Assignment Network Simulation

Peter De Wolf & Wout Vekemans

May 2, 2014

Exercise 1: Bandwidth restrictions on Kotnet

1. When we leave out the uploading connection, the throughput of the FTP connection is relatively constant. See figure 1. The throughput rate is limited by the bandwidth cap.

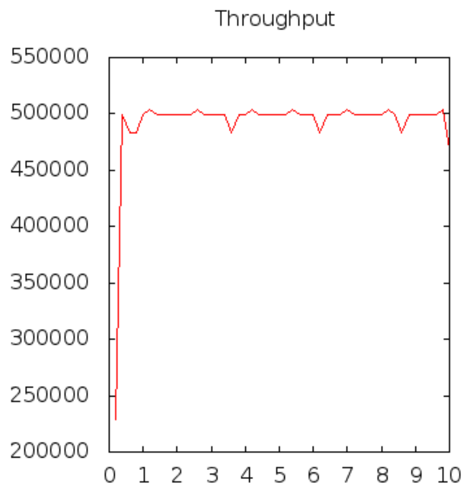

Figure 1: Throughput of the main FTP connection

2. When re-enabling the uploading connection, we see that the download rate drops when the upload starts. This is caused by the fact that the ACKs of the download need to travel through the same connection as the packets uploaded by the user. This results in a lot of dropped ACKs, which causes the server to restart sending, and therefore decreasing the throughput of the download link. See figure 2.

3. When a fixed amount of upload bandwidth is allocated to both applications, there wont be a drop in the download throughput. It would be like there were to separate upload connections: one for the ACKs of the FTP connection, and one for the data packets of the CBR connection, so they would both have more or less constant throughput. The ACKs and the uploaded data packets would not 'steal' eachother's bandwidth.

4. When the connection bandwidth is more limited, we get sort of the same result as in part 1, but with more packet loss. When the connection between server and user is slower, the buffers get full very fast,

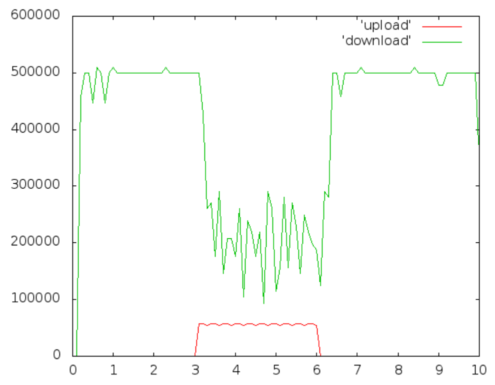

Figure 2: Throughput of the FTP and UDP connection

and more packets need to be dropped. This causes large fluctuations in download througput.

5. The upload connection stays constant at the level corresponding to the 30k rate.
   To improve download with capped connections, we would use some kind of round robin scheduler. This will allow all active users to have equal bandwidth allocation.

6. (a) When there is 10 times more capacity and there are 10 equal users, the bandwidth will be equally divided between users. The only packet loss will be the ones 'accidentaly' getting lost, like in the first section of this question. All users will have a throughput similar to that in section 1 of this question.
   When there are five users with the same connection, the throughput per user will be better, because there will be more bandwidth available.

   (b) When there are 10 users starting at random times, the uploads won't start all at the same time. The drops in the download connection wont occur at the same time as well, so the total throughput of the FTP connection will be more or less constant.

Exercise 2: Tahoe and Reno versus bursty web traffic

1. In the beginning the buffers can compensate for the bursts. This causes for a delayed effect of the bursts. The bursts start to affect the connection once the buffers are completely filled. When this happens, the will be packet loss and the main FTP connection notices this loss. As response to this the main FTP connection will slow down the input rate (fast recovery to threshold followed by additive increase).

2. The threshold in our example is 80. TCP uses slow start to converge to an optimal window size. In the beginning there will be an exponential increase of the congestion window until the threshold is reached. Once this occures, the congestion window will increase additive (AIMD). When the connection detects congestion, TCP uses fast recovery to avoid it. After the fast recovery decrease will occur until the size is half the size at which congestion was detected. For window size see figure 3.

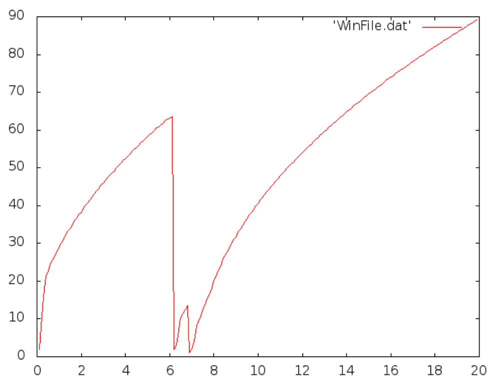

Figure 3: Window size for default TCP

3. AIMD or additive increase and multiple decrease is an algorithm used for TCP congestion avoidance. Additive increase and multiple decreases causes the size resemble a sawtooth wave. This sawtooth converges around the optimum. Once the slow start threshold is reached, only additive increase is used by TCP to increase the congestion window. Fast recovery decreases the window size until it has half the size of the size at which congestion was detected. TCP does this using multiple decreases.

4. The difference with the normal TCP is that Reno uses fast recovery. When one packet is lost, the congestion window decreases multiple times until the size is at half of the value of the window size at which congestion was detected. When more than one packet is lost, the window size decreases to zero. For the graph, see figure 4.

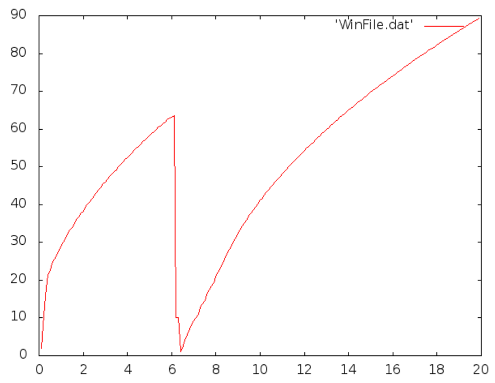

Figure 4: Window size for Reno

Code exercise 1

```
#Create simulator
set ns [new Simulator]

$ns color 0 Blue
$ns color 1 Red
$ns color 2 Green

#trace file
set tf [open opdracht1.tr w]
$ns trace-all $tf

#nam tracefile
set nf [open opdracht1.nam w]
$ns namtrace-all $nf

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

        exec nam out1.nam &
        exit 0
}

# create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]

#and links
$ns duplex-link $n0 $n2 10Mb 0.2ms DropTail
$ns duplex-link $n1 $n2 10Mb 0.2ms DropTail
$ns duplex-link $n2 $n3 10Mb 0.2ms DropTail
$ns simplex-link $n3 $n4 256kb 0.2ms DropTail
$ns simplex-link $n4 $n3 4Mb 0.2ms DropTail
$ns duplex-link $n4 $n5 100Mb 0.3ms DropTail
$ns duplex-link $n5 $n6 100Mb 0.3ms DropTail
$ns duplex-link $n5 $n7 100Mb 0.3ms DropTail

#ftp download 6 to 1
set tcp [new Agent/TCP]
$ns attach-agent $n6 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 80

set ftp [new Application/FTP]
$ftp attach-agent $tcp

#UDP upload 0 to 7
set udp0 [new Agent/UDP]
```

```
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set fid_ 2
$udp0 set packetSize_ 1500
$udp0 set rate_ 30000

set null0 [new Agent/Null]
$ns attach-agent $n7 $null0
$ns connect $udp0 $null0


#start and stop
$ns at 0.1 "$ftp start"
$ns at 3.0 "$cbr0 start"
$ns at 6.0 "$cbr0 stop"
$ns at 9.9 "$ftp stop"
$ns at 10.0 "finish"

#run
$ns run
```

Code exercise 2

```
#create a simulator
set ns [new Simulator]

#trace files: nam and tr
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf

$ns color 0 Blue
$ns color 1 Red

#finish procedure
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $tf
        close $nf
        #exec xgraph WinFile.dat &
        #xexec xgraph WinFile2.dat &
        exec nam out.nam &
        exit 0
}

#defining the topology
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n0 $n2 10Mb 10ms DropTail
$ns duplex-link $n0 $n4 10Mb 10ms DropTail
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n1 $n5 10Mb 10ms DropTail
$ns queue-limit $n0 $n1 20

set numberOfBursts 3
set filesPerBurst 40

#set up the tcp connection
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n3 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n2 $sink1
$ns connect $tcp1 $sink1
$tcp1 set fid_ 1
$tcp1 set windowd_ 80

#setting up the ftp over tcp connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1


#making the generators for filesize and sending times
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 inter-arrival times of TCP transfer at each source i
set RV [new RandomVariable/Exponential]
$RV set avg_ 0.05
$RV use-rng $rng1

#Random size of files to transmit
set RVSize [new RandomVariable/Pareto]
$RVSize set avg_ 15000
$RVSize set shape_ 1.5
$RVSize use-rng $rng2

#defining the transfers
for {set i 1} {$i<=$numberOfBursts} { incr i} {
        set t [expr $i * 5]
        for {set j 1} {$j <= $filesPerBurst} { incr j} {
                set tcp($i,$j) [new Agent/TCP/Reno]
                $ns attach-agent $n5 $tcp($i,$j)
                set sink($i,$j) [new Agent/TCPSink]
                $ns attach-agent $n4 $sink($i,$j)
                $ns connect $tcp($i,$j) $sink($i,$j)
                $tcp($i,$j) set fid_ 0
                $tcp($i,$j) set window_ 80

                set ftp($i,$j) [new Application/FTP]
                $ftp($i,$j) attach-agent $tcp($i,$j)

                #setting the time and size of the transfers
                set t [expr $t + [$RV value]]
                set Conct($i,$j) $t
                set Size($i,$j) [expr [$RVSize value]]
                $ns at $Conct($i,$j) "$ftp($i,$j)␣send␣$Size($i,$j)"
}}

set winfile [open WinFile.dat w]
set winfile2 [open WinFile2.dat w]
```

```
#procedure for plotting window size
proc plotWindow {tcpSource file} {
        global ns
        set time 0.1
        set now [$ns now]
        set cwnd [$tcpSource set cwnd_]
        puts $file "$now $cwnd"
        $ns at [expr $now+$time] "plotWindow $tcpSource $file"
}
$ns at 0.1 "plotWindow $tcp1 $winfile"

proc plotThreshold {tcpSource file} {
        global ns
        set time 0.1
        set now [$ns now]
        set ssthresh [$tcpSource set ssthresh_]
        puts $file "$now $ssthresh"
        $ns at [expr $now + $time] "plotThreshold $tcpSource $file"
}
$ns at 0.1 "plotThreshold $tcp1 $winfile2"

#schedule events
$ns at 0.0 "$ftp1 start"
$ns at 20.0 "$ftp1 stop"
$ns at 20.0 "finish"

$ns run
```