Δίκτυα Επικοινωνιών

2014-2015

9η εργαστηριακή άσκηση:

Επαναληπτικά Θέματα στη Μελέτη Πρωτοκόλλων με το NS2

Εξάμηνο : 6ο

Ονοματεπώνυμο : Δασούλας Γεώργιος
Α.Μ. :03112010

Περιεχόμενο

Στη συγκεκριμένη και τελευταία εργαστηριακή άσκηση , μελετώνται επαναληπτικά θέματα με τη χρήση του προσομοιωτή NS2. Αρχικά, δημιουργούμε την παρακάτω τοπολογία, της οποίας ο κώδικας σε tcl αρχείο δίνεται στην εκφώνηση.

(α) Να σχεδιάσετε την τοπολογία της προσομοίωσης, ώστε να φαίνονται τα ονόματα των κόμβων, οι ταχύτητες των μεταξύ τους ζεύξεων και το μέγεθος ουράς σε κάθε ζεύξη. Να αναφέρετε τις εντολές με τις οποίες ορίζονται τα παραπάνω.

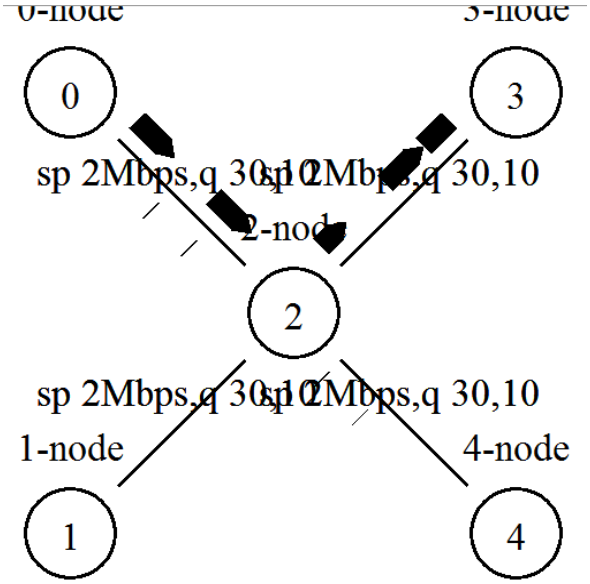
Για την εμφάνιση ονομάτων και σχολίων πάνω στους κόμβους και στις ζεύξεις χρησιμοποιούμε τις παρακάτω εντολές :

```
$n($i) label "$i-node"
```

```
$ns duplex-link-op $n($i) $n(2) label "sp 2Mbps,q 30,10 "
```

για τα ονόματα των κόμβων και τα στοιχεία της ζεύξης

Στο παρακάτω σχήμα φαίνεται η τοπολογία μαζί με τις ετικέτες , αν και όχι τόσο ευκρινώς :



(β) Τροποποιήστε τον κώδικα της προσομοίωσης, ώστε η τοπολογία να φαίνεται στο NAM όπως απεικονίζεται στο Σχήμα 1.

Για τη σωστή απεικόνιση της τοπολογίας όπως ζητείται πρέπει να προσθέσουμε τις παρακάτω :

```
$ns duplex-link-op $n(2) $n(0) orient left-up
$ns duplex-link-op $n(2) $n(3) orient right-up
$ns duplex-link-op $n(2) $n(4) orient right-down
$ns duplex-link-op $n(2) $n(1) orient left-down
```

Η σωστή απεικόνιση με τους κόμβους στη σωστή θέση φαίνεται επίσης στο προηγούμενο σχήμα.

(γ) Να αναφέρετε ποιες εφαρμογές τρέχουν σε κάθε κόμβο και δημιουργούν δεδομένα προς μετάδοση. Ποιες είναι οι σχετικές εντολές;

Για τον κόμβο 0 έχουμε τις εφαρμογές CBR και FTP και για τον κόμβο 1 έχουμε τις εφαρμογές Exponential και TELNET. Οι εντολές που καθορίζουν αυτές τις εφαρμογές στο στρώμα μεταφοράς είναι οι παρακάτω :

```
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

set exp3 [new Application/Traffic/Exponential]
$exp3 attach-agent $udp3
```

```
set telnet4 [new Application/Telnet]
$telnet4 attach-agent $tcp4
```

(δ) Τι πρωτόκολλο στο επίπεδο μεταφοράς χρησιμοποιείται για την κίνηση κάθε εφαρμογής;

Το πρωτόκολλο UDP χρησιμοποιείται για τη CBR , καθώς και την εκθετική (Exponential) εφαρμογή - γεννήτρια κίνησης . Το πρωτόκολλο TCP χρησιμοποιείται για την FTP και την Telnet εφαρμογή .

(ε) Πού κατευθύνεται η κίνηση κάθε εφαρμογής; Με ποιες εντολές του κώδικα ορίζεται;

Οι κίνησεις της κάθε εφαρμογής ορίζονται με βάση τις συνδέσεις των udp και tcp agents με τα αντίστοιχα null και sink agents. Συγκεκριμένα , οι παρακάτω εντολές :

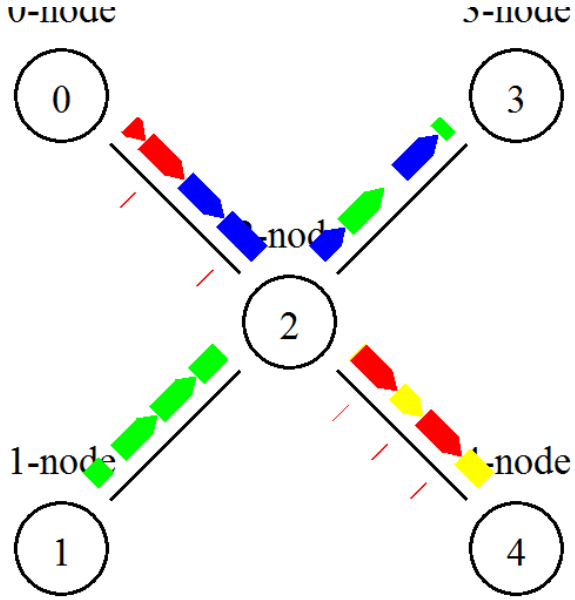
```
$ns connect $udp1 $null1
$ns connect $tcp2 $sink2
$ns connect $udp3 $null3
$ns connect $tcp4 $sink4
```

αφού έχουν γίνει attach οι αντίστοιχοι agents στους αντίστοιχους κόμβους ορίζουν τις εξής κινήσεις :
CBR κίνηση από τον κόμβο 0 στον κόμβο 3
FTP κίνηση από τον κόμβο 0 στον κόμβο 4
Exponential κίνηση από τον κόμβο 1 στον κόμβο 3
Telnet κίνηση από τον κόμβο 1 στον κόμβο 4

(στ) Χρωματίστε τις τέσσερις ροές δεδομένων, ώστε να διακρίνονται στο NAM. Παραθέστε ένα σχετικό screenshot

Με τις παρακάτω εντολές στην αρχή του tcl αρχείου ορίζουμε το χρώματα της κάθε ροής

```
$ns color 1 blue
$ns color 2 red
$ns color 3 green
$ns color 4 yellow
```



(Στο παραπάνω σχήμα αφαιρέσαμε τις ετικέτες από τις ζεύξεις που είχαμε τοποθετήσει στο πρώτο ερώτημα για να είναι πιο ευκρινές το σχήμα)

(ζ) Με βάση τα περιεχόμενα του αρχείου ίχνους (lab9.tr) και με τη βοήθεια κατάλληλου script σε γλώσσα awk, υπολογίστε τα ακόλουθα για κάθε μία από τις τέσσερις ροές δεδομένων:
1. το χρόνο μετάδοσης πακέτων,
2. το συνολικό πλήθος των πακέτων και τον όγκο δεδομένων (byte) που αποστέλλονται από τους κόμβους πηγής n(0) και n(1) (εξαιρώντας τις αναμεταδόσεις και τις αποστολές επιβεβαιώσεων),
3. το συνολικό πλήθος των πακέτων και τον όγκο δεδομένων που χάνονται,
4. το συνολικό πλήθος των πακέτων και τον όγκο δεδομένων που λαμβάνονται από τους κόμβους προορισμού n(3) και n(4) (εξαιρώντας τις λήψεις επιβεβαιώσεων). Σχολιάστε τα αποτελέσματά σας.

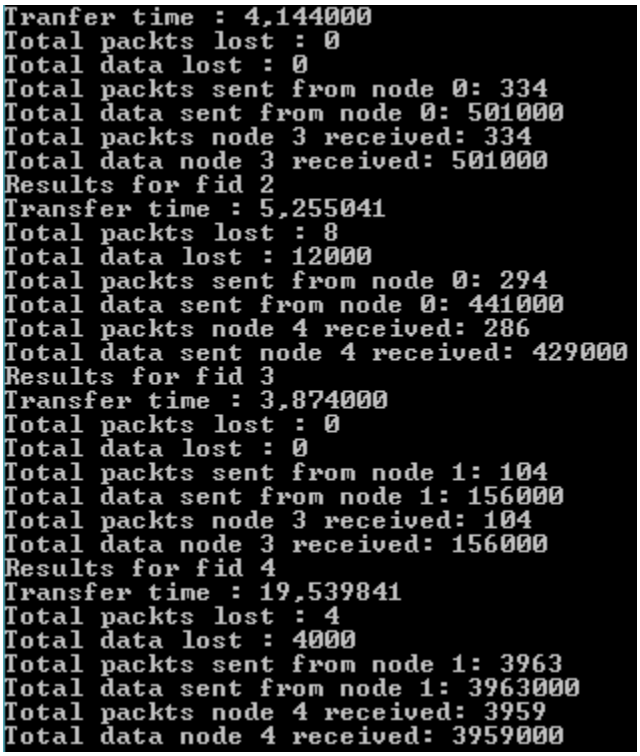
Το αρχείο awk το οποίο εκτελέστηκε είναι το εξής :

```
BEGIN {
    # Do we need to fix the decimal mark?
    if (sprintf(sqrt(2)) ~ /,/) dmfix = 1;
    droppedpackets[1]=0;
    droppedpackets[2]=0;
    droppedpackets[3]=0;
    droppedpackets[4]=0;
    droppeddata[1]=0;
    droppeddata[2]=0;
    droppeddata[3]=0;
    droppeddata[4]=0;
    packtsstartfrom[1]=0;
    packtsstartfrom[2]=0;
    packtsstartfrom[3]=0;
```

```
    packtsstartfrom[4]=0;
    datastartfrom[1]=0;
    datastartfrom[2]=0;
    datastartfrom[3]=0;
    datastartfrom[4]=0;
}
{
# Apply dm fix if needed
if (dmfix) sub(/\./, ",", $0);
}
/^r/&&(/cbr/||/exp/||/tcp/){
    lasttransfer[$8]=$2;
    if(($6 > 40) && ($4 == 3))
    {
        pnode3[$8]++;
        dnode3[$8]+=$6;
    }
    if(($4 == 4) && ($6 > 40))
    {
        pnode4[$8]++;
        dnode4[$8]+=$6;
    }
    }
/^d/&&(/cbr/||/exp/||/tcp/){
    droppedpackets[$8]++;
    droppeddata[$8]+= $6;
    }
/^\+/&&(/cbr/||/exp/||/tcp/){
    if($6 >40)
    {
        if($3 == 0 || $3 ==1){
        packtsstartfrom[$8]++;
        datastartfrom[$8]+=$6;}
    }
}
END{
    printf("Results for fid 1\n");
    printf("Tranfer time : %f\n",(lasttransfer[1] -0.15));
    printf("Total packts lost : %d\n",droppedpackets[1]);
    printf("Total data lost : %d\n",droppeddata[1]);
    printf("Total packts sent from node 0: %d\n" ,packtsstartfrom[1]);
    printf("Total data sent from node 0: %d\n" ,datastartfrom[1]);
```

```
    printf("Total packts node 3 received: %d\n" ,pnode3[1]);
    printf("Total data node 3 received: %d\n" ,dnode3[1]);
    printf("Results for fid 2\n");
    printf("Transfer time : %f\n",(lasttransfer[2] -0.3));
    printf("Total packts lost : %d\n",droppedpackets[2]);
    printf("Total data lost : %d\n",droppeddata[2]);
    printf("Total packts sent from node 0: %d\n" ,packtsstartfrom[2]);
    printf("Total data sent from node 0: %d\n" ,datastartfrom[2]);
    printf("Total packts node 4 received: %d\n" ,pnode4[2]);
    printf("Total data sent node 4 received: %d\n" ,dnode4[2]);
    printf("Results for fid 3\n");
    printf("Transfer time : %f\n",(lasttransfer[3] -0.45));
    printf("Total packts lost : %d\n",droppedpackets[3]);
    printf("Total data lost : %d\n",droppeddata[3]);
    printf("Total packts sent from node 1: %d\n" ,packtsstartfrom[3]);
    printf("Total data sent from node 1: %d\n" ,datastartfrom[3]);
    printf("Total packts node 3 received: %d\n" ,pnode3[3]);
    printf("Total data node 3 received: %d\n" ,dnode3[3]);
    printf("Results for fid 4\n");
    printf("Transfer time : %f\n",(lasttransfer[4] -0.6));
    printf("Total packts lost : %d\n",droppedpackets[4]);
    printf("Total data lost : %d\n",droppeddata[4]);
    printf("Total packts sent from node 1: %d\n" ,packtsstartfrom[4]);
    printf("Total data sent from node 1: %d\n" ,datastartfrom[4]);
    printf("Total packts node 4 received: %d\n" ,pnode4[4]);
    printf("Total data node 4 received: %d\n" ,dnode4[4]);
    }
```

Τα αποτελέσματα που πήραμε ήταν τα παρακάτω :



(η) Πόσο είναι το ελάχιστο μέγεθος της ουράς αναμονής της ζεύξης μεταξύ των κόμβων n(2) και n(4), ώστε να μην παρατηρούνται απώλειες πακέτων; Περιγράψτε τη μεθοδολογία εντοπισμού του.

Το ελάχιστο μέγεθος της ουράς αναμονής το βρίσκουμε με δοκιμές πάνω στον κώδικα, ώστε να βρούμε το μικρότερο queuelimit που δεν παρατηρούμε απορριφθέτα πακέτα στα αποτελέσματα του αρχείου awk . Συγκεκριμένα αλλάζουμε τις εντολές μόνο για την περίπτωση των κόμβων 2 και 4 :

```
$ns queue-limit $n(2) $n(4) queuesize
```

```
 $ns queue-limit $n(4) $n(2) queuesize
```

Βρίσκουμε δοκιμαστικά ότι για queuesize 20 έχουμε την πρώτη απώλεια πακέτων , οπότε το ελάχιστο μέγεθος της ουράς αναμονής είναι 21.

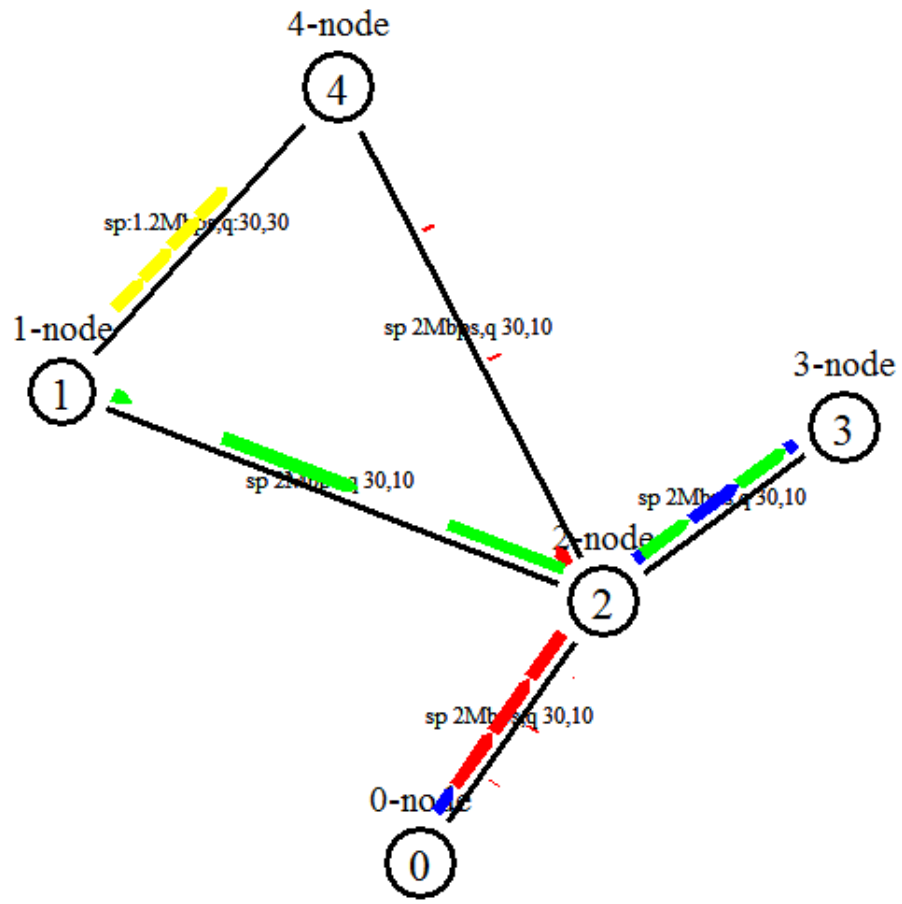
(θ) Τροποποιήστε τον κώδικα προσομοίωσης του ερωτήματος (ζ), ώστε να δημιουργήσετε μια αμφίδρομη ζεύξη μεταξύ των κόμβων n(1) και n(4) με τα ακόλουθα χαρακτηριστικά: (i) εύρος ζώνης 1.2Mbps, (ii) ουρά τύπου DropTail, (iii) καθυστέρηση 50ms και (iv) μέγεθος ουράς αναμονής ίσο με 30. Τι παρατηρείτε ως προς τις διαδρομές που ακολουθούν τώρα οι ροές δεδομένων σε σχέση με τις διαδρομές που ακολουθούνται στην αρχική τοπολογία; Παραθέστε ένα σχετικό screenshot. Υπολογίστε το πλήθος των πακέτων που χάνονται σε κάθε κόμβο για κάθε μία από τις ροές δεδομένων.

Για τη διαμόρφωση της ζητούμενης ζεύξης αρκεί να προσθέσουμε στο tcl αρχείο τις παρακάτω εντολές :

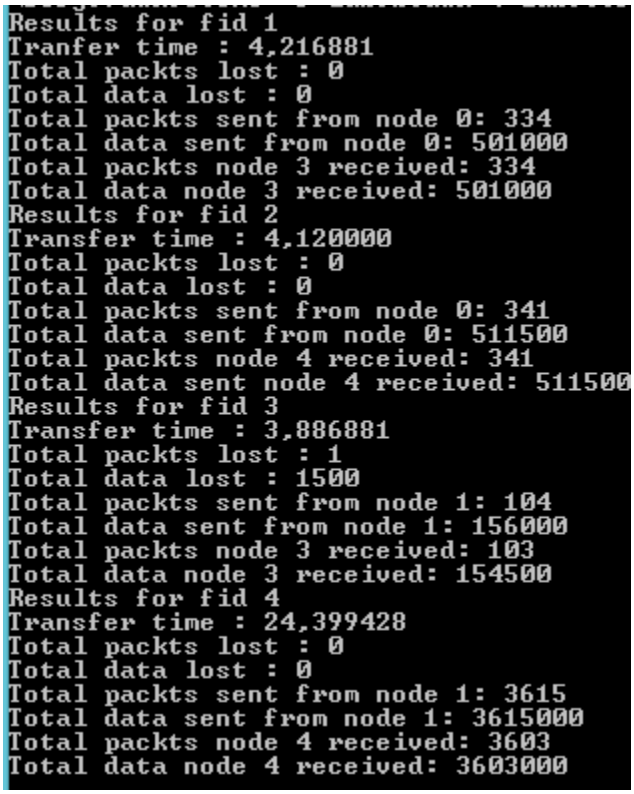
```
$ns duplex-link $n(1) $n(4) 1.2Mb 50ms DropTail
```

```
$ns queue-limit $n(1) $n(4) 30
```

```
$ns queue-limit $n(4) $n(1) 30
$ns duplex-link-op $n(1) $n(4) label "sp:1.2Mbps,q:30,30 "
```



Όπως φαίνεται από το σχήμα , η μόνη ροή που επηρεάζεται είναι η **1 -> 4**, η οποία πρίν ήταν **1 -> 2 -> 4** , ενώ τώρα είναι **1 -> 4**.
Επίσης ,χρησιμοποιώντας το ίδιο αρχείο awk πήραμε τα παρακάτω αποτελέσματα :



Παρατήρηση:

Βλέπουμε ότι έχουμε μείωση των πακέτων που χάθηκαν , λόγω της μείωσης της συμφόρησης στις άλλες ζεύξεις μέσω της δημιουργίας της ζεύξης **1 -> 4**.

(ι) Τροποποιήστε τον κώδικα του προηγούμενου ερωτήματος ώστε οι διαδρομές που ακολουθούν οι τέσσερις ροές δεδομένων στη νέα τοπολογία να είναι ίδιες με τις αρχικές. Χρησιμοποιήστε την έννοια του κόστους ζεύξης για να το επιτύχετε αυτό, θέτοντας το ανάλογο της καθυστέρησης της εκάστοτε ζεύξης (π.χ. 10 msec 1 μονάδα κόστους). Προσοχή: Σε περίπτωση ροής TCP, θα πρέπει τόσο τα πακέτα δεδομένων όσο και οι επιβεβαιώσεις τους να κινούνται στην ίδια διαδρομή (προφανώς με αντίθετες φορές). Παραθέστε ένα σχετικό screenshot

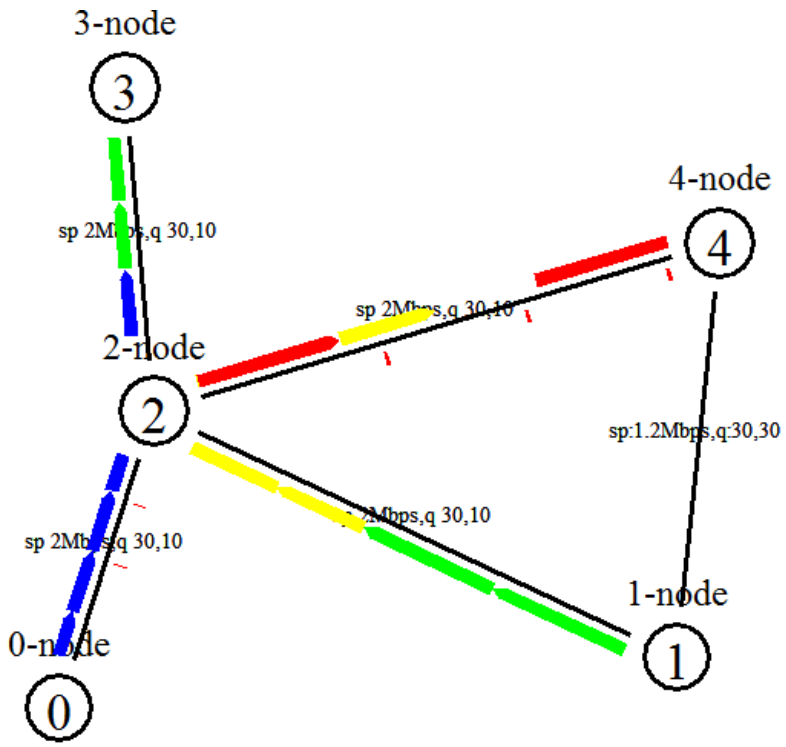
Παρακάτω βρίσκεται ο κώδικας tcl αλλαγμένος (προσθέσαμε τις εντολές για το κόστος) :

```
set ns [new Simulator]
$ns color 1 blue
$ns color 2 red
$ns color 3 green
$ns color 4 yellow
set tf [open lab9.tr w]
$ns trace-all $tf
set nf [open lab9.nam w]
$ns namtrace-all $nf
proc finish {} {
 global ns tf nf
 $ns flush-trace
 close $tf
 close $nf
 exit 0
```

```
}
for {set i 0} {$i < 5} {incr i} {
 set n($i) [$ns node]
 $n($i) label "$i-node"
}
for {set i 0} {$i < 5} {incr i} {
 if {$i != 2} {
 $ns duplex-link $n($i) $n(2) 2Mb 20ms DropTail
 $ns queue-limit $n($i) $n(2) 30
 $ns queue-limit $n(2) $n($i) 10
 $ns duplex-link-op $n($i) $n(2) label "sp 2Mbps,q 30,10 "
 $ns cost $n($i) $n(2) 2
 $ns cost $n(2) $n($i) 2
 }
}
$ns duplex-link $n(1) $n(4) 1.2Mb 50ms DropTail
$ns queue-limit $n(1) $n(4) 30
$ns queue-limit $n(4) $n(1) 30
$ns duplex-link-op $n(1) $n(4) label "sp:1.2Mbps,q:30,30 "
$ns cost $n(1) $n(4) 5
$ns cost $n(4) $n(1) 5
set udp1 [new Agent/UDP]
$udp1 set fid_ 1
$udp1 set packetSize_ 1500
$ns attach-agent $n(0) $udp1
set null1 [new Agent/Null]
$ns attach-agent $n(3) $null1
$ns connect $udp1 $null1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1500
$cbr1 set rate_ 1.0mb
$cbr1 set random_ off
set tcp2 [new Agent/TCP]
$tcp2 set packetSize_ 1460
$tcp2 set fid_ 2
$ns attach-agent $n(0) $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink2
$ns connect $tcp2 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set udp3 [new Agent/UDP]
$udp3 set fid_ 3
$udp3 set packetSize_ 1500
$ns attach-agent $n(1) $udp3
set null3 [new Agent/Null]
$ns attach-agent $n(3) $null3
$ns connect $udp3 $null3
```

```
set exp3 [new Application/Traffic/Exponential]
$exp3 attach-agent $udp3
$exp3 set packetSize_ 1500
$exp3 set rate_ 1.2mb
set tcp4 [new Agent/TCP]
$tcp4 set packetSize_ 960
$tcp4 set fid_ 4
$ns attach-agent $n(1) $tcp4
set sink4 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink4
$ns connect $tcp4 $sink4
set telnet4 [new Application/Telnet]
$telnet4 attach-agent $tcp4
$telnet4 set interval_ 0.001
# Events
$ns at 0.15 "$cbr1 start"
$ns at 0.3 "$ftp2 start"
$ns at 0.45 "$exp3 start"
$ns at 0.6 "$telnet4 start"
$ns at 4.15 "$cbr1 stop"
$ns at 4.3 "$ftp2 stop"
$ns at 4.45 "$exp3 stop"
$ns at 4.6 "$telnet4 stop"
$ns at 25.0 "finish"
$ns run
```

Στο παρακάτω screenshot φαίνονται όλες οι κινήσεις οι οποίες είναι ίδιες με τις αρχικές:



Χρησιμοποιήσαμε αυτά τα κόστη , όπως φαίνονται στον κώδικα για να επιβαρύνουμε περισσότερο τη ζεύξη **1 -> 4,** ώστε να προτιμηθεί η άλλη διαδρομή. Στην ουσία πήραμε ως κόστος κάθε ζεύξης το 1/10 της καθυστέρησής της.

(ια) Τροποποιήστε τον κώδικα του προηγούμενου ερωτήματος ώστε να συμβαίνει διακοπή της ζεύξης μεταξύ των κόμβων n(2) και n(4) μεταξύ των χρονικών στιγμών 1.5 και 3.0 sec. Πόσα πακέτα χάνονται στην περίπτωση αυτή για κάθε μία από τις ροές δεδομένων; Σε ποια χρονική στιγμή μετά την επαναφορά της ζεύξης συνεχίζουν οι ροές δεδομένων από τους κόμβους n(0) και n(1) στον n(4);

Για τη διακοπή της ζεύξης προσθέσαμε στο τέλος του αρχείου tcl τις παρακάτω εντολές :

```
$ns rtmodel-at 1.5 down $n(2) $n(4)
$ns rtmodel-at 3.0 up $n(2) $n(4)
```

```
Results for fid 1
Tranfer time : 4,096881
Total packts lost : 0
Total data lost : 0
Total packts sent from node 0: 334
Total data sent from node 0: 501000
Total packts node 3 received: 334
Total data node 3 received: 501000
Results for fid 2
Transfer time : 4,394960
Total packts lost : 5
Total data lost : 7500
Total packts sent from node 0: 88
Total data sent from node 0: 132000
Total packts node 4 received: 82
Total data sent node 4 received: 123000
Results for fid 3
Transfer time : 3,850881
Total packts lost : 0
Total data lost : 0
Total packts sent from node 1: 104
Total data sent from node 1: 156000
Total packts node 3 received: 104
Total data node 3 received: 156000
Results for fid 4
Transfer time : 22,077920
Total packts lost : 6
Total data lost : 6000
Total packts sent from node 1: 3966
Total data sent from node 1: 3966000
Total packts node 4 received: 3959
Total data node 4 received: 3959000
```

Η ροή 2 ξεκινάει περίπου τη χρονική στιγμή 3.91 ενώ η ροή 4 ξεκινάει τη χρονική στιγμή 4.67. Αυτά αναφέρονται στις αρχικές διαδρομές μετά την επαναφορά της ζεύξης. Για κάποιο λόγο όταν κάνουμε την προσομοίωση στο NAM και η ροή 3 κόβεται με τη διακοπή της ζεύξης , χωρίς να μπορώ να το εξηγήσω, η οποία επαναφέρεται τη χρονική στιγμή 2.82 περίπου.

(ιβ) Τροποποιήστε τον κώδικα της προσομοίωσης του ερωτήματος (ια), ώστε οι ροές από τους κόμβους n(0) και n(1) προς τον n(4) να συνεχίζουν να δρομολογούνται και μετά τη διακοπή της ζεύξης n(2)-n(4) μέσω εναλλακτικής διαδρομής. Πόσα πακέτα χάνονται στην περίπτωση αυτή για κάθε μία από τις ροές δεδομένων; Σε ποια χρονική στιγμή μετά τη διακοπή της ζεύξης συνεχίζουν οι ροές δεδομένων από τους κόμβους n(0) και n(1) στον n(4) και μέσω ποιας διαδρομής; Σε ποια χρονική στιγμή μετά την επαναφορά της ζεύξης συνεχίζουν οι ροές δεδομένων από τους κόμβους n(0) και n(1) στον n(4) μέσω της n(2)-n(4);

Η δυνατότητα εύρεσης εναλλακτικής διαδρομής και μετά τη διακοπή της ζεύξης επιτυγχάνεται μέσω της δυναμικής δρομολόγησης , δηλαδή μέσω της αποστολής agents , οι οποίοι να ενημερώνουν τους κόμβους για τη λειτουργικότητα των ζεύξεων.

Προσθέτουμε συνεπώς στην αρχή του κώδικά μας τις παρακάτω εντολές :

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

```
Results for fid 1
Tranfer time : 4,096881
Total packts lost : 0
Total data lost : 0
Total packts sent from node 0: 334
Total data sent from node 0: 501000
Total packts node 3 received: 334
Total data node 3 received: 501000
Results for fid 2
Transfer time : 4,394960
Total packts lost : 5
Total data lost : 7500
Total packts sent from node 0: 88
Total data sent from node 0: 132000
Total packts node 4 received: 82
Total data sent node 4 received: 123000
Results for fid 3
Transfer time : 3,850881
Total packts lost : 0
Total data lost : 0
Total packts sent from node 1: 104
Total data sent from node 1: 156000
Total packts node 3 received: 104
Total data node 3 received: 156000
Results for fid 4
Transfer time : 22,077920
Total packts lost : 6
Total data lost : 6000
Total packts sent from node 1: 3966
Total data sent from node 1: 3966000
Total packts node 4 received: 3959
Total data node 4 received: 3959000
```

Η ροή 3 ακολουθεί την 1->4, τη χρονική στιμή 1.67. Η ροή 1 τη διαδρομή 0->2->1->4, τη χρονική στιγμή 1.77.

Βλέπουμε από την εκτέλεση στο NAM ότι μετά τη διακοπή της ζεύξης , λόγω της ενημέρωσης από τους rtProto agents έχουμε τις εξής νέες δρομολογήσεις :
0 -> 2 -> 1 -> 4 , και 1 -> 4.

(ιγ) Τροποποιήστε τον αρχικό κώδικα, ώστε να μετατρέψετε την ενσύρματη τοπολογία του δικτύου σε ασύρματη, χρησιμοποιώντας το πρότυπο IEEE 802.11. Επίσης, αντικαταστήστε την εκθετική κίνηση (Exponential) με κίνηση CBR πακέτων μεγέθους 1500 byte και ρυθμού 1.2 Mbps. Χρησιμοποιήστε επίπεδο πλέγμα μήκους 600 m και πλάτους 400 m και τοποθετήστε τον κόμβο n(0) στο κέντρο του, ενώ τους υπόλοιπους κόμβους σε κορυφές τετραγώνου, οι οποίες να απέχουν από τον n(0) απόσταση ίση με 200 m

Για τη δημιουργία της ζητούμενης ασύρματης τοπολογίας τροποποίησαμε τον κώδικα tcl ως εξής :

```
set opt(chan) Channel/WirelessChannel
set opt(prop) Propagation/TwoRayGround
set opt(ant) Antenna/OmniAntenna
set opt(ll) LL
set opt(ifq) Queue/DropTail/PriQueue
set opt(ifqlen) 25
set opt(netif) Phy/WirelessPhy
set opt(mac) Mac/802_11
set opt(rp) AODV
set opt(nn) 5
set opt(gridx) 600
set opt(gridy) 400
$opt(mac) set basicRate_ 1Mb
$opt(mac) set dataRate_ 11Mb
set ns [new Simulator]
$ns color 1 blue
$ns color 2 red
$ns color 3 green
$ns color 4 yellow
$ns use-newtrace
set tf [open lab9.tr w]
$ns trace-all $tf
set nf [open lab9.nam w]
$ns namtrace-all-wireless $nf $opt(gridx) $opt(gridy)
proc finish {} {
 global ns tf nf
 $ns flush-trace
 close $tf
 close $nf
 exit 0
}
set topo [new Topography]
$topo load_flatgrid $opt(gridx) $opt(gridy)
create-god $opt(nn)
$ns node-config -adhocRouting $opt(rp) \
-llType $opt(ll) \
-macType $opt(mac) \
-ifqType $opt(ifq) \
-ifqLen $opt(ifqlen) \
-antType $opt(ant) \
-propType $opt(prop) \
-phyType $opt(netif) \
-channel [new $opt(chan)] \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace OFF

set n(0) [$ns node]
$n(0) random-motion 0
$n(0) set X_ 300.0
$n(0) set Y_ 200.0
```

```
$n(0) set Z_ 0.0
set n(1) [$ns node]
$n(1) random-motion 0
$n(1) set X_ 100.0
$n(1) set Y_ 200.0
$n(1) set Z_ 0.0
set n(2) [$ns node]
$n(2) random-motion 0
$n(2) set X_ 300.0
$n(2) set Y_ 0.0
$n(2) set Z_ 0.0
set n(3) [$ns node]
$n(3) random-motion 0
$n(3) set X_ 500.0
$n(3) set Y_ 200.0
$n(3) set Z_ 0.0
set n(4) [$ns node]
$n(4) random-motion 0
$n(4) set X_ 300.0
$n(4) set Y_ 400.0
$n(4) set Z_ 0.0


set udp1 [new Agent/UDP]
$udp1 set fid_ 1
$udp1 set packetSize_ 1500
$ns attach-agent $n(0) $udp1
set null1 [new Agent/Null]
$ns attach-agent $n(3) $null1
$ns connect $udp1 $null1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1500
$cbr1 set rate_ 1.0mb
$cbr1 set random_ off
set tcp2 [new Agent/TCP]
$tcp2 set packetSize_ 1460
$tcp2 set fid_ 2
$ns attach-agent $n(0) $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink2
$ns connect $tcp2 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set udp3 [new Agent/UDP]
$udp3 set fid_ 3
$udp3 set packetSize_ 1500
$ns attach-agent $n(1) $udp3
set null3 [new Agent/Null]
$ns attach-agent $n(3) $null3
$ns connect $udp3 $null3
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
$cbr3 set packetSize_ 1500
```

```
$cbr3 set rate_ 1.2mb
$cbr1 set random_ off
set tcp4 [new Agent/TCP]
$tcp4 set packetSize_ 960
$tcp4 set fid_ 4
$ns attach-agent $n(1) $tcp4
set sink4 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink4
$ns connect $tcp4 $sink4
set telnet4 [new Application/Telnet]
$telnet4 attach-agent $tcp4
$telnet4 set interval_ 0.001
for {set i 0} {$i < $opt(nn) } {incr i} {
$ns initial_node_pos $n($i) 20
$ns at 0.15 "$cbr1 start"
$ns at 0.3 "$ftp2 start"
$ns at 0.45 "$cbr3 start"
$ns at 0.6 "$telnet4 start"
$ns at 4.15 "$cbr1 stop"
$ns at 4.3 "$ftp2 stop"
$ns at 4.45 "$cbr3 stop"
$ns at 4.6 "$telnet4 stop"
$ns at 25.0 "finish"
$ns run
```

(ιδ) Με τη βοήθεια κατάλληλου script σε γλώσσα awk, υπολογίστε τα ακόλουθα για κάθε μία από τις τέσσερις ροές δεδομένων:

1. ***το χρόνο μετάδοσης πακέτων,***
2. ***το συνολικό πλήθος των πακέτων και τον όγκο δεδομένων (byte) που αποστέλλονται από τους κόμβους πηγής n(0) και n(1) (εξαιρώντας τις αναμεταδόσεις και τις αποστολές επιβεβαιώσεων),***
3. ***το συνολικό πλήθος των πακέτων και τον όγκο δεδομένων που χάνονται,***
4. ***το συνολικό πλήθος των πακέτων και τον όγκο δεδομένων που λαμβάνονται από τους κόμβους προορισμού n(3) και n(4) (εξαιρώντας τις λήψεις επιβεβαιώσεων). Σχολιάστε τα αποτελέσματά σας.***

Αν τροποποιήσουμε το αρχείο awk ώστε να συνάδει με το new trace , προκύπτουν τα παρακάτω αποτελέσματα:

```
Results for fid 1
Tranfer time :  4.005710 sec
Total packts lost : 4
Total data lost : 6080
Total packts node 3 received: 330
Total data node 3 received: 494920
Results for fid 2
Transfer time : 4.390680
Total packts lost : 7
Total data lost : 10640
Total packts node 4 received: 200
Total data sent node 4 received: 299860
Results for fid 3
Transfer time : 4.087510
Total packts lost : 51
Total data lost : 77520
Total packts node 3 received: 350
Total data node 3 received: 523980
Results for fid 4
Transfer time : 24.398600
Total packts lost : 14
Total data lost : 14280
Total packts node 4 received: 2893
Total data node 4 received: 2892720
```

(ιε) Με βάση την απάντησή σας στο προηγούμενο ερώτημα, έχουν ολοκληρωθεί όλες οι ροές δεδομένων; Αν όχι, πόσο πρέπει να αυξηθεί ο χρόνος προσομοίωσης ώστε να ολοκληρωθούν και οι τέσσερις ροές;

Αν αυξήσουμε το χρόνο προσομοίωσης , θα δούμε πως ο συνολικός χρόνος μετάδοσης της ροής 4 αυξάνει , συνεπώς συμπεραίνουμε πως πρίν σίγουρα δεν είχαν ολοκληρωθεί όλες οι ροές δεδομένων . Αυτό προκύπτει άλλωστε και από τον υπολογισμό της καθυστέρησης της κάθε ζεύξης. Συνεπώς , αν αυξήσουμε το χρόνο προσομοίωσης στα 40 seconds παρατηρούμε πως έχουν ολοκληρωθεί και οι τέσσερις ροές.