Rip Implementation in ns2

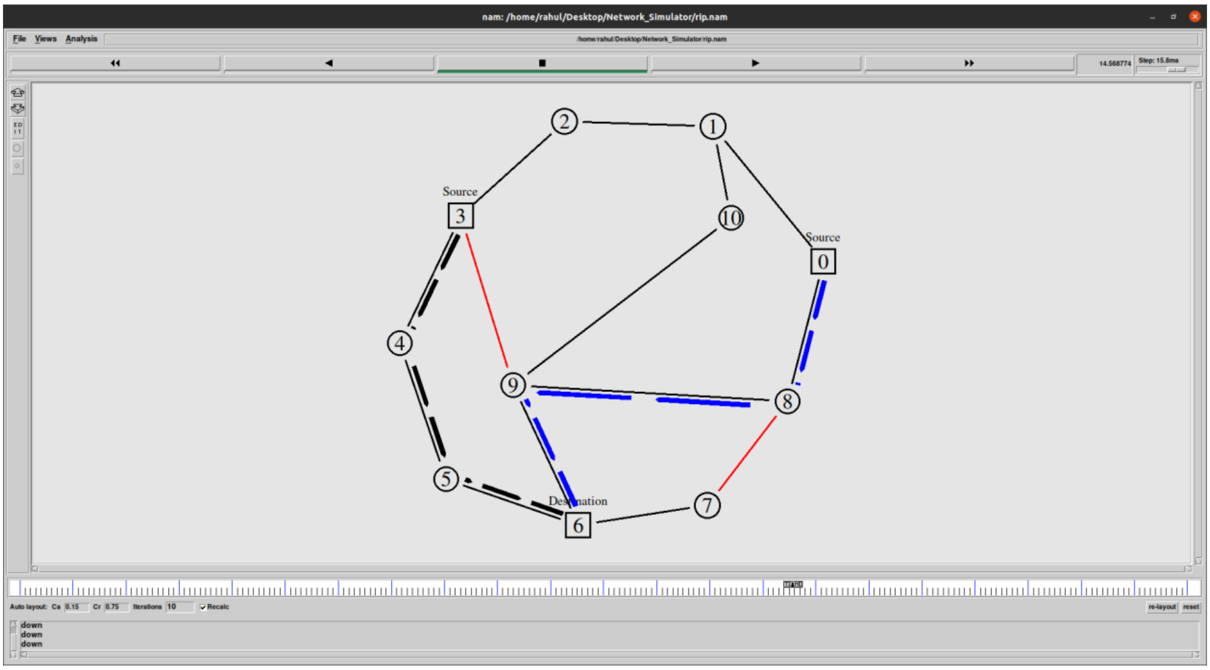
Group - 2

1. Rahul Katinni	S20200010091
2. Santosh Kakkera	S20200010082
3. Maneesh Sharma	S20200010075
4. Tejaditya Nachiketa	S20200010153

Implementation

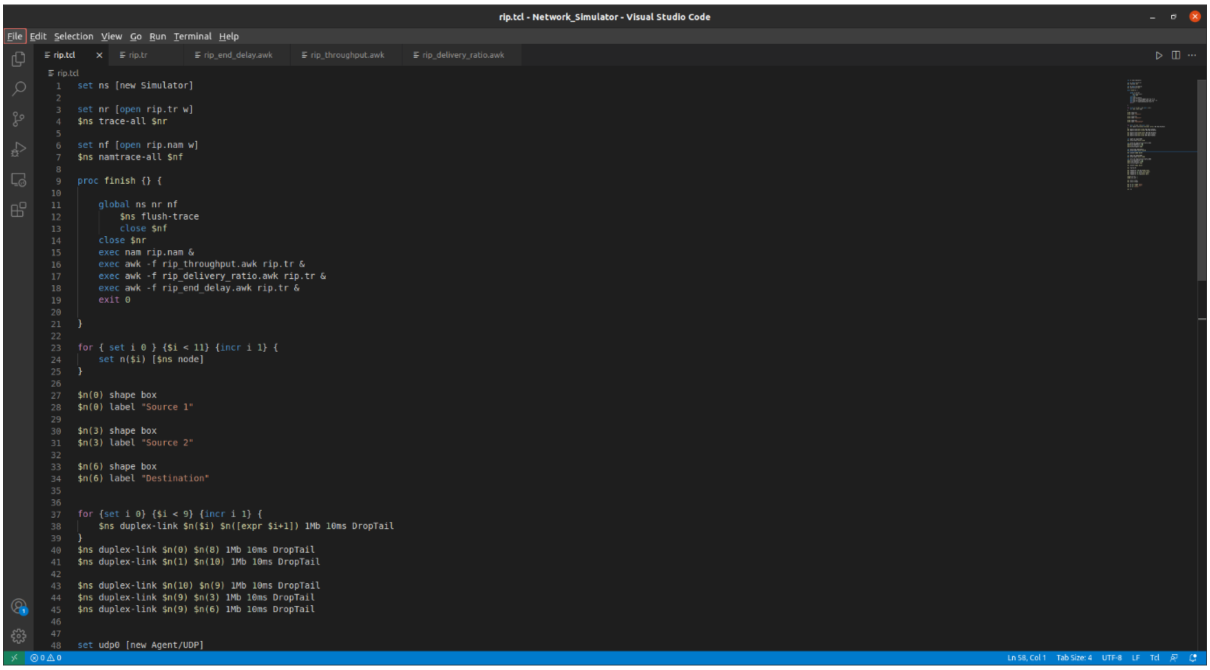
We have implemented RIP protocol which uses distance vector routing in the scenario where there are two source nodes transmitting data to the same destination node. We have also considered scenarios where a link gets dropped in between transmission and how the algorithm will react to these scenarios was simulated on NS2.

Topology

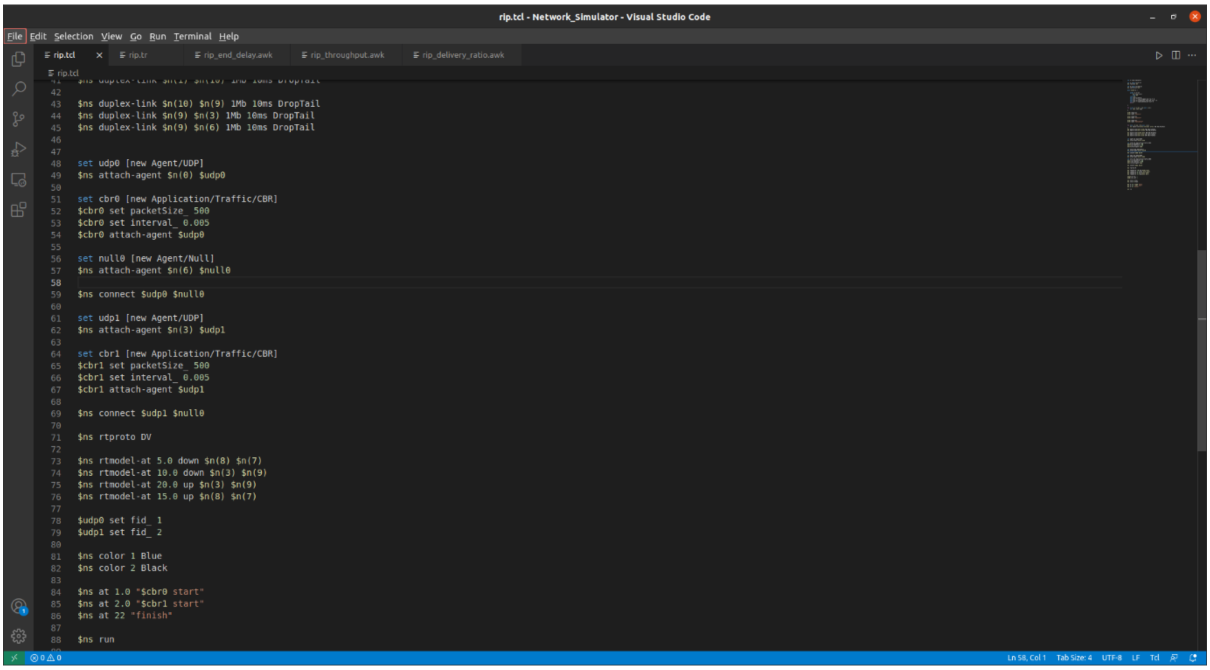

nam: /home/rahul/Desktop/Network_Simulator/rip.nam
File
Views
Analysis
/home/rahul/Desktop/Network_Simulator/rip.nam
14.568774
Step: 15.8ms
Source
Source
Destination
0
1
2
3
4
5
6
7
8
9
10
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset

```
down
down
down
```

Code

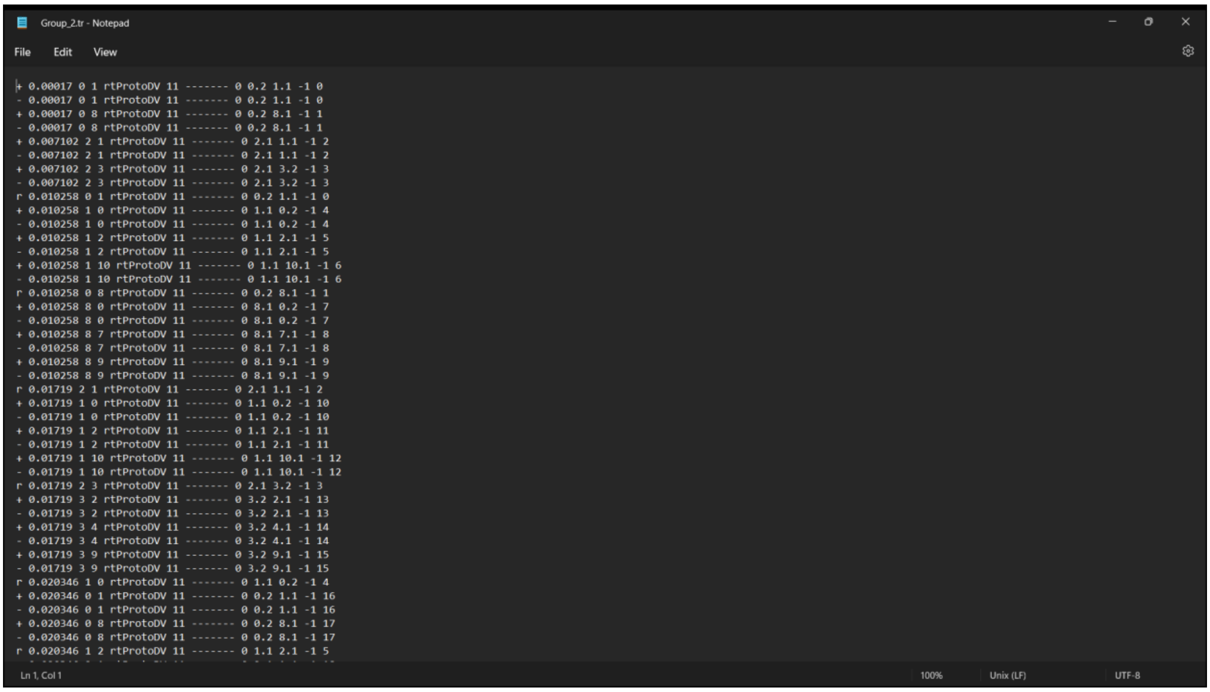

rip.tcl - Network_Simulator - Visual Studio Code
File Edit Selection View Go Run Terminal Help
rip.tcl
rip.tr
rip_end_delay.awk
rip_throughput.awk
rip_delivery_ratio.awk
rip.tcl
Ln 58, Col 1 Tab Size: 4 UTF-8 LF Tcl

```
set ns [new Simulator]

set nr [open rip.tr w]
$ns trace-all $nr

set nf [open rip.nam w]
$ns namtrace-all $nf

proc finish {} {

    global ns nr nf
        $ns flush-trace
        close $nf
    close $nr
    exec nam rip.nam &
    exec awk -f rip_throughput.awk rip.tr &
    exec awk -f rip_delivery_ratio.awk rip.tr &
    exec awk -f rip_end_delay.awk rip.tr &
    exit 0

}

for { set i 0 } {$i < 11} {incr i 1} {
    set n($i) [$ns node]
}

$n(0) shape box
$n(0) label "Source 1"

$n(3) shape box
$n(3) label "Source 2"

$n(6) shape box
$n(6) label "Destination"


for {set i 0} {$i < 9} {incr i 1} {
    $ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail
}
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail

$ns duplex-link $n(10) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(3) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(6) 1Mb 10ms DropTail


set udp0 [new Agent/UDP]
```


rip.tcl - Network_Simulator - Visual Studio Code
File Edit Selection View Go Run Terminal Help
rip.tcl
rip.tr
rip_end_delay.awk
rip_throughput.awk
rip_delivery_ratio.awk
Ln 58, Col 1 Tab Size: 4 UTF-8 LF Tcl

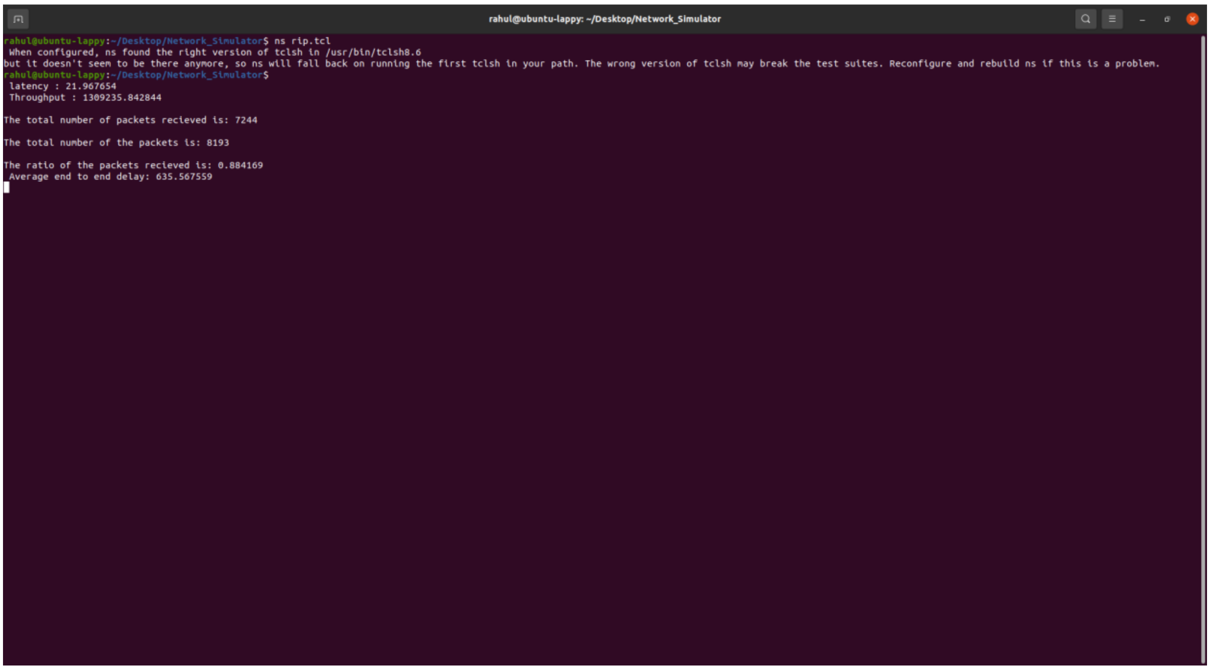
```
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail

$ns duplex-link $n(10) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(3) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(6) 1Mb 10ms DropTail


set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(6) $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]
$ns attach-agent $n(3) $udp1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

$ns connect $udp1 $null0

$ns rtproto DV

$ns rtmodel-at 5.0 down $n(8) $n(7)
$ns rtmodel-at 10.0 down $n(3) $n(9)
$ns rtmodel-at 20.0 up $n(3) $n(9)
$ns rtmodel-at 15.0 up $n(8) $n(7)

$udp0 set fid_ 1
$udp1 set fid_ 2

$ns color 1 Blue
$ns color 2 Black

$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 22 "finish"

$ns run
```

Trace file



```
+ 0.00017 0 1 rtProtoDV 11 ------- 0 0.2 1.1 -1 0
- 0.00017 0 1 rtProtoDV 11 ------- 0 0.2 1.1 -1 0
+ 0.00017 0 8 rtProtoDV 11 ------- 0 0.2 8.1 -1 1
- 0.00017 0 8 rtProtoDV 11 ------- 0 0.2 8.1 -1 1
+ 0.007102 2 1 rtProtoDV 11 ------- 0 2.1 1.1 -1 2
- 0.007102 2 1 rtProtoDV 11 ------- 0 2.1 1.1 -1 2
+ 0.007102 2 3 rtProtoDV 11 ------- 0 2.1 3.2 -1 3
- 0.007102 2 3 rtProtoDV 11 ------- 0 2.1 3.2 -1 3
r 0.010258 0 1 rtProtoDV 11 ------- 0 0.2 1.1 -1 0
+ 0.010258 1 0 rtProtoDV 11 ------- 0 1.1 0.2 -1 4
- 0.010258 1 0 rtProtoDV 11 ------- 0 1.1 0.2 -1 4
+ 0.010258 1 2 rtProtoDV 11 ------- 0 1.1 2.1 -1 5
- 0.010258 1 2 rtProtoDV 11 ------- 0 1.1 2.1 -1 5
+ 0.010258 1 10 rtProtoDV 11 ------- 0 1.1 10.1 -1 6
- 0.010258 1 10 rtProtoDV 11 ------- 0 1.1 10.1 -1 6
r 0.010258 0 8 rtProtoDV 11 ------- 0 0.2 8.1 -1 1
+ 0.010258 8 0 rtProtoDV 11 ------- 0 8.1 0.2 -1 7
- 0.010258 8 0 rtProtoDV 11 ------- 0 8.1 0.2 -1 7
+ 0.010258 8 7 rtProtoDV 11 ------- 0 8.1 7.1 -1 8
- 0.010258 8 7 rtProtoDV 11 ------- 0 8.1 7.1 -1 8
+ 0.010258 8 9 rtProtoDV 11 ------- 0 8.1 9.1 -1 9
- 0.010258 8 9 rtProtoDV 11 ------- 0 8.1 9.1 -1 9
r 0.01719 2 1 rtProtoDV 11 ------- 0 2.1 1.1 -1 2
+ 0.01719 1 0 rtProtoDV 11 ------- 0 1.1 0.2 -1 10
- 0.01719 1 0 rtProtoDV 11 ------- 0 1.1 0.2 -1 10
+ 0.01719 1 2 rtProtoDV 11 ------- 0 1.1 2.1 -1 11
- 0.01719 1 2 rtProtoDV 11 ------- 0 1.1 2.1 -1 11
+ 0.01719 1 10 rtProtoDV 11 ------- 0 1.1 10.1 -1 12
- 0.01719 1 10 rtProtoDV 11 ------- 0 1.1 10.1 -1 12
r 0.01719 2 3 rtProtoDV 11 ------- 0 2.1 3.2 -1 3
+ 0.01719 3 2 rtProtoDV 11 ------- 0 3.2 2.1 -1 13
- 0.01719 3 2 rtProtoDV 11 ------- 0 3.2 2.1 -1 13
+ 0.01719 3 4 rtProtoDV 11 ------- 0 3.2 4.1 -1 14
- 0.01719 3 4 rtProtoDV 11 ------- 0 3.2 4.1 -1 14
+ 0.01719 3 9 rtProtoDV 11 ------- 0 3.2 9.1 -1 15
- 0.01719 3 9 rtProtoDV 11 ------- 0 3.2 9.1 -1 15
r 0.020346 1 0 rtProtoDV 11 ------- 0 1.1 0.2 -1 4
+ 0.020346 0 1 rtProtoDV 11 ------- 0 0.2 1.1 -1 16
- 0.020346 0 1 rtProtoDV 11 ------- 0 0.2 1.1 -1 16
+ 0.020346 0 8 rtProtoDV 11 ------- 0 0.2 8.1 -1 17
- 0.020346 0 8 rtProtoDV 11 ------- 0 0.2 8.1 -1 17
r 0.020346 1 2 rtProtoDV 11 ------- 0 1.1 2.1 -1 5
```

Outputs



```
rahul@ubuntu-lappy:~/Desktop/Network_Simulator$ ns rip.tcl
 When configured, ns found the right version of tclsh in /usr/bin/tclsh8.6
but it doesn't seem to be there anymore, so ns will fall back on running the first tclsh in your path. The wrong version of tclsh may break the test suites. Reconfigure and rebuild ns if this is a problem.
rahul@ubuntu-lappy:~/Desktop/Network_Simulator$
 latency : 21.967654
 Throughput : 1309235.842844

The total number of packets recieved is: 7244

The total number of the packets is: 8193

The ratio of the packets recieved is: 0.884169
 Average end to end delay: 635.567559
```

Observations and Challenges

While implementing RIP in ns2 we were able to observe how queuing happens at nodes and how the DV algorithm communicates and routes the data between nodes and at what point the packet dropping starts due to full queue.

The challenges we faced were designing the nodes and proper topology to implement all the scenarios and then extracting data from trace file and also calculation of the end - to - end delay was a bit confusing.