Fundamentals of Computer Networks (CSE 534)

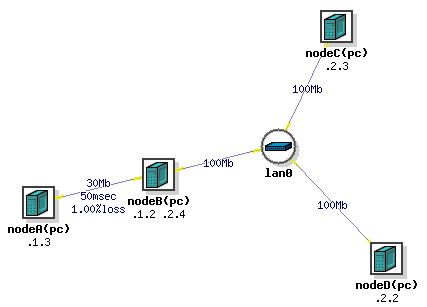
Project 1

Team:

Sivaram Mothiki (109396387)

**A1.** The NS script that defines the topology in the emulab example is

|  |
| --- |
| set ns [new Simulator]  source tb\_compat.tcl  #nodes  set nodeA [$ns node]  set nodeB [$ns node]  set nodeC [$ns node]  set nodeD [$ns node]  #lan hub  set link0 [$ns duplex-link $nodeB $nodeA 30Mb 50ms DropTail]  tb-set-link-loss $link0 0.01  set lan0 [$ns make-lan "$nodeD $nodeC $nodeB " 100Mb 0ms]  # Set the OS on a couple.  tb-set-node-os $nodeA FBSD-STD  tb-set-node-os $nodeB FBSD-STD  tb-set-node-os $nodeC FBSD-STD  tb-set-node-os $nodeD FBSD-STD  $ns rtproto Static  # Go!  $ns run |



We have removed the code that is meant to define Lan 0 and replaced it with a separate nodeE.

Such that nodeE can connect to nodeA,NodeB,nodeC with link 1,link2 ,link 3

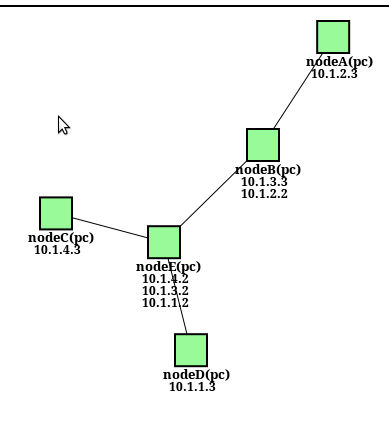
Below is the modified NS script

# This is a simple ns script. Comments start with #.

|  |
| --- |
| set ns [new Simulator]  source tb\_compat.tcl  set nodeA [$ns node]  set nodeB [$ns node]  set nodeC [$ns node]  set nodeD [$ns node]  set nodeE [$ns node  set link0 [$ns duplex-link $nodeE $nodeB 100Mb 0ms DropTail]  set link1 [$ns duplex-link $nodeE $nodeC 100Mb 0ms DropTail]  set link2 [$ns duplex-link $nodeE $nodeD 100Mb 0ms DropTail]  set link3 [$ns duplex-link $nodeB $nodeA 30Mb 50ms DropTail]  # Set the OS on a couple.  tb-set-node-os $nodeA FBSD-STD  tb-set-node-os $nodeB FBSD-STD  tb-set-node-os $nodeC FBSD-STD  tb-set-node-os $nodeD FBSD-STD  tb-set-node-os $nodeE FBSD-STD  $ns rtproto Static  # Go!  $ns run |

Every Node is able to ping every other node .

IP of A 10.1.2.3

IP of B 10.1.2.2 ,10.1.3.3

IP of C 10.1.4.3

IP of D 10.1.3.3

IP of E 10.1.4.3 , 10.1.3.2 ,10.1.1.2

**B1)**

nodeE is connected to nodes A,B,C . We have to configure nodeE as a router

Ns file for defining the topology is

Now we have to install click on Node E so we have changed node E operating system to Ubuntu Linux as click needs latest GCC compiler which is not present in FREE BSD.

We have downloaded click software . Click is used to configure any host act as a router to support networking functionality .

Instructions to install click .

|  |
| --- |
| 1) First Downloaded click from <http://www.read.cs.ucla.edu/click/download> to /groups/CSE534/<groupname>/ (wget //www.read.cs.ucla.edu/click/click-2.0.1.tar.gz)  2) Extract the tar file and cd to the click extracted folder ($ tar xzf click-\clickversion.tar.gz)  3) Configure the software ./configure --disable-linuxmodule --enable-local  4) sudo make -j2 and then sudo make install.  5) To test for installation success run userlevel/click conf/test.click |

NS Script for Defining topology

set ns [new Simulator]

source tb\_compat.tcl

set nodeA [$ns node]

set nodeB [$ns node]

set nodeC [$ns node]

set nodeD [$ns node]

set nodeE [$ns node]

set link0 [$ns duplex-link $nodeE $nodeB 100Mb 0ms DropTail]

set link1 [$ns duplex-link $nodeE $nodeC 100Mb 0ms DropTail]

set link2 [$ns duplex-link $nodeE $nodeD 100Mb 0ms DropTail]

set link3 [$ns duplex-link $nodeB $nodeA 30Mb 50ms DropTail]

# Set the OS on a couple.

tb-set-node-os $nodeA FBSD-STD

tb-set-node-os $nodeB FBSD-STD

tb-set-node-os $nodeC FBSD-STD

tb-set-node-os $nodeD FBSD-STD

tb-set-node-os $nodeE UBUNTU10-STD

$ns rtproto Static

# Go!

$ns run

|  |
| --- |
| S0::FromDevice(eth1);  S1::FromDevice(eth3);  S2::FromDevice(eth4);  Sw0:: EtherSwitch();  q0::Queue ;  q1::Queue ;  q2::Queue ;  D0:: ToDevice(eth1);  D1:: ToDevice(eth3);  D2:: ToDevice(eth4);  S0 -> [0] Sw0 [0] -> q0 -> D0;  S1 -> [1] Sw0 [1] -> q1 -> D1;  S2 -> [2] Sw0 [2] -> q2 -> D2; |

Click Script to configure Node E as a router

We have created 3 queues for e Ethernet connections and configured from and to ports .

**B2**

We keep the above click script as it is and change the network topology divide the hosts I to subnets

|  |
| --- |
| set ns [new Simulator]  source tb\_compat.tcl  # Nodes  set nodeA [$ns node]  set nodeB [$ns node]  set nodeC [$ns node]  set nodeD [$ns node]  set nodeE [$ns node]  tb-set-node-os $nodeA FBSD-STD  tb-set-node-os $nodeB FBSD -STD  tb-set-node-os $nodeC FBSD -STD  tb-set-node-os $nodeD FBSD -STD  tb-set-node-os $nodeE UBUNTU10-STD  # Links  set link0 [$ns duplex-link $nodeA $nodeB 100mb 0ms DropTail]  set link1 [$ns duplex-link $nodeB $nodeE 100mb 0ms DropTail]  set link2 [$ns duplex-link $nodeC $nodeE 100mb 0ms DropTail]  set link3 [$ns duplex-link $nodeD $nodeE 100mb 0ms DropTail]  tb-set-link-loss $link0 0.01  tb-set-ip-link $nodeE $link1 10.11.12.13  tb-set-ip-link $nodeE $link2 10.11.12.14  tb-set-ip-link $nodeE $link3 10.11.12.15  tb-set-ip-link $nodeA $link0 10.1.20.1  tb-set-ip-link $nodeB $link0 10.1.20.2  tb-set-ip-link $nodeB $link1 10.1.1.1  tb-set-ip-link $nodeC $link2 10.1.1.2  tb-set-ip-link $nodeD $link3 10.1.1.3  tb-set-netmask $link0 "255.255.255.0"  tb-set-netmask $link1 "255.255.255.0"  tb-set-netmask $link1 "255.255.255.0"  tb-set-netmask $link2"255.255.255.0"  # $ns rtproto Static  $ns run |

A network card only receives packets that are meant for it , but if we set the host to promiscuous mode . The host accepts all the packets that want to flow through it .

NodeE : sudo ifconfig eth2 promisc

NodeE : sudo ifconfig eth3 promisc

NodeE : sudo ifconfig eth4 promisc

Nodes belong to same subnet are able to ping . I’e B,C,D are reachable to each other .

A has information about B only and has no information about C,D and vice versa .

Inorder to make A-D , A-C connection we have to add routing table entries into A for the path A-D

We add route to A-D using route add command.

**Node A: sudo route add -net 10.1.1.0/24 10.1.20.1**

Node B should forward packets to D and D should be able to get back to A so we add route to D as well

**Node D: sudo route add -net 10.1.20.0/24 10.1.1.3**

A similar approach can be followed to C as well.

**C)**

**Set1 :**  For this experiment we have considered the topology given in the emulab tutorial with Lan0 Hub.

In this experiment we made Node A as both ttcp sender and receiver.

Below are the details of the

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **protcol** | **Buffer Length** | **N** | **time** | **Throughput** |
|  | UDP | 8192 | 200000 | 0.52 | 297195.59 KB/sec |
|  | TCP | 8192 | 200000 | 10.58 | 151224.15 KB/sec |

**Set2:** Now We configure Node A as sender and Node C as Receiver.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **protcol** | **Buffer Length** | **N** | **time** | **Throughput** |
| UDP | 8192 | 2000 | 1.49 | 10709.97 KB/sec |
| TCP | 8192 | 2000 | 163.76 | 97.91 KB/sec |

**Set3:**  Now we modify the topology by replacing lan with node E. We configure A as ttcp sender and Node C as ttcp Receiver

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **protcol** | **Buffer Length** | **N** | **time** | **Throughput** |
| UDP | 8192 | 20000 | 45.67 | 10752.95 KB/sec |
| UDP | 8192 | 20000 | 163.76 | 3503.33 KB/sec |

**Set4 :** Now Node E act as an Ethernet switch for the usecase set 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **protcol** | **Buffer Length** | **N** | **time** | **Throughput** |
| UDP | 8192 | 20000 | 15.34 | 10430.24 KB/sec |
| TCP | 8192 | 20000 | 1350.73 | 118.45 KB/sec |

A sample example for Set 3:

**TCP**

From Node A

ttcp -t -s -n 20000 10.1.2.3

ttcp-t: buflen=8192, nbuf=20000, align=16384/0, port=5001 tcp -> 10.1.2.3

ttcp-t: socket

ttcp-t: connect

ttcp-t: 163840000 bytes in 45.65 real seconds = 3504.75 KB/sec +++

ttcp-t: 20000 I/O calls, msec/call = 2.34, calls/sec = 438.09

ttcp-t: 0.0user 1.9sys 0:45real 4% 15i+222d 242maxrss 0+2pf 75412+9csw

**Node C**:

ttcp -r -s

ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001 tcp

ttcp-r: socket

ttcp-r: accept from 10.1.4.3

ttcp-r: 163840000 bytes in 45.67 real seconds = 3503.33 KB/sec +++

ttcp-r: 113143 I/O calls, msec/call = 0.41, calls/sec = 2477.36

ttcp-r: 0.0user 0.4sys 0:45real 1% 15i+222d 232maxrss 0+1pf 113141+9csw

**UDP**

**From A**

ttcp -t -u -s -n 20000 10.1.2.3

ttcp-t: buflen=8192, nbuf=20000, align=16384/0, port=5001 udp -> 10.1.2.3

ttcp-t: socket

ttcp-t: 163840000 bytes in 14.88 real seconds = 10752.95 KB/sec +++

ttcp-t: 20502 I/O calls, msec/call = 0.74, calls/sec = 1377.86

ttcp-t: 0.0user 1.3sys 0:14real 8% 16i+224d 242maxrss 0+2pf 496+0csw

**From C**

ttcp -r -u -s

ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001 udp

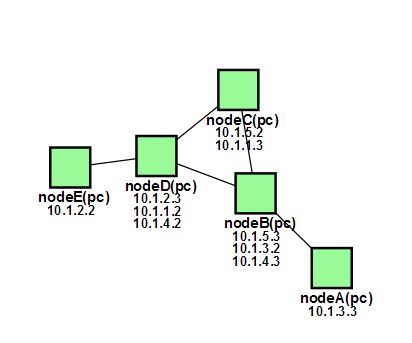
ttcp-r: socket

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **protcol** | **set1** | **set2** | **set3** | **set4** |
| UDP | 297195.59 KB/sec | 10709.97 KB/sec | 10752.95 KB/sec | 10430.24 KB/sec |
| TCP | 151224.15 KB/sec | 97.91 KB/sec | 3503.33 KB/sec | 118.45 KB/sec |

­­­

**D1)**

The topological diagram is shown below .



We used all nodes that were booted with FreeBSD because free bsd OS has all the network utilities installed .

Instructions for installing RIP according to http://www.labbook.cs.purdue.edu/ch/emulab/exp17.3.php.

We have to log into each node and run the following two commands

sudo sysctl -w net.inet.ip.forwarding=1

sudo routed -s -P rdisc\_interval=15

After some time each node shares the routing information of each other node

Routing table Entries for Node E

|  |
| --- |
| Destination Gateway Flags Refs Use Netif Expire  default 155.98.36.1 UGSc 8 193 fxp0  10.1.1/24 link#5 UC 1 0 fxp4  10.1.1.3 link#5 UHLW 1 411 fxp4  10.1.2/24 link#3 UC 0 0 fxp2  10.1.3/24 10.1.4.3 UGc 0 0 fxp1  10.1.4/24 link#2 UC 1 0 fxp1  10.1.4.3 link#2 UHLW 1 411 fxp1  10.1.5/24 10.1.1.3 UGc 0 0 fxp4  127.0.0.1 127.0.0.1 UH 11 127 lo0 |

Routing table entries of B

|  |
| --- |
| Internet:  Destination Gateway Flags Refs Use Netif Expire  default 155.98.36.1 UGSc 8 162 fxp0  10.1.1/24 10.1.5.2 UGc 0 0 fxp4  10.1.2/24 10.1.4.2 UGc 0 0 fxp2  10.1.3/24 link#4 UC 0 0 fxp3  10.1.4/24 link#3 UC 1 0 fxp2  10.1.4.2 link#3 UHLW 1 411 fxp2  10.1.5/24 link#5 UC 1 0 fxp4  10.1.5.2 link#5 UHLW 1 411 fxp4  127.0.0.1 127.0.0.1 UH 11 125 lo0 |

Routing table entries of A

|  |
| --- |
| Internet:  Destination Gateway Flags Refs Use Netif Expire  default 155.98.36.1 UGSc 8 178 fxp0  10.1.1/24 10.1.3.2 UGc 0 0 fxp1  10.1.2/24 10.1.3.2 UGc 0 0 fxp1  10.1.3/24 link#2 UC 1 0 fxp1  10.1.3.2 link#2 UHLW 3 411 fxp1  10.1.4/23 10.1.3.2 UGc 0 0 fxp1  127.0.0.1 127.0.0.1 UH 11 127 lo010.1.5.2  127.0.0.1 127.0.0.1 UH 11 125 lo0 |

Trace route to A-E path from A

> traceroute 10.1.2.2

traceroute to 10.1.2.2 (10.1.2.2), 64 hops max, 44 byte packets

1 nodeB-link4 (10.1.3.2) 0.348 ms 0.159 ms 0.138 ms

2 nodeD-link0 (10.1.4.2) 0.367 ms 0.248 ms 0.238 ms

3 nodeE-link3 (10.1.2.2) 0.505 ms 0.345 ms 0.336 ms

Traceroute of E-A path

traceroute 10.1.3.3

traceroute to 10.1.3.3 (10.1.3.3), 64 hops max, 44 byte packets

1 nodeD-link3 (10.1.2.3) 0.298 ms 0.171 ms 0.268 ms

2 nodeB-link0 (10.1.4.3) 0.452 ms 0.254 ms 0.255 ms

3 nodeA-link4 (10.1.3.3) 0.380 ms 0.345 ms 0.337 ms

Ping of A from E

> ping -c 5 10.1.3.3

PING 10.1.3.3 (10.1.3.3): 56 data bytes

64 bytes from 10.1.3.3: icmp\_seq=0 ttl=62 time=0.474 ms

64 bytes from 10.1.3.3: icmp\_seq=1 ttl=62 time=0.394 ms

64 bytes from 10.1.3.3: icmp\_seq=2 ttl=62 time=0.421 ms

64 bytes from 10.1.3.3: icmp\_seq=3 ttl=62 time=0.434 ms

64 bytes from 10.1.3.3: icmp\_seq=4 ttl=62 time=0.458 ms

--- 10.1.3.3 ping statistics ---

5 packets transmitted, 5 packets received, 0% packet loss

round-trip min/avg/max/stddev = 0.394/0.436/0.474/0.028 ms

Below are the details of NS file for D1 .We used protocol manual for this experiment as we are defining our own RIP protocol.

set ns [new Simulator]

source tb\_compat.tcl

set nodeA [$ns node]

set nodeB [$ns node]

set nodeC [$ns node]

set nodeD [$ns node]

set nodeE [$ns node]

set link0 [$ns duplex-link $nodeD $nodeB 100Mb 0ms DropTail]

set link1 [$ns duplex-link $nodeC $nodeB 100Mb 0ms DropTail]

set link2 [$ns duplex-link $nodeD $nodeC 100Mb 0ms DropTail]

set link3 [$ns duplex-link $nodeE $nodeD 100Mb 0ms DropTail]

set link4 [$ns duplex-link $nodeB $nodeA 100Mb 0ms DropTail]

# Set the OS on a couple.

tb-set-node-os $nodeA FBSD-STD

tb-set-node-os $nodeB FBSD-STD

tb-set-node-os $nodeC FBSD-STD

tb-set-node-os $nodeD FBSD-STD

tb-set-node-os $nodeE FBSD-STD

$ns rtproto Manual

# Go!

$ns run

**D2)**

Link B-D was put down now the expected behavior is for path A-E the original path is

A-B-D-E Now It has to take the path A-B-C-D-E.

We have taken down the link 0 by adding the following to lines to the above script

**$ns at 600.0 "link0 down"**

**$ns at 1800.0 "link0 up"**

set ns [new Simulator]

source tb\_compat.tcl

set nodeA [$ns node]

set nodeB [$ns node]

set nodeC [$ns node]

set nodeD [$ns node]

set nodeE [$ns node]

set link0 [$ns duplex-link $nodeD $nodeB 100Mb 0ms DropTail]

set link1 [$ns duplex-link $nodeC $nodeB 100Mb 0ms DropTail]

set link2 [$ns duplex-link $nodeD $nodeC 100Mb 0ms DropTail]

set link3 [$ns duplex-link $nodeE $nodeD 100Mb 0ms DropTail]

set link4 [$ns duplex-link $nodeB $nodeA 100Mb 0ms DropTail]

# Set the OS on a couple.

tb-set-node-os $nodeA FBSD-STD

tb-set-node-os $nodeB FBSD-STD

tb-set-node-os $nodeC FBSD-STD

tb-set-node-os $nodeD FBSD-STD

tb-set-node-os $nodeE FBSD-STD

**$ns at 600.0 "link0 down"**

**$ns at 1800.0 "link0 up"**

$ns rtproto Manual

# Go!

$ns run

After 10 minutes the link B-D goes down and after 30 minutes it again come up.

Routing tables of A after B-D link down are

> netstat -r -n

Routing tables

Internet:

Destination Gateway Flags Refs Use Netif Expire

default 155.98.36.1 UGSc 5 77 em0

10.1.1/24 10.1.3.2 UGc 0 0 em2

10.1.2/24 10.1.3.2 UGc 0 219 em2

10.1.3/24 link#3 UC 1 0 em2

10.1.3.2 00:d0:b7:9a:4a:cb UHLW 3 411 em2 896

10.1.4/23 10.1.3.2 UGc 0 0 em2

127.0.0.1 127.0.0.1 UH 11 133 lo0

Traceroute from A to E

> traceroute 10.1.2.2

traceroute to 10.1.2.2 (10.1.2.2), 64 hops max, 44 byte packets

1 nodeB-link4 (10.1.3.2) 0.227 ms 0.242 ms 0.167 ms

2 \* \* \*

3 \* \* \*

\*\*\*\*\*

14 \* \* \*

15 nodeE-link3 (10.1.2.2) 0.744 ms 0.706 ms 0.645 ms

Traceroute has taken lot of time reaching E and once the route is adjusted after 20 mins

The same Traceroute command became quick

**> traceroute 10.1.2.2**

**traceroute to 10.1.2.2 (10.1.2.2), 64 hops max, 44 byte packets**

**1 nodeB-link4 (10.1.3.2) 0.209 ms 0.233 ms 0.172 ms**

**2 nodeC-link1 (10.1.5.2) 0.282 ms 0.346 ms 0.270 ms**

**3 nodeD-link2 (10.1.1.2) 0.392 ms 0.360 ms 0.397 ms**

**4 nodeE-link3 (10.1.2.2) 0.644 ms 0.740 ms 0.648 ms**

The link was up after 20 mins and RIP got adjusted according to the previous topology .

**D3)**

The D-E link was brought down using the same approach

Adding the following single line in the script file **$ns at 600.0 "link3 down"**

set ns [new Simulator]

source tb\_compat.tcl

set nodeA [$ns node]

set nodeB [$ns node]

set nodeC [$ns node]

set nodeD [$ns node]

set nodeE [$ns node]

set link0 [$ns duplex-link $nodeD $nodeB 100Mb 0ms DropTail]

set link1 [$ns duplex-link $nodeC $nodeB 100Mb 0ms DropTail]

set link2 [$ns duplex-link $nodeD $nodeC 100Mb 0ms DropTail]

set link3 [$ns duplex-link $nodeE $nodeD 100Mb 0ms DropTail]

set link4 [$ns duplex-link $nodeB $nodeA 100Mb 0ms DropTail]

# Set the OS on a couple.

tb-set-node-os $nodeA FBSD-STD

tb-set-node-os $nodeB FBSD-STD

tb-set-node-os $nodeC FBSD-STD

tb-set-node-os $nodeD FBSD-STD

tb-set-node-os $nodeE FBSD-STD

**$ns at 600.0 "link3 down"**

$ns rtproto Manual

Below are the details of routing table entries for every 5 minute intervals

netstat -r -n

Routing tables

Internet:

**Destination Gateway Flags Refs Use Netif Expire**

default 155.98.36.1 UGSc 4 180 fxp0

10.1.1/24 link#2 UC 1 0 fxp4

10.1.1.3 00:03:47:94:ca:26 UHLW 1 411 fxp4 846

10.1.2/24 link#3 UGc 0 0 fxp4

10.1.2.2 00:03:47:94:ca:26 UHLW 1 411 fxp4 846- Link E entry

10.1.3/24 link#0 UGc 0 0 fxp1

10.1.3.2 00:02:b3:3f:74:bd UHLW 0 411 fxp1 846

10.1.4/23 10.1.2.3 UC 1 0 fxp1

127.0.0.1 127.0.0.1 UH 11 125 lo0

**Next 5 minutes**

Internet:

**Destination Gateway Flags Refs Use Netif Expire**

default 155.98.36.1 UGSc 6 25 fxp0

10.1.1/24 link#2 UC 1 0 fxp4

10.1.1.3 00:03:47:94:ca:26 UHLW 1 411 fxp4 232

10.1.2/24 link#3 UGc 0 0 fxp4

10.1.2.2 00:03:47:94:ca:26 UHLW 1 411 fxp4 232

10.1.3/24 link#0 UGc 0 0 fxp1

10.1.3.2 00:02:b3:3f:74:bd UHLW 0 411 fxp1 232

10.1.4/23 10.1.2.3 UC 1 0 fxp1

127.0.0.1 127.0.0.1 UH 11 140 lo0

**Next 5minutes**

Internet:

**Destination Gateway Flags Refs Use Netif Expire**

default 155.98.36.1 UGSc 6 37 fxp0

10.1.1/24 link#2 UC 1 0 fxp4

10.1.1.3 00:03:47:94:ca:26 UHLW 1 411 fxp4 232

10.1.2/24 link#3 UGc 0 0 fxp4

10.1.2.2 00:03:47:94:ca:26 UHLW 1 411 fxp4 232

10.1.3/24 link#0 UGc 0 0 fxp1

10.1.3.2 00:02:b3:3f:74:bd UHLW 0 414 fxp1 232

10.1.4/23 10.1.2.3 UC 1 0 fxp1

127.0.0.1 127.0.0.1 UH 11 130 lo0

**Next 5 minutes**

Internet:

**Destination Gateway Flags Refs Use Netif Expire**

default 155.98.36.1 UGSc 6 57 fxp0

10.1.1/24 link#2 UC 1 0 fxp4

10.1.2/24 link#3 UGc 0 0 fxp4

10.1.2.2 link#3 UHLW 1 411 fxp4

10.1.3/24 link#0 UGc 0 0 fxp1

10.1.4/23 10.1.2.3 UC 0 0 fxp1

By analyzing the above tables we can study the evolution of RIP for D