Experiment 10 : Install Network Simulator 2/3. Create a wired network using dumbbell topology. Attach agents, generate both FTP and CBR traffic, and transmit the traffic. Vary the data rates and evaluate the performance using metric throughput, delay, jitter and packet loss.

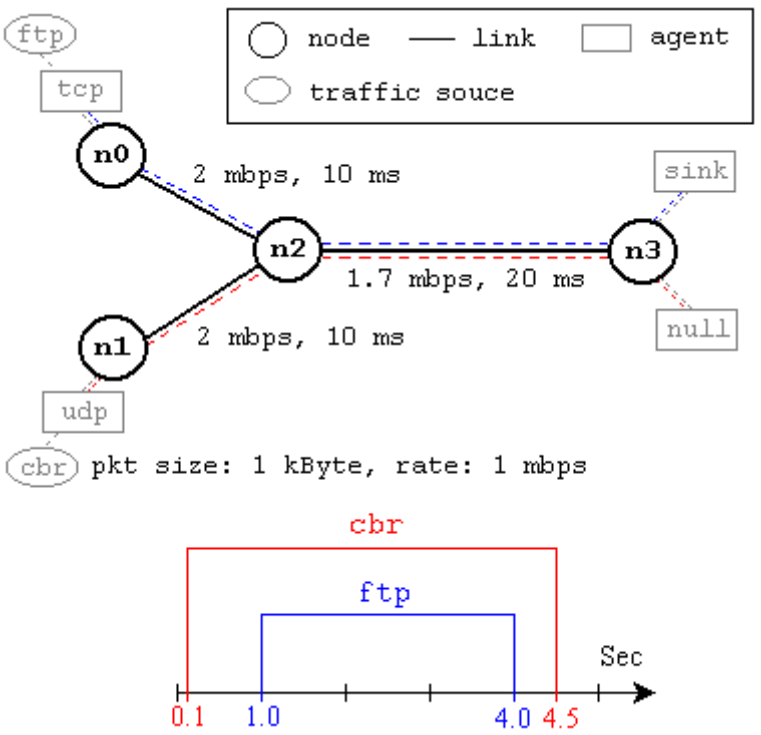

Figure : A Simple Network Topology

Explanation:

This network consists of 4 nodes (n0, n1, n2, n3) as shown in above figure. The duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay. The duplex link between n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A "tcp" agent is attached to n0, and a connection is established to a tcp "sink" agent attached to n3. As default, the maximum size of a packet that a "tcp" agent can generate is 1KByte. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. A "udp" agent that is attached to n1 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received. A "ftp" and a "cbr" traffic generator are attached to "tcp" and "udp" agents respectively, and the "cbr" is configured to generate 1 KByte packets at the rate of 1 Mbps. The "cbr" is set to start at 0.1 sec and stop at 4.5 sec, and "ftp" is set to start at 1.0 sec and stop at 4.0 sec.

Initial Setup

```
set ns [new Simulator]                                          ; # preamble initialization

Queue/JoBS set drop_front_ false                                ; # use drop-tail
Queue/JoBS set trace_hop_ true                                  ; # enable statistic traces
Queue/JoBS set adc_resolution_type_ 0                           ; # see ''commands at a glance''
Queue/JoBS set shared_buffer_ 1                                 ; # all classes share a common buffer
Queue/JoBS set mean_pkt_size_ 4000                              ; # we expect to receive 500-Byte pkts
Queue/Demarker set demarker_arrvs1_ 0                           ; # reset arrivals everywhere
Queue/Demarker set demarker_arrvs2_ 0
Queue/Demarker set demarker_arrvs3_ 0
Queue/Demarker set demarker_arrvs4_ 0
Queue/Marker set marker_arrvs1_ 0
Queue/Marker set marker_arrvs2_ 0
Queue/Marker set marker_arrvs3_ 0
Queue/Marker set marker_arrvs4_ 0

set router(1) [$ns node]                                        ; # set first router
set router(2) [$ns node]                                        ; # set second router
set source [$ns node]                                           ; # set source
set sink [$ns node]                                             ; # set traffic sink
set bw 10000000                                                 ; # 10 Mbps
set delay 0.001                                                 ; # 1 ms
set buff 500                                                    ; # 500 packets
```

Creating the JoBS links

```
$ns duplex-link $router(1) $router(2) $bw $delay JoBS           ; # Creates the JoBS link
$ns_ queue-limit $router(1) $router(2) $buff
set l [$ns_ get-link $router(1) $router(2)]
set q [$l queue]
$q init-rdcs -1 2 2 2     ; # Classes 2, 3 and 4 are bound by proportional delay differentiation with a factor of 2
$q init-rlcs -1 2 2 2     ; # Classes 2, 3 and 4 are bound by proportional loss differentiation with a factor of 2
$q init-alcs 0.01 -1 -1 -1                                      ; # Class 1 is provided with a loss rate bound of 1%
$q init-adcs 0.005 -1 -1 -1                                     ; # Class 1 is provided with a delay bound of 5 ms
$q init-arcs -1 -1 -1 500000                                    ; # Class 4 is provided with a minimumthroughput of 500 Kbps
$q link [$l link]                                               ; # The link is attached to the queue (required)
$q trace-file jobstrace                                         ; # Trace per-hop, per-class metrics to the file jobstrace
$q sampling-period 1                                            ; # Reevaluate rate allocation upon each arrival
$q id 1                                                         ; # Assigns an ID of 1 to the JoBS queue
$q initialize                                                   ; # Proceed with the initialization
```

Marking the traffic

Marking the traffic is handled by Marker objects. Markers are FIFO queues that set the class index of each packet. To ensure accuracy of the simulations, it is best to configure these queues to have a very large buffer, so that no packets are dropped in the Marker. Demarkers are used to gather end-to-end delay statistics.

```
$ns_ simplex-link $source $router(1) $bw $delay Marker          ; # set-up marker
$ns_ queue-limit $source $router(1) [expr $buff*10]      ; # Select huge buffers for markers
$ns_ queue-limit $router(1) $source [expr $buff*10]          ; # to avoid traffic drops
set q [$ns_ get-queue $source $router(1)]                          ; #in the marker
$q marker_type 2                                                  ; # Statistical marker
$q marker_frc 0.1 0.2 0.3 0.4          ; # 10% Class 1, 20% Class 2, 30% Class 3, 40% Class 4.
$ns_ simplex-link $router(2) $sink $bw $delay Demarker          ; # set-up demarker
$ns_ queue-limit $router(2) $sink [expr $buff*10]
$q trace-file e2e                                  ; # trace end-to-end delays to file e2e
```

Creating the Agent

The following OTcl code fragment creates a TCP agent and sets it up:

```
set tcp [new Agent/TCP]                                  ; # create sender agent
$tcp set fid_ 2                                          ; # set IP-layer flow ID
set sink [new Agent/TCPSink]                             ; # create receiver agent
$ns attach-agent $n0 $tcp                                ; # put sender on node $n0
$ns attach-agent $n3 $sink                               ; # put receiver on node $n3
$ns connect $tcp $sink                                   ; # establish TCP connection
set ftp [new Application/FTP]                  ; # create an FTP source "application"
$ftp attach-agent $tcp                                   ; # associate FTP with the TCP sender
$ns at 1.2 "$ftp start"                        ; #arrange for FTP to start at time 1.2 sec
```