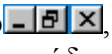
ΔΙΚΤΥΑ ΕΠΙΚΟΙΝΩΝΙΩΝ

Εργαστηριακή Άσκηση 3

1 Μετάδοση δεδομένων σε δίκτυο με σύνθετη τοπολογία

Στην άσκηση αυτή θα ασχοληθείτε με τη μετάδοση δεδομένων μεταξύ κόμβων που συνδέονται σε δίκτυο με σχετικά σύνθετη τοπολογία. Προς τούτο θα ορίσετε στο NS2 ένα δίκτυο που αποτελείται από εννέα κόμβους και θα επιλέξετε δύο κόμβους του δικτύου, που δεν συνδέονται άμεσα μεταξύ τους, να ανταλλάσσουν δεδομένα. Σκοπός είναι να δείτε πώς επηρεάζεται η δρομολόγηση της ροής των δεδομένων μεταξύ των κόμβων αποστολής και λήψης από τον αριθμό των παρεμβαλλόμενων κόμβων και ζεύξεων, καθώς και από το κόστος των ζεύξεων.

Για να αρχίσετε, θα πρέπει να δημιουργήσετε ένα αρχείο, π.χ. "exercise3.tcl", με τον τρόπο πού περιγράφεται στην *Εργαστηριακή Άσκηση 1,* χρησιμοποιώντας τον κώδικα της *Ενότητας 2.1* εκείνης της άσκησης ως υπόδειγμα. Όπως αναφέρθηκε προηγουμένως, 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 τρέχει το NAM και το *Xgraph*, θα πρέπει να ανοίγετε πάντα αρχεία *trace* ".tr" ή ".nam", να τα αρχικοποιείτε και να ορίζετε μια διαδικασία που να τα κλείνει.

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 προσομοιώσετε ένα αρχείο "**.tcl*" πρέπει κάθε φορά να το σώζετε (*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. Ο ολοκληρωμένος κώδικας στο τέλος της Ενότητας παρατίθεται για συμβουλευτικό σκοπό και μόνο, π.χ. σε περίπτωση λάθους κτλ.

1.1 Τοπολογία

Όπως πάντα, το πρώτο βήμα είναι ο ορισμός της τοπολογίας. Εισάγετε τις παρακάτω γραμμές στο αρχικό *template* του κώδικα της *Εργαστηριακής Άσκησης 1*, παράγραφος *2.1*, ώστε να δημιουργήσετε *9* κόμβους. Με τον τρόπο αυτό μπορείτε να δημιουργήσετε έναν μεγάλο αριθμό κόμβων σε ένα *Tcl array* αντί να δίνετε σε κάθε κόμβο το δικό του όνομα..

```
for  {set i 0} {$i < 9} {incr i} {
     set n($i) [$ns node]
}
```

Το παρακάτω τμήμα κώδικα *Tcl* δημιουργεί αμφίδρομες ζεύξεις μεταξύ των κόμβων.

```
for  {set i 0 } {$i < 9} {incr i} {
    $ns duplex-link $n($i) $n([expr ($i+1)%5]) 1Mb 10ms DropTail
}

$ns duplex-link $n(3) $n(0) 1Mb 10ms DropTail
$ns duplex-link $n(5) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 1Mb 10ms DropTail
$ns duplex-link $n(2) $n(7) 1Mb 10ms DropTail
```

Μπορείτε τώρα να σώσετε, με *File→ Save*, και να προσομοιώσετε το *script*. Μπορεί να φανεί στο NAM, ότι η τοπολογία είναι λίγο άκομψη. Μπορείτε να κάνετε κλικ στο κουμπί "Re-layout", ίσως και στο "Reset", μερικές φορές για να φανεί καλύτερα.

1.2 Τα γεγονότα (events)

Δημιουργήστε τώρα δύο *UDP agents* με πηγές που παράγουν κίνηση *CBR* και προσαρτήστε τους στους κόμβους "n5" και "n7", γράφοντας τον παρακάτω κώδικα μετά από αυτόν της *§1.1*. Έπειτα δημιουργήστε δύο *Sink Agents* και προσαρτήστε τους στους ίδιους κόμβους αλλά αντίστροφα. Επίσης, είναι προτιμότερο οι δυο ροές να είναι χρωματιστές, κάτι που είδαμε στην προηγούμενη Εργαστηριακή Άσκηση.

```
#Στρώμα ζεύξης, κόμβος 5: πηγή και προορισμός
set udp5 [new Agent/UDP]
$ns attach-agent $n(5) $udp5
$udp5 set fid_ 5
$ns color 5 red
set sink5 [new Agent/LossMonitor]
$ns attach-agent $n(5) $sink5

#κόμβος 7: πηγή και προορισμός
set udp7 [new Agent/UDP]
$ns attach-agent $n(7) $udp7
$udp7 set fid_ 7
$ns color 7 blue
set sink7 [new Agent/LossMonitor]
$ns attach-agent $n(7) $sink7

#Σύνδεση των πηγών και των προορισμών
$ns connect $udp5 $sink7
$ns connect $udp7 $sink5

#Στρώμα εφαρμογής
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
set cbr7 [new Application/Traffic/CBR]
$cbr7 attach-agent $udp7
```

Είναι επιθυμητό, ο πρώτος *CBR agent* να αρχίσει να στέλνει στα *0.5 sec* και ο δεύτερος στα *0.8 sec*. Αντίστοιχα, ο πρώτος θα σταματήσει στα *2 sec*, ενώ ο δεύτερος στα *2.5 sec*. Η διαδικασία κλεισίματος καλείται στα *3 sec*.

```
#Ορισμός γεγονότων
$ns at 0.5 "$cbr5 start"
$ns at 1 "$cbr7 start"
$ns at 2 "$cbr5 stop"
$ns at 2.5 "$cbr7 stop"
$ns at 3 "finish"
```

Μπορείτε τώρα να αρχίσετε το script πληκτρολογώντας "`ns exercise3.tcl`" και ύστερα τρέξτε το *animation*.

1.3 Ερωτήσεις

- Ποια διαδρομή ακολουθούν τα πακέτα;
- Ελέγξτε αν η ροή των πακέτων και από τις δυο πλευρές ακολουθεί τη διαδρομή με τα λιγότερα βήματα.
- Υπάρχει συντομότερη διαδρομή από αυτήν που ακολουθούν, όσον αφορά τον αριθμό των βημάτων (ζεύξεων);

2 Στατική και δυναμική δρομολόγηση

Στην προηγούμενη παράγραφο είδατε ότι η κίνηση ακολουθεί τη συντομότερη διαδρομή (αυτήν με τον μικρότερο αριθμό βημάτων) από τον κόμβο "5" στον κόμβο "7" και αντίστροφα. Θα δούμε στη συνέχεια τη διαφορά μεταξύ στατικής και δυναμικής δρομολόγησης και πώς το δίκτυο αντιμετωπίζει τις μεταβολές της τοπολογίας του στην κάθε περίπτωση. Γι' αυτόν τον λόγο θα προσθέσουμε ένα ενδιαφέρον χαρακτηριστικό, που είδαμε και στην *Εργαστηριακή Άσκηση 2*. Κάντε τη ζεύξη μεταξύ των κόμβων "1" και "2" (που χρησιμοποιείται για τη μετάδοση) να διακοπεί για *0.4 sec*. Συμβουλευτείτε τον κώδικα στην τελευταία σελίδα για τη θέση που εισάγονται οι εντολές.

```
$ns rtmodel-at 1.2 down $n(1) $n(2)
$ns rtmodel-at 1.6 up $n(1) $n(2)
```

Μπορείτε τώρα να αρχίσετε το *script* πάλι και θα δείτε ότι μεταξύ *1.2* και *1.6 sec* η ζεύξη θα διακοπεί και όλα τα δεδομένα που στέλνονται από τους δυο κόμβους χάνονται. Παρ' όλ' αυτά, οι δυο κόμβοι συνεχίζουν να εκπέμπουν πακέτα.

Αλλάξτε τώρα τη δρομολόγηση σε δυναμική (*routing protocol distance vector*), έτσι ώστε να μπορούν οι κόμβοι να καθορίζουν αυτόματα τα θέματα δρομολόγησης και να αντιμετωπίζουν τέτοια προβλήματα. Συνεπώς προσθέστε την επόμενη γραμμή στην αρχή του *Tcl script*, μετά τη δημιουργία του *object* προσομοίωσης.

```
$ns rtproto DV
```

Αρχίστε πάλι την προσομοίωση και τρέξτε το NAM.

2.1 Ερωτήσεις

- Εξηγήστε γιατί, με τη στατική δρομολόγηση, οι κόμβοι εξακολουθούν να στέλνουν πακέτα και μετά τη διακοπή της ζεύξης.
- Τα πακέτα που χάθηκαν, θα ξαναμεταδοθούν από τους αντίστοιχους κόμβους, όταν επανέλθει η σύνδεση;
- Τι παρατηρείτε όατν γίνεται διακοπή ζεύξης και έχουμε δυναμική δρομολόγηση; Περιγράψτε με απλά λόγια τη διαδικασία που λαμβάνει χώρα στο *animation*.
- Ποιος από όλους τους κόμβους καθορίζει από ποια διαδρομή θα προωθηθούν κάθε φορά τα πακέτα;
- Για ποιο λόγο τα πακέτα ακολουθούν τις συγκεκριμένες διαδρομές αφότου πέσει η σύνδεση;
- Θα μπορούσαν να δρομολογηθούν από άλλους κόμβους;

3 Καθορισμός κόστους ζεύξης

Έως αυτό το σημείο θεωρούσαμε ότι η προτιμώμενη δρομολόγηση των πακέτων είναι από εκείνη τη διαδρομή απ' όπου η ροή θα περάσει από τους λιγότερους κόμβους. Αυτό ισχύει διότι στην περίπτωση της προσομοίωσής μας θεωρείται εξ' ορισμού ότι όλες οι ζεύξεις έχουν κόστος ίσο με μια (1) μονάδα. Στην πραγματικότητα όμως, η διαδρομή που ακολουθείται σε κάθε περίπτωση βασίζεται στο κόστος των ζεύξεων μεταξύ των κόμβων. Ας δούμε πώς επηρεάζει η αλλαγή του κόστους των ζεύξεων την δρομολόγηση των πακέτων. Για να γίνει αυτό πιο παραστατικό, θα θεωρήσουμε ότι το κόστος μιας ζεύξης είναι ανάλογο της καθυστέρησής της. Αυξήστε τις καθυστερήσεις των ζεύξεων στις παρακάτω τιμές, αλλάζοντας τις κατάλληλες εντολές ορισμού των ζεύξεων.

```
$ns duplex-link $n(3) $n(0) 1Mb 40ms DropTail
$ns duplex-link $n(5) $n(8) 1Mb 40ms DropTail
$ns duplex-link $n(6) $n(7) 1Mb 30ms DropTail
$ns duplex-link $n(2) $n(7) 1Mb 20ms DropTail
```

Εισάγετε τώρα στον κώδικα, κάτω από τον ορισμό της τοπολογίας και των ζεύξεων, εντολές τις μορφής:

```
$ns cost $n(x) $n(y) z
```

έτσι ώστε να αυξήσετε το κόστος των ζεύξεων ανάλογα με την καθυστέρησή τους. Όπου "x" και "y"είναι οι αριθμοί των κόμβων και "z" είναι η τιμή του κόστους της ζεύξης. Δηλαδή με την παραπάνω εντολή, το κόστος της ζεύξης από τον κόμβο "x" στον κόμβο "y" αυξάνεται σε "z" μονάδες κόστους, μόνο όμως προς τη μια φορά (x→y). Για να αυξηθεί το κόστος και προς την άλλη φορά της σύνδεσης (y→x) θα πρέπει να εισαχθεί η εντολή:

```
$ns cost $n(y) $n(x) z
```

Θυμίζουμε ότι για καθυστέρηση *10ms* το κόστος της ζεύξης είναι *1* μονάδα. Τρέξτε την προσομοίωση και περιγράψτε τι συμβαίνει χρησιμοποιώντας το NAM.

3.1 Ερωτήσεις

- Ποιες διαδρομές ακολουθούν τα πακέτα πριν, μετά και κατά τη διάρκεια της πτώσης της σύνδεσης;
- Για ποιον λόγο τα πακέτα ακολουθούν τις συγκεκριμένες διαδρομές;
- Θα μπορούσαν να δρομολογηθούν από άλλους κόμβους;

4 Παρακολούθηση εκθετικής κίνησης με το Xgraph

Στην ενότητα 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 f0 [open out0.tr w]
set f1 [open out1.tr w]
```

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

```
proc record {} {
      global sink5 sink7 f0 f1
      set ns [Simulator instance]
#Ορισμός της ώρας που η διαδικασία θα ξανακληθεί
      set time 0.1
#Καταγραφή των bytes
      set bw5 [$sink5 set bytes_]
      set bw7 [$sink7 set bytes_]
#Λήψη της τρέχουσας ώρας
      set now [$ns now]
#Υπολογισμός του bandwidth και καταγραφή αυτού στο αρχείο
      puts $f0 "$now [expr (($bw5/$time)*8)/1000000]"
      puts $f1 "$now [expr (($bw7/$time)*8)/1000000]"
#Κάνει την μεταβλητή bytes 0
      $sink5 set bytes_ 0
      $sink7 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 f0 f1
```

```
      $ns flush-trace
      close $nf
      close $f0
      close $f1
exit 0
}
```

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

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

Θα πρέπει τέλος να σβήσετε τις εντολές που προκαλούν την διακοπή της σύνδεσης στα *1.2sec* και την αποκατάσταση της στα *1.6sec*, ώστε να μην υπάρχουν προβλήματα στη ροή των δεδομένων. Αφού γράψετε τα παραπάνω, μπορείτε να δείτε ταυτόχρονα τις γραφικές παραστάσεις του ρυθμού μετάδοσης των δεδομένων που συνέλεξαν και οι δύο *sink agents* για τις δύο ροές πληκτρολογώντας την εντολή "`xgraph out0.tr out1.tr`".

Αλλάξτε την κίνηση του ενός αποστολέα από σταθερής ροής (CBR) σε εκθετική θέτοντας "Exponential" όπου υπάρχει "CBR". Για να βγάλετε σωστά συμπεράσματα, θα πρέπει να αυξήσετε τον χρόνο αποστολής της κίνησης, και φυσικά ανάλογα και της προσομοίωσης, τουλάχιστον σε 15 δευτερόλεπτα και να ξανατρέξετε το *script*.

4.1 Ερωτήσεις

- Εξηγείστε την γραφική παράσταση της κίνησης για CBR και Exponential.

Σημείωση: Μπορείτε να σώσετε τις γραφικές παραστάσεις πατώντας "Print Screen". Ύστερα ανοίξτε το "Word" και κάντε *Paste* την εικόνα.
Για επαλήθευση και διόρθωση τυχόν λαθών, ο κώδικας υπάρχει ολοκληρωμένος στις επόμενες σελίδες.

Ολοκληρωμένος κώδικας για την Εργαστηριακή Άσκηση 3

```
set ns [new Simulator]

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

set f0 [open out0.tr w]
set f1 [open out1.tr w]

$ns rtproto DV

proc finish {} {
        global ns nf f0 f1
        $ns flush-trace
        close $nf
        close $f0
        close $f1
exit 0
}

proc record {} {
      global sink5 sink7 f0 f1
      set ns [Simulator instance]
      set time 0.1
      set bw5 [$sink5 set bytes_]
      set bw7 [$sink7 set bytes_]
      set now [$ns now]
      puts $f0 "$now [expr (($bw5/$time)*8)/1000000]"
      puts $f1 "$now [expr (($bw7/$time)*8)/1000000]"
      $sink5 set bytes_ 0
      $sink7 set bytes_ 0
      $ns at [expr $now+$time] "record"
}

for  {set i 0} {$i < 9} {incr i} {
      set n($i) [$ns node]
}

for  {set i 0 } {$i < 9} {incr i} {
    $ns duplex-link $n($i) $n([expr ($i+1)%5]) 1Mb 10ms DropTail
}

$ns duplex-link $n(3) $n(0) 1Mb 10ms DropTail
$ns duplex-link $n(5) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(6) $n(7) 1Mb 10ms DropTail
$ns duplex-link $n(2) $n(7) 1Mb 10ms DropTail

set udp5 [new Agent/UDP]
$ns attach-agent $n(5) $udp5
$udp5 set fid_ 5
$ns color 5 red
set sink5 [new Agent/LossMonitor]
$ns attach-agent $n(5) $sink5
```

```
set udp7 [new Agent/UDP]
$ns attach-agent $n(7) $udp7
$udp7 set fid_ 7
$ns color 7 blue
set sink7 [new Agent/LossMonitor]
$ns attach-agent $n(7) $sink7

$ns connect $udp5 $sink7
$ns connect $udp7 $sink5

set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
set cbr7 [new Application/Traffic/CBR]
$cbr7 attach-agent $udp7

$ns at 0.0 "record"
$ns at 0.5 "$cbr5 start"
$ns at 1 "$cbr7 start"
$ns rtmodel-at 1.2 down $n(1) $n(2)
$ns rtmodel-at 1.6 up $n(1) $n(2)
$ns at 2 "$cbr5 stop"
$ns at 2.5 "$cbr7 stop"
$ns at 3 "finish"
$ns run
```