Lab 6

Jingshi Yang z5110579

Exercise 1:

Question 1

Node 0: node 1

Node 1: node 0, 2, 4

Node 2: node 1, 3

Node 3: node 2, 5

Node 4: node 1, 5

Node 5: node 4, 3

Routes:

0 -> 1 -> 4 -> 5

2 -> 3 -> 5

Yes, it changes over time because at time 1s, the link between node 1 and node 4 is gone.

Question 2

At time 1.0, the link between node 1 and node 4 is gone,

At time 1.2, the link between node 1 and node 4 is recovered and Node 1 starts to transmit packets to Node 4

Yes, no flow through node 1 to node 4 because of no link between the two nodes.

Question 3

I don't observe any additional traffic.

However, when node 1 find that it cannot transmit packets to node 4, it starts to transmit packets to node 2 -> node 3 -> node 5

At time 1.2s, when the traffic between node 1 and node 4 is solved, node 1 starts to transmit packets on its original way.

Question 4

Before uncommented this line, the route is 0 -> 1 -> 4 -> 5 and 2 -> 3 -> 5. After change, the route is 0 -> 1 ->2 -> 3 -> 5.

Because if the flow goes 0 -> 1 -> 4 -> 5, the cost is 5 which is higher compared with the cost 4 if the flow goes 0 -> 1 -> 2 -> 3 -> 5, the flow decides to go 0 -> 1 -> 2 -> 3 -> 5.

Question 5

There are two flows flowing into node 1, that means there are 3 paths now: 0 -> 1 -> 4 -> 5, 2 -> 3 -> 5, 2 -> 1 -> 4 -> 5.

The reason is that the default cost of each link is 1, hence the cost of the path 2 -> 3 -> 5 is 4 and the cost of the path 2 -> 1 -> 4 -> 5 is 4 as well, they are equal, hence there are two ways to 5 from 2. The effect of the uncommented line is allowing multiple flow flowing into Node 1.

Exercise 2:

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]
$ns trace $f1

set f2 [open tcp2.tr w]
$ns trace $f2

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

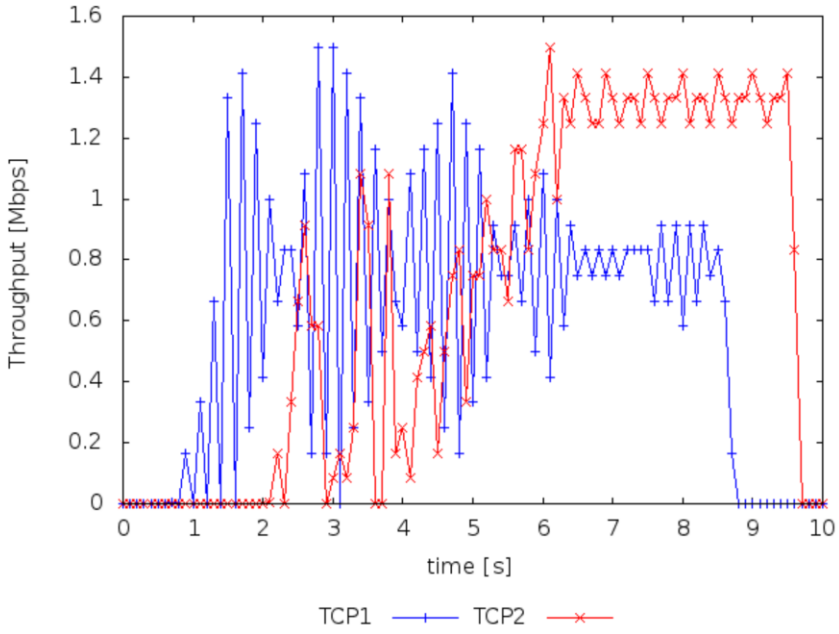
```
        set now [$ns now]
        #Calculate the bandwidth (in MBit/s) and write it to the files
        puts $f1 "$now [expr $bw1/$time*8/1000000]"
        puts $f2 "$now [expr $bw2/$time*8/1000000]"
        #Reset the bytes_ values on the traffic sinks
        $sink1 set bytes_ 0
        $sink2 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
```

Throughput.plot:

```
set xlabel "time [s]"

set ylabel "Throughput [Mbps]"

set key bel


plot "tcp1.tr" u ($1):($2) t "TCP1" w lp lt rgb "blue", "tcp2.tr" u ($1):($2) t
"TCP2" w lp lt rgb "red"

pause -1
```

Output:



Comment on graph:

1. From simulation, there are 4 flows:
    1. n3 -> n2 -> n4 -> n5
    2. n0 -> n1 -> n2 -> n4 -> n5
    3. n7 -> n6 -> n1 -> n0
    4. n7 -> n6 -> n1 -> n2 -> n3

2. The reason why throughput increases first then fluctuate is slow start.

3. For tcp2 (red), there are many packets dropped at 2.6s and one packet dropped at 3.6s from the simulation.

4. Two flows share the link n1-n6, two flows share the link n1-n2, and two flows share the link n2-n4 by observing the simulation.

5. After 6s, the sum of throughput tcp1 and tcp2 doesn't reach 2.5 and tcp1 (blue) has lower throughput than tcp2 (red), the reason is that

tcp1(blue) has more competitors (At n1, tcp1 has two competitors, at n2, it also has two competitors, but tcp2 only has two competitors at n2), it causes tcp1 has higher delay, hence tcp1 has lower throughput.

Exercise 3:

Question 1

When data size is 2000.

192.168.1.103

2 fragments have been created when data size is 2000.

Question 2

Yes. Because in reply message, there are three ipv4 fragments displayed.

Question 3

There are 3 fragments of the first packet with data size of 3500 bytes

The first one:

ID: 31355	length: 1500 bytes	flags: 1	offset: 0

The second one:

ID: 31355	length: 1500 bytes	flags: 1	offset: 185

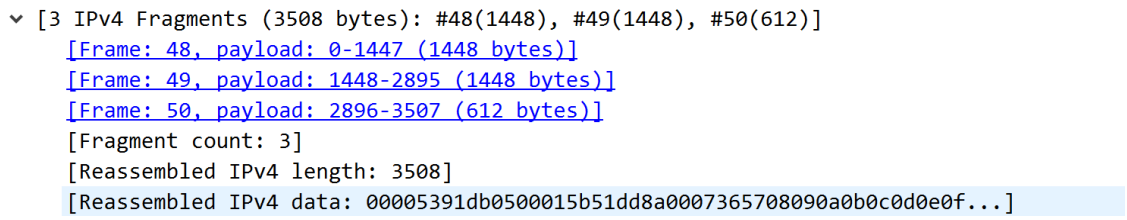
The third one:

ID: 31355	length: 568 bytes	flags: 0	offset: 370

Question 4

No. Fragmentation of fragments does not occur when the size is 3500 bytes

Because from the Wireshark, we can see that there is no fragmentation for a fragment



Question 5

Whole packets will be resent if one fragment is lost because there is no mechanism for a router to request a resend for one fragment.