KU Leuven

Assignment:

Network Simulation

Report

Authors:
Giuseppe Callari
Xavier Goás Aguililla

Professor:
prof. dr. D. Hughes

May 2nd, 2014

Contents

1 Exercise 1

1.1 Question 1 and 2

In figure 1 we can clearly see the difference between the scenario with both an uploader and a downloader and that without an uploader. In the scenario with an uploader, around the time the upload starts (3.0s), the throughput for the downloader drops drastically. In the scenario without an uploader, indicated in green, the throughput of the connection remains roughly constant.

The download speed is limited by the link bandwidth. A maximum throughput of about 475 packets per second is achieved, and we can quickly calculate using the FTP packet size that this comes down to about 475 packets/s · 1040 bytes/packet · 8 bits/byte ≈ 4Mbps, which corresponds to the downstream link bandwidth of the modem.

In figure 2 we see both the upload and download throughput on the same graph; we can clearly see the aforementioned drop in download speed at the moment the upload over CBR starts. The speed is throttled to about the same as the upload (256Kbps). This is due to the fact that both download packets destined for a node in the LAN and download request packets departing the LAN have to pass through the same node; both kinds of packets are generated at the same rate, but due to the upload taking up all the upstream bandwidth, the small request packets are transmitted at a much slower rate. Accordingly, the download transfers are also started at a slower rate, such that the download throughput is limited to that of the upload. In a real scenario, this would not happen, since only one node would be regulating the transfer; we do not have the bottleneck of having to use two modem nodes.

1.2 Question 3

We expect performance for CBR to be excellent, since a certain amount of bandwidth is always guaranteed for the upload connection. This does imply that the upload connection has a certain privilege over the download connection; to ensure this, we could tell the router to drop downstream packets rather than upstream packets. This would result in increased packet loss for the download connection, especially if we would run multiple upload connections at the same time.

1.3 Question 4

We can see in figures 3 and 4 that both the bridled and unbridled connection show similar patterns but differ at the tail end of the speed drop from 3.0s on. At 6.0s the CBR upload is stopped, but not all packets have been transferred yet due to packet loss.

The connection layer ensures these lost packets still arrive at their destination. To do this, the TCP connection persists while not all packets have arrived at their destination. Looking at figure 4, we see that the TCP upstream connection persists longer in the situation with the limited bandwidth. The packets transferred while the connection persists take up a certain amount of bandwidth. Only around the 9.5 second mark are all the packets transferred, and in figure 3 we can see that the FTP download is negatively affected by this transfer: the throughput is very low until the 9.5 second mark.

1.4 Question 5

When we impose a fixed bitrate on the CBR connection, we ensure that this connection never takes more than its preallocated share of bandwidth. Thus, when we allocate a small percentage of the total connection bandwidth, we ensure that other connections on the same link will not suffer from greatly limited performance, since TCP will not allocate more bandwidth than this fixed rate. TCP can then dynamically allocate bandwidth to other connections.

We set up a simulation in NS/2 and verified our assertion. We observed no packet loss, and only a very slight drop in throughput for the download connection in the period from 3.0s to 6.0s. The speed drop is due to packets from two different origins arriving at a router and being inserted into its buffer. Of course, in this scenario, packets cannot be immediately forwarded to their destination, causing a drop in throughput, which is small because of the limited amount of packets being sent by the CBR application.

1.5 Question 6

1.5.1 Question 6-1

A naive response would be to say that we expect performance to stay the same, since we have ten times the bandwidth for ten times the users. In actuality, this is not true. Consider the output of a simulation we made for this scenario.

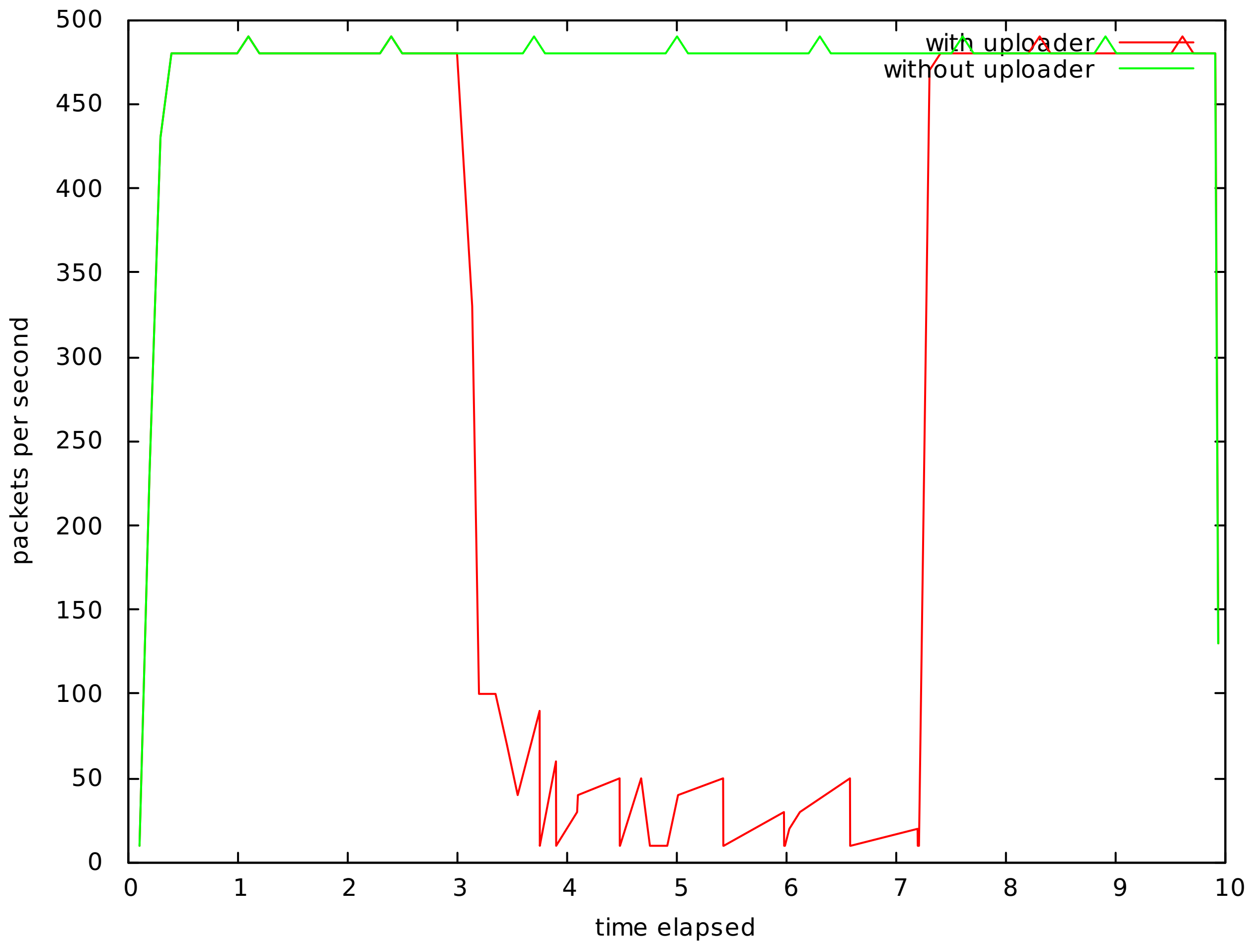

Figure 1: Download throughput compared

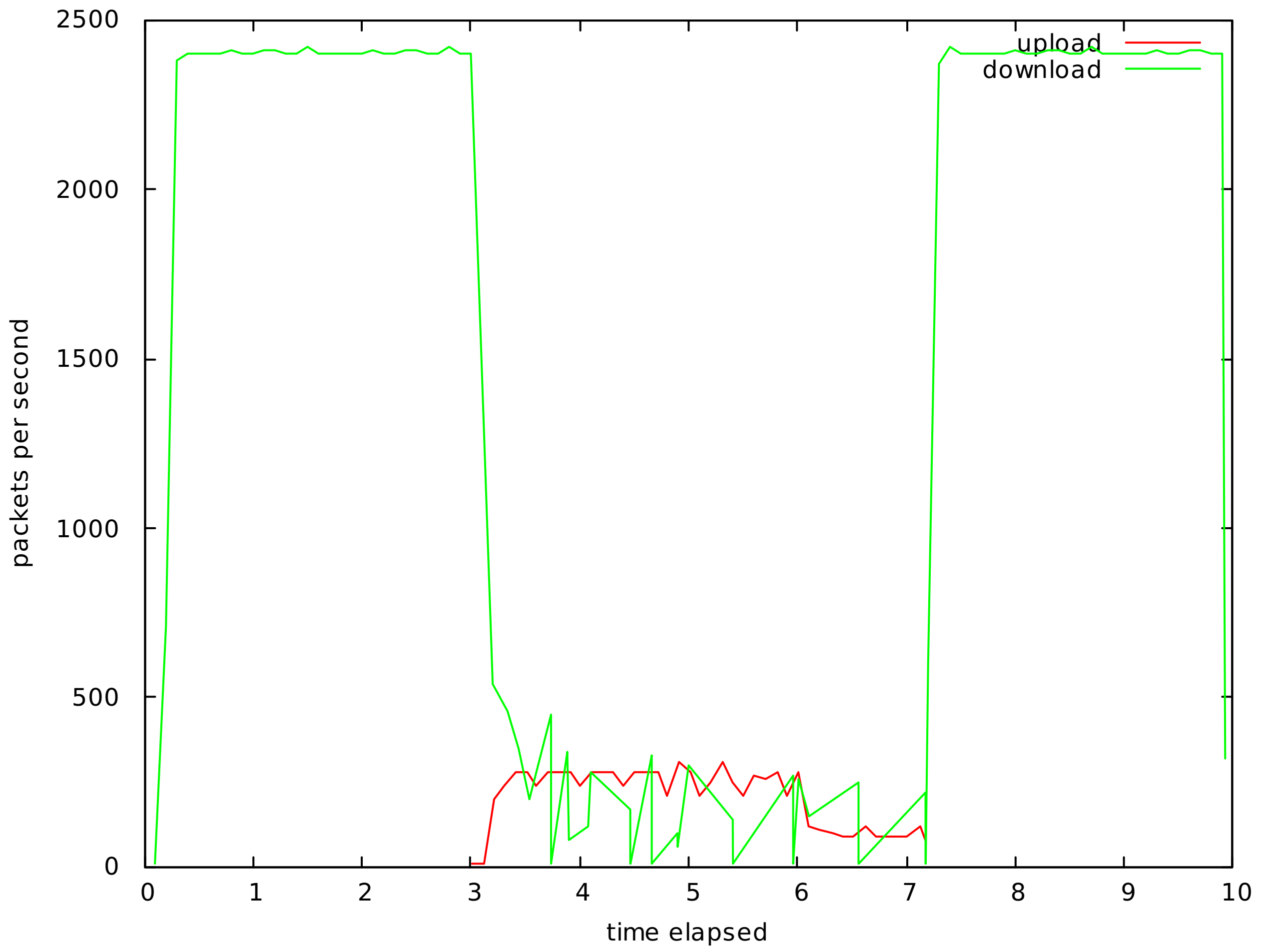

Figure 2: Upload and download throughput

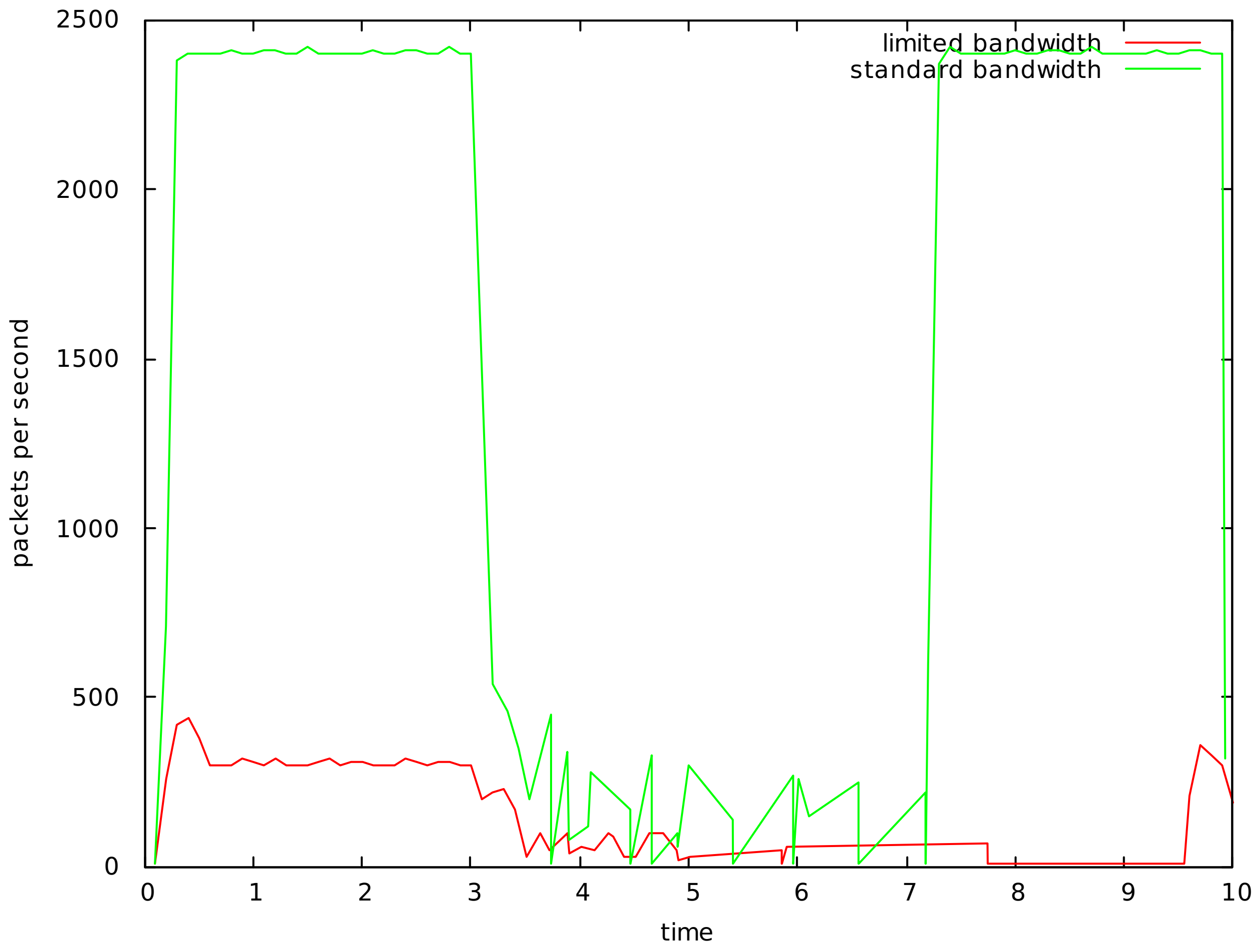

Figure 3: Download throughput

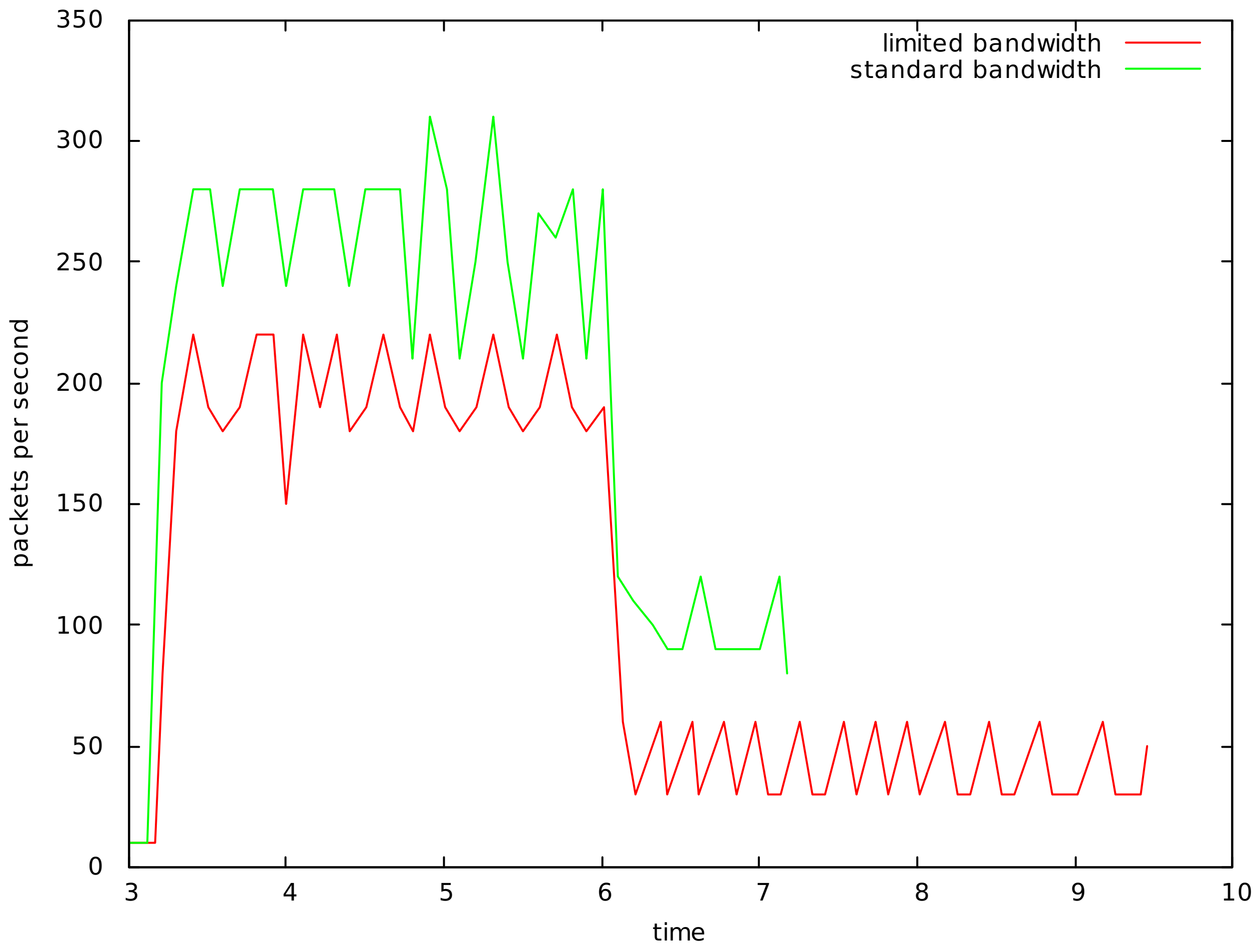

Figure 4: Upload throughput

We can clearly see that all transfers have very variable levels of throughput when executed simultaneously. The reason for this is that much more packets are being sent to the router simultaneously; the router can only process as much packets at a time as fit in its buffer, so once its buffer is full and more packets arrive, packet loss will occur. This is expressed in the valleys in the graph measuring the throughput.

If we have a smaller amount of nodes emitting packets, the probability of this scenario is also smaller. So, if we have five nodes, packet loss is less likely to occur than with ten nodes, and this will ensure a better throughput.

1.5.2 Question 6-2

If the same activities are performed at random times, performance will be better than if all activities are performed all at the same time. The reason for this is that any given point in time, there will be less than ten users performing the same activity simultaneously (at least, the probability of this scenario is overwhelmingly greater than the converse case).

We have made a small simulation for this case, whose source code is also contained in the listings.

2 Exercise 2

2.1 Question 1

In figure 5 we have plotted the throughput for both the long-standing FTP connection and the burst FTP connections. We observe the following:

1. We can clearly see sharp drops in throughput for the long-standing FTP connection at the 5.0, 10.0 and 15.0 second marks.
2. Throughput for the long-standing FTP connection increases once the two second mark after the start of each burst is passed.
3.

2.2 Questions 2 and 3

We can distinguish very clearly the three alternating phases of the algorithm in figure 6:

1. the multiplicative increase phase: this phase is entered at the start of the algorithm or after a multiplicative decrease phase. On the graph, we can see this as a exponential increase in the congestion window size from the start value. This is because the number of packets transmitted during a single transmission round is doubled after every transmission round.
2. the additive increase phase: this phase is entered once the congestion window size has reached the current threshold. From this point on, the window size is increased more slowly until packet loss occurs. On the graph, we can see a smooth increase in window size until . . .
3. the multiplicative decrease phase: starts when packet loss occurs. The congestion window is set to its start value (in this case, 1) and a new threshold is set, which is half of the previous maximum congestion window size (except for the first drop, where half of the initial threshold is chosen). On the graph, we can identify this as a very sharp drop in window size and a sudden lowering of the threshold.

Let us look more closely at the sawtooth pattern starting a little before the 12s mark and ending around the 20s mark. We can see that at the start of the pattern, the congestion window size has a value of 1. The multiplicative increase phase of the algorithm starts at this point. For a little while, the congestion window size increases until the threshold is reached around 12s. At this point, we can see the window size increase slow down, as is reflected by the smooth linear increase in the graph (congestion window increases by one every RTT); this is the additive increase phase. This continues until packet loss occurs around the 15.5s mark. At this point, the multiplicative decrease phase starts, in which congestion avoidance is active; the congestion window size drops sharply to its start value, which is clearly visible on the graph. Once this value is reached, a new 'sawtooth' pattern starts.

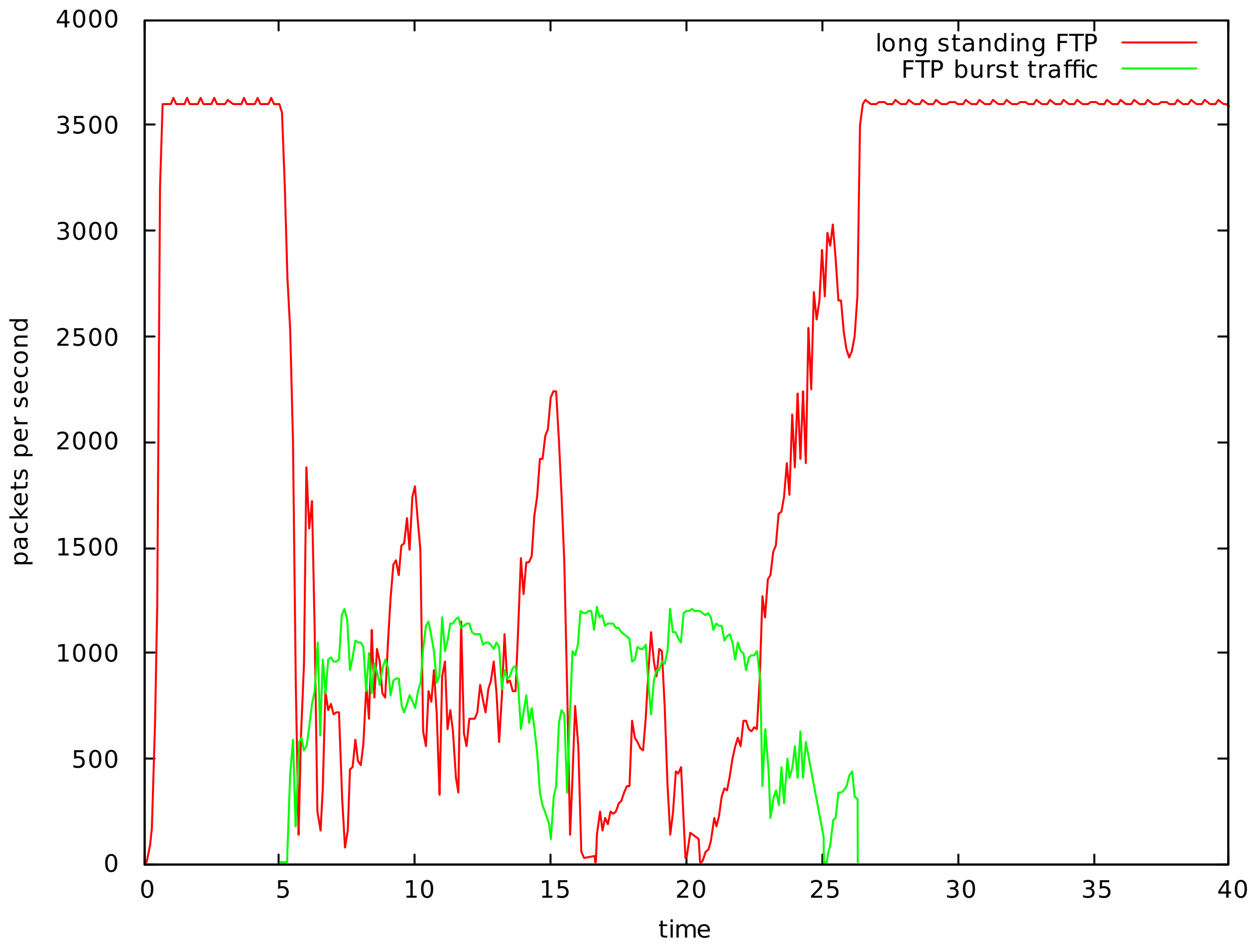

Figure 5: Throughput

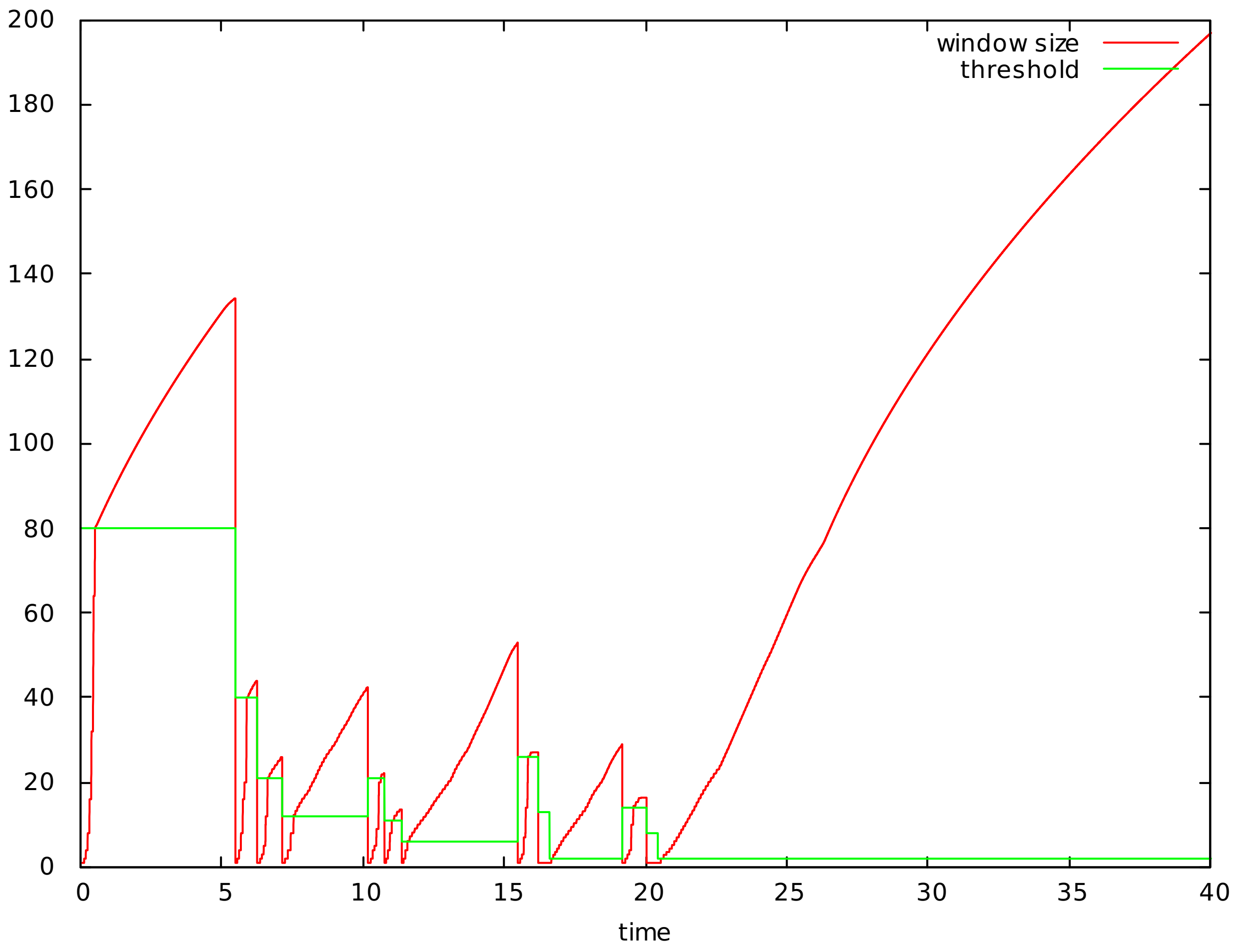

Figure 6: Congestion window size and slow start thresholds

2.3 Question 4

We can see the evolution of the congestion window size and threshold in figure 7. The figure largely resembles figure 6 except for the fact that the congestion window size is reset in two different ways: either it is reset to 0 or 1 when a packet times out, as in Tahoe, or it is reset to a newly calculated size when packet loss occurs. This size is half of the congestion window size when the loss occurred; once the congestion window is resized, the threshold is also set to this value. This is TCP Reno's 'fast recovery'.

For instance, a little after the 15s mark, a timeout occurs and the congestion window size drops to 1. From here on, it displays the same behavior as Tahoe until before the 20s mark, when packet loss occurs. The window size at this point is around 25; accordingly, the next window size will be 12, and the threshold is also set to this value.

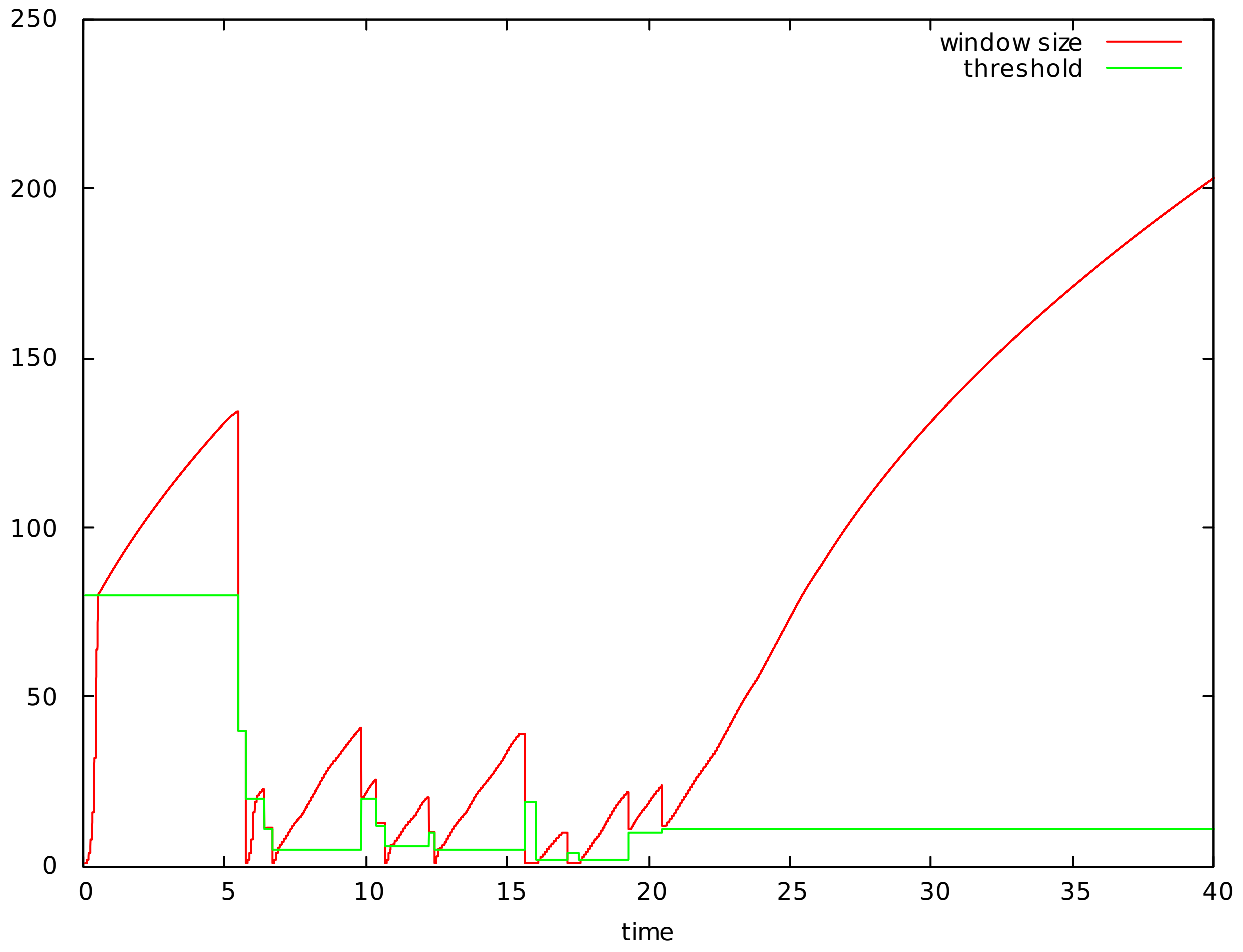

Figure 7: Congestion window size and slow start thresholds (with the TCP Reno algorithm)

3 Source code listings

Warning: we output trace files to stdout; this will mess with the terminal display on some systems due to the large amount of data emitted!

3.1 Base setups

3.1.1 Exercise 1

```
set ns [new Simulator]

#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        # $ns flush-trace
        close $tf
        close $nf
        exit 0
}

set lan_nodes_0 [$ns node]
set lan_nodes_1 [$ns node]
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

$ns duplex-link $lan_nodes_0 $lan_router 10Mb 10ms DropTail
$ns duplex-link $lan_nodes_1 $lan_router 10Mb 10ms DropTail

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 256Kb 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 4Mb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail


# TCP connection, KUL server is agent, node 1 is sink
set tcp_agent [new Agent/TCP]
$tcp_agent set fid_ 1
set tcp_sink [new Agent/TCPSink]

$ns attach-agent $kul_servers $tcp_agent
$ns attach-agent $lan_nodes_1 $tcp_sink

$ns connect $tcp_agent $tcp_sink
```

```
# UDP connection, node 0 is agent, Internet server is sink
set udp_agent [new Agent/UDP]
$udp_agent set fid_ 2
set udp_sink [new Agent/Null]

$ns attach-agent $lan_nodes_0 $udp_agent
$ns attach-agent $internet_servers $udp_sink

$ns connect $udp_agent $udp_sink

set ftp [new Application/FTP]
$ftp set type_ FTP
$ftp attach-agent $tcp_agent
$ftp set window_ 80
set cbr [new Application/Traffic/CBR]
$cbr set type_ CBR
$cbr attach-agent $udp_agent
$cbr set packetSize_ 1500

$ns at 0.1 "$ftp start"
$ns at 3.0 "$cbr start"
$ns at 6.0 "$cbr stop"
$ns at 9.9 "$ftp stop"
$ns at 10.0 "finish"

$ns run
```

3.1.2 Exercise 2

```
set ns [new Simulator]

#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#log file
set logger [open logger.tr w]
set filelogger [open files.tr w]

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf



proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}


array set lan_nodes {}
array set internet_nodes {}
set router [$ns node]
set modem [$ns node]
```

```
for {set i 0} {$i < 2} {incr i} {
        set lan_nodes($i) [$ns node]
        $ns duplex-link $lan_nodes($i) $router 10Mb 10ms DropTail
}

$ns duplex-link $router $modem 10Mb 10ms DropTail
$ns queue-limit $router $modem 20

for {set i 0} {$i < 2} {incr i} {
        set internet_nodes($i) [$ns node]
        $ns duplex-link $internet_nodes($i) $modem 10Mb 10ms DropTail
}


# TCP connection, KUL server is agent, node 1 is sink
set tcp_agent [new Agent/TCP]
$tcp_agent set fid_ 1
$tcp_agent set window_ 80
set tcp_sink [new Agent/TCPSink]

$ns attach-agent $internet_nodes(0) $tcp_agent
$ns attach-agent $lan_nodes(0) $tcp_sink

$ns connect $tcp_agent $tcp_sink

# FTP app
set ftp [new Application/FTP]
$ftp set type_ FTP
$ftp attach-agent $tcp_agent


## Crazy burst traffic simulator


# Generators for random size of files.
set rep 1
set rng1 [new RNG]
set rng2 [new RNG]
for {set i 0} {$i < $rep} {incr i} {
    $rng1 next-substream
    $rng2 next-substream
}

# Random size of files to transmit
set size_svar [new RandomVariable/Pareto]
$size_svar set avg_ 150000
$size_svar set shape_ 1.5
$size_svar use-rng $rng1

# Random times in two second interval
set time_svar [new RandomVariable/Exponential]
$time_svar set avg_ 0.05
$time_svar use-rng $rng2
# We now define the beginning times of transfers and the transfer sizes

#pg90 and 54

# Number of sources
set NodeNb 3
# Number of flows per source node
```

```
set NumberFlows 40
# TCP Sources , destinations , connections
for { set i 1} { $i <= $NodeNb } { incr i } {
        for { set j 1} { $j <= $NumberFlows } { incr j } {
                set tcpsrc($i,$j) [new Agent/TCP]
                set tcp_snk($i,$j) [new Agent/TCPSink]
                set k [expr $i * $NumberFlows + $j];
                $tcpsrc($i,$j) set fid_ $k
                $tcpsrc($i,$j) set window_ 2000
                $ns attach-agent $lan_nodes(1) $tcp_snk($i,$j)
                $ns attach-agent $internet_nodes(1) $tcpsrc($i,$j)
                $ns connect $tcpsrc($i,$j) $tcp_snk($i,$j)
                set ftp_array($i,$j) [$tcpsrc($i,$j) attach-source FTP]
                }
        }


# Arrivals of sessions follow a Poisson process.
# set the beginning time of next transfer from source and attributes
# update the number of flows
for {set i 1} {$i <=$NodeNb} {incr i } {
    set t [expr $i * 5.0]
    for {set j 1} {$j<=$NumberFlows} {incr j } {
        set size [expr [$size_svar value]]
        $ns at $t "$ftp_array($i,$j) send $size"
        puts $filelogger "$i-$j $t $size"
        set addedTime [$time_svar value]
        set t [expr $t + $addedTime]
        # puts "$t"
    }
}

# Printing the window size

proc printWindow {} {
        global ns tcp_agent logger
        set time 0.001
        set now [ $ns now ]
        set cwnd [ $tcp_agent set cwnd_ ]
        set ssthresh [ $tcp_agent set ssthresh_ ]
        puts $logger "$now $cwnd $ssthresh"
        $ns at [expr $now + $time] "printWindow"
}

$ns at 0.1 "printWindow"
$ns at 0.1 "$ftp start"
$ns at 40 "$ftp stop"
$ns at 40 "finish"
$ns run
```

3.2 Exercises

3.2.1 Exercise 1.1

```
set ns [new Simulator]

#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
```

```
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}

set lan_nodes_0 [$ns node]
$lan_nodes_0 label "oh_hai"
set lan_nodes_1 [$ns node]
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

$ns duplex-link $lan_nodes_0 $lan_router 10Mb 10ms DropTail
$ns duplex-link $lan_nodes_1 $lan_router 10Mb 10ms DropTail

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 256Kb 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 4Mb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail


# TCP connection, KUL server is agent, node 1 is sink
set tcp_agent [new Agent/TCP]
$tcp_agent set fid_ 1
set tcp_sink [new Agent/TCPSink]

$ns attach-agent $kul_servers $tcp_agent
$ns attach-agent $lan_nodes_1 $tcp_sink

$ns connect $tcp_agent $tcp_sink


set ftp [new Application/FTP]
$ftp set type_ FTP
$ftp attach-agent $tcp_agent
$ftp set window_ 80

$ns at 0.1 "$ftp_start"
$ns at 9.9 "$ftp_stop"
$ns at 10.0 "finish"

$ns run
```

3.2.2 Exercise 1.2

```
set ns [new Simulator]
```

```
#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}

set lan_nodes_0 [$ns node]
set lan_nodes_1 [$ns node]
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

$ns duplex-link $lan_nodes_0 $lan_router 10Mb 10ms DropTail
$ns duplex-link $lan_nodes_1 $lan_router 10Mb 10ms DropTail

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 256Kb 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 4Mb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail


# TCP connection, KUL server is agent, node 1 is sink
set tcp_agent [new Agent/TCP]
$tcp_agent set fid_ 1
set tcp_sink [new Agent/TCPSink]

$ns attach-agent $kul_servers $tcp_agent
$ns attach-agent $lan_nodes_1 $tcp_sink

$ns connect $tcp_agent $tcp_sink

# UDP connection,  node 0 is agent, Internet server is sink
set udp_agent [new Agent/UDP]
$udp_agent set fid_ 2
set udp_sink [new Agent/Null]

$ns attach-agent $lan_nodes_0 $udp_agent
$ns attach-agent $internet_servers $udp_sink

$ns connect $udp_agent $udp_sink

set ftp [new Application/FTP]
```

```
$ftp set type_ FTP
$ftp attach-agent $tcp_agent
$ftp set window_ 80

set cbr [new Application/Traffic/CBR]
$cbr set type_ CBR
$cbr attach-agent $udp_agent
$cbr set packetSize_ 1500

$ns at 0.1 "$ftp_start"
$ns at 3.0 "$cbr_start"
$ns at 6.0 "$cbr_stop"
$ns at 9.9 "$ftp_stop"
$ns at 10.0 "finish"

$ns run
```

3.2.3 Exercise 1.3

```
set ns [new Simulator]

#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}

set lan_nodes_0 [$ns node]
set lan_nodes_1 [$ns node]
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

$ns duplex-link $lan_nodes_0 $lan_router 10Mb 10ms DropTail
$ns duplex-link $lan_nodes_1 $lan_router 10Mb 10ms DropTail

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 256Kb 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 4Mb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail
```

```
# TCP connection, KUL server is agent, node 1 is sink
set tcp_agent [new Agent/TCP]
$tcp_agent set fid_ 1
set tcp_sink [new Agent/TCPSink]

$ns attach-agent $kul_servers $tcp_agent
$ns attach-agent $lan_nodes_1 $tcp_sink

$ns connect $tcp_agent $tcp_sink

# UDP connection,  node 0 is agent, Internet server is sink
set udp_agent [new Agent/UDP]
$udp_agent set fid_ 2
set udp_sink [new Agent/Null]

$ns attach-agent $lan_nodes_0 $udp_agent
$ns attach-agent $internet_servers $udp_sink

$ns connect $udp_agent $udp_sink

set ftp [new Application/FTP]
$ftp set type_ FTP
$ftp attach-agent $tcp_agent
$ftp set window_ 80
set cbr [new Application/Traffic/CBR]
$cbr set type_ CBR
$cbr attach-agent $udp_agent
$cbr set packetSize_ 1500

$ns at 0.1 "$ftp start"
$ns at 3.0 "$cbr start"
$ns at 6.0 "$cbr stop"
$ns at 9.9 "$ftp stop"
$ns at 10.0 "finish"

$ns run
```

3.2.4 Exercise 1.4

```
set ns [new Simulator]

#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}

set lan_nodes_0 [$ns node]
set lan_nodes_1 [$ns node]
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

$ns duplex-link $lan_nodes_0 $lan_router 10Mb 10ms DropTail
$ns duplex-link $lan_nodes_1 $lan_router 10Mb 10ms DropTail

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 100Kb 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 510Kb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail


# TCP connection, KUL server is agent, node 1 is sink
set tcp_agent [new Agent/TCP]
$tcp_agent set fid_ 1
set tcp_sink [new Agent/TCPSink]

$ns attach-agent $kul_servers $tcp_agent
$ns attach-agent $lan_nodes_1 $tcp_sink

$ns connect $tcp_agent $tcp_sink

# UDP connection,  node 0 is agent, Internet server is sink
set udp_agent [new Agent/UDP]
$udp_agent set fid_ 2
set udp_sink [new Agent/Null]

$ns attach-agent $lan_nodes_0 $udp_agent
$ns attach-agent $internet_servers $udp_sink

$ns connect $udp_agent $udp_sink

set ftp [new Application/FTP]
$ftp set type_ FTP
$ftp attach-agent $tcp_agent
$ftp set window_ 80
set cbr [new Application/Traffic/CBR]
$cbr set type_ CBR
$cbr attach-agent $udp_agent
$cbr set packetSize_ 1500

$ns at 0.1 "$ftp start"
$ns at 3.0 "$cbr start"
$ns at 6.0 "$cbr stop"
$ns at 9.9 "$ftp stop"
$ns at 10.0 "finish"

$ns run
```

3.2.5 Exercise 1.5

```
set ns [new Simulator]
```

```
#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}

set lan_nodes_0 [$ns node]
set lan_nodes_1 [$ns node]
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

$ns duplex-link $lan_nodes_0 $lan_router 10Mb 10ms DropTail
$ns duplex-link $lan_nodes_1 $lan_router 10Mb 10ms DropTail

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 256Kb 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 4Mb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail


# TCP connection, KUL server is agent, node 1 is sink
set tcp_agent [new Agent/TCP]
$tcp_agent set fid_ 1
set tcp_sink [new Agent/TCPSink]

$ns attach-agent $kul_servers $tcp_agent
$ns attach-agent $lan_nodes_1 $tcp_sink

$ns connect $tcp_agent $tcp_sink

# UDP connection,  node 0 is agent, Internet server is sink
set udp_agent [new Agent/UDP]
$udp_agent set fid_ 2
set udp_sink [new Agent/Null]

$ns attach-agent $lan_nodes_0 $udp_agent
$ns attach-agent $internet_servers $udp_sink

$ns connect $udp_agent $udp_sink

set ftp [new Application/FTP]
```

```
$ftp set type_ FTP
$ftp attach-agent $tcp_agent
$ftp set window_ 80

set cbr [new Application/Traffic/CBR]
$cbr set type_ CBR
$cbr attach-agent $udp_agent
$cbr set packetSize_ 1500
$cbr set rate_ 30Kb

$ns at 0.1 "$ftp start"
$ns at 3.0 "$cbr start"
$ns at 6.0 "$cbr stop"
$ns at 9.9 "$ftp stop"
$ns at 10.0 "finish"

$ns run
```

3.2.6 Exercise 1.6.1a

```
set ns [new Simulator]

#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}

array set lan_nodes {}
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

for {set i 0} {$i < 10} {incr i} {
        set lan_nodes($i) [$ns node]
        $ns duplex-link $lan_nodes($i) $lan_router 10Mb 10ms DropTail
}

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 2MB 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 40Mb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail
```

```
# TCP connection, node 0 is agent, Internet server is sink
array set tcp_agents {}
array set tcp_sinks {}

for {set i 0} {$i < 10} {incr i} {
    set tcp_agents($i) [new Agent/TCP]
    $ns attach-agent $kul_servers $tcp_agents($i)
    set tcp_sinks($i) [new Agent/TCPSink]
    $tcp_sinks($i) set fid_ $i
    $ns attach-agent $lan_nodes($i) $tcp_sinks($i)
    $ns connect $tcp_agents($i) $tcp_sinks($i)
}

# UDP connection, node 0 is agent, Internet server is sink
array set udp_agents {}
set udp_sink [new Agent/Null]
$ns attach-agent $internet_servers $udp_sink

for {set i 0} {$i < 10} {incr i} {
    set udp_agents($i) [new Agent/UDP]
    $udp_agents($i) set fid_ $i
    $ns attach-agent $lan_nodes($i) $udp_agents($i)
    $ns connect $udp_agents($i) $udp_sink
}


array set ftp {}
for {set i 0} {$i < 10} {incr i} {
    set ftp($i) [new Application/FTP]
    $ftp($i) set type_ FTP
    $ftp($i) attach-agent $tcp_agents($i)
    $ftp($i) set packetSize_ 1500
    $ftp($i) set window_ 80
}

array set cbr {}
for {set i 0} {$i < 10} {incr i} {
    set cbr($i) [new Application/Traffic/CBR]
    $cbr($i) set type_ CBR
    $cbr($i) attach-agent $udp_agents($i)
    $cbr($i) set packetSize_ 1500
}


for {set i 0} {$i < 10} {incr i} {
    $ns at 0.1 "$ftp($i) start"
    $ns at 3.0 "$cbr($i) start"
    $ns at 6.0 "$cbr($i) stop"
    $ns at 9.9 "$ftp($i) stop"
}
$ns at 10.0 "finish"

$ns run
```

3.2.7 Exercise 1.6.1b

```
set ns [new Simulator]

#trace file
```

```
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}

array set lan_nodes {}
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

for {set i 0} {$i < 5} {incr i} {
        set lan_nodes($i) [$ns node]
        $ns duplex-link $lan_nodes($i) $lan_router 10Mb 10ms DropTail
}

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 2MB 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 40Mb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail


# TCP connection, node 0 is agent, Internet server is sink
array set tcp_agents {}
array set tcp_sinks {}

for {set i 0} {$i < 5} {incr i} {
    set tcp_agents($i) [new Agent/TCP]
    $ns attach-agent $kul_servers $tcp_agents($i)
    set tcp_sinks($i) [new Agent/TCPSink]
    $tcp_sinks($i) set fid_ $i
    $ns attach-agent $lan_nodes($i) $tcp_sinks($i)
    $ns connect $tcp_agents($i) $tcp_sinks($i)
}

# UDP connection, node 0 is agent, Internet server is sink
array set udp_agents {}
set udp_sink [new Agent/Null]
$ns attach-agent $internet_servers $udp_sink

for {set i 0} {$i < 5} {incr i} {
    set udp_agents($i) [new Agent/UDP]
    $udp_agents($i) set fid_ $i
    $ns attach-agent $lan_nodes($i) $udp_agents($i)
    $ns connect $udp_agents($i) $udp_sink
}


array set ftp {}
for {set i 0} {$i < 5} {incr i} {
    set ftp($i) [new Application/FTP]
    $ftp($i) set type_ FTP
    $ftp($i) attach-agent $tcp_agents($i)
    $ftp($i) set packetSize_ 1500
    $ftp($i) set window_ 80
}

array set cbr {}
for {set i 0} {$i < 5} {incr i} {
    set cbr($i) [new Application/Traffic/CBR]
    $cbr($i) set type_ CBR
    $cbr($i) attach-agent $udp_agents($i)
    $cbr($i) set packetSize_ 1500
}


for {set i 0} {$i < 5} {incr i} {
    $ns at 0.1 "$ftp($i) start"
    $ns at 3.0 "$cbr($i) start"
    $ns at 6.0 "$cbr($i) stop"
    $ns at 9.9 "$ftp($i) stop"
}
$ns at 10.0 "finish"

$ns run
```

3.2.8 Exercise 1.6.2

```
set ns [new Simulator]

#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf

proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}

set n 10
array set lan_nodes {}
set lan_router [$ns node]
set modem_uplink [$ns node]
set modem_downlink [$ns node]
set telenet_network [$ns node]
set kul_servers [$ns node]
set internet_servers [$ns node]

for {set i 0} {$i < $n} {incr i} {
        set lan_nodes($i) [$ns node]
        $ns duplex-link $lan_nodes($i) $lan_router 10Mb 10ms DropTail
}

$ns duplex-link $lan_router $modem_uplink 10Mb 0.2ms DropTail

$ns simplex-link $modem_uplink $modem_downlink 2MB 0.2ms DropTail
$ns simplex-link $modem_downlink $modem_uplink 40Mb 0.2ms DropTail

$ns duplex-link $modem_downlink $telenet_network 100Mb 0.3ms DropTail

$ns duplex-link $telenet_network $kul_servers 100Mb 0.3ms DropTail
$ns duplex-link $telenet_network $internet_servers 100Mb 0.3ms DropTail


# TCP connection, node 0 is agent, Internet server is sink
array set tcp_agents {}
array set tcp_sinks {}

for {set i 0} {$i < $n} {incr i} {
    set tcp_agents($i) [new Agent/TCP]
    $ns attach-agent $kul_servers $tcp_agents($i)
    set tcp_sinks($i) [new Agent/TCPSink]
    $tcp_sinks($i) set fid_ $i
    $ns attach-agent $lan_nodes($i) $tcp_sinks($i)
    $ns connect $tcp_agents($i) $tcp_sinks($i)
}

# UDP connection, node 0 is agent, Internet server is sink
array set udp_agents {}
set udp_sink [new Agent/Null]
$ns attach-agent $internet_servers $udp_sink

for {set i 0} {$i < $n} {incr i} {
    set udp_agents($i) [new Agent/UDP]
    $udp_agents($i) set fid_ $i
    $ns attach-agent $lan_nodes($i) $udp_agents($i)
    $ns connect $udp_agents($i) $udp_sink
}


array set ftp {}
for {set i 0} {$i < $n} {incr i} {
    set ftp($i) [new Application/FTP]
    $ftp($i) set type_ FTP
    $ftp($i) attach-agent $tcp_agents($i)
    $ftp($i) set packetSize_ 1500
    $ftp($i) set window_ 80
}

array set cbr {}
for {set i 0} {$i < $n} {incr i} {
    set cbr($i) [new Application/Traffic/CBR]
    $cbr($i) set type_ CBR
    $cbr($i) attach-agent $udp_agents($i)
    $cbr($i) set packetSize_ 1500
}
```

```
set rng1 [new RNG]
$rng1 next-substream
# Random times in two second interval
set time_svar [new RandomVariable/Uniform]
$time_svar set avg_ 5.0
$time_svar use-rng $rng1
$time_svar set min_ 0.1
$time_svar set max_ 10.0

set rng2 [new RNG]
$rng2 next-substream
# Random times in two second interval
set duration_svar [new RandomVariable/Uniform]
$duration_svar set avg_ 2.0
$duration_svar use-rng $rng2
$duration_svar set min_ 0.1
$duration_svar set max_ 5.0

for {set i 0} {$i < $n} {incr i} {
    set time [expr [$time_svar value]]
    set duration [expr [$duration_svar value]]
    $ns at $time "$ftp($i) start"
    $ns at [expr $time + $duration] "$ftp($i) stop"
}

for {set i 0} {$i < $n} {incr i} {
    set time [expr [$time_svar value]]
    set duration [expr [$duration_svar value]]
    $ns at $time "$cbr($i) start"
    $ns at [expr $time + $duration] "$cbr($i) stop"
}


# for {set i 0} {$i < 10} {incr i} {
#     $ns at 0.1 "$ftp($i) start"
#     $ns at 3.0 "$cbr($i) start"
#     $ns at 6.0 "$cbr($i) stop"
#     $ns at 9.9 "$ftp($i) stop"
# }

$ns at 10.0 "finish"

$ns run
```

3.2.9 Exercise 2.4

```
set ns [new Simulator]

#trace file
set tf [open /dev/stdout w]
$ns trace-all $tf

#log file
set logger [open logger.tr w]
set filelogger [open files.tr w]

#nam tracefile
set nf [open simplified.nam w]
$ns namtrace-all $nf
```

```
proc finish {} {
        #finalize trace files
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        exit 0
}


array set lan_nodes {}
array set internet_nodes {}
set router [$ns node]
set modem [$ns node]


for {set i 0} {$i < 2} {incr i} {
        set lan_nodes($i) [$ns node]
        $ns duplex-link $lan_nodes($i) $router 10Mb 10ms DropTail
}

$ns duplex-link $router $modem 10Mb 10ms DropTail
$ns queue-limit $router $modem 20

for {set i 0} {$i < 2} {incr i} {
        set internet_nodes($i) [$ns node]
        $ns duplex-link $internet_nodes($i) $modem 10Mb 10ms DropTail
}


# TCP connection, KUL server is agent, node 1 is sink
set tcp_agent [new Agent/TCP/Reno]
$tcp_agent set fid_ 1
$tcp_agent set window_ 80
set tcp_sink [new Agent/TCPSink]

$ns attach-agent $internet_nodes(0) $tcp_agent
$ns attach-agent $lan_nodes(0) $tcp_sink

$ns connect $tcp_agent $tcp_sink

# FTP app
set ftp [new Application/FTP]
$ftp set type_ FTP
$ftp attach-agent $tcp_agent


## Crazy burst traffic simulator


# Generators for random size of files.
set rep 1
set rng1 [new RNG]
set rng2 [new RNG]
for {set i 0} {$i < $rep} {incr i} {
    $rng1 next-substream
    $rng2 next-substream
}
```

```
# Random size of files to transmit
set size_svar [new RandomVariable/Pareto]
$size_svar set avg_ 150000
$size_svar set shape_ 1.5
$size_svar use-rng $rng1

# Random times in two second interval
set time_svar [new RandomVariable/Exponential]
$time_svar set avg_ 0.05
$time_svar use-rng $rng2
# We now define the beginning times of transfers and the transfer sizes

#pg90 and 54

# Number of sources
set NodeNb 3
# Number of flows per source node
set NumberFlows 40
# TCP Sources , destinations , connections
for { set i 1} { $i <= $NodeNb } { incr i } {
        for { set j 1} { $j <= $NumberFlows } { incr j } {
                set tcpsrc($i,$j) [new Agent/TCP]
                set tcp_snk($i,$j) [new Agent/TCPSink]
                set k [expr $i * $NumberFlows + $j];
                $tcpsrc($i,$j) set fid_ $k
                $tcpsrc($i,$j) set window_ 2000
                $ns attach-agent $lan_nodes(1) $tcp_snk($i,$j)
                $ns attach-agent $internet_nodes(1) $tcpsrc($i,$j)
                $ns connect $tcpsrc($i,$j) $tcp_snk($i,$j)
                set ftp_array($i,$j) [$tcpsrc($i,$j) attach-source FTP]
                }
        }


# Arrivals of sessions follow a Poisson process.
# set the beginning time of next transfer from source and attributes
# update the number of flows
for {set i 1} {$i <=$NodeNb} {incr i } {
    set t [expr $i * 5.0]
    for {set j 1} {$j<=$NumberFlows} {incr j } {
        set size [expr [$size_svar value]]
        puts $filelogger "$i-$j $t $size"
        $ns at $t "$ftp_array($i,$j) send $size"
        set addedTime [$time_svar value]
        set t [expr $t + $addedTime]
        # puts "$t"
    }
}

# Printing the window size

proc printWindow {} {
        global ns tcp_agent logger
        set time 0.001
        set now [ $ns now ]
        set cwnd [ $tcp_agent set cwnd_ ]
        set ssthresh [ $tcp_agent set ssthresh_ ]
        puts $logger "$now $cwnd $ssthresh"
        $ns at [expr $now + $time] "printWindow"
}
```

```
$ns at 0.1 "printWindow"
$ns at 0.1 "$ftp start"
$ns at 40.0 "$ftp stop"
$ns at 40.0 "finish"
$ns run
```