


ΔΙΚΤΥΑ ΕΠΙΚΟΙΝΩΝΙΩΝ

Εργαστηριακή Άσκηση 2

1. Συνθετότερα προβλήματα με το NS2

Στο πρώτο μέρος της Άσκησης 2 θα ορίσουμε στο NS2 μια τοπολογία δικτύου με τέσσερις κόμβους, στην οποία ένας κόμβος λειτουργεί ως δρομολογητής και προωθεί τα δεδομένα που στέλνουν δύο κόμβοι στον τέταρτο κόμβο. Θα βρούμε έναν τρόπο να διακρίνουμε τις ροές δεδομένων των δύο κόμβων αποστολής και θα δείξουμε πώς μπορεί να επιβλέπεται κάθε ουρά, ώστε να βλέπουμε πόσο γεμάτη είναι και πόσα πακέτα απορρίπτονται.

Σημείωση: Είναι σκόπιμο να ανατρέξετε στο φυλλάδιο της προηγούμενης Άσκησης, 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 (Save). Για να εμφανιστεί ο δρομέας (cursor) στο “Command Prompt”, όταν είναι ανοιχτά το NAM ή το *Xgraph*, πρέπει πρώτα να κλείσετε τα παραθύρά 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. Μια έκδοση του κώδικα δίδεται στο τέλος της Ενότητας για συμβουλευτικό σκοπό και μόνο, π.χ. σε περίπτωση λάθους, κ.λπ.

1.1 Τοπολογία

Όπως πάντα, το πρώτο βήμα είναι ο ορισμός της τοπολογίας του δικτύου. Θα πρέπει να δημιουργήσετε ένα αρχείο, π.χ. “lab2a.tcl”, με τον τρόπο που περιγράφεται στην *Εργαστηριακή Άσκηση 1*, χρησιμοποιώντας τον κώδικα της *Ενότητας 2.2* εκείνης της άσκησης ως υπόδειγμα. Όπως ειπώθηκε 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”, να τα αρχικοποιείτε και να ορίζετε μια διαδικασία που να τα κλείνει. Εισάγετε τώρα τις παρακάτω γραμμές στο αρχικό *template* του προηγούμενου κώδικα, ώστε να δημιουργήσετε 4 κόμβους.

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

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

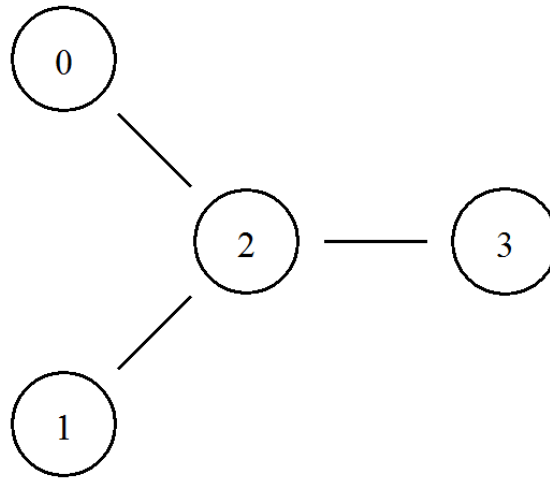
```
$ns duplex-link $n0 $n2 4Mb 8ms DropTail
$ns duplex-link $n1 $n2 4Mb 8ms DropTail
$ns duplex-link $n3 $n2 4Mb 8ms DropTail
```

Μπορείτε τώρα να σώσετε το αρχείο, με *File* → *Save*, και να τρέξετε το *script*. Μπορεί να φανεί στο NAM, ότι η τοπολογία είναι λίγο άκομψη. Μπορείτε να κάνετε κλικ στο κουμπί “re-layout” για να την κάνετε να φαίνεται καλύτερη, αλλά θα ήταν καλό να υπάρχει περισσότερος έλεγχος στο σχέδιο. Προσθέστε τις επόμενες γραμμές στο *Tcl script*, ξανασώστε και ξαναρχίστε το.

```
$ns duplex-link-op $n0 $n2 orient right-down
```

```
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

Προφανώς αντιλαμβάνεστε τι κάνει αυτός ο κώδικας, όταν κοιτάξετε τώρα την τοπολογία στο παράθυρο του NAM. Θα πρέπει να έχει τη μορφή που φαίνεται στην Εικόνα 1.



Εικόνα 1 - Αναπαράσταση της τοπολογίας τεσσάρων κόμβων στο NAM

Παρατηρήστε ότι τα μέρη του NAM που έχουν σχέση με την αυτόματη σχεδίαση έχουν εξαφανιστεί, επειδή τώρα αναλαμβάνετε εσείς το σχέδιο. Μπορείτε να πειραματιστείτε με αυτό, προς το παρόν όμως αφήστε την τοπολογία όπως είναι.

1.2 Τα γεγονότα (events)

Δημιουργήστε τώρα δύο *UDP agents* με πηγές που παράγουν κίνηση *CBR* και προσαρτήστε τους στους κόμβους “n0” και “n1”. Έπειτα δημιουργήστε έναν *Sink Agent* και προσαρτήστε τον στον κόμβο “n3”.

Δημιουργία ενός UDP agent και «προσάρτησή» του στον κόμβο «n0»

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
```

Δημιουργία μιας πηγής κίνησης CBR traffic source και «τοποθέτησή» της στον «udp0»

```
set cbr0 [new Application/Traffic/CBR]
```

Προσδιορισμός της κίνησης δεδομένων που παράγεται στον κόμβο «n0»

```
$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

Δημιουργία ενός UDP agent και «προσάρτησή» του στον κόμβο «n1»

```
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
```

Δημιουργία μιας πηγής κίνησης CBR και «τοποθέτησή» της στον «udp1»

```
set cbr1 [new Application/Traffic/CBR]
```

Προσδιορισμός της κίνησης δεδομένων που παράγεται στον κόμβο «n1»

```
$cbr1 set packetSize_ 1500
$cbr1 set interval_ 0.0045
$cbr1 attach-agent $udp1
```

Δημιουργία δύο agents *sink0* και *sink1* για τη λήψη δεδομένων

```
set sink0 [new Agent/LossMonitor]
$ns attach-agent $n3 $sink0
set sink1 [new Agent/LossMonitor]
$ns attach-agent $n3 $sink1
```

Σύνδεση των δύο CBR agents με τους *sink agents*

```
$ns connect $udp0 $sink0
$ns connect $udp1 $sink1
```

Είναι επιθυμητό να αρχίσει να στέλνει ο πρώτος *CBR agent* όταν $t = 0.6 \text{ sec}$ και να σταματήσει όταν το $t = 9.0 \text{ sec}$, ενώ ο δεύτερος *CBR agent* να αρχίσει όταν $t = 1.6 \text{ sec}$ και να σταματήσει όταν $t = 8.0 \text{ sec}$.

```
$ns at 0.6 "$cbr0 start"
$ns at 1.6 "$cbr1 start"
$ns at 8.0 "$cbr1 stop"
$ns at 9.0 "$cbr0 stop"
```

Αρχίστε το script πληκτρολογώντας “ns lab2a.tcl” και ύστερα τρέξτε το *animation*.

- Εξηγήστε τις παρατηρήσεις σας.

Είναι προφανές ότι κάποια πακέτα από τις ροές χάνονται, αλλά δεν είναι εύκολο να τα προσδιορίσουμε. Και οι δύο ροές παριστάνονται με μαύρο χρώμα, οπότε ο μόνος τρόπος να διαπιστωθεί τι συμβαίνει στα πακέτα είναι να τα παρακολουθεί κάποιος στο NAM κάνοντας κλικ πάνω τους. Στις επόμενες ενότητες αναφέρεται ο τρόπος διάκρισης των δύο διαφορετικών ροών και ο τρόπος παρακολούθησης του τι πραγματικά συμβαίνει στην ουρά της ζεύξης από “n2” προς “n3”.

1.3 Σημάδεμα ροών

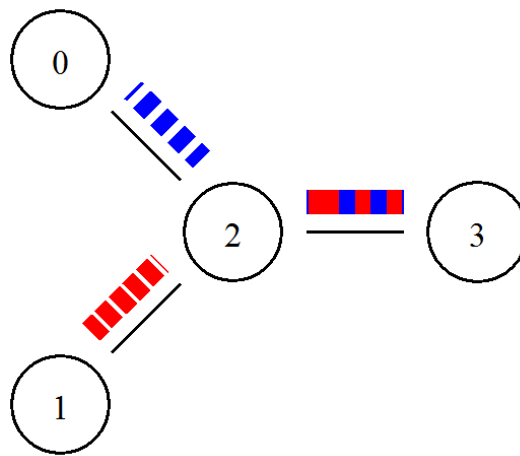
Προσθέστε τις παρακάτω δύο γραμμές στους δυο ορισμούς των *udp agents* αντίστοιχα.

```
$udp0 set class_ 1
$udp1 set class_ 2
```

Προσθέστε τώρα το παρακάτω τμήμα κώδικα στο *Tcl script*, κατά προτίμηση στην αρχή μετά τη δημιουργία του αντικείμενου (*object*) προσομοίωσης.

```
$ns color 1 Blue
$ns color 2 Red
```

Ο κώδικας αυτός σας επιτρέπει να αντιστοιχίσετε διαφορετικά χρώματα σε κάθε ροή πακέτου. Σώστε και τρέξτε το *script* και θα δείτε στο NAM ότι η μια ροή θα είναι μπλε ενώ η άλλη θα είναι κόκκινη, όπως φαίνεται στην *Εικόνα 2*.



Εικόνα 2 – Αναπαράσταση της κίνησης με έγχρωμη ροή πακέτων στο NAM

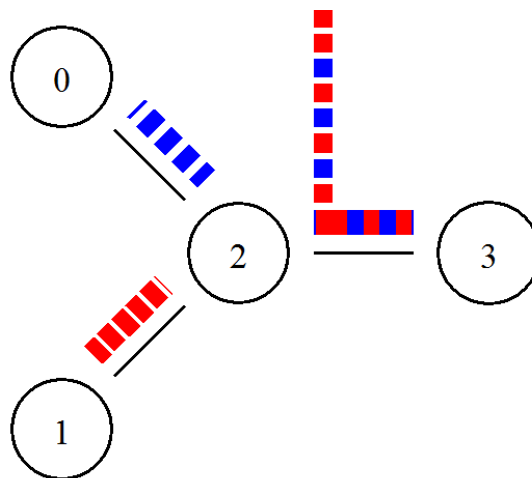
Αν παρατηρήσετε τη ζεύξη από “n2” προς “n3” για λίγο, και θα διαπιστώσετε ότι, μετά από παρέλευση κάποιου χρονικού διαστήματος, η κατανομή μεταξύ μπλε και κόκκινων πακέτων δεν είναι πλέον ίδια. Στην επόμενη ενότητα αναφέρεται ο τρόπος με τον οποίο μπορείτε να δείτε μέσα στην ουρά αναμονής κάθε ζεύξης για να παρατηρήσετε τι συμβαίνει.

1.4 Παρακολούθηση ουράς με το NAM

Πρέπει να προσθέσετε την παρακάτω γραμμή ορισμού της θέσης της ουράς στον κώδικά σας, για να παρακολουθήσετε την ουρά της ζεύξης από “n2” προς “n3”.

```
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

Σώστε και τρέξτε πάλι το αρχείο, αρχίστε το NAM και θα δείτε μια εικόνα παρόμοια με την παρακάτω.




Εικόνα 3 – Αναπαράσταση στο NAM της ουράς πακέτων στη ζεύξη

Τώρα, μπορείτε να βλέπετε τα πακέτα στην ουρά, ενώ μετά από λίγο θα παρατηρήσετε πως απορρίπτονται κάποια πακέτα. Θα διαπιστώσετε ότι απορρίπτονται κυρίως μπλε πακέτα, αλλά δεν μπορούμε να περιμένουμε δικαιοσύνη από μια απλή ουρά *DropTail*.

Για να βελτιώσετε την ουρά αναμονής, κάνοντάς την πιο “δίκαιη”, χρησιμοποιήστε μια ουρά *SFQ* (Stochastic Fair Queuing) για τη ζεύξη από “n2” προς “n3”. Για να το κάνετε αυτό, αλλάξτε

την γραμμή που ορίζει τη ζεύξη μεταξύ “n2” και “n3” με την παρακάτω γραμμή:

```
$ns duplex-link $n3 $n2 4Mb 8ms SFQ
```

Σημείωση: Θυμηθείτε ότι πρέπει να κλείνεται το NAM **μόνο** πατώντας το “X” δεξιά επάνω στην μπάρα του παραθύρου  και **όχι** από το μενού **File**.

1.5 Παρακολούθηση ουράς με το Xgraph

Για να μπορέσετε να δείτε τα αποτελέσματα της προσομοίωσης και σε γραφική παράσταση, θα πρέπει να ορίσετε μια διαδικασία καταγραφής της κίνησης σε ένα αρχείο εξόδου *trace*, προσθέτοντας τις παρακάτω γραμμές στο *script*, κάτω από τη εντολή ορισμού της θέσης της “ουράς” που είδαμε στην §1.4.

Διαδικασία (procedure) καταγραφής της κίνησης

```
proc record {} {  
    global sink0 sink1 f0 f1  
    set ns [Simulator instance]
```

Ορισμός της χρονικής περιόδου που θα ξανακληθεί η διαδικασία.

```
set time 0.25
```

Καταγραφή των bytes

```
set bw0 [$sink0 set bytes_  
set bw1 [$sink1 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 $f1 "$now [expr $bw1/$time*8/1000000]"
```

Κάνει την τιμή bytes_ 0

```
$sink0 set bytes_ 0  
$sink1 set bytes_ 0
```

Επαναπρογραμματισμός της διαδικασίας.

```
$ns at [expr $now+$time] "record"  
}
```

Η διαδικασία της καταγραφής θα πρέπει να κληθεί στην αρχή της προσομοίωσης, βάζοντας τις παρακάτω γραμμές στους καθορισμούς των χρόνων των γεγονότων (events).

Αρχή της καταγραφής δεδομένων

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

Επίσης, πρέπει να εισαχθούν και οι παρακάτω γραμμές για τη δημιουργία του αρχείου του *Xgraph*, μετά τη δημιουργία αρχείου εγγραφής για το NAM.

Δημιουργία του αρχείου εγγραφής που θα χρησιμοποιηθεί με το Xgraph

```
set f0 [open lab2a0.tr w]
set f1 [open lab2a1.tr w]
```

Τέλος, στη διαδικασία κλεισίματος των αρχείων θα πρέπει να προστεθούν οι εξής εντολές:

Κλείσιμο των αρχείων εξόδου

```
close $f0
close $f1
```

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 lab2a0.tr lab2a1.tr”.

1.6 Ερωτήσεις

- Ποια είναι η μέγιστη τιμή του ρυθμού μεταφοράς των δεδομένων για τις δυο ροές, όπως προκύπτει από τις γραφικές παραστάσεις του Xgraph (για ουρά DropTail και ουρά SFQ);
- Ποια είναι η ελάχιστη τιμή του ρυθμού μεταφοράς των δεδομένων για τις δυο ροές, στο διάστημα που αποστέλλουν δεδομένα και οι δύο πηγές, για κάθε τύπο ουράς;
- Ποιο είναι το μέγιστο ποσοστό των πακέτων που χάνονται από την μπλε και κόκκινη ροή για τους δύο τύπους ουρών; Ποιες είναι οι μέγιστες απώλειες κάθε ροής σε bit/sec;
- Παρατηρώντας το *animation*, εκτιμήστε το ποσοστό των πακέτων που χάνονται από την μπλε και από την κόκκινη ροή, όταν χρησιμοποιείται η ουρά SFQ;
- Είναι τα ποσοστά αυτά αναμενόμενα, αν λάβουμε υπόψη τον ρυθμό μετάδοσης κάθε πηγής, τη χωρητικότητα των ζεύξεων και τον τύπο ουράς;
- Στις γραφικές παραστάσεις του Xgraph υπάρχουν διαστήματα με μηδενικό ρυθμό μετάδοσης και διαστήματα που η μια ροή από μόνη της ξεπερνά τον καθορισμένο ρυθμό μετάδοσης της αντίστοιχης πηγής στη ζεύξη 2-3. Πώς ερμηνεύετε αυτή τη συμπεριφορά για κάθε τύπο ουράς;

2 Μετάδοση δεδομένων σε δίκτυο με σύνθετη τοπολογία

Στις υπόλοιπες ενότητες 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 έντεκα 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.

2.1 Τοπολογία

Όπως πάντα, το πρώτο βήμα είναι ο ορισμός της τοπολογίας του δικτύου. Θα πρέπει να δημιουργήσετε ένα αρχείο, π.χ. “lab2b.tcl”, όπως και στην προηγούμενη ενότητα. Εισάγετε τις παρακάτω γραμμές στο αρχικό *template* του κώδικα, ώστε να δημιουργήσετε 11 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} {$i < 11} {incr i} {
    set n($i) [$ns node]
}
```

Το παρακάτω τμήμα κώδικα *Tcl* δημιουργεί αμφίδρομες ζεύξεις μεταξύ των πρώτων 9 κόμβων, ορίζοντας καθυστέρηση ζεύξης ίση με 30 ms.

```
for {set i 0 } {$i < 9} {incr i} {
    $ns duplex-link $n($i) $n([expr ($i+1)%9]) 2Mb 30ms DropTail
}
```

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

```
$ns duplex-link $n(9) $n(2) 2Mb 15ms DropTail
$ns duplex-link $n(9) $n(4) 2Mb 20ms DropTail
$ns duplex-link $n(10) $n(4) 2Mb 10ms DropTail
$ns duplex-link $n(10) $n(8) 2Mb 30ms 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.

2.2 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 §2.1. 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 blue
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_ 0.025
```

```
$cbr0 attach-agent $udp0
```

```
set cbr3 [new Application/Traffic/CBR]
```

```
$cbr3 set packetSize_ 1500
```

```
$cbr3 set interval_ 0.025
```

```
$cbr3 attach-agent $udp3
```

Είναι επιθυμητό, ο πρώτος *CBR agent* να αρχίσει να στέλνει όταν $t = 0.5 \text{ sec}$ και ο δεύτερος όταν $t = 1.1 \text{ sec}$. Αντίστοιχα, ο πρώτος θα σταματήσει όταν $t = 3.4 \text{ sec}$, ενώ ο δεύτερος όταν $t = 4.0 \text{ sec}$. Η διαδικασία κλεισίματος καλείται όταν $t = 4.5 \text{ sec}$.

Ορισμός γεγονότων

```
$ns at 0.5 "$cbr0 start"
```

```
$ns at 1.1 "$cbr3 start"
```

```
$ns at 3.4 "$cbr3 stop"
```

```
$ns at 4.0 "$cbr0 stop"
```

```
$ns at 4.5 "finish"
```

Μπορείτε τώρα να αρχίσετε το script πληκτρολογώντας για παράδειγμα “ns lab2b.tcl” και ύστερα τρέξετε το *animation*.

2.3 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;

3 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: τη διακοπή ζεύξης. Κάνετε τη ζεύξη μεταξύ των κόμβων “1” και “2” (που χρησιμοποιείται για τη μετάδοση) να διακοπεί όταν $t = 1.8 \text{ sec}$ και να επανέλθει όταν $t = 2.5 \text{ sec}$. Για να υλοποιήσετε το παραπάνω, τροποποιήστε τον ορισμό γεγονότων ως ακολούθως:

Ορισμός γεγονότων

```
$ns at 0.5 "$cbr0 start"
$ns at 1.1 "$cbr3 start"
$ns rtmodel-at 1.8 down $n(1) $n(2)
$ns rtmodel-at 2.5 up $n(1) $n(2)
$ns at 3.4 "$cbr3 stop"
$ns at 4.0 "$cbr0 stop"
$ns at 4.5 "finish"
```

Μπορείτε τώρα να αρχίσετε το *script* πάλι και θα δείτε ότι για το χρονικό διάστημα μεταξύ 1.8 και 2.5 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.

3.1 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;

4 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.

4.1 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.

5 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 = red` και `color = orange`, 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_ 750k
```

```
$exp3 attach-agent $udp3
```

Για να βγάλετε σωστά συμπεράσματα, θα πρέπει να αυξήσετε τον χρόνο αποστολής της κίνησης, και φυσικά ανάλογα και της προσομοίωσης, τουλάχιστον σε 30 δευτερόλεπτα και να ξανατρέξετε το *script*.

5.1 Ερωτήσεις

- Ποιος είναι ο μέγιστος ρυθμός μετάδοσης που επιτυγχάνεται για τις δύο περιπτώσεις κίνησης, βάσει των γραφικών παραστάσεων που σχεδιάσατε;
- Αιτιολογείστε 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 $t=30+(a/10)\text{sec}$, 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.

Ολοκληρωμένος κώδικας για την Εργαστηριακή Άσκηση 2, Ενότητα 1

```
set ns [new Simulator]

$ns color 1 Blue
$ns color 2 Red

set nf [open lab2a.nam w]
$ns namtrace-all $nf

set f0 [open lab2a0.tr w]
set f1 [open lab2a1.tr w]

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 4Mb 8ms DropTail
$ns duplex-link $n1 $n2 4Mb 8ms DropTail
$ns duplex-link $n3 $n2 4Mb 8ms DropTail

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
# Προσδιορισμός της θέσης της γραμμής που αναπαριστά την ουρά στο NAM
$ns duplex-link-op $n2 $n3 queuePos 0.5

proc record {} {
    global sink0 sink1 f0 f1
    set ns [Simulator instance]
    set time 0.25
    set bw0 [$sink0 set bytes_]
    set bw1 [$sink1 set bytes_]
    set now [$ns now]
    puts $f0 "$now [expr $bw0/$time*8/1000000]"
    puts $f1 "$now [expr $bw1/$time*8/1000000]"
    $sink0 set bytes_ 0
    $sink1 set bytes_ 0
    $ns at [expr $now+$time] "record"
}

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

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
$udp0 set class_ 1

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
```

```
$udp1 set class_ 2

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 1500
$cbr1 set interval_ 0.0045
$cbr1 attach-agent $udp1

set sink0 [new Agent/LossMonitor]
$ns attach-agent $n3 $sink0

set sink1 [new Agent/LossMonitor]
$ns attach-agent $n3 $sink1

$ns connect $udp0 $sink0
$ns connect $udp1 $sink1

$ns at 0.0 "record"
$ns at 0.6 "$cbr0 start"
$ns at 1.6 "$cbr1 start"
$ns at 8.0 "$cbr1 stop"
$ns at 9.0 "$cbr0 stop"
$ns at 9.5 "finish"

$ns run
```