ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ

ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ ΚΑΙ ΜΗΧΑΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΏΝ

ΔΊΚΤΥΑ ΕΠΙΚΟΙΝΩΝΙΏΝ

(2018-2019)

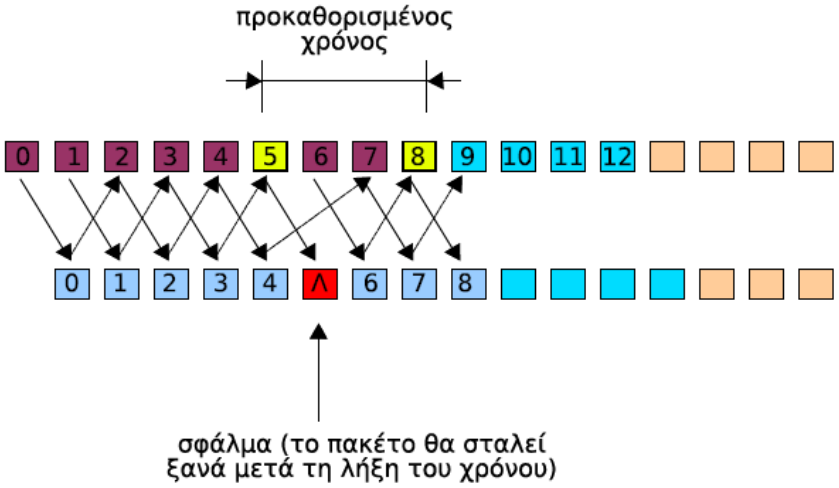
4η ΕΡΓΑΣΤΗΡΙΑΚΉ ΑΝΑΦΟΡΆ

Ονοματεπώνυμο : Χρήστος Τσούφης

Α.Μ. : 03117176

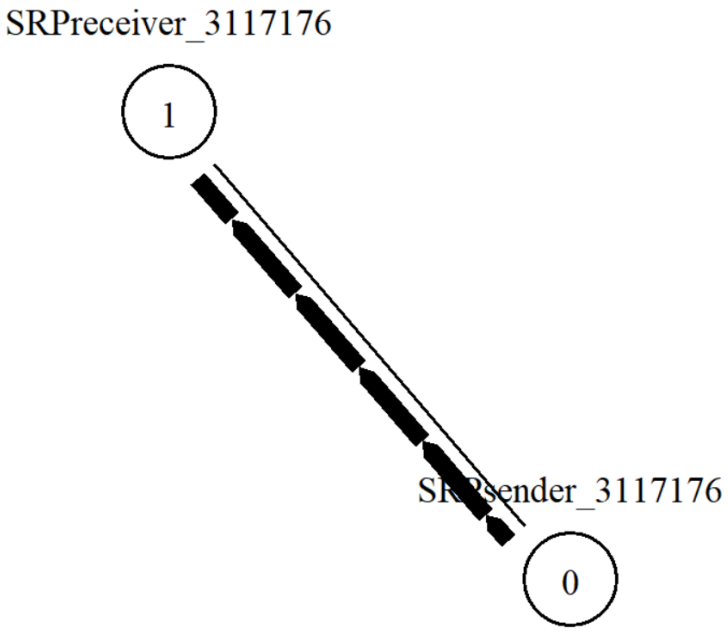
1. Πρωτόκολλο Selective Repeat

Παραλλαγή του πρωτοκόλλου SWP είναι το Selective Repeat και σκοπός της παρούσας άσκησης αποτελεί η μελέτη αυτού του πρωτοκόλλου ώστε με χρήση της εξίσωσης απόδοσης να μελετηθεί η απόδοσή του σε διάφορες περιπτώσεις εναλλάσσοντας τις παραμέτρους που την απαρτίζουν. Στο πρωτόκολλο Selective Repeat (SR), το οποίο είναι ικανό να γνωρίζει ακριβώς ποια πακέτα χάθηκαν ή καταστράφηκαν ώστε να ξαναστείλει μόνο αυτά, ο παραλήπτης χρησιμοποιεί ένα buffer προκειμένου να αποθηκεύσει τα πακέτα που φτάνουν μετά από κάποιο μη αναγνωρισμένο πακέτο, προκειμένου να μεταδώσει ξανά μόνο τα πακέτα που έχουν καταστραφεί. Όταν το πρόβλημα γίνει αντιληπτό, ο αποστολέας ξαναστέλνει το πακέτο, όμως σε αυτή την περίπτωση εκτός σειράς. Όμως στο αμέσως ανώτερο στρώμα του OSI layer, το network layer, τα πακέτα διοχετεύονται πάντα σε σειρά. Η απόδοση του SR δίνεται ως το κλάσμα του χρόνου αξιοποίησης του SR από τα "καθαρά" δεδομένα (net data) που θέλουμε να μεταδώσουμε προς το συνολικό χρόνο: $n_{SR} = \min\{1, \frac{WT}{\tau+2p+\tau ACK}\}$.



2. Σενάριο προσομοίωσης

Τοπολογία:



Πηγαίος κώδικας:

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

# Open the nam trace file
set nf [open lab4.nam w]
$ns namtrace-all $nf
set trf [open lab4.tr w]
$ns trace-all $trf

# Create two nodes
set n(0) [$ns node]
set n(1) [$ns node]
$ns at 0.0 "$n(0) label
SRPsender_3117176 "
$ns at 0.0 "$n(1) label
SRPreceiver_3117176"

# Create a duplex link between the nodes
$ns duplex-link $n(0) $n(1) 10Mb 8ms
DropTail
# The following line sets the queue limit of
the two simplex links that connect node0
and node1 to the number specified.
$ns queue-limit $n(0) $n(1) 100
$ns queue-limit $n(1) $n(0) 100

# Create TCP agent
set tcp0 [new Agent/TCP/Sack1]
# Set window size
$tcp0 set window_ 6
# The initial size of the congestion window
on slow-start
$tcp0 set windowInit_ 6
# Disable modelling the initial
SYN/SYNACK exchange
$tcp0 set syn_ false
# Set packet size
$tcp0 set packetSize_ 2000
# Attach TCP agent to node n(0)
$ns attach-agent $n(0) $tcp0
# Create TCP sink
set sink0 [new Agent/TCPSink]
# Attach TCP sink to node n(1)
$ns attach-agent $n(1) $sink0
# Connect TCP agent to sink
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

# Define a 'finish' procedure
proc finish {} {
global ns nf trf
$ns flush-trace
# Close the trace file
close $nf
close $trf
exit 0
}

# Events
$ns at 0.3 "$ftp0 start"
$ns at 5.0 "$ftp0 stop"
$ns at 6.0 "finish"

# Run the simulation
$ns run
```

3. Ανάλυση αρχείου ίχνους (trace file)

Κατόπιν εκτέλεσης της προσομοίωσης οι πληροφορίες για τα πακέτα αποθηκεύονται σε ένα trace file. Αναλύοντας το trace file με rules τα οποία ορίζουμε για το awk μπορούμε να ανακτήσουμε τις πληροφορίες που θέλουμε. Μπορούμε να καταγράψουμε τα δεδομένα που θέλουμε. Η εκτέλεση του awk script για την ανάλυση του tracefile μας επιστρέφει τα εξής στοιχεία:

```
C:\Users\Chris Tsoufis\Desktop>awk95.exe -f lab4.awk < lab4.tr
Total Data received     : 3257840 Bytes
Total Packets received  : 1597
```

Πηγαίος κώδικας:

```
BEGIN {
data=0;
packets=0;
}
/^r/&&/tcp/ {
data+=$6;
packets++;
}
END{
printf("Total Data received\t: %d Bytes\n", data);
printf("Total Packets received\t: %d\n", packets);
}
```

4. Μελέτη της απόδοσης Selective Repeat

Ερωτήσεις:

a) *Πώς πρέπει να τροποποιηθεί ο κώδικας της προσομοίωσης ώστε η ζεύξη μεταξύ των δύο κόμβων της διάταξης να απεικονίζεται σε οριζόντια θέση, όπως φαίνεται στο Σχήμα 1 και, ο FTP agent να αρχίζει την αποστολή των πακέτων τη χρονική στιγμή t = 0,5 + 0,1*AM sec(δηλ. t = 1,1 sec), ενώ όταν t = 4,0 + 0,2*AM sec (δηλ. t = 5,2 sec), η αποστολή να ολοκληρώνεται;*

Με προσθήκη στο script κάτω από τον ορισμό της ζεύξης την εξής γραμμή κώδικα

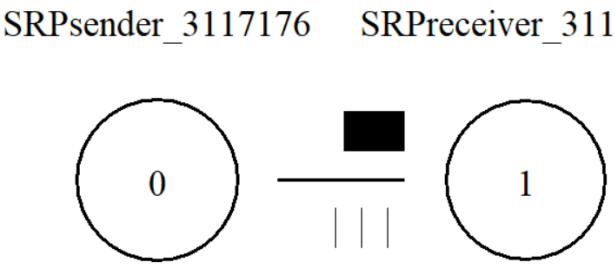
$ns duplex-link-op $n(0) $n(1) orient right

για να εμφανίζονται οι δυο κόμβοι σε οριζόντια διάταξη, αλλά και με κατάλληλη τροποποίηση του χρόνου

$ns at 1.1 "$ftp0 start"

$ns at 5.2 "$ftp0 stop"

προκύπτει:



b) *Να επαληθεύσετε κατά πόσον ισχύει ή όχι η εξίσωση σε περίπτωση απουσίας σφαλμάτων:* $\eta = min\{\frac{W \times TRANSP}{S}, 1\}$.

Από τα δεδομένα προκύπτει:

- W = μέγεθος παραθύρου ⇒ W=6
- TRANSP = χρόνος μετάδοσης tcp πακέτου = $\frac{tcp_size}{\max_bitrate} = \frac{2.000\ bytes \times 8}{10.000.000\ \frac{bit}{sec}} = \frac{16.000}{10.000.000}$sec ⇒ TRANSP = 0,0016 sec
- PROP = καθυστέρηση διάδοσης πακέτου = καθυστέρηση ζεύξης ⇒ PROP = 0,008 sec
- TRANSA = χρόνος μετάδοσης ack πακέτου = $\frac{ack_size}{\max_bitrate} = \frac{40\ bytes \times 8}{10.000.000\ \frac{bit}{sec}} = \frac{320}{10.000.000}$sec ⇒ TRANSA = 0,000032 sec

- S = TRANSP + 2×PROP + TRANSA = 0,0016 + 2×0,008 + 0,000032 ⇒ S = 0,017632 sec

Άρα, $\frac{\mathrm{W} \times \mathrm{TRANSP}}{\mathrm{S}} = \frac{6 \times 0{,}0016\ sec}{0{,}017632\ \mathrm{sec}} = \frac{0{,}0096\ sec}{0{,}017632\ \mathrm{sec}} = 0{,}544464$

Οπότε η = min{0,544464, 1} = 0,544464.

Βρίσκουμε την απόδοση του πρωτοκόλλου πειραματικά χρησιμοποιώντας το ρυθμό μετάδοσης που θα υπολογιστεί παρακάτω:

$\eta_{\pi\epsilon\iota\rho} = \frac{bitrate}{\max\_bitrate} = \frac{5.525.391{,}3232\ \frac{bits}{sec}}{10.000.000\frac{bit}{sec}} \Rightarrow \eta_{\pi\epsilon\iota\rho} = 0{,}5525$

Συμπεραίνουμε, λοιπόν, ότι η δοθείσα εξίσωση σε περίπτωση απουσίας σφαλμάτων ισχύει αλλά με μια μικρή απόκλιση.

c) *Ποιος είναι ο αριθμός των πακέτων που παρελήφθησαν; Πόσα δεδομένα παρελήφθησαν από τον παραλήπτη κατά τη διάρκεια της προσομοίωσης;*
Η εκτέλεση του προγράμματος ανάλυσης awk.exe μας δίνει τα παρακάτω αποτελέσματα:

```
C:\Users\Chris Tsoufis\Desktop>awk95.exe -f lab4.awk < lab4.tr
Total Data received     : 2843720 Bytes
Total Packets received  : 1394
```

Από τα οποία είναι φανερό ότι ο συνολικός αριθμός των πακέτων ισούται με 1.394 πακέτα και τα αντίστοιχα bytes είναι ίσα με 2.843.720 bytes.

d) *Τροποποιήστε κατάλληλα το πρόγραμμα awk, ώστε να προσδιορίζει τη συνολική διάρκεια μετάδοσης των δεδομένων (η οποία περιλαμβάνει και την ολοκλήρωση μετάδοσης όλων των επιβεβαιώσεων). Υπολογίστε το ρυθμό μετάδοσης δεδομένων και τη χρησιμοποίηση του καναλιού.*
Πηγαίος κώδικας:

```
BEGIN {
data=0;
packets=0;
last_ts=0;
}
/^r/&&/tcp/ {
data+=$6;
packets++;
}
/^r/&&/ack/ {
last_ts=$2;
}
END{
printf("Total Data received\t: %d Bytes\n", data);
printf("Total Packets received\t: %d\n", packets);
printf("Last Packet received\t: %s sec\n", last_ts);}
```

Και :

```
C:\Users\Chris Tsoufis\Desktop>awk95.exe -f lab4.awk < lab4.tr
Total Data received     : 2843720 Bytes
Total Packets received  : 1394
Last Packet received    : 5.217312 sec
```

όπου φαίνεται ότι η μετάδοση των δεδομένων ολοκληρώνεται στα 5,217312 sec, άρα η συνολική διάρκεια μετάδοσης είναι:
time = 5,217312 – 1,1 sec ⇒ time = 4,117312 sec.
Ο μέσος ρυθμός μετάδοσης είναι:
bitrate = $\frac{2.843.720\ bytes \times 8}{4{,}117312\ \text{sec}} = \frac{22.749.760\ \text{bits}}{4{,}117312\ \text{sec}} \approx 5.525.391{,}3232\ \frac{bits}{sec}$ ⇒ bitrate ≈ 5,525 Mbits
και η χρησιμοποίηση του καναλιού:
usage = $\frac{bitrate}{\max\_bitrate} \times 100\% = \eta_{\pi\epsilon\iota\rho} \times 100\%$ ⇒ usage = 55,25%

e) *Με βάση την εξίσωση που παρατίθεται νωρίτερα, υπολογίστε τη θεωρητική τιμή της χρησιμοποίησης του καναλιού, θεωρώντας ότι το μέγεθος των πακέτων αυξάνεται κατά 40 byte λόγω επικεφαλίδων TCP και IP, και ότι οι επαληθεύσεις (ACK) έχουν μέγεθος 40 byte. Ισχύει η εξίσωση; Αν όχι, πού οφείλεται η απόκλιση;*
Το μέγεθος των πακέτων είναι πλέον tcp_size’ = 2.040 bytes, οπότε:

- W = μέγεθος παραθύρου ⇒ W=6
- TRANSP’ = χρόνος μετάδοσης tcp πακέτου = $\frac{tcp_size\prime}{\max_bitrate} = \frac{2.040\ bytes \times 8}{10.000.000\ \frac{bit}{sec}} = \frac{16.320}{10.000.000}$sec ⇒ TRANSP’ = 0,001632 sec
- PROP = καθυστέρηση διάδοσης πακέτου = καθυστέρηση ζεύξης ⇒ PROP = 0,008 sec
- TRANSA = χρόνος μετάδοσης ack πακέτου = $\frac{ack_size}{\max_bitrate} = \frac{40\ bytes \times 8}{10.000.000\ \frac{bit}{sec}} = \frac{320}{10.000.000}$sec ⇒ TRANSA = 0,000032 sec
- S’ = TRANSP’ + 2×PROP + TRANSA = 0,001632 + 2×0,008 + 0,000032 ⇒ S’ = 0,017664 sec

Άρα, $\frac{\text{W} \times \text{TRANSP}\prime}{\text{S}\prime} = \frac{6 \times 0{,}001632\ sec}{0{,}017664\ \text{sec}} = \frac{0{,}009792\ sec}{0{,}017664\ \text{sec}} = 0{,}5543$

Οπότε η’ = min{0,5543, 1} = 0,5543
Παρατηρείται ότι η νέα απόδοση η’ είναι μεγαλύτερη αλλά πιο «κοντά» στην τιμή της $\eta_{\pi\epsilon\iota\rho}$.

f) *Διατηρώντας σταθερό το μέγεθος του παραθύρου, αλλάξτε το μήκος των πακέτων, ώστε η θεωρητική απόδοση του πρωτοκόλλου να λάβει τη μέγιστη τιμή της. Για ποιο μήκος πακέτων συμβαίνει αυτό; Υπολογίστε πειραματικά την απόδοση του πρωτοκόλλου (χρησιμοποίηση του καναλιού) για το μήκος πακέτου που προσδιορίσατε εδώ. Υπάρχει απόκλιση μεταξύ πειραματικής και θεωρητικής τιμής;*
η’’ = 1⇒ $\frac{\text{W} \times \text{TRANSP}\prime\prime}{\text{S}\prime\prime} = 1$ ⇒ W×TRANSP’’ = S’’ ⇒ 6×TRANSP’’ = TRANSP’’ + 2×PROP + TRANSA ⇒ 5×$\frac{tcp\_size\prime\prime}{\max\_bitrate}$ = 2×0,008 + 0,000032 ⇒ tcp_size’’ = $\frac{10.000.000\frac{bit}{sec} \times 0{,}016032\ \text{sec}}{5} = \frac{160.320}{5}$ bits = $\frac{20.040}{5}$ bytes ⇒ tcp_size’’ = 4.008 bytes
Ορίζουμε το μέγεθος πακέτου στο script ίσο με την παραπάνω τιμή και εκτελούμε το πρόγραμμα ανάλυσης awk.exe:

```
C:\Users\Chris Tsoufis\Desktop>awk95.exe -f lab4.awk < lab4.tr
Total Data received     : 5128776 Bytes
Total Packets received  : 1267
Last Packet received    : 5.219053 sec
```

Υπολογίζοντας ξανά τον ρυθμό μετάδοσης χρησιμοποιώντας τα νέα δεδομένα προκύπτει:
bitrate' = $\frac{8 \times 5.128.776 \text{ bytes}}{5{,}219053 - 1{,}1\ sec} = \frac{41.030.208\ bits}{4.119053\ sec} \approx 9.961.077{,}946\ \frac{bits}{sec}$ ⇒ bitrate' ≈ 9,961 Mbits
οπότε η απόδοση πρωτοκόλλου πειραματικά είναι:
usage' = $\frac{bitrate'}{\max\_bitrate} \times 100\% = \frac{9.961.077{,}946\ \frac{bits}{sec}}{10.000.000\ \frac{bits}{sec}} 100\%$ ⇒ usage' = 99,61%
τιμή πολύ κοντά στην μονάδα.

g) *Διατηρώντας το μήκος πακέτου που υπολογίσατε στο ερώτημα (f), αυξήστε (ΑΜ+6) (δηλ. 12) φορές το ρυθμό μετάδοσης της ζεύξης και ρυθμίστε το μέγεθος του παραθύρου, ώστε και πάλι η απόδοση να λάβει τη μέγιστη τιμή της. Για ποιο μέγεθος παραθύρου συμβαίνει αυτό; Πόσα περισσότερα bits απαιτούνται για την αναπαράσταση των αριθμών ακολουθίας πακέτων του πρωτοκόλλου Selective Repeat στην περίπτωση αυτή;*
Θα είναι:
max_bitrate''' = 12×max_bitrate = 6×10 Mbps ⇒ max_bitrate''' = 60 Mbps.
Πρέπει:
η''' = 1 ⇒ W'''×TRANSP''' = TRANSP''' + 2×PROP + TRANSA ⇒
(W'''- 1) $\frac{tcp\_size''}{\max\_bitrate'''}$ = 2×PROP + TRANSA ⇒
(W'''- 1) $\frac{4.008\ bytes}{60.000.000 \frac{bits}{sec}}$ = 0,016032 sec ⇒
W'''- 1 = $\frac{0{,}016032 \times 60.000.000\ bits}{4.008\ bytes \times 8} = \frac{961.920\ bits}{32.064\ bits}$ = 30 ⇒ W''' = 31
Είναι γνωστός για το μήκος παραθύρου ο τύπος:
W = $\frac{Max\_Seq+1}{2}$ ⇒ Max_Seq = 2×W – 1
Όπου Max_Seq είναι ο μέγιστος αριθμός μιας ακολουθίας από bits.
Έτσι, για το αρχικό μήκος παραθύρου W=6:
Max_Seq = 2×6 – 1 = 11 (1011 binary)
Άρα χρειάζονται συνολικά 4 bits, ενώ για μήκος παραθύρου W''' = 31:
Max_Seq''' = 2×31 – 1 = 61 (111101 binary)
Άρα χρειάζονται συνολικά 6 bits, δηλ. 2 bits περισσότερα.

h) *Εφαρμόστε τώρα το πρωτόκολλο για την παραμετροποίηση του ερωτήματος (g), θεωρώντας όμως ζεύξη με (ΑΜ+2) (δηλ. 8) φορές καθυστέρηση διάδοσης. Υπολογίστε την απόδοση του πρωτοκόλλου στη νέα αυτή ζεύξη τόσο θεωρητικά, όσο και πειραματικά. Αιτιολογείστε τυχόν αποκλίσεις που παρατηρούνται.*
Έχοντας σταθερό παράθυρο ίσο με W'''= 31 και σταθερό μήκος πακέτου ίσο με tcp_size'' = 4.008 bytes, αυξάνοντας την καθυστέρηση της ζεύξης (και άρα την καθυστέρηση διάδοσης) σε PROP'''' = 8×PROP = 8×0,008 sec ⇒ PROP'''' = 0,064 sec είναι λογικό πως η απόδοση του πρωτοκόλλου θα μειωθεί.

Η θεωρητική τιμή είναι:

$$\eta'''' = \frac{W \times TRANSP''}{S''''} = \frac{31 \times \frac{tcp\_size''}{\max\_bitrate''''}}{TRANSP''' + 2\times PROP'''' + TRANSA} = \frac{31 \times \frac{4.008\ bytes \times 8}{60.000.000\ \frac{bits}{sec}}}{\frac{4.008\ bytes \times 8}{60.000.000\ \frac{bits}{sec}} + 2\times 0{,}064\ sec + 0{,}000032\ \text{sec}} =$$

$$\frac{31 \times 32.064\ bits}{32.064\ bits + 7.680.000\ \text{bits} + 1.920\ bits} = \frac{993.984\ \text{bits}}{7.713.984\ \text{bits}} \Rightarrow \eta'''' \approx 0{,}128$$

Για την αντίστοιχη πειραματική τιμή γίνεται αλλαγή στο script στην καθυστέρηση ζεύξης σε 64 ms και εκτελείται το πρόγραμμα awk.exe:

```
C:\Users\Chris Tsoufis\Desktop>awk95.exe -f lab4.awk < lab4.tr
Total Data received     : 4015576 Bytes
Total Packets received  : 992
Last Packet received    : 5.229629 sec
```

Ο ρυθμός μετάδοσης πλέον μεταβάλλεται σε:

$$\text{bitrate}'''' = \frac{8 \times 4.015.576\ \text{bytes}}{5{,}229629 - 1{,}1\ \text{sec}} = \frac{32.124.608\ bits}{4{,}129629\ \text{sec}} \approx 7.779.054{,}244\ \frac{bits}{sec} \Rightarrow \text{bitrate}'''' \approx 7{,}779\ \text{Mbits}$$

Και η απόδοση του πρωτοκόλλου πειραματικά είναι:

$$\text{usage}'''' = \frac{bitrate''''}{\max\_bitrate''''} = \frac{7.779.054{,}244\ \frac{bits}{sec}}{60.000.000\ \frac{bits}{sec}} \Rightarrow \text{usage}'''' = 0{,}129$$

τιμή πολύ κοντά στη θεωρητική.

Πηγαίος κώδικας:

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

# Open the nam trace file
set nf [open lab4.nam w]
$ns namtrace-all $nf
set trf [open lab4.tr w]
$ns trace-all $trf

# Create two nodes
set n(0) [$ns node]
set n(1) [$ns node]
$ns at 0.0 "$n(0) label SRPsender_3117176 "
$ns at 0.0 "$n(1) label SRPreceiver_3117176"
set tcp0 [new Agent/TCP/Sack1]
# Set window size
$tcp0 set window_ 31
# The initial size of the congestion window on slow-start
$tcp0 set windowInit_ 31
# Disable modelling the initial SYN/SYNACK exchange
$tcp0 set syn_ false
# Set packet size
# Create a duplex link between the nodes
$ns duplex-link $n(0) $n(1) 60Mb 64ms DropTail
# The following line sets the queue limit of the two simplex links that connect node0 and node1 to the number specified.
$ns queue-limit $n(0) $n(1) 100
$ns queue-limit $n(1) $n(0) 100

$ns duplex-link-op $n(0) $n(1) orient right

# Create TCP agent
```

```
$tcp0 set packetSize_ 4008
# Attach TCP agent to node n(0)
$ns attach-agent $n(0) $tcp0
# Create TCP sink
set sink0 [new Agent/TCPSink]
# Attach TCP sink to node n(1)
$ns attach-agent $n(1) $sink0
# Connect TCP agent to sink
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

# Define a 'finish' procedure
proc finish {} {
global ns nf trf
$ns flush-trace
# Close the trace file
close $nf
close $trf
exit 0
}

# Events
$ns at 1.1 "$ftp0 start"
$ns at 5.2 "$ftp0 stop"
$ns at 6.0 "finish"

# Run the simulation
$ns run
```

i) *Εφαρμόστε το πρωτόκολλο Go Back N αντί του Selective Repeat στην τελευταία παραμετροποίηση της προσομοίωσης και μετρήστε την απόδοση του πρωτοκόλλου αυτού πειραματικά. Διαφέρουν οι πειραματικές αποδόσεις των δύο πρωτοκόλλων; Γιατί;*
Για να χρησιμοποιηθεί το πρωτόκολλο Go Back N αντί του Selective Repeat γίνεται αλλαγή της γραμμή κώδικα για τον ορισμό του TCP agent αποστολής set tcp0 [new Agent/TCP/Sack1] σε set tcp0 [new Agent/TCP/Reno].
Γίνεται εκτέλεση του προγράμματος ανάλυσης awk.exe για να εντοπισθούν τυχόν διαφορές:

```
C:\Users\Chris Tsoufis\Desktop>awk95.exe -f lab4.awk < lab4.tr
Total Data received     : 4015576 Bytes
Total Packets received  : 992
Last Packet received    : 5.229629 sec
```

Παρατηρείται ότι τα αποτελέσματα δεν άλλαξαν κι αυτό οφείλεται στο γεγονός ότι η προσομοίωσή δεν έχει σφάλματα, όπως π.χ. διακοπές της ζεύξης. Η καλύτερη λειτουργία και απόδοση του πρωτοκόλλου Selective Repeat έναντι του Go Back N θα φαινόταν ξεκάθαρα στην αντίθετη περίπτωση, αφού το πρώτο θα έστελνε ξανά ακριβώς όσα πακέτα καταστράφηκαν ή χάθηκαν σε αντίθεση με το δεύτερο.