Practical 5

Aim :- TCP Variants in Wireless Networks
Objective: Study the performance of different TCP variants in a wireless environment.
Tasks:
1) Implement TCP Tahoe, Reno, and NewReno.
2) Create a wireless network with mobile nodes and simulate TCP traffic.
3) Compare the performance of different TCP variants in terms of throughput and packet loss.

Theory :-

TCP Variants

TCP (Transmission Control Protocol) is a fundamental protocol used for data transmission over networks. Different variants of TCP have been developed to optimize performance in various network environments, especially in wireless settings.

1. TCP Tahoe

Definition: TCP Tahoe is one of the earliest variants of TCP designed to improve congestion control in networks.

Key Features:

- **Slow Start:** It starts with a small congestion window and increases it exponentially with each acknowledgment received until it reaches a threshold.
- **Congestion Avoidance:** After reaching the threshold, it switches to linear growth of the congestion window.
- **Fast Retransmit:** If three duplicate ACKs are received, TCP Tahoe assumes packet loss and reduces the congestion window to one, restarting the slow start process.

2. TCP Reno

Definition: TCP Reno is an enhancement of TCP Tahoe, introducing more efficient congestion control mechanisms.

Key Features:

- **Fast Recovery:** After detecting packet loss through three duplicate ACKs, TCP Reno halves the congestion window but continues to send new packets instead of reverting to the slow start phase.
- **Improved Throughput:** This approach helps to maintain a higher throughput in scenarios with packet loss, particularly in wireless environments where losses can occur frequently.

3. TCP NewReno

Definition: TCP NewReno builds upon TCP Reno, addressing some of its limitations regarding the recovery of multiple lost packets in a single transmission.

Key Features:

- **Partial Acknowledgments:** NewReno allows for more efficient handling of multiple packet losses by utilizing partial acknowledgments, which help in determining which packets have been successfully received.
- **Improved Recovery:** It can recover from multiple lost packets without completely dropping to the slow start phase, thus maintaining better throughput and reducing delay.

Wireless Network Simulation

Definition: A wireless network simulation involves creating a virtual environment to study the behavior of TCP variants under wireless conditions.

Key Components:

- **Mobile Nodes:** The simulation includes several mobile nodes that represent devices in a wireless network, capable of moving and changing their connectivity dynamically.
- **TCP Traffic:** The nodes generate TCP traffic to emulate real-world applications, allowing for the analysis of how different TCP variants perform under varying network conditions.

Performance Comparison

Objective: The main goal is to compare the performance of TCP Tahoe, Reno, and NewReno in terms of:

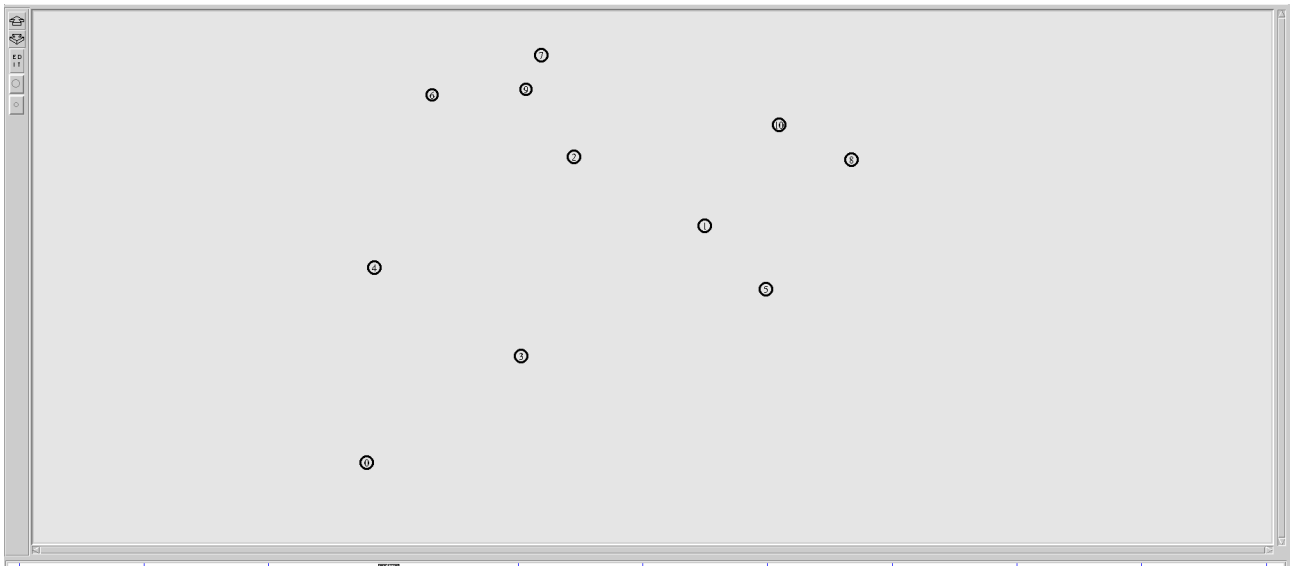
- **Throughput:** The amount of data successfully transmitted over the network in a given time period. Higher throughput indicates better performance, especially in wireless settings where bandwidth may be limited.
- **Packet Loss:** The frequency of lost packets during transmission. Lower packet loss rates signify a more reliable protocol, crucial in wireless networks where interference can disrupt communication.

Code:-

```
# Simulation Parameters
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(rp) AODV
set val(nn) 11
set val(x) 500
set val(y) 400
set val(stop) 10
set val(energymodel) EnergyModel
set val(initialenergy) 1000

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

```
# Open trace and NAM files
set tf [open tcp_variants.tr w]
$ns trace-all $tf
set nf [open tcp_variants.nam w]
$ns namtrace-all-wireless $nf $val(x) $val(y)

# Create topography and load a flat grid
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Create Global Operations Director (GOD) and a wireless channel
create-god $val(nn)
set chan_1_ [new $val(chan)]

# Node configuration
$ns node-config -adhocRouting $val(rp) \
   -llType $val(ll) \
   -macType $val(mac) \
   -ifqType $val(ifq) \
   -ifqLen $val(ifqlen) \
   -antType $val(ant) \
   -propType $val(prop) \
   -phyType $val(netif) \
   -channel $chan_1_ \
   -topoInstance $topo \
   -energyModel $val(energymodel) \
   -initialEnergy $val(initialenergy)

# Create nodes and set initial positions
for {set i 0} {$i < $val(nn)} {incr i} {
   set node_($i) [$ns node]
   $node_($i) set X_ [expr 10+round(rand()*480)]
   $node_($i) set Y_ [expr 10+round(rand()*380)]
   $node_($i) set Z_ 0.0
}

# Set mobility for nodes
for {set i 0} {$i < $val(nn)} {incr i} {
   $ns at [expr 0.2+round(rand())] "$node_($i) setdest [expr 10+round(rand()*480)] [expr
10+round(rand()*380)] [expr 60+round(rand()*30)]"
}

# Setup TCP variants and traffic flows
set tcp_tahoe [new Agent/TCP]
$tcp_tahoe set class_ 1
$tcp_tahoe set type_ Tahoe
$ns attach-agent $node_(0) $tcp_tahoe

set tcp_reno [new Agent/TCP]
$tcp_reno set class_ 1
$tcp_reno set type_ Reno
```

```
$ns attach-agent $node_(1) $tcp_reno

set tcp_newreno [new Agent/TCP]
$tcp_newreno set class_ 1
$tcp_newreno set type_ NewReno
$ns attach-agent $node_(2) $tcp_newreno

# Attach sinks to receive TCP traffic
set sink1 [new Agent/TCPSink]
$ns attach-agent $node_(3) $sink1
$ns connect $tcp_tahoe $sink1

set sink2 [new Agent/TCPSink]
$ns attach-agent $node_(4) $sink2
$ns connect $tcp_reno $sink2

set sink3 [new Agent/TCPSink]
$ns attach-agent $node_(5) $sink3
$ns connect $tcp_newreno $sink3

# Start CBR traffic to evaluate performance
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $tcp_tahoe
$cbr set packetSize_ 512
$cbr set interval_ 0.1
$cbr set rate_ 1mb
$cbr set maxpkts_ 10000
$ns at 0.4 "$cbr start"

# Simulation stop and finish procedure
proc finish {} {
    global ns tf nf
    $ns flush-trace
    close $tf
    close $nf
    exec nam tcp_variants.nam &
    exit 0
}

# Simulation stop time
$ns at $val(stop) "finish"
$ns run
```


7
9
6
10
2
8
1
4
5
3
0