ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ

ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ ΚΑΙ ΜΗΧΑΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΊΚΤΥΑ ΕΠΙΚΟΙΝΩΝΙΏΝ

(2018-2019)

2η ΕΡΓΑΣΤΗΡΙΑΚΉ ΑΝΑΦΟΡΆ

Ονοματεπώνυμο : Χρήστος Τσούφης

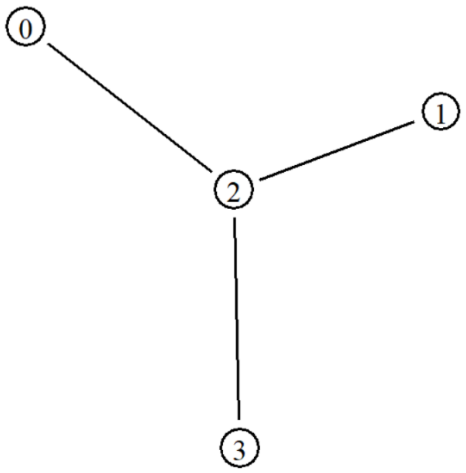
Α.Μ. : 03117176

1. Συνθετότερα προβλήματα με το NS2

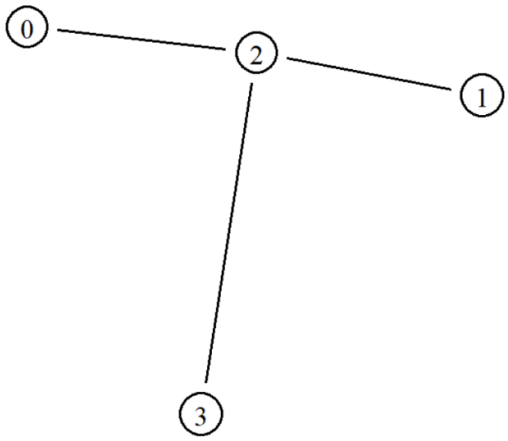
Στο πρώτο μέρος της Άσκησης 2 ορίζεται στο NS2 μια τοπολογία δικτύου με τέσσερις κόμβους, στην οποία ένας κόμβος λειτουργεί ως δρομολογητής και προωθεί τα δεδομένα που στέλνουν δύο κόμβοι στον τέταρτο κόμβο. Επίσης, θα βρεθεί ένας τρόπος να διακρίνονται οι ροές δεδομένων των δύο κόμβων αποστολής και θα δειχθεί πώς μπορεί να επιβλέπεται κάθε ουρά, ώστε να φαίνεται πόσο γεμάτη είναι και πόσα πακέτα απορρίπτονται.

1.1. Τοπολογία

Αρχικά δημιουργούνται 4 κόμβοι και έπειτα, τρεις αμφίδρομες ζεύξεις μεταξύ τους. Προκύπτει έτσι:



Και με την εντολή re-layout:



Και ο πηγαίος κώδικας:

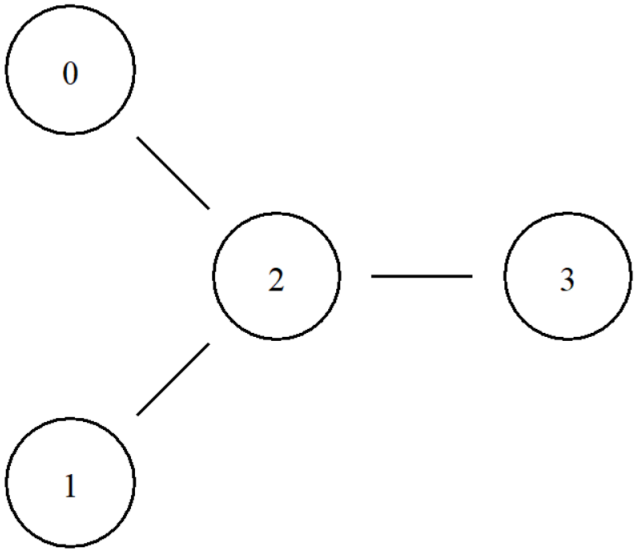
```
set ns [new Simulator]

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

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
```

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

$ns at 9.5 "finish"
$ns run
```

Και με την προσθήκη:

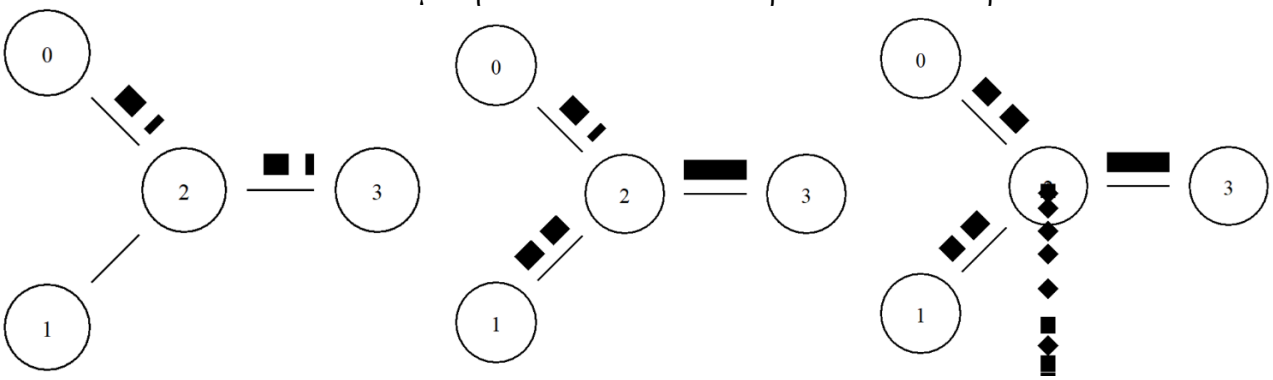
```
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

Προκύπτει:



1.2. Τα γεγονότα (events)

Ύστερα, δημιουργούνται δύο UDP agents με πηγές που παράγουν κίνηση CBR και συνδέονται με τους κόμβους “n0” και “n1”. Κατόπιν, δημιουργείται ένας Sink Agent και συνδέεται με τον κόμβο “n3”. Είναι επιθυμητό να αρχίσει να στέλνει ο πρώτος CBR agent όταν t = 0.6 sec και να σταματήσει όταν το t = 9.0 sec, ενώ ο δεύτερος CBR agent να αρχίσει όταν t = 1.6 sec και να σταματήσει όταν t = 8.0 sec. Προκύπτουν τα παρακάτω:



1 2 3

Είναι προφανές ότι κάποια πακέτα από τις ροές χάνονται, αλλά δεν είναι εύκολο να προσδιορισθούν. 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".

Το animation επιβεβαιώνει ότι κάποια πακέτα χάνονται. Η κάθε ζεύξη όπως ορίστηκε μπορεί να μεταφέρει κάθε χρονική στιγμή μέχρι και 4.000.000 $\frac{bits}{sec}$ ∗ 0,008 sec = 32.000 $bits$ = 4.000 bytes

ενώ η ζεύξη μεταξύ των κόμβων n0 και n2 μεταφέρει $\frac{1.500\ bytes}{0{,}005\ sec}$ * 0,008 sec = 2.400 bytes

και η ζεύξη μεταξύ των κόμβων n1 και n2 μεταφέρει $\frac{1.500\ bytes}{0{,}0045\ sec}$ * 0,008 sec ≈2.666,7 bytes

Το άθροισμά τους (2.400+2.666,7) = 5.066,7 bytes που προσπαθούν να στείλουν οι δύο πηγές μέσω της 3ης ζεύξης είναι μεγαλύτερο από όσο «αντέχει» η συγκεκριμένη ζεύξη οπότε θα προκύψει και η απώλεια πακέτων που παρατηρείται.

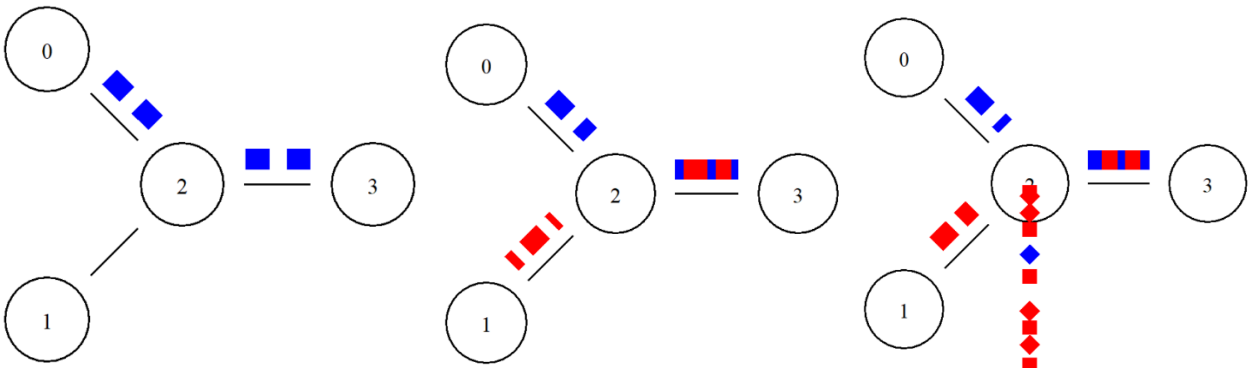
Και ο πηγαίος κώδικας:

```
set ns [new Simulator]

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

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

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

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

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

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

$ns at 0.6 "$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
```

1.3. Σημάδεμα ροών

Οι προσθήκες στον κώδικα αυτό επιτρέπει την αντιστοιχία διαφορετικών χρωμάτων σε κάθε ροή πακέτου. Μετά από παρέλευση κάποιου χρονικού διαστήματος διαπιστώνει κανείς για τη ζεύξη από "n2" προς "n3" ότι η κατανομή μεταξύ μπλε και κόκκινων πακέτων δεν είναι πλέον ίδια. Στην επόμενη ενότητα αναφέρεται ο τρόπος με τον οποίο φαίνεται η ουρά αναμονής κάθε ζεύξης ώστε να παρατηρηθεί τι συμβαίνει.



1 2 3

Και ο πηγαίος κώδικας:

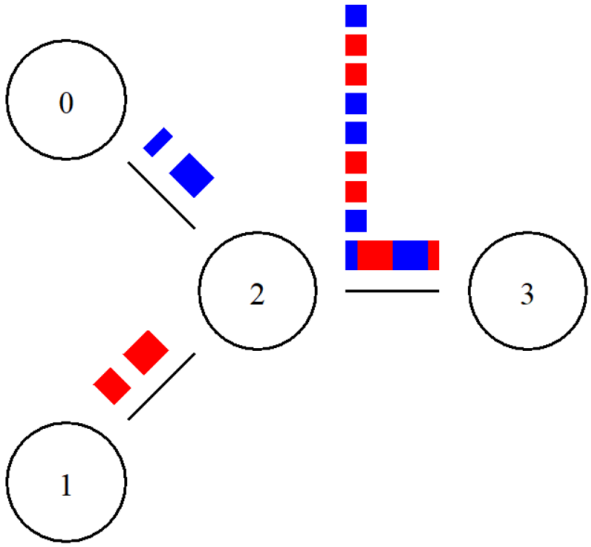
```
set ns [new Simulator]

$ns color 1 Blue
$ns color 2 Red

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

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

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

$ns at 0.6 "$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
```

1.4. Παρακολούθηση ουράς με το NAM

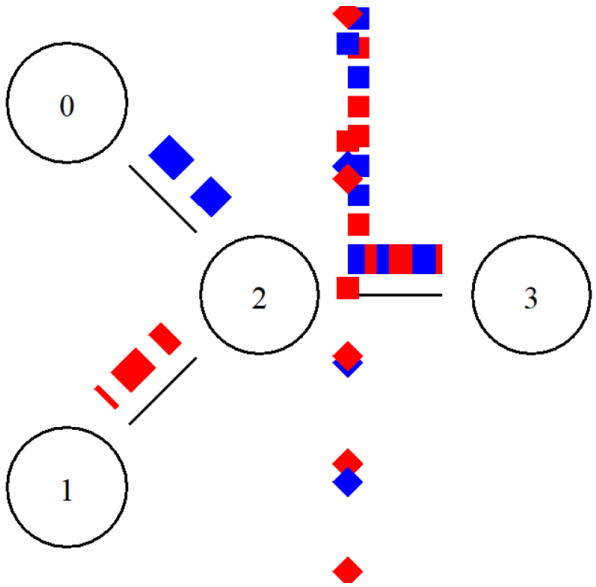
Με την προσθήκη της παρακάτω γραμμής ορισμού της θέσης της ουράς στον κώδικα και παρακολουθώντας την ουρά της ζεύξης από “n2” προς “n3”, προκύπτει:

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



Για να βελτιωθεί η ουρά αναμονής, κάνοντάς την πιο "δίκαιη", χρησιμοποιείται μια ουρά SFQ (Stochastic Fair Queuing) για τη ζεύξη από "n2" προς "n3". Για να γίνει αυτό, αρκεί να αλλάξει η γραμμή που ορίζει τη ζεύξη μεταξύ "n2" και "n3" με την παρακάτω γραμμή οπότε προκύπτει:

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



Και ο πηγαίος κώδικας:

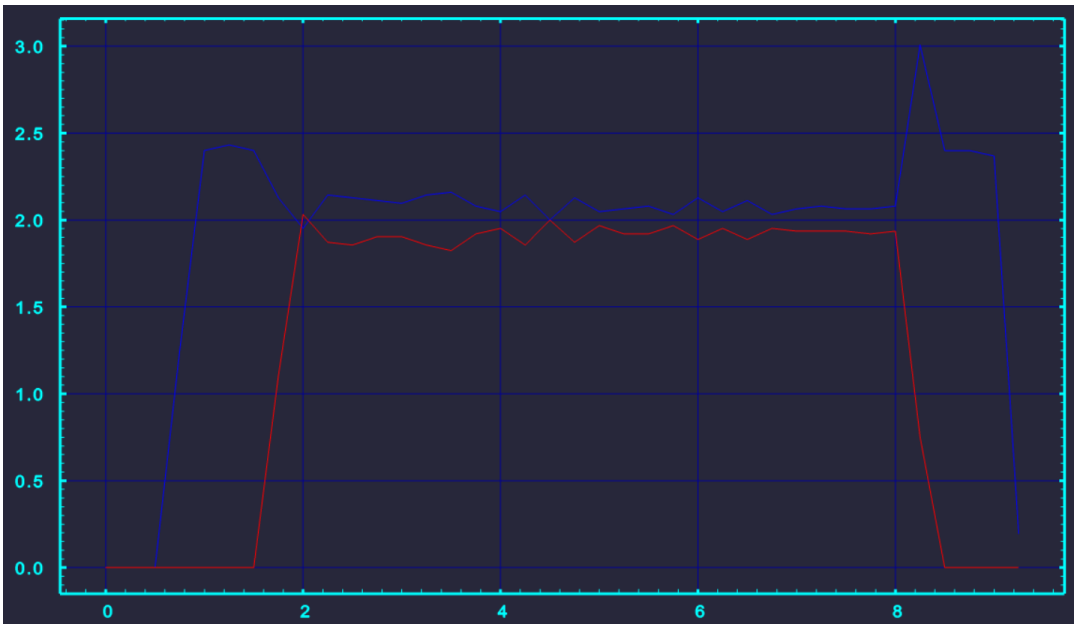
```
set ns [new Simulator]

$ns color 1 Blue
$ns color 2 Red

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

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 SFQ

$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

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

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

$ns at 0.6 "$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
```

1.5. Παρακολούθηση ουράς με το Xgraph

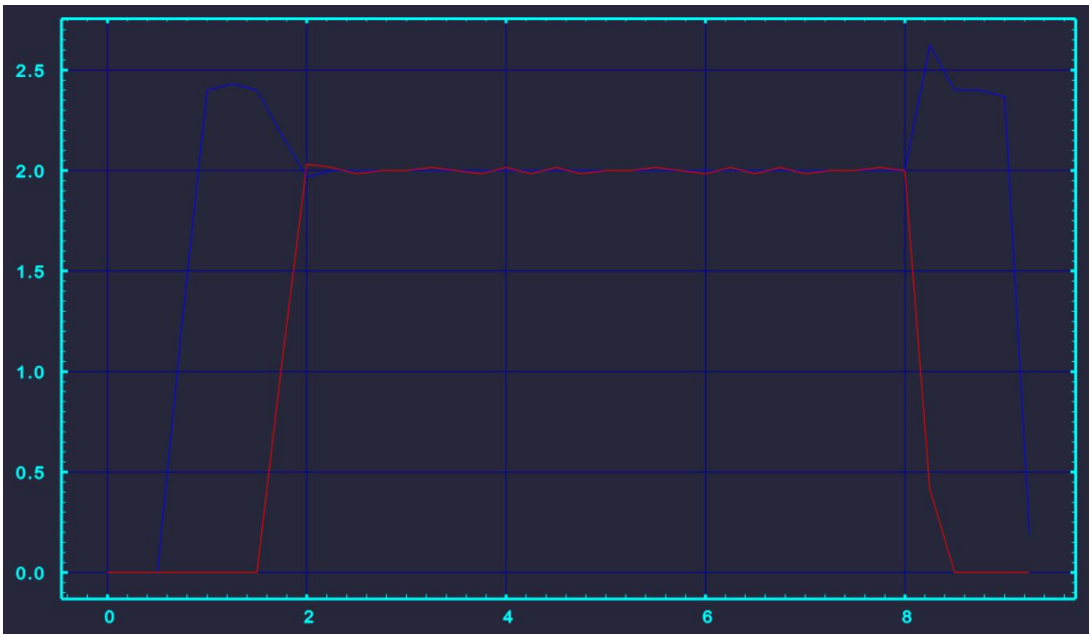
Για να φανούν τα αποτελέσματα της προσομοίωσης και σε γραφική παράσταση, θα πρέπει να ορισθεί μια διαδικασία καταγραφής της κίνησης σε ένα αρχείο εξόδου trace, προσθέτοντας τις κατάλληλες γραμμές στο script, κάτω από τη εντολή ορισμού της θέσης της “ουράς”.

Με DropTail:



Γράφημα 1

Με SFQ:



Γράφημα 2

Και ο πηγαίος κώδικας:

```
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 SFQ ή 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

$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
```

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);*
  Μεγεθύνοντας τις γραφικές παραστάσεις στο Xgraph προκύπτουν τα παρακάτω:
  DropTail ουρά:
  Μέγιστη τιμή ρυθμού μεταφοράς δεδομένων της ροής από τον κόμβο n0 (μπλε ροή): 3 Mbps
  Μέγιστη τιμή ρυθμού μεταφοράς δεδομένων της ροής από τον κόμβο n1 (κόκκινη ροή): 2,03 Mbps
  SFQ ουρά:
  Μέγιστο μπλε ροής: 2,62 Mbps
  Μέγιστο κόκκινης ροής: 2,02 Mbps

- *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;*
  Αντίστοιχη εργασία γίνεται και για τις ελάχιστες τιμές
  DropTail ουρά:
  Ελάχιστη τιμή ρυθμού μεταφοράς δεδομένων της ροής από τον κόμβο n0 (μπλε ροή): 1,95 Mbps
  Ελάχιστη τιμή ρυθμού μεταφοράς δεδομένων της ροής από τον κόμβο n1 (κόκκινη ροή): 1,83 Mbps
  SFQ ουρά:
  Ελάχιστο μπλε ροής: 1,97 Mbps
  Ελάχιστο κόκκινης ροής: 1,98 Mbps

- *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;*
  Για να υπολογισθούν παρακάτω οι μέγιστες απώλειες των ροών αφαιρείται από το ρυθμό μετάδοσης της κάθε ροής η ελάχιστη τιμή που υπολογίστηκε, ενώ για να βρεθούν τα αντίστοιχα ποσοστά διαιρείται αυτή η διαφορά με το ρυθμό μετάδοσης. Παρακάτω υπολογίζονται αυτοί οι ρυθμοί για κάθε μία ροή:

  Μπλε ροή: $\frac{1500\ bytes}{0{,}005\ sec} * 8 = 2.400.000\ \frac{bits}{sec} = 2{,}4$ Mbps
  Κόκκινη ροή: $\frac{1500\ bytes}{0{,}0045\ sec} * 8 = 2.666.666{,}66\ \frac{bits}{sec} \approx 2{,}66$ Mbps

Droptail ουρά:
Μπλε ροή:

Μέγιστες απώλειες: (2,4 – 1,95) Mbps = 0.45 Mbps = 450.000 $\frac{bits}{sec}$

Μέγιστο ποσοστό χαμένων πακέτων: $\frac{0,45}{2,4}$ * 100% ≈ 18,75%

Κόκκινη ροή:

Μέγιστες απώλειες: (2,66 – 1,83) Mbps = 0.83 Mbps = 830.000 $\frac{bits}{sec}$

Μέγιστο ποσοστό χαμένων πακέτων: $\frac{0,83}{2,66}$ * 100% ≈ 31,2%

SFQ ουρά:
Μπλε ροή:

Μέγιστες απώλειες: (2,4 – 1,97) Mbps = 0,43 Mbps = 430.000 $\frac{bits}{sec}$

Μέγιστο ποσοστό χαμένων πακέτων: $\frac{0,43}{2,4}$ * 100% ≈ 17,91 %

Κόκκινη ροή:

Μέγιστες απώλειες: (2,66 – 1,98) Mbps = 0,68 Mbps = 680.000 $\frac{bits}{sec}$

Μέγιστο ποσοστό χαμένων πακέτων: $\frac{0,68}{2,66}$ * 100% ≈ 25,56 %

Επιβεβαιώνεται και από εδώ ότι τα ποσοστά των χαμένων πακέτων και για τις δύο ροές μειώνονται χρησιμοποιώντας την ουρά SFQ.

- *Παρατηρώντας το animation, εκτιμήστε το ποσοστό των πακέτων που χάνονται από την μπλε και από την κόκκινη ροή, όταν χρησιμοποιείται η ουρά SFQ;*
  Με την ουρά SFQ χάνονται περίπου τα μισά πακέτα, δηλαδή το 50% των συνολικών πακέτων. Παρατηρώντας το animation προκύπτει ότι το ποσοστό των πακέτων που χάνονται είναι σαφώς μεγαλύτερο για την κόκκινη ροή. Πιο συγκεκριμένα, για διάστημα 100 msec χάνονται 11 κόκκινα και 8 μπλε πακέτα. Αυτά αντιστοιχούν σε 15*500*8 + 2*1.000*8 = 76.000 "κόκκινα" bit και 11*500*8 + 2*1.000*8 = 60.000 "μπλε" bit. Κανονικά στο χρονικό διάστημα των 100 msec θα έπρεπε να στέλνονται $2,62*10^5$ "μπλε" bit και $2,02*10^5$ "κόκκινα" bit. Άρα, έχουμε απώλεια 23% στα "μπλε" bit και 37% στα "κόκκινα" bit.

- *Είναι τα ποσοστά αυτά αναμενόμενα, αν λάβουμε υπόψη τον ρυθμό μετάδοσης κάθε πηγής, τη χωρητικότητα των ζεύξεων και τον τύπο ουράς;*
  Στην αρχή υπολογίσθηκε ότι και οι δύο κόμβοι μαζί στέλνουν στον δρομολογητή κόμβο περίπου 5.066 $bytes$, εκ των οποίων μπορούν να μεταφερθούν στην 3η ζεύξη μόνο τα 4.000. Έτσι, αναμένεται μια απώλεια των (5.066−4.000) $bytes$ = 1.066 $bytes$.
  Με χρήση ουράς DropTail χάνεται το 18,75% των μπλε πακέτων (450 $bytes$) και το 31,2% των κόκκινων πακέτων (830 $bytes$) άρα συνολικά (450 + 830) $bytes$ = 1.280 $bytes$, αριθμός λίγο παραπάνω από τον αναμενόμενο.
  Από την άλλη, με χρήση της ουράς SFQ βρίσκουμε ότι χάνονται συνολικά (17,91% * 2.400) + (25,56% * 2.666) bytes = (429,84 + 681,42) bytes = 1.111,26 bytes αριθμός αρκετά πιο κοντά στον αναμενόμενο.

  Τα ποσοστά αυτά είναι αναμενόμενα με ένα ποσοστό σφάλματος 10%. Παρ'όλα αυτά, εμφανίζουν το πρόβλημα της απώλειας πακέτων αφού αθροιστικά ο ρυθμός αποστολής των πακέτων από τις πηγές 0 και 1 είναι μεγαλύτερος από τη χωρητικότητα της ζεύξης

2-3 (2.02 + 2.62 = 4.62 > 4). Μάλιστα, παρατηρείται μια σχετικά δίκαιη κατανομή των πακέτων από την ουρά SFQ σε σχέση με την ουρά DropTail. Αν και το ποσοστό απωλειών της κόκκινης ροής φαίνεται να είναι πολύ μεγαλύτερο από αυτό της μπλε ροής είναι λογικό να συμβαίνει αυτό αφού και ο ρυθμός μετάδοσης πακέτων της κόκκινης ροής είναι μεγαλύτερος από της μπλε.

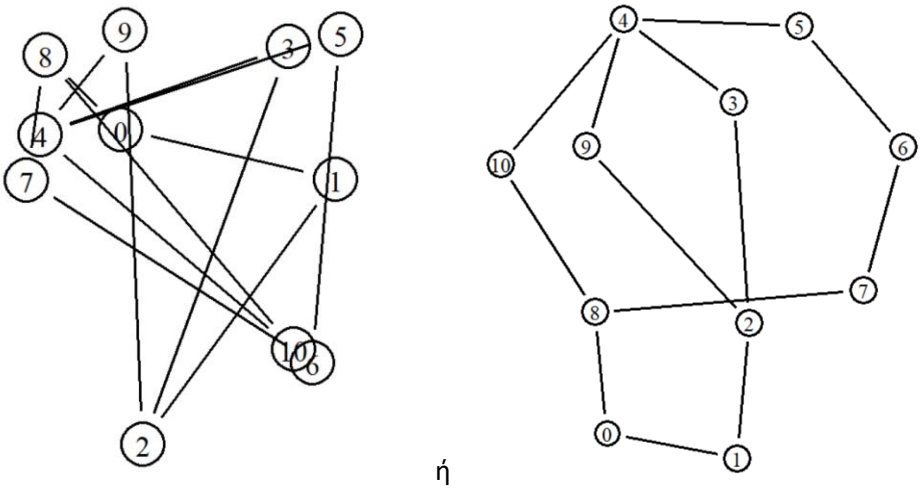
- *Στις γραφικές παραστάσεις του Xgraph υπάρχουν διαστήματα με μηδενικό ρυθμό μετάδοσης και διαστήματα που η μια ροή από μόνη της ξεπερνά τον καθορισμένο ρυθμό μετάδοσης της αντίστοιχης πηγής στη ζεύξη 2-3. Πώς ερμηνεύετε αυτή τη συμπεριφορά για κάθε τύπο ουράς;*
  Τα διαστήματα στα οποία ο ρυθμός μετάδοσης είναι μηδενικός είναι τα διαστήματα εκείνα στα οποία οι δύο κόμβοι δεν στέλνουν πακέτα και δεν υπάρχει κίνηση στο δίκτυο. Με χρήση της ουράς DropTail, παρατηρείται στο Γράφημα 1 ότι στο τέλος όπου μεταδίδει μόνο ο κόμβος n0, ο ρυθμός μετάδοσης φτάνει στα 4 $Mbps$, δηλαδή σε μια τιμή μεγαλύτερη από αυτή της πηγής του (2,4 $Mbps$). Αυτό συμβαίνει διότι μέχρι πριν λίγο η ουρά αναμονής ήταν συνεχώς γεμάτη, αλλά όταν ο κόμβος n1 σταμάτησε την αποστολή κόκκινων πακέτων, η ουρά άρχισε να αδειάζει ταυτόχρονα όμως και με την διέλευση από τον δρομολογητή των μπλε πακέτων από τον κόμβο n0. Το παραπάνω δεν συμβαίνει τόσο εμφανώς στο Γράφημα 2 που αναφέρεται στην ουρά SFQ λόγω της πιο «δίκαιης» αντιμετώπισής της.

2. Μετάδοση δεδομένων σε δίκτυο με σύνθετη τοπολογία

Αυτή η ενότητα αφορά την μετάδοση δεδομένων μεταξύ κόμβων που συνδέονται σε δίκτυο με σχετικά σύνθετη τοπολογία. Θα οριστεί στο NS2 ένα δίκτυο που αποτελείται από έντεκα κόμβους και θα επιλεχθούν δύο κόμβοι του δικτύου, που δεν συνδέονται άμεσα μεταξύ τους, να ανταλλάσσουν δεδομένα. Σκοπός είναι να παρατηρηθεί 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

Με τον παρακάτω κώδικα δημιουργούνται 11 κόμβοι και ορίζονται αμφίδρομες ζεύξεις μεταξύ των πρώτων 9 κόμβων, ορίζοντας καθυστέρηση ζεύξης ίση με 30 msec και τέλος , δημιουργούνται οι υπόλοιπες ζεύξεις οπότε προκύπτει:



ή

Και ο πηγαίος κώδικας:

```
set ns [new Simulator]

set nf [open lab2b.nam w]
$ns namtrace-all $nf

for {set i 0} {$i < 11} {incr i} {
set n($i) [$ns node]
}

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

$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

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

$ns at 10.0 "finish"
$ns run
```

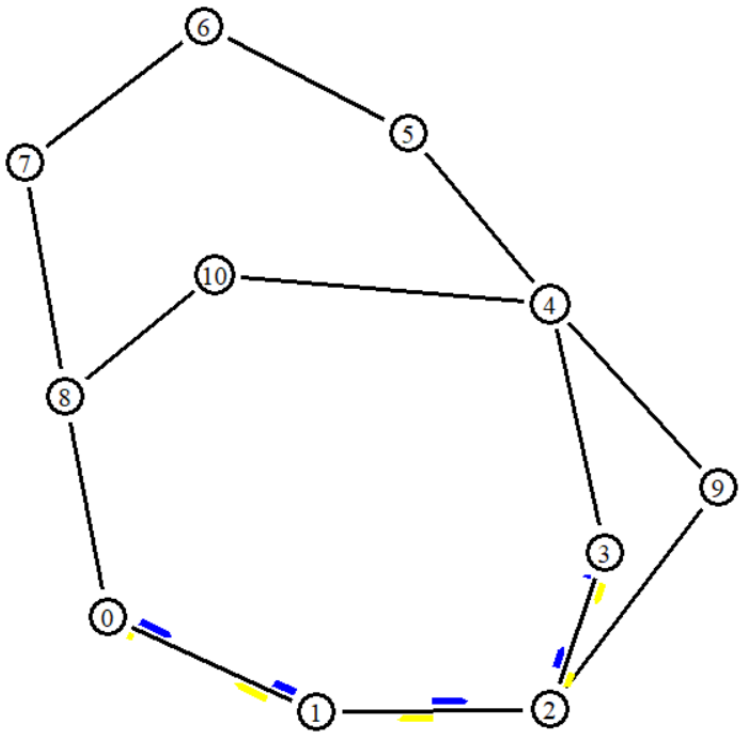
3.2. Τα γεγονότα (events)

Ο πηγαίος κώδικας:

```
set ns [new Simulator]

set nf [open lab2b.nam w]
$ns namtrace-all $nf

for {set i 0} {$i < 11} {incr i} {
set n($i) [$ns node]
}

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

$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

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

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

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

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

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

$ns at 0.5 "$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"

$ns run
```

Οπότε προκύπτει:



2.3. Ερωτήσεις

- *Ποια διαδρομή ακολουθούν τα πακέτα;*
  Τα πακέτα εκτελούν αρχικά, για το χρονικό διάστημα 0.5 - 1.1 sec, την διαδρομή 0-1-2-3. Στη συνέχεια, για 1.1 - 3.4 sec εκτελούν την διαδρομή 0-1-2-3 και 3-2-1-0. Τέλος, για 3.4 – 4 sec εκτελούν μόνο την διαδρομή 0-1-2-3.

- *Ελέγξτε αν η ροή των πακέτων και από τις δυο πλευρές ακολουθεί τη διαδρομή με τα λιγότερα βήματα.*
  Από το animation φαίνεται εύκολα πως οι δύο ροές των πακέτων ακολουθούν την ελάχιστη διαδρομή από πλευράς βημάτων, αφού χρησιμοποιούν μόνο 3 ζεύξεις, ενώ η αμέσως επόμενη ελάχιστη διαδρομή χρησιμοποιεί 5 ζεύξεις.

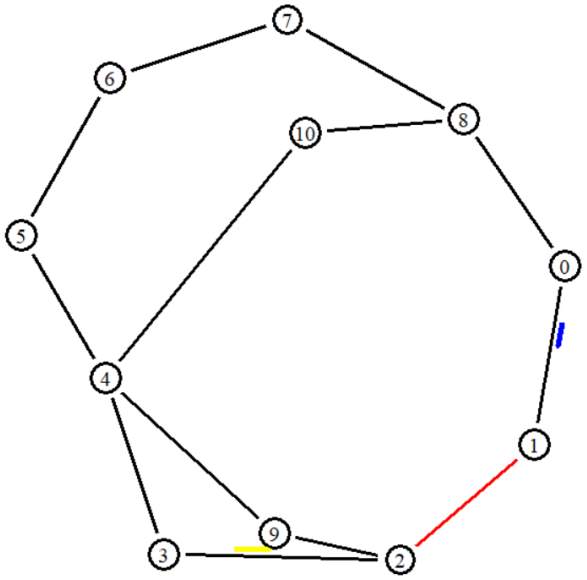
- *Υπάρχει συντομότερη διαδρομή από αυτήν που ακολουθούν, όσον αφορά τη συνολική καθυστέρηση κάθε ροής;*
  Όσον αφορά τη συνολική καθυστέρηση ροής, για να βρούμε τι συμβαίνει θα πρέπει να αθροίσουμε την καθυστέρηση κάθε ζεύξης για κάθε πιθανή διαδρομή. Έχουμε 6 πιθανές διαδρομές:
  1: 0-1-2-3 με καθυστέρηση 90 msec
  2: 0-8-10-4-3 με καθυστέρηση 100 msec
  3: 0-8-7-6-5-4-3 με καθυστέρηση 180 msec
  4:0-1-2-9-4-3 με καθυστέρηση 125 msec
  5: 0-8-10-4-9-2-3 με καθυστέρηση 135 msec
  6: 0-8-7-6-5-4-9-2-3 με καθυστέρηση 245 msec
  Άρα, δεν υπάρχει διαδρομή με μικρότερη συνολική καθυστέρηση.

- *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;*
  Οι δύο αυτές εντολές καθορίζουν το μέγιστο μέγεθος των πακέτων που μπορούν να σταλούν από τον κόμβο 0 και τον κόμβο 3 αντίστοιχα. Αν αφαιρεθούν, οι εντολές αυτές παρατηρούμε ότι αντί για ένα πακέτο των 1500 byte στέλνονται 2 πακέτα, ένα των 1000 byte και ένα των 500 byte.

3. Στατική και Δυναμική Δρομολόγηση

Στην προηγούμενη ενότητα διαπιστώθηκε ότι η κίνηση ακολουθεί τη συντομότερη διαδρομή (αυτήν με τον μικρότερο αριθμό βημάτων) από τον κόμβο “0” στον κόμβο “3” και αντίστροφα. Στη συνέχεια θα διαπιστωθεί η διαφορά μεταξύ στατικής και δυναμικής δρομολόγησης και πώς το δίκτυο αντιμετωπίζει τις μεταβολές της τοπολογίας του στην κάθε περίπτωση. Γι’ αυτόν τον λόγο θα προστεθεί ένα ενδιαφέρον χαρακτηριστικό: η διακοπή ζεύξης. Η ζεύξη μεταξύ των κόμβων “1” και “2” (που χρησιμοποιείται για τη μετάδοση) να διακοπεί όταν t = 1,8 sec και να επανέλθει όταν t = 2,5 sec. Παρ’ όλ’ αυτά, οι δυο κόμβοι συνεχίζουν να εκπέμπουν πακέτα.

Προκύπτει:

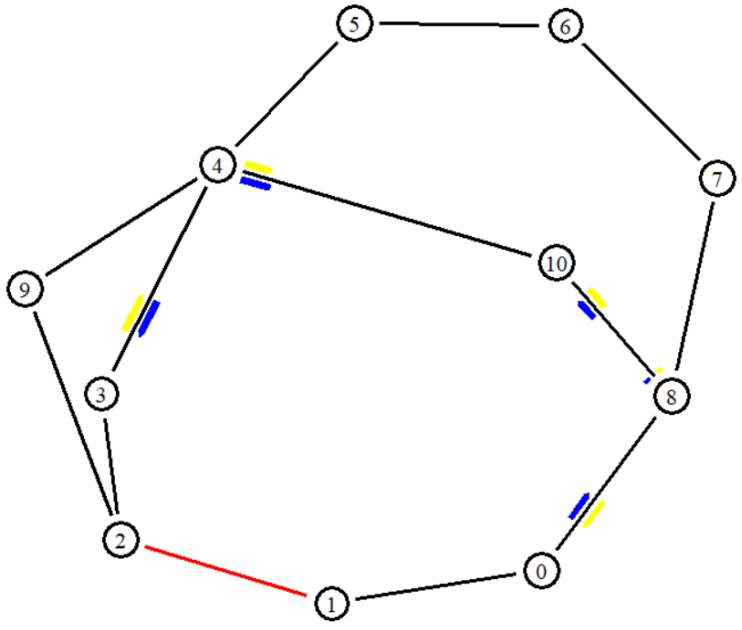


Και ο πηγαίος κώδικας:

```
set ns [new Simulator]

set nf [open lab2b.nam w]
$ns namtrace-all $nf

for {set i 0} {$i < 11} {incr i} {
set n($i) [$ns node]
}

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

$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

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

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

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

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

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

$ns at 0.5 "$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"

$ns run
```

Όταν η δρομολόγηση γίνει δυναμική, προκύπτει:



Και ο πηγαίος κώδικας:

```
set ns [new Simulator]

Agent/rtProto/Direct set preference_ 200
$ns rtproto DV

set nf [open lab2b.nam w]
$ns namtrace-all $nf

for {set i 0} {$i < 11} {incr i} {
```

```
set n($i) [$ns node]
}

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

$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

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

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

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

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

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

$ns at 0.5 "$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"

$ns run
```

3.1. Ερωτήσεις

- *Εξηγήστε γιατί, με τη στατική δρομολόγηση, οι κόμβοι εξακολουθούν να στέλνουν πακέτα και μετά τη διακοπή της ζεύξης.*
  Στη στατική δρομολόγηση, οποιεσδήποτε αλλαγές στην τοπολογία του δικτύου δεν γίνονται αντιληπτές, οπότε το δίκτυο δεν αλλάζει την κίνηση του και οι κόμβοι αποστολής συνεχίζουν να στέλνουν πακέτα κανονικά ακόμα κι αν αυτά χάνονται.

- *Τα πακέτα που χάθηκαν, θα ξαναμεταδοθούν από τους αντίστοιχους κόμβους, όταν επανέλθει η σύνδεση;*
  Όπως και παραπάνω, λόγω της στατικής δρομολόγησης, το δίκτυο δεν έχει επίγνωση για το αν τα πακέτα που στέλνονται έχουν ληφθεί ή χαθεί οπότε όταν η σύνδεση επανέλθει δεν θα αναμεταδοθούν τυχόν χαμένα πακέτα.

- *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;*
  Στην δυναμική δρομολόγηση όταν προκύψει διακοπή της ζεύξης, τα πακέτα εξακολουθούν για ένα μικρό χρονικό διάστημα (περίπου 100 msec) να χρησιμοποιούν τη διαδρομή 0-1-2-3 και να χάνονται. Στη συνέχεια, αλλάζουν διαδρομή και αρχίζουν σταδιακά να κυκλοφορούν μέσω της 0-8-10-4-3. Τέλος, με την επαναφορά της ζεύξης σε λειτουργία και αφότου περάσουν περίπου 250 msec τα πακέτα εκτελούν την αρχική διαδρομή 0-1-2-3. Η αρχική διαδρομή δεν συμπίπτει με τη μόνιμη διαδρομή δρομολόγησης κατά τη διάρκεια της διακοπής. Αυτό συμβαίνει γιατί τα πακέτα δεν αντιλαμβάνονται εξαρχής ότι η ζεύξη 1-2 έχει διακοπεί. Αρχικά χάνονται και εν συνεχεία προωθούνται από τον κόμβο 0 στον κόμβο 8, από τον κόμβο 3 στον κόμβο 4 και από τον κόμβο 2 στον κόμβο 9. Ουσιαστικά, λόγω της δυναμικής δρομολόγησης δοκιμάζονται οι εναπομείναντες πιθανές διαδρομές που μπορούν να ακολουθήσουν τα πακέτα και επιλέγεται η βέλτιστη χρονικά. Έτσι, τα πακέτα καταλήγουν να ακολουθούν τη διαδρομή 0-8-10-4-3.

- *Με βάση το animation, προσδιορίστε για κάθε ροή τη χρονική στιγμή όπου παρατηρείται η μόνιμη διαδρομή δρομολόγησης κατά τη διάρκεια διακοπής.*
  Εκτελώντας το animation παρατηρούνται τα εξής:
  Η μπλε ροή αποκαθίσταται την χρονική στιγμή 1.861 sec.
  Η κίτρινη ροή αποκαθίσταται την χρονική στιγμή 1.907 sec.

- *Για ποιο λόγο τα πακέτα ακολουθούν τις συγκεκριμένες διαδρομές αφότου πέσει η σύνδεση, στην αρχική και τη μόνιμη κατάσταση;*
  Η δυναμική δρομολόγηση κάνει τους κόμβους, όταν γίνει κάποια αλλαγή, να ψάξουν για εναλλακτική διαδρομή. Υπήρχε περίπτωση η πρώτη διαδρομή που θα βρισκόταν να μην είναι και η ελάχιστη οπότε αυτή θα άλλαζε ξανά. Στη συγκεκριμένη τοπολογία αυτό δε συνέβη καθώς η πρώτη διαδρομή που βρέθηκε έτυχε να είναι και η ελάχιστη. Οπότε, στη γενική περίπτωση, μπορεί τα πακέτα να ακολουθήσουν αρκετές διαδρομές, έτσι ώστε να μην χάνονται, αλλά τελικά θα ακολουθήσουν την ελάχιστη, όταν αυτή βρεθεί. Τα πακέτα ακολουθούν τη διαδρομή 0-8-10-4-3 γιατί αφότου αποκοπεί η ζεύξη 1-2 είναι η πιο γρήγορη διαδρομή. (4 βήματα - 100 msec).

- *Θα μπορούσαν να δρομολογηθούν από άλλους κόμβους;*
  Υπάρχουν άλλες δύο διαδρομές που μπορούν να ακολουθήσουν τα πακέτα και είναι οι εξής: 0-8-10-4-9-2-3 (6 βήματα - 135 msec) και 0-8-7-6-5-4-3 (6 βήματα - 180 msec).

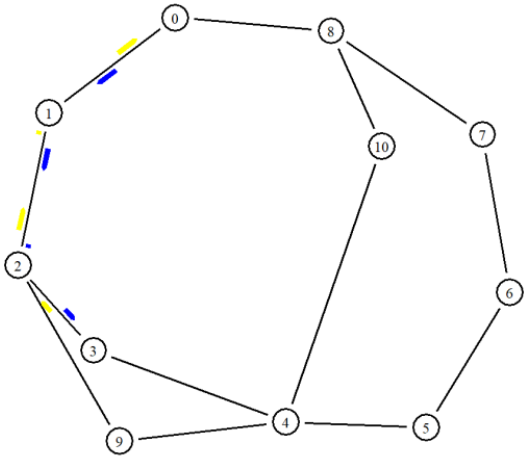
- *Ποιος από όλους τους κόμβους καθορίζει από ποια διαδρομή θα προωθηθούν κάθε φορά τα πακέτα;*
  Στην γενική περίπτωση η διαδρομή καθορίζεται από τον δρομολογητή κόμβο που βρίσκεται αμέσως πριν από την διακοπτόμενη ζεύξη γιατί είναι αυτός στον οποίο ήδη έχουν αποσταλεί κάποια πακέτα. Όταν διακοπεί η σύνδεση παρατηρείται ότι οι κόμβοι 1 και 2 στέλνουν πακέτα των 11 byte με το όνομα rtProtoDV στους κόμβους 0 και 3 αντίστοιχα, οι οποίοι με τη σειρά τους τα στέλνουν προς όλες τι πιθανές διαδρομές. Αφού δοκιμαστούν όλες οι διαδρομές επιλέγεται η πιο σύντομη (λιγότεροι κόμβοι) και επαναπροσδιορίζεται η νέα διαδρομή.

4. Καθορισμός κόστους ζεύξης

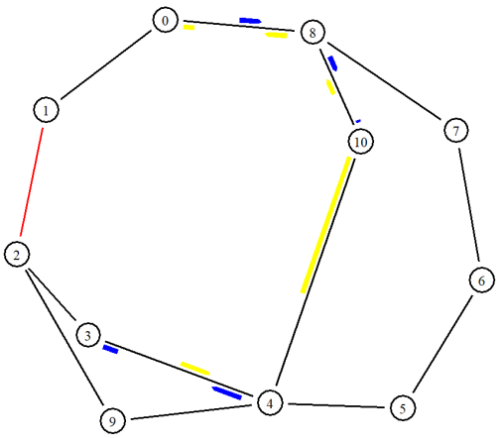
Έως αυτό το σημείο λαμβανόταν υπόψη ότι η προτιμώμενη δρομολόγηση των πακέτων είναι από εκείνη τη διαδρομή απ' όπου η ροή θα περάσει από τους λιγότερους κόμβους. Αυτό ισχύει διότι στην περίπτωση της προσομοίωσής θεωρείται, εξ' ορισμού, ότι όλες οι ζεύξεις έχουν κόστος ίσο με μια (1) μονάδα. Στην πραγματικότητα όμως, η διαδρομή που ακολουθείται σε κάθε περίπτωση βασίζεται στο κόστος των ζεύξεων μεταξύ των κόμβων. Θεωρείστε λοιπόν ότι το κόστος μιας ζεύξης είναι ανάλογο της καθυστέρησής της, υποθέτοντας κόστος ίσο με d/10 στην περίπτωση καθυστέρησης d msec.
Γι' αυτό εισάγεται στον κώδικα: `$ns cost $n(x) $n(y) z`, 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).

4.1. Ερωτήσεις

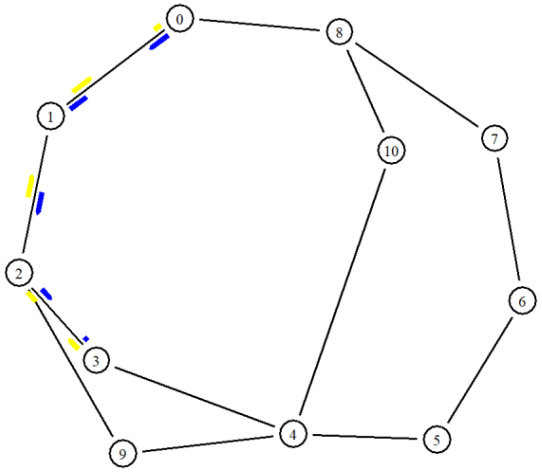
- *Ποιες διαδρομές ακολουθούν τα πακέτα πριν, κατά τη διάρκεια και μετά την πτώση της σύνδεσης για τις δύο ροές;*
  Τα πακέτα επιλέγουν να ακολουθήσουν τις διαδρομές με το λιγότερο κόστος. Αυτό σημαίνει, ότι πριν τη διακοπή ακολουθούν τη διαδρομή 0-1-2-3 (κόστος 9), κατά τη διάρκεια επιλέγουν τη διαδρομή 0-8-10-4-3 (κόστος 10) και μετά την διακοπή πάλι την αρχική διαδρομή 0-1-2-3.



Κίνηση πριν την πτώση ζεύξης



Κίνηση κατά τη διάρκεια της πτώσης ζεύξης

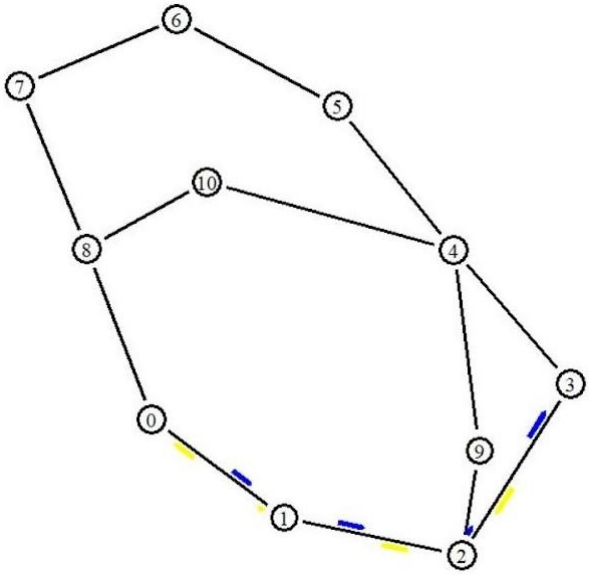


Κίνηση μετά την επαναφορά της ζεύξης

- *Για ποιον λόγο τα πακέτα ακολουθούν τις συγκεκριμένες διαδρομές;*
  Τα πακέτα έχουν ως πρώτο κριτήριο επιλογής το κόστος της διαδρομής και ως δεύτερο κριτήριο την ταχύτητα της διαδρομής. Στη συγκεκριμένη περίπτωση το κόστος είναι ανάλογο της ταχύτητας και για αυτό τα παραπάνω κριτήρια ταυτίζονται.

- *Θα μπορούσαν να δρομολογηθούν από άλλους κόμβους;*
  Εφόσον ο καθορισμός των διαδρομών γίνεται με βάση το κόστος και επειδή σε κάθε μία από τις τρεις περιπτώσεις υπάρχει μοναδική διαδρομή με ελάχιστο κόστος, συμπεραίνει κανείς ότι τα πακέτα δεν 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.*
  Η μόνιμη διαδρομή για τη μπλε ροή παρατηρείται τη χρονική στιγμή t = 2.773 sec, ενώ για τη κίτρινη ροή παρατηρείται την t = 2.799 sec.



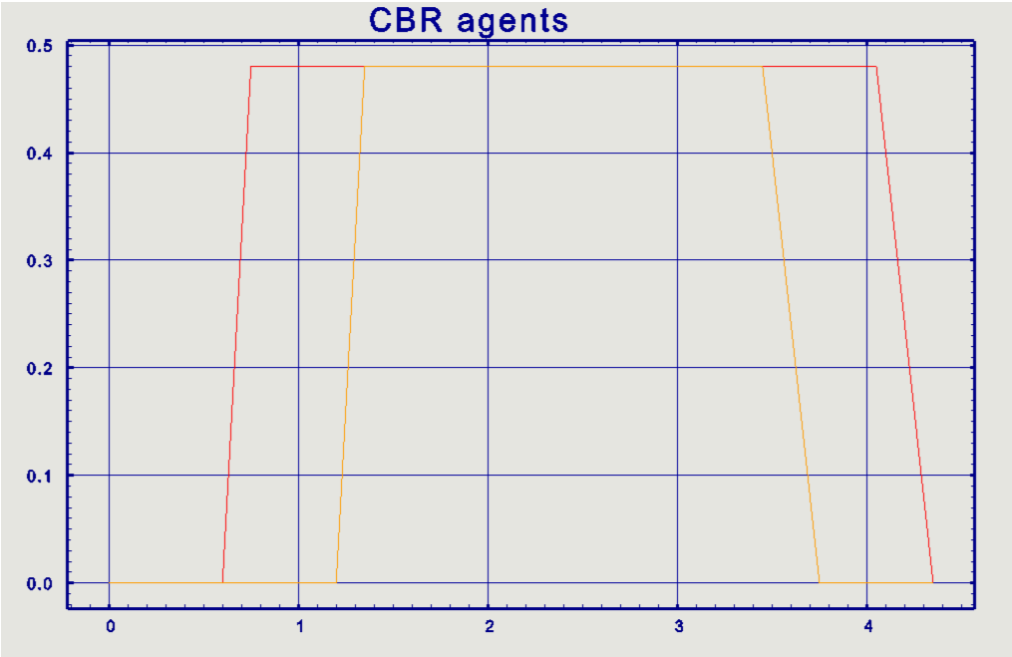
- *Ποιος είναι ο ρόλος της εντολής Agent/rtProto/Direct set preference_ 200; Τι παρατηρείτε στη δρομολόγηση των πακέτων αν αφαιρεθεί η εντολή αυτή από τον κώδικα της προσομοίωσης; Αιτιολογείστε γιατί συμβαίνει αυτό.*
  Η δυναμική δρομολόγηση έχει τη δυνατότητα να χρησιμοποιεί διάφορες στρατηγικές δρομολόγησης. Με την εντολή set preference_ ορίζεται το κατά πόσο είναι επιθυμητό να χρησιμοποιηθεί η κάθε στρατηγική. Όταν αφαιρεθεί η παραπάνω εντολή, οι αλλαγές που απαιτούνται στις διαδρομές που ακολουθούν τα πακέτα γίνονται ακαριαία. Έτσι, όταν διακόπτεται και όταν επανέρχεται η ζεύξη έχουμε άμεση αλλαγή διαδρομής. Άρα, ο ρόλος της εντολής είναι να καθυστερεί την μετάβαση αυτή ώστε να γίνεται πιο ομαλή. Το εύρος των τιμών που μπορούμε να χρησιμοποιήσουμε ως όρισμα είναι 0−255, με το 0 να θεωρείται ότι η στρατηγική χρησιμοποιείται σε πλήρη βαθμό ενώ με το 255 το αντίθετο. Έχοντας την παραπάνω εντολή στο script και παρατηρώντας ότι το 200 είναι πολύ κοντά στο 255, η στρατηγική Direct δεν είναι αυτή που χρησιμοποιείται κατά κύριο λόγο, ενώ αφαιρώντας τη φαίνεται πως αυτό αλλάζει. Έτσι παρατηρούμε ότι ενώ πριν η μόνιμη διαδρομή για τις δύο ροές παρατηρήθηκε στα 2,79 sec, πλέον παρατηρείται στα 2,6 sec.

5. Παρακολούθηση εκθετικής κίνησης με το Xgraph

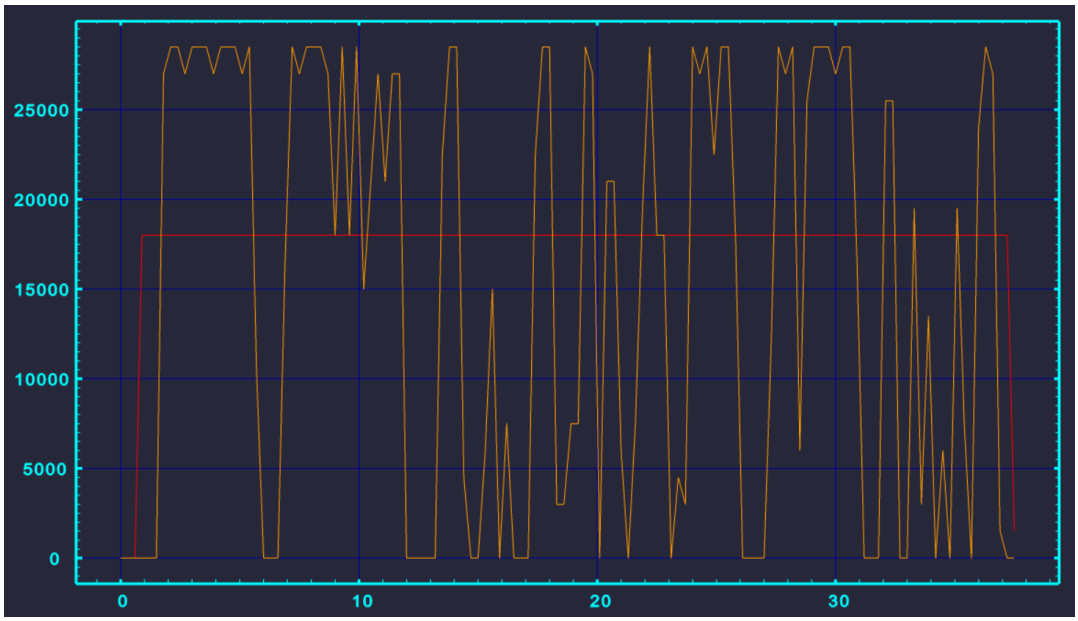
Στην ενότητα αυτή θα χρησιμοποιηθεί το “Xgraph” για να δημιουργηθούν γραφικές παραστάσεις της κίνησης στο δίκτυο που ήδη μελετάται.

5.1. Ερωτήσεις

- *Ποιος είναι ο μέγιστος ρυθμός μετάδοσης που επιτυγχάνεται για τις δύο περιπτώσεις κίνησης, βάσει των γραφικών παραστάσεων που σχεδιάσατε;*
  Με αλλαγή του set time σε 0,3 από 0,015 προκύπτουν τα εξής:
  Για την περίπτωση όπου η κίνηση και των δύο αποστολέων είναι γραμμική προκύπτει για την κίνηση cbr0 μέγιστος ρυθμός 0.48 Mbit/sec (κόκκινο) και για την cbr3 0.48 Mbit/sec (μπλε).



Για την περίπτωση όπου η κίνηση του ενός αποστολέα είναι εκθετική προκύπτει για την κίνηση cbr μέγιστος ρυθμός 0.48 Mbit/sec (κόκκινο) και για την exp 0.8 Mbit/sec (μπλε).



- *Αιτιολογείστε τις μέγιστες τιμές που προσδιορίσατε παραπάνω, χρησιμοποιώντας τις παραμέτρους που θέσατε για τη διαμόρφωση των δύο πηγών κίνησης (CBR και Exponential).*
  Για την πρώτη περίπτωση και οι δύο πηγές στέλνουν 1500 byte κάθε 0.025 sec άρα αυτό αντιστοιχεί σε 12000 bit κάθε 0.025 sec. Άρα, ο ρυθμός μετάδοσης είναι 0.48 Mbit/sec Για την δεύτερη περίπτωση η cbr έχει ρυθμό 0.48 Mbit/sec, ενώ η exp έχει ρυθμό 0.75 Mbit/sec που καθορίζεται από την εντολή $exp3 set rate_ 750k. Άρα, παρατηρεί κανείς ότι για τις γραμμικές πηγές ο μέγιστος ρυθμός δεν ξεπερνά το ρυθμό που υπολογίσθηκε, ενώ για την εκθετική πηγή συμβαίνει το αντίθετο. Αυτό βέβαια οφείλεται στη περίοδο δειγματοληψίας της record (αν αυξηθεί ο μέγιστος ρυθμός σταθεροποιείται στα 0.75 Mbit/sec).

- *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)sec, όπου a τα δύο τελευταία ψηφία του αριθμού μητρώου σας.*
  t = 37,6 sec
  Για τις γραμμικές πηγές (άρα και ροές) ισχύει ότι σε χρόνο t στέλνουν επιτυχώς στον προορισμό 37,6 * 0,48 Mbit = 2,256 MByte. Για την εκθετική ροή μορφοποιούμε λίγο την διαδικασία record ως εξής:

```
proc record {} {

global sink0 sink3 f0 f3

set ns [Simulator instance]

set time 0.3

set bw0 [$sink3 set bytes_]

set bw3 [$sink0 set bytes_]
```

```
set now [$ns now]

puts $f0 "$now [expr ($bw0)]"

puts $f3 "$now [expr ($bw3)]"

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

}
```

Έτσι, βρίσκουμε ότι σε χρόνο 37,6 sec λαμβάνονται επιτυχώς στον προορισμό 2,2 MByte.

Και ο πηγαίος κώδικας:

```
set ns [new Simulator]

Agent/rtProto/Direct set
preference_ 200
$ns rtproto DV

set nf [open lab2b.nam w]
$ns namtrace-all $nf

set f0 [open out0.tr w]
set f3 [open out3.tr w]

for {set i 0} {$i < 11} {incr i} {
set n($i) [$ns node]
}

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

$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

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

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

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

set cbr0 [new
Application/Traffic/CBR]
$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.025
$cbr0 attach-agent $udp0

set exp3 [new
Application/Traffic/Exponential]
$exp3 set packetSize_ 1500
$exp3 set interval_ 0.025
$exp3 set rate_ 750k
$exp3 attach-agent $udp3

proc record {} {
global sink0 sink3 f0 f3
set ns [Simulator instance]
set time 0.3
set bw0 [$sink3 set bytes_]
set bw3 [$sink0 set bytes_]
set now [$ns now]
puts $f0 "$now [expr ($bw0)]"
puts $f3 "$now [expr ($bw3)]"
$sink0 set bytes_ 0
$sink3 set bytes_ 0
$ns at [expr $now+$time] "record"
}

proc finish {} {
global ns nf f0 f3
$ns flush-trace
close $nf
close $f0
close $f3
exit 0
}

$ns at 0.0 "record"
$ns at 0.5 "$cbr0 start"
$ns at 1.1 "$exp3 start"
$ns at 36.5 "$exp3 stop"
$ns at 37.1 "$cbr0 stop"
$ns at 37.6 "finish"

$ns run
```