



Εργαστήριο Δικτύων

8^η Διάλεξη:

- Network Simulator – NS

Περίγραμμα παρουσίασης

- Εισαγωγή στον ns
- Τα τμήματα (components) του ns
- Η δομή του ns
- Ξεκινώντας...
 - Το 1^ο script
 - Εκτέλεση προσομοιώσεων
- Trace analysis
- Παραδείγματα

Εισαγωγή

- Εργαλείο για την έρευνα στο πεδίο των δικτύων
- Προϊόν ανοιχτού κώδικα (open source)
- Εγκαθίσταται σε: Unix, SunOS, Linux, Windows κ.α.
- Σύνδεσμοι:
 - <http://www.isi.edu/nsnam/ns/>
 - http://nsnam.isi.edu/nsnam/index.php/Main_Page

Εισαγωγή (συν.)

- Στόχοι-Δυνατότητες:
 - ανάπτυξη ερευνητικής δραστηριότητας
 - εύκολη σύγκριση παρόμοιων λύσεων (π.χ. πρωτόκολλα)
 - συνεργασία
 - έλεγχος θεωρητικών αποτελεσμάτων
 - βελτίωση της εκπαιδευτικής διαδικασίας
 - πολλαπλά επίπεδα μελέτης
 - application <-> physical

Εισαγωγή (συν.)

- Δικτυακά πρωτόκολλα που υποστηρίζει
 - ενσύρματα, ασύρματα, δορυφορικά
 - TCP, UDP, multicast
 - web, telnet, ftp
 - δρομολόγησης σε ad-hoc δίκτυα

Εισαγωγή (συν.)

Ερευνητές από όλο τον κόσμο προσθέτουν νέες δυνατότητες:

- [AODV-UU](#)
- [BlueHoc](#)
- **OBS**
- **IR-UWB**

Discrete event

- Προσομοίωση διακριτού γεγονότος (discrete event simulation)
 - όλα είναι γεγονότα (events)
 - ο ης κρατά τη λίστα με τα events που εκκρεμούν
 - παίρνει το πιο άμεσο και το εκτελεί μέχρι τέλους
 - κάθε γεγονός συμβαίνει σε μια στιγμή του εικονικού χρόνου, αλλά παίρνει μια αυθαίρετη ποσότητα πραγματικού χρόνου

Discrete event

Consider two nodes
on an Ethernet:



simple
queuing
model:

$t=1$, A enqueues pkt on LAN
 $t=1.01$, LAN dequeues pkt
and triggers B

detailed
CSMA/CD
model:

$t=1.0$: A sends pkt to NIC
A's NIC starts carrier sense
 $t=1.005$: A's NIC concludes cs,
starts tx
 $t=1.006$: B's NIC begins receiving pkt
 $t=1.01$: B's NIC concludes pkt
B's NIC passes pkt to app

Τα τμήματα (components) του ns

- ns (Network Simulator)
- nam (Network AniMator)
- προ-επεξεργαστικά
 - Γεννήτριες τοπολογιών, σχημάτων κίνησης
- μετα-επεξεργαστικά
 - ανάλυση και επεξεργασία

ns models

- Traffic models and applications:
 - web, FTP, telnet, constant-bit rate, Real Audio
- Transport protocols:
 - unicast: TCP (Reno, Vegas, etc.), UDP
 - multicast: SRM
- Routing and queueing:
 - wired routing, ad hoc rtg and directed diffusion
 - queueing protocols: drop-tail, RED, fair queueing, etc.
- Physical media:
 - wired (point-to-point, LANs), wireless (multiple propagation models), satellite

Η δομή του ns-2

- object-oriented
- υλοποιημένος σε C++ με frontend OTcl
- Χρήση
 - C++: επεξεργασία ανά πακέτο, αλλαγή υπάρχουσας κλάσης
 - OTcl: configuration, setup, «μια-και-έξω» πράγματα

Η δομή του ns-2 (συν.)

- Σε C++ υλοποιούνται:
 - πρωτόκολλα σε διάφορα επίπεδα (MAC, routing κλπ)
 - ορισμός και επεξεργασία πακέτων
 - βασικές λειτουργίες του event-driven προσομοιωτή
- Σε OTcl καθορίζονται:
 - δομή δικτύου
 - χρησιμοποιούμενα πρωτόκολλα
 - είδος κίνησης
 - γενικές παράμετροι σχετιζόμενοι με την προσομοίωση

Βασικές εντολές tcl

variables:

```
set x 10  
puts "x is $x"
```

functions and expressions:

```
set y [pow x 2]  
set y [expr x*x]
```

control flow:

```
if {$x > 0} { return $x } else {  
return [expr -$x] }  
while { $x > 0 } {  
puts $x  
incr x -1  
}
```

procedures:

```
proc pow {x n} {  
if {$n == 1} { return $x }  
set part [pow x [expr $n-1]]  
return [expr $x*$part]  
}
```

Also lists, associative arrays, etc.

=> can use a real programming language to build network topologies, traffic models, etc.

Το πρώτο script

```
[kokkinop@zenon ~]$ ns
% set ns [new Simulator]
_o4
% $ns at 1 "puts \"Hello!!\""
1
% $ns at 1.5 "exit"
2
% $ns run
Hello!!
[kokkinop@zenon ~]$
```

create a simulator, put in var ns
set ns [new Simulator]

schedule an event at time t=1 to print Hello
\$ns at 1 "puts \"Hello!\""

and exit at a later time
\$ns at 1.5 "exit"

run the simulator, executing events
\$ns run

ns από file

```
# Create a simulator object  
set ns [new Simulator]
```

```
# Save nam and trace data  
set nf [open out.nam w]  
$ns namtrace-all $nf  
set tf [open out.tr w]  
$ns trace-all $tf
```

```
# Declare finish procedure  
proc finish {} {  
    global ns nf  
    $ns flush-trace  
    close $nf  
    exec nam out.nam &  
    exit 0 }  
}
```

```
# Set up finish time  
$ns at 5.0 "finish"
```

```
# Start simulation  
$ns run
```

Το πρώτο script (συν.)

```
# Creating nodes
```

```
set n0 [$ns node]
```

```
set n1 [$ns node]
```

```
...
```

```
...
```

```
...
```

```
# Setting up links between nodes (specifies bandwidth, delay, and queue  
type DropTail, RED, CBQ, FQ, SFQ, DRR)
```

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

```
...
```

```
...
```

```
...
```

Το πρώτο script (συν.)

Create a UDP connection source and attach it to node n0

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

Create the connection's sink and attach it to node n1

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

Create a CBR traffic source and attach it to udp0

for TCP connections we can also create ftp or telnet traffic

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

```
# Connect source to sink
$ns connect $udp0 $null0
```

```
# Start and stop traffic
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Το πρώτο script (συν.)

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
$ns trace-all $tf
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp01

set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
$ns connect $udp0 $null0

$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0 }

$ns at 5.0 "finish"

$ns run
```

Άλλες εντολές

```
# Create a number of nodes
```

```
for {set i 0} {$i < 7} {incr i}
{
    set n($i) [$ns node]
}
```

```
# Connect the nodes
```

```
for {set i 0} {$i < 7} {incr i}
{
    $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}
```

```
# Disable / Enable links
```

```
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
```

```
# Set up dynamic routing protocol, other: Static, Session, DV, cost,
multi-path
```

```
$ns rproto DV
```

Εκτέλεση προσομοιώσεων

- Δημιουργία tcl αρχείου: π.χ. test.tcl
- Ορισμός των κατάλληλων μεταβλητών περιβάλλοντος που σχετίζονται με τον NS
- Εκτέλεση προσομοίωσης: π.χ. ns test.tcl
- Ανάλυση αποτελεσμάτων που περιέχονται στο trace (.tr) αρχείο

Trace analysis

- Όλες οι πληροφορίες αποθηκεύονται στο trace αρχείο

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id
-------	------	-----------	---------	----------	----------	-------	-----	----------	----------	---------	--------

```
r : receive (at to_node)
+ : enqueue (at queue)          src_addr : node.port (3.0)
- : dequeue (at queue)         dst_addr : node.port (0.0)
d : drop      (at queue)
```

```
r 1.3556 3 2 ack 40 ----- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ----- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ----- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ----- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ----- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ----- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ----- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ----- 2 1.0 3.1 157 207
```

Προσομοίωση ασύρματων δικτύων (cont)

Χαρακτηριστικά δικτύου:

\$ns_ node-config

- topoInstance \$topo
- energyModel "EnergyModel"
- initialEnergy (in Joules)
- rxPower (in W)
- txPower (in W)
- agentTrace ON or OFF
- routerTrace ON or OFF
- macTrace ON or OFF
- movementTrace ON or OFF

Προσομοίωση ασύρματων δικτύων - Παράδειγμα

Define options

```
set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/TwoRayGround
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
set val(ifq)            Queue/DropTail/PriQueue
set val(ll)             LL
set val(ant)            Antenna/OmniAntenna
set val(ifqlen)         50
set val(nn)             2
set val(rp)             DSDV
```

Προσομοίωση ασύρματων δικτύων - Παράδειγμα (cont)

Initialize Global Variables

```
set ns_ [new Simulator]
set tracefd [open simple.tr w]
$ns_ trace-all $tracefd
```

set up topography object

```
set topo [new Topography]
```

```
$topo load_flatgrid 500 500
```

Create God

```
create-god $val(nn)
```

Προσομοίωση ασύρματων δικτύων - Παράδειγμα (cont)

configure node

```
$ns_ node-config -adhocRouting $val(rp) \  
    -llType $val(ll) \  
    -macType $val(mac) \  
    -ifqType $val(ifq) \  
    -ifqLen $val(ifqlen) \  
    -antType $val(ant) \  
    -propType $val(prop) \  
    -phyType $val(netif) \  
    -channelType $val(chan) \  
    -topoInstance $topo \  
    -agentTrace ON \  
    -routerTrace ON \  
    -macTrace OFF \  
    -movementTrace OFF
```

Προσομοίωση ασύρματων δικτύων - Παράδειγμα (cont)

Create nodes

```
for {set i 0} {$i < $val(nn) } {incr i} {  
    set node_($i) [$ns_ node]  
    $node_($i) random-motion 0  
}
```

Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes

```
$node_(0) set X_ 5.0
```

```
$node_(0) set Y_ 2.0
```

```
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 390.0
```

```
$node_(1) set Y_ 385.0
```

```
$node_(1) set Z_ 0.0
```

Προσομοίωση ασύρματων δικτύων - Παράδειγμα (cont)

```
# Now produce some node movements, Node_(1) starts to move towards node_(0)
```

```
$ns_ at 50.0 "$node_(1) setdest 25.0 20.0 15.0"
```

```
$ns_ at 10.0 "$node_(0) setdest 20.0 18.0 1.0"
```

```
# Node_(1) then starts to move away from node_(0)
```

```
$ns_ at 100.0 "$node_(1) setdest 490.0 480.0 15.0"
```

```
# Setup traffic flow between nodes. TCP connections between node_(0) and node_(1)
```

```
set tcp [new Agent/TCP]
```

```
$tcp set class_ 2
```

```
set sink [new Agent/TCPSink]
```

```
$ns_ attach-agent $node_(0) $tcp
```

```
$ns_ attach-agent $node_(1) $sink
```

```
$ns_ connect $tcp $sink
```

```
set ftp [new Application/FTP]
```

```
$ftp attach-agent $tcp
```

```
$ns_ at 10.0 "$ftp start"
```

Προσομοίωση ασύρματων δικτύων - Παράδειγμα (cont)

```
# Tell nodes when the simulation ends
```

```
for {set i 0} {$i < $val(nn)} {incr i} {
```

```
    $ns_ at 150.0 "$node_($i) reset";
```

```
}
```

```
$ns_ at 150.0 "stop"
```

```
$ns_ at 150.01 "puts \"NS EXITING...\" ; $ns_ halt"
```

```
proc stop {} {
```

```
    global ns_ tracefd
```

```
    $ns_ flush-trace
```

```
    close $tracefd
```

```
}
```

```
puts "Starting Simulation..."
```

```
$ns_ run
```

Κατασκευή ενός νέου πρωτ/λου:

Ορισμός πακέτου σχετιζόμενου με το πρωτόκολλο:

// Ορισμός δομής

```
struct hdr_ping {  
    char ret;  
    double send_time;  
};
```

// Ορισμός packet id

```
enum packet_t {  
    PT_TCP,  
    PT_UDP,  
    .....,  
  
    PT_TFRC,  
    PT_TFRC_ACK,  
    PT_PING, // packet protocol ID for our ping-agent  
    PT_NTTYPE  
};
```

Κατασκευή ενός νέου πρωτ/λου (cont):

C++ κώδικας που υλοποιεί το πρωτόκολλο:

```
void PingProtocol::recv(Packet* pkt, Handler*) {  
    // Access the IP header for the received packet:  
    hdr_ip* hdr_ip = (hdr_ip*)pkt->access(off_ip_);  
    // Access the Ping header for the received packet:  
    hdr_ping* hdr = (hdr_ping*)pkt->access(off_ping_);  
    ...  
    // Create a new packet  
    Packet* pktret = allocpkt();  
    ...  
    // Send the packet  
    send(pktret, 0);  
    ...  
    // Discard the packet  
    Packet::free(pkt);  
}
```

Κατασκευή ενός νέου πρωτ/λου (cont):

Κώδικας σχετιζόμενος με την διασύνδεση tcl scripts και πρωτοκόλλου:

```
int PingProtocol::command(int argc, const char*const* argv) {  
    if (argc == 2) {  
        if (strcmp(argv[1], "send") == 0) {  
            ..  
            return (TCL_OK);  
        }  
    }  
    return (Agent::command(argc, argv));  
}
```

Δημιουργία Poisson κίνησης

Δημιουργία Exponential μεταβλητής

```
set InterArrivalTime [new RandomVariable/Exponential]
$InterArrivalTime set avg_ [expr 1.0/$lambda]
set pktSize [new RandomVariable/Constant]
$pktSize set val_ 1000
```

Δημιουργία κόμβου πηγής

```
set src [new Agent/UDP]
$ns attach-agent $n1 $src
$src set packetSize_ 1000
```

Δημιουργία κόμβου προορισμού

```
set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $src $sink
```

Δημιουργία Poisson κίνησης (cont)

Συνάρτηση αποστολής ενός πακέτου ανά εκθετικά χρονικά διαστήματα

```
proc sendpacket {} {  
    global ns src InterArrivalTime pktSize  
    set time [$ns now]  
    $ns at [expr $time + [$InterArrivalTime value]] "sendpacket"  
    set bytes [$pktSize value]  
    $src send 1000  
}
```

Εκκίνηση συνάρτησης

```
$ns at 0.0001 "sendpacket"
```

NS-3

- ns-3 is a free, open source software project building and maintaining a discrete-event network simulator for research and education
- ns-3 is not backwards-compatible with ns-2
- However, several models already ported to ns-3 e.g., Random number generators, OLSR, error models, recent WiFi Phy models

OPNET

OPNET a set of commercial product addressing application performance management, network performance management, and network R&D

