COMP3331 18s2

Haibo Wang(z5135009)

Exercise1

Q1:

Node1 communicates with node 0 and node 2. Node 4 do packets follow

It doesn't change over time.

Q2:

At time 1.0, no throughputs go through node 1 and node 4, at time 1.2 it's back to normal.

It's not change.

Q3:

Yes, at 1.0 second, node 1 changes transmission path, at 1.2 second, it's back to normal.

Q4:

At the beginning, the nodes exchange information, and then node 1 do not transmit data to node 4, instead sending message which are sent by node 0 to node 2. Because we change cost between n1 and n4, it's greater that before.\

Q5:

We have two path from start to end. Because first we change the cost between n1 and n4 and also n3 and n5, then the two paths's cost from start to end will be same, second, we set multipath to true, which means if we have two paths and they have same cost, we can choose a path by random.

Exercise2

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

#Define different colors
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Yellow


#Open the nam trace file
set namf [open out.nam w]
$ns namtrace-all $namf

set f1 [open tcp1.tr w]
set f2 [open tcp2.tr w]
```

```
    #Define a 'finish' procedure
    proc finish {} {
        global ns namf f1 f2
        $ns flush-trace
        #Close the trace amd nam files
        close $namf
        close $f1
        close $f2
        #Execute nam on the trace file
        exec nam out.nam &
        # Execute gnuplot to display the two trace files tcp1.tr and tcp2.tr
        #exec gnuplot throughput.plot &
        exit 0
    }

    #Create eight 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]
    set n6 [$ns node]
    set n7 [$ns node]

    #Create links between the nodes
    $ns duplex-link $n0 $n1 10Mb 10ms DropTail
    $ns duplex-link $n1 $n2 2.5Mb 40ms DropTail
    $ns duplex-link $n2 $n3 10Mb 10ms DropTail
    $ns duplex-link $n1 $n6 2.5Mb 40ms DropTail
    $ns duplex-link $n4 $n2 2.5Mb 40ms DropTail
    $ns duplex-link $n4 $n6 2.5Mb 40ms DropTail
    $ns duplex-link $n4 $n5 10Mb 10ms DropTail
    $ns duplex-link $n6 $n7 10Mb 10ms DropTail

    # set the correct orientation for all nodes
    $ns duplex-link-op $n0 $n1 orient right
    $ns duplex-link-op $n3 $n2 orient right
    $ns duplex-link-op $n1 $n2 orient up
    $ns duplex-link-op $n1 $n6 orient right
    $ns duplex-link-op $n6 $n4 orient up
    $ns duplex-link-op $n6 $n7 orient right
    $ns duplex-link-op $n4 $n5 orient right
    $ns duplex-link-op $n2 $n4 orient right

    #Set Queue limit and Monitor the queue for the link between node 2 and
node 4
    $ns queue-limit $n2 $n4 10
    $ns duplex-link-op $n2 $n4 queuePos 0.5

    #Create a TCP agent and attach it to node n0
    set tcp1 [new Agent/TCP]
    $ns attach-agent $n0 $tcp1
```

```
#Sink for traffic at Node n5
set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1

#Connect
$ns connect $tcp1 $sink1
$tcp1 set fid_ 1

#Setup FTP over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

#Create a TCP agent and attach it to node n3
set tcp2 [new Agent/TCP]
$ns attach-agent $n3 $tcp2
#Sink for traffic at Node n5
set sink2 [new Agent/TCPSink]
$ns attach-agent $n5 $sink2
#Connect
$ns connect $tcp2 $sink2
$tcp2 set fid_ 2
#Setup FTP over TCP connection
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

#Create a TCP agent and attach it to node n7
set tcp3 [new Agent/TCP]
$ns attach-agent $n7 $tcp3
#Sink for traffic at Node n0
set sink3 [new Agent/TCPSink]
$ns attach-agent $n0 $sink3
#Connect
$ns connect $tcp3 $sink3
$tcp3 set fid_ 3
#Setup FTP over TCP connection
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3

#Create a TCP agent and attach it to node n7
set tcp4 [new Agent/TCP]
$ns attach-agent $n7 $tcp4

#Sink for traffic at Node n3
set sink4 [new Agent/TCPSink]
$ns attach-agent $n3 $sink4

#Connect
$ns connect $tcp4 $sink4
$tcp4 set fid_ 4

#Setup FTP over TCP connection
set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
```

```
    proc record {} {
            global sink1 sink2 f1 f2
            #Get an instance of the simulator
            set ns [Simulator instance]
            #Set the time after which the procedure should be called again
            set time 0.1
            #How many bytes have been received by the traffic sinks at n5?
            set bw1 [$sink1 set bytes_]
            set bw2 [$sink2 set bytes_]

            #Get the current time
            set now [$ns now]
            #Calculate the bandwidth (in MBit/s) and write it to the files
            puts $f1 "$now [expr $bw1/$time*8/1000000]"
            puts $f2 "$now [expr $bw2/$time*8/1000000]"
            #Reset the bytes_ values on the traffic sinks
            $sink1 set bytes_ 0
            $sink2 set bytes_ 0
            #Re-schedule the procedure
            $ns at [expr $now+$time] "record"
    }


    #Schedule events for all the agents
    #start recording
    $ns at 0.0 "record"

    #start FTP sessions
    $ns at 0.5 "$ftp1 start"
    $ns at 2.0 "$ftp2 start"
    $ns at 3.0 "$ftp3 start"
    $ns at 4.0 "$ftp4 start"

    #Stop FTP sessions
    $ns at 8.5 "$ftp1 stop"
    $ns at 9.5 "$ftp2 stop"
    $ns at 9.5 "$ftp3 stop"
    $ns at 7.0 "$ftp4 stop"

    #Call the finish procedure after 10 seconds of simulation time
    $ns at 10.0 "finish"

    #Run the simulation
    $ns run
```

Exercise3

Q1:

2000 and 3500 cause fragmenation. because data size is greater that max transmation unit. (MTU = 1500)
192.168.1.103.

2 fragments.

Q2:

Yes, because according to the rule of ICMP, server need to reply the same number of frame size as client's request.

Q3:

ID: 0x7a7b Length: 1500 Flag: 0x2000 Offset: 0

Q4:

No, because the routers have same MTU.

Q5:

client wili need resent this packe, and the original one cann't be reassemable to a complete fragment.