

Netwerksimulatie – Verslag 1

Philippe Dellaert
3e Bachelor Computerwetenschappen-Elektrotechniek

Wouter Van Ranst
3e Bachelor Computerwetenschappen-Bedrijfsbeheer

26 april 2007

Inhoudsopgave

1	Vaststellingen	1
2	Algoritmes	2
2.1	Standaard	2
2.2	NewReno	2
2.3	Vegas	2
3	Discussie	3
4	Figuren	4
A	Code	7
A.1	Exercise 2	7

1 Vaststellingen

In bijgevoegde figuren zijn onze resultaten te zien van de simulatie. In figuren 1, 2 en 3 is de throughput te zien in nodes 1, 3, 5 en 7 respectievelijk voor standaard TCP, NewReno en Vegas. In figuren 4, 5 en 6 worden de droprates in node 0 voorgesteld voor de verschillende protocols.

Uit figuren 1, 2 en 3 blijkt duidelijk dat de throughput bij de normale TCP implementatie lager ligt dan bij NewReno, en dat de Vegas implementatie een betere throughput geeft.

Uit figuur 6 kunnen we dan weer afleiden dat bij de Vegas implementatie de droprate veel kleiner is dan bij de andere implementaties en enkel in het begin voorkomt. Bij standaard TCP en NewReno worden er regelmatig packets gedropt (figuren 1 en 2).

2 Algoritmes

2.1 Standaard

In de standaard TCP implementatie wordt begonnen in de slow start fase. In deze fase wordt het zendvenster telkens verhoogd met het maximaal aantal bytes dat de zender bereid is te zenden telkens er een acknowledgment binnenkomt. Als een zekere threshold is bereikt, wordt overgeschakeld op de congestion avoidance fase.

In deze fase wordt het zend venster telkens met een segmentgrootte vergroot per Round Trip Time, dit is de tijd tussen het verzenden van een pakket en het ontvangen van de bijhorende ACK.

Als er drie gelijke acknowledgements zijn toegekomen, of als zich een timeout heeft voorgedaan, wordt het zend venster vergroot, en opnieuw geprobeerd. Als dit uiteindelijk is gelukt, en een ACK voor nieuwe data wordt ontvangen, wordt de grootte van het zendvenster opnieuw gereduceerd tot de waarde gedefinieerd door de threshold. Zo start dus opnieuw de congestion avoidance fase.

In figuur 1 is te zien (oscillatorisch karakter) dat de throughput duidelijk aangetast wordt vanaf het moment dat er een zendfout optreedt (dus een timeout of drie gelijke ACKs). In figuur 4 is ook te zien dat er als zo'n zendfout optreedt pakketten worden gedropt, tot de fout is opgelost.

2.2 NewReno

NewReno is een modificatie van standaard TCP. Zo gebruikt NewReno dezelfde slow start en congestion avoidance fases, maar worden zendfouten anders behandeld.

In de standaard implementatie wordt er van uit gegaan dat in de ACK die uiteindelijk het verloren gegane pakket bevestigt ook alle andere vorige pakketten worden bevestigd. In NewReno hoeft dit niet het geval te zijn. Als de vorige pakketten niet worden bevestigd wil dit zeggen dat de ACK een partiële ACK is, en wordt het eerste onbevestigde pakket herzonden, en wordt de recovery procedure verder gezet.

Deze recovery procedure wordt pas gestopt als alle pakketten voor de fout effectief bevestigd zijn. Dit heeft natuurlijk als voordeel dat ontbrekende pakketten sneller worden gedetecteerd, en dat er dus veel sneller kan worden gerecovered. Hierdoor zal de throughput minder lijden aan de fout, en dit is te merken aan figuur 2. Uiteraard worden hier ook nog pakketten gedropt, zoals te zien is in figuur 5, maar minder snel dan bij de standaard implementatie.

2.3 Vegas

Vegas houdt de throughput in de gaten. Als de throughput te hoog wordt, zal het congestion window worden verkleind, om te voorkomen dat er zo snel gaat worden verzonden dat het netwerk het niet meer aan kan. Als de throughput te laag wordt, zal het congestion window worden vergroot. Dit is namelijk een teken dat er sneller kan worden gezonden.

Op die manier wordt een vrij constante throughput bereikt, omdat vermeden wordt dat de throughput zakt doordat het netwerk het zendtempo niet meer aankan en packets zou moeten beginnen droppen, of doordat het netwerk

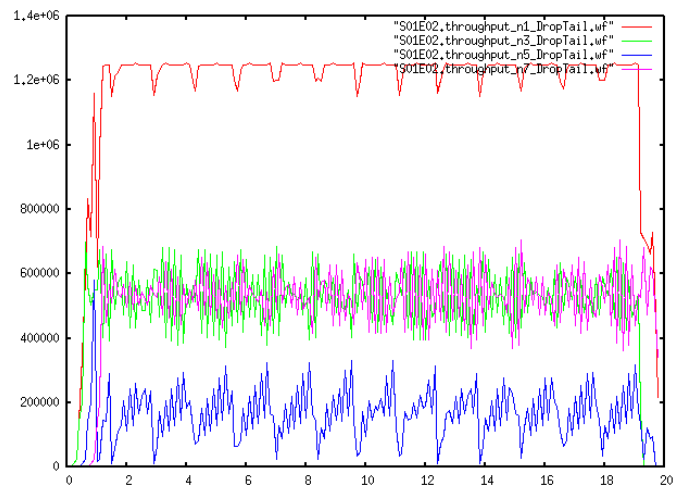
meer pakketten zou aankunnen maar er geen meer krijgt omdat het congestion window te klein is.

Deze uitstekende karakteristiek vertaalt zich ook in nagenoeg geen gedropte packets, zoals te zien is in figuur 6, aangezien de packets die zouden worden gedropt door het algoritme worden geanticipeerd.

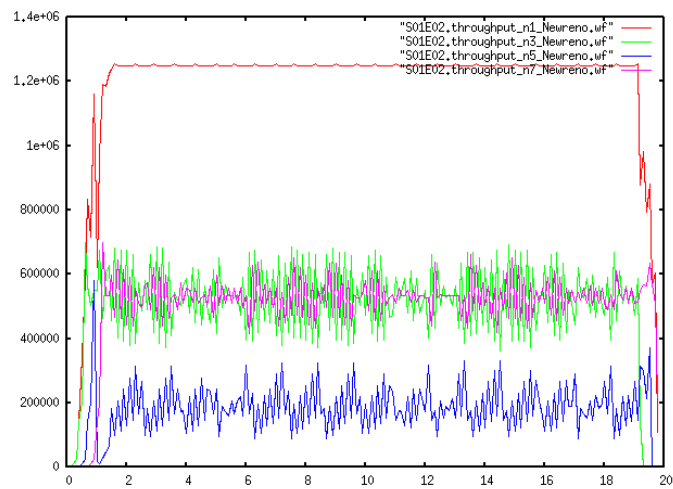
3 Discussie

Vegas is duidelijk de meest efficiënte implementatie van de drie hier besproken. Er worden bijna geen pakketten gedropt, doordat dit tijdig wordt vermeden, en er wordt een constante throughput gehaald.

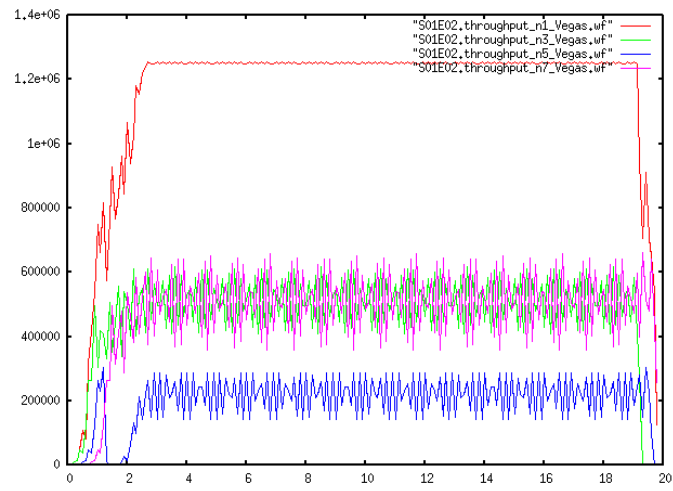
4 Figuren



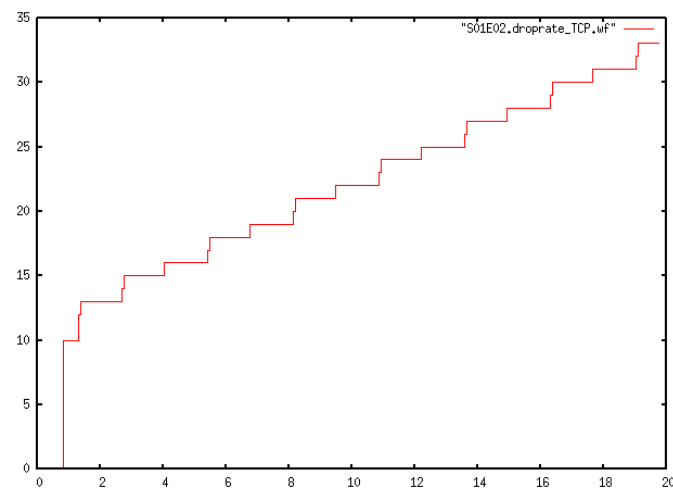
Figuur 1: Throughput in nodes 1, 3, 5 en 7 voor de normale TCP implementatie



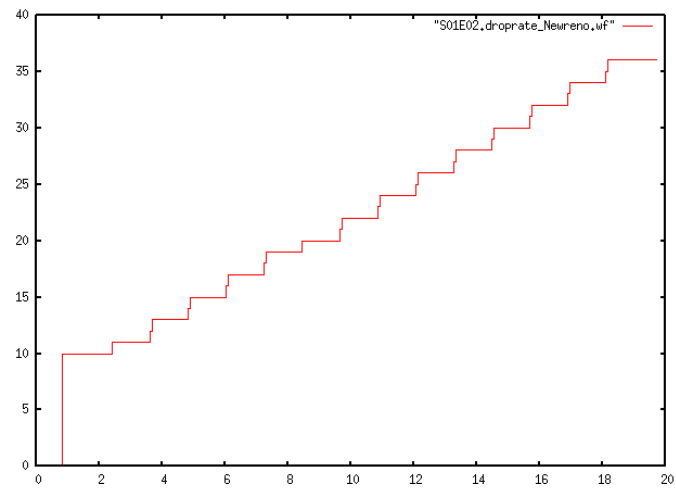
Figuur 2: Throughput in nodes 1, 3, 5 en 7 voor de NewReno implementatie



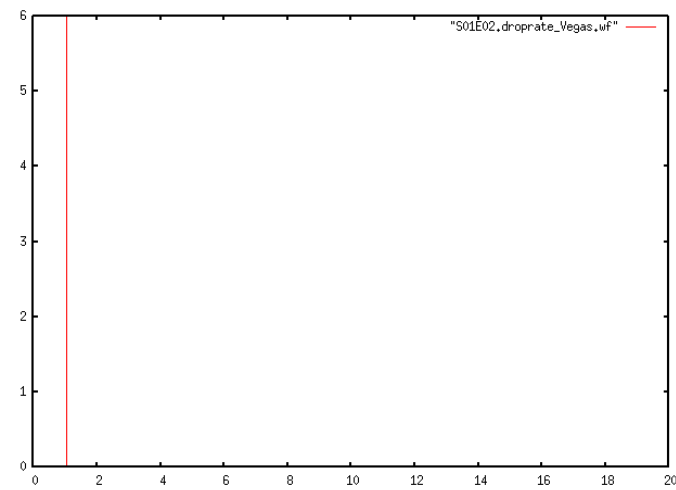
Figuur 3: Throughput in nodes 1, 3, 5 en 7 voor de Vegas implementatie



Figuur 4: Droprate in node 0 voor de normale TCP implementatie



Figuur 5: Droprate in node 0 voor de NewReno implementatie



Figuur 6: Droprate in node 0 voor de Vegas implementatie

A Code

A.1 Exercise 2

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

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

#trace file
set tf [open S01E02.out.tr w]
set wft [open S01E02_t.winfile w]
$ns trace-all $tf

#nam tracefile
set nf [open S01E02.out.nam w]
$ns namtrace-all $nf

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

    exec nam S01E02.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]
set n6 [$ns node]
set n7 [$ns node]

# Variables
# Number of FTP connections
set numFTP 3
# Setting connections
# Connection 1
set ftpConnections(0,origin) $n2
set ftpConnections(0,destination) $n3
set ftpConnections(0,start) 0.1
set ftpConnections(0,stop) 19.1
```

```

# Connection 2
set ftpConnections(1,origin) $n4
set ftpConnections(1,destination) $n5
set ftpConnections(1,start) 0.4
set ftpConnections(1,stop) 19.5
# Connection 3
set ftpConnections(2,origin) $n6
set ftpConnections(2,destination) $n7
set ftpConnections(2,start) 0.7
set ftpConnections(2,stop) 19.7

#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 $n0 $n6 10Mb 10ms DropTail
$ns duplex-link $n3 $n1 10Mb 10ms DropTail
$ns duplex-link $n5 $n1 10Mb 10ms DropTail
$ns duplex-link $n7 $n1 10Mb 10ms DropTail
#$ns duplex-link $n0 $n1 10Mb 10ms RED
#$ns duplex-link $n0 $n2 10Mb 10ms RED
#$ns duplex-link $n0 $n4 10Mb 10ms RED
#$ns duplex-link $n0 $n6 10Mb 10ms RED
#$ns duplex-link $n3 $n1 10Mb 10ms RED
#$ns duplex-link $n5 $n1 10Mb 10ms RED
#$ns duplex-link $n7 $n1 10Mb 10ms RED
#$ns duplex-link $n0 $n1 10Mb 10ms SFQ
#$ns duplex-link $n0 $n2 10Mb 10ms SFQ
#$ns duplex-link $n0 $n4 10Mb 10ms SFQ
#$ns duplex-link $n0 $n6 10Mb 10ms SFQ
#$ns duplex-link $n3 $n1 10Mb 10ms SFQ
#$ns duplex-link $n5 $n1 10Mb 10ms SFQ
#$ns duplex-link $n7 $n1 10Mb 10ms SFQ

#Give node position (for NAM)
$ns duplex-link-op $n2 $n0 orient right-down
$ns duplex-link-op $n4 $n0 orient right
$ns duplex-link-op $n6 $n0 orient right-up
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n3 orient right-up
$ns duplex-link-op $n1 $n5 orient right
$ns duplex-link-op $n1 $n7 orient right-down
$ns duplex-link-op $n0 $n1 queuePos 0.5

#Queue limiting n0-n1
$ns queue-limit $n0 $n1 20

# Setting winfiles
for {set i 0} {$i < $numFTP} {incr i} {
    set wf($i) [open S01E02_$i.wf w]
}

```

```

}

# Generating FTP Connections
for {set i 0} {$i < $numFTP} {incr i} {
    #set tcp($i) [new Agent/TCP]
    #set tcp($i) [new Agent/TCP/Newreno]
    set tcp($i) [new Agent/TCP/Vegas]
    $ns attach-agent $ftpConnections($i,origin) $tcp($i)
    set sink($i) [new Agent/TCPSink]
    $ns attach-agent $ftpConnections($i,destination) $sink($i)
    $ns connect $tcp($i) $sink($i)
    $tcp($i) set fid_ $i
    $tcp($i) set packetSize_ 552
    $tcp($i) set window_ 60
    set ftp($i) [new Application/FTP]
    $ftp($i) attach-agent $tcp($i)

    # Widow plotting
    $ns at 0.1 "plotWindow $tcp($i) $wf($i)"

    #Timing
    $ns at $ftpConnections($i,start) "$ftp($i) start"
    $ns at $ftpConnections($i,stop) "$ftp($i) stop"
}

#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"
}

#Procedure for plotting window size TOTAL
proc plotWindowTotal {file} {
    global ns numFTP tcp
    set time 0.1
    set now [$ns now]
    for {set i 0} {$i < $numFTP} {incr i} {
        set cwnd($i) [$tcp($i) set cwnd_]
    }
    set total 0
    for {set i 0} {$i < $numFTP} {incr i} {
        set total [expr $total+$cwnd($i)]
    }
    puts $file "$now $total"
    $ns at [expr $now+$time] "plotWindowTotal $file"
}

```

```
# Plotting total
$ns at 0.1 "plotWindowTotal $wft"

#Timing
$ns at 20.0 "finish"

#running the network simulator
$ns run
```