Exercise 1: Understanding the Impact of Network Dynamics on Routing

Question 1. Which nodes communicate with which other nodes? Which route do the packets follow? Does it change over time?

0:1

1: 0,4

4: 1,5

5: 4,3

3: 5,2

2: 3

The packets follow the path 0-1-4-5 and 2-3-5 , It does not change over time.

Question 2: What happens at time 1.0 and at time 1.2? Does the route between the communicating nodes change as a result of that?

At time 1.0 there is an error happened at the link between 1-4, the packets can get through the 1-4 link and just stop at node 1. It is repaired at time 1.2. The route between the communicating nodes does not change, it still try to send packets through the same route.

Question 3: Did you observe any additional traffic as compared to Step 3 above? How does the network react to the changes that take place at time 1.0 and time 1.2 now?

Yes, At time 1.0, the link 1-4 was broken, the network detected it and try another routes to send packets to the destination node. The route has been changed to 0-1-2-3-5. At time 1.2, the link 1-4 was repaired, and then change to the original route.

Question 4: How does this change affect the routing? Explain why.

At former scenario the cost between is all the same as 1. Since the cost between nodes 1 and 4 were changed to 3, the final cost of route 0-1-4-5 was changed to 5 which is bigger than the route 0-1-2-3-5 cost 4. Therefore, the network select the smaller link cost route to send packets.

Question 5: Describe what happens and deduce the effect of the line you just uncommented.

The final cost of 0-1-4-5 was changed to the route 0-1-2-3-5. Since the node be changed, it able to select multi-Path when paths have the same cost.

(*) Exercise 2: Setting up NS2 simulation for measuring TCP throughput

Exercise2.tcl

```
#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 &
   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 $n1 $n6 2.5Mb 40ms DropTail
$ns duplex-link $n2 $n3 10Mb 10ms DropTail
$ns duplex-link $n2 $n4 2.5Mb 40ms DropTail
$ns duplex-link $n4 $n5 10Mb 10ms DropTail
$ns duplex-link $n4 $n6 2.5Mb 40ms 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 $n1 $n2 orient up
$ns duplex-link-op $n3 $n2 orient right
$ns duplex-link-op $n2 $n4 orient right
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n1 $n6 orient right
$ns duplex-link-op $n6 $n7 orient right
$ns duplex-link-op $n4 $n6 orient down


#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 sink3 sink4 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_]
        set bw3 [$sink3 set bytes_]
        set bw4 [$sink4 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
        $sink3 set bytes_ 0
        $sink4 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
```

Throughput.plot

```
set xlabel "time[s]"
set ylabel "Throughput[Mbps]"
set key bel
plot  "tcp1.tr" u ($1):($2) t "TCP1" w lp, "tcp2.tr" u ($1):($2) t "TCP2" w lp

set term png
set output "TCPThroughput.png"
replot

pause -1
```

Exercise 3: Understanding IP Fragmentation

Question 1: Which data size has caused fragmentation and why? Which host/router has fragmented the original datagram? How many fragments have been created when data size is specified as 2000?

The data size 2000 caused fragmentation, because in the fragment offset filed, it not 0 since one segment contains data more than MTU.

router has fragmented the original datagram.

2 segments have been created when data size is specified as 2000.

Question 2: Did the reply from the destination 8.8.8.8. for 3500-byte data size also get fragmented? Why and why not?

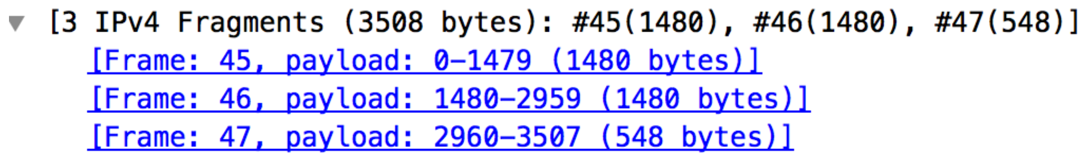
Yes, it is, the reason is the same as above.

Question 3: Give the ID, length, flag and offset values for all the fragments of the first packet sent by 192.168.1.103 with data size of 3500 bytes?

Fragments 1: ID 31355,length 1480, flag0x01 and offset 0

Fragment 2:ID 31355,length 1480, flag 0x01 and offset 1480

Fragment 3:ID 31355, length 548, flag 0x00 and offset 2960



```
▼ [3 IPv4 Fragments (3508 bytes): #45(1480), #46(1480), #47(548)]
     [Frame: 45, payload: 0-1479 (1480 bytes)]
     [Frame: 46, payload: 1480-2959 (1480 bytes)]
     [Frame: 47, payload: 2960-3507 (548 bytes)]
```

Question 4: Has fragmentation of fragments occurred when data of size 3500 bytes has been used? Why and why not?

No, the mtu for both are same, so no more frag occur.

Question 5: What will happen if for our example one fragment of the original datagram from 192.168.1.103 is lost?

It will lead to discard the whole original segment. But if It wok on the TCP protocol , TCP will request sender to resend the segment to recover this packet lost.