Prof. Danny Hughes
danny.hughes@cs.kuleuven.be

Computer Networks: Network Simulation

Toon Nolten r0258654
toon.nolten@student.kuleuven.be

May 3, 2013

1 Bandwidth restrictions on KotNet

1.1 First, leave out the connection of the uploader. Plot the throughput of the main FTP connection and discuss the download behaviour of a KotNet/Telenet modem with bandwidth caps. How do bandwidth limitations affect the download connection?

The download connection uses the entire available download bandwidth. (see Figure: 1)

1.2 What happens if you now enable both upload and download activities? Plot and compare the throughput of both streams. Do you spot any problems? If so, explain.

The throughput of the FTP connection decreases dramatically when the CBR connection starts. (see Figure: 2) This is unexpected because download and upload use separate simplex links, which means upload traffic can't affect download traffic. However FTP makes use of a TCP connection, which requires acknowledgements, these acknowledgements are slowed down because the CBR traffic is saturating the upload link. Because the acknowledgements arrive more slowly on the uploading side of the FTP connection, the congestion window advances more slowly and the dowload throughput with it.

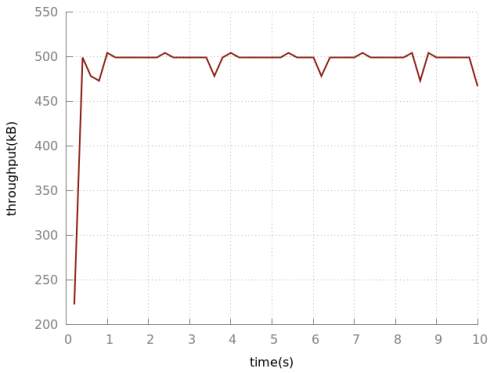

Figure 1: FTP throughput

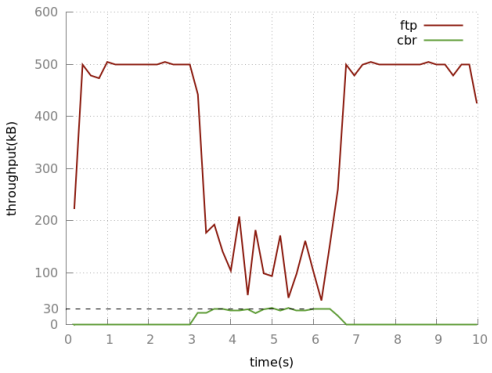

Figure 2: FTP & CBR throughput

1.3 Consider now that the model is able to guarantee a certain amount of upload bandwidth to every application, which performance do you expect for both applications (FTP and CBR)?

The performance of the FTP application will seem unaffected by the CBR traffic because it doesn't need alot of upload bandwidth to get the acknowledgements through. The CBR application will be slowed down a bit because it can't use all the available upload bandwidth.

1.4 What would be the result if bandwidth was even more restricted? For example, up to September 2012 KotNet limitations were 512kbps for downstream traffic and 100kbps for upstream traffic.

The FTP application would be slowed down alot. Not only is the download bandwidth far smaller, but because the download is slowed down, there have to be more acknowledgements, which requires more upload bandwidth and that bandwidth is shared with a CBR application.

1.5 Configure the upload connection with a fixed *rate_* of 30k. Explain the simulated result (no plot required). What would be your (extra) solution(s) to improve the download experience on shared LANs behind capped Internet connections? How could you practically realise this?

The FTP application is unaffected by the CBR traffic, this is because there's more than enough upload bandwidth for the CBR traffic and the acknowledgements combined. The solution is to try to share bandwidth in a way that is fair. In this case the network should drop packets from the CBR application, so there is enough bandwidth available for the acknowledgements.

1.6 Suppose we have in the future a KotNet/Telenet cable subscription that has 10 times more capacity: i.e. a connection that has a downstream capacity of 40Mbps with an upstream capacity of 2Mbps.

1.6.1 What performance can you expect if there are 10 users behind this connection and all performing the same activities as in this exercise? Why?

The result would be approximately the same as one user on the present network, since the amount of users has increased just as much as the available capacity. In practice the bandwidth would probably not be distributed completely fairly, so some users would get slightly more bandwidth, some slightly less.

1.6.2 What performance can you expect if there are only 5 users using the network? Why?

Double the current performance. There's only five times as many users and ten times as much capacity, so every user gets roughly double the previous capacity.

2 Tahoe and Reno versus bursty web traffic

2.1 Investigate the throughput of the main FTP application. Can you indicate the effects the individual bursts of web traffic have on its total throughput? Why do these effects not immediately become visible when each burst starts (i.e. at 5, 10 or resp. 15 seconds)?

When the first burst of 'http' traffic starts, the throughput of the FTP application drops very fast. It slowly recovers after each burst but drops fast a few times. The result is that the FTP throughput is dramatically reduced because of a few small (2 second) bursts of 'http' traffic. The effect is not immediately visible because the bursts exist of connections that are started sequentially, the first few connections don't make much of a difference but when more connections are opened the congestion control kicks in.

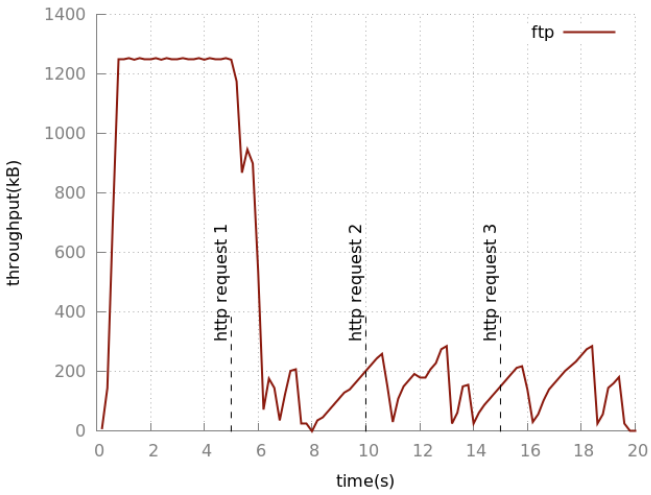

Figure 3: FTP throughput with intermittent http requests

2.2 Plot the congestion window size and slow start thresholds in another graph. Can you identify the different phases of the TCP algorithm? When is the slow start threshold recalculated and how? Take one 'sawtooth' pattern in the graph and indicate a few reasonable values for the congestion window on the graph. What is the first interval the TCP congestion avoidance algorithm is active?

The slow start threshold is recalculated when a timeout occurs. The new slow start threshold is set to half of the current slow start threshold. The first interval wherein the congestion avoidance algorithm is active is around the 6^{th} second.

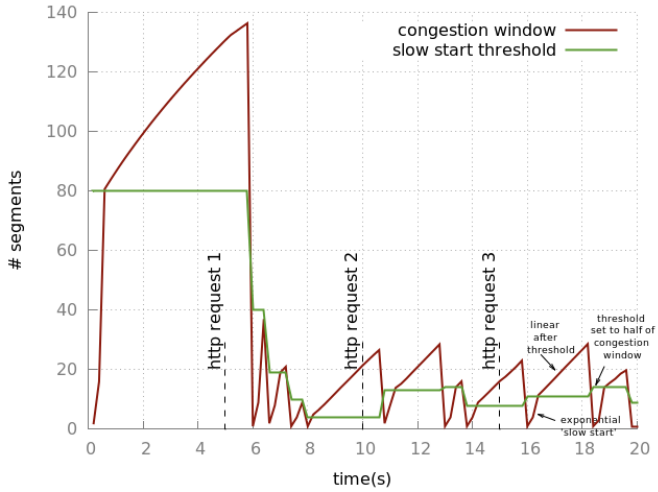

Figure 4: Congestion window & Slow start threshold for TCP Tahoe

2.3 Change the TCP implementation of the main FTP connection into *Reno*. Plot the congestion window and slow start thresholds. Do you spot the differences with the default TCP *Tahoe* implementation? Why does the window sometimes still drop to zero?

Yes the congestion window does not always drop to zero as in TCP Tahoe. That is an important improvement in TCP Reno, the congestion window starts at the new slow start threshold instead of falling back to zero. It still drops to zero sometimes, when the slow start threshold is too small.

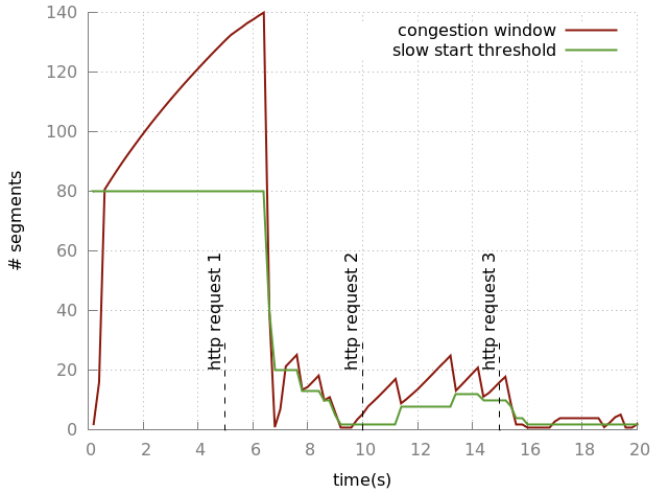

Figure 5: Congestion window & Slow start threshold for TCP Reno

Listing 1: Source code used for exercise 2

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

$ns color 1 Blue
$ns color 2 Red
$ns color 3 Green

#file size of each transfer
set filesize [open ex21.size.dat w]

#congestion window
set wnd [open ex21.wnd.dat w]

#slow start threshold
set ssthresh [open ex21.thresh.dat w]

#trace file
set tf [open /tmp/ex21.out.tr w]
$ns trace-all $tf

#nam tracefile
#set nf [open /tmp/ex1.out.nam w]
#$ns namtrace-all $nf

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

        #exec nam /tmp/ex1.out.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]

#and links
$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

#Set Queue Size of link (n0-n1) to 20
$ns queue-limit $n0 $n1 20

#connections
#Setup a TCP connection
set tcp1 [new Agent/TCP]
$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 window_ 80

#Setup a FTP over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

#Generators
set rng [new RNG]
$rng seed 0

set RVtiming [new RandomVariable/Exponential]
$RVtiming set avg_ 0.05
$RVtiming use-rng $rng

set RVsize [new RandomVariable/Pareto]
$RVsize set avg_ 150000
$RVsize set shape_ 1.5
$RVsize use-rng $rng

#120 connections
set nbConn 120
set Tadd 0

for {set i 0} {$i < $nbConn} {incr i} {
        set tcp_src [new Agent/TCP]
        $ns attach-agent $n5 $tcp_src
        set tcp_snk [new Agent/TCPSink]
        $ns attach-agent $n4 $tcp_snk
        $ns connect $tcp_src $tcp_snk

        set ftp [$tcp_src attach-source FTP]

        set fileSize [$RVsize value]
```

```
        $tcp_src set size $fileSize

        set startT [expr [expr $i/40 + 1] * 5]
        if {[expr [expr $i-1]/40] < [expr $i/40]} {
                set Tadd 0
        }
        $ns at [expr $startT + $Tadd] "$ftp start"
        set Tadd [expr $Tadd + [$RVtiming value]]

        puts $filesize "[expr $i/40 + 1] \t [expr $startT + $Tadd] \t $fileSize"
}

#logging congestion window and slow start threshold
proc logWndThresh {} {
        global wnd ssthresh ns tcp1

        puts $wnd "[$ns now] [$tcp1 set cwnd_]"
        puts $ssthresh "[$ns now] [$tcp1 set ssthresh_]"

        $ns at [expr [$ns now] + 0.2] "logWndThresh"
}

$ns at 0.1 "$ftp1 start"
$ns at 0.2 "logWndThresh"
$ns at 19.9  "$ftp1 stop"
$ns at 20 "finish"

$ns run
```