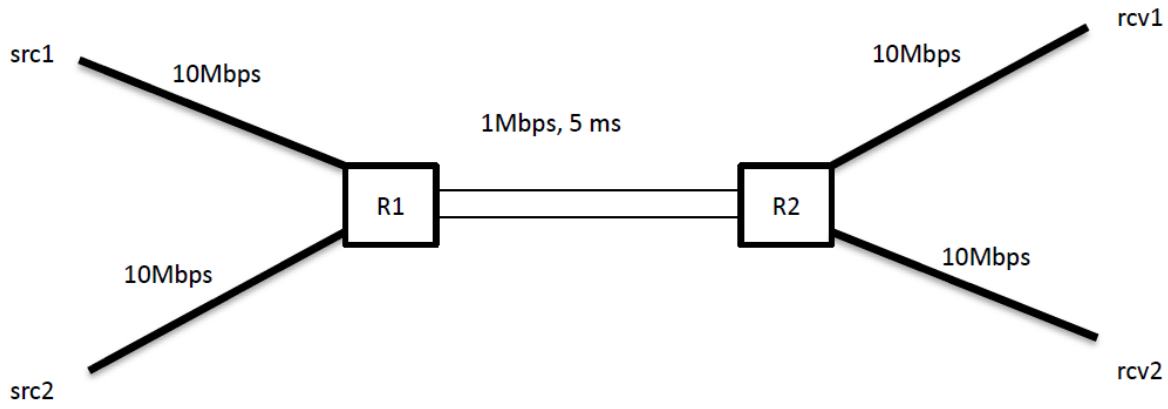
Programming Assignment 5

Chenjie Luo (UIN: 324007289)
Yu-Wen Chen(UIN: 227009499)

a. Description

The project is divided into two parts: Chenjie is responsible for writing the script for simulation, and Yu-Wen is responsible for writing the report based on the simulation result.

b. Test Setup



We based on the handout to implement the simulation environment, and the configurations are as below:

1. Two routers R1, R2 connected with a duplex link with 1Mbps speed, 5ms latency and droptail queue.
2. Two TCP connections: connection between src1 and rcv1 and connection between src2 and rcv2. Each TCP connection's application sender is FTP.
3. Three different test cases:
   (a) The delays of link src1-R1 and link R2-rcv1 are set to 5ms; the delays of link src2-R1 and link R2-rcv2 are set to 12.5ms.
   (b) The delays of link src1-R1 and link R2-rcv1 are set to 5ms; the delays of link src2-R1 and link R2-rcv2 are set to 20ms.
   (c) The delays of link src1-R1 and link R2-rcv1 are set to 5ms; the delays of link src2-R1 and link R2-rcv2 are set to 27.5ms.
4. Two TCP version can be selected: TCP SACK or TCP VEGAS

c. Procedure

We wrote a script to simulate three different test cases in each two TCP versions so there are totally 6 scenarios. In each scenario, we measured the throughputs of src1 and src2 in every 0.5ms except for the first 100ms of the simulation. We then calculated the average

throughputs of src1 and src2 and the ratio of the average throughput of src1 to the average throughput of src2. Finally, we used the average throughputs and ratio to compare the performance of the two TCP protocol in three different test cases.

d. Results

	SACK_1	SACK_2	SACK_3	VEGAS_1	VEGAS_2	VEGAS_3
Avg. throughput for src1	0.5238127546	0.5454530885	0.5652321870	0.5833188648	0.6875191987	0.7500233723
Avg. throughput for src2	0.4761706845	0.4545581302	0.4347790317	0.4166677796	0.3125208681	0.2500166945
Ratio of avg. throughput	1.1000525057	1.1999633319	1.3000447256	1.3999615360	2.1999145299	2.9998931624
_1 means case 1: src1-R1 and R2-rcv1 end-2-end delay = 5 ms, and src2-R1 and R2-rcv2 end-2-end delay = 12.5 ms _2 means case 2: src1-R1 and R2-rcv1 end-2-end delay = 5 ms, and src2-R1 and R2-rcv2 end-2-end delay = 20 ms _3 means case 3: src1-R1 and R2-rcv1 end-2-end delay = 5 ms, and src2-R1 and R2-rcv2 end-2-end delay = 27.5 ms						

The table above describes the average throughput of src1, the average throughput of src2 and ratio of average throughput of src1 to average throughput of src2 in each simulation cases. From this table, we can observe that with the increase of latency between src2 and rcv2, the average throughput of src1 increases, whereas the average throughput of src2 decreases in all simulation cases. Also, we can observe that the performance of TCP VEGAS is better than the performance of TCP SACK.

e. Questions

1. As the latency between src2 and rcv2 increases, RTT between src2 and rcv2 also increase, which decreases the average throughput of src2 and the traffic of the link between R1 and R2. Thus, src1 can send more packets to rcv1 due to the decrease of traffic, so the average throughput of src1 increases.

2. The reason why the performance of TCP VEGAS is better than that of TCP SACK is that TCP VEGAS measure the throughput to detect the congestion in the link, and it can detect the congestion more accurately than TCP SACK does, which measure the RTT and packet loss to detect the congestion. Furthermore, since the TCP SACK detects the congestion in the condition when a packet loses or times out, the retransmission rate of TCP SACK is higher than that of TCP VEGAS, which detects the congestion before a packet loses. Thus, the utilization of link bandwidth by TCP VEGAS is better.

f. Code

```
set ns [new Simulator]

if {$argc != 2} {
        puts "Invalid input! Please choose TCP Version and case number(ranging from 1 to 3)."
        exit 0
}

set f1 [open throughput1_table.tr w]
set f2 [open throughput2_table.tr w]
set nf [open out.nam w]
set tf [open out.tr w]

$ns namtrace-all $nf
$ns trace-all $tf

set count 0
set curr1 0
set curr2 0

# initialize the file with labels
proc initialize {} {
        global ns tcpsink1 tcpsink2 f1 f2 curr1 curr2 count
        puts $f1 "Time    Throughput(Mbs)"
        puts $f2 "Time    Throughput(Mbs)"
}

# finish the simulation and close the files
proc finish {} {
        global ns f1 f2 nf tf
        $ns flush-trace
        close $f1
        close $f2
        close $nf
        close $tf
        exec xgraph throughput1_table.tr throughput2_table.tr -geometry 800x400 &
        exit 0
}

proc record {} {
        global ns tcpsink1 tcpsink2 f1 f2 curr1 curr2 count
        # set the time gap to 0.5
        set time 0.5
        set now [$ns now]
```

```
        set bytes_sink1 [$tcpsink1 set bytes_]
        set bytes_sink2 [$tcpsink2 set bytes_]

        if {$now == 100} {
                $tcpsink1 set bytes_ 0
                $tcpsink2 set bytes_ 0
        } elseif {$now > 100 && $now < 400} {
                set throughput1 [expr $bytes_sink1/$time*8/1000000]
                set throughput2 [expr $bytes_sink2/$time*8/1000000]
                # total data received in bits (bytes_sink(bytes) * 8)
                set curr1 [expr $curr1 + $bytes_sink1*8]
                set curr2 [expr $curr2 + $bytes_sink2*8]
                set count [expr $count + 1]
                if {$count % 2 == 0} {
                        puts $f1 "$now     $throughput1"
                        puts $f2 "$now     $throughput2"
                } else {
                        puts $f1 "$now    $throughput1"
                        puts $f2 "$now    $throughput2"
                }
                $tcpsink1 set bytes_ 0
                $tcpsink2 set bytes_ 0
        } elseif {$now == 400} {
                # average throughtput in Mbps = total data received / (count(start from 100 ms)
* time interval) / 1000000
                set avg_throughput1 [ expr $curr1/($count*$time)/1000000]
                set avg_throughput2 [ expr $curr2/($count*$time)/1000000]
                puts "Average throughput for src1 : $avg_throughput1"
                puts "Average throughput for src2 : $avg_throughput2 "
                # ratio of average throughput
                set ratio_out [expr $avg_throughput1/$avg_throughput2 ]
                puts "Ratio of average throughput : $ratio_out"
        }
        $ns at [expr $now + $time] "record"
}

set R1 [$ns node]
set R2 [$ns node]
set src1 [$ns node]
set src2 [$ns node]
set rec1 [$ns node]
set rec2 [$ns node]

$ns duplex-link $R1 $R2 1Mb 5ms DropTail
```

```
if {[lindex $argv 0] eq "VEGAS"} {
	puts "TCP Version is set to: VEGAS"
	set tcp1 [new Agent/TCP/Vegas]
	set tcp2 [new Agent/TCP/Vegas]
} elseif {[lindex $argv 0] eq "SACK"} {
	puts "TCP Version is set to: SACK"
	set tcp1 [new Agent/TCP/Sack1]
	set tcp2 [new Agent/TCP/Sack1]
} else {
	puts "Invalid input for TCP version"
	exit 0
}

if {[lindex $argv 1] eq "1"} {
	puts "Case 1: \nsrc1-R1 and R2-rcv1 end-2-end delay = 5 ms\nsrc2-R1 and R2-rcv2 end-
2-end delay = 12.5 ms"
	$ns duplex-link $src1 $R1 10Mb 5ms DropTail
	$ns duplex-link $src2 $R1 10Mb 12.5ms DropTail
	$ns duplex-link $R2 $rec1 10Mb 5ms DropTail
	$ns duplex-link $R2 $rec2 10Mb 12.5ms DropTail
} elseif {[lindex $argv 1] eq "2"} {
	puts "Case 2: \nsrc1-R1 and R2-rcv1 end-2-end delay = 5 ms\nsrc2-R1 and R2-rcv2 end-
2-end delay = 20 ms"
	$ns duplex-link $src1 $R1 10Mb 5ms DropTail
	$ns duplex-link $src2 $R1 10Mb 20ms DropTail
	$ns duplex-link $R2 $rec1 10Mb 5ms DropTail
	$ns duplex-link $R2 $rec2 10Mb 20ms DropTail
} elseif {[lindex $argv 1] eq "3"} {
	puts "Case 3: \nsrc1-R1 and R2-rcv1 end-2-end delay = 5 ms\nsrc2-R1 and R2-rcv2 end-
2-end delay = 27.5 ms"
	$ns duplex-link $src1 $R1 10Mb 5ms DropTail
	$ns duplex-link $src2 $R1 10Mb 27.5ms DropTail
	$ns duplex-link $R2 $rec1 10Mb 5ms DropTail
	$ns duplex-link $R2 $rec2 10Mb 27.5ms DropTail
} else {
	puts "Invalid input for case number"
	exit 0
}

$ns attach-agent $src1 $tcp1
$ns attach-agent $src2 $tcp2

set tcpsink1 [new Agent/TCPSink]
```

```
set tcpsink2 [new Agent/TCPSink]

$ns attach-agent $rec1 $tcpsink1
$ns attach-agent $rec2 $tcpsink2
$ns connect $tcp1 $tcpsink1
$ns connect $tcp2 $tcpsink2

set ftp1 [new Application/FTP]
set ftp2 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp2 attach-agent $tcp2

$R1 shape circle
$R2 shape circle
$R1 color brown
$R2 color brown

$ns at 0 "$ftp1 start"
$ns at 0 "$ftp2 start"
$ns at 0 "initialize"
$ns at 100 "record"

$ns at 400 "$ftp1 stop"
$ns at 400 "$ftp2 stop"

$ns at 401 "finish"
$ns run
```