ΔΙΚΤΥΑ ΕΠΙΚΟΙΝΩΝΙΩΝ

Εργαστηριακή Άσκηση 4

Επίδοση πρωτοκόλλου Selective Repeat

1. Πρωτόκολλο Selective Repeat

Σε αυτή την εργαστηριακή άσκηση θα μελετηθεί η επίδοση του πρωτοκόλλου *Selective Repeat* (πρωτόκολλο επιλεκτικής επανάληψης με αναμετάδοση). Η λειτουργία του πρωτοκόλλου *Selective Repeat* περιγράφεται αναλυτικά στο βιβλίο «Δίκτυα Υπολογιστών» του A. Tanenbaum (Έκδοση 5η), ενότητα 3.4.3, αποσπάσματα από την οποία παραθέτονται στην ακόλουθη υποενότητα. Το πρωτόκολλο αυτό μπορεί να χρησιμοποιηθεί, εκτός από το στρώμα ζεύξης δεδομένων, και στο στρώμα μεταφοράς, ώστε να παρέχεται υπηρεσία με εγγυημένη παράδοση δεδομένων πάνω από αναξιόπιστο δίκτυο.

1.1 Θεωρητικό Υπόβαθρο

Το πρωτόκολλο *Go Back N* που είδαμε στην προηγούμενη εργαστηριακή άσκηση δουλεύει καλά όταν τα σφάλματα είναι σπάνια. Όταν όμως η γραμμή είναι κακή, σπαταλά πολύ εύρος ζώνης σε αναμεταδόσεις πλαισίων. Μια εναλλακτική στρατηγική για την αντιμετώπιση των σφαλμάτων είναι να επιτρέπουμε στον παραλήπτη να αποδέχεται και να αποθηκεύει προσωρινά τα πλαίσια που ακολουθούν ένα κατεστραμμένο ή χαμένο πλαίσιο. Ένα τέτοιο πρωτόκολλο δεν απορρίπτει πλαίσια απλώς και μόνο επειδή ένα προηγούμενο πλαίσιο καταστράφηκε ή χάθηκε.

Στο πρωτόκολλο αυτό, τόσο ο αποστολέας όσο και ο παραλήπτης διατηρούν ένα παράθυρο με αποδεκτούς αριθμούς ακολουθίας. Το μέγεθος του παραθύρου αποστολής αρχίζει στο 0 και φτάνει μέχρι κάποιο προκαθορισμένο μέγιστο MAX_SEQ. Αντίθετα, το παράθυρο λήψης έχει σταθερό μέγεθος και είναι ίσο με MAX_SEQ. Ο παραλήπτης δεσμεύει ένα χώρο προσωρινής αποθήκευσης για κάθε αριθμό ακολουθίας μέσα στο σταθερό του παράθυρο. Με κάθε περιοχή προσωρινής αποθήκευσης σχετίζεται ένα bit που δείχνει αν η περιοχή είναι γεμάτη ή άδεια. Όποτε φτάνει ένα πλαίσιο, ελέγχεται ο αριθμός ακολουθίας του για να δούμε αν βρίσκεται εντός του παραθύρου. Αν συμβαίνει αυτό και αν το πλαίσιο δεν έχει ήδη παραληφθεί, γίνεται αποδεκτό και αποθηκεύεται. Η ενέργεια αυτή γίνεται χωρίς να εξετάζουμε αν περιέχει το επόμενο πακέτο που αναμένεται από το επίπεδο δικτύου ή όχι. Φυσικά, το πλαίσιο θα πρέπει να παραμείνει στο επίπεδο συνδέσμου μετάδοσης δεδομένων και να μη μεταβιβαστεί στο επίπεδο δικτύου μέχρι να παραδοθούν με τη σωστή σειρά στο επίπεδο δικτύου όλα τα πλαίσια με χαμηλότερους αριθμούς ακολουθίας.

Η μη σειριακή λήψη δημιουργεί μερικά προβλήματα τα οποία δεν παρουσιάζονται στα πρωτόκολλα όπου τα πλαίσια γίνονται αποδεκτά μόνο με τη σωστή σειρά, όπως το *Stop and Wait* και το *Go Back N*. Για παράδειγμα, μια αιτία προβλήματος είναι εάν όταν ο παραλήπτης προχώρα το παράθυρο του, η νέα περιοχή έγκυρων αριθμών ακολουθίας επικαλύπτει την παλιά. Έτσι, η επόμενη δέσμη πλαισίων θα μπορούσε να είναι είτε αντίγραφα (αν χάθηκαν όλες οι επιβεβαιώσεις) είτε νέα πλαίσια (αν έφτασαν όλες οι επιβεβαιώσεις). Ο παραλήπτης δεν μπορεί δυστυχώς να διακρίνει ανάμεσα στις δύο αυτές περιπτώσεις.

Ο τρόπος με τον οποίο λύνεται το πρόβλημα είναι να εξασφαλίσουμε ότι, όταν ο παραλήπτης θα προχωρήσει το παράθυρο του, δεν θα υπάρχει επικάλυψη με το αρχικό παράθυρο. Για να εξασφαλίσουμε ότι δεν υπάρχει επικάλυψη, το μέγιστο μέγεθος του παραθύρου αποστολής/λήψης θα πρέπει να είναι το πολύ το μισό του εύρους των αριθμών ακολουθίας. Για παράδειγμα, αν χρησιμοποιούνται 4 bit για αριθμούς ακολουθίας, οι αριθμοί θα έχουν τιμές από 0 έως 15. Άρα θα πρέπει ανά πάσα στιγμή να εκκρεμούν μόνο οκτώ μη επιβεβαιωμένα πλαίσια. Με αυτόν τον τρόπο, αν ο παραλήπτης έχει μόλις

αποδεχτεί τα πλαίσια 0 έως 7 και έχει προχωρήσει το παράθυρο του για να επιτρέψει την αποδοχή των πλαισίων 8 έως 15, θα μπορεί να διακρίνει χωρίς αμφιβολία αν τα επόμενα πλαίσια είναι αναμεταδόσεις (0 έως 7) ή νέα πλαίσια (8 έως 15). Γενικά, το μέγεθος του παραθύρου για το πρωτόκολλο *Selective Repeat* θα είναι (MAX_SEQ+1)/2. Έτσι, για αριθμούς ακολουθίας των 3 bit, το μέγεθος του παραθύρου είναι τέσσερα, ενώ για αριθμούς ακολουθίας των 4 bit, το μέγεθος του παραθύρου είναι οκτώ.

Μια ενδιαφέρουσα ερώτηση είναι η εξής: πόσες περιοχές προσωρινής αποθήκευσης πρέπει να έχει ο παραλήπτης; Σε καμία περίπτωση δεν πρόκειται να αποδεχθεί πλαίσια των οποίων οι αριθμοί ακολουθίας βρίσκονται πάνω από το άνω όριο του παραθύρου λήψης. Κατά συνέπεια, το πλήθος περιοχών προσωρινής αποθήκευσης που απαιτούνται είναι ίσο με το μέγεθος του παραθύρου λήψης, και όχι ίσο με το εύρος των αριθμών ακολουθίας. Στο παραπάνω παράδειγμα με αριθμούς ακολουθίας των 4 bit, απαιτούνται οκτώ περιοχές με αριθμούς 0 έως 7. Όταν φτάνει το πλαίσιο i τοποθετείται στην περιοχή i mod 8.

Το πρωτόκολλο *Selective Repeat* χρησιμοποιεί μια πιο αποτελεσματική στρατηγική από το πρωτόκολλο *Go Back N* για την αντιμετώπιση των σφαλμάτων. Όποτε ο παραλήπτης έχει λόγους να υποπτευθεί ότι έχει συμβεί κάποιο σφάλμα, στέλνει μια αρνητική επιβεβαίωση (NAK) στον αποστολέα. Ένα τέτοιο πλαίσιο είναι μια αίτηση αναμετάδοσης του πλαισίου που προσδιορίζεται στο NAK. Υπάρχουν δύο περιπτώσεις στις οποίες ο παραλήπτης πρέπει να υποψιαστεί ότι έχει συμβεί κάποιο σφάλμα: όταν φτάνει ένα κατεστραμμένο πλαίσιο ή όταν φτάνει ένα πλαίσιο διαφορετικό από αυτό που αναμενόταν (υπάρχει πιθανότητα χαμένου πλαισίου). Για να αποφευχθούν οι πολλαπλές αιτήσεις για την αναμετάδοση του ίδιου χαμένου πλαισίου, ο παραλήπτης πρέπει να παρακολουθεί αν έχει ήδη στείλει ένα πλαίσιο NAK για το συγκεκριμένο πλαίσιο δεδομένων. Η μεταβλητή no_nak στο πρωτόκολλο *Selective Repeat* είναι αληθής αν δεν έχει σταλεί ακόμη κάποιο πλαίσιο NAK για το πλαίσιο με αριθμό frame_expected. Αν το NAK παραμορφωθεί ή χαθεί δεν υπάρχει πρόβλημα, αφού κάποτε θα λήξει ο χρόνος αναμονής του αποστολέα και έτσι αυτός θα αναμεταδώσει σε κάθε περίπτωση το χαμένο πλαίσιο. Αν φτάσει κάποιο εσφαλμένο πλαίσιο αφού έχει σταλεί ένα πλαίσιο NAK που χάθηκε, η μεταβλητή no_nak θα είναι αληθής οπότε θα ξεκινήσει το βοηθητικό χρονόμετρο. Όταν λήξει, θα σταλεί μια επιβεβαίωση έτσι ώστε να συγχρονίσει τον αποστολέα με την τρέχουσα κατάσταση του παραλήπτη.

1.2 Προετοιμασία για την Άσκηση

Για να αρχίσετε, θα πρέπει να δημιουργήσετε ένα αρχείο, π.χ. “lab4.tcl”, με τον τρόπο που περιγράφεται σε προηγούμενες ασκήσεις. Ως γνωστό, αυτός ο κώδικας θα είναι πάντοτε παρόμοιος. Θα πρέπει πάντοτε να δημιουργείτε ένα αντικείμενο προσομοίωσης, να αρχίζετε την προσομοίωση με την ίδια εντολή και, αν θέλετε να παρακολουθήσετε την προσομοίωση γραφικά, θα πρέπει να χρησιμοποιήσετε το NAM.

Όπως και στην προηγούμενη άσκηση, για την ανάλυση των αποτελεσμάτων της προσομοίωσης θα χρησιμοποιηθεί η γλώσσα προγραμματισμού awk. Πρόκειται, όπως είδαμε, για γλώσσα script που επιτρέπει την ανάλυση και επεξεργασία δεδομένων από αρχεία με απλό και αποτελεσματικό τρόπο. Για το σκοπό αυτό θα δημιουργήσουμε ένα αρχείο με κατάληξη .awk που θα περιγράφει τις διαδικασίες που απαιτούνται για την ανάλυση των δεδομένων. Το αρχείο αυτό εκτελείται με τη βοήθεια του προγράμματος awk.exe.

Σημείωση: Είναι σκόπιμο να ανατρέξετε στο φυλλάδιο της Εργαστηριακής Άσκησης 1, για να θυμηθείτε τις διαδικασίες με τις οποίες σώζονται και τρέχουν τα αρχεία. Θυμίζουμε ότι πριν αρχίσετε την προσομοίωση με ένα αρχείο “*.tcl” πρέπει να το σώσετε (File > Save). Για να εμφανιστεί ο δρομέας (cursor) στο “Command Prompt”, όταν είναι ανοιχτά το NAM ή το Xgraph, πρέπει πρώτα να κλείσετε τα παράθυρά τους. Κλείστε τα μόνο πατώντας το “X” δεξιά επάνω στην μπάρα του παραθύρου, όχι από το μενού File. Στο τέλος του εργαστηρίου κάντε “log off” πριν φύγετε. Στο επόμενο εργαστήριο θα πρέπει να παραδώσετε μια αναφορά με απαντήσεις και σχόλια σχετικά με τις ερωτήσεις αυτής της Εργαστηριακής Άσκησης.

2. Σενάριο προσομοίωσης

2.1 Αρχικοποίηση προσομοίωσης – Δημιουργία αρχείου ίχνους

Αρχικά πρέπει να δημιουργηθεί το αντικείμενο της προσομοίωσης.

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

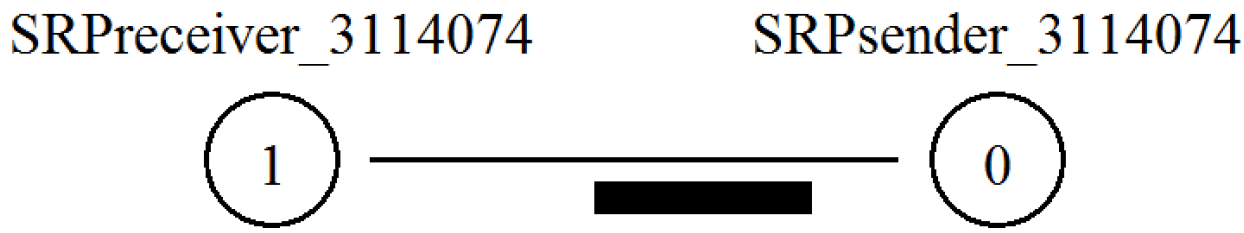
Όπως και σε προηγούμενες ασκήσεις, προκειμένου να επαληθεύσουμε την ορθότητα του σεναρίου προσομοίωσης, θα χρησιμοποιήσουμε αρχεία ίχνους (trace files), στα οποία αποθηκεύονται όλα τα γεγονότα που λαμβάνουν χώρα κατά τη διάρκεια της προσομοίωσης.

```
# 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

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

2.2 Τοπολογία

Η τοπολογία της προσομοίωσης είναι η απλούστερη δυνατή, με δύο κόμβους n(0) και n(1) να συνδέονται μεταξύ τους με μία πλήρως αμφίδρομη ζεύξη, όπως απεικονίζεται στο Σχήμα 1. Προσοχή: στα ονόματα των κόμβων θα πρέπει να **ενσωματώσετε και τον αριθμό μητρώου σας**.



Σχήμα 1 - Αναπαράσταση της τοπολογίας των δύο κόμβων στο NAM

Create two nodes

```
set n(0) [$ns node]
set n(1) [$ns node]


$ns at 0.0 "$n(0) label SRPsender MyStudentRecordNumber "
$ns at 0.0 "$n(1) label SRPreceiver_MyStudentRecordNumber"
```

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
```

2.3 Το στρώμα μεταφοράς

Το πρωτόκολλο *Selective Repeat* εμπεριέχεται στο πρωτόκολλο μεταφοράς TCP στην παραμετροποίηση SACK (Selective Acknowledgement). Για το λόγο αυτό θα δημιουργήσουμε μια TCP σύνδεση από τον κόμβο n(0) στον κόμβο n(1). Αυτό γίνεται με τις ακόλουθες εντολές:

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
```

2.4 Το στρώμα εφαρμογής

Θεωρούμε ως εφαρμογή που θα δημιουργήσει την κίνηση, τη μεταφορά ενός αρχείου απείρου μεγέθους με FTP, έτσι ώστε να υπάρχει πάντα πληροφορία για μετάδοση μεταξύ των χρόνων 0.3 sec και 5.0 sec. Η δημιουργία της γεννήτριας κίνησης FTP και η σύνδεσή της με το στρώμα μεταφοράς γίνονται με τις παρακάτω εντολές:

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

2.5 Εκτέλεση του σεναρίου

Τέλος, ορίζουμε τα γεγονότα της προσομοίωσης:

Events

```
$ns at 0.3 "$ftp0 start"
$ns at 5.0 "$ftp0 stop"
$ns at 6.0 "finish"
```

Run the simulation

```
$ns run
```

Με βάση τα παραπάνω, τη χρονική στιγμή t = 0,3 sec, ο FTP agent αρχίζει την αποστολή των πακέτων, ενώ όταν t = 5,0 sec, η αποστολή ολοκληρώνεται. Αφού σώσουμε το αρχείο, το εκτελούμε με την εντολή **ns lab4.tcl**. Με την εκτέλεση αυτής της εντολής θα πρέπει να έχουν δημιουργηθεί τα αρχεία **lab4.nam** και **lab4.tr.** Με την εντολή **nam lab4.nam** μπορούμε να δούμε την τοπολογία του δικτύου καθώς και την κίνηση που έχει δημιουργηθεί. Το αρχείο **lab4.tr** περιέχει πληροφορίες για όλα τα γεγονότα που συνέβησαν κατά την προσομοίωση.

3. Ανάλυση αρχείου ίχνους (trace file)

Αφού έχουμε δημιουργήσει το σενάριο προσομοίωσης, το έχουμε εκτελέσει και έχουμε δημιουργήσει τα αρχεία αποτελεσμάτων, τα αρχεία αυτά πρέπει να αναλυθούν ώστε να πάρουμε τις πληροφορίες που θέλουμε.

3.1 Μορφή αρχείου ίχνους (trace file)

Το αρχείο **lab4.tr** που δημιουργήθηκε προηγουμένως περιέχει πληροφορίες της παρακάτω μορφής:

```
- 0.507432 1 0 ack 40 ------- 0 1.0 0.0 1 5
r 0.508664 0 1 tcp 2040 ------- 0 0.0 1.0 2 2
+ 0.508664 1 0 ack 40 ------- 0 1.0 0.0 2 6
- 0.508664 1 0 ack 40 ------- 0 1.0 0.0 2 6
r 0.509896 0 1 tcp 2040 ------- 0 0.0 1.0 3 3
+ 0.509896 1 0 ack 40 ------- 0 1.0 0.0 3 7
- 0.509896 1 0 ack 40 ------- 0 1.0 0.0 3 7
r 0.511232 1 0 ack 40 ------- 0 1.0 0.0 0 4
+ 0.511232 0 1 tcp 2040 ------- 0 0.0 1.0 4 8
```

Όπως είδαμε και στην προηγούμενη εργαστηριακή άσκηση, κάθε γραμμή του αρχείου αυτού αντιστοιχεί σε ένα γεγονός που συνέβη κατά τη διάρκεια της προσομοίωσης. Ο πρώτος χαρακτήρας κάθε γραμμής υποδηλώνει το είδος του γεγονότος που συνέβη. Ο χαρακτήρας "+" σημαίνει είσοδο του πακέτου σε ουρά αναμονής, ο χαρακτήρας "-" σημαίνει αποχώρηση από ουρά αναμονής, ο χαρακτήρας "r" σημαίνει επιτυχημένη λήψη πακέτου, και ο χαρακτήρας "d" σημαίνει απόρριψη πακέτου.

Η δεύτερη λέξη της κάθε γραμμής είναι η χρονική στιγμή κατά την οποία συνέβη το γεγονός που καταγράφεται. Οι επόμενες δύο λέξεις περιγράφουν μεταξύ ποιων κόμβων βρίσκεται το πακέτο, η επόμενη λέξη είναι το είδος του πακέτου και η έκτη λέξη περιγράφει το μέγεθος του πακέτου (συμπεριλαμβάνονται οι επικεφαλίδες TCP και IP). Οι παύλες αντιστοιχούν σε πεδία που δεν χρησιμοποιούνται στο παράδειγμα.

Ο πρώτος αριθμός μετά τις παύλες είναι το αναγνωριστικό της ροής (flow ID), στην οποία ανήκει το πακέτο. Το flow ID ακολουθείται από την διεύθυνση αποστολέα και προορισμού (IP.port), από τον αύξοντα αριθμό (sequence number) του πακέτου και τέλος από έναν μοναδικό αριθμό (unique number) του πακέτου.

3.2 Ανάλυση με τη γλώσσα awk

Όπως είδαμε και στην προηγούμενη εργαστηριακή άσκηση, η γλώσσα awk είναι σχεδιασμένη ώστε να επιτρέπει την εύκολη ανάλυση αρχείων με δεδομένα. Ένα πρόγραμμα awk αποτελείται από τρία τμήματα. Το πρώτο τμήμα του προγράμματος ορίζεται με την εντολή BEGIN {} και περιλαμβάνει όλες τις εντολές που θα γίνουν μία φορά, κατά την εκκίνηση της του προγράμματος. Εδώ μπορούν να αρχικοποιηθούν μεταβλητές, να ανοιχθούν αρχεία, κ.λπ.

Το δεύτερο τμήμα του προγράμματος αποτελείται από ένα σύνολο από κανόνες που θα εκτελεστούν για κάθε γραμμή του αρχείου εισόδου. Αυτοί οι κανόνες αποτελούνται από δύο κομμάτια. Το πρώτο κομμάτι ορίζει σε ποιες γραμμές του αρχείου εισόδου αναφέρεται ο κανόνας, και το δεύτερο ορίζει ποιες λειτουργίες θα πραγματοποιηθούν για αυτές τις γραμμές. Για παράδειγμα ο κανόνας:

```
/^r/&&/tcp/ {
      packets++;
```

```
	data+=$6;
}
```

ορίζει ότι, όταν συναντήσουμε μια γραμμή του αρχείου εισόδου που να ξεκινάει με τον χαρακτήρα "r" (/^r/) και (&&) που περιλαμβάνει τη λέξη tcp (/tcp/), τότε η τιμή της μεταβλητής packets αυξάνεται κατά 1, και η τιμή της μεταβλητής data αυξάνεται κατά την τιμή που βρίσκεται στην έκτη λέξη ($6) της γραμμής την οποία εξετάζουμε.

Το τρίτο τμήμα ορίζεται από την εντολή END{} και περιλαμβάνει τις λειτουργίες που θα πραγματοποιηθούν αφού διαβαστεί ολόκληρο το αρχείο εισόδου, στο τέλος της εκτέλεσης του προγράμματος.

3.3 Υπολογισμός του πλήθους των πακέτων και της ποσότητας των δεδομενων που ελήφθησαν

Για να μπορούμε να υπολογίσουμε την επίδοση του πρωτοκόλλου *Selective Repeat* πρέπει να μετρήσουμε την ποσότητα των δεδομένων που ελήφθησαν κατά τη διάρκεια της προσομοίωσης. Για κάθε πακέτο που έλαβε ο αποδέκτης, η ποσότητα των δεδομένων αυξάνεται κατά το μέγεθος του πακέτου. Συνεπώς πρέπει να εντοπίσουμε τις γραμμές του αρχείου ίχνους που φανερώνουν ορθή λήψη πακέτου TCP. Αυτές οι γραμμές αρχίζουν με τον χαρακτήρα "r" και περιλαμβάνουν την λέξη tcp. Για κάθε τέτοια γραμμή που εντοπίζεται η μεταβλητή packets (αριθμός πακέτων που ελήφθησαν) αυξάνεται κατά ένα, ενώ η μεταβλητή data (πλήθος δεδομένων που ελήφθησαν) αυξάνεται κατά το μέγεθος του πακέτου. Έτσι δημιουργείται το παρακάτω πρόγραμμα awk:

```
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);
}
```

3.4 Εκτέλεση του προγράμματος ανάλυσης

Για την εκτέλεση προγραμμάτων awk, χρησιμοποιείται ο διερμηνέας (interpreter) awk με παραμέτρους το όνομα του αρχείου που περιγράφει τις διαδικασίες της ανάλυσης και το όνομα του αρχείου που περιλαμβάνει τα δεδομένα που θα αναλυθούν. Εάν ο παραπάνω κώδικας έχει αποθηκευτεί στο αρχείο **lab4.awk** και τα δεδομένα βρίσκονται στο αρχείο **lab4.tr**, τότε εκτελούμε την εντολή:

```
awk.exe -f lab4.awk < lab4.tr
```

Αυτή η εντολή θα εμφανίσει στην οθόνη τον αριθμό των πακέτων και το πλήθος των δεδομένων που ελήφθησαν.

4. Μελέτη της απόδοσης Selective Repeat

4.1 Ερωτήσεις

(α) 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[1] sec, ενώ όταν t = 4,0 + 0,2*AM sec, η αποστολή να ολοκληρώνεται;

(β) Να επαληθεύσετε κατά πόσον ισχύει ή όχι η εξίσωση σε περίπτωση απουσίας σφαλμάτων:

$$\eta = \min\left\{\frac{W \times TRANSP}{S}, 1\right\}$$

Όπου:

η → απόδοση πρωτοκόλλου, η οποία μετράται πειραματικά από την ακόλουθη σχέση: (πραγματικός ρυθμός μετάδοσης δεδομένων)/(ρυθμός μετάδοσης ζεύξης)

W → μέγεθος παραθύρου

TRANSP → χρόνος μετάδοσης ενός πακέτου δεδομένων, ο οποίος ισούται με (μήκος πακέτου)/(ρυθμός μετάδοσης)

PROP → καθυστέρηση διάδοσης του πακέτου

TRANSA → χρόνος μετάδοσης μιας επιβεβαίωσης

S → ολικός χρόνος που απαιτείται για τη μετάδοση ενός πακέτου και της σχετικής επιβεβαίωσης, ο οποίος ισούται με: ***S*** = ***TRANSP*** + 2 x ***PROP*** + ***TRANSA***

(γ) Ποιος είναι ο αριθμός των πακέτων που παρελήφθησαν; Πόσα δεδομένα παρελήφθησαν από τον παραλήπτη κατά τη διάρκεια της προσομοίωσης;

(δ) 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.

(ε) 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;

(στ) 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;

(ζ) Διατηρώντας το μήκος πακέτου που υπολογίσατε στο ερώτημα (στ), αυξήστε (AM+6) 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

[1] AM είναι το τελευταίο ψηφίο του αριθμού μητρώου σας.

απαιτούνται για την αναπαράσταση των αριθμών ακολουθίας πακέτων του πρωτοκόλλου *Selective Repeat* στην περίπτωση αυτή;

(η) Εφαρμόστε τώρα το πρωτόκολλο για την παραμετροποίηση του ερωτήματος (ζ), θεωρώντας όμως ζεύξη με (ΑΜ+2) φορές καθυστέρηση διάδοσης. Υπολογίστε την απόδοση του πρωτοκόλλου στη νέα αυτή ζεύξη τόσο θεωρητικά, όσο και πειραματικά. Αιτιολογείστε τυχόν αποκλίσεις που παρατηρούνται.

(θ) 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 όλες τις προσομοιώσεις που χρειάζεται να εκτελέσετε για να απαντήσετε στα παραπάνω ερωτήματα, να συμπεριλάβετε τον αντίστοιχο κώδικα.