


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. 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. “lab3.tcl”, με τον τρόπο που 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 πατώντας **μόνο το “X”** δεξιά επάνω στην μπάρα του παραθύρου , όχι από το μενού **File**. Στο τέλος του εργαστηρίου κάντε “log off” πριν φύγετε.

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 πρώτων 7 κόμβων, ορίζοντας καθυστέρηση ζεύξης ίση με 40 ms.

```
for {set i 0} {$i < 7} {incr i} {  
    $ns duplex-link $n($i) $n([expr ($i+1)%7]) 2Mb 40ms DropTail  
}
```

Το παρακάτω τμήμα κώδικα *Tcl* δημιουργεί τις υπόλοιπες ζεύξεις της τοπολογίας.

```
$ns duplex-link $n(7) $n(1) 2Mb 20ms DropTail
```

```

$ns duplex-link $n(7) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(8) $n(5) 2Mb 10ms DropTail
$ns duplex-link $n(8) $n(2) 2Mb 40ms DropTail

```

Μπορείτε τώρα να σώσετε, με *File* → *Save*, και να προσομοιώσετε το *script*. Μπορεί να φανεί στο NAM, ότι η τοπολογία είναι λίγο άκομψη. 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 “n0” και “n3”, 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.

Κόμβος 0: πηγή και προορισμός

```

set udp0 [new Agent/UDP]
$udp0 set packetSize_ 1500
$ns attach-agent $n(0) $udp0
$udp0 set fid_ 0
$ns color 0 green
set sink0 [new Agent/LossMonitor]
$ns attach-agent $n(0) $sink0

```

Κόμβος 3: πηγή και προορισμός

```

set udp3 [new Agent/UDP]
$udp3 set packetSize_ 1500
$ns attach-agent $n(3) $udp3
$udp3 set fid_ 3
$ns color 3 yellow
set sink3 [new Agent/LossMonitor]
$ns attach-agent $n(3) $sink3

```

Σύνδεση των πηγών και των προορισμών

```

$ns connect $udp0 $sink3
$ns connect $udp3 $sink0

```

Στρώμα εφαρμογής

```

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.015
$cbr0 attach-agent $udp0

```

```
set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 1500
$cbr3 set interval_ 0.015
$cbr3 attach-agent $udp3
```

Είναι επιθυμητό, ο πρώτος *CBR agent* να αρχίσει να στέλνει όταν $t = 0.2 \text{ sec}$ και ο δεύτερος όταν $t = 0.7 \text{ sec}$. Αντίστοιχα, ο πρώτος θα σταματήσει όταν $t = 4.2 \text{ sec}$, ενώ ο δεύτερος όταν $t = 3.7 \text{ sec}$. Η διαδικασία κλεισίματος καλείται όταν $t = 4.5 \text{ sec}$.

Ορισμός γεγονότων

```
$ns at 0.2 "$cbr0 start"
$ns at 0.7 "$cbr3 start"
$ns at 3.7 "$cbr3 stop"
$ns at 4.2 "$cbr0 stop"
$ns at 4.5 "finish"
```

Μπορείτε τώρα να αρχίσετε το script πληκτρολογώντας για παράδειγμα “ns lab3.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 τη συνολική καθυστέρηση κάθε ροής;
- Ποιος είναι ο ρόλος των εντολών `$udp0 set packetSize_ 1500` και `$udp3 set packetSize_ 1500`; Τι παρατηρείτε στις ροές των πακέτων αν αφαιρεθούν οι γραμμές αυτές από τον κώδικα της προσομοίωσης;

2 Στατική και δυναμική δρομολόγηση

Στην προηγούμενη ενότητα είδατε ότι η κίνηση ακολουθεί τη συντομότερη διαδρομή (αυτήν με τον μικρότερο αριθμό βημάτων) από τον κόμβο “0” στον κόμβο “3” 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*. Κάνετε τη ζεύξη μεταξύ των κόμβων “1” και “2” (που χρησιμοποιείται για τη μετάδοση) να διακοπεί όταν $t = 1.7 \text{ sec}$ και να επανέλθει όταν $t = 2.7 \text{ sec}$. Για να υλοποιήσετε το παραπάνω, τροποποιήστε τον ορισμό γεγονότων ως ακολούθως:

Ορισμός γεγονότων

```
$ns at 0.2 "$cbr0 start"
```

```

$ns at 0.7 "$cbr3 start"
$ns rtmodel-at 1.7 down $n(1) $n(2)
$ns rtmodel-at 2.7 up $n(1) $n(2)
$ns at 3.7 "$cbr3 stop"
$ns at 4.2 "$cbr0 stop"
$ns at 4.5 "finish"

```

Μπορείτε τώρα να αρχίσετε το *script* πάλι και θα δείτε ότι για το χρονικό διάστημα μεταξύ 1.7 και 2.7 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 τις επόμενες γραμμές στην αρχή του *Tcl script*, μετά τη δημιουργία του *object* προσομοίωσης.

```

Agent/rtProto/Direct set preference_ 200
$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 όταν γίνεται διακοπή ζεύξης και έχουμε δυναμική δρομολόγηση; Περιγράψτε με απλά λόγια τη διαδικασία που λαμβάνει χώρα στο *animation*. Συμπίπτει η αρχική με τη μόνιμη διαδρομή δρομολόγησης για τις δύο ροές κατά τη διάρκεια της διακοπής;
- Με βάση το *animation*, προσδιορίστε για κάθε ροή τη χρονική στιγμή όπου παρατηρείται η μόνιμη διαδρομή δρομολόγησης κατά τη διάρκεια διακοπής.
- Για ποιο λόγο τα πακέτα ακολουθούν τις συγκεκριμένες διαδρομές αφότου πέσει η σύνδεση, στην αρχική και τη μόνιμη κατάσταση;
- Θα μπορούσαν να δρομολογηθούν από άλλους κόμβους;
- Ποιος από όλους τους κόμβους καθορίζει από ποια διαδρομή θα προωθηθούν κάθε φορά τα πακέτα;

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. Θεωρείστε λοιπόν ότι το κόστος μιας ζεύξης είναι ανάλογο της καθυστέρησής της, υποθέτοντας κόστος ίσο με $d/10$ στην περίπτωση καθυστέρησης d ms.

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 ($x \rightarrow y$). Για να αυξηθεί το κόστος και προς την άλλη φορά της σύνδεσης ($y \rightarrow x$) θα πρέπει να εισαχθεί η εντολή:

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

Τρέξτε την προσομοίωση και περιγράψτε τι συμβαίνει χρησιμοποιώντας το NAM.

3.1 Ερωτήσεις

- Ποιες διαδρομές ακολουθούν τα πακέτα πριν, κατά τη διάρκεια και μετά την πτώση της σύνδεσης για τις δύο ροές;
- Για ποιον λόγο τα πακέτα ακολουθούν τις συγκεκριμένες διαδρομές;
- Θα μπορούσαν να δρομολογηθούν από άλλους κόμβους;
- Μετά την αποκατάσταση της ζεύξης μεταξύ των κόμβων “1” και “2”, προσδιορίστε με βάση το animation τη χρονική στιγμή όπου παρατηρείται η μόνιμη διαδρομή δρομολόγησης για κάθε ροή.
- Μετά την αποκατάσταση της ζεύξης μεταξύ των κόμβων “1” και “2”, προσδιορίστε με βάση το animation τη χρονική στιγμή όπου παρατηρείται η μόνιμη διαδρομή δρομολόγησης για κάθε ροή.
- Ποιος είναι ο ρόλος της εντολής `Agent/rtProto/Direct set preference_ 200;` Τι παρατηρείτε στη δρομολόγηση των πακέτων αν αφαιρεθεί η εντολή αυτή από τον κώδικα της προσομοίωσης; Αιτιολογείστε γιατί συμβαίνει αυτό.

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 f3 [open out3.tr w]
```

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

```
proc record {} {
```

```
    global sink0 sink3 f0 f3
```

```
    set ns [Simulator instance]
```

Ορισμός της χρονικής περιόδου που θα ξανακληθεί η διαδικασία

```
    set time 0.015
```

Καταγραφή των bytes

```
    set bw0 [$sink3 set bytes_]
```

```
    set bw3 [$sink0 set bytes_]
```

Ορισμός του χρόνου της τρέχουσας καταγραφής

```
    set now [$ns now]
```

Υπολογισμός του bandwidth και καταγραφή αυτού στο αρχείο

```
    puts $f0 "$now [expr (($bw0/$time)*8)/1000000]"
```

```
    puts $f3 "$now [expr (($bw3/$time)*8)/1000000]"
```

Κάνει την μεταβλητή bytes 0

```
    $sink0 set bytes_ 0
```

```
    $sink3 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 f3
```

```
    $ns flush-trace
```

```
    close $nf
```

```
    close $f0
```

```
    close $f3
```

```
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 τη διακοπή και την αποκατάσταση της σύνδεσης, 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 out3.tr”. Θα διαπιστώσετε ότι οι δύο γραφικές παραστάσεις απεικονίζονται με το ίδιο χρώμα, με αποτέλεσμα να είναι δύσκολο να προσδιορίσετε σε ποια κίνηση αντιστοιχούν. Μπορείτε, ωστόσο, να ορίσετε το χρώμα κάθε γραφικής παράστασης τροποποιώντας τα αρχεία εισόδου. Πιο συγκεκριμένα, αφού τρέξετε την προσομοίωση, ανοίξτε τα αρχεία .tr που έχουν δημιουργηθεί και προσθέστε τις γραμμές `color = green` και `color = yellow`, αντίστοιχα. Πρέπει να τονιστεί, ότι οι γραμμές αυτές πρέπει να προηγούνται των δεδομένων απεικόνισης, δηλαδή, πρέπει να τοποθετηθούν στην αρχή κάθε αρχείου.

Αλλάξτε την κίνηση του ενός αποστολέα από σταθερής ροής (CBR) σε εκθετική θέτοντας “Exponential” όπου υπάρχει “CBR”. Για παράδειγμα, τροποποιήστε τις εντολές ως ακολούθως:

```
set exp3 [new Application/Traffic/Exponential]
$exp3 set packetSize_ 1500
$exp3 set rate_ 800k
$exp3 attach-agent $udrp3
```

Για να βγάλετε σωστά συμπεράσματα, θα πρέπει να αυξήσετε τον χρόνο αποστολής της κίνησης, και φυσικά ανάλογα και της προσομοίωσης, τουλάχιστον σε 20 δευτερόλεπτα και να ξανατρέξετε το *script*.

4.1 Ερωτήσεις

- Ποιος είναι ο μέγιστος ρυθμός μετάδοσης που επιτυγχάνεται για τις δύο περιπτώσεις κίνησης, βάσει των γραφικών παραστάσεων που σχεδιάσατε;
- Αιτιολογίστε τις μέγιστες τιμές που προσδιορίσατε παραπάνω, χρησιμοποιώντας τις παραμέτρους που θέσατε για τη διαμόρφωση των δύο πηγών κίνησης (CBR και Exponential).
- Υπολογίστε το πλήθος των bytes που λαμβάνονται επιτυχώς στον προορισμό για κάθε ροή, θεωρώντας ότι και οι δύο ροές ολοκληρώνονται σε χρόνο $t=20+(a/10)\text{sec}$, όπου a τα δύο τελευταία ψηφία του αριθμού μητρώου σας.

Σημείωση: Μπορείτε να σώσετε τις γραφικές παραστάσεις πατώντας “Print Screen”. Ύστερα ανοίξτε το “Word” και κάντε *Paste* την εικόνα.