Network Simulator 2: Simulazione di reti wireless 802.11

Marco Di Felice

[1]Università of Bologna
Dipartimento di Scienze dell'Informazione
Corso di Simulazione
mail: difelice@cs.unibo.it

Bologna, 23 Ottobre 2008

Network Simulator version 2 (Ns2)

Ns2 è un simulatore ad eventi discreti orientato al networking.

Ns2 è stato sviluppato all'interno del progetto VINT (Berkeley, 1989), nell'ambito dello sviluppo di tool per l'analisi e la simulazione di architetture di rete.

Sorgenti e Documentazione di Ns2:

`http://www.isi.edu/nsnam/ns`

Network Simulator version 2 (Ns2): Componenti

`http://www.isi.edu/nsnam/ns`

1. Il simulatore **ns**
2. Il tool **NAM** (Network Animator) per visualizzare l'output della simulazione
3. Strumenti di pre-processing
   - Generatore di traffico, e di topologie di rete
4. Strumenti di post-processing
   - Tool per analisi di tracce (awk, perl, ...)

Network Simulator version 2 (Ns2)

Ns2 è un simulatore ad eventi discreti orientato al networking.

Ns2 consente di simulare molte tipologie di reti IP (LAN/WAN) implementando:

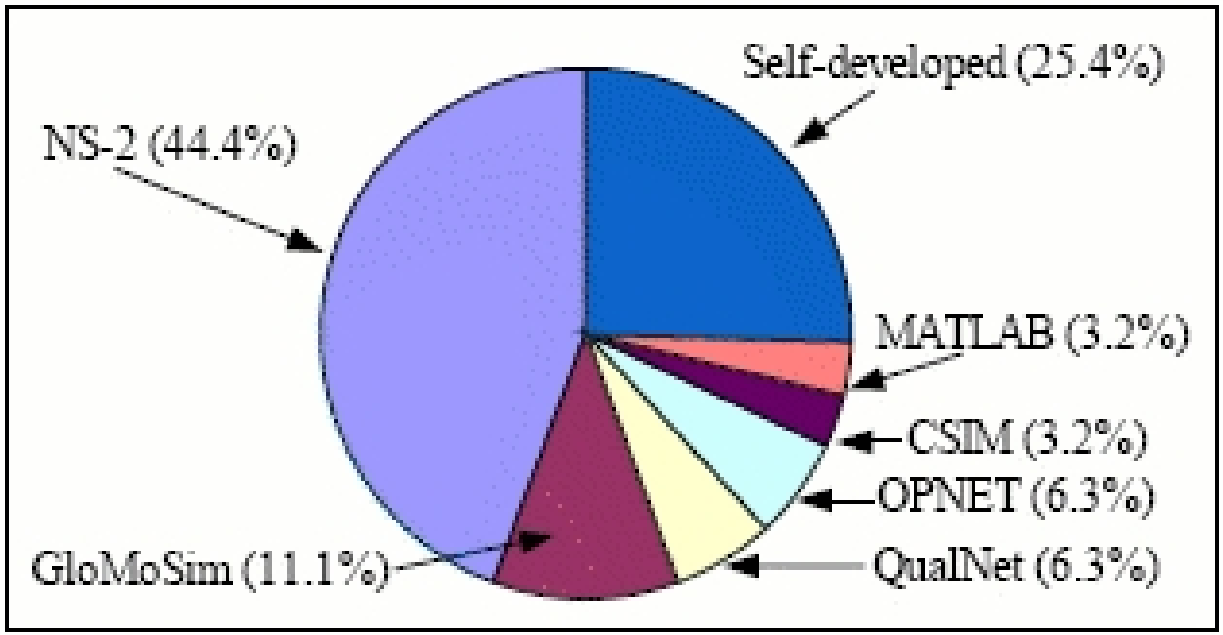
- protocolli di rete (MAC, routing, trasporto)
- modelli di sorgenti di traffico (es. CBR, FTP)
- meccanismi di gestione delle code (es. FIFO, RED)

Ns2 offre inoltre un supporto specifico per la simulazione di reti wireless 802.11 in modalità ad hoc/infrastructured.

Impiego di Ns2

Base Utente: 10K, Istituti: 1K

Statistica sui tool di simulazione utilizzati negli articoli scientifici presso la conferenza **MobiHoc** tra il 2000 ed il 2004.



S. Kurkowsky, T. Camp, M. Colagrosso, "**MANET Simulation Studies: The Current State and New Simulation Tools**"

Perchè utilizzare Ns2

Pro:

- Modularità
- Disponibilità di modelli di simulazione/documentazione
- Supporto specifico per molte tipologie di rete
- Costi d'utilizzo

Contro:

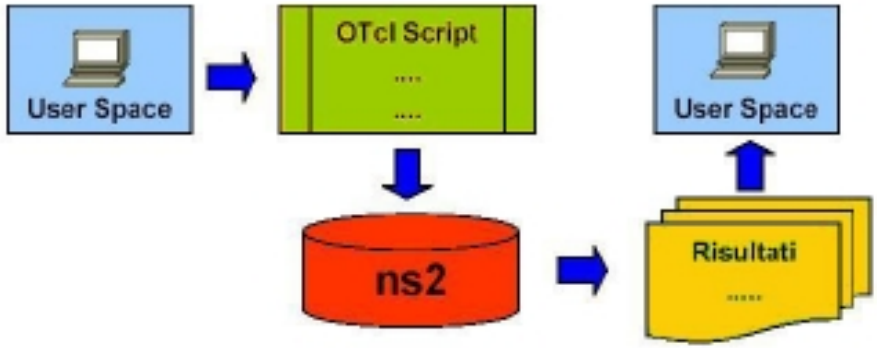
- Scalabilità
- Prestazioni (simulazione monolitica)
- Codice non commentato/non testato

Struttura

- Scritto in due linguaggi: OTCL e C++

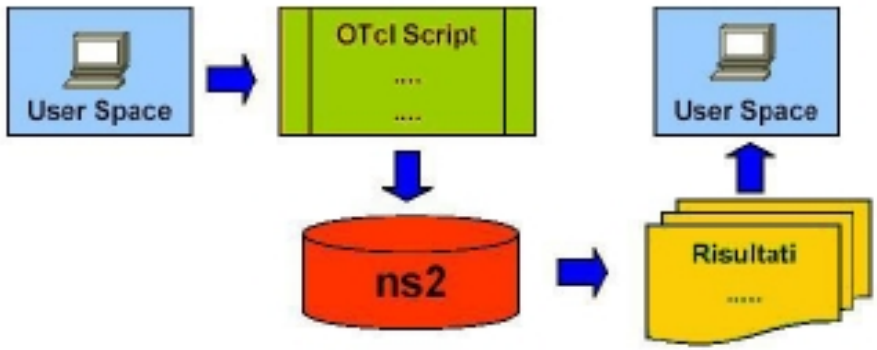
Struttura

- Scritto in due linguaggi: OTCL e C++



Struttura

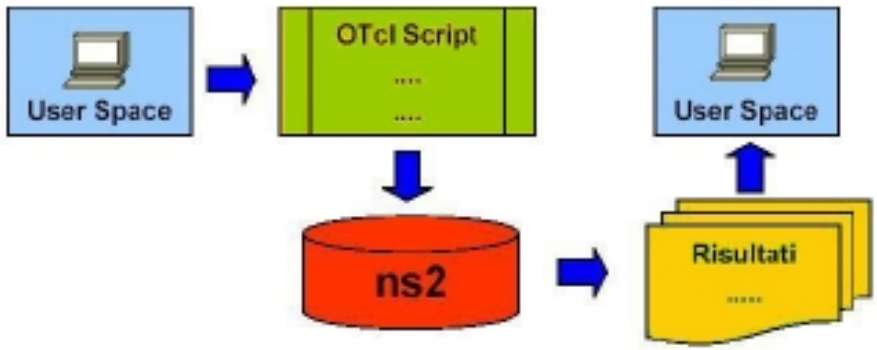
- Scritto in due linguaggi: OTCL e C++



- OTCL: interfaccia utente per la configurazione delle simulazioni

Struttura

- Scritto in due linguaggi: OTCL e C++



- OTCL: interfaccia utente per la configurazione delle simulazioni
- C++: implementazione dei componenti del simulatore e dei Network Object.

Network Object

Ns2 mette a disposizione una libreria molto ampia di modelli.

Ogni modello (Network Object) dispone di un'implementazione (in C++) e di una controparte in OTCL.

Modi di interazione dell'utente con il simulatore:

1. Creazione/modifica di Network Object (in C++)

Network Object

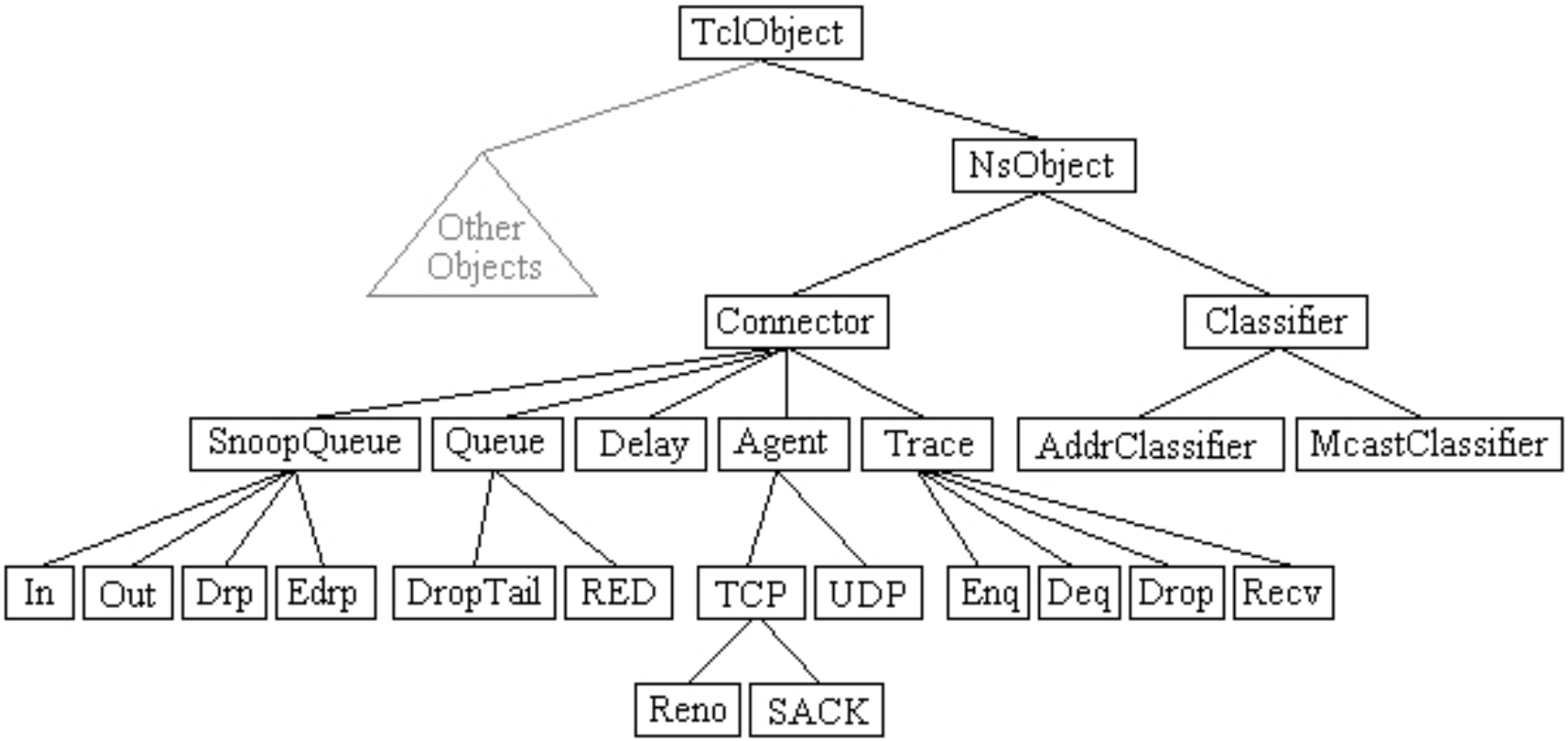
Ns2 mette a disposizione una libreria molto ampia di modelli.

Ogni modello (Network Object) dispone di un'implementazione (in C++) e di una controparte in OTCL.

Modi di interazione dell'utente con il simulatore:

1. Creazione/modifica di Network Object (in C++)
2. Composizione di Network Object in fase di configurazione della simulazione (tramite OTCL).

Gerarchia dei Network Object



Simulazione ad eventi discreti

La simulazione è **event-driven**.
Un evento può essere l'invio o la ricezione di un pacchetto da parte di una componente di rete oppure la gestione di un timer.

Struttura di un evento: <ID_, time_, handler_>

Lo **scheduler** del simulatore gestisce gli eventi del sistema in una lista ordinata. Azioni dello scheduler:

1. Prelevare l'evento successivo nella lista.

Simulazione ad eventi discreti

La simulazione è **event-driven**.
Un evento può essere l'invio o la ricezione di un pacchetto da parte di una componente di rete oppure la gestione di un timer.

Struttura di un evento: <ID_, time_, handler_>

Lo **scheduler** del simulatore gestisce gli eventi del sistema in una lista ordinata. Azioni dello scheduler:

1. Prelevare l'evento successivo nella lista.
2. Gestire l'avanzamento del tempo della simulazione.

Simulazione ad eventi discreti

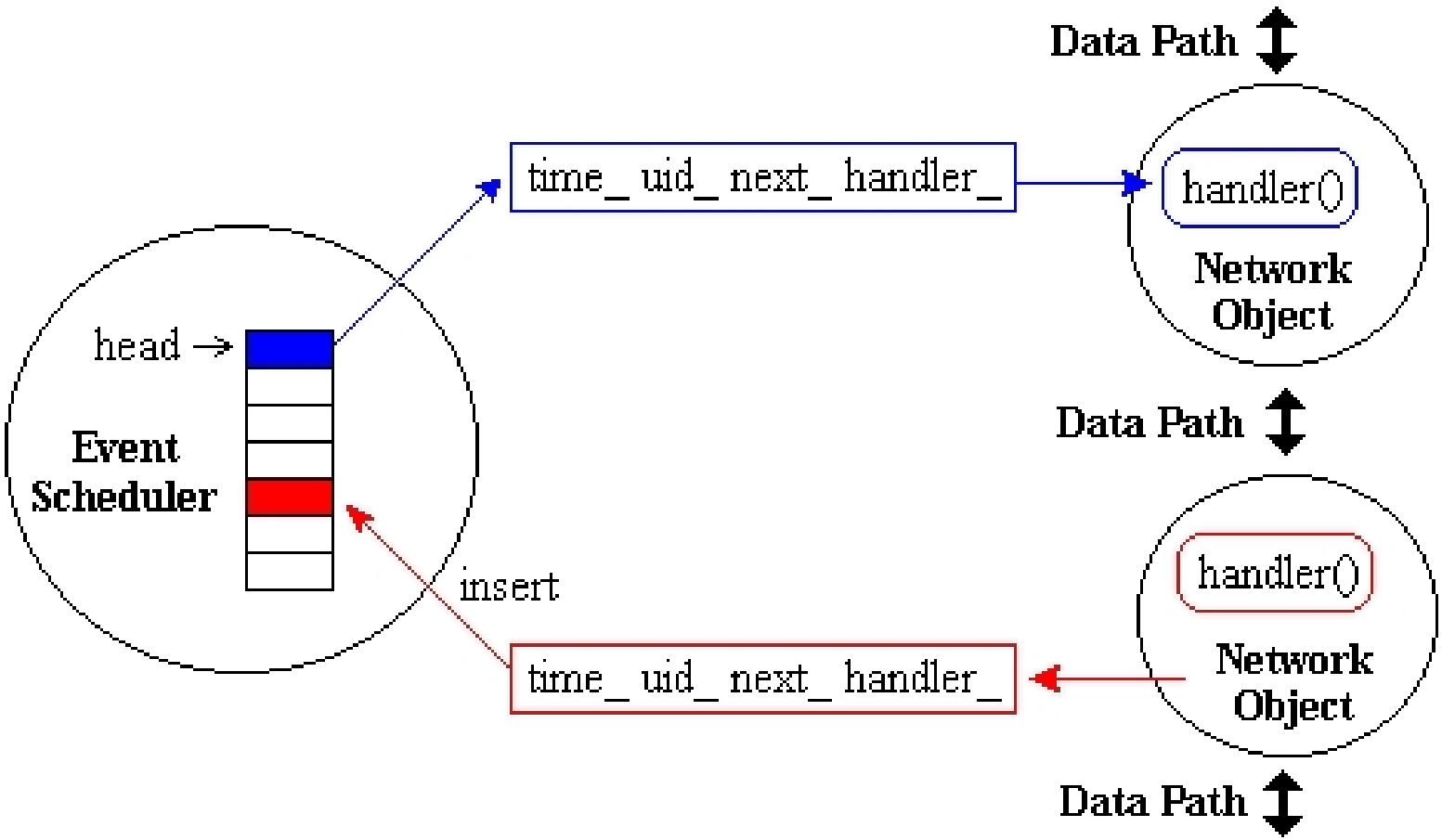
La simulazione è **event-driven**.
Un evento può essere l'invio o la ricezione di un pacchetto da parte di una componente di rete oppure la gestione di un timer.

Struttura di un evento: <ID_, time_, handler_>

Lo **scheduler** del simulatore gestisce gli eventi del sistema in una lista ordinata. Azioni dello scheduler:

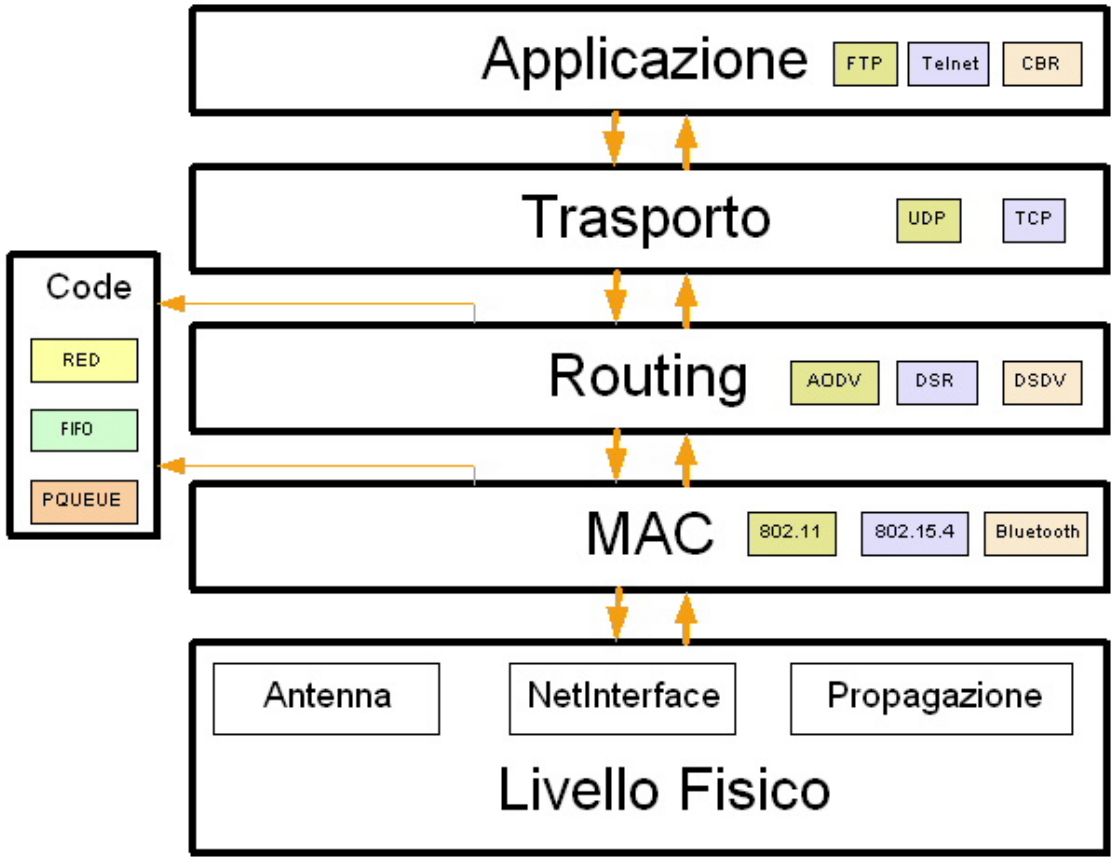
1. Prelevare l'evento successivo nella lista.
2. Gestire l'avanzamento del tempo della simulazione.
3. Richiamare l'handler opportuno dell'evento.

Simulazione ad eventi discreti



Simulazione di reti wireless

L'oggetto **MobileNode** implementa la struttura ISO/OSI di un dispositivo wireless mobile.



Usare ns2

Utilizzo generale:

```
ns script-file.tcl [parametri]
```

Lo script OTCL contiene:

- Inizializzazione dello scheduler di sistema

Usare ns2

Utilizzo generale:

```
ns script-file.tcl [parametri]
```

Lo script OTCL contiene:

- Inizializzazione dello scheduler di sistema
- Definizione dello stack protocollare

Usare ns2

Utilizzo generale:

```
ns script-file.tcl [parametri]
```

Lo script OTCL contiene:

- Inizializzazione dello scheduler di sistema
- Definizione dello stack protocollare
- Creazione della topologia della rete

Usare ns2

Utilizzo generale:

```
ns script-file.tcl [parametri]
```

Lo script OTCL contiene:

- Inizializzazione dello scheduler di sistema
- Definizione dello stack protocollare
- Creazione della topologia della rete
- Creazione delle sorgenti di traffico e delle connessioni tra nodi

Usare ns2

Utilizzo generale:

```
ns script-file.tcl [parametri]
```

Lo script OTCL contiene:

- Inizializzazione dello scheduler di sistema
- Definizione dello stack protocollare
- Creazione della topologia della rete
- Creazione delle sorgenti di traffico e delle connessioni tra nodi
- Impostazione dei parametri della simulazione (inizio, durata, output)

Il linguaggio OTCL

Estensione ad oggetti del linguaggio **TCL**

- Comando `set` per assegnare un valore ad una variabile (Es. `set x 0`).
- Keyword `$` per indicare il valore di una variabile (Es. `set temp $x`).
- Costrutti di selezione: `if` (`if` < *expr* > ... `else` ...)
- Costrutti iterativi: `for, while`
- Funzioni: `proc` (`proc` name {par1, ...parn} { ... return $x })
- Librerie varie (grafica, matematica, operazioni su file): (Es. `open:` `set file1 [open filename w]`)

Scheduler ed eventi

- Creazione dello scheduler di sistema:

```
set ns [new Simulator]
```

- Avvio della simulazione:

```
$ns run
```

- Scheduling di eventi:

```
$ns at <time> <event>
```

Es. Terminazione della simulazione dopo 300 secondi:

```
$ns at 300 "finish"
```

Avvio/Interruzione di una sorgente di traffico:

```
$ns at 0.4 "$cbr start"
$ns at 0.8 "$cbr stop"
```

Configurazione dei nodi mobili (1/3)

- Tipologia del canale:

```
set val(chan) Channel/WirelessChannel
```

- Modello di Propagazione (TwoRayGround, FreeSpace):

```
set val(prop) Propagation/TwoRayGround
```

- Network Interface:

```
set val(netif) Phy/WirelessPhy
Opt. Phy/WirelessPhy set Pt_ 2.07983391e-01
Opt. Phy/WirelessPhy set RXThresh_2.591168e-08
Opt. Phy/WirelessPhy set CSThresh_ 3.497734e-09
```

- Protocollo di Livello MAC:

```
set val(mac) Mac/Mac802.11
Opt. Mac/802_11 set CWMin_ 31
Opt. Mac/802_11 set CWMax_ 1023
```

Configurazione dei nodi mobili (2/3)

- Tipologia dell' Antenna (Direzionale,OmniDirezionale):
  `set val(ant) Antenna/OmniAntenna`
  Opt. `Antenna/OmniAntenna set Gt_ 1.0`
  Opt. `Antenna/OmniAntenna set Gr_ 1.0`
- Gestione delle code:
  `set val(ifq) Queue/DropTail/PriQueue`
  `set val(ifq) Queue/RED`
  Opt. `set val(ifqlen) 50.0`
- Link Layer Protocol:
  `set val(ll) LL`
  Opt. `LL set delay 50us`

Configurazione dei nodi mobili (3/3)

- Protocollo di Routing:

```
set val(rp) AODV
set val(rp) DSR
```

- Area di Simulazione:

```
set val(x) 500
set val(y) 500
```

Configurazione e creazione dei nodi con node-config:

```
$ns_ node-config -adhocRouting $val(rp)
                 -llType $val(ll)
                 -macType $val(mac)
                 - ...
                 -livello $val(livello)
```

Generazione del traffico e creazione delle connessioni

- La configurazione delle sorgenti di traffico comprende le fasi seguenti:
  1. Definizione delle sorgenti di traffico

- Sono disponibili diverse tipologie di traffico corrispondenti ad applicazioni (CBR, Telnet, FTP) e sorgenti di traffico differenti (traffico esponenziale, Pareto)
- In base alla tipologia di traffico, è possibile definire il protocollo di trasporto dati (TCP o UDP).

Generazione del traffico e creazione delle connessioni

- La configurazione delle sorgenti di traffico comprende le fasi seguenti:
  1. Definizione delle sorgenti di traffico
  2. Definizione del protocollo di trasporto

- Sono disponibili diverse tipologie di traffico corrispondenti ad applicazioni (CBR, Telnet, FTP) e sorgenti di traffico differenti (traffico esponenziale, Pareto)
- In base alla tipologia di traffico, è possibile definire il protocollo di trasporto dati (TCP o UDP).

Generazione del traffico e creazione delle connessioni

- La configurazione delle sorgenti di traffico comprende le fasi seguenti:
  1. Definizione delle sorgenti di traffico
  2. Definizione del protocollo di trasporto
  3. Creazione del **Datapath** tra l'agente applicazione/trasporto
- Sono disponibili diverse tipologie di traffico corrispondenti ad applicazioni (CBR, Telnet, FTP) e sorgenti di traffico differenti (traffico esponenziale, Pareto)
- In base alla tipologia di traffico, è possibile definire il protocollo di trasporto dati (TCP o UDP).

Generazione del traffico e creazione delle connessioni

- La configurazione delle sorgenti di traffico comprende le fasi seguenti:
  1. Definizione delle sorgenti di traffico
  2. Definizione del protocollo di trasporto
  3. Creazione del **Datapath** tra l'agente applicazione/trasporto
  4. Creazione della connessione tra il nodo sorgente/destinazione
- Sono disponibili diverse tipologie di traffico corrispondenti ad applicazioni (CBR, Telnet, FTP) e sorgenti di traffico differenti (traffico esponenziale, Pareto)
- In base alla tipologia di traffico, è possibile definire il protocollo di trasporto dati (TCP o UDP).

Esempio di connessioni (1/2)

Connessione **FTP** tra il nodo 0 ed il nodo 1.

```
# Setup a TCP connection at node 0
set tcp [new Agent/TCP]
# Setup an FTP connection at node 0
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns attach-agent $n0 $tcp
# Setup a TCP sink at node 1
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
```

```
$ns at 10.0 "$ftp start"
```

Esempio di connessioni (2/2)

Connessione **CBR** tra il nodo 0 ed il nodo 1.

```
# Setup a UDP connection at node 0
set udp [new Agent/UDP]
# Setup a CBR connection at node 0
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$ns attach-agent $n0 $udp
# Setup a UDP sink at node 1
set null [new Agent/Null]
$ns attach-agent $n1 $null
$ns connect $udp $null

$ns at 10.0 "$cbr start"
```

Creazione della Topologia della rete

- Posizionamento iniziale dei nodi

```
set $node_($ID) set X_ XVALUE
set $node_($ID) set Y_ XVALUE
set $node_($ID) set Z_ XVALUE
```

- Pattern di movimento:

```
ns at TIME "$node_($ID) setdest X,Y,Z,SPEED"
ns at 2.0 "$node_(0) setdest 1.0,1.0,0.0,4.0"
```

Creazione della topologia di rete

Esempio: Catena lineare di nodi.

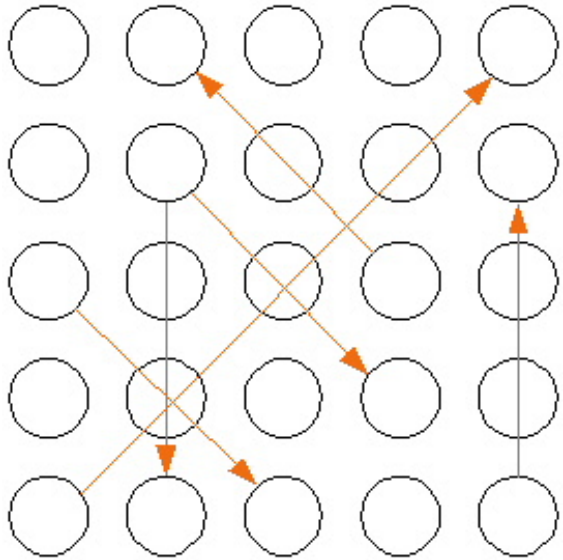
▸ Script OTCL completo.


```
set baseline 0
set distance 60
for {set i 0} {$i < $val(nn)} {incr i} {
  $node_($i) set X_ [expr $baseline]
  $node_($i) set Y_ 0.0
  $node_($i) set Z_ 0.0
  set baseline [expr $baseline + $distance]
}
```

Creazione della topologia di rete

Esempio: Mesh 5x5 di nodi.

```
set baseX 0
set baseY 0
for {set i 0} {$i<5} {incr i} {
  for {set j 0} {$i<5} {incr i} {
    $node_($i) set X_ [expr $baseX]
    $node_($i) set Y_ [expr $baseY]
    $node_($i) set Z_ 0.0
    set baseX [expr $baseX+50]
  }
  set baseY [expr $baseY+50]
  set baseX 0
}
```



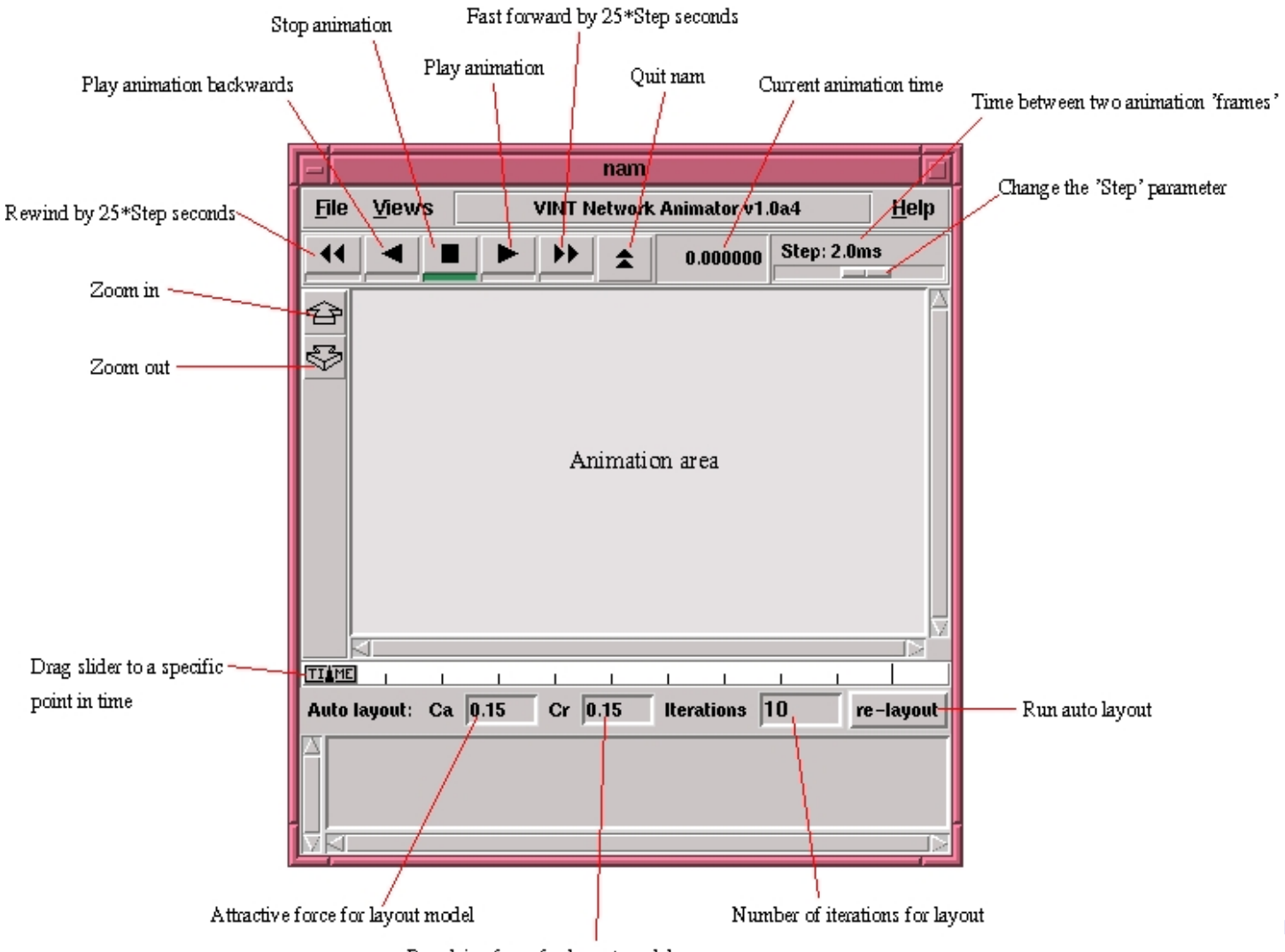
▸ Script OTCL completo.

Output delle simulazioni: File di traccia

```
s 10.00000 _0_ MAC --- 0 RTS 44 [253e 1 0 0]
r 10.00041 _1_ MAC --- 0 RTS 44 [253e 1 0 0]
s 10.00042 _1_ MAC --- 0 CTS 38 [2404 0 0 0]
r 10.00075 _0_ MAC --- 0 CTS 38 [2404 0 0 0]
s 10.00076 _0_ MAC --- 100 cbr 1112 [13a 1 0 800]
r 10.00982 _1_ MAC --- 100 cbr 1112 [13a 1 0 800]
```

- Il primo campo indica la tipologia dell'evento (spedizione, ricezione, drop).
- Il secondo rappresenta il tempo di simulazione, il terzo l'ID del nodo, il quarto il livello in cui è stato generato l'evento
- A seguire, sono collocate le informazioni relative all'header di livello MAC.

Output delle simulazioni: visualizzazione grafica



Analisi dei dati

Ns2 non mette a disposizione strumenti per effettuare l'analisi dei dati ed il calcolo degli indici di stima.

1. Esecuzione delle simulazioni con metodo delle prove ripetute

Analisi dei dati

Ns2 non mette a disposizione strumenti per effettuare l'analisi dei dati ed il calcolo degli indici di stima.

1. Esecuzione delle simulazioni con metodo delle prove ripetute
2. Estrazione degli indici di stima significativi rispetto ai parametri che si intende stimare.

Analisi dei dati

Ns2 non mette a disposizione strumenti per effettuare l'analisi dei dati ed il calcolo degli indici di stima.

1. Esecuzione delle simulazioni con metodo delle prove ripetute
2. Estrazione degli indici di stima significativi rispetto ai parametri che si intende stimare.
3. Calcolo delle medie, intervalli di confidenza, etc

Analisi dei dati

Ns2 non mette a disposizione strumenti per effettuare l'analisi dei dati ed il calcolo degli indici di stima.

1. Esecuzione delle simulazioni con metodo delle prove ripetute
2. Estrazione degli indici di stima significativi rispetto ai parametri che si intende stimare.
3. Calcolo delle medie, intervalli di confidenza, etc
4. Generazione dei grafici

Creazione dei modelli

L'implementazione del modello dipende dalle esigenze specifiche dell'utente/progettista di rete.

In generale, ogni nuovo Network Object deve:

- derivare dalla classe NsObject.
- disporre di un oggetto OTCL corrispondente.
- implementare i seguenti metodi:
  1. `recv(Packet* p, Handler* h)`:
     per gestire il flusso dei pacchetti con livelli adiacenti
  2. `command(int argc, const char*const* argv)`:
     per gestire il passaggio di parametri e l'interazione a run-time con il corrispondente OTCLObject.

▸ Implementazione del protocollo Mac 802.11DCF in Ns2.