Investigation of TCP performance over different switch buffer sizes

Marcos Portnoi
University of Delaware
mportnoi@udel.edu

1 List of Abbreviations and Acronyms

bps	Bits Per Second
CSV	Comma-Separated Values
cwnd	Congestion window
FTP	File Transfer Protocol
Gbps	Giga ($1*10^9$) Bits Per Second
LAN	Local Area Network
MB	Mega ($1*10^6$) Bytes
ms	Milliseconds
PDU	Protocol Data Unit
Pkt	Packet
s	Seconds
ssthresh	Slow-start Threshold
Tcl	Tool Command Language
TCP	Transmission Control Protocol
WAN	Wide Area Network

2 Contents

3 Objective

Evaluate TCP performance from the point of view of hosts conducting FTP file transfers, using metrics of choice, as a means of comparison to two different network topologies for connecting local hosts to a high-speed Ethernet link to a WAN.

4 Description of the problem

A small local network (LAN) has hosts connected to a WAN through a 10 Gbps Ethernet switch. The switch has a specific buffer size, where PDUs are queued until the outgoing link is ready for transmission. Hosts conduct FTP file transfers through this switch and WAN to other hosts located off site.

Suppose an intermediate 1 Gbps switch is connected between the hosts and the 10 Gbps switch (such that the hosts are connected to the 1 Gbps switch through a 1 Gbps Ethernet link, and the 1 Gbps switch is connected to the 10 Gbps switch through a 10 Gbps link). If this 1 Gbps switch has a larger buffer size compared to the 10 Gbps switch (one or two orders of magnitude larger), how would the TCP performance, as viewed from the hosts engaged in FTP transfers, compare to the original topology?

The hypothesis is, therefore, that connecting hosts first to a 1 Gbps switch with large buffers, and then to a 10 Gbps switch with smaller buffers, could present better FTP transfer performance than connecting the hosts directly to the small-buffer 10 Gbps switch. The choice could be justified by relevant difference of prices between the 1 Gbps and 10 Gbps switches, such that it could be more economical to purchase a 1 Gbps switch with large buffers and a cheaper 10 Gbps switch, than to purchase a 10 Gbps switch with large buffers in order to achieve gains in FTP transfer performance.

In particular, the throughput, as a measure of the quantity of bytes transferred between one FTP client and its FTP server over time, will be analyzed in this report.

5 Description of the tools used

Experiments will be conducted utilizing the ns-2 network simulator (ns-2 The Network Simulator, 2010). The Tcl[1] scripts used to construct the simulations are available in Appendix A.

The scripts periodically record trace information into files. The trace information includes values of TCP variables, queue statistics, and throughput. All files have CSV format (where the commas are actually semicolons), but one that is pure text. Table 1 is a reference to these files names and an explanation of their contents.

Table 1: Trace files generated by the simulations and description of their contents.

File Name	Description of Contents
queueAvgStats.csv	Records average queue size statistics on source queues, sink queues, and switch queues.
queueStats.csv	Records total of counters from queues, such as arrived number of packets, throughput in arrival queues, lost packets, and others.
simParams.txt	Contains simulation parameters, such as simulation time, queue size limits (buffer sizes), and TCP configuration parameters.
tcpVariablesTrace.csv	Periodically records values of TCP variables, such as congestion window (cwnd) and slow-start threshold (ssthresh).
throughputs.csv	Contains periodically recorded throughputs as measured in TCP sinks.

At the end of the simulations, some files were converted to spreadsheets and manipulated with Microsoft Excel[2], which facilitates viewing the data, performing calculations and crafting graphs.

6 Topologies

Two basic topologies were devised to lead the experiments.

[1] http://en.wikipedia.org/wiki/Tcl
[2] http://office.microsoft.com/en-us/excel/

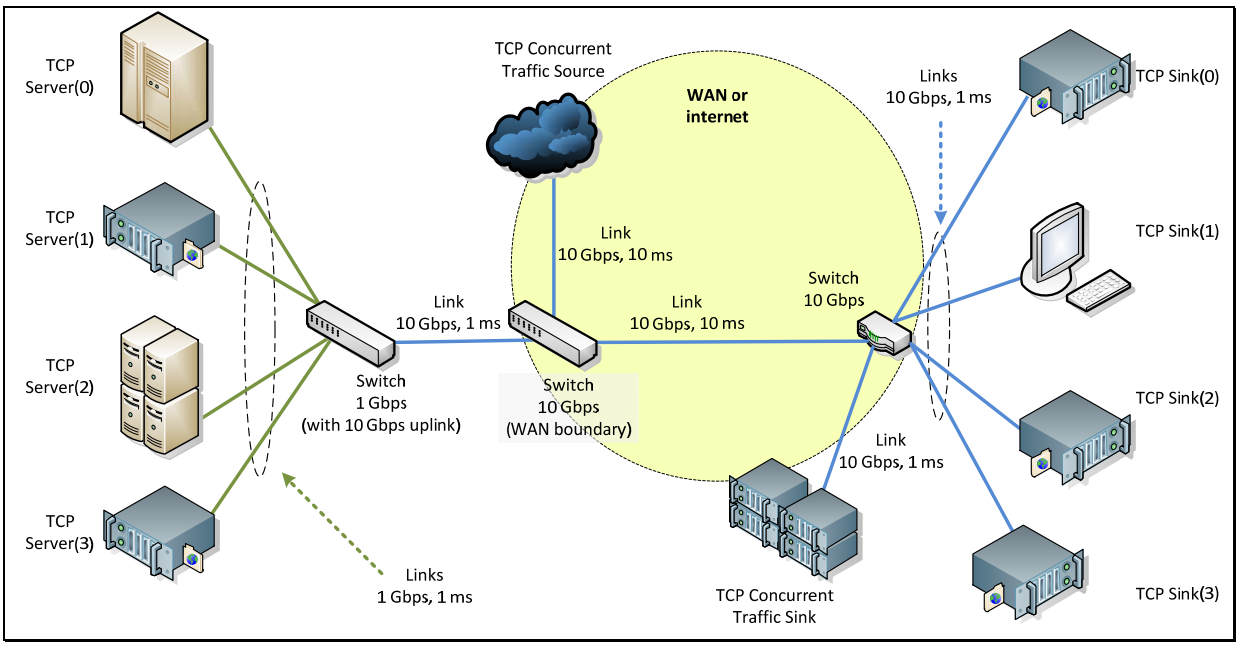

Figure 1: Topology where hosts in the LAN are connected to the WAN through a large-buffer 1 Gbps switch, and then to a 10 Gbps switch (the switches are interconnected through a 10 Gbps link).

All of the topologies contain, in the left edge, four nodes. Attached to each of these nodes, there is an FTP traffic generator. Each FTP traffic generator produces, therefore, one TCP stream. The TCP streams are directed to four TCP sinks on the right edge of the topologies, such that each TCP stream connects to one TCP sink. Two 10 Gbps switches define the WAN boundaries; the link between them is considered the WAN, or internet. This link bandwidth is 10 Gbps and the propagation delay is set to 10 ms.

To simulate concurrent WAN traffic, one extra FTP generator is connected to the left side 10 Gbps switch and its sink lies also on the right edge of the topologies. This link bandwidth is also 10 Gbps and its propagation delay, 10 ms. All other links have propagation delay of 1 ms.

The particular dissimilarities between the topologies are explored next.

6.1 Topology 1

In Topology 1 (Figure 1), the four FTP traffic generators are connected to the 10 Gbps edge switch through an intermediate 1 Gbps switch. The 1 Gbps switch is connected to the 10 Gbps switch through a 10 Gbps link; the other connections between the 1 Gbps switch to the FTP traffic generators occur through 1 Gbps links. In this topology, the effects on the effective throughput of the FTP traffic will be measured for diverse buffer sizes in the 1 Gbps switch.

6.2 Topology 2

Topology 2 (Figure 2) represents the elimination of the intermediate 1 Gbps switch; all FTP traffic generators are directly connected to the 10 Gbps WAN boundary switch through 10 Gbps links. The 10 Gbps WAN boundary switch remains with a small buffer size as compared to the 1 Gbps switch.

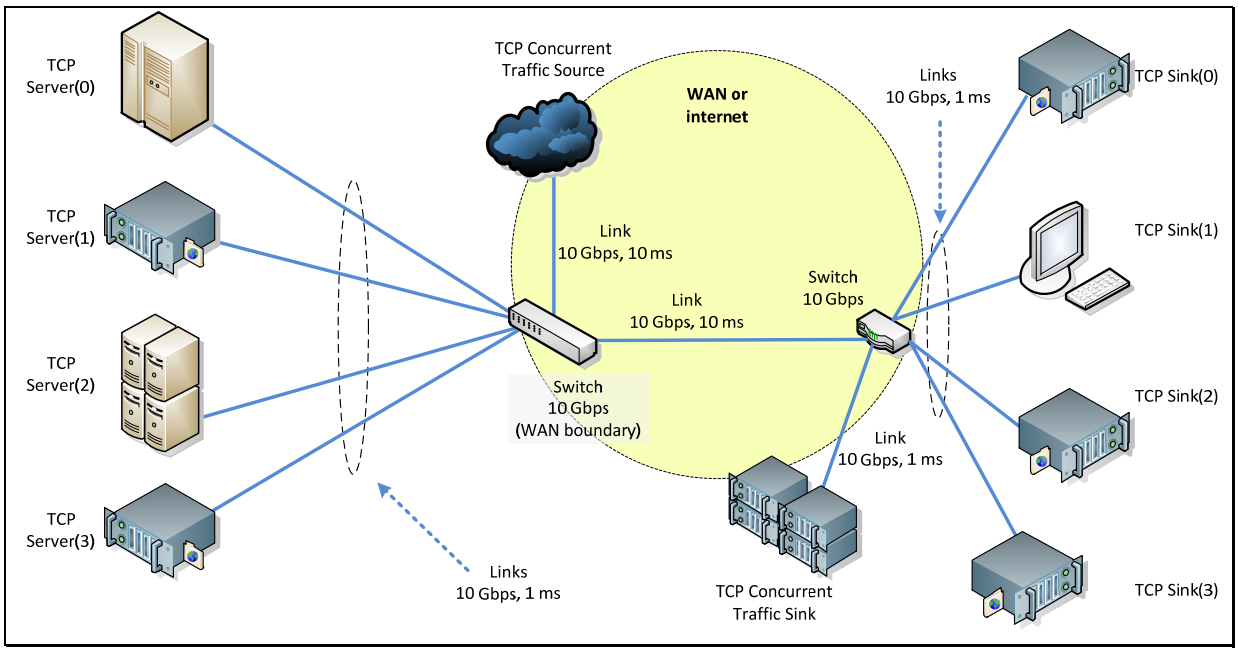

Figure 2: Topology where hosts in the LAN are connected to the WAN directly through a 10 Gbps switch.

7 Simulation Results and Analysis

7.1 Simulation with topology 1: switch 1 Gbps with 1000-packet buffer

In this experiment, Topology 1 was used. The simulation time was set to 1000 seconds and the buffer size of the 1 Gbps switch was configured to 1000 IP packets (the switch buffer size is controlled by means of setting the maximum output queue size of the link; in ns2, the queue size unit is IP packets). The buffer size for all other switches (including the 10 Gbps one used by the concurrent traffic generator) was set to 50 packets. The IP packet size is 1000 bytes, and the TCP advertised window size for the TCP senders is 64000 packets[3]. The concurrent traffic generator was started at time 0 seconds, and all FTP traffic generators were started at time 500 seconds. The results for throughput measured at the respective traffic sinks is in Figure 3.

The throughput displayed by the concurrent traffic generator (tcpConcurTraffic) evidences a gradual, linear increase until it reaches approximately 2,500 Mbps at 500 seconds. At this time, the FTP generators are started (in synchronism) and contend for the available bandwidth, causing a sharp drop in the concurrent traffic throughput. This drop is a result of the TCP congestion avoidance mechanism, which reduces the congestion window size in order to throttle down the sender's transmission speed. After this first drop and as the four FTP generators create traffic, the throughput of the concurrent traffic generator proceeds unaffected and gradually increasing until the simulation stops at 1000 seconds.

Albeit the concurrent traffic is the sole traffic in the entire topology until time 500 seconds, it never reaches the full available bandwidth of 10 Gbps due to its slow throughput growth. This can be credited to a combination of TCP's congestion avoidance/control algorithm, IP packet size of 1000 bytes, and buffer size. It appears that, very early after the traffic generation is started, and before being able to

[3] It is unclear, from ns2 documentation, whether the simulator's TCP window size unit is a transport layer PDU, or a network layer PDU.

reach the full available bandwidth, a packet loss event removes TCP from the slow start phase (where its throughput would virtually grow exponentially) and triggers the linear-growth algorithm.

Each of the FTP traffic generators is able to reach 1 Gbps of throughput (in Figure 3, all four FTP start synchronized and follow the same pattern, their lines in the graph overlapped by the generator named tcpSink(3)); each of them do have a dedicated 1 Gbps link provided by the 1 Gbps switch. They are able to sustain the 1 Gbps maximum throughput for roughly 18 seconds, after which TCP congestion control is triggered by packet losses and the transmission speed of each FTP generator is throttled down to approximately 570 Mbps.

Moreover, as indicated in literature, the large buffer size tends to keep the FTP generator traffics synchronized.

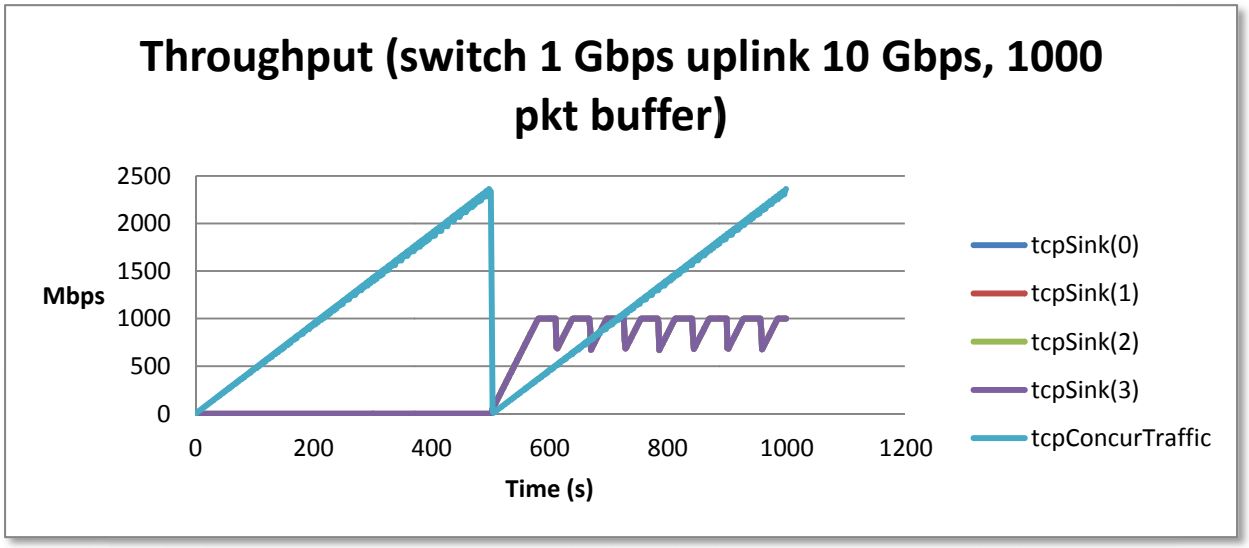

Figure 3: Throughput measured at TCP sinks for Topology 1, 1000-packet buffer size in 1 Gbps switch. The lines for tcpSink(0), tcpSink(1), and tcpSink(2) are overlapped by the purple line of tcpSink(3) (they are coincident). Maximum throughput of all sources is 6362 Mbps, and average of 2886 Mbps.

Table 2 shows the throughput values as measured at the arrival queues of the TCP sink nodes (over the duration of the traffic generation, which was 500 seconds). Basically, these values indicate the effective throughput achieved by each FTP generator. Roughly, each FTP generator is capable of attaining 85% of the respective available bandwidth. This can be credited to TCP's congestion avoidance/control algorithm which provokes the typical saw tooth behavior (Huston, 2006) as seen in Figure 3. The WAN available bandwidth of 10 Gbps is never fully utilized during the simulation. Interesting to notice is that, whereas the concurrent traffic graph displays spikes before packet losses (and subsequent speed decrease through the congestion control algorithm), the FTP generators graphs show plateaus, where the maximum throughput of 1 Gbps is sustained for some time before a packet loss triggers congestion control. This is directly related to the buffer size, which is 50 packets for the concurrent traffic, and 1000 packets for the FTP generators. A larger buffer size tends to smooth out the graph close to the peak throughput rate.

Table 2: Throughput as measured at the arrival queues of sink nodes (switch 1 Gbps uplink 10 Gbps, 1000 pkt buffer).

Node Queue	Throughput (Mbps) over 500 seconds
TCPSink(0)	857.3466477
TCPSink(1)	857.3466477
TCPSink(2)	857.346631
TCPSink(3)	857.3006214
Average	**857.335137**
Sum	**3429.3405478**

7.2 Simulation with topology 1: switch 1 Gbps with 1000-packet buffer; concurrent TCP traffic generator uses 8000-byte packets

This experiment also uses Topology 1 and shares the same settings as Experiment 7.1, with the exception that the concurrent traffic generator (tcpConcurTraffic) now generates 8000-byte IP packets, and the simulation was run for 3000 s to allow for stabilization of traffic. The intent of the large packet size is to force the concurrent traffic throughput to have faster rise and reach the maximum available bandwidth for it (10 Gbps) before the FTP generators are started. As a reminder, the switch directly connected to the concurrent traffic generator has 50-packet buffer size, and the FTP generators still work with 1000-byte IP packets.

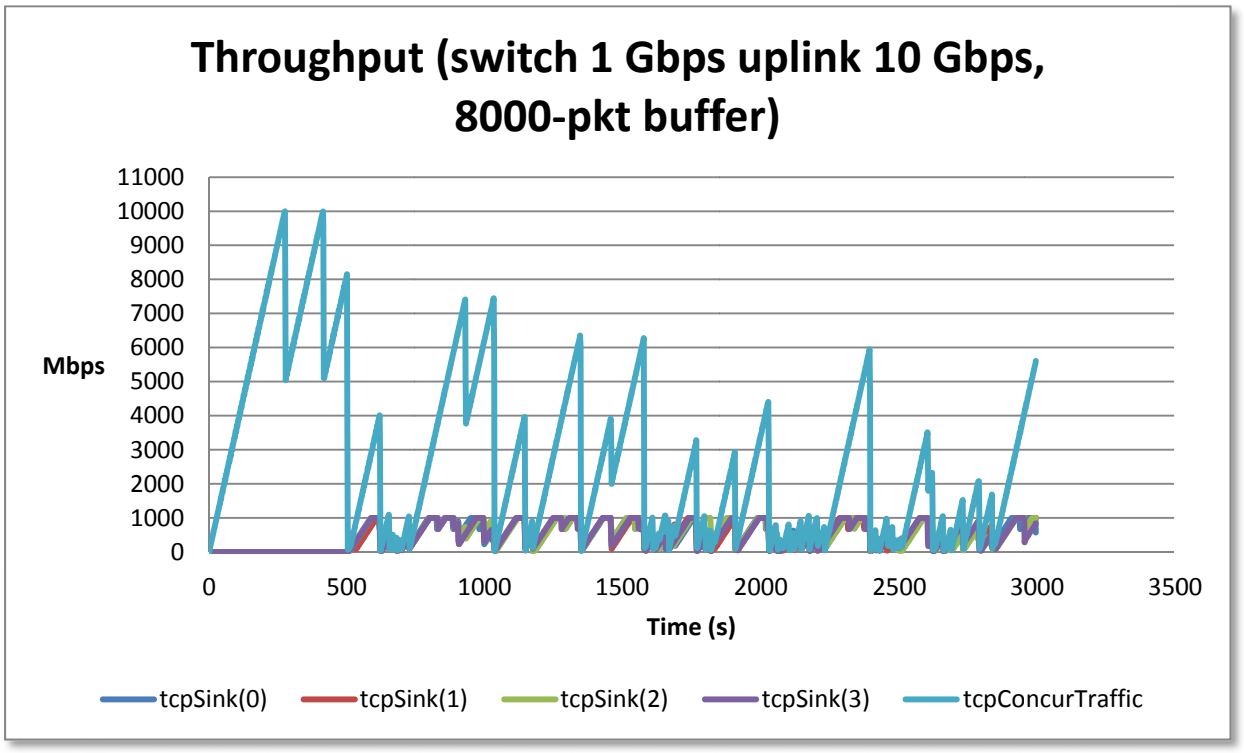

Figure 4: Throughput measured at TCP sinks for Topology 1, 1000-packet buffer size in 1 Gbps switch, and packet size set to 8000 bytes. Maximum throughput of all sources from is 9997 Mbps, and average of 4569 Mbps.

From Figure 4, it can be attested that the concurrent traffic generator is able to reach the full available bandwidth of 10 Gbps faster than the previous simulation 7.1. This 10 Gbps throughput cannot be

sustained, however, as the TCP congestion avoidance mechanism cuts down the transmission speed by half after packet losses. Again, the TCP saw tooth behavior is witnessed.

As the FTP generators are started at 500 seconds, a more erratic performance is displayed, as compared to simulation 7.1. The result is that, over the 500 seconds during which the FTP generators are active, the throughput measured at the arrival queues of the TCP sinks is lower than those of simulation 7.1 by a factor of approximately 36% (the new throughput is approximately 64% of that in simulation 7.1). Again, the 1000-packet buffer size in the 1 Gbps switch has a tendency to maintain the FTP traffics synchronized.

Since the concurrent traffic generator is now effectively using more bandwidth, this creates more contention with the FTP generators; the traffic pattern does not seem to stabilize but near 1500 s of simulation.

Table 3: Throughput as measured at the arrival queues of sink nodes (switch 1 Gbps uplink 10 Gbps, 1000-pkt buffer, Concur 8000-byte packet size).

Node Queue	Throughput (Mbps) over 500 seconds
TCPSink(0)	553.6858963
TCPSink(1)	544.0242065
TCPSink(2)	559.1622102
TCPSink(3)	543.6314026
Average	**550.1259289**
Sum	**2200.5037156**

Maximum throughput of all sources from Figure 4 was 9997 Mbps, and average of 5578 Mbps. As compared to simulation 7.1, clearly the concurrent traffic generator was responsible for elevating the overall maximum and average throughput statistics; the FTP generators average was lower in this simulation, however, as may be verified in Table 3.

7.3 Simulation with topology 2: switch 10 Gbps only with 50-packet buffer

This simulation uses topology 2, where the 1 Gbps switch is eliminated and all FTP traffic generators are directly connected to a 10 Gbps switch (thus, each FTP generator has a dedicated 10 Gbps link to the switch).

All switches have 50-packet buffers. The IP packet size is 1000 bytes. The simulation lasts 1000 seconds; the concurrent traffic generator starts at time 0 s, and the FTP generators activate at time 500 s.

From the analysis of Figure 5, the FTP generators now lack synchronization, even though they do start at the same time. The saw tooth peaks did not exceed 2,500 Mbps for the FTP generators during the simulation, and there are no plateaus of sustained throughput as observed in simulation 7.1. The concurrent traffic is significantly impacted by the contending FTP generators in this simulation; after the FTP generators are activated, the concurrent traffic throughput is never allowed to go beyond 500 Mbps approximately.

By contrasting the numbers in Table 2, Table 3, and This result may be an indication that the slower 1 Gbps switch outperformed the faster 10 Gbps switch due to the former's larger buffer size.

Table 4, the configuration used in simulation 7.1 presents better throughput performance for the FTP generators. Despite the fact that, in this simulation 7.3, each FTP generator could develop transmission

speeds superior to 1 Gbps (since they were directly connected to a 10 Gbps switch), their throughput in average were worse than those achieved in simulation 7.1 (857 Mbps for the latter, and 689 Mbps for the former).

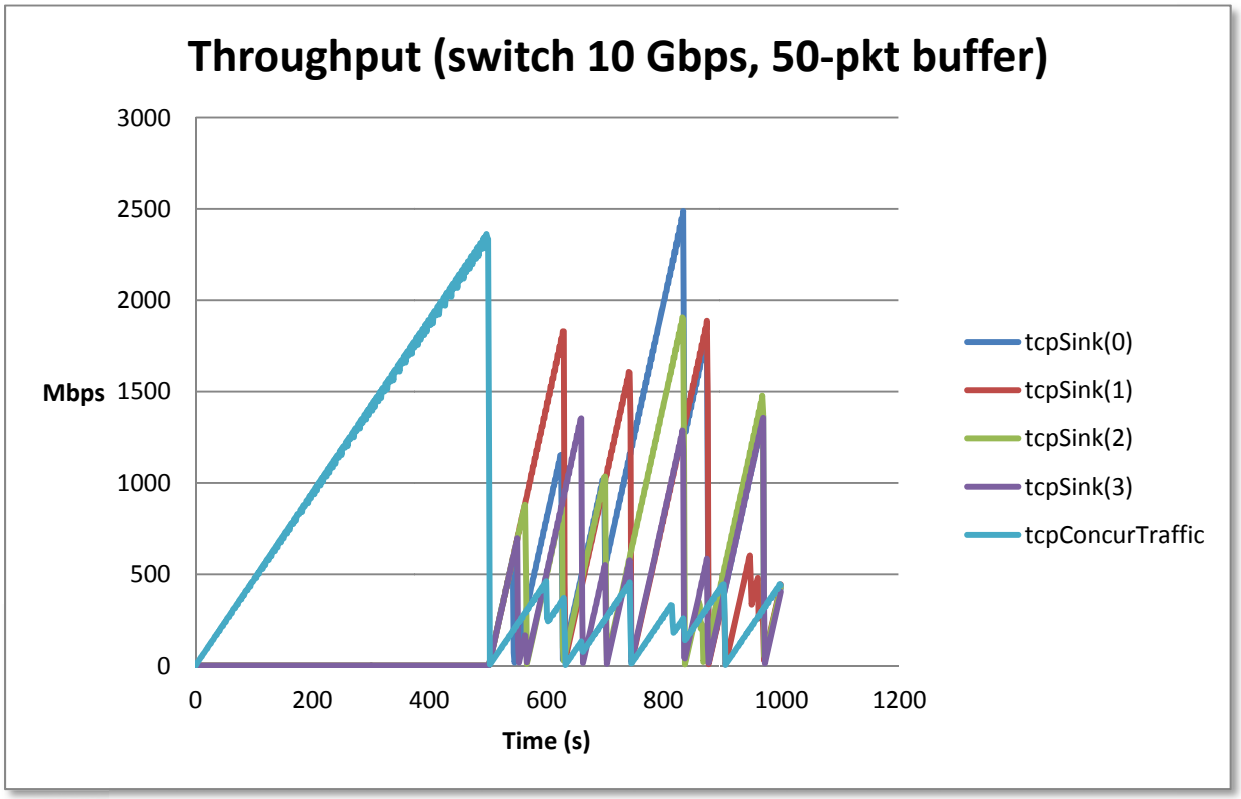

Figure 5: Throughput measured at TCP sinks for Topology 2, 50-packet buffer size in all switches, and packet size set to 1000 bytes. Maximum throughput of all sources is 7141 Mbps, and average of 2082 Mbps.

This result may be an indication that the slower 1 Gbps switch outperformed the faster 10 Gbps switch due to the former's larger buffer size.

Table 4: Throughput as measured at the arrival queues of sink nodes (switch 10 Gbps only, 50-packet buffer).

Node Queue	Throughput (Mbps) over 500 seconds
TCPSink(0)	877.4849248
TCPSink(1)	745.1676218
TCPSink(2)	631.2969069
TCPSink(3)	502.6122323
Average	**689.1404215**
Sum	**2756.5616858**

7.4 Simulation with topology 1: switch 1 Gbps with 50-packet buffer

This essay shares all attributes and settings of simulation 7.1, but now all buffers are of size 50 packets. Figure 6 contains the throughput graphs, and Table 5 summarizes the measured throughput of the TCP sinks only.

With smaller buffer sizes, all traffic graphs now show spikes and no plateaus of sustained throughput. The addition of the intermediate 1 Gbps, 50-packet buffer switch does, however, increase the total buffer size in the path between one FTP generator and its TCP sink, as compared to essay 7.3, where this intermediate switch did not exist, and all FTP generators were directly connected to a 10 Gbps switch with 50-packet buffer size. This might account for the synchronization observed among all FTP generators (Figure 6) and the better throughput performance evidenced in Table 5, as compared to simulation 7.3 (see This result may be an indication that the slower 1 Gbps switch outperformed the faster 10 Gbps switch due to the former's larger buffer size.

Table 4). In summary, with an intermediate, 50-packet buffer 1 Gbps switch, average throughput measured for the FTP generators at their sinks was 718 Mbps. Without the 1 Gbps, with a direct connection to a 10 Gbps, 50-packet buffer switch, the measured average throughput was 689 Mbps.

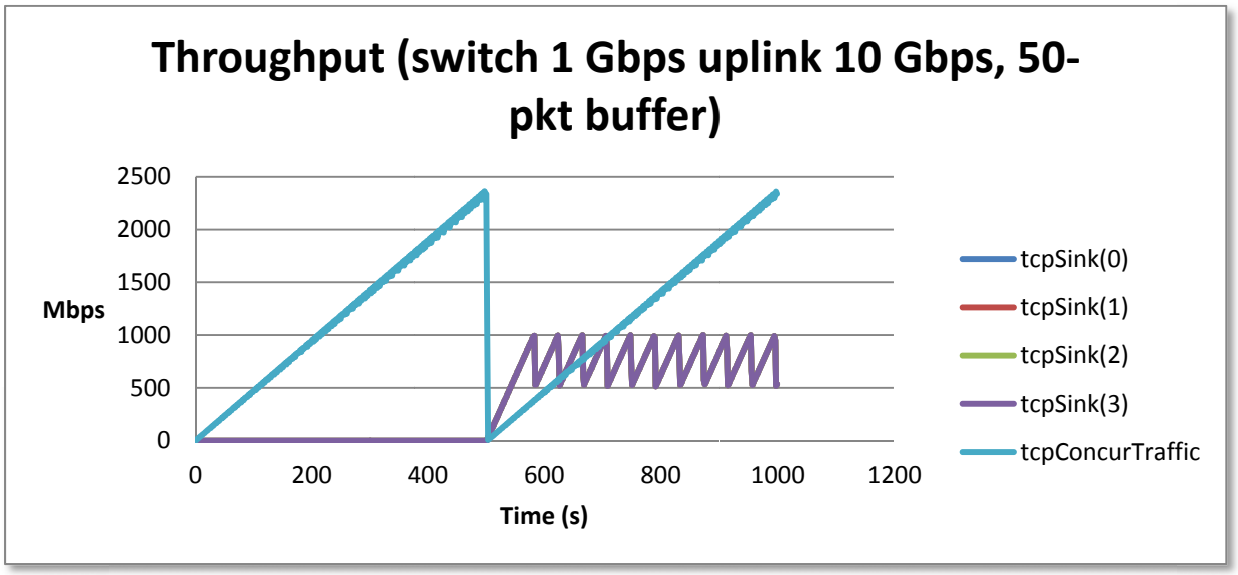

Figure 6: Throughput measured at TCP sinks for Topology 1, 50-packet buffer size in all switches. The lines for tcpSink(0), tcpSink(1), and tcpSink(2) are overlapped by the purple line of tcpSink(3) (they are coincident). Maximum throughput of all sources is 6259 Mbps, and average of 2612 Mbps.

It can be seen that, in regards to throughput, the FTP generators benefit from having a 1 Gbps switch with larger buffer size between their connection to the edge 10 Gbps switch with small buffer size.

Table 5: Throughput as measured at the arrival queues of sink nodes (switch 1 Gbps, 50-packet buffer).

Node Queue	Throughput (Mbps) over 500 seconds
TCPSink(0)	718.8000275
TCPSink(1)	718.8000275
TCPSink(2)	718.8000275
TCPSink(3)	718.8000109
Average	**718.8000234**
Sum	**2875.2000934**

It is interesting to perform two additional simulations in order to verify the extent of benefit that the buffer size can pose to the throughput of the FTP generators. The next two essays fulfill this interest.

7.5 Simulation with topology 1: switch 1 Gbps with 10,000-packet buffer

This simulation copies all parameters for simulation 7.1, but the buffer size for the 1 Gbps switch was set to 10,000 packets, or one order of magnitude higher than the former.

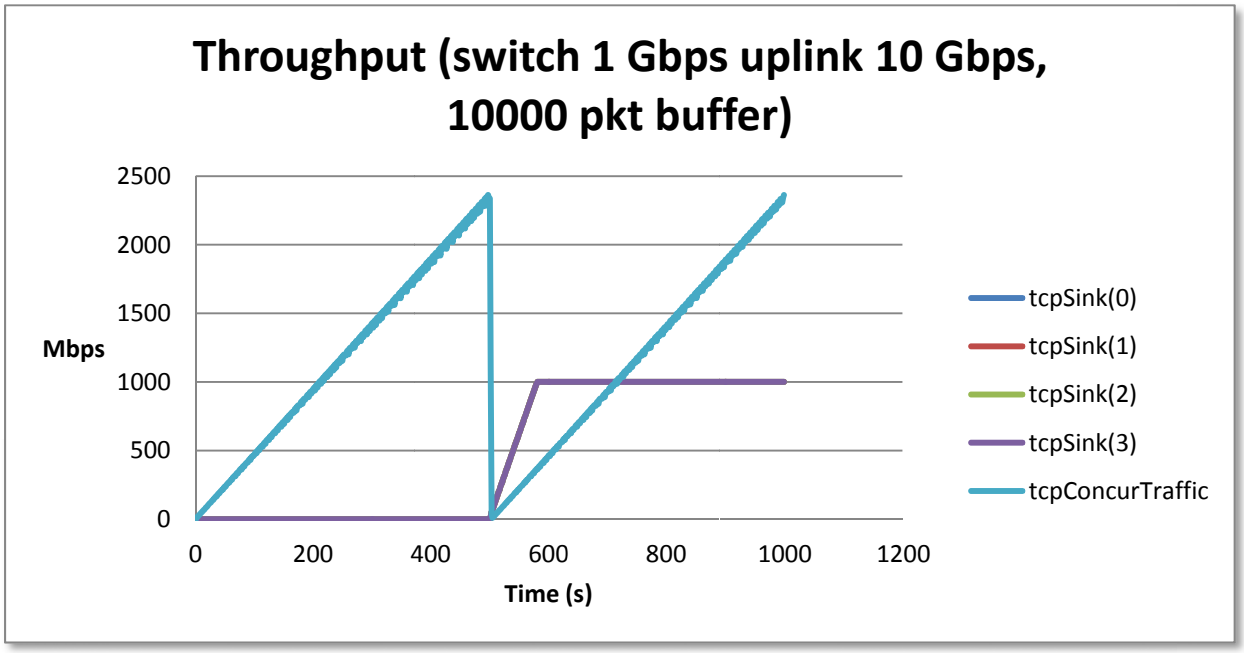

Figure 7: Throughput measured at TCP sinks for Topology 1, 10,000-packet buffer size in 1 Gbps switch, 50 packets for the rest. The lines for tcpSink(0), tcpSink(1), and tcpSink(2) are overlapped by the purple line of tcpSink(3) (they are coincident). Maximum throughput of all sources is 6361 Mbps, and average of 3017 Mbps.

Clearly, the larger buffer size of the 1 Gbps switch results in a throughput, for the FTP generators, that approach the maximum available (Table 6), and it tends to be sustained (Figure 7). As the concurrent traffic grows and eventually the sum of all traffic in the 10 Gbps link reaches this limit, packet losses will occur and congestion control will likely trigger a decrease of transmission speeds on all traffic generators. This will occur sometime beyond 1000 s, the end of this simulation, and is therefore not shown in the graph in Figure 7.

Table 6: Throughput as measured at the arrival queues of sink nodes (switch 1 Gbps, 10,000-packet buffer).

Node Queue	Throughput (Mbps) over 500 seconds
TCPSink(0)	923.3922221
TCPSink(1)	923.3922054
TCPSink(2)	923.3921888
TCPSink(3)	923.3459296
Average	**923.3806365**
Sum	**3693.5225459**

7.6 Simulation with topology 2: switch 10 Gbps only with 1,000-packet buffer

This essay mirrors all settings in simulation 7.3, but the 10 Gbps switch now has a buffer size of 1,000 packets.

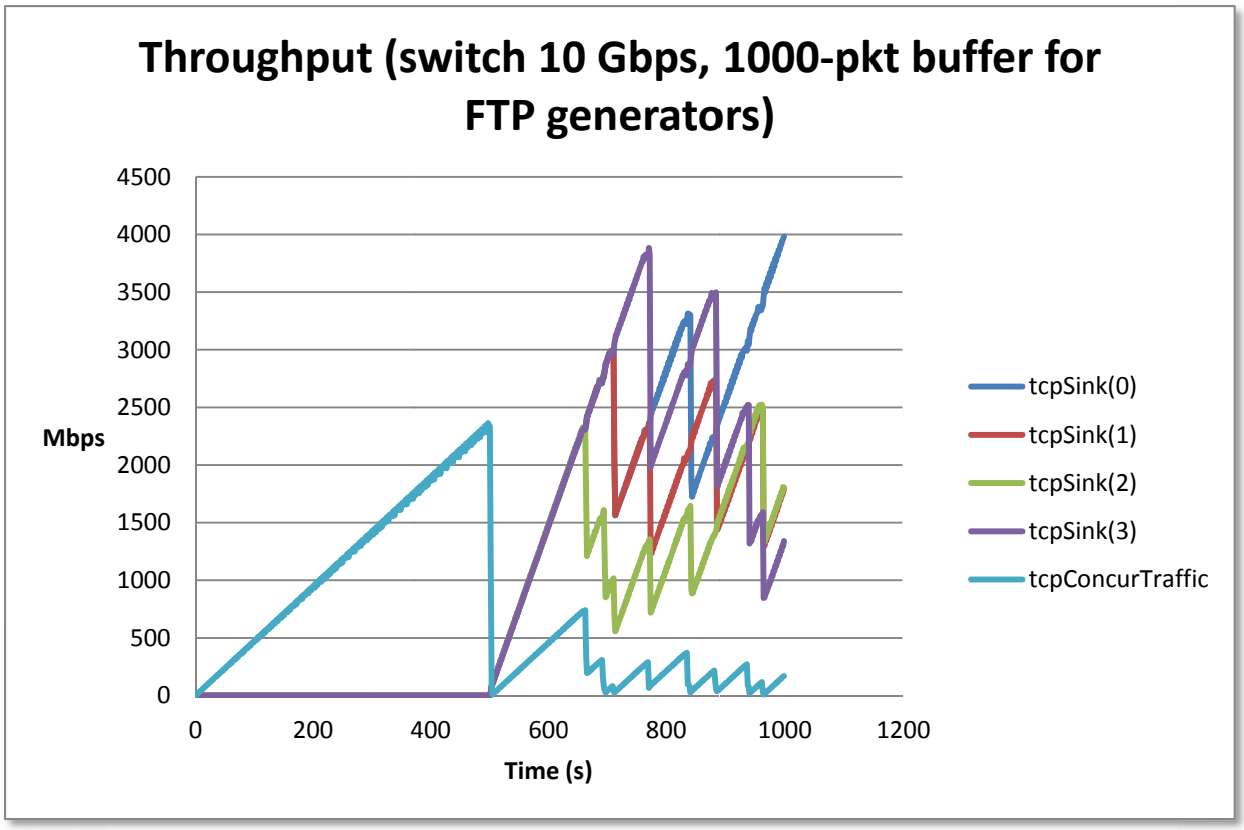

Figure 8: Throughput measured at TCP sinks for Topology 2, 1000-packet buffer size in all switches, and packet size set to 1000 bytes. Maximum throughput of all sources is 10,000 Mbps, and average of 4413 Mbps.

Figure 8 indicates that the buffer size is insufficient to cause synchronization of the FTP traffic, but their throughput measurements are significantly improved when compared to those in simulation 7.3, as seen in Table 7.

Most notably, with a 1 Gbps switch, a large buffer size can only make the throughput of the FTP generators reach the maximum available bandwidth of 1 Gbps for each generator. With a 10 Gbps switch, using a 1,000-packet buffer size allows the FTP generators to develop higher transmission speeds that outperform those of the 1 Gbps switch. Table 7 tells that each FTP generator was able to transmit on average at speeds beyond 1 Gbps.

Table 7: Throughput as measured at the arrival queues of sink nodes (switch 10 Gbps only, 1000-packet buffer).

Node Queue	Throughput (Mbps) over 500 seconds
TCPSink(0)	2214.591276
TCPSink(1)	1779.878132
TCPSink(2)	1340.549451
TCPSink(3)	2102.94666
Average	**1859.49138**
Sum	**7437.965519**

8 Conclusion, comments on findings

This document reports several simulations intended to investigate the throughput performance of four FTP traffic generators in two topologies. In one topology, the FTP generators are connected to a 1 Gbps switch, then to a 10 Gbps switch through a 10 Gbps link, then to a WAN and finally to their FTP sources (the WAN is represented by a 10 Gbps link with 10 ms propagation delay). In the other topology, the 1 Gbps switch is eliminated, and the FTP generators are directly connected to the 10 Gbps switch before the WAN.

The hypothesis to investigate is whether using an intermediate 1 Gbps switch with large buffers will yield better throughput performance measured on the FTP generators when compared to directly connecting them to a 10 Gbps switch with considerably smaller buffer size (specifically, 50 packets).

The simulations conducted show that, when the 10 Gbps has a small, 50-packet buffer size, there is benefit from using an intermediate 1 Gbps switch, even if that switch also has a small buffer size of 50 packets (because its buffer adds to the total path buffer size). However, when the 10 Gbps switch is equipped with a larger buffer size (in this study, a 1,000-packet buffer size, which represents approximately 1 MB), it easy outperforms[4] a topology with the intermediate 1 Gbps switch, no matter what buffer size the latter holds.

Therefore, the use of an intermediate 1 Gbps switch is justified only if the 10 Gbps switch has small buffers, in the order of 50 packets.

Note: It is important to observe that the simulations focus on long-duration FTP transfers. For short durations, the results may vary.

9 References

ns-2 The Network Simulator. (2010). *ns-2 The Network Simulator*.

Huston, G. (2006). Gigabit TCP. *The Internet Protocol Journal (IPJ), 9*(2), 2-26.

Kurose, J. F., & Ross, K. W. (2009). *Computer Networking: A Top-Down Approach* (5th ed.). Addison-Wesley Publishing Company.

Tanenbaum, A. (2002). *Computer Networks.* Prentice Hall Professional Technical Reference.

[4] Performance, here, is throughput as measured in the TCP sinks for the FTP generators.

Appendix A: Tcl simulation scripts

Presented here are the two main simulation scripts, written in Tcl. For the different simulations, a few parameters were modified. Refer to each simulation report.

Table 8: Tcl script for topology 1 (with intermediate 1 Gbps switch).

```
#This simulation investigation intends to observe the behavior of a number of
TCP flows
#when traversing through switches of different bandwidths and different
buffer sizes.
#
#The question is:  is TCP performance better when going through a slower
switch, but with
#larger buffers, then through a higher-speed switch with small buffers, or is
TCP
#performance better when going through a higher speed switch only with small
buffers?
#
#In this simulation, we will use a 1 Gbps switch and a 10 Gbps switch.
Buffer sizes of
#each will be modified.
#
#Observe an important comment about ns behavior regarding the TCP variables
cwnd_ and
#window_, in the TCP source creation section further below.
#
# August 2010 Marcos Portnoi.

# --- Random Numbers and Seed
#Seed default random number generator with current time
global defaultRNG
$defaultRNG seed 0

#Number of TCP traffic sources (used to create traffic generators, nodes,
etc.)
set tcpGenerators 4

#Turn tracing on=1 or off=0
#be careful:  ns can generate enormous trace files
set traceAll 0
set traceQueue 1

#Queue size limits; ns default is 50
set tcpSourceQueueLimit 1000
set tcpConcurTrafficQueueLimit 50
set switch1GbpsQueueLimit 1000
set switch10GbpsQueueLimit 50

#Create simulator object
set ns [new Simulator]
set MAX_TIME 500.0; #set maximum simulation time; use at least a decimal to
prevent this constant being interpreted as a integer

# --- Define color index (class color)
# list of colors available at ~ns/man/mann/colors.n
```

```
$ns color 0 red
$ns color 1 blue
$ns color 2 chocolate
$ns color 3 yellow
$ns color 4 green
$ns color 5 tan
$ns color 6 gold
$ns color 7 black
$ns color 8 white
$ns color 9 darkblue
$ns color 10 cyan
$ns color 11 magenta
$ns color 12 orange
$ns color 13 bisque
$ns color 14 purple
$ns color 15 coral
$ns color 16 grey
$ns color 17 khaki
$ns color 18 AntiqueWhite
$ns color 19 aquamarine
$ns color 20 azure
$ns color 21 DarkSalmon
$ns color 22 DarkSeaGreen
$ns color 23 firebrick

#Use dinamic routing
#$ns rtproto DV

#Open files for trace if corresponding flags are set
if {$traceAll} {
    set nf [open out.nam w]
    set tr [open out.tr w]
    $ns namtrace-all $nf
    $ns trace-all $tr
}
if {$traceQueue} {
    set queueStats [open queueStats.csv w]
    set queueAvgStats [open queueAvgStats.csv w]
}

set throughputs [open throughputs.csv w]
set tcpVariablesTrace [open tcpVariablesTrace.csv w]
set simParams [open simParams.txt w]

#===================================================================
#Create nodes
#create main TCP sources and sinks
for {set i 0} {$i < $tcpGenerators} {incr i} {
    set source($i) [$ns node]
    set sink($i) [$ns node]
}

set switch1Gbps [$ns node]
set switch10Gbps [$ns node]
set switch10GbpsOut [$ns node]
#Concurrent traffic
set concurTraffic [$ns node]
```

```
set sinkConcurTraffic [$ns node]

#Connect nodes
#default queue size limit is 50
for {set i 0} {$i < $tcpGenerators} {incr i} {
    $ns duplex-link $source($i) $switch1Gbps 1000Mb 1ms DropTail
    $ns queue-limit $source($i) $switch1Gbps $tcpSourceQueueLimit
    $ns duplex-link $switch10GbpsOut $sink($i) 10000Mb 1ms DropTail
    $ns queue-limit $switch10GbpsOut $sink($i) $switch10GbpsQueueLimit
}

$ns duplex-link $switch1Gbps $switch10Gbps 1000Mb 1ms DropTail
$ns queue-limit $switch1Gbps $switch10Gbps $switch1GbpsQueueLimit
$ns duplex-link $switch10Gbps $switch10GbpsOut 10000Mb 10ms DropTail
$ns queue-limit $switch10Gbps $switch10GbpsOut $switch10GbpsQueueLimit
$ns duplex-link $concurTraffic $switch10Gbps 10000Mb 10ms DropTail
$ns queue-limit $concurTraffic $switch10Gbps $tcpConcurTrafficQueueLimit
$ns duplex-link $switch10GbpsOut $sinkConcurTraffic 10000Mb 1ms DropTail
$ns queue-limit $switch10GbpsOut $sinkConcurTraffic $switch10GbpsQueueLimit

#Set visual orientation for nam
for {set i 0} {$i < $tcpGenerators} {incr i} {
    $ns duplex-link-op $source($i) $switch1Gbps queuePos 0.5
    $ns duplex-link-op $switch10GbpsOut $sink($i) queuePos 0.5
}
$ns duplex-link-op $switch1Gbps $switch10Gbps queuePos 0.5
$ns duplex-link-op $switch10Gbps $switch10GbpsOut queuePos 0.5
$ns duplex-link-op $concurTraffic $switch10Gbps queuePos 0.5
$ns duplex-link-op $switch10GbpsOut $sinkConcurTraffic queuePos 0.0

#modify here for automatic placement of n tcp sources
$ns duplex-link-op $source(0) $switch1Gbps orient 1.67
$ns duplex-link-op $source(1) $switch1Gbps orient 1.83
$ns duplex-link-op $source(2) $switch1Gbps orient 0.17
$ns duplex-link-op $source(3) $switch1Gbps orient 0.33
$ns duplex-link-op $switch1Gbps $switch10Gbps orient 0.0
$ns duplex-link-op $switch10Gbps $switch10GbpsOut orient 0.0
#modify here for automatic placement of n tcp sinks
$ns duplex-link-op $switch10GbpsOut $sink(0) orient 0.33
$ns duplex-link-op $switch10GbpsOut $sink(1) orient 0.17
$ns duplex-link-op $switch10GbpsOut $sink(2) orient 1.83
$ns duplex-link-op $switch10GbpsOut $sink(3) orient 1.67
$ns duplex-link-op $concurTraffic $switch10Gbps orient 1.5
$ns duplex-link-op $switch10GbpsOut $sinkConcurTraffic orient 1.5

#================================================================================
============================
#Create TCP agents and TCPSinks, attach them to nodes and link them
#
#Also, set here the maximum window_ size (cwnd_ or congestion window) for TCP
#Here is the catch:  ns allows TCP algorithm to grow cnwd accordingly if
there is no congestion.  The
#TCP variables trace file can demonstrate that.  However, the maximum burst
of TCP PDUs (segments) is
#limited to the maximum (advertised) window_ size, regardless whether cwnd_
is larger.  Thus, if in a specific moment,
#cwnd_ value is 2000 and window_ is set to 1000, TCP will send a burst of
```

```
only 1000 PDUs and then wait for
#the corresponding ACKs (since the max advertised window_ size is 1000).
#
for {set i 0} {$i < $tcpGenerators} {incr i} {
    set connection_list [$ns create-connection-list TCP/Reno $source($i)
TCPSink $sink($i) $i]
    set tcpSource($i) [lindex $connection_list 0]
    set tcpSink($i) [lindex $connection_list 1]
    $tcpSource($i) set fid_ $i
    $tcpSink($i) set fid_ $i
    $tcpSource($i) set window_ 64000
    #$tcpSource($i) set windowInit_ 64000
    #$tcpSource($i) set packetSize_ 8000
    $tcpSource($i) set maxburst_ 64000
    $tcpSink($i) set window_ 64000
}
#flowID for Concurrent traffic
set concurId 100
set connection_list [$ns create-connection-list TCP/Reno $concurTraffic
TCPSink $sinkConcurTraffic $concurId]
set tcpConcurTraffic [lindex $connection_list 0]
set tcpSinkConcurTraffic [lindex $connection_list 1]
$tcpConcurTraffic set fid_ $concurId
$tcpSinkConcurTraffic set fid_ $concurId
$tcpConcurTraffic set window_ 64000
#$tcpConcurTraffic set windowInit_ 64000
#$tcpConcurTraffic set packetSize_ 8000
$tcpConcurTraffic set maxburst_ 64000
$tcpSinkConcurTraffic set window_ 64000

#Create FTP applications and attach them to agents
for {set i 0} {$i < $tcpGenerators} {incr i} {
    set ftp($i) [new Application/FTP]
    $ftp($i) attach-agent $tcpSource($i)
}
set ftpConcur [new Application/FTP]
$ftpConcur attach-agent $tcpConcurTraffic


#Create queue monitors if corresponding trace flag is set
if {$traceQueue} {
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        set qmonSource($i) [$ns monitor-queue $source($i) $switch1Gbps ""]
        set qmonSink($i) [$ns monitor-queue $switch10GbpsOut $sink($i) ""]
    }
    set qmonSwitch1_10 [$ns monitor-queue $switch1Gbps $switch10Gbps ""]
    set qmonSwitch10_out [$ns monitor-queue $switch10Gbps $switch10GbpsOut
""]
    set qmonConcur_Switch10 [$ns monitor-queue $concurTraffic $switch10Gbps
""]
    set qmonSwitch10_SinkConcur [$ns monitor-queue $switch10GbpsOut
$sinkConcurTraffic ""]
}

#Create statistics record procedure
proc record {time} {
    global tcpSource tcpConcurTraffic tcpSink tcpSinkConcurTraffic
```

```
throughputs tcpGenerators tcpVariablesTrace
    #Get instance of simulator
    set ns [Simulator instance]

    #Get current time
    set now [$ns now]
    puts "Recording at $now s..."

    #print header only once to sink throughputs file
    if {$now == 0} {
        puts $throughputs "Throughput MBps"
        puts -nonewline $throughputs "time;"
        #tcpSources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts -nonewline $throughputs "tcpSink($i);"
        }
        #concurrent traffic sink
        puts $throughputs "tcpSinkConcurTraffic"
    }

    #How many bytes have been received by traffic sinks?
    # Problem here is that bytes_ in tcp-sink.cc is a INT value... max is
2147483647, and this is easily surpassed in gigabit links; bytes_ then WRAPS
    # To avoid this for 10Gbps of throughput, we can use the relation:
    # time = max_int_bytes_ / throughput
    # where time is the time step or resolution where bytes_ should be read
from TCPSink, and then set to zero again.
    # Thus, for the usual max_int, we have:
    # time = 2147483647 / 10,000,000,000 * 8 (multiply by 8 to equalize the
units to bytes) = 1.7179 s
    # Anything below that value should work accordingly, without integer
wrapping.
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        set bw($i) [$tcpSink($i) set bytes_]
    }
    set bwConcur [$tcpSinkConcurTraffic set bytes_]

    #Calculate bandwidth (MBit/s) and write to file
    puts -nonewline $throughputs "$now;"
    puts -nonewline "$now "
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        puts -nonewline $throughputs "[expr $bw($i)/$time*8.0/1000000.0];"
        puts -nonewline "[expr $bw($i)/$time*8.0/1000000.0];"
    }
    #concurrent traffic bandwidth (MBit/s)
    puts $throughputs "[expr $bwConcur/$time*8.0/1000000.0]"
    puts "[expr $bwConcur/$time*8.0/1000000.0]"

    #Reset bytes_ values on traffic sinks
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        $tcpSink($i) set bytes_ 0
    }
    $tcpSinkConcurTraffic set bytes_ 0

    #record TCP variables
    #print header only once to tcpVariablesTrace file
    if {$now == 0} {
```

```
        puts $tcpVariablesTrace
"node;time;awnd_;rttvar_;backoff_;cwnd_;ssthresh_;rtt_;srtt_"
    }
    #tcpSources
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        puts $tcpVariablesTrace "tcpSource($i);$now;[$tcpSource($i) set
awnd_];[$tcpSource($i) set rttvar_];[$tcpSource($i) set
backoff_];[$tcpSource($i) set cwnd_];[$tcpSource($i) set
ssthresh_];[$tcpSource($i) set rtt_];[$tcpSource($i) set srtt_]"
    }
    #tcp concurrent traffic
    puts $tcpVariablesTrace "tcpConcurTraffic;$now;[$tcpConcurTraffic set
awnd_];[$tcpConcurTraffic set rttvar_];[$tcpConcurTraffic set
backoff_];[$tcpConcurTraffic set cwnd_];[$tcpConcurTraffic set
ssthresh_];[$tcpConcurTraffic set rtt_];[$tcpConcurTraffic set srtt_]"

    #Re-schedule procedure
    $ns at [expr $now+$time] "record $time"
}

#Schedule
$ns at 0.0 "record 1.5"
$ns at 0.0 "puts \"Starting simulation...\""
#ftp generators start synchronized!
for {set i 0} {$i < $tcpGenerators} {incr i} {
    #$ns at [expr 0.0 + $i] "$ftp($i) start"
    $ns at 0.0 "$ftp($i) start"
}
$ns at 0.0 "$ftpConcur start"
for {set i 0} {$i < $tcpGenerators} {incr i} {
    $ns at $MAX_TIME "$ftp($i) stop"
}
$ns at $MAX_TIME "$ftpConcur stop"
$ns at $MAX_TIME "finish"

#Finish procedure
proc finish {} {
    global ns nf tr throughputs tcpGenerators queueStats queueAvgStats
tcpSource tcpConcurTraffic simParams
    global qmonSource qmonSink
    global qmonSwitch1_10 qmonSwitch10_out qmonConcur_Switch10
qmonSwitch10_SinkConcur
    global tcpSourceQueueLimit tcpConcurTrafficQueueLimit
switch1GbpsQueueLimit switch10GbpsQueueLimit
    global MAX_TIME traceAll traceQueue
    if {$traceAll} {
        $ns flush-trace
        #close trace files
        close $nf
        close $tr
    }
    close $throughputs
    puts "Ending simulation.  Simulation time:  $MAX_TIME s"
    puts "                                      ====="

    #---------------- generating queue statistics ----------------------
```

```
    #outputting queue stats to screen, if trace flag is set
    #queue sums stats
    if {$traceQueue} {
        puts "\nStatistics:"
        puts "                                    Arrived
Lost                      Departed"
        puts "                      Packets      Bytes    Throughput(Mbps)
Packets     Bytes    Packets     Bytes    Throughput(Mbps)"
        puts "-------------------------------------------------------------------
--------------------"

        #sources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts -nonewline [format "Queue qmonSource($i):   %7d   %10d
%10f" [eval $qmonSource($i) set parrivals_] [eval $qmonSource($i) set
barrivals_] [expr [$qmonSource($i) set barrivals_]/$MAX_TIME*8.0/1000000.0]]
            puts -nonewline [format "  %7d   %10f" [eval $qmonSource($i) set
pdrops_] [eval $qmonSource($i) set bdrops_]]
            puts [format "  %7d   %10d   %10f" [eval $qmonSource($i) set
pdepartures_] [eval $qmonSource($i) set bdepartures_] [expr [$qmonSource($i)
set bdepartures_]/$MAX_TIME*8.0/1000000.0]]
        }
        #sinks
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts -nonewline [format "Queue qmonSink($i):   %7d   %10d   %10f"
[eval $qmonSink($i) set parrivals_] [eval $qmonSink($i) set barrivals_] [expr
[$qmonSink($i) set barrivals_]/$MAX_TIME*8.0/1000000.0]]
            puts -nonewline [format "  %7d   %10d" [eval $qmonSink($i) set
pdrops_] [eval $qmonSink($i) set bdrops_]]
            puts [format "  %7d   %10d   %10f" [eval $qmonSink($i) set
pdepartures_] [eval $qmonSink($i) set bdepartures_] [expr [$qmonSink($i) set
bdepartures_]/$MAX_TIME*8.0/1000000.0]]
        }
        #others
        set j 0
        foreach i {$qmonSwitch1_10 $qmonSwitch10_out $qmonConcur_Switch10
$qmonSwitch10_SinkConcur} {
            puts -nonewline [format "Queue $i:   %7d   %10d   %10f" [eval $i
set parrivals_] [eval $i set barrivals_] [expr [eval $i set
barrivals_]/$MAX_TIME*8.0/1000000.0]]
            puts -nonewline [format "  %7d   %10d" [eval $i set pdrops_]
[eval $i set bdrops_]]
            puts [format "  %7d   %10d   %10f" [eval $i set pdepartures_]
[eval $i set bdepartures_] [expr [eval $i set
bdepartures_]/$MAX_TIME*8.0/1000000.0]]
        }

        #average queue sizes
        puts "\nAverage Queue Size:        Packets         Bytes    sum_
(pkts)"
        puts "-------------------------------------------------------------------
--------"

        #sources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            set bytesInt($i) [eval $qmonSource($i) get-bytes-integrator]
            set pktsInt($i) [eval $qmonSource($i) get-pkts-integrator]
```

```
            set avg_queue_b($i) [expr [$bytesInt($i) set sum_]/$MAX_TIME]
            set avg_queue_p($i) [expr [$pktsInt($i) set sum_]/$MAX_TIME]
            puts [format "Queue Source($i):          %5.2f      %8.2f
[$pktsInt($i) set sum_]" $avg_queue_p($i) $avg_queue_b($i)]
        }


        #sinks
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            set bytesInt($i) [eval $qmonSink($i) get-bytes-integrator]
            set pktsInt($i) [eval $qmonSink($i) get-pkts-integrator]
            set avg_queue_b($i) [expr [$bytesInt($i) set sum_]/$MAX_TIME]
            set avg_queue_p($i) [expr [$pktsInt($i) set sum_]/$MAX_TIME]
            puts [format "Queue Sink($i):          %5.2f      %8.2f
[$pktsInt($i) set sum_]" $avg_queue_p($i) $avg_queue_b($i)]
        }

        #others
        set j 0
        foreach i {$qmonSwitch1_10 $qmonSwitch10_out $qmonConcur_Switch10
$qmonSwitch10_SinkConcur} {
            incr j
            set bytesInt($j) [eval $i get-bytes-integrator]
            set pktsInt($j) [eval $i get-pkts-integrator]
            set avg_queue_b($j) [expr [$bytesInt($j) set sum_]/$MAX_TIME]
            set avg_queue_p($j) [expr [$pktsInt($j) set sum_]/$MAX_TIME]
            puts [format "Queue $i:          %5.2f      %8.2f
[$pktsInt($j) set sum_]" $avg_queue_p($j) $avg_queue_b($j)]
        }
        #-----------------------
        #outputting to csv files
        #queue sums stats
        puts $queueStats "Simulation time;$MAX_TIME"
        puts $queueStats
"node;arrived_packets;arrived_bytes;arrived_throughputMbps;lost_packets;lost_
bytes;departed_packets;departed_bytes;departed_throughputMbps"

        #sources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts $queueStats "qmonSource($i);[eval $qmonSource($i) set
parrivals_];[eval $qmonSource($i) set barrivals_];[expr [$qmonSource($i) set
barrivals_]/$MAX_TIME*8.0/1000000.0];[eval $qmonSource($i) set pdrops_];[eval
$qmonSource($i) set bdrops_];[eval $qmonSource($i) set pdepartures_];[eval
$qmonSource($i) set bdepartures_];[expr [$qmonSource($i) set
bdepartures_]/$MAX_TIME*8.0/1000000.0]"
        }
        #sinks
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts $queueStats "qmonSink($i);[eval $qmonSink($i) set
parrivals_];[eval $qmonSink($i) set barrivals_];[expr [$qmonSink($i) set
barrivals_]/$MAX_TIME*8.0/1000000.0];[eval $qmonSink($i) set pdrops_];[eval
$qmonSink($i) set bdrops_];[eval $qmonSink($i) set pdepartures_];[eval
$qmonSink($i) set bdepartures_];[expr [$qmonSink($i) set
bdepartures_]/$MAX_TIME*8.0/1000000.0]"
        }
        #others
        set j 0
```

```
        foreach i {$qmonSwitch1_10 $qmonSwitch10_out $qmonConcur_Switch10
$qmonSwitch10_SinkConcur} {
            puts $queueStats "$i;[eval $i set parrivals_];[eval $i set
barrivals_];[expr [eval $i set barrivals_]/$MAX_TIME*8.0/1000000.0];[eval $i
set pdrops_];[eval $i set bdrops_];[eval $i set pdepartures_];[eval $i set
bdepartures_];[expr [eval $i set bdepartures_]/$MAX_TIME*8.0/1000000.0]"
        }

        #average queue sizes
        puts $queueAvgStats
"queue;average_queue_size_packets;average_queue_size_bytes;sum_pkts"
        #sources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            set bytesInt($i) [eval $qmonSource($i) get-bytes-integrator]
            set pktsInt($i) [eval $qmonSource($i) get-pkts-integrator]
            set avg_queue_b($i) [expr [$bytesInt($i) set sum_]/$MAX_TIME]
            set avg_queue_p($i) [expr [$pktsInt($i) set sum_]/$MAX_TIME]
            puts $queueAvgStats
"Source($i);$avg_queue_p($i);$avg_queue_b($i);[$pktsInt($i) set sum_]"
        }


        #sinks
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            set bytesInt($i) [eval $qmonSink($i) get-bytes-integrator]
            set pktsInt($i) [eval $qmonSink($i) get-pkts-integrator]
            set avg_queue_b($i) [expr [$bytesInt($i) set sum_]/$MAX_TIME]
            set avg_queue_p($i) [expr [$pktsInt($i) set sum_]/$MAX_TIME]
            puts $queueAvgStats
"Sink($i);$avg_queue_p($i);$avg_queue_b($i);[$pktsInt($i) set sum_]"
        }

        #others
        set j 0
        foreach i {$qmonSwitch1_10 $qmonSwitch10_out $qmonConcur_Switch10
$qmonSwitch10_SinkConcur} {
            incr j
            set bytesInt($j) [eval $i get-bytes-integrator]
            set pktsInt($j) [eval $i get-pkts-integrator]
            set avg_queue_b($j) [expr [$bytesInt($j) set sum_]/$MAX_TIME]
            set avg_queue_p($j) [expr [$pktsInt($j) set sum_]/$MAX_TIME]
            puts $queueAvgStats
"$i;$avg_queue_p($j);$avg_queue_b($j);[$pktsInt($j) set sum_]"
        }
        #--------------------
    }
    #------------ end of queue statistics generation ------------------

    #------ Print simulation parameters -------

    #print simulation time
    puts $simParams "Simulation time:  $MAX_TIME s\n"

    #print queue limits
    puts $simParams "Queue Limits:"
    puts $simParams "Queue                          Limit (packets)"
    puts $simParams "-----------------------------------------------"
```

```
    foreach i {$tcpSourceQueueLimit $tcpConcurTrafficQueueLimit
$switch1GbpsQueueLimit $switch10GbpsQueueLimit} {
        puts $simParams [format "%-28s:     %5d" $i [expr $i]]
    }

    #print TCP sources configuration parameters
    puts $simParams "\n\nTCP Configuration Parameters:"
    puts $simParams "node               window_    windowInit_   packetSize_
tcpTick_   maxburst_    maxcwnd_"
    puts $simParams "----------------------------------------------------------
---------------------------------"
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        puts $simParams [format "tcpSource($i):        %5d          %5d
%5d       %5.3f      %6d      %6d" [$tcpSource($i) set window_]
[$tcpSource($i) set windowInit_] [$tcpSource($i) set packetSize_]
[$tcpSource($i) set tcpTick_] [$tcpSource($i) set maxburst_] [$tcpSource($i)
set maxcwnd_]]
    }
    puts $simParams [format "tcpConcurTraffic:   %6d          %5d         %5d
%5.3f      %6d      %6d" [$tcpConcurTraffic set window_] [$tcpConcurTraffic
set windowInit_] [$tcpConcurTraffic set packetSize_] [$tcpConcurTraffic set
tcpTick_] [$tcpConcurTraffic set maxburst_] [$tcpConcurTraffic set maxcwnd_]]

    #------ end of printing simulation parameters ------

    if {$traceAll} {
        puts "Generating nam..."
        exec nam out.nam &
        #puts "Generating xgraph..."
        #exec xgraph throughputs.csv -geometry 800x600 &
    }
    exit 0
}

$ns run
```

Table 9: Tcl script for topology 2 (without intermediate 1 Gbps switch).

```
#This simulation investigation intends to observe the behavior of a number of
TCP flows
#when traversing through switches of different bandwidths and different
buffer sizes.
#
#The question is:  is TCP performance better when going through a slower
switch, but with
#larger buffers, then through a higher-speed switch with small buffers, or is
TCP
#performance better when going through a higher speed switch only with small
buffers?
#
#In this simulation, we will use 10 Gbps switches only.  Buffer sizes of
#each will be modified.
#
#Observe an important comment about ns behavior regarding the TCP variables
cwnd_ and
```

```
#window_, in the TCP source creation section further below.
#
# August 2010 Marcos Portnoi.

# --- Random Numbers and Seed
#Seed default random number generator with current time
global defaultRNG
$defaultRNG seed 0

#Number of TCP traffic sources (used to create traffic generators, nodes,
etc.)
set tcpGenerators 4

#Turn tracing on=1 or off=0
#be careful:  ns can generate enormous trace files
set traceAll 0
set traceQueue 1

#Queue size limits; ns default is 50
set tcpSourceQueueLimit 50
set tcpConcurTrafficQueueLimit 50
set switch10GbpsQueueLimit 50
#set switch1GbpsQueueLimit 1000

#Create simulator object
set ns [new Simulator]
set MAX_TIME 500.0; #set maximum simulation time; use at least a decimal to
prevent this constant being interpreted as a integer

# --- Define color index (class color)
# list of colors available at ~ns/man/mann/colors.n
$ns color 0 red
$ns color 1 blue
$ns color 2 chocolate
$ns color 3 yellow
$ns color 4 green
$ns color 5 tan
$ns color 6 gold
$ns color 7 black
$ns color 8 white
$ns color 9 darkblue
$ns color 10 cyan
$ns color 11 magenta
$ns color 12 orange
$ns color 13 bisque
$ns color 14 purple
$ns color 15 coral
$ns color 16 grey
$ns color 17 khaki
$ns color 18 AntiqueWhite
$ns color 19 aquamarine
$ns color 20 azure
$ns color 21 DarkSalmon
$ns color 22 DarkSeaGreen
$ns color 23 firebrick

#Use dinamic routing
```

```
#$ns rtproto DV

#Open files for trace if corresponding flags are set
if {$traceAll} {
    set nf [open out.nam w]
    set tr [open out.tr w]
    $ns namtrace-all $nf
    $ns trace-all $tr
}
if {$traceQueue} {
    set queueStats [open queueStats.csv w]
    set queueAvgStats [open queueAvgStats.csv w]
}

set throughputs [open throughputs.csv w]
set tcpVariablesTrace [open tcpVariablesTrace.csv w]
set simParams [open simParams.txt w]

#================================================================================
#Create nodes
#create main TCP sources and sinks
for {set i 0} {$i < $tcpGenerators} {incr i} {
    set source($i) [$ns node]
    set sink($i) [$ns node]
}

#set switch1Gbps [$ns node]
set switch10Gbps [$ns node]
set switch10GbpsOut [$ns node]
#Concurrent traffic
set concurTraffic [$ns node]
set sinkConcurTraffic [$ns node]

#Connect nodes
#default queue size limit is 50
for {set i 0} {$i < $tcpGenerators} {incr i} {
    $ns duplex-link $source($i) $switch10Gbps 10000Mb 1ms DropTail
    $ns queue-limit $source($i) $switch10Gbps $tcpSourceQueueLimit
    $ns duplex-link $switch10GbpsOut $sink($i) 10000Mb 1ms DropTail
    $ns queue-limit $switch10GbpsOut $sink($i) $switch10GbpsQueueLimit
}

#$ns duplex-link $switch1Gbps $switch10Gbps 1000Mb 1ms DropTail
#$ns queue-limit $switch1Gbps $switch10Gbps $switch1GbpsQueueLimit
$ns duplex-link $switch10Gbps $switch10GbpsOut 10000Mb 10ms DropTail
$ns queue-limit $switch10Gbps $switch10GbpsOut $switch10GbpsQueueLimit
$ns duplex-link $concurTraffic $switch10Gbps 10000Mb 10ms DropTail
$ns queue-limit $concurTraffic $switch10Gbps $tcpConcurTrafficQueueLimit
$ns duplex-link $switch10GbpsOut $sinkConcurTraffic 10000Mb 1ms DropTail
$ns queue-limit $switch10GbpsOut $sinkConcurTraffic $switch10GbpsQueueLimit

#Set visual orientation for nam
for {set i 0} {$i < $tcpGenerators} {incr i} {
    $ns duplex-link-op $source($i) $switch10Gbps queuePos 0.5
    $ns duplex-link-op $switch10GbpsOut $sink($i) queuePos 0.5
}
#$ns duplex-link-op $switch1Gbps $switch10Gbps queuePos 0.5
```

```
$ns duplex-link-op $switch10Gbps $switch10GbpsOut queuePos 0.5
$ns duplex-link-op $concurTraffic $switch10Gbps queuePos 0.5
$ns duplex-link-op $switch10GbpsOut $sinkConcurTraffic queuePos 0.0

#modify here for automatic placement of n tcp sources
$ns duplex-link-op $source(0) $switch10Gbps orient 1.67
$ns duplex-link-op $source(1) $switch10Gbps orient 1.83
$ns duplex-link-op $source(2) $switch10Gbps orient 0.17
$ns duplex-link-op $source(3) $switch10Gbps orient 0.33
#$ns duplex-link-op $switch1Gbps $switch10Gbps orient 0.0
$ns duplex-link-op $switch10Gbps $switch10GbpsOut orient 0.0
#modify here for automatic placement of n tcp sinks
$ns duplex-link-op $switch10GbpsOut $sink(0) orient 0.33
$ns duplex-link-op $switch10GbpsOut $sink(1) orient 0.17
$ns duplex-link-op $switch10GbpsOut $sink(2) orient 1.83
$ns duplex-link-op $switch10GbpsOut $sink(3) orient 1.67
$ns duplex-link-op $concurTraffic $switch10Gbps orient 1.5
$ns duplex-link-op $switch10GbpsOut $sinkConcurTraffic orient 1.5

#================================================================================
============================
#Create TCP agents and TCPSinks, attach them to nodes and link them
#
#Also, set here the maximum window_ size (cwnd_ or congestion window) for TCP
#Here is the catch:  ns allows TCP algorithm to grow cnwd accordingly if
there is no congestion.  The
#TCP variables trace file can demonstrate that.  However, the maximum burst
of TCP PDUs (segments) is
#limited to the maximum (advertised) window_ size, regardless whether cwnd_
is larger.  Thus, if in a specific moment,
#cwnd_ value is 2000 and window_ is set to 1000, TCP will send a burst of
only 1000 PDUs and then wait for
#the corresponding ACKs (since the max advertised window_ size is 1000).
#
for {set i 0} {$i < $tcpGenerators} {incr i} {
    set connection_list [$ns create-connection-list TCP/Reno $source($i)
TCPSink $sink($i) $i]
    set tcpSource($i) [lindex $connection_list 0]
    set tcpSink($i) [lindex $connection_list 1]
    $tcpSource($i) set fid_ $i
    $tcpSink($i) set fid_ $i
    $tcpSource($i) set window_ 64000
    #$tcpSource($i) set windowInit_ 64000
    #$tcpSource($i) set packetSize_ 8000
    $tcpSource($i) set maxburst_ 64000
    $tcpSink($i) set window_ 64000
}
#flowID for Concurrent traffic
set concurId 100
set connection_list [$ns create-connection-list TCP/Reno $concurTraffic
TCPSink $sinkConcurTraffic $concurId]
set tcpConcurTraffic [lindex $connection_list 0]
set tcpSinkConcurTraffic [lindex $connection_list 1]
$tcpConcurTraffic set fid_ $concurId
$tcpSinkConcurTraffic set fid_ $concurId
$tcpConcurTraffic set window_ 64000
#$tcpConcurTraffic set windowInit_ 64000
```

```
#$tcpConcurTraffic set packetSize_ 8000
$tcpConcurTraffic set maxburst_ 64000
$tcpSinkConcurTraffic set window_ 64000

#Create FTP applications and attach them to agents
for {set i 0} {$i < $tcpGenerators} {incr i} {
    set ftp($i) [new Application/FTP]
    $ftp($i) attach-agent $tcpSource($i)
}
set ftpConcur [new Application/FTP]
$ftpConcur attach-agent $tcpConcurTraffic


#Create queue monitors if corresponding trace flag is set
if {$traceQueue} {
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        set qmonSource($i) [$ns monitor-queue $source($i) $switch10Gbps ""]
        set qmonSink($i) [$ns monitor-queue $switch10GbpsOut $sink($i) ""]
    }
    #set qmonSwitch1_10 [$ns monitor-queue $switch1Gbps $switch10Gbps ""]
    set qmonSwitch10_out [$ns monitor-queue $switch10Gbps $switch10GbpsOut
""]
    set qmonConcur_Switch10 [$ns monitor-queue $concurTraffic $switch10Gbps
""]
    set qmonSwitch10_SinkConcur [$ns monitor-queue $switch10GbpsOut
$sinkConcurTraffic ""]
}

#Create statistics record procedure
proc record {time} {
    global tcpSource tcpConcurTraffic tcpSink tcpSinkConcurTraffic
throughputs tcpGenerators tcpVariablesTrace
    #Get instance of simulator
    set ns [Simulator instance]

    #Get current time
    set now [$ns now]
    puts "Recording at $now s..."

    #print header only once to sink throughputs file
    if {$now == 0} {
        puts $throughputs "Throughput MBps"
        puts -nonewline $throughputs "time;"
        #tcpSources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts -nonewline $throughputs "tcpSink($i);"
        }
        #concurrent traffic sink
        puts $throughputs "tcpSinkConcurTraffic"
    }

    #How many bytes have been received by traffic sinks?
    # Problem here is that bytes_ in tcp-sink.cc is a INT value... max is
2147483647, and this is easily surpassed in gigabit links; bytes_ then WRAPS
    # To avoid this for 10Gbps of throughput, we can use the relation:
    # time = max_int_bytes_ / throughput
    # where time is the time step or resolution where bytes_ should be read
```

```
from TCPSink, and then set to zero again.
    # Thus, for the usual max_int, we have:
    # time = 2147483647 / 10,000,000,000 * 8 (multiply by 8 to equalize the
units to bytes) = 1.7179 s
    # Anything below that value should work accordingly, without integer
wrapping.
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        set bw($i) [$tcpSink($i) set bytes_]
    }
    set bwConcur [$tcpSinkConcurTraffic set bytes_]

    #Calculate bandwidth (MBit/s) and write to file
    puts -nonewline $throughputs "$now;"
    puts -nonewline "$now "
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        puts -nonewline $throughputs "[expr $bw($i)/$time*8.0/1000000.0];"
        puts -nonewline "[expr $bw($i)/$time*8.0/1000000.0];"
    }
    #concurrent traffic bandwidth (MBit/s)
    puts $throughputs "[expr $bwConcur/$time*8.0/1000000.0]"
    puts "[expr $bwConcur/$time*8.0/1000000.0]"

    #Reset bytes_ values on traffic sinks
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        $tcpSink($i) set bytes_ 0
    }
    $tcpSinkConcurTraffic set bytes_ 0

    #record TCP variables
    #print header only once to tcpVariablesTrace file
    if {$now == 0} {
        puts $tcpVariablesTrace
"node;time;awnd_;rttvar_;backoff_;cwnd_;ssthresh_;rtt_;srtt_"
    }
    #tcpSources
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        puts $tcpVariablesTrace "tcpSource($i);$now;[$tcpSource($i) set
awnd_];[$tcpSource($i) set rttvar_];[$tcpSource($i) set
backoff_];[$tcpSource($i) set cwnd_];[$tcpSource($i) set
ssthresh_];[$tcpSource($i) set rtt_];[$tcpSource($i) set srtt_]"
    }
    #tcp concurrent traffic
    puts $tcpVariablesTrace "tcpConcurTraffic;$now;[$tcpConcurTraffic set
awnd_];[$tcpConcurTraffic set rttvar_];[$tcpConcurTraffic set
backoff_];[$tcpConcurTraffic set cwnd_];[$tcpConcurTraffic set
ssthresh_];[$tcpConcurTraffic set rtt_];[$tcpConcurTraffic set srtt_]"

    #Re-schedule procedure
    $ns at [expr $now+$time] "record $time"
}

#Schedule
$ns at 0.0 "record 1.5"
$ns at 0.0 "puts \"Starting simulation...\""
#ftp generators start synchronized!
for {set i 0} {$i < $tcpGenerators} {incr i} {
    #$ns at [expr 0.0 + $i] "$ftp($i) start"
```

```
    $ns at 0.0 "$ftp($i) start"
}
$ns at 0.0 "$ftpConcur start"
for {set i 0} {$i < $tcpGenerators} {incr i} {
    $ns at $MAX_TIME "$ftp($i) stop"
}
$ns at $MAX_TIME "$ftpConcur stop"
$ns at $MAX_TIME "finish"

#Finish procedure
proc finish {} {
    global ns nf tr throughputs tcpGenerators queueStats queueAvgStats
tcpSource tcpConcurTraffic simParams
    global qmonSource qmonSink
    #global qmonSwitch1_10
    global qmonSwitch10_out qmonConcur_Switch10 qmonSwitch10_SinkConcur
    global tcpSourceQueueLimit tcpConcurTrafficQueueLimit
    #global switch1GbpsQueueLimit
    global switch10GbpsQueueLimit
    global MAX_TIME traceAll traceQueue
    if {$traceAll} {
        $ns flush-trace
        #close trace files
        close $nf
        close $tr
    }
    close $throughputs
    puts "Ending simulation.  Simulation time:  $MAX_TIME s"
    puts "                                      ====="

    #---------------- generating queue statistics ----------------------

    #outputting queue stats to screen, if trace flag is set
    #queue sums stats
    if {$traceQueue} {
        puts "\nStatistics:"
        puts "                                  Arrived
Lost                    Departed"
        puts "                        Packets      Bytes    Throughput(Mbps)
Packets     Bytes    Packets     Bytes    Throughput(Mbps)"
        puts "-------------------------------------------------------------
-------------------"

        #sources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts -nonewline [format "Queue qmonSource($i):   %7d   %10d
%10f" [eval $qmonSource($i) set parrivals_] [eval $qmonSource($i) set
barrivals_] [expr [$qmonSource($i) set barrivals_]/$MAX_TIME*8.0/1000000.0]]
            puts -nonewline [format "  %7d   %10f" [eval $qmonSource($i) set
pdrops_] [eval $qmonSource($i) set bdrops_]]
            puts [format "  %7d   %10d   %10f" [eval $qmonSource($i) set
pdepartures_] [eval $qmonSource($i) set bdepartures_] [expr [$qmonSource($i)
set bdepartures_]/$MAX_TIME*8.0/1000000.0]]
        }
        #sinks
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts -nonewline [format "Queue qmonSink($i):   %7d   %10d   %10f"
```

```
[eval $qmonSink($i) set parrivals_] [eval $qmonSink($i) set barrivals_] [expr
[$qmonSink($i) set barrivals_]/$MAX_TIME*8.0/1000000.0]]
            puts -nonewline [format "  %7d   %10d" [eval $qmonSink($i) set
pdrops_] [eval $qmonSink($i) set bdrops_]]
            puts [format "  %7d   %10d   %10f" [eval $qmonSink($i) set
pdepartures_] [eval $qmonSink($i) set bdepartures_] [expr [$qmonSink($i) set
bdepartures_]/$MAX_TIME*8.0/1000000.0]]
        }
        #others
        set j 0
        foreach i {$qmonSwitch10_out $qmonConcur_Switch10
$qmonSwitch10_SinkConcur} {
            puts -nonewline [format "Queue $i:   %7d   %10d   %10f" [eval $i
set parrivals_] [eval $i set barrivals_] [expr [eval $i set
barrivals_]/$MAX_TIME*8.0/1000000.0]]
            puts -nonewline [format "  %7d   %10d" [eval $i set pdrops_]
[eval $i set bdrops_]]
            puts [format "  %7d   %10d   %10f" [eval $i set pdepartures_]
[eval $i set bdepartures_] [expr [eval $i set
bdepartures_]/$MAX_TIME*8.0/1000000.0]]
        }

        #average queue sizes
        puts "\nAverage Queue Size:        Packets         Bytes    sum_
(pkts)"
        puts "-------------------------------------------------------------
--------"

        #sources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            set bytesInt($i) [eval $qmonSource($i) get-bytes-integrator]
            set pktsInt($i) [eval $qmonSource($i) get-pkts-integrator]
            set avg_queue_b($i) [expr [$bytesInt($i) set sum_]/$MAX_TIME]
            set avg_queue_p($i) [expr [$pktsInt($i) set sum_]/$MAX_TIME]
            puts [format "Queue Source($i):          %5.2f      %8.2f
[$pktsInt($i) set sum_]" $avg_queue_p($i) $avg_queue_b($i)]
        }


        #sinks
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            set bytesInt($i) [eval $qmonSink($i) get-bytes-integrator]
            set pktsInt($i) [eval $qmonSink($i) get-pkts-integrator]
            set avg_queue_b($i) [expr [$bytesInt($i) set sum_]/$MAX_TIME]
            set avg_queue_p($i) [expr [$pktsInt($i) set sum_]/$MAX_TIME]
            puts [format "Queue Sink($i):          %5.2f      %8.2f
[$pktsInt($i) set sum_]" $avg_queue_p($i) $avg_queue_b($i)]
        }

        #others
        set j 0
        foreach i {$qmonSwitch10_out $qmonConcur_Switch10
$qmonSwitch10_SinkConcur} {
            incr j
            set bytesInt($j) [eval $i get-bytes-integrator]
            set pktsInt($j) [eval $i get-pkts-integrator]
            set avg_queue_b($j) [expr [$bytesInt($j) set sum_]/$MAX_TIME]
```

```
            set avg_queue_p($j) [expr [$pktsInt($j) set sum_]/$MAX_TIME]
            puts [format "Queue $i:          %5.2f      %8.2f
[$pktsInt($j) set sum_]" $avg_queue_p($j) $avg_queue_b($j)]
        }
        #-----------------------
        #outputting to csv files
        #queue sums stats
        puts $queueStats "Simulation time;$MAX_TIME"
        puts $queueStats
"node;arrived_packets;arrived_bytes;arrived_throughputMbps;lost_packets;lost_
bytes;departed_packets;departed_bytes;departed_throughputMbps"

        #sources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts $queueStats "qmonSource($i);[eval $qmonSource($i) set
parrivals_];[eval $qmonSource($i) set barrivals_];[expr [$qmonSource($i) set
barrivals_]/$MAX_TIME*8.0/1000000.0];[eval $qmonSource($i) set pdrops_];[eval
$qmonSource($i) set bdrops_];[eval $qmonSource($i) set pdepartures_];[eval
$qmonSource($i) set bdepartures_];[expr [$qmonSource($i) set
bdepartures_]/$MAX_TIME*8.0/1000000.0]"
        }
        #sinks
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            puts $queueStats "qmonSink($i);[eval $qmonSink($i) set
parrivals_];[eval $qmonSink($i) set barrivals_];[expr [$qmonSink($i) set
barrivals_]/$MAX_TIME*8.0/1000000.0];[eval $qmonSink($i) set pdrops_];[eval
$qmonSink($i) set bdrops_];[eval $qmonSink($i) set pdepartures_];[eval
$qmonSink($i) set bdepartures_];[expr [$qmonSink($i) set
bdepartures_]/$MAX_TIME*8.0/1000000.0]"
        }
        #others
        set j 0
        foreach i {$qmonSwitch10_out $qmonConcur_Switch10
$qmonSwitch10_SinkConcur} {
            puts $queueStats "$i;[eval $i set parrivals_];[eval $i set
barrivals_];[expr [eval $i set barrivals_]/$MAX_TIME*8.0/1000000.0];[eval $i
set pdrops_];[eval $i set bdrops_];[eval $i set pdepartures_];[eval $i set
bdepartures_];[expr [eval $i set bdepartures_]/$MAX_TIME*8.0/1000000.0]"
        }

        #average queue sizes
        puts $queueAvgStats
"queue;average_queue_size_packets;average_queue_size_bytes;sum_pkts"
        #sources
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            set bytesInt($i) [eval $qmonSource($i) get-bytes-integrator]
            set pktsInt($i) [eval $qmonSource($i) get-pkts-integrator]
            set avg_queue_b($i) [expr [$bytesInt($i) set sum_]/$MAX_TIME]
            set avg_queue_p($i) [expr [$pktsInt($i) set sum_]/$MAX_TIME]
            puts $queueAvgStats
"Source($i);$avg_queue_p($i);$avg_queue_b($i);[$pktsInt($i) set sum_]"
        }


        #sinks
        for {set i 0} {$i < $tcpGenerators} {incr i} {
            set bytesInt($i) [eval $qmonSink($i) get-bytes-integrator]
```

```
            set pktsInt($i) [eval $qmonSink($i) get-pkts-integrator]
            set avg_queue_b($i) [expr [$bytesInt($i) set sum_]/$MAX_TIME]
            set avg_queue_p($i) [expr [$pktsInt($i) set sum_]/$MAX_TIME]
            puts $queueAvgStats
"Sink($i);$avg_queue_p($i);$avg_queue_b($i);[$pktsInt($i) set sum_]"
        }

        #others
        set j 0
        foreach i {$qmonSwitch10_out $qmonConcur_Switch10
$qmonSwitch10_SinkConcur} {
            incr j
            set bytesInt($j) [eval $i get-bytes-integrator]
            set pktsInt($j) [eval $i get-pkts-integrator]
            set avg_queue_b($j) [expr [$bytesInt($j) set sum_]/$MAX_TIME]
            set avg_queue_p($j) [expr [$pktsInt($j) set sum_]/$MAX_TIME]
            puts $queueAvgStats
"$i;$avg_queue_p($j);$avg_queue_b($j);[$pktsInt($j) set sum_]"
        }
        #--------------------
    }
    #------------ end of queue statistics generation ------------------

    #------ Print simulation parameters -------

    #print simulation time
    puts $simParams "Simulation time:  $MAX_TIME s\n"

    #print queue limits
    puts $simParams "Queue Limits:"
    puts $simParams "Queue                          Limit (packets)"
    puts $simParams "-----------------------------------------------"
    foreach i {$tcpSourceQueueLimit $tcpConcurTrafficQueueLimit
$switch10GbpsQueueLimit} {
        puts $simParams [format "%-28s:     %5d" $i [expr $i]]
    }

    #print TCP sources configuration parameters
    puts $simParams "\n\nTCP Configuration Parameters:"
    puts $simParams "node               window_    windowInit_   packetSize_
tcpTick_   maxburst_    maxcwnd_"
    puts $simParams "-------------------------------------------------------
---------------------------------"
    for {set i 0} {$i < $tcpGenerators} {incr i} {
        puts $simParams [format "tcpSource($i):        %5d          %5d
%5d       %5.3f      %6d      %6d" [$tcpSource($i) set window_]
[$tcpSource($i) set windowInit_] [$tcpSource($i) set packetSize_]
[$tcpSource($i) set tcpTick_] [$tcpSource($i) set maxburst_] [$tcpSource($i)
set maxcwnd_]]
    }
    puts $simParams [format "tcpConcurTraffic:   %6d          %5d          %5d
%5.3f      %6d      %6d" [$tcpConcurTraffic set window_] [$tcpConcurTraffic
set windowInit_] [$tcpConcurTraffic set packetSize_] [$tcpConcurTraffic set
tcpTick_] [$tcpConcurTraffic set maxburst_] [$tcpConcurTraffic set maxcwnd_]]

    #------ end of printing simulation parameters ------
```

```
    if {$traceAll} {
        puts "Generating nam..."
        exec nam out.nam &
        #puts "Generating xgraph..."
        #exec xgraph throughputs.csv -geometry 800x600 &
    }
    exit 0
}

$ns run
```