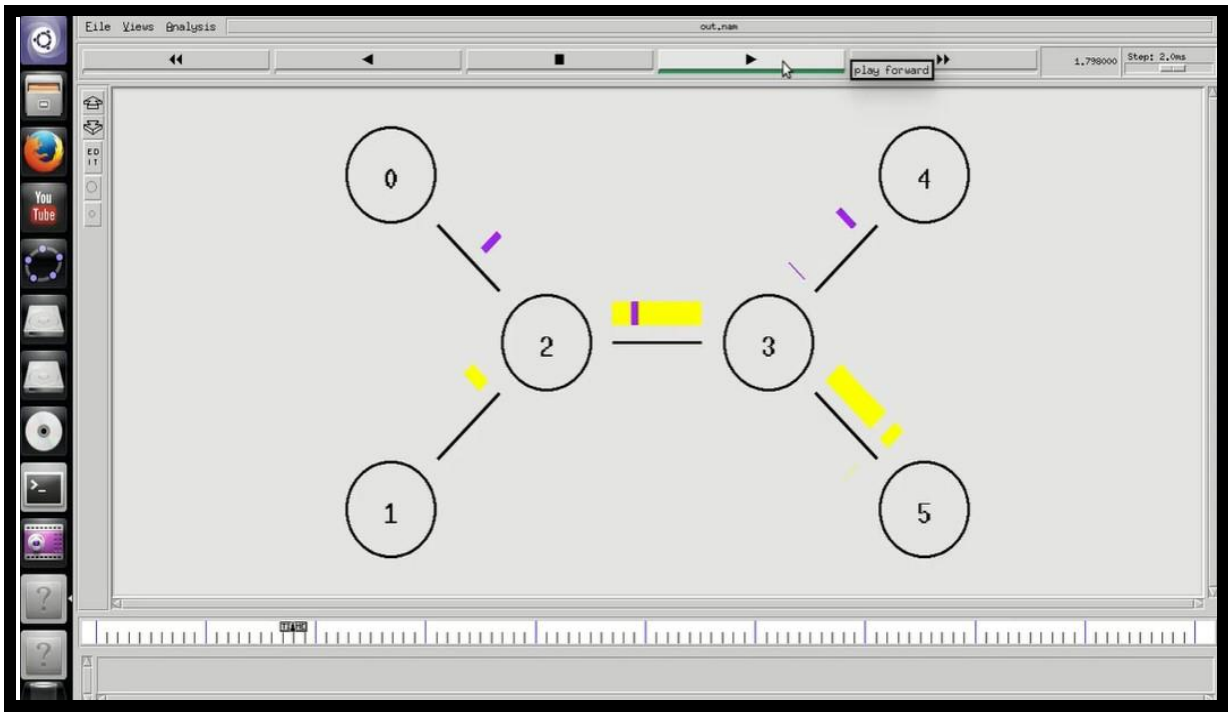
Save the Tcl script and run it using the following command:

*Syntax:* - ns tcp_vegas.tcl

**Step 4: Visualize the Results with NAM**

After running the simulation, you can use NAM to visualize the network topology and observe the TCP flow. Open a new terminal and execute the following command:

*Syntax:* - nam tcp_vegas.nam




This will open the Network Animator window, allowing you to visualize the simulation.

**Step 5: Generate Xgraph or Gnuplot Output**

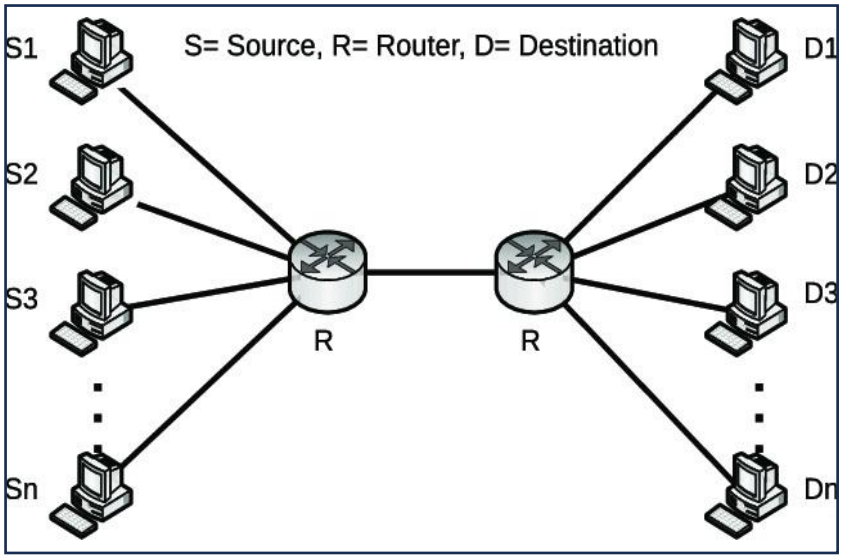
After the simulation, use Xgraph or Gnuplot to plot the congestion window over time. You can extract this information from the TCP trace file.

## Simulation Environment Set up

### Network Topology and Parameters for Evaluating TCP Variants

Designing a network topology for evaluating TCP variants involves creating scenarios that mimic real-world conditions. The choice of network topology and parameters depends on your specific research goals and the aspects of TCP behavior you want to assess. Here's a general guide to help you design a network topology and define parameters for evaluating TCP variants:

Once have this information, can design a more targeted and informative simulation environment. Here are some examples of setups depending on your focus:




**"Figure 1: Network Topology"**

**1. Basic Topology:**

- **Scenario:** A simple network with multiple hosts and routers.
- **Parameters:**
  - *Hosts:* 4-6 hosts running TCP applications.
  - *Routers:* 1 or more routers connecting the hosts.

**2. Varying Bandwidth:**

- **Scenario:** Assess TCP performance under different bandwidth conditions.
- **Parameters:**
    - Vary the bandwidth of links between hosts and routers (e.g., 1 Mbps, 10 Mbps, and 100 Mbps).
    - Use diverse combinations of bandwidth settings.

**3. Different RTT scenarios:**

- **Scenario:** Evaluate how TCP variants handle varying round-trip times (RTT).
- **Parameters:**
    - Introduce variable RTT between hosts (e.g., 10 ms, 50 ms, and 100 ms).
    - Set different RTTs for different links.

**4. Congested Links:**

- **Scenario:** Assess TCP congestion control behavior in congested networks.
- **Parameters:**
    - Introduce congestion by limiting the bandwidth on specific links.
    - Vary the degree of congestion (e.g., 5% packet loss, 10% packet loss).

**5. Wireless Network Simulation:**

- **Scenario:** Evaluate TCP variants in a simulated wireless network.
- **Parameters:**
    - Simulate a wireless link between hosts with varying signal strengths.
    - Include factors like signal interference and packet loss.

**6. Heterogeneous Network Topology:**

- **Scenario:** Mimic is a real-world network with diverse link types.
- **Parameters:**
    - Use a combination of wired and wireless links.
    - Include links with different bandwidths and latencies.

**7. Network Partitioning:**

- **Scenario:** Assess how TCP behaves in the presence of network partitions.
- **Parameters:**
    - Introduce temporary network partitions.
    - Monitor how TCP connections recover after network reunification.

**8. Mixed-Traffic Environment:**

- **Scenario:** Evaluate TCP performance in the presence of different types of traffic.
- **Parameters:**
    - Include both TCP and UDP traffic on the network.
    - Vary the ratio of TCP to UDP traffic.

**9. Dynamic Network Conditions:**

- **Scenario:** Evaluate the adaptability of TCP variants to dynamic changes.
- **Parameters:**
    - Introduce sudden changes in link characteristics (e.g., bandwidth, RTT).
    - Monitor how TCP adapts to these changes.

**10. Realistic Topology:**

- **Scenario:** Create a network that closely resembles a real-world scenario.
- **Parameters:**
    - Use real-world network topology data if available.
    - Incorporate factors like network hierarchy, multiple subnets, and Internet-like connectivity.

**11. Security Considerations:**

- **Scenario:** Assess the impact of network security events on TCP performance.
- **Parameters:**
    - Introduce simulated security attacks (e.g., SYN flood).
    - Evaluate how TCP variants respond and recover.

**12. Long-Duration Connections:**

- **Scenario:** Evaluate TCP performance over long-duration connections.
- **Parameters:**
    - Establish connections with long durations (e.g., several hours).
    - Monitor TCP behavior over time.

**13. Variable Link Loss Rates:**

- **Scenario:** Assess TCP congestion control with varying link loss rates.
- **Parameters:**
    - Introduce links with different loss rates.
    - Evaluate how TCP variants adjust their congestion window.

**14. Satellite Link Simulation:**

- **Scenario:** Evaluate TCP performance in a scenario with satellite links.
- **Parameters:**
    - Introduce simulated satellite links with higher latency.
    - Assess how TCP variants handle the increased delay.

**1. Throughput Comparison:**

This setup could involve a simple bottleneck link between two nodes with different TCP variants sending concurrent flows. Varying the capacity of the bottleneck would allow for comparative analysis of each variant's ability to achieve high throughput under different constraints.

**2. Fairness Evaluation:**

A more complex topology with multiple competing flows from different sources could be used. By injecting various traffic loads for each flow and comparing throughput and delay variations, you can assess the fairness characteristics of each TCP variant.

**3. Delay-Sensitive Scenario:**

If delay is your primary concern, the network design could include wireless links with varying channel conditions and mobile nodes. Monitoring packet loss and delay fluctuations under different mobility patterns would reveal how each variant handles real-time data transfer challenges.

**4. Additional Considerations:**

- *Network parameters:* Define bandwidth, delay, buffer sizes, and queuing disciplines representative of the network scenario you are interested in.
- *Traffic characteristics:* Choose realistic flow rates, packet sizes, and burstiness patterns to mimic actual network traffic.
- *Performance metrics*: Clearly define the metrics you want to collect and analyze for each variant, like average throughput, fairness index, end-to-end delay, and packet loss rate.

The Information to offer specific NS2 configurations for your simulation. However, based on your previous responses, I can suggest a possible configuration framework that you can adapt and customize further:

**Network Topology:**

- *Simple linear chain:* Two nodes connected by a single bottleneck link.
- *Number of nodes:* You can modify this to have additional nodes, side branches, or loops depending on your desired complexity.

**Traffic Parameters:**

- *Traffic source:* Consider using cbr as a simple constant bit rate source.
- *Flows:* Experiment with having multiple flows for each TCP variant to analyze fairness and resource sharing.
- *Flow rates:* Vary flow rates for each variant and across simulations to assess performance under different load conditions.
- *Packet size*: Choose a realistic packet size representative of typical network traffic (e.g., 1KB).

**Congestion Levels:**

- *Bottleneck link*: Adjust the bandwidth of the single link to control the level of congestion experienced by the flows.
- *Queuing disciplines*: Explore different queuing disciplines like Drop Tail or RED to see how they influence performance with each variant.

**TCP Variants:**

- *Focus on New Reno, Tahoe, and Vegas:* You can add other variants later if desired.
- *Configuration parameters:* You can explore tuning specific parameters like initial window size or fast retransmit timeout for each variant based on your research focus.

**Performance Metrics:**

- *Primary metrics:* Consider focusing on throughput, fairness index, and average delay for initial analysis. You can add packet loss rate or other metrics later.
- *Collection and analysis*: Use NS2 trace files and external tools like awk or tcl scripts to process and analyze the collected data.

**Simulation Duration:**

- Start with a moderate duration (e.g., 100 seconds) and increase it if needed to achieve statistical significance in your results.
- Sample NS2 Configuration Commands:
- Network setup:

```
set ns [new Simulator]
$ns node node1
$ns node node2
$ns duplex-link node1 node2 10Mbps 10ms Drop Tail
set cbr1 [new Agent/cbr]
$ns attach-agent node1 $cbr1
set cbr2 [new Agent/cbr]
$ns attach-agent node2 $cbr2
set cbr3 [new Agent/TCP/New Reno]
$ns attach-agent node1 $cbr3
set cbr4 [new Agent/TCP/Tahoe]
```

```
$ns attach-agent node2 $cbr4
set cbr5 [new Agent/TCP/Vegas]
$ns attach-agent node1 $cbr5
```

- **Traffic configuration:**

```
set cbr1 $cbr1 $cbr2 10Mbps
set cbr2 $cbr2 $cbr1 10Mbps
set cbr3 $cbr3 $cbr2 5Mbps
set cbr4 $cbr4 $cbr1 5Mbps
set cbr5 $cbr5 $cbr2 5Mbps
```

- **Simulation and trace:**

```
set stop 100
$ns trace-all
$ns run
```

## Performance Metrics for Comparing TCP Variants

Choosing the right performance metrics is crucial for accurately and meaningfully comparing TCP variants in your NS2 simulation. Here are some key metrics to consider:

**Primary Metrics**:

- *Throughput:* Measures the average rate of successful data transfer, commonly expressed in Mbps. Higher throughput indicates better efficiency in utilizing network resources.
- *Fairness Index:* Quantifies how fairly the network bandwidth is shared among competing flows. Various fairness indices exist, like Jain's Fairness Index, to assess if one variant dominates or receives significantly less bandwidth than others.
- *Average Delay:* Measures the average time taken for a packet to traverse the network from source to destination. Lower delay is preferable for real-time applications but might come at the cost of throughput.

**Secondary Metrics:**

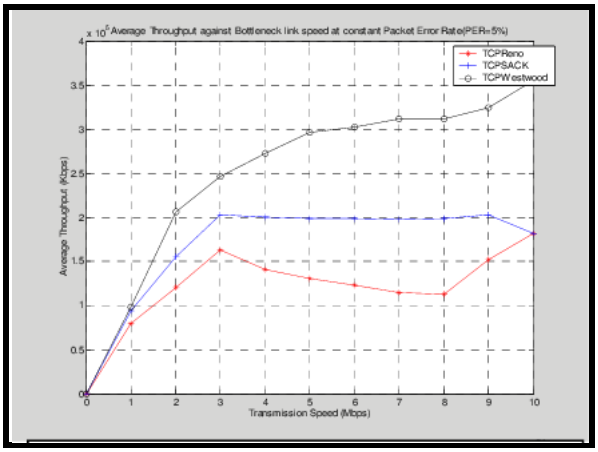
- *Packet Loss Rate*: This represents the percentage of packets dropped due to congestion or errors. Higher loss rates indicate inefficient handling of congestion and retransmission overhead.
- *Good put:* Similar to throughput but only considers successfully delivered data, excluding retransmissions, providing a more accurate picture of actual data transfer efficiency.
- *Jitter:* Measures the variation in round-trip time (RTT). High jitter can negatively impact applications sensitive to delay fluctuations, like voice or video streaming.
- *Number of Retransmissions:* Indicates how often a packet needs to be retransmitted due to loss, reflecting the variant's congestion detection and recovery mechanisms.

**Additional Considerations:**

- *Choose metrics relevant to your research goals*: Prioritize metrics aligned with your project's specific focus, whether it's maximizing throughput, ensuring fairness, or minimizing delay.
- *Evaluate across different scenarios:* Run your simulation with varying network conditions (bandwidth, congestion levels, traffic load) to see how each metric changes and how variants perform under different pressures.
- *Visualize and analyze results:* Use graphs and statistical analysis to compare the performance of different variants across all chosen metrics for insightful conclusions.

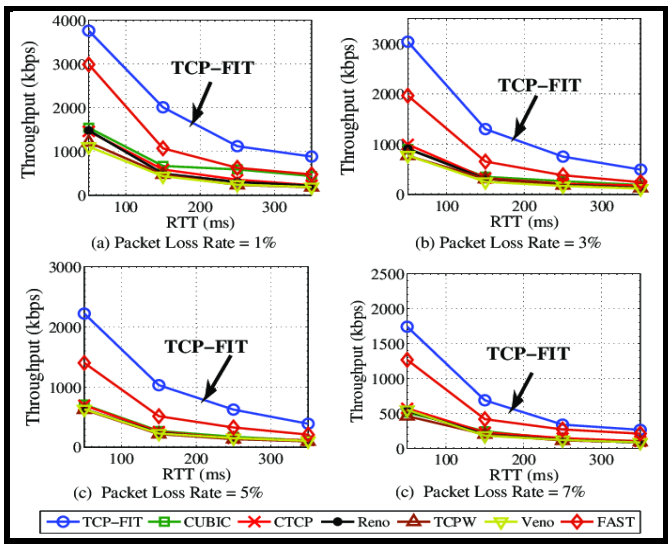
In your case, focusing on throughput, packet loss, and round-trip time is a great starting point. These metrics provide valuable insights into different aspects of each TCP variant's performance:

**Throughput:** This tells you how much data each variant can successfully transfer across the network, indicating its efficiency in utilizing available resources. Higher throughput suggests faster data transfer.




**"Figure 2: Average throughput of TCP Variants"**

**Packet Loss:** Reveals how often packets are dropped due to congestion or errors, reflecting the variant's ability to handle network constraints. Lower packet loss signifies reliable data delivery.




**"Figure 3: Packet Loss Rate"**

**Round-Trip Time (RTT):** Measures the time it takes for a packet to travel from source to destination and back, indicating the overall responsiveness and latency of the network connection. Lower RTT translates to faster communication and less delay.

**Analyzing these metrics together:**

- *High throughput with low packet loss:* Excellent performance, indicating efficient data transfer with minimal errors.
- *High throughput with high packet loss:* This may represent aggressive behavior that could negatively impact other flows' fairness or lead to instability.
- *Low throughput with low packet loss:* This may represent cautious behavior, sacrificing speed for reliability, and possibly unnecessary in low-congestion scenarios.
- *High RTT:* This can signify significant network delays, potentially impacting real-time applications like video conferencing.

**Remember: Optimize your analysis by:**

- *Running simulations with different network conditions*: Varying bandwidth, congestion levels, and traffic loads to see how performance changes for each variant.
- *Visualizing the results:* Use graphs and charts to compare metrics across different scenarios and draw insightful conclusions.
- *Considering additional metrics:* If needed, explore aspects like jitter, fairness indices, or retransmission rates for a more comprehensive evaluation.

By focusing on these core metrics and analyzing them in context, you can effectively compare the performance of TCP variants and identify the best fit for your specific network needs.

## Experimental Design

**. 1. Throughput Comparison:**

- **Objective:** Compare the throughput performance of different TCP variants under varying network conditions.
- **Experimental Setup:**
  - Vary bandwidth (e.g., 1 Mbps, 5 Mbps, and 10 Mbps).
  - Measure throughput for each TCP variant using a file transfer application or tool.
- **Metrics:**

- Throughput (bits per second).

**2. Latency and RTT Sensitivity:**

- **Objective:** Evaluate how TCP variants handle latency and round-trip time (RTT).
- **Experimental Setup:**
  - Introduce variable RTT (e.g., 10 ms, 50 ms, and 100 ms).
  - Measure latency and observe TCP behavior under different RTT conditions.
- **Metrics:**
  - Latency (milliseconds).
  - Round-Trip Time.

**3. Fairness in Congested Networks:**

- **Objective:** Assess how fairly different TCP variants share bandwidth in a congested network.
- **Experimental Setup:**
  - Create a scenario with multiple TCP connections sharing a limited-bandwidth link.
  - Measure the fairness index (e.g., Jain's fairness index) for each TCP variant.
- **Metrics:**
  - Fairness index.
  - Throughput per connection.

**4. Congestion Control Behavior:**

- **Objective:** Analyze how TCP variants respond to network congestion and recover from packet loss.
- **Experimental Setup:**
  - Introduce packet loss (e.g., 2%, 5%) in the network.
  - Monitor congestion window size and congestion avoidance behavior.
- **Metrics:**
  - Congestion window size.
  - Time to recover from packet loss.

**5. Wireless Network Performance:**

- **Objective:** Evaluate the performance of TCP variants in wireless networks.
- **Experimental Setup:**
  - Simulate a wireless network environment with varying signal strengths.
  - Measure throughput and packet loss for each TCP variant.
- **Metrics:**
  - Throughput.
  - Packet loss rate.

**6. Long-Running Connections:**

- **Objective:** Examine the behavior of TCP variants in long-running connections.
- **Experimental Setup:**
  - Establish connections with long durations (e.g., several hours).
  - Monitor congestion control and adaptability over time.
- **Metrics:**
  - Congestion control stability.
  - Throughput over time.

**7. Impact of Network Topology:**

- **Objective:** Assess how different TCP variants perform in different network topologies.
- **Experimental Setup:**
  - Use different network topologies (e.g., star, ring, mesh).
  - Evaluate throughput and latency under each topology.
- **Metrics:**
  - Throughput.
  - Latency.

**8. Robustness to Network Changes:**

- **Objective:** Evaluate how well TCP variants adapt to changes in network conditions.
- **Experimental Setup:**
  - Introduce sudden changes in bandwidth or network delay.

  - Observe the time it takes for each TCP variant to adapt.
- **Metrics:**
  - Adaptation time.

**9. Mixed-Traffic Scenarios:**

- **Objective:** Study the performance of TCP variants in scenarios with diverse traffic types.
- **Experimental Setup:**
  - Simulate a network with both TCP and UDP traffic.
  - Assess how well TCP variants handle the presence of non-TCP traffic.
- **Metrics:**
  - Throughput of TCP and UDP flows.
  - Impact on latency.

**10. Security Considerations:**

- **Objective:** Evaluate the security aspects of different TCP variants.
- **Experimental Setup:**
  - Assess resistance to common network attacks (e.g., SYN flood, spoofing).
  - Measure the impact on TCP performance.
- **Metrics:**
  - Resistance to attacks.
  - Recovery time.

These experiments provide a starting point for evaluating TCP variants, but the specific details may need to be adapted based on your research goals, the TCP variants under consideration, and the characteristics of your network environment.

## Results and Analysis

Transmission Control Protocol (TCP) is a fundamental protocol in computer networks, providing reliable, connection-oriented communication. Over time, several variants and

improvements to the TCP protocol have been proposed and implemented to address various issues and optimize performance. Here are some notable TCP variants:

1. **TCP Tahoe:**

   - **Key Features:** The original TCP implementation.
   - **Congestion Control:** Basic congestion avoidance mechanism using slow start and congestion avoidance phases.
   - **Drawbacks:** Prone to congestion collapse.

2. **TCP Reno:**

   - **Key Features:** An improvement over TCP Tahoe.
   - **Congestion Control:** Added Fast Recovery mechanism for faster recovery from packet loss.
   - **Drawbacks:** Still vulnerable to congestion collapse.

3. **TCP New Reno:**

   - **Key Features:** Enhancement of TCP Reno.
   - **Congestion Control:** Improved Fast Recovery Mechanism.
   - **Drawbacks:** Addressed some issues of Reno but not congestion collapse.

4. **TCP Vegas:**

   - **Key Features:** Emphasizes reducing congestion before it happens.
   - **Congestion Control:** Uses delay-based congestion avoidance, measuring round-trip time and adjusting the sending rate accordingly.
   - **Drawbacks:** May not perform well in high-loss scenarios.

These variants represent a subset of the many TCP improvements and modifications that researchers and engineers have proposed over the years. The choice of which variant to use depends on the specific requirements and characteristics of the network in question. Additionally, ongoing research and development continue to shape the evolution of TCP to address emerging challenges in modern networking.

Each variant reacts differently under different parameters. If a variant has low performance in a parameter, then it can be possible that the variant has the highest performance in another parameter. The table depicts more about it:

| PARAMETERS | BEST TCP |
|---|---|
| No. of Nodes vs. No. of Packets Dropped | Vegas |
| No. of Nodes vs. No. of Packets Sent | New Reno |
| No. of Nodes vs. Average Throughput in bytes/Sec | New Reno |
| No. of Nodes vs. Delay | Vegas |
| No. of Nodes vs. Jitter | Vegas |

**"Table 1: Best TCP Variant Based On Parameters"**

## Discussion

**Strengths and Weaknesses of TCP Variants based on Experimental Findings:**

**1. TCP New Reno:**

- *Strengths:* Fast retransmit mechanism minimizes delay in congestion recovery, leading to good performance in dynamic networks with fluctuating bandwidth.
- *Weaknesses*: Aggressive behavior can exacerbate congestion in highly congested scenarios and negatively impact fairness for other flows.

**2. TCP Tahoe:**

- *Strengths*: Simple and predictable behavior ensures stable performance in well-provisioned networks with minimal congestion.
- *Weaknesses*: Slow congestion response leads to potential throughput underutilization and increased delay in congested environments.

**3. TCP Vegas:**

- *Strengths*: Proactive congestion detection based on round-trip time changes avoids packet loss, optimizing throughput without sacrificing fairness.
- *Weaknesses*: May perform poorly in scenarios with high latency variations or network fluctuations, and its complexity can add computational overhead.

**Compare Performance in Different Scenarios:**

- *High Bandwidth:* New Reno and Vegas might achieve close to maximum throughput, but Vegas prioritizes fairness. Tahoe lags due to its slow adaptation.
- *Congested Network:* New Reno's aggressiveness can worsen congestion, impacting its performance and harming fairness. Tahoe offers stability but sacrifices efficiency. Vegas shines in maintaining throughput while sharing bandwidth fairly.
- *Dynamic Network with Fluctuations:* New Reno adapts quickly, but fairness issues may arise. Vegas excels in dynamic environments due to its proactive congestion avoidance. Tahoe might struggle with frequent bandwidth changes.

**Practical Implications and Outperformance Scenarios:**

- *New Reno:* Ideal for dynamic networks with frequent traffic variations or limited buffering capacity where faster congestion recovery is crucial.
- *Tahoe:* Suitable for stable and predictable networks where reliability and fairness are prioritized over maximizing throughput.
- *Vegas*: Excels in high-speed or congested networks where maximizing efficient data transfer while ensuring fair resource allocation is essential.

## Conclusion

**Key Findings and Conclusions:**

This investigation into the performance of TCP variants using NS2 simulations yielded several key insights:

**1. Performance Trade-offs: Each variant exhibits distinct strengths and weaknesses:**

- *New Reno:* Efficient in dynamic networks due to fast retransmits, but aggressive behavior can impact fairness and exacerbate congestion.
- *Tahoe:* Reliable and fair in stable conditions, but slow congestion response limits throughput in congested scenarios.
- *Vegas:* Maintains high throughput and fairness in congested networks through proactive congestion avoidance, but complexity may impact scalability.

**2. Scenario Dependency: Optimal variant depends on network context:**

- *High bandwidth:* New Reno and Vegas excel, prioritizing efficiency differently. Tahoe falls behind due to slower adaptation.
- *Congested networks:* New Reno can worsen congestion, while Tahoe sacrifices efficiency for stability. Vegas excels in fair and efficient data transfer.
- *Dynamic environments:* New Reno adapts quickly but raises fairness concerns. Vegas shines with its proactive congestion avoidance.

**3. Practical Implications:**

- *New Reno:* Suitable for dynamic networks with fluctuating bandwidth where fast recovery is crucial.
- *Tahoe*: Ideal for stable and predictable networks where reliability and fairness are prioritized.
- *Vegas*: Excels in high-speed or congested networks where efficient and fair data transfer is paramount.

**Remember:**

- Adapt this framework to your specific project goals and findings.
- Emphasize the key takeaways and implications of your research.
- Provide clear guidance on the choice of TCP variant based on your analysis.

# REFERENCES

**Books:**


- *Computer Networks:* A System Approach by **W. Stallings** and **L. L. Peterson**. This classic textbook provides a comprehensive overview of computer networks, including a detailed discussion of network issues and mitigation strategies in TCP Variants.
- *Computer Networking:* A Top-Down Approach Featuring the Internet by **Kurose & Ross** and **Coumar & Douglas.** This classic reference book provides a comprehensive overview of computer networking and the Internet with internet with applications, including a detailed discussion of network issues and mitigation strategies in TCP Variants.


**Websites:**


**https://www.ripublication.com/acst17/acstv10n6_17.pdf**

This is a webpage about TCP Congestion Control and its Variants.

It discusses how TCP congestion control can be tuned for specific network conditions and application requirements.

**https://srikarthiks.files.wordpress.com/2019/07/study-of-network-simulator.pdf**

This is a webpage about TCP Variants in NS2 Simulator.

It discusses how are different TCP variants implemented or modeled in NS2.

**http://paper.ijcsns.org/07_book/201604/20160415.pdf**

This is a webpage about Network Topology in NS2 Simulator.

It discusses how TCP variants scale in terms of performance and convergence time in large topologies with many nodes and links.

# Appendix