

# ΔΙΚΤΥΑ ΕΠΙΚΟΙΝΩΝΙΩΝ

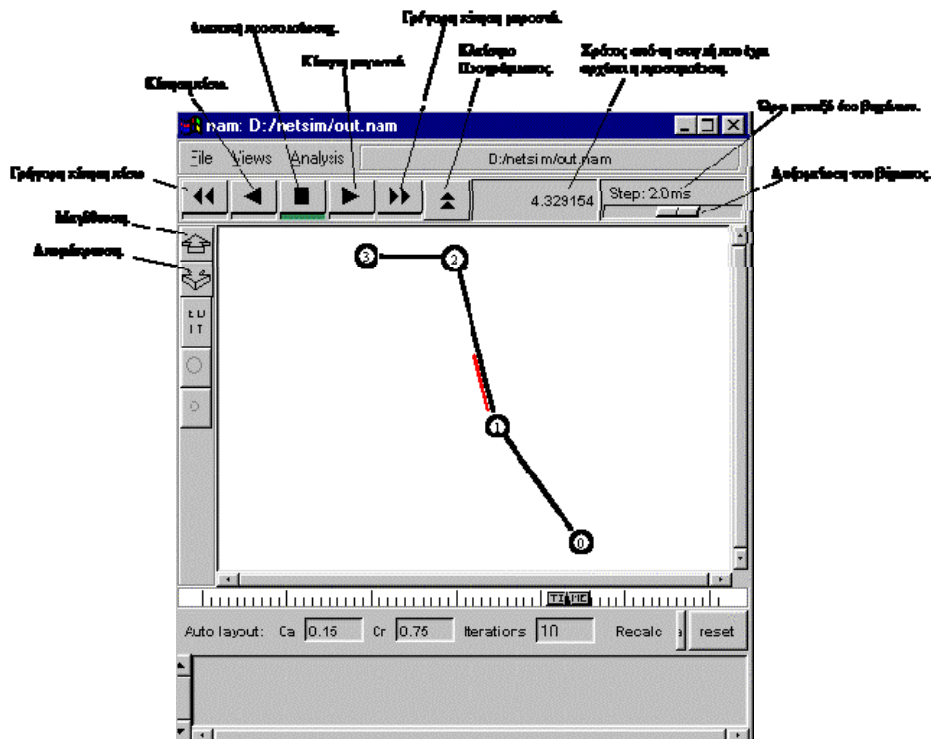
## Εργαστηριακή Άσκηση 1

### 1. Εισαγωγή στο Network Simulator v2 (NS2)

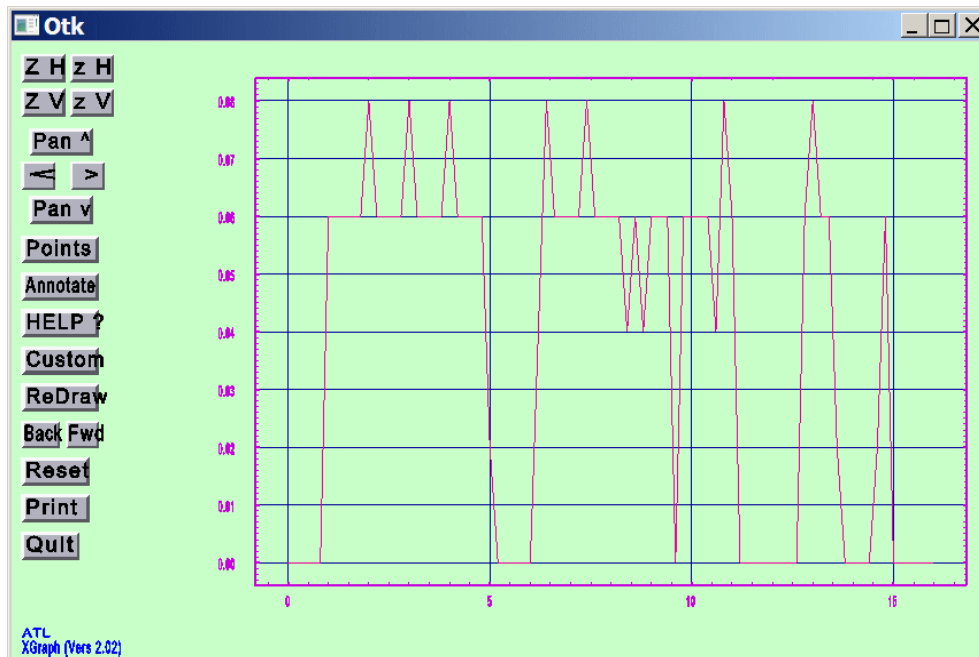
Το NS2 (Network Simulator version 2) είναι ένα πρόγραμμα για προσομοιώσεις δικτύων που διατίθεται δωρεάν. Ο δικτυακός τόπος της ομάδας εξέλιξης του NS είναι <http://www.isi.edu/nsnam/ns/>. Η ιστοσελίδα αυτή περιέχει βοήθεια σχετικά με το NS καθώς και τον πηγαίο κώδικά του. Οι οδηγίες χρήσεως του NS βρίσκονται στο φάκελο που βρίσκεται το εκτελέσιμο αρχείο του NS και είναι ένα αρχείο *pdf* με όνομα “ns\_doc.pdf”.

Το εκτελέσιμο αρχείο του NS είναι το “ns.exe”. Άλλα βοηθητικά προγράμματα που θα χρησιμοποιηθούν είναι το Network Animator (nam.exe) και το Xgraph (xgraph.exe). Το πρώτο χρησιμοποιείται για την γραφική απεικόνιση μιας προσομοίωσης ενώ το δεύτερο για τη δημιουργία γραφικών παραστάσεων.

Η διάταξη του δικτύου που θέλουμε να προσομοιώσουμε καθώς και όλα τα υπόλοιπα πρέπει να οριστούν σε ένα αρχείο κειμένου. Η γλώσσα που χρησιμοποιείται είναι η “Tcl”. Μπορείτε να αρχίσετε το NS από ένα παράθυρο εντολών (command prompt) και να τρέξετε τα αρχεία με την εντολή “ns αρχείο\_Tcl” υποθέτοντας ότι είσαστε στον κατάλογο (directory) που βρίσκεται το αρχείο και ότι το “αρχείο\_Tcl” είναι το όνομα του κειμένου σε γλώσσα *Tcl* που περιγράφει την προσομοίωση. Τα αποτελέσματα της προσομοίωσης θα είναι είτε αρχεία που προορίζονται για χρήση στο NAM και το όνομά τους θα έχει τη μορφή *όνομα\_αρχείου.nam* είτε αρχεία που προορίζονται για χρήση στο Xgraph (*όνομα\_αρχείου.tr*). Μπορεί επίσης να είναι κείμενο που θα εμφανίζεται στο “command prompt” και θα δίνει τα αποτελέσματα των ζητούμενων μετρήσεων. Η εικόνα του Σχ. 1 δείχνει τις βασικές λειτουργίες του *Network Animator*, ενώ η εικόνα του Σχ. 2 δείχνει μια γραφική παράσταση χρησιμοποιώντας το *Xgraph*. Μπορείτε να σώσετε τις γραφικές παραστάσεις του *Xgraph* πατώντας “Print Screen”, ανοίγοντας το “Word” και κάνοντας *Paste* την εικόνα.



Σχήμα 1 – Βασικές λειτουργίες του Network Animator (NAM)



Σχήμα 2 – Γραφική παράσταση με τη χρήση του Xgraph

## 2. Το πρώτο Tcl script – Γνωριμία με το NAM

Στην άσκηση αυτή θα γράψετε βήμα προς βήμα ένα *Tcl script* για NS, το οποίο προσομοιώνει μια απλή τοπολογία δικτύου. Θα μάθετε πώς εγκαθίστανται κόμβοι και ζεύξεις, πώς να στέλνετε δεδομένα από τον ένα κόμβο στον άλλον, πώς να παρακολουθείτε μια ουρά αναμονής και πώς να χρησιμοποιήσετε το NAM (Network Animator) για να αναπαραστήσετε γραφικά την προσομοίωση που κάνετε.

### 2.1 Πώς να αρχίσετε

Μπορείτε να γράψετε τα *Tcl scripts* με οποιονδήποτε επεξεργαστή κειμένου, όπως το *Notepad* ή το *Word*. Για τη δημιουργία του αρχείου *Tcl* θα πρέπει να δημιουργήσετε πρώτα ένα αρχείο *Notepad* με κατάληξη *“.tcl”*, όπου θα γράψετε το *script* σε γλώσσα *Tcl*, όπως περιγράφεται στη συνέχεια. Έπειτα για να το σώσετε θα πρέπει να κάνετε *File* → *Save* (όχι *Save As*). Δηλαδή, αν θέλετε να ονομάσετε το αρχείο *“lab1.tcl”*, πληκτρολογήστε στο *“Command Prompt”* *“notepad lab1.tcl”* και πατήστε *“Enter”*. Δεν θα μπορείτε να το μετονομάσετε (*Rename*) αργότερα.

Ξεκινήστε γράφοντας ένα βασικό *“template”*, το οποίο μπορείτε να χρησιμοποιείτε για όλα τα αρχικά *Tcl scripts*. Αρχικά, πρέπει να δημιουργήσετε ένα αντικείμενο (object) προσομοίωσης, το οποίο πραγματοποιείται με την εντολή:

```
set ns [new Simulator]
```

Στη συνέχεια χρειάζεται να δώσετε εντολή να ανοιχτεί ένα αρχείο από το πρόγραμμα για να γραφτούν τα δεδομένα που πρόκειται να χρησιμοποιηθούν για το NAM, δηλαδή τα *“trace data”*:

```
set nf [open lab1.nam w]
```

```
$ns namtrace-all $nf
```

Η πρώτη γραμμή ανοίγει ένα αρχείο με όνομα *“lab1.nam”* για εγγραφή και του προσδίδει το *file handle* *“nf”*. Στη δεύτερη γραμμή λέμε στο αντικείμενο της προσομοίωσης που δημιουργήθηκε προηγουμένως να γράφει όλα τα δεδομένα που έχουν σχέση με το NAM σ’ αυτό το αρχείο. Το επόμενο βήμα έχει ως στόχο να προσθέσει μια διαδικασία *“finish”* η οποία κλείνει το αρχείο δεδομένων *trace*.

```

proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exit 0
}

```




Ο ανωτέρω κώδικας θα σας είναι πιο σαφής αργότερα, αφού τον τρέξετε και δείτε τι κάνει. Η επόμενη γραμμή λέει στο αντικείμενο προσομοίωσης να εκτελέσει τη διαδικασία “finish” μετά από χρόνο προσομοίωσης 8.0 sec. Το NS σας παρέχει έναν πολύ απλό τρόπο να θέτετε τη σειρά των γεγονότων με την εντολή “at”. Τέλος, η τελευταία γραμμή κώδικα αρχίζει την προσομοίωση.

```

$ns at 8.0 "finish"
$ns run

```

Αφού σώσετε το αρχείο, όπως περιγράφηκε παραπάνω, μπορείτε να το τρέξετε γράφοντας “ns lab1.tcl” σε ένα παράθυρο εντολών. Αφού πραγματοποιηθεί η προσομοίωση, θα πρέπει να δημιουργηθεί ένα αρχείο “*.nam*”, στην περίπτωση μας έχει όνομα “*lab1.nam*”, που θα έχει αποθηκευμένα τα δεδομένα της προσομοίωσης που είναι απαραίτητα για την γραφική απεικόνιση της. Αυτή μπορείτε να τη δείτε πληκτρολογώντας την εντολή “nam lab1.nam” στο παράθυρο εντολών. Θα πρέπει να ανοίξει ο *Network Animator* (NAM), αλλά δεν θα δείτε τίποτα παρά μια κενή εικόνα, διότι μέχρι τώρα δεν έχετε ορίζει *objects* (nodes, links, κλπ.) ή *events*, δηλαδή δεν υπάρχουν δεδομένα για να προσομοιωθούν. Θα ορίσουμε τα *objects* στην Ενότητα 2.2 και τα *events* στην Ενότητα 2.3. Θα πρέπει να χρησιμοποιήσετε τον κώδικα από αυτή την ενότητα ως αρχικό σημείο στις άλλες ενότητες παρακάτω.

**Σημείωση:** Μην κλείνετε τα *notepads*. Πριν προσομοιώσετε το αρχείο \*.tcl πρέπει κάθε φορά να το κάνετε *Save*. Για να εμφανιστεί ο κέρσορας 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*.

## 2.2 Δύο κόμβοι και μία ζεύξη

Στην ενότητα αυτή, πρόκειται να ορίσετε μια πολύ απλή τοπολογία με δύο κόμβους που συνδέονται με μία ζεύξη. Οι παρακάτω δύο γραμμές ορίζουν τους δύο κόμβους. (Σημείωση: Πρέπει να εισάγετε τον παρακάτω κώδικα πριν από τη γραμμή “\$ns at 8.0 “finish””).

```

set n0 [$ns node]
set n1 [$ns node]

```


Ένα αντικείμενο νέου κόμβου δημιουργείται με την εντολή “\$ns node”. Ο ανωτέρω κώδικας δημιουργεί δύο κόμβους και τους αντιστοιχεί στα *handles* “n0” and “n1”. Η επόμενη γραμμή κώδικα ορίζει τη ζεύξη που συνδέει τους δύο κόμβους.

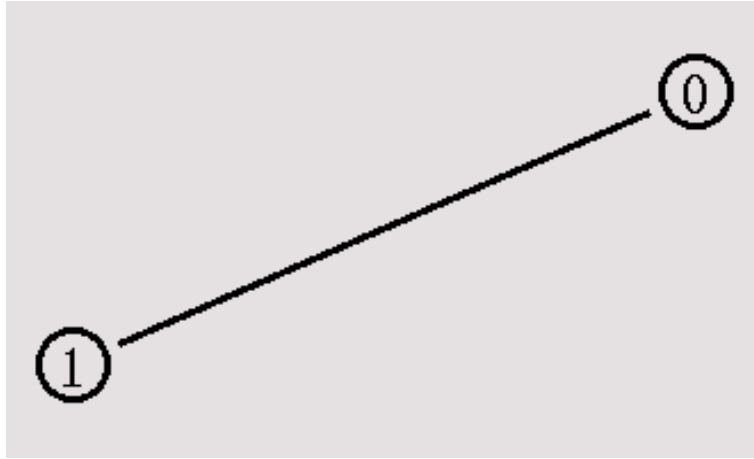
```

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

```

Η γραμμή αυτή λέει στο αντικείμενο της προσομοίωσης να συνδέσει τους κόμβους “n0” και “n1” με μια αμφίδρομη ζεύξη εύρους ζώνης 2Mbps, καθυστέρησης 10ms και της οποίας η ουρά αναμονής είναι *DropTail*. Τώρα μπορεί να σώσετε το αρχείο σας, *File* → *Save*, και να τρέξετε το *script* πληκτρολογώντας “ns lab1.tcl”. Αφού πραγματοποιηθεί η προσομοίωση, θα δημιουργηθεί ένα αρχείο με όνομα “*lab1.nam*” στον φάκελο που βρίσκεστε. Πληκτρολογήστε την εντολή “nam lab1.nam” στο παράθυρο εντολών και το NAM θα αρχίσει αυτόματα, οπότε και θα πρέπει να δείτε ως έξοδο την εικόνα του Σχ. 3.

**Σημείωση:** Μην κλείνετε τα *notepads*. Πριν προσομοιώσετε το αρχείο \*.tcl πρέπει κάθε φορά να το κάνετε *Save*. Για να εμφανιστεί ο κέρσορας 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 φυλλαδίου υπάρχει για συμβουλευτικό σκοπό, π.χ. σε περίπτωση λάθους κτλ, μια έκδοση του κώδικα αυτής της Άσκησης.



Σχήμα 3 – Αναπαράσταση δύο κόμβων και μιας ζεύξης στο NAM

## 2.3 Αποστολή δεδομένων

Το παράδειγμα βέβαια δεν είναι πολύ ικανοποιητικό ακόμα, αφού μπορείτε να δείτε μόνο την τοπολογία χωρίς να συμβαίνει τίποτε άλλο. Έτσι, το επόμενο βήμα είναι να σταλούν δεδομένα από τον κόμβο “n0” στον κόμβο “n1”. Στο NS, τα δεδομένα στέλνονται πάντα από έναν “agent” σε έναν άλλον. Έτσι, το επόμενο βήμα έχει ως στόχο να δημιουργήσει ένα “agent object” που στέλνει δεδομένα από τον κόμβο “n0”, και ένα άλλο “agent object” που λαμβάνει τα δεδομένα στον κόμβο “n1”. Ο παρακάτω κώδικας πρέπει να εισαχθεί μετά τις εντολές που βάλατε στην προηγούμενη ενότητα (Ενότητα 2.2).

```
#Δημιουργία ενός UDP agent και «προσάρτησή» του στον κόμβο «n0»
set udp0 [new Agent/UDP]
$udp0 set packetSize_ 1000
$ns attach-agent $n0 $udp0

#Δημιουργία μιας πηγής κίνησης CBR και «τοποθέτησή» της στο «udp0»
set traffic0 [new Application/Traffic/CBR]
#Προσδιορισμός της κίνησης για τον κόμβο n0
$traffic0 set packetSize_ 1000
$traffic0 set interval_ 0.005
$traffic0 attach-agent $udp0
```

Οι γραμμές αυτές δημιουργούν έναν “UDP agent” και τον προσαρτούν στον κόμβο “n0”. Στη συνέχεια προσαρτούν μια γεννήτρια κίνησης σταθερού ρυθμού *bit* (CBR: constant bit rate) στον “UDP agent”. Το μέγεθος πακέτου, “packetSize”, τίθεται ίσο προς *1000 bytes* και θα στέλνεται ένα πακέτο κάθε *0.005 sec*. Μπορείτε να βρείτε σχετικές παραμέτρους για κάθε τύπο “agent” στο εγχειρίδιο του NS.

Η επόμενη γραμμή κώδικα δημιουργεί έναν “Sink Agent” που δρα ως υποδοχέας της κίνησης και τον προσαρτά στον κόμβο “n1”.

```
set sink [new Agent/LossMonitor]
```

```
$ns attach-agent $n1 $sink
```

Οι δύο Agents πρέπει τώρα να συνδεθούν μεταξύ τους.

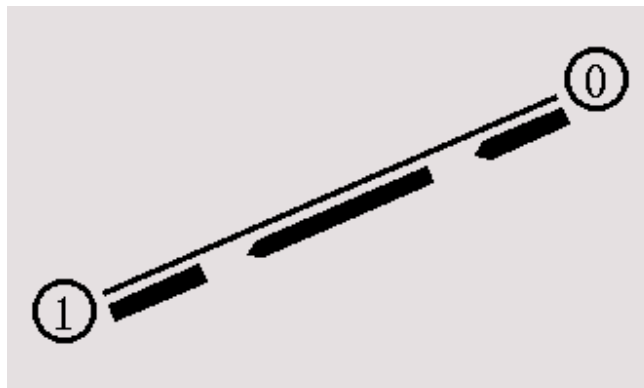
```
$ns connect $udp0 $sink
```

Στη συνέχεια θα πρέπει να πούμε στον “CBR agent” πότε να στείλει τα δεδομένα και πότε να σταματήσει. Οι παρακάτω γραμμές κώδικα πρέπει να τεθούν μόλις πριν την γραμμή “\$ns at 8.0 “finish””.

```
$ns at 1.0 "$traffic0 start"
```

```
$ns at 7.0 "$traffic0 stop"
```

Η εξήγηση των παραπάνω γραμμών είναι προφανής. Μπορείτε τώρα να σώσετε το αρχείο και να αρχίσετε πάλι την προσομοίωση. Όταν κάνετε κλικ στο κουμπί “play” του παραθύρου NAM, θα δείτε ότι μετά 1.0 sec προσομοίωσης, ο κόμβος 0 αρχίζει να στέλνει πακέτα δεδομένων στον κόμβο 1, όπως φαίνεται στην εικόνα του Σχ. 4. Αν θέλετε να κάνετε πιο αργό το NAM, χρησιμοποιείστε το “Step”, δηλαδή την μπάρα πάνω δεξιά. Προτείνουμε, στη συνέχεια, πειραματιστείτε με το NAM και το *Tcl script*. Μπορείτε να κάνετε κλικ σε οποιοδήποτε πακέτο στο παράθυρο NAM για να το παρακολουθήσετε, όπως επίσης να κάνετε κλικ απευθείας στη ζεύξη για να λάβετε μερικά στατιστικά.



Σχήμα 4 – Αναπαράσταση αποστολής κίνησης μεταξύ δύο κόμβων στο NAM

## 2.4 Ερωτήσεις

- Ποιος είναι ο ρυθμός μετάδοσης σε *bit/sec*;
- Ποιος είναι ο συνολικός αριθμός *bytes* και *bits* που μεταφέρθηκαν από την αρχή ως το τέλος της προσομοίωσης;
- Πόσα *bytes* υπάρχουν πάνω στη γραμμή ζεύξης κάθε στιγμή; Επιβεβαιώστε την απάντησή σας από το *animation*.
- Ποια είναι η απάντησή σας στο προηγούμενο ερώτημα, αν διπλασιαστεί η καθυστέρηση της ζεύξης; Επιβεβαιώστε την απάντησή σας από το *animation*.
- Εάν υποθέσουμε ότι σε κάθε πακέτο οι επικεφαλίδες του *IP* και του *UDP* μαζί έχουν μήκος 40 *byte*, ποιος είναι ο καθαρός ρυθμός μετάδοσης των δεδομένων σε *bit/sec*;
- Ποιες παράμετροι μπορεί να αλλαχθούν για να μεταβληθεί ο ρυθμός μετάδοσης και με ποιες εντολές επιτυγχάνονται αυτές οι αλλαγές;
- Αν επιθυμούμε να έχουμε καθαρό ρυθμό μετάδοσης δεδομένων ίσο με 1.2 *Mbit/sec*, μεταβάλλοντας κάθε φορά μία από τις ανωτέρω παραμέτρους, ποιες τιμές προτείνετε για κάθε μία; Ελέγξτε κάθε φορά αν οι απαντήσεις σας δίνουν ρυθμό μετάδοσης μικρότερο από τη χωρητικότητα της ζεύξης.
- Για ποιες τιμές των ανωτέρω παραμέτρων θα αρχίσει να παρατηρείται οριακά η απώλεια πακέτων; Επιβεβαιώστε την απάντησή σας τρέχοντας το *tcl script* και το *animation*.

**Σημείωση:** Κλείσετε το NAM, μόνο πατώντας το “X” δεξιά επάνω στην μπάρα του παραθύρου

### 3. Μελέτη απλής τοπολογίας με το Xgraph

Στην άσκηση αυτή θα χρησιμοποιήσετε το “Xgraph” για να δημιουργήσετε γραφικές παραστάσεις της κίνησης στο δίκτυο που ήδη μελετάτε. Οι παρακάτω εντολές θα πρέπει να προστεθούν στην αρχή του *script*, κάτω από την εντολή “\$ns namtrace-all \$nf”. Θα πρέπει κατ’ αρχήν να δημιουργηθεί ένα αρχείο όπου θα καταχωρηθούν όλα τα δεδομένα της προσομοίωσης σχετικά με το “Xgraph”:

```
set f [open lab1.tr w]
```

Επίσης χρειαζόμαστε μια διαδικασία (procedure) καταγραφής των δεδομένων της κίνησης:

```
proc record {} {  
    global sink f  
    set ns [Simulator instance]  
    #Ορισμός της ώρας που η διαδικασία θα ξανακληθεί.  
    set time 0.15  
    #Καταγραφή των bytes.  
    set bw [$sink set bytes_  
    #Λήψη της τρέχουσας ώρας  
    set now [$ns now]  
    #Υπολογισμός του bandwidth και καταγραφή αυτού στο αρχείο  
    puts $f "$now [expr (($bw/$time)*8)/1000000]"  
    #Κάνει την τιμή bytes_ 0  
    $sink set bytes_ 0  
    #Επαναπρογραμματισμός της διαδικασίας  
    $ns at [expr $now+$time] "record"  
}
```

Επιπλέον, τροποποιήστε τις εντολές διαδικασίας κλεισίματος των αρχείων, ώστε να συμπεριλάβετε το κλείσιμο του ανοιχτού αρχείου για το Xgraph ως εξής:

```
proc finish {} {  
    global ns nf f  
    $ns flush-trace  
    close $nf  
    close $f  
    exit 0  
}
```

Τέλος, η επόμενη γραμμή κώδικα καλεί την διαδικασία καταγραφής της κίνησης και πρέπει να είναι η πρώτη από τις εντολές καθορισμού των γεγονότων (events).

```
$ns at 0.0 "record"
```

Αφού γράψετε τα παραπάνω, μπορείτε να τα σώσετε είτε στο ίδιο αρχείο είτε να ανοίξετε άλλο αρχείο με όνομα π.χ. “lab1b.tcl”, με τη διαδικασία που περιγράφηκε στην *Ενότητα 2.1*, και να σώσετε το script εκεί. Έπειτα για να το προσομοιώσετε, ανοίξετε ένα “Command Prompt” και πληκτρολογήστε “ns lab1b.tcl” ή όποιο άλλο όνομα έχετε δώσει στο αρχείο. Για να δείτε τα αποτελέσματα της προσομοίωσης σε γραφική παράσταση, πληκτρολογήστε “xgraph lab1.tr”.

### 3.1 Ερωτήσεις

- Μεταβάλλοντας την τιμή του μήκους του πακέτου διαπιστώστε και σχολιάστε πώς μεταβάλλεται η γραφική παράσταση της μεταφερόμενης κίνησης.
- Ποιο είναι το μέγιστο μήκος πακέτου που μπορεί να αποσταλεί χωρίς να ξεπερνάται η χωρητικότητα της γραμμής;
- Διατηρώντας σταθερό το μήκος πακέτου, μεταβάλλετε τον ρυθμό μετάδοσης. Τι παρατηρείτε στη γραφική παράσταση της μεταφερόμενης κίνησης και πώς το ερμηνεύετε;
- Αυξήστε την καθυστέρηση της γραμμής σύνδεσης των δυο κόμβων σε  $0.5$  δευτερόλεπτα. Τι παρατηρείτε στην γραφική παράσταση της μεταφερόμενης κίνησης; Επαναφέρετε την καθυστέρηση στην αρχική τιμή.
- Πώς επηρεάζει τη γραφική παράσταση ο χρόνος που επαναλαμβάνεται η διαδικασία “record”. Προτείνετε έναν κατάλληλο χρόνο για να επιτύχετε μια γραφική παράσταση στιγμιαίας κίνησης και μια μέση.
- Η κίνηση μεταξύ των δυο κόμβων είναι σταθερής ροής (CBR). Αλλάξτε την κίνηση σε εκθετική θέτοντας “Exponential” όπου υπάρχει “CBR”. Εξηγήστε τη γραφική παράσταση της κίνησης. Για να βγάλετε σωστά συμπεράσματα, θα πρέπει να αυξήσετε τον χρόνο αποστολής της κίνησης, και φυσικά της προσομοίωσης, τουλάχιστον σε  $20$  δευτερόλεπτα και να ξανατρέξετε το *script*.

**Σημείωση:** Μπορείτε να σώσετε τις γραφικές παραστάσεις πατώντας “Print Screen”. Ύστερα ανοίξτε το “Word” και κάντε *Paste* την εικόνα. Για συμβουλευτικούς λόγους καθώς και για επαλήθευση και διόρθωση τυχόν λαθών, ο κώδικας υπάρχει ολοκληρωμένος στο τέλος της άσκησης. Στο επόμενο εργαστήριο θα πρέπει να παραδώσετε μια αναφορά με απαντήσεις και σχόλια σχετικά με τις ερωτήσεις αυτής της Άσκησης.

## Ολοκληρωμένος κώδικας για την Εργαστηριακή Άσκηση 1

```
set ns [new Simulator]

# αρχείο γραφικής αναπαράστασης προσομοίωσης (NAM)
set nf [open lab1.nam w]
$ns namtrace-all $nf

# αρχείο καταγραφής κίνησης
set xf [open lab1.tr w]

# διαδικασία καταγραφής
proc record {} {
    global sink xf
    set ns [Simulator instance]
    set time 0.15
    set bw [$sink set bytes_]
    set now [$ns now]
    puts $xf "$now [expr ((($bw/$time)*8)/1000000)] "
    $sink set bytes_ 0
    $ns at [expr $now+$time] "record"
}

# διαδικασία τερματισμού
proc finish {} {
    global ns nf xf
    $ns flush-trace
    close $nf
    close $xf
    exit 0
}

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

set agent0 [new Agent/UDP]
$agent0 set packetSize_ 1000
$ns attach-agent $n0 $agent0

set traffic0 [new Application/Traffic/CBR]
$traffic0 set packetSize_ 1000
$traffic0 set interval_ 0.005
$traffic0 attach-agent $agent0

set sink [new Agent/LossMonitor]
$ns attach-agent $n1 $sink
$ns connect $agent0 $sink
```



```
$ns at 0.0 "record"  
$ns at 1.0 "$traffic0 start"  
$ns at 7.0 "$traffic0 stop"  
$ns at 8.0 "finish"
```

```
$ns run
```