# ΔΙΚΤΥΑ ΕΠΙΚΟΙΝΩΝΙΩΝ

# Εργαστηριακή Άσκηση 4

### *Πρωτόκολλα Stop and Wait και Go Back N*

## *1. Πρωτόκολλα κυλιομένου παραθύρου*

Στα πρωτόκολλα κυλιόμενου παραθύρου, κάθε αποστελλόμενο πακέτο περιέχει έναν αύξοντα αριθμό που μεταβάλλεται από το *0* μέχρι κάποια μέγιστη τιμή. Η μέγιστη τιμή είναι συνήθως $2^n$*-1*, οπότε ο αύξων αριθμός χωρά άνετα σε πεδίο των *n bit*. Το πρωτόκολλο παύσης και αναμονής (stop-and-wait) χρησιμοποιεί *n = 1*, περιορίζοντας τους αύξοντες αριθμούς στο *0* και στο *1*, αλλά οι περισσότερο προωθημένες παραλλαγές μπορούν να χρησιμοποιήσουν αυθαίρετη τιμή του *n*.

Η ουσία όλων των πρωτοκόλλων κυλιόμενου παραθύρου είναι ότι σε κάθε χρονική στιγμή ο πομπός διατηρεί ένα σύνολο αυξόντων αριθμών που αντιστοιχούν σε πακέτα που μπορεί να στείλει. Αυτά τα πακέτα λέγεται ότι πέφτουν μέσα στο παράθυρο αποστολής. Παρομοίως ο δέκτης διατηρεί επίσης ένα παράθυρο λήψης, που αντιστοιχεί στο σύνολο των πακέτων που του επιτρέπεται να δεχθεί. Το παράθυρο του πομπού και το παράθυρο του δέκτη, δεν είναι ανάγκη να έχουν το ίδιο κατώτερο και ανώτερο όριο ή ακόμη να έχουν το ίδιο μέγεθος. Σε μερικά πρωτόκολλα, έχουν σταθερό μέγεθος, αλλά σε άλλα μπορεί να μεγαλώνουν ή να μικραίνουν καθώς στέλνονται ή λαμβάνονται πακέτα.

Οι αύξοντες αριθμοί μέσα στο παράθυρο αποστολής του πομπού αντιπροσωπεύουν πακέτα που στάλθηκαν, αλλά δεν έχουν επιβεβαιωθεί ακόμη. Στον πομπό, κάθε φορά που φθάνει ένα νέο πακέτο προς αποστολή, δίδεται σε αυτό ο επόμενος μεγαλύτερος αύξων αριθμός και το άνω όριο του παραθύρου αυξάνεται κατά ένα. Όταν φθάνει στον πομπό μια επιβεβαίωση, το κάτω όριο του παραθύρου αυξάνεται κατά ένα. Κατ' αυτόν τον τρόπο, το παράθυρο αποστολής διατηρεί διαρκώς τη λίστα των πακέτων που εστάλησαν και δεν επιβεβαιώθηκαν ακόμα.

Επειδή τα πακέτα που βρίσκονται κάποια στιγμή στο παράθυρο αποστολής μπορεί τελικά να χαθούν ή να καταστραφούν κατά τη μετάδοσή τους, ο πομπός πρέπει να κρατάει όλα αυτά τα πακέτα στη μνήμη του για πιθανή αναμετάδοση. Έτσι, εάν το μέγιστο μέγεθος παραθύρου είναι *n*, ο πομπός χρειάζεται *n* χώρους προσωρινής αποθήκευσης για να κρατά τα πακέτα που δεν έχουν ακόμη επιβεβαιωθεί. Εάν το παράθυρο φτάσει στο μέγιστο μέγεθός του, ο πομπός πρέπει να σταματήσει την αποστολή μέχρι να ελευθερωθεί μία θέση προσωρινής αποθήκευσης.

Το παράθυρο λήψης του δέκτη αντιστοιχεί στον αριθμό των πακέτων που μπορεί να λάβει ο δέκτης. Κάθε λαμβανόμενο πακέτο που πέφτει έξω από το παράθυρο απορρίπτεται χωρίς σχόλια. Όταν λαμβάνεται ένα πακέτο του οποίου ο αύξων αριθμός είναι ίσος με το κάτω όριο του παραθύρου λήψης, περνά στο ανώτερο στρώμα, παράγεται μια επιβεβαίωσή του και το παράθυρο ολισθαίνει κατά μία θέση. Το παράθυρο του δέκτη παραμένει πάντοτε στο αρχικό του μέγεθος, αντίθετα προς το παράθυρο του πομπού. Σημειώστε ότι μέγεθος παραθύρου λήψης ίσο με *1* σημαίνει ότι ο δέκτης δέχεται μόνο τα πακέτα που έρχονται με τη σωστή σειρά. Αυτό όμως δεν ισχύει στα μεγαλύτερα παράθυρα. Αντίθετα, τα δεδομένα διοχετεύονται στο ανώτερο στρώμα πάντοτε με τη σωστή σειρά, ανεξάρτητα από το μέγεθος του παραθύρου του κατωτέρου στρώματος.

## *2. Πρωτόκολλα Stop and Wait και Go Back N*

Σε αυτή την εργαστηριακή άσκηση θα συγκριθεί η επίδοση του πρωτοκόλλου Go back N (GBN) με αυτή του Stop and Wait (SW). Στο πρώτο πρωτόκολλο, επιτρέπουμε στον πομπό να μεταδώσει μέχρι *w* πακέτα προτού σταματήσει, αντί μόνο ενός που ισχύει στο SW. Αν καταστραφεί ή χαθεί ένα πακέτο στη μέση ενός μεγάλου συρμού, ο δέκτης απλά απορρίπτει όλα τα πακέτα που ακολουθούν, χωρίς να στείλει επιβεβαιώσεις

για τα απορριφθέντα πακέτα. Αυτή η στρατηγική αντιστοιχεί σε παράθυρο λήψης μεγέθους *1*. Με άλλα λόγια, ο δέκτης δε δέχεται κανένα πακέτο εκτός από το επόμενο πακέτο που πρέπει να παραδώσει στο ανώτερο στρώμα. Τελικά, ο πομπός θα εξαντλήσει τον χρόνο του και θα αναμεταδώσει όλα τα ανεπιβεβαίωτα πακέτα με τη σειρά, ξεκινώντας με το κατεστραμμένο ή το χαμένο. Αυτή η προσέγγιση μπορεί να σπαταλήσει μεγάλο μέρος του εύρους ζώνης, εάν ο ρυθμός εμφάνισης σφαλμάτων είναι υψηλός.

Αν και το πρωτόκολλο δεν καταχωρεί σε προσωρινή μνήμη στον δέκτη τα πακέτα που φθάνουν μετά από κάποιο λάθος, δεν μπορεί να αποφύγει εντελώς το πρόβλημα του χώρου προσωρινής αποθήκευσης. Εφόσον ο πομπός μπορεί να χρειασθεί να μεταδώσει ξανά όλα τα ανεπιβεβαίωτα πακέτα σε κάποια μελλοντική χρονική στιγμή, θα πρέπει να κρατά όλα τα μεταδοθέντα πακέτα, μέχρι να μάθει μετά βεβαιότητας ότι έχουν ληφθεί από τον δέκτη. Όταν φθάσει επιβεβαίωση του πακέτου *n*, επιβεβαιώνεται επίσης, αυτόματα, η λήψη των πακέτων *n-1*, *n-2*, κ.ο.κ. Η ιδιότητα αυτή είναι ιδιαίτερα σημαντική, όταν μερικά από τα προηγούμενα πακέτα που μετέφεραν επιβεβαιώσεις έχουν χαθεί ή παραμορφωθεί.

Η λειτουργία των πρωτοκόλλων Stop and Wait και Go back N περιγράφονται αναλυτικά στο βιβλίο «Δίκτυα Υπολογιστών» (Andrew Tanenbaum, 4η Έκδοση, Εκδόσεις Κλειδάριθμος), στις ενότητες 3.3.3 και 3.4.2 αντιστοίχως. Τα πρωτόκολλα αυτά μπορούν να χρησιμοποιηθούν, εκτός από το στρώμα ζεύξης δεδομένων, και στο στρώμα μεταφοράς, ώστε να παρέχεται υπηρεσία με εγγυημένη παράδοση δεδομένων πάνω από αναξιόπιστο δίκτυο.

Για να αρχίσετε, θα πρέπει να δημιουργήσετε ένα αρχείο, π.χ. "`lab4.tcl`", με τον τρόπο που περιγράφεται σε προηγούμενες ασκήσεις. Θα πρέπει πάντοτε να δημιουργείτε ένα αντικείμενο προσομοίωσης, να αρχίζετε την προσομοίωση με την εντολή `ns` και, αν θέλετε να μελετήσετε τα αποτελέσματα της προσομοίωσης με το NAM ή/και το Xgraph, θα πρέπει να δημιουργείτε κατάλληλα αρχεία trace "`.tr`" ή "`.nam`", να τα αρχικοποιείτε και να ορίζετε μια διαδικασία που να τα κλείνει.

Ένας εύκολος τρόπος για την ανάλυση των αποτελεσμάτων της προσομοίωσης, είναι η γλώσσα προγραμματισμού awk. Αυτή η γλώσσα είναι μια πολύ απλή 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.

## *3. Σενάριο προσομοίωσης*

### *3.1 Αρχικοποίηση προσομοίωσης – Δημιουργία αρχείου ίχνους*

Αρχικά πρέπει να δημιουργηθεί το αντικείμενο της προσομοίωσης.

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

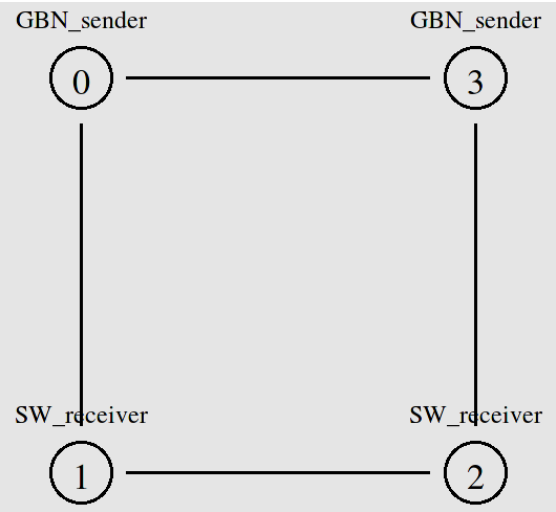
Στις προηγούμενες ασκήσεις μάθαμε πώς να δημιουργούμε ένα animation της προσομοίωσης, ώστε να επαληθεύσουμε την ορθότητα του σεναρίου που δημιουργήσαμε, καθώς επίσης να δημιουργούμε αρχεία για γραφική απεικόνιση με το Xgraph. Σε αυτή την άσκηση θα μάθουμε επιπλέον να χρησιμοποιούμε τα

αρχεία ίχνους (trace file). Όπως θα δούμε και στη συνέχεια, στο αρχείο ίχνους αποθηκεύονται όλα τα γεγονότα που λαμβάνουν χώρα κατά τη διάρκεια της προσομοίωσης.

```
# Set the nam trace file
set nf [open lab4.nam w]
$ns namtrace-all $nf

# Set the trace file
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
}
```

### *3.2 Τοπολογία*

Η προσομοίωση βασίζεται σε τοπολογία δακτυλίου (ring) στην οποία 4 κόμβοι συνδέονται μεταξύ τους ανά δύο με πλήρως αμφίδρομες ζεύξεις, όπως φαίνεται στο παρακάτω σχήμα.




*Σχήμα 1 - Αναπαράσταση της τοπολογίας δακτυλίου των τεσσάρων κόμβων στο NAM*

Εισάγετε τις παρακάτω γραμμές κώδικα, ώστε να δημιουργήσετε την εν λόγω τοπολογία:

```
# Create the nodes
for {set i 0} {$i < 4} {incr i} {
        set n($i) [$ns node]
}

$ns at 0.0 "$n(0) label GBN_sender"
$ns at 0.0 "$n(1) label SW_sender"
$ns at 0.0 "$n(3) label GBN_receiver"
$ns at 0.0 "$n(2) label SW_receiver"

# Create a duplex link between the nodes
for {set i 0} {$i < 4} {incr i} {
        $ns duplex-link $n($i) $n([expr ($i+1)%4]) 2Mb 50ms DropTail
}

$ns duplex-link-op $n(0) $n(3) orient right
$ns duplex-link-op $n(1) $n(2) orient right
$ns duplex-link-op $n(0) $n(1) orient down
$ns duplex-link-op $n(3) $n(2) orient down
```

Ας σημειωθεί ότι η εντολή `$ns at 0.0 "$n(i) label TEXT"` εμφανίζει τη χρονική στιγμή 0.0 ετικέτα (label) με περιεχόμενο TEXT η οποία προσαρτάται στον κόμβο `n(i)`.

### *3.3 Το στρώμα μεταφοράς*

Τα πρωτόκολλα κυλιόμενου παραθύρου εμπεριέχονται στο πρωτόκολλο μεταφοράς TCP. Για το λόγο αυτό θα δημιουργήσουμε δύο συνδέσεις TCP: η πρώτη από τον κόμβο n0 στον κόμβο n3 και η δεύτερη από τον n1 στον n2. Αυτό γίνεται με τις ακόλουθες εντολές:

```
# Define color index
$ns color 0 red
$ns color 1 green

# Setup go-back-n sender-receiver
set tcp0 [new Agent/TCP/Reno]
$tcp0 set window_ 7
# Disable modelling the initial SYN/SYNACK exchange
$tcp0 set syn_ false
```

```
# The initial size of the congestion window on slow-start
$tcp0 set windowInit_ 7
# Set flow ID
$tcp0 set fid_ 0
$ns attach-agent $n(0) $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $n(3) $sink0

$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
# Setup stop-and-wait sender-receiver
set tcp1 [new Agent/TCP/Reno]
$tcp1 set window_ 1
# Disable modelling the initial SYN/SYNACK exchange
$tcp1 set syn_ false
# The initial size of the congestion window on slow-start
$tcp1 set windowInit_ 1
# Set flow ID
$tcp1 set fid_ 1
$ns attach-agent $n(1) $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $n(2) $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
```

Το μέγεθος του παραθύρου του Go back N καθορίζεται με την εντολή:

```
$tcp0 set window_ 7
```

Αντίστοιχα, στην περίπτωση του Stop and Wait το μέγεθος παραθύρου ορίζεται ίσο με 1:

```
$tcp1 set window_ 1
```

Στις παραπάνω εντολές, έχουμε ορίσει ως εφαρμογή που θα δημιουργήσει τις ροές κίνησης, το πρωτόκολλο FTP.

### *3.4 Εκτέλεση του σεναρίου*

Τέλος, ορίζουμε τα γεγονότα της προσομοίωσης:

```
# Events
$ns at 0.5 "$ftp0 produce 50"
$ns at 0.5 "$ftp1 produce 50"
$ns at 10.0 "finish"

# Run the simulation
$ns run
```

Με βάση τα παραπάνω, τη χρονική στιγμή t = 0,5 sec, οι δύο FTP agents αρχίζουν *ταυτόχρονα* την αποστολή 50 πακέτων, ενώ όταν t = 10 sec, η προσομοίωση 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.

## *4. Ανάλυση αποτελεσμάτων με τη βοήθεια του NAM*

- Με τη βοήθεια του NAM, εντοπίστε τη χρονική στιγμή που ολοκληρώνεται η μετάδοσης των 50 πακέτων FTP στην περίπτωση του πρωτοκόλλου: (i) Go back N και (ii) Stop and Wait.
- Ποιο είναι το ελάχιστο μέγεθος παραθύρου εκπομπής που εξασφαλίζει ελάχιστο χρόνο μετάδοσης του συνόλου των πακέτων στο πρωτόκολλο Go back N;
- Με βάση το ελάχιστο αυτό μέγεθος παραθύρου που προσδιορίσατε στο προηγούμενο ερώτημα, τροποποιήστε τις εντολές

  ```
  $tcp0 set window_ X
  $tcp0 set windowInit_ X
  ```

  εκτελέστε την προσομοίωση και υπολογίστε τη χρονική στιγμή που ολοκληρώνεται η μετάδοσης των 50 πακέτων FTP για το πρωτόκολλο Go back N με τη βοήθεια του NAM.
- Εκτελέστε πάλι την προσομοίωση με το μέγεθος παραθύρου του πρωτοκόλλου Go back N που βρήκατε στο δεύτερο ερώτημα όταν η ζεύξη n(0)-n(3) έχει δεκαπλάσια καθυστέρηση διάδοσης. Εντοπίστε τη χρονική στιγμή που ολοκληρώνεται η μετάδοση των 50 πακέτων FTP στον κόμβο n3 στην περίπτωση αυτή.

# 5. Ανάλυση αποτελεσμάτων με τη βοήθεια του αρχείου ίχνους (trace file)

## 5.1 Μορφή αρχείου ίχνους (trace file)

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

```
+ 0.5 1 2 tcp 40 ------- 1 1.0 2.0 0 1
- 0.5 1 2 tcp 40 ------- 1 1.0 2.0 0 1
r 0.55064 0 3 tcp 40 ------- 0 0.0 3.0 0 0
+ 0.55064 3 0 ack 40 ------- 0 3.0 0.0 0 2
- 0.55064 3 0 ack 40 ------- 0 3.0 0.0 0 2
r 0.55064 1 2 tcp 40 ------- 1 1.0 2.0 0 1
+ 0.55064 2 1 ack 40 ------- 1 2.0 1.0 0 3
```

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 flow ID της ροής στην οποία ανήκει το πακέτο. Η τιμή του πεδίου αυτού καθορίζεται από εντολές της μορφής `$tcpX set fid_ Y.` `To 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.

## 5.2 Ανάλυση με τη γλώσσα awk

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/ {
      flow_id = $8;
      if (flow_id == 0) {
            data_0 += $6;
```

```
            packets_0++;
      }
      if (flow_id == 1) {
            data_1 += $6;
            packets_1++;
      }
}
```

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 ανάλογα με την τιμή του `flow_id` (όγδοο πεδίο της γραμμής την οποία εξετάζουμε):

- αν το `flow_id` είναι ίσο με 0 τότε η μεταβλητή packets_0 αυξάνεται κατά 1, και η τιμή της μεταβλητής data_0 αυξάνεται κατά την τιμή που βρίσκεται στο έκτο πεδίο ($6).
- αν το `flow_id` είναι ίσο με 1 τότε η μεταβλητή packets_1 αυξάνεται κατά 1, και η τιμή της μεταβλητής data_1 αυξάνεται κατά την τιμή που βρίσκεται στο έκτο πεδίο ($6).

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.

### *5.3 Υπολογισμός του πλήθους των πακέτων και της ποσότητας των δεδομένων που ελήφθησαν*

Για να μελετήσουμε τα δύο πρωτόκολλα 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=0;
        packets_0=0;
        data_1=0;
        packets_1=0;
}
/^r/&&/tcp/ {
        flow_id = $8;
        if (flow_id == 0) {
                data_0 += $6;
                packets_0++;
        }
        if (flow_id == 1) {
```

```
                data_1 += $6;
                packets_1++;
        }
}
END {
        printf("Total Data received for flow ID 0\t: %d Bytes\n", data_0);
        printf("Total Packets received for flow ID 0\t: %d\n", packets_0);
        printf("Total Data received for flow ID 1\t: %d Bytes\n", data_1);
        printf("Total Packets received for flow ID 1\t: %d\n", packets_1);
}
```

### *5.4 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 `trace.awk` και τα δεδομένα βρίσκονται στο αρχείο `lab4.tr`, τότε εκτελούμε την εντολή:

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

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

### *5.5 Ερωτήσεις*

- Ποιος είναι ο αριθμός των πακέτων που παρελήφθησαν, πόσα δεδομένα παρελήφθησαν από τον παραλήπτη κατά τη διάρκεια της προσομοίωσης για κάθε ροή κίνησης;
- Εξετάζοντας το αρχείο ίχνους, προσδιορίστε σε πόσο χρόνο απεστάλησαν αυτά τα δεδομένα. Ποιος ο ρυθμός μετάδοσης δεδομένων και ποια είναι η χρησιμοποίηση του καναλιού;