Ex. No. 11	**TCP and UDP**	**Karthik D**
16 October 2021	**UCS1511 - Networks Lab**	**195001047**

Aim

To analyze the difference between a TCP and UDP connection sharing a bottle-neck link using the Network Simulator (NS2) tool and using a simple tcl-script generated network schedule comprising of FTP traffic using TCP packets

Question

Write a .tcl script to simulate:

1. Create 6 nodes and the links between the nodes as
    a. 0 →2 2Mb 10 ms duplex link
    b. 1→2 2Mb 10 ms duplex link
    c. 2→3 0.3Mb 100ms simplex link
    d. 3→2 0.3Mb 100ms simplex link (link 2→3 is a bottleneck)
    e. 3→4 0.5Mb 40ms duplex link
    f. 3→5 0.5Mb 40ms duplex link
2. Align the nodes properly
3. Set Queue Size of link (n2-n3) to 10 (or) 5
4. Set up a TCP connection over 0 and 4 and its flow id,window size, packet size
5. Set up a UDP connection over 1 and 5 with flow id, type, packet size, rate, random field
6. Set different colors for TCP and UDP
7. Run the simulation for 5 seconds, and show the simulation in network animator andin trace file.

Analyze the performance of TCP and UDP from the simulation

Algorithm

1. Connection Setup

Step 1: Instantiate a new Simulator object. Definite the routing protocol as Link-State (LS). Define colors for different flows for visual distinction

Step 2: Set the simulator to record the simulations in the NAM format. Supply the output file name to be generated at the end

Step 3: Define six different nodes in the network simulator — n0 to n5. Create duplex-links and simplex-links as per the specification with the specified bandwidths and propagation delays. Use droptail queueing if required

Step 4: Create and attach a UDP agent instance to node n0 and a *null* agent instance to node n5 to receive these packets. Setup a logical connection between the n0 and n5 i.e. set their destination IP and port addresses. Attach a CBR agent as the application layer protocol to the UDP agent.

Step 5: Create and attach a TCP agent instance to node n1 and a TCP sink agent instance to node n4 to receive these packets. Setup a logical connection between the n1 and n4 i.e. set their destination IP and port addresses. Attach an FTP agent as the application layer protocol to the TCP agent.

Step 6: Set other connection parameters including window size, class, packet size and flow id

Step 7: Schedule the CBR and FTP connections to operate from between specific time instances

Step 8: Terminate the simulation after 5s. Flush the simulation data collected to the NAM file object created in step-2. Run the generated output trace file using the NAM simulator.

Step 9: Extract the sequence numbers from the trace file and plot them for later analysis

TCL Program Code

1. tcp-udp.tcl - Trace file and simulation generation script for tcp-udp comparison

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

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

# Open the trace file
set nf [open tcp-udp.tr w]
$ns trace-all $nf

#Open the NAM trace file
set nf [open tcp-udp.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
```

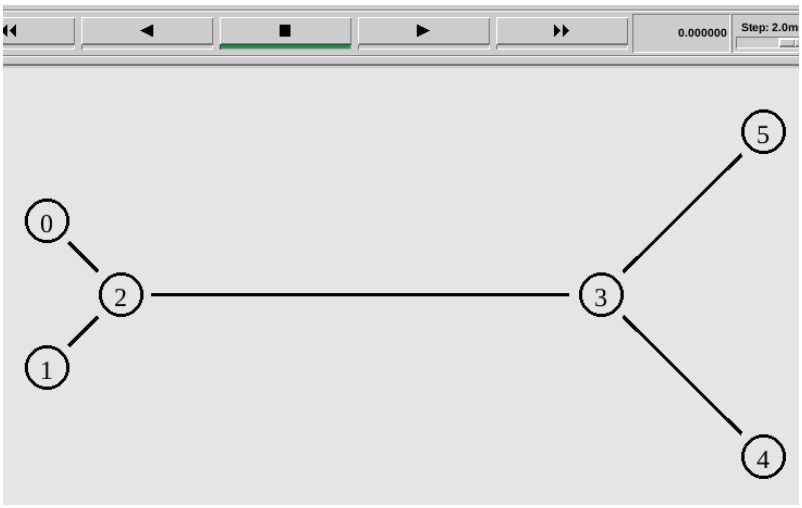
```
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the NAM trace file
    close $nf
    # Extract graph points and plot them
    exec awk -f extract_seqno_tcp.awk tcp-udp.tr > tcp_graph.tr &
    exec awk -f extract_seqno_udp.awk tcp-udp.tr > udp_graph.tr &
    exec xgraph tcp_graph.tr udp_graph.tr &
    #Execute NAM on the trace file
    exec nam tcp-udp.nam &
    exit 0
}

#Create six 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]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 40ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient right-down
$ns duplex-link-op $n3 $n5 orient right-up
```

```
#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

$tcp set window_ 1000
$tcp set packetSize_ 400

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
# Create a logic connection between the two agents
# Done by assigning respective IPs mutually
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 400
$cbr set rate_ 0.2mb
$cbr set random_ false

#Schedule events for the CBR and FTP agents
$ns at 0.5 "$cbr start"
```

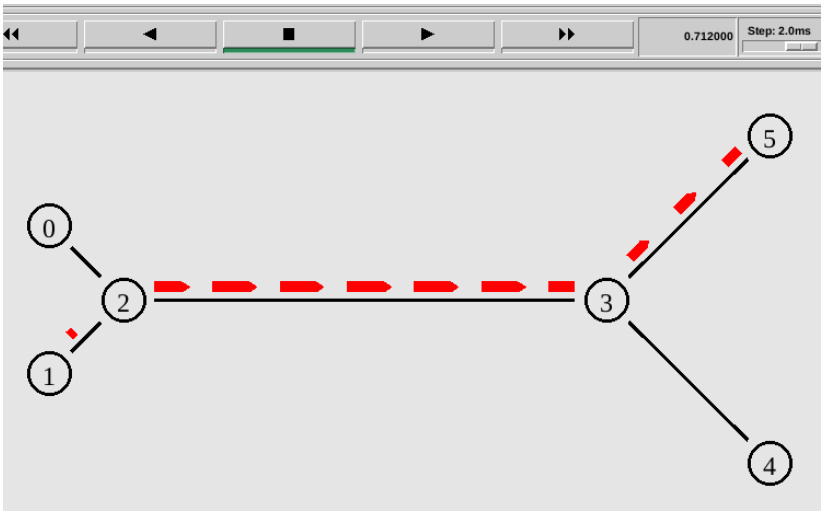
```
$ns at 1.0 "$ftp start"
$ns at 5.0 "$ftp stop"
$ns at 5.0 "$cbr stop"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Sample Output

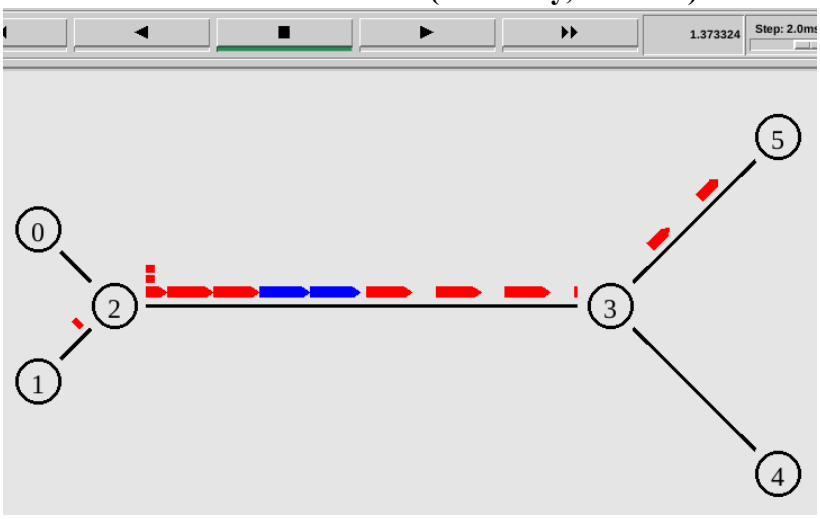
a. Initial State - Connection Setup



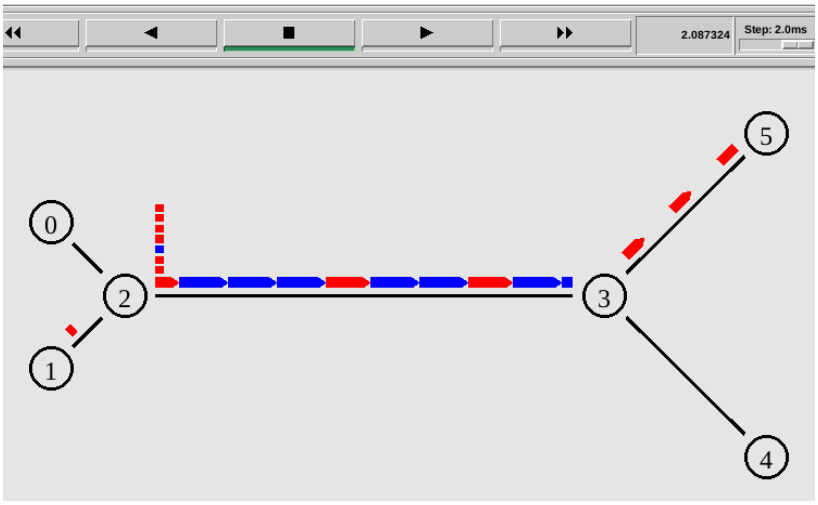
b. UDP transmission starts



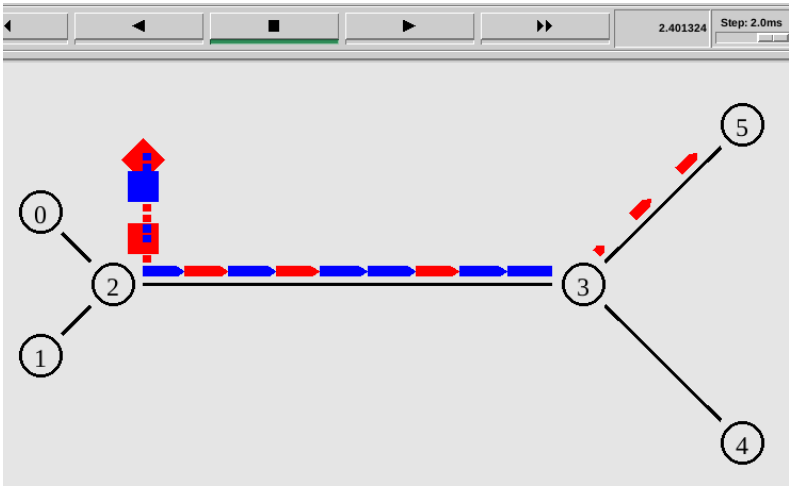
c. **TCP Transmission - Slow Start (currently, rwnd=2)**



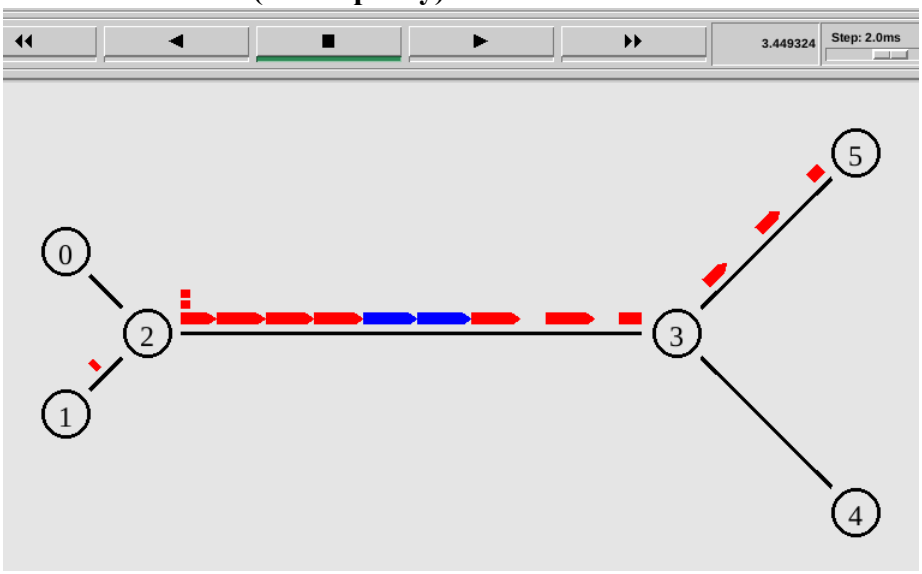
d. **TCP Transmission - Exponential Increase (currently, rwnd=8)**



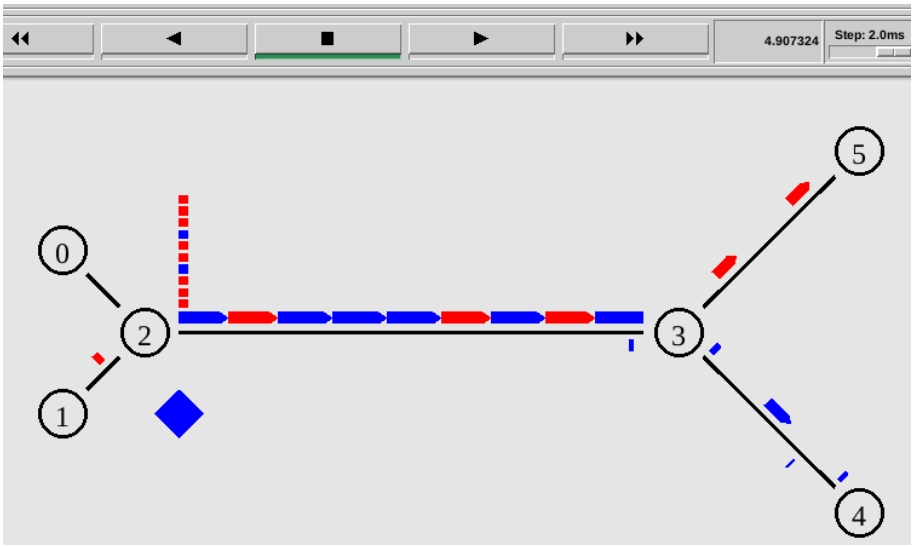
e. **Congestion occurs in the bottleneck link - Packets dropped**



f. TCP Slow Restart (Tahoe policy)



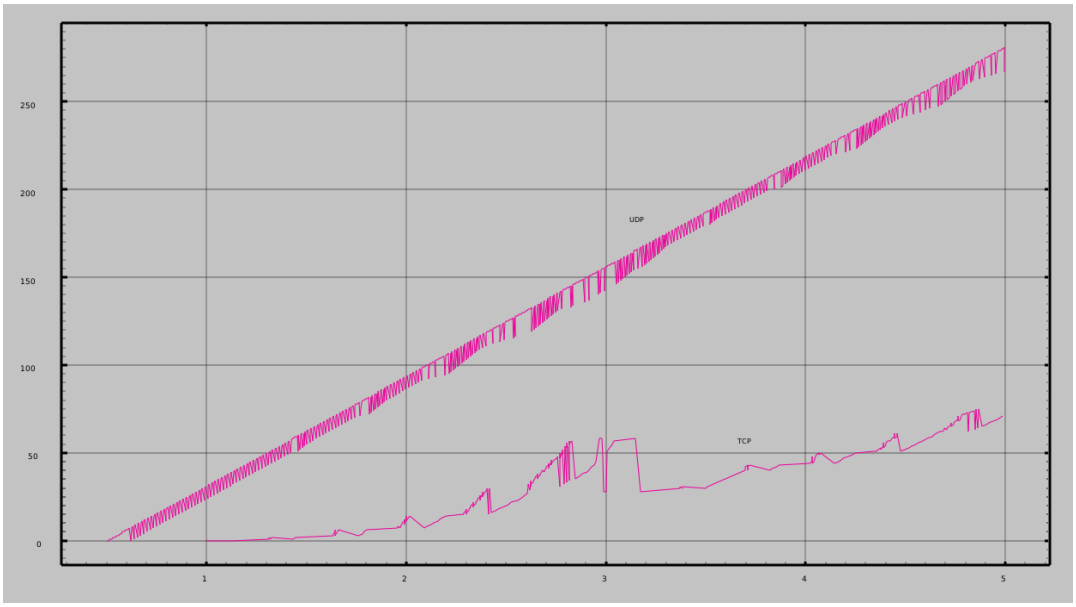
g. TCP Exponential Increase (again) followed by Congestion



Differences Observed between TCP and UDP

1. TCP uses policy to detect and overcome congestion using its slow-start policy. Whenever congestion occurs, it starts again, after lowering the threshold.
   UDP on the other hand, continues to transmit packets in spite of congestion and packet-loss. This is because, it does not offer any form of congestion control or error control (except the optional checksum)

2. Due to slow-start and restarts, TCP transmits a much lower number of packets overall when compared to UDP over the entire duration. The following graph shows the plot of sequence number against time for packets that reached the destination successfully.



The lower graph (labelled “TCP” is for the TCP connection)
Note that both TCP and UDP connections were given the same packet-size, bottleneck and transmission rate

3. TCP ensures error control by making sure all the packets reach the destination. UDP gives no such guarantees.

Result

Analyzed the difference between TCP and UDP agents using the Network Simulator (NS2) tool and using a simple tcl-script generated network schedule consisting of CBR and FTP traffic. Through this implementation, the following aspects were understood:

1. Basic scripting for NS and NAM using TCL language
2. Difference between the TCP and UDP protocols
3. Working with NS2 and NAM to observe network traffic conceptually
4. Working with Xgraph and awk to extract and plot trace information