ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Θέμα: Επέκταση του εξομοιωτή NS2 για την υποστήριξη ενός νέου πρωτοκόλλου

ΦΟΙΤΗΤΗΣ: ΧΡΗΣΤΟΣ ΠΑΝΑΓΙΩΤΗΣ ΜΠΟΥΡΟΠΟΥΛΟΣ

ΕΠΙΒΛΕΠΩΝ: ΔΗΜΗΤΡΗΣ ΑΜΠΕΛΙΩΤΗΣ

ΑΝΤΙΡΡΙΟ 2019

ΠΕΡΙΕΧΌΜΕΝΑ

1.Εισαγωγή

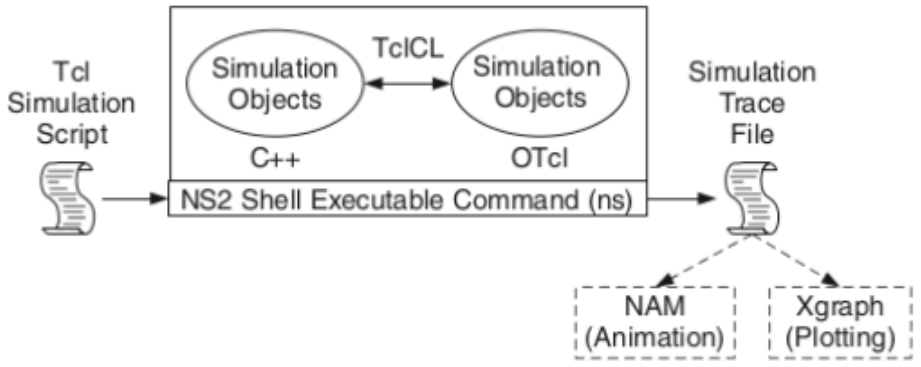
1.1 Τα δίκτυα είναι από τους μεγαλύτερους ευρενητικούς κλάδους στην επιστήμη των υπολογιστών. Ωστόσο, η έρευνα και ο πειραματισμός σε πραγματικά δίκτυα είναι κάτι το δύσκολο. Οι περιορισμένοι πόροι, οι δύσκολες προυποθέσεις για πολύπλοκα συστήματα, η ρύθμιση τοπολογιών και συστημάτων και άλλα προβλήματα έχουν ως αποτέλεσμα την δυσκολία στην μελέτη αυτής της επιστήμης.

1.2 Βέβαια, τα παραπάνω προβλήματα μπορούν να αντιμετωπιστούν αποτελεσματικά με την χρήση ενός προσομοιωτή. Χωρίς κάποιο κόστος δίνεται η δυνατότητα στον ερευνητή να πειραματιστεί και να μελετήσει σε βάθος μέσω των προσομοιώσεων των δικτύων. Επίσης, η συλλογή δεδομένων γίνεται ευκολότερη και παράλληλα υπάρχει αξιόπιστη αναπαράσταση του πραγματικού περιβάλλοντος. Μπορούμε ακόμη να επαναλάβουμε το πείραμα όσες φορές επιθυμούμε με ασφάλεια και πρακτικότητα. Από την άλλη πλευρά μπορεί να οδηγηθούμε σε έλλειψη ακρίβειας αποτελεσμάτων.

2. NS2

2.1 Ένας από τους πιο διάσημους προσομοιωτές είναι ο NS2(Network Simulator version-2). Είναι λοιπόν ένας προσομοιωτής δικτύων διακριτών γεγονότων, ο οποίος έχει αποδειχθεί πολύ χρήσιμος στην μελέτη των δικτύων. Υποστηρίζει ενσύρματα και ασύρματα δίκτυα και πρωτόκολλα δικτύων όπως TCP, UDP, αλγόριθμους δρομολόγησης και άλλα. Χάρη στην ευελιξία που τον χαρακτηρίζει και στην δυνατότητα της επεκτασιμότητας, είναι ιδιαίτερα δημοφιλής στην κοινότητα της επιστήμης των δικτύων από την πρώτη κιόλας στιγμή της δημιουργίας του, το 1989.

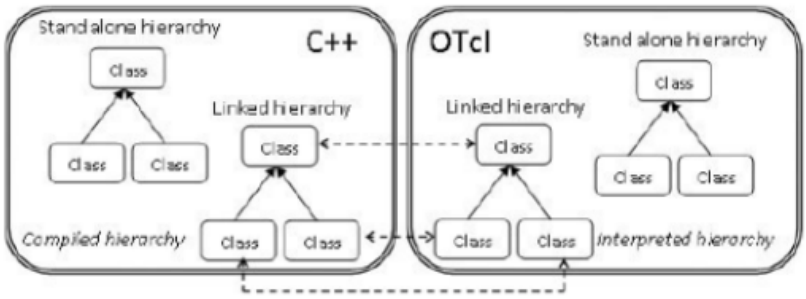
2.2 Ο NS2 αποτελείται από 2 γλώσσες , την C ++ και την Object-oriented Tool Command Language (OTcl). Η C++ είναι η γλώσσα με τις back-end λειτουργίες παρέχοντας υψηλές ταχύτητες εκτέλεσης. Υλοποιεί τις λειτουργίες της στο επίπεδο του πυρήνα. Αντίθετα η OTcl έχει front-end λειτουργίες και αποτελεί μικρό τμήμα του πυρήνα. Ουσιαστικά, τα πρωτόκολλα υλοποιούνται από την C++ ενώ η OTcl παρέχει ένα "χερούλι" στον χρήστη ώστε να χρησιμοποιεί αυτά τα πρωτόκολλα. Η σύνδεση μεταξύ της C ++ και της OTcl γίνεται από την tclCL(Tcl with Classes Library).



Εικόνα 2.1 - Δομή του NS2.

Ο NS2 παρέχει πολλά ενσωματωμένα C++ αντικείμενα τα οποία μπορούν να χρησιμοποιηθούν μέσω TCL scripts. Βέβαια, ένας πιο έμπειρος χρήστης που έχει σκοπό να σχεδιάσει ο ίδιος το δικό του πρωτόκολλο, μπορεί να το κάνει δημιουργώντας τα δικά του κατάλληλα C ++ αντικείμενα και σχεδιάζοντας την κατάλληλη διεπαφή OTcl ώστε να έχει πρόσβαση μέσω των TCL αρχείων. Όπως προαναφέραμε ο κώδικας της C++ γλώσσας υλοποιείται πιο γρήγορα από την OTcl, ωστόσο οι αλλαγές στον C ++ κώδικα απαιτούν περισσότερο χρόνο και προσπάθεια καθώς χρειάζονται να ξανά μεταγλωτιστούν.

2.3 Και οι δύο γλώσσες , η C++ και η OTcl, αποτελούνται από δύο διαφορετικές ιεραρχίες κλάσεων, την linked hierarchy και την standalone hierarchy. Οι linked hierarchy κλάσεις των δύο γλωσσών συνδέονται μεταξύ τους χρησιμοποιώντας την TclCL, όπως φαίνεται στο παρακάτω σχήμα(2.2). Με άλλα λόγια , για κάθε διαθέσιμη linked hierarchy κλάση στην C++, υπάρχει μία αντίστοιχη κλάση στην OTcl , δημιουργώντας έτσι μία ένα-προς-ένα αντιστοιχία. Η OTcl linked hierarchy κλάση ονομάζεται “interpreted hierarchy”, ενώ η αντίστοιχη στην C++ ονομάζεται “compiled hierarchy”. Ο δεύτερος τύπος ιεραρχιών κλάσεων, η standalone hierarchy, δεν είναι συνδεδεμένη για τις δύο γλώσσες. Οι κλάσεις αυτές δεν αποτελούν μέρος ούτε στην interpreted hierarchy ούτε στην compiled hierarchy.

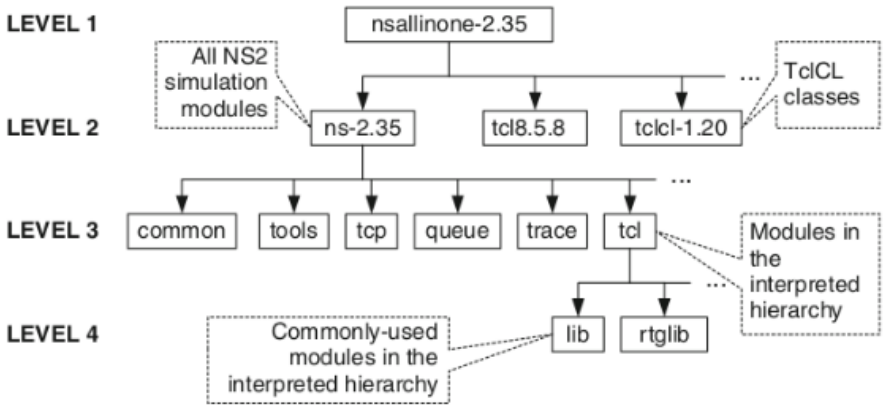


Εικόνα 2.2 - Οι κλάσεις για τις 2 γλώσσες.

2.4 Συμπεραίνουμε λοιπόν, πως ο χρήστης μπορεί να προγραμματίσει όποια απο τις δύο γλώσσες επιθυμεί ανάλογα με τον σκοπό που έχει. Ωστόσο , ο βέλτιστος και ο ευκολότερος τρόπος όταν επιθυμούμε να δημιουργήσουμε μια προσομοίωση με τα υπάρχοντα πρωτόκολλα προτιμάται ο προγραμματισμός στην OTcl γλώσσα, ενώ όταν θέλουμε να τροποποιήσουμε ή να προσθέσουμε κάποιο πρωτόκολλο προτιμάται ο προγραμματισμός στην C++ γλώσσα.

3. NS2 directories

3.1 Βλέποντας την εικόνα 3.1 μπορούμε να δούμε σχηματικά την δομή του λογισμικού που εγκαταστήσαμε. Μπορούμε να δούμε τις υποκατηγόριες και τους φακέλους που εμπεριέχονται χρησιμοποιώντας τις εντολές cd και ls.



Εικόνα 3.1- NS2 directories και φάκελοι

Συνοπτικά βλέποντας μερικούς φακέλους:

Level 2
ns-2.35: Εδώ εμπεριέχονται όλες οι ενότητες της προσομοίωσης του NS2. Περιέχει τα στοιχεία της C++ και της TCL .
Tcl8.5.10: περιέχει τα πακέτα της TCL
nam1.15: περιέχει τα πακέτα του NAM (network animator)
tclcl1.20:περιέχει τις κλάσεις τις TclCL
Level 3
common: οι φάκελοι αυτού του επιπέδου περιέχουν κοινά πακέτα προώθησης
queue: περιέχει μονάδες ουράς αναμονής
TCP: περιέχει το Transmission Control Protocol
tcl: περιέχει την interpreted ιεραρχία
Level 4
edu: περιέχει κάποια εκπαιδευτικά ns scripts για την επίδειξη της προσομοίωσης
lib: περιέχει τον OTcl κώδικα

4. Συντακτικό TCL συντοπτικά

4.1 Σε αυτό το σημείο ας δούμε μερικά από τα βασικά του συντακτικού της TCL γλώσσας.

Command	Output
set x 3	3
set x y	y
set y $x	3
set d $x + $y	3 + 3

puts “etsi ektupwnei string”	etsi ektupwnei string
puts “ H timh tou x einai $x “	H timh tou x einai 3
set d [expr $x+2]	5

4.2 Σύνταξη συνθήκης -if

```
if { συνθήκη } {
        γραμμή εντολών που εκτελείται εάν ισχύει η συνθήκη.
  }else {
        γραμμή εντολών που εκτελείται εάν δεν ισχύει η συνθήκη
}
```

Πολλαπλή συνθήκη -if

```
if{πρώτη συνθήκη}{
        γραμμή εντολών που εκτελείται εάν ισχύει η πρώτη συνθήκη.
}elseif{δεύτερη συνθήκη}{
        γραμμή εντολών που εκτελείται εάν ισχύει η δεύτερη συνθήκη.
}else{
        γραμμή εντολών που εκτελείται εάν δεν ισχύει καμία συνθήκη.
}
```

4.3 Σύνταξη ελέγχου καταστάσεων ροής -while/for

```
while{συνθήκη} {
        εντολές που εκτελούνται
}
```

```
for {αρχικοποιήση_μεταβλητής} {συνθήκη} {αύξηση/μείωση μεταβλητής} {
        εντολές που εκτελούνται
}
```

4.4 Τελεστές

- , +	Μείον , συν
* , /	Πολλαπλασιασμος, διαίρεση
<, >	Μικρότερο από, μεγαλύτερο από

4.5 Μερικές Μαθηματικές συναρτήσεις

abs(x)	sqrt(x)	log(x)
cos(x)	int(x)	log10(x)
sin(x)	round(x)	
tan(x)	double(x)	

5. Προσθήκη δικού μας πρωτοκόλλου

5.1 Σε αυτό το σημείο της αναφοράς θα δούμε πως μπορούμε να προσθέσουμε το δικό μας πρωτόκολλο στον NS2. Πιο συγκεκριμένα, θα προσθέσουμε το πρωτόκολλο ping. Είναι μία λειτουργία για τον εντοπισμό της διαθεσιμότητας ενός απομακρυσμένου πόρου στο δίκτυο. Με το ping αποστέλλεται στον δέκτη ένα πακέτο δεδομένων και στη συνέχεια ο πομπός που έστειλε το πακέτο περίμενει για μία echo reply, δηλαδή την απάντηση στο πακέτο δεδομένων που έστειλε. Επίσης, ιδιαίτερα χρήσιμος για την ανάλυση και την καλύτερη κατανόηση της τοπολογίας του δικτύου είναι ο υπολογισμός του round-trip-time, δηλαδή ο συνολικός χρόνος που χρειάστηκε για να ολοκληρωθεί η διαδικασία που περιγράψαμε.

5.2

Αφού έχουμε εγκαταστήσει με επιτυχία το πρόγραμμα ανοίγουμε τον φάκελο ns-allinone-2.35. Προχωρώντας στο ns-2.35/apps πρέπει πρώτα να φτιάξουμε το header αρχείο ‘myping.h’ όπου εδώ, σε κάθε νέο ping packet που θα δημιουργείται, θα δηλώνονται οι σχετικές πληροφορίες στο header κάθε πακέτου που θα πρέπει να μεταφερθούν. Ο κώδικας:

```
1. #ifndef ns_ping_h
2. #define ns_ping_h
3.
4. #include "agent.h"
5. #include "tclcl.h"
6. #include "packet.h"
7. #include "address.h"
8. #include "ip.h"
9.
```

```
10.
11.
12. struct hdr_myping {
13.         char ret;
14.         double send_time;      // η στιγμή που το πακέτο ping φεύγει από τον πομπό
15.         double rcv_time;       // η στιγμή που το πακέτο ping φτάνει στον δέκτη
16.         int seq;               // αριθμός ακολουθίας
17.
18.         static int offset_;
19.         inline static int& offset() { return offset_; }
20.         inline static hdr_myping* access(const Packet* p) {  // μέθοδοι που χρησιμοποιούμε
                                   στη συνέχεια για να αποκτήσουμε πρόσβαση στο ping header
21.                 return (hdr_myping*) p->access(offset_);
22.         }
23. };
24.
25. class MyPingAgent : public Agent { // η υποκλάση MyPingAgent  της κλάσης Agent
26. public:
27.         MyPingAgent();
28.         int seq;
29.         int oneway;
30.         virtual int command(int argc, const char*const* argv);
31.         virtual void recv(Packet*, Handler*);   //Δήλωση συναρτήσεων command, recv
32. };
33.
34. #endif
```

Η μεταβλητή 'ret' είναι ίση με 0 όταν βρίσκεται σε κίνηση από τον αποστολέα προς τον κόμβο-παραλήπτη που θέλουμε, ενώ έχει την τιμή 1 όταν θα επιστρέφει πίσω. Οι εντολές 'send_time' και 'rcv_time' αποθηκεύουν την χρονική στιγμή που το πακέτο στάλθηκε από τον αποστολέα και την χρονική στιγμή που το πακέτο λήφθηκε απο τον παραλήπτη αντίστοιχα. Στην εντολή 'seq' αποθηκεύεται ο αριθμός ακολουθίας του πακέτου. Πιο κάτω δηλώνουμε την κλάση 'MyPingAgent' ως υποκλάση της 'Agent'.

5.3 Στη συνέχεια στο ίδιο path θα πρέπει να φτιάξουμε το C++ αρχείο. Σε αυτό το σημείο θα οριστεί η σύνδεση μεταξύ του C++ και TCL κώδικα.

```
1. #include "myping.h"
2.
3. int hdr_myping::offset_;
4. static class PingHeaderClass : public PacketHeaderClass {
```

```
public:
        PingHeaderClass() : PacketHeaderClass("PacketHeader/MyPing",
                                              sizeof(hdr_myping)) {
                bind_offset(&hdr_myping::offset_);
        }
} class_mypinghdr;
                                        //ορίζουμε την σύνδεση μεταξύ του κώδικα C ++ και TCL

static class MyPingClass : public TclClass {
public:
        MyPingClass() : TclClass("Agent/MyPing") {}
        TclObject* create(int, const char*const*) {
                return (new MyPingAgent());
        }
} class_myping;


// Στην συνέχεια δημιουργείται ο constructor της κλάσης 'MyPingAgent'. Ο constructor είναι μια συνάρτηση-μέλος μιας κλάσης ο οποίος αρχικοποιεί τα αντικείμενα της κλάσης. Στην C++ ο constructor καλείται αυτόματα όταν ένα αντικείμενο  δημιουργείται , δεν επιστρέφει τιμές, έχει το ίδιο όνομα με την κλάση,.εάν δεν ορίσουμε εμείς ο μεταγλωτιστής της C++ ορίζει από μόνος του αυτόματα.


MyPingAgent::MyPingAgent() : Agent(PT_MYPING), seq(0), oneway(0)
{
        bind("packetSize_", &size_);   // Ενώνει τις μεταβλητές που έχουν πρόσβαση και στον                                       //κώδικα της C++ και της TCL.
}

//Η συνάρτηση 'command()' καλείται όταν μια εντολή στον κώδικα TCL για την κλάση 'MyPingAgent' εκτελείται. Όταν για παράδειγμα εκτελείται η εντολή "$p1 send" στέλνουμε ping πακέτα

int MyPingAgent::command(int argc, const char*const* argv)
{
  if (argc == 2) {
    if (strcmp(argv[1], "send") == 0) {         //εάν στέλνουμε το πακέτο
      // Δημιούργησε νέο πακέτο
      Packet* pkt = allocpkt();
      //Πρόσβαση στο Ping header του νέου πακέτου
      hdr_myping* hdr = hdr_myping::access(pkt);
      // Όρισε τη μεταβλητη  'ret' ίση με το  0, ώστε ο δέκτης να ξέρει ότι πρέπει να στείλει ένα echo
      // packet
      hdr->ret = 0;
```

```
46.      hdr->seq = seq++;
47.      // Αποθήκευσε την  τρέχουσα ώρα στο πεδίο 'send_time'
48.      hdr->send_time = Scheduler::instance().clock();
49.      //Στείλε το πακέτο
50.      send(pkt, 0);
51.      // επέστρεψε 'TCL_OK'
52.      return (TCL_OK);
53.
54.    }
55.
56.
57.    else if (strcmp(argv[1], "start-WL-brdcast") == 0) {  // εάν στέλνουμε broadcast μετάδοση
58.      Packet* pkt = allocpkt();
59.
60.      hdr_ip* iph = HDR_IP(pkt);
61.      hdr_myping* ph = hdr_myping::access(pkt);
62.
63.      iph->daddr() = IP_BROADCAST;                  //προχώρησε στο ip header , αποθήκευσε
64.      iph->dport() = iph→sport();                   την broadcast ip, port, εκχώρησε ret=0
65.      ph->ret = 0;                                  και στείλε το πακέτο
66.      send(pkt, (Handler*) 0);
67.      return (TCL_OK);
68.    }
69.
70.    else if (strcmp(argv[1], "oneway") == 0) {
71.      oneway=1;
72.      return (TCL_OK);
73.    }
74.  }
75.  return (Agent::command(argc, argv));
76. }
77.
78. //Η συνάρτηση 'recv()'  είναι αυτή που ορίζει ποιές πράξεις πρέπει να γίνουν όταν ένα
    πακέτο λαμβάνεται. Εάν το πεδίο 'ret' του πακέτου έχει την τιμή 0, τότε ένα πακέτο με τις
    ίδιες τιμές πρέπει να δημιουργηθεί αλλάζοντας την τιμή 'ret' σε 1.
79.
80. void MyPingAgent::recv(Packet* pkt, Handler*)
81. {
82.   // Προχώρα στην IP header του ληφθέντος πακέτου
83.   hdr_ip* hdrip = hdr_ip::access(pkt);
84.
85.   // Προχώρα στο Ping header του ληφθέντος πακέτου
86.   hdr_myping* hdr = hdr_myping::access(pkt);
87.
88.
89.   // σε περίπτωση που το πακέτο που λάβαμε είναι για broadcast  μετάδοση
90.   if ((u_int32_t)hdrip->daddr() == IP_BROADCAST) {
91.     if (hdr->ret == 0) {          //εάν το πακέτο δεν έχει ληφθεί από κάποιον κόμβο

92.
```

```
93.     printf("Recv BRDCAST Ping REQ : at %d.%d from %d.%d\n", here_.addr_,
    here_.port_,        hdrip->saddr(), hdrip→sport()); //
94.     Packet::free(pkt);          // εκτύπωσε ότι λήφθηκε και ελευθέρωσε
95.
96.     //δημιούργησε πακέτο (reply)
97.     Packet* pktret = allocpkt();
98.
99.     hdr_myping* hdrret = hdr_myping::access(pktret);
100.            hdr_ip* ipret = hdr_ip::access(pktret);
101.
102.            hdrret->ret = 1;
103.
104.            // πρόσθεσε port και  brdcast address
105.            ipret->daddr() = IP_BROADCAST;
106.            ipret->dport() = ipret->sport();
107.
108.            send(pktret, 0);
109.
110.          } else {  //αλλίως το πακέτο αυτό έχει ήδη την τιμή ret και επιστρέφει
111.            printf("Recv BRDCAST Ping REPLY : at %d.%d from %d.%d\n", here_.addr_,
    here_.port_, hdrip->saddr(), hdrip->sport());
112.            Packet::free(pkt);
113.          }
114.          return;
115.        }
116.        // Εάν το πεδίο 'ret'  = 0
117.        if (hdr->ret == 0) {
118.              //Αποθήκευσε την ώρα που στάλθηκε το πακέτο
119.          double stime = hdr->send_time;
120.          int rcv_seq = hdr->seq;
121.          // Διέγραψε το πακέτο
122.          Packet::free(pkt);
123.          //Δημιούργησε νέο πακέτο
124.          Packet* pktret = allocpkt();
125.          // Προχώρα στο Ping header του νέου πακέτου
126.          hdr_myping* hdrret = hdr_myping::access(pktret);
127.          //Βάλε την τιμή 'ret' = 1 ώστε ο παραλήπτης να μην στείλει άλλο πακέτο echo
128.          hdrret->ret = 1;
129.          //Στείλε την ώρα αποστολής
130.          hdrret->send_time = stime;
131.          hdrret->rcv_time = Scheduler::instance().clock();
132.          hdrret->seq = rcv_seq;
133.          //Στείλε το πακέτο
134.          send(pktret, 0);
135.        } else {  //το πακέτο έχει ret==1 και το πακέτο έχει επιστρέψει
136.
137.          char out[100];
138.
139.          if (oneway)
140.              sprintf(out, "%s recv %d %d %3.1f %3.1f", name(),
```

```
141.                    hdrip->src_.addr_ >> Address::instance().NodeShift_[1],
142.                    hdr->seq, (hdr->rcv_time - hdr->send_time) * 1000,
143.                    (Scheduler::instance().clock()-hdr->rcv_time) * 1000);
144.            else sprintf(out, "%s recv %d %3.1f", name(),
145.                    hdrip->src_.addr_ >> Address::instance().NodeShift_[1],
146.                    (Scheduler::instance().clock()-hdr->send_time) * 1000);
147.            Tcl& tcl = Tcl::instance();
148.            tcl.eval(out);
149.            Packet::free(pkt);
150.          }
151.        }
```

5.4 Εκτός από τα δύο αρχεία που προσθέσαμε θα πρέπει να τροποποιήσουμε ορισμένα υπάρχοντα αρχεία. Αρχικά θα χρειαστούμε ένα νέο τύπο πακέτου για το ping agent , οπότε πρώτα θα επεξεργαστούμε το αρχείο 'packet.h'. Εκεί θα βρούμε τους ορισμούς των πακέτων πρωτοκόλλων (TCP , UDP , ..). Οι αλλαγές στο packet.h:

a)

……...

```
// insert new packet types here

static const packet_t PT_MYPING = 73;    // <----------------Εδώ αλλάζω


static packet_t       PT_NTYPE = 74; // Αυτό πρέπει να είναι τελευταίο
```

……..

b)

…….

```
class p_info {
public:
        p_info()
        {
                initName();
        }
        const char* name(packet_t p) const {
                if ( p <= p_info::nPkt_ ) return name_[p];
                return 0;
        }
        static bool data_packet(packet_t type) {
                return ( (type) == PT_TCP || \
                        (type) == PT_TELNET || \
                        (type) == PT_CBR || \
                        (type) == PT_AUDIO || \
```

```
                        (type) == PT_VIDEO || \
                        (type) == PT_ACK || \
                        (type) == PT_SCTP || \
                        (type) == PT_SCTP_APP1 || \
                        (type) == PT_HDLC || \
                           (type) == PT_MYPING \        // < – Προσθέτω εδώ

                        );
…….
```

c)

```
……..

name_[PT_TFRC]= "tcpFriend";
                name_[PT_TFRC_ACK]= "tcpFriendCtl";
                name_[PT_PING]="ping";

                name_[PT_PBC] = "PBC";

                name_[PT_MYPING]="MyPing";  // <---------------------- Προσθέτω εδώ
……
```

d)

```
……..

#define DATA_PACKET(type) ( (type) == PT_TCP || \
                        (type) == PT_TELNET || \
                        (type) == PT_CBR || \
                        (type) == PT_AUDIO || \
                        (type) == PT_VIDEO || \
                        (type) == PT_ACK || \
                        (type) == PT_SCTP || \
                        (type) == PT_SCTP_APP1 || \
                                (type) == PT_MYPING\    // < – Προσθέτω εδώ
                                )

…….


      .....
 }
```

Συνεχίζοντας τις αλλαγές, σειρά έχει το αρχείο ns-3.35/tcl/lib/ns-default.tcl. Σε αυτό το αρχείο ορίζονται όλες οι τιμές των TCL αντικειμένων. Προσθέτουμε την εξής γραμμή για να θέσουμε το default μέγεθος του πακέτου που θα χρησιμοποιεί ο νέος agent.

Agent/MyPing set packetSize_64

Τέλος , θα πρέπει να επεξεργαστούμε το ‘Makefile’ και να προσθέσουμε το αρχείο ‘myping.o’ .

5.5 Σημαντικό είναι πριν εγκαταστήσουμε το πρόγραμμα να ξανά κάνουμε μεταγλώτιση. Οπότε πρέπει να εκτελεστούν οι ακόλουθες εντολές :

```
make clean
make
./install
```

6 Παράδειγμα του νέου πρωτοκόλλου με κώδικα TCL

Αφού τελειώσαμε την διαδικασία δημιουργίας του νέου πρωτοκόλλου ας δούμε ένα απλό παράδειγμα στα ενσύρματα δίκτυα. Σημαντικό είναι να μην ξεχνάμε να ορίσουμε την συνάρτηση ‘recv()’ και να δηλώσουμε το μέγεθος το μέγεθος του ping πακέτου.

```
1.  #Create a simulator object
2.  set ns [new Simulator]
3.
4.  #Open a trace file
5.  set nf [open out.nam w]
6.  $ns namtrace-all $nf
7.
8.  #Define a 'recv' function for the class 'Agent/MyPing'
9.  Agent/MyPing instproc recv {from rtt} {
10.         $self instvar node_
11.         puts "node [$node_ id] received ping answer from \
12.           $from with round-trip-time $rtt ms."
13. }
14. Agent/MyPing set packetSize_ 64
15.
16. #Define a 'finish' procedure
17. proc finish {} {
18.       global ns nf
19.       $ns flush-trace
20.       close $nf
21.       exec nam out.nam &
22.       exit 0
23. }
24.
25. #Create 2 nodes
26. set n0 [$ns node]
27. set n1 [$ns node]
28.
29.
```

```
30. #Connect the nodes with two links
31. $ns duplex-link $n0 $n1 1Mb 15ms DropTail
32.
33.
34. #Create  ping agents and attach them to the nodes
35. set p0 [new Agent/MyPing]
36. $ns attach-agent $n0 $p0
37.
38. set p1 [new Agent/MyPing]
39. $ns attach-agent $n1 $p1
40.
41.
42.
43. #Connect the  agents
44. $ns connect $p0 $p1
45.
46.
47. #Schedule events
48. $ns at 0.1 "$p0 send"
49.
50.
51. $ns at 1.0 "finish"
52.
53. #Run the simulation
54. $ns run
```

Το output από το terminal:

node 0 received ping answer from 1 with round-trip-time 31.0 ms.

Οπότε ο κόμβος 0 έστειλε ping στον κόμβο 1 και έλαβε το echo reply σε χρόνο 31.0 ms.

Πηγές που χρησιμοποιήθηκαν:

"*Computer Network Simulation Using NS2*" ,Ajit Kumar Nayak, Satyananda Champati Rai, Rajib Mall

"*Introduction to Network-Simulator-NS2*", Teerawat Issariyakul, Ekram Hossain