



Simulatore di rete NS2

Roberto Petroccia
petroccia@di.uniroma1.it

Versione base delle slide fornite da: Prof.ssa Gaia Maselli



Riferimenti

- Architettura e utilizzo del Network Simulator NS2
 - <http://www.isi.edu/nsnam/ns/>
- Tutorials:
 - Marc Greis's tutorial
<http://www.isi.edu/nsnam/ns/tutorial/index.html>
 - "NS for Beginners" di Altma e Jimenez
 - Ns Manual
 - "NS by Example" di J. Chung e M. Claypool
<http://nile.wpi.edu/>



Installazione

<http://www.isi.edu/nsnam/ns/ns□build.html>

- All□in□one (Consigliata) Piattaforme supportate: Unix, Unix-like, Windows
- Ultima versione (17 Giugno, 2009)

<http://sourceforge.net/projects/nsnam/files/allinone/ns□allinone□2.34/>

Comprende diverse componenti:

Tcl release 8.4.18 (necessaria)

Tk release 8.4.18 (necessaria)

Otcl release 1.13 (necessaria)

TclCL release 1.19 (necessaria)

Ns release 2.34 (necessaria)

Nam release 1.13 (opzionale)

Xgraph version 12 (opzionale)

Cweb version 3.4g (opzionale)

SGB version 1.0(?) (opzionale)

Gt□itm e sgb2ns 1.1 (opzionale)

Zlib version 1.2.3 (opzionale, necessaria se si usa Nam)

- *Oppure...*

Installazione e configurazione separata dei singoli componenti



Outline

- Architettura del Network Simulator NS2
- Utilizzo di NS2
- Esempi di utilizzo del simulatore (protocolli UDP e TCP)



Network Simulator NS2

- Simulatore ad eventi discreti (sviluppato all'UC Berkley)
- Simulazione a livello di pacchetto
- Modellazione dal livello DataLink a livello Applicazione
 - Protocolli livello MAC
 - Algoritmi di routing: Dijkstra, etc...
 - Meccanismi di gestione delle code dei router: DropTail, Random Early Detection/Drop (RED) e (Class-Based Queueing) CBQ
 - Protocolli di rete: TCP, UDP (over IP e IPv6)
 - Sorgenti di traffico: FTP, Telnet, Web, CBR e VBR
- Codice sorgente di **pubblico dominio (open source)**
- In continua evoluzione, aggiornato e modificato da ricercatori e studenti di tutto il mondo
- Sito ufficiale: <http://www.isi.edu/nsnam/ns/>



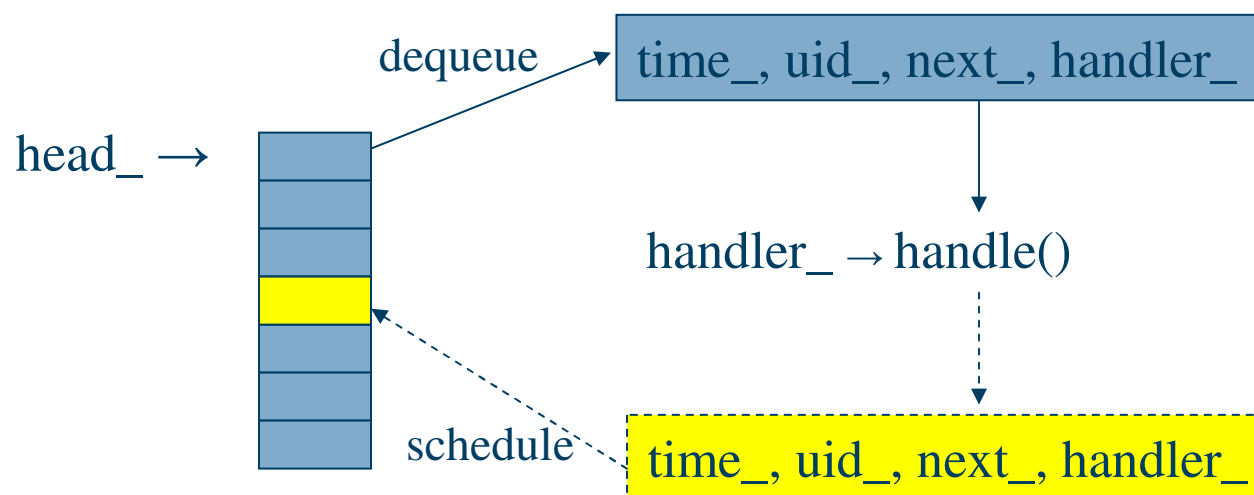
Network Simulator NS2

- Approccio **modulare**
- Implementato in ***tcl*** (Tool Command Language) e ***C++***
- Il linguaggio di scripting ***tcl*** è usato per eseguire i comandi dell'utente, ovvero per descrivere lo scenario simulativo
 - Configurare topologia, nodi, canale, e schedulare gli eventi
- Il linguaggio ***C++*** è usato per implementare il simulatore
 - Implementazione dei protocolli di rete (mac, network, transport, application)



Implementazione di NS2

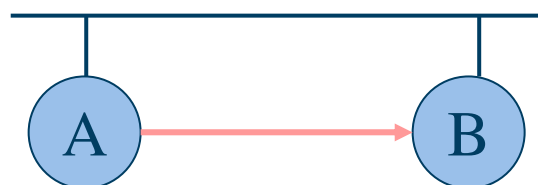
- Simulatore di rete
 - a eventi discreti (le variabili di stato assumono valori discreti)
 - con modello del tempo continuo (le variabili di stato sono sempre definite)
 - avanzamento del tempo event-driven
- Modellazione ad eventi
 - Lo scheduler
 - Mantiene la lista di eventi che devono essere eseguiti
 - Estrae il primo evento dalla coda e lo esegue invocando l'handler associato
 - Ogni evento è eseguito in un istante di tempo (simulato) virtuale, ma impiega una durata arbitraria di tempo reale
- NS usa un singolo thread di controllo





Esempio

Consideriamo due nodi A e B, con A che spedisce un pacchetto a B



modello
CSMA/CD

t=1.0:

- A invia il pacchetto alla NIC
- NIC di A inizia il carrier sense

t=1.005:

- NIC di A conclude il cs, e inizia la trasmissione

t=1.006:

- NIC di B inizia a ricevere il pacchetto

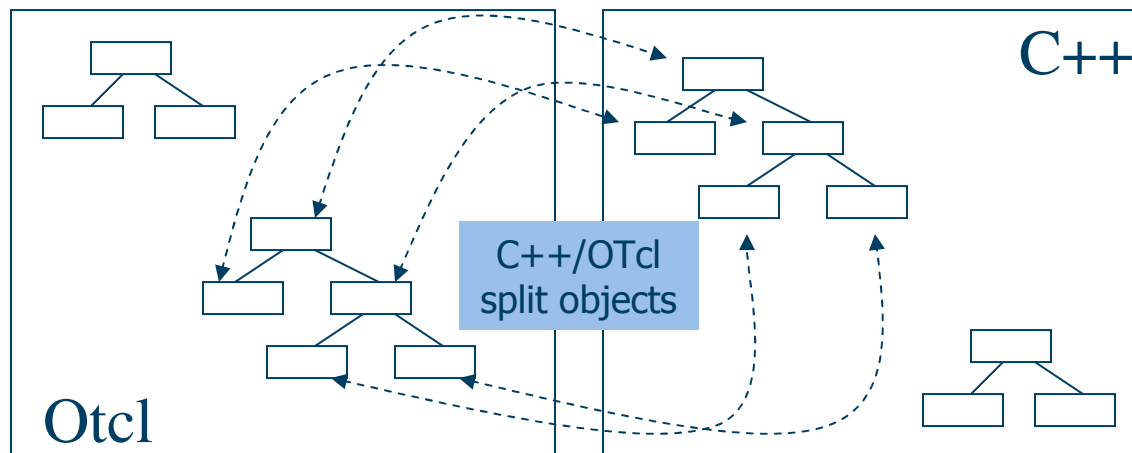
t=1.01:

- NIC di B conclude ricezione
- NIC di B passa il pacchetto all'applicazione



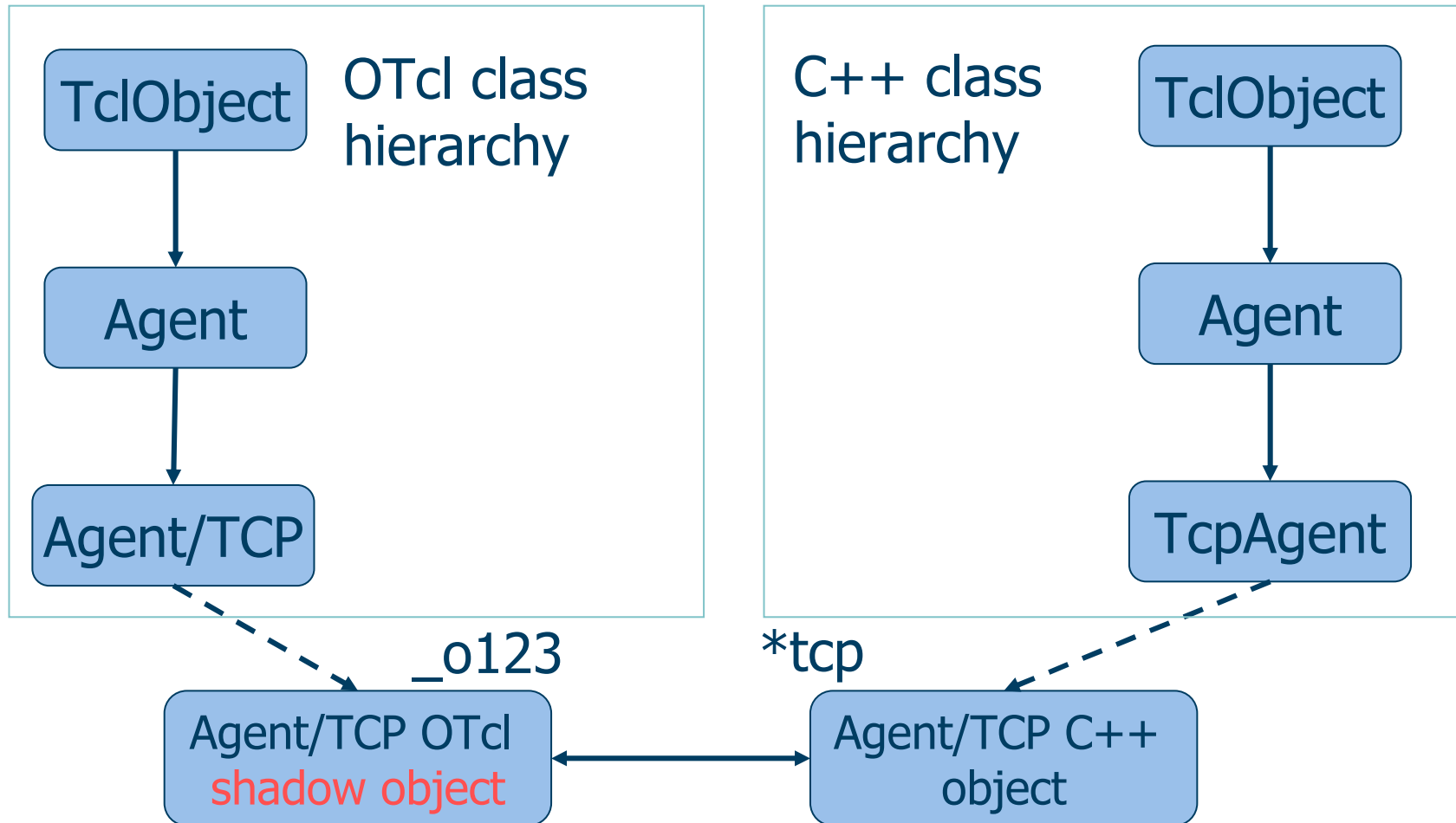
NS2: struttura orientata agli oggetti

- Il simulatore supporta una gerarchia di classi in C++ (*gerarchia compilata*), e una simile gerarchia di classi all'interno dell'interprete OTcl (*gerarchia interpretata*).
- Le due gerarchie sono strettamente legate l'una con l'altra; c'è una corrispondenza one-to-one tra una classe nella gerarchia interpretata e una nella gerarchia compilata.





Esempio: split object



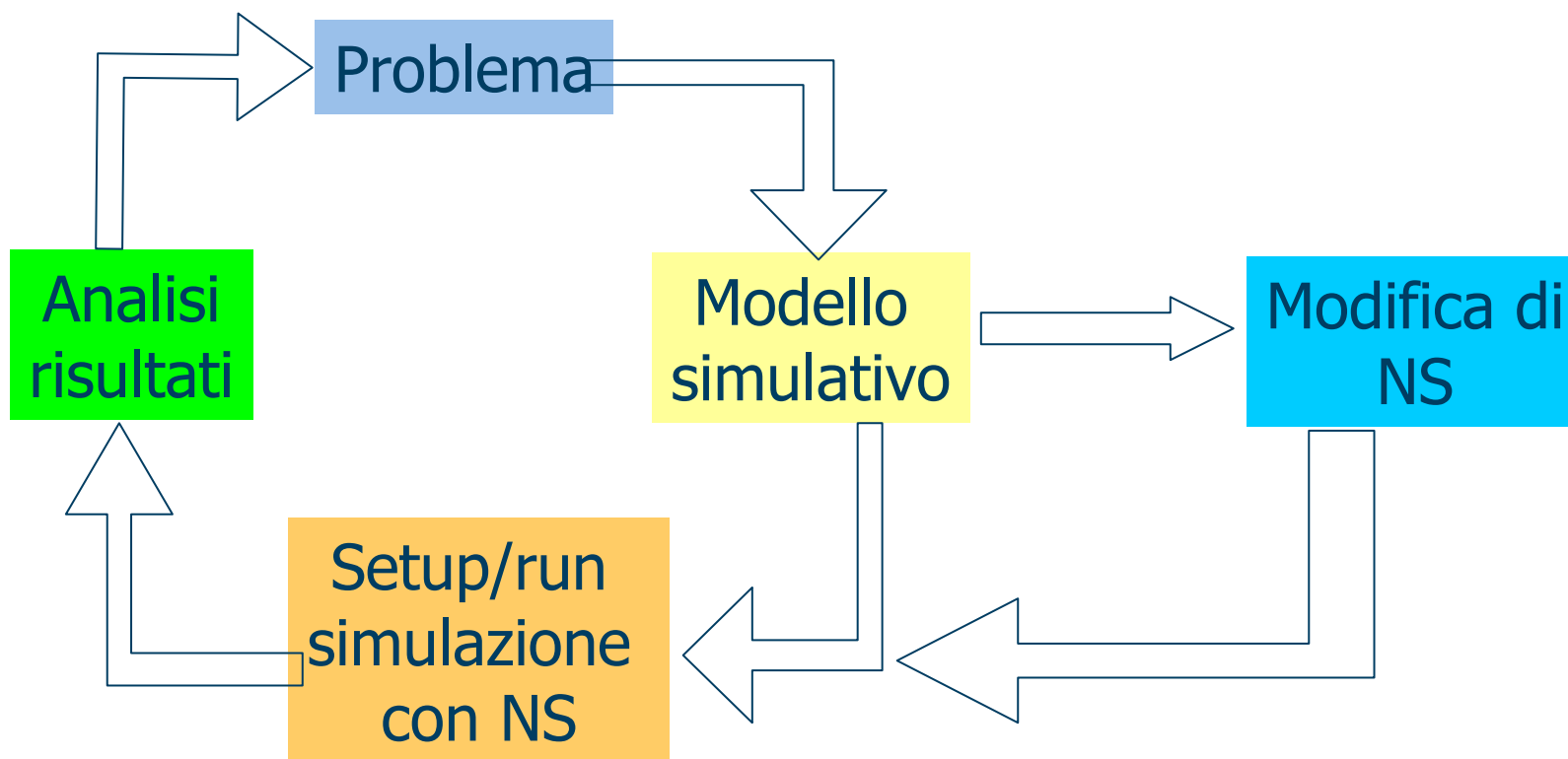


Outline

- Architettura del Network Simulator NS2
- **Utilizzo di NS2**
- Esempi di utilizzo del simulatore (protocolli UDP e TCP)

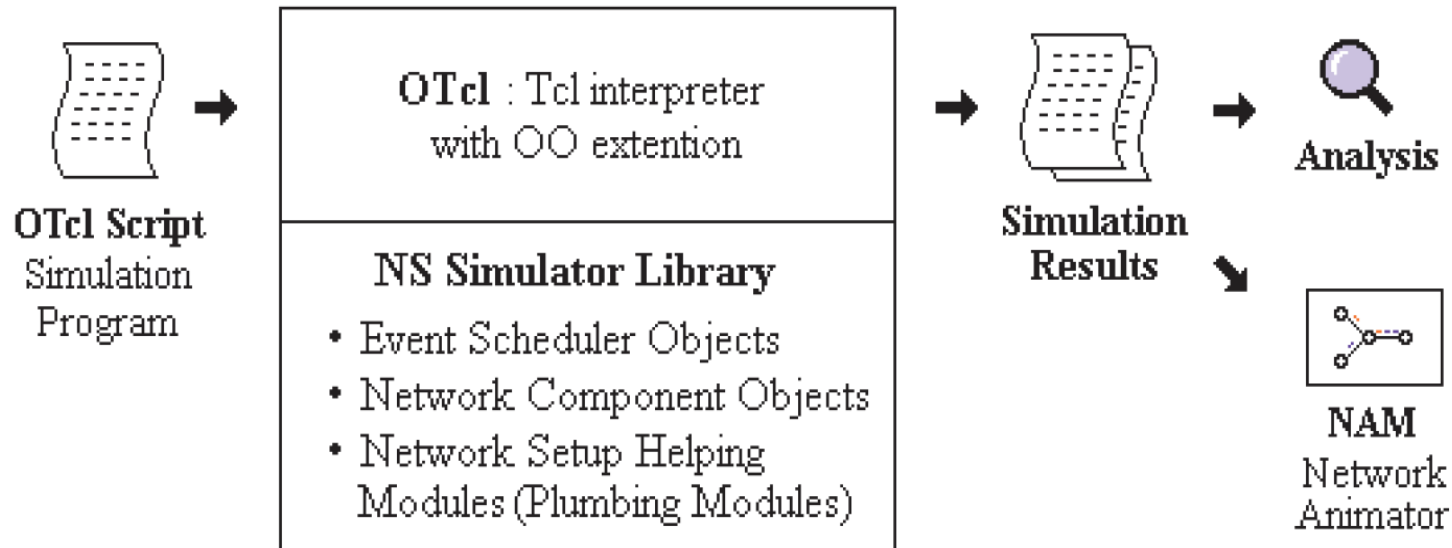


Usare NS2: fasi





Architettura di NS2



- Dove operare:
 - Tcl: script per costruire il modello di rete che si vuole simulare
 - C++: per implementare nuovi protocolli è necessario creare o modificare classi C++
- Due tipi di output
 - out.tr -> trace file per successiva elaborazione
 - out.nam -> file per visualizzazione grafica



Passi per eseguire la simulazione

1. Descrivere lo scenario simulativo in uno script tcl
 - Gestione del simulatore (inizializzazione e terminazione)
 - Definizione topologia (nodi, link)
 - Definizione degli agenti (TCP, UDP)
 - Definizione delle applicazioni (FTP, CBR)
 - Schedulazione degli eventi
 - Generazione file di trace

2. Eseguire la simulazione
 - NS interpreta lo script Otcl

3. Visualizzare e analizzare i risultati
 - Visualizzazione tramite "nam"
 - Analisi dei risultati (file di trace)

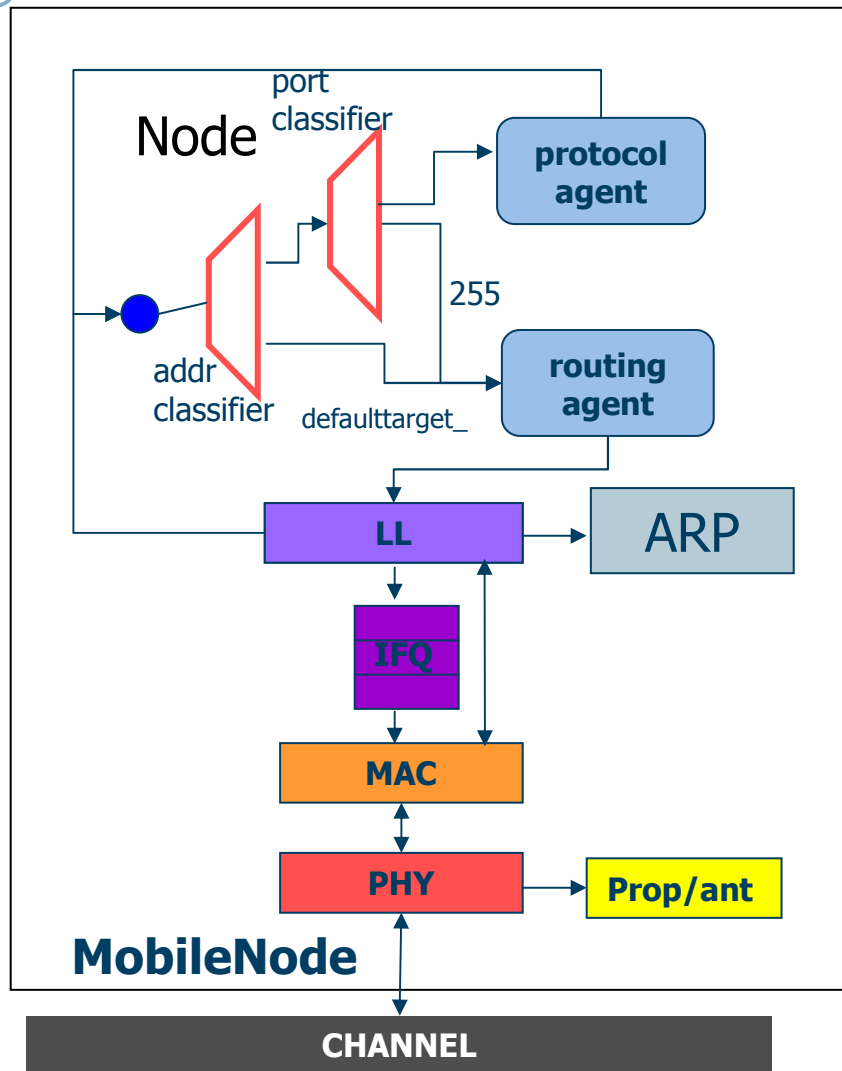


Descrivere lo scenario: TCL basics

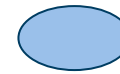
<pre>set b 0</pre>	b=0
<pre>set x \$a</pre>	x=a
<pre>set x [expr \$a+\$b]</pre>	x=a+b
<pre># comment</pre>	Commento
<pre>set file1 [open filename w]</pre>	Crea il file "file1"
<pre>puts</pre>	Stampa output
<pre>exec</pre>	Esegue un comando Unix
<pre>if {expression} { <execute some commands> } else { <execute some commands> }</pre>	Struttura comando if
<pre>for {set i 0} {\$i < 5} {incr i} { <execute some commands> }</pre>	Ciclo for



Struttura di un nodo



Classifier: Forwarding



Agent: Protocol Entity



Node Entry



LL: Link layer object



IFQ: Interface queue



MAC: Mac object



PHY: Net interface



Radio propagation/
antenna models



Configurazione e inizializzazione di una rete wireless

```
#-----  
#           --- Parametri del sistema di trasmissione ---  
#-----  
set val(chan)          Channel/WirelessChannel           ;# Tipo di canale\\  
set val(prop)          Propagation/TwoRayGround         ;# Mod. di propagazione\\  
set val(netif)         Phy/WirelessPhy                 ;# Tipo di interfaccia\\  
set val(mac)           Mac/802_11                      ;# Tipo di MAC \\  
set val(ifq)           Queue/DropTail/PriQueue         ;# Tipologia di coda \\  
set val(ll)            LL                              ;# Link Layer\\  
set val(ant)           Antenna/OmniAntenna             ;# Modello di antenna\\  
set val(ifqlen)        50                             ;# Num. di pacch. in IFQ\\  
set val(nn)            200                             ;# Numero di nodi\\  
set val(adhocRouting)  AODV                            ;# Protocollo di routing\\  
#-----  
#           --- Configurazione dei nodi ---  
#-----  
$ns node-config  
-adhocRouting $val(adhocRouting) \  
-llType $val(ll) \  
-macType $val(mac) \  
-ifqType $val(ifq) \  
-ifqLen $val(ifqlen) \  
-antType $val(ant) \  
-propType $val(prop) \  
-phyType $val(netif) \  
-channel $chan \  
-topoInstance $topo \  
-agentTrace OFF \  
-routerTrace OFF \  
-macTrace ON \  
-movementTrace OFF  
#-----
```



Definizione della topologia

- Creazione degli oggetti di base

- Simulatore

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

(creazione scheduler)

(riferito come \$ns)

- Nodi

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

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

(node è metodo di Simulator)

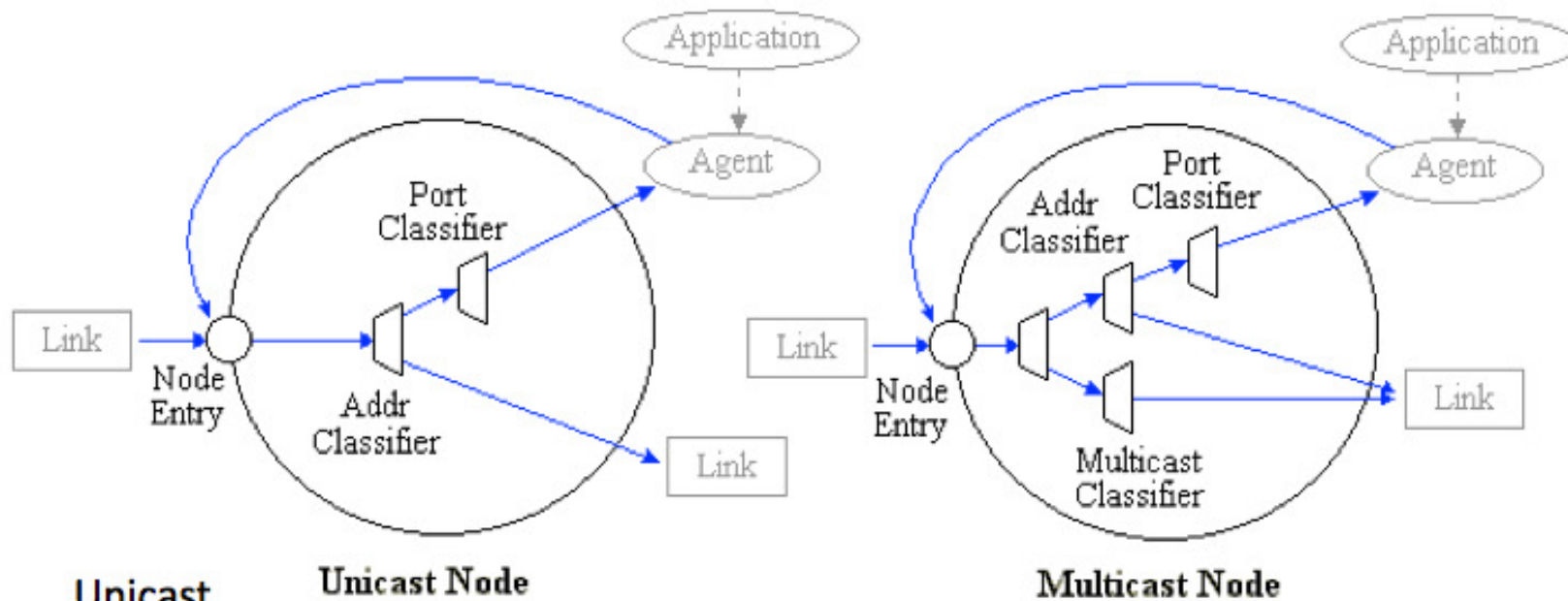
(riferite come \$n0, \$n1)



- Tipo di canale e quindi di link fa parte della definizione del nodo



Nodi e routing



Unicast

Unicast Node

- \$ns rproto type
(type: Static, Session, Distance Vector, cost, multi-path)

Multicast

- \$ns multicast (right after set \$ns [new Scheduler])
- \$ns mrtproto type
(type: CtrMcast, DM, ST, BST)



Definizione di agenti e applicazioni

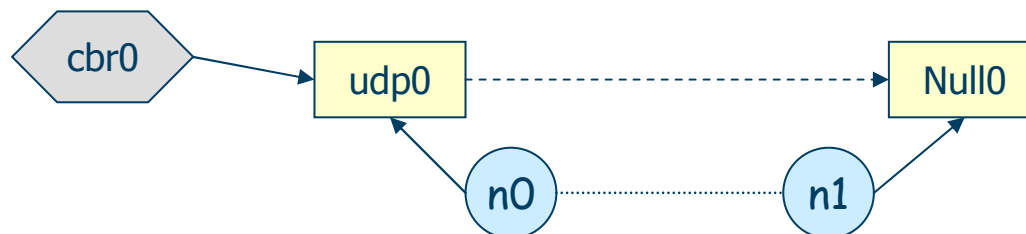
- Agenti (entità che rappresentano il livello di trasporto)

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set Null0 [new Agent/Null]
$ns attach-agent $n1 $Null0
$ns connect $udp0 $Null0
```



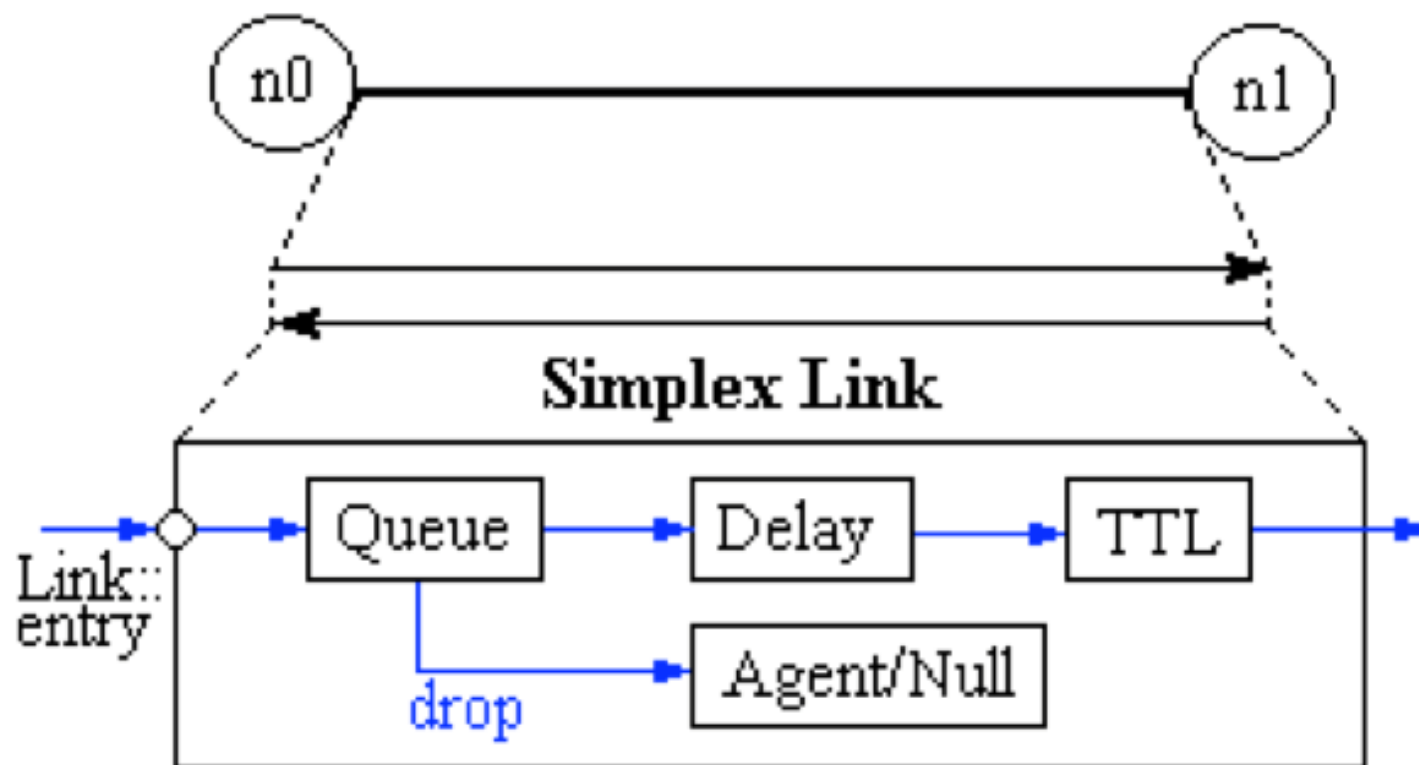
- Applicazioni (entità che generano il traffico)

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
```



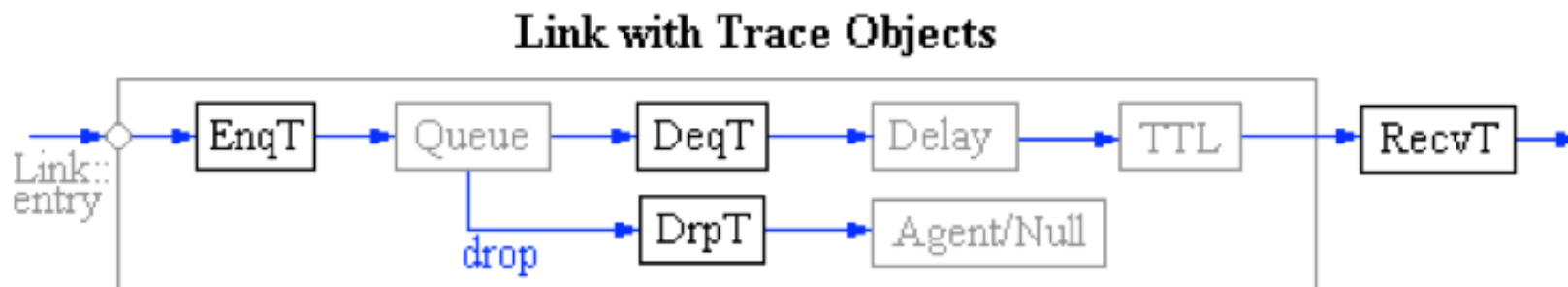


Link





Link



- `$ns trace-all file or`
- `$ns namtrace-all file`
- `$ns create-trace {type file src dst}`



Schedulazione degli eventi

- Lo scenario simulativo definito (topologia, agenti, e applicazioni) deve essere "animato"
- Stabilire *quando* eseguire gli eventi
- La maggior parte degli eventi sono nascosti all'utente, poiché generati da altri eventi
- Gli eventi vengono schedulati usando il comando

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

- Lo scheduler viene avviato tramite il comando

```
$ns run
```



Esempio di schedulazione degli eventi

- Schedulazione dell'avvio e terminazione di una applicazione CBR

```
$ns at 0.5 "$cbr0 start"
```

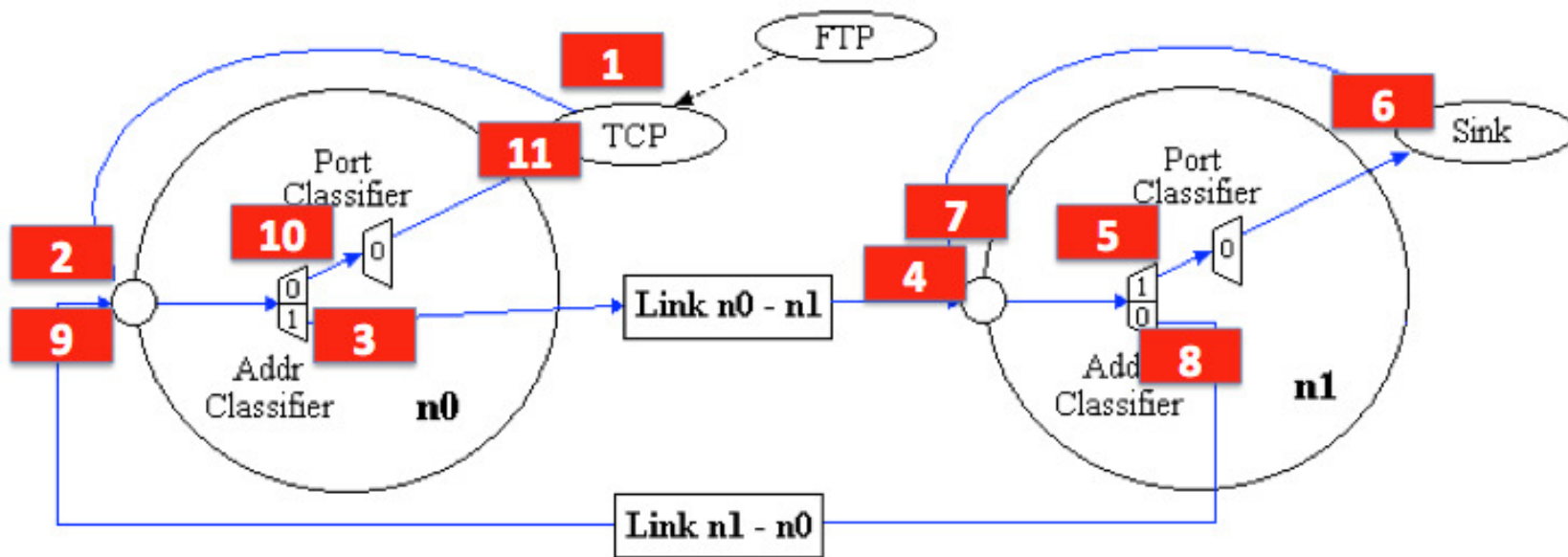
```
$ns at 5.5 "$cbr0 stop"
```

- Schedulazione di una procedura "finish" definita dall'utente

```
$ns at 150.0 "finish"
```




Packet flow





Generazione file di trace

```
#Open the Trace file  
Set tracefile1 [open out.tr w]  
$ns trace-all $tracefile1
```

Pointers to the trace files

```
#Open the NAM trace file  
Set namfile [open out.nam w]  
$ns namtrace-all $namfile
```

Trace files

`trace-all` e `namtrace-all` sono metodi della classe Simulator, per tracciare tutti gli eventi



Eseguire la simulazione

L'esecuzione della simulazione avviene facendo interpretare lo script Otcl a NS

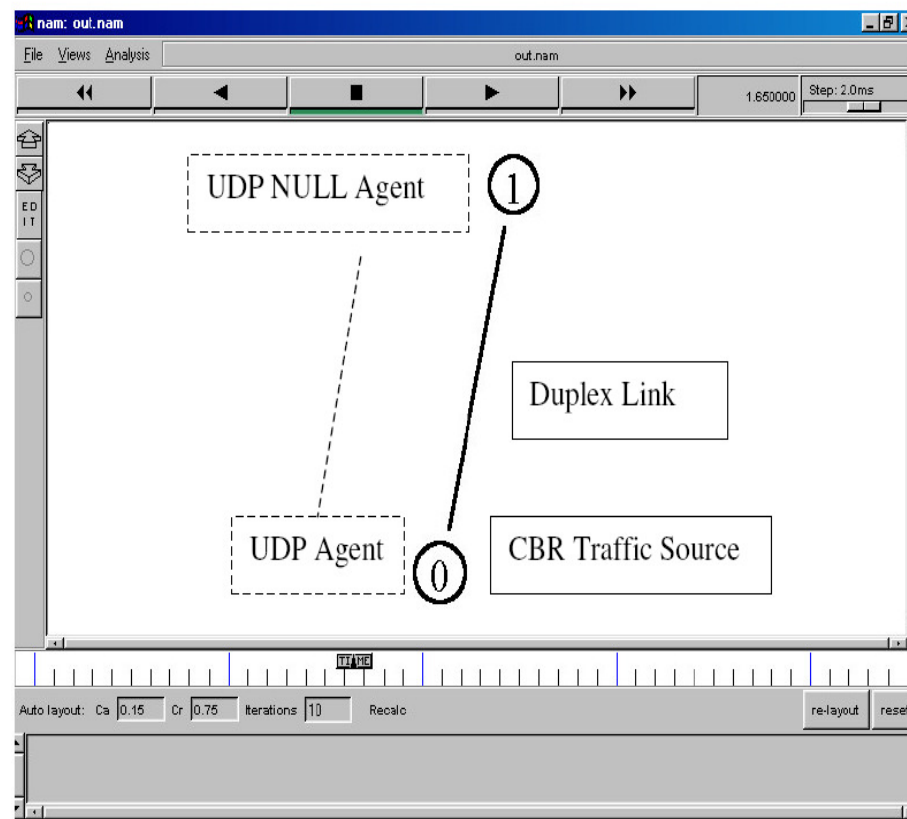
```
ns mio_script.tcl
```



Visualizzare i risultati: NAM

Animazione

NS genera un file di tracce (<nomefile>.nam) che permette di visualizzare un'animazione della simulazione mediante lo strumento NAM (Network Animator Module)





Analizzare i risultati

- NS produce file di trace contenute righe con il seguente formato:

<event> <time> <_node num_><layer> --- <seq. num> <pckt type> <pckt size> <mac info> ---<src dst ttl info><tcp info>

```
s 60.314477381 _2_ AGT --- 801 tcp 1040 [0 0 0 0] ----- [2:1 9:1 32 0] [1 0] 0 0
s 60.314477381 _2_ AGT --- 802 tcp 1040 [0 0 0 0] ----- [2:1 9:1 32 0] [2 0] 0 0
r 60.344950681 _9_ AGT --- 801 tcp 1060 [13a 9 a 800] ----- [2:1 9:1 254 9] [1 0] 2 0
r 60.355489347 _9_ AGT --- 802 tcp 1060 [13a 9 a 800] ----- [2:1 9:1 254 9] [2 0] 2 0
s 60.355489347 _9_ AGT --- 804 ack 40 [0 0 0 0] ----- [9:1 2:1 32 0] [2 0] 0 0
```

- Utilizzo

- **grep:** il comando unix `grep` permette di filtrare un file, creandone uno nuovo che contiene solo le righe contenenti una particolare sequenza di caratteri

ES. `grep "_2_" trace1.tr > trace2.tr`

produce un file contenente:

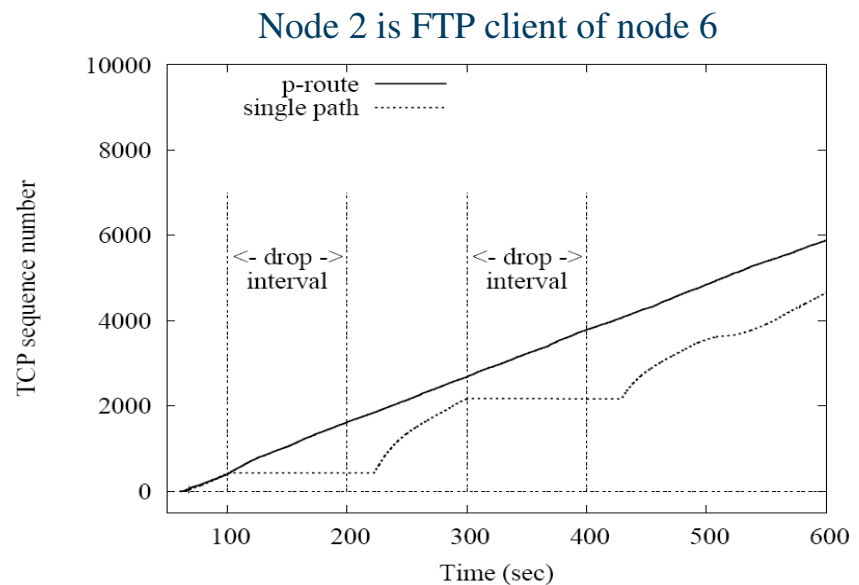
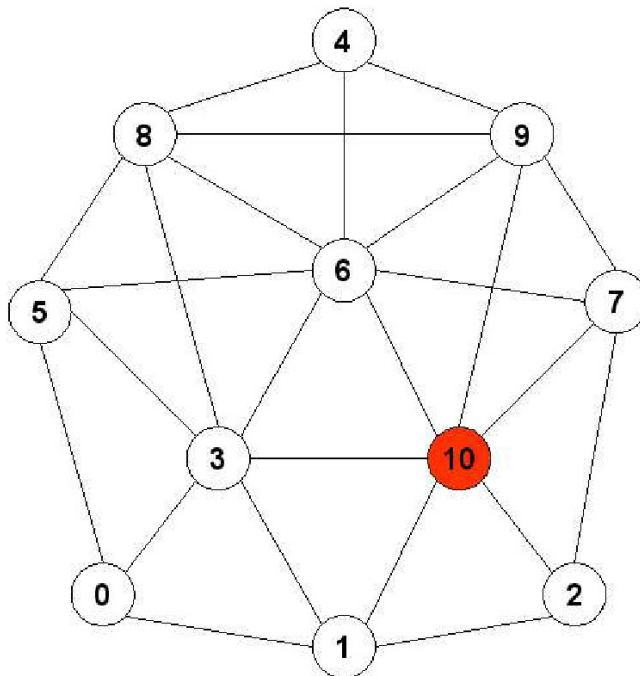
```
s 60.314477381 _2_ AGT --- 801 tcp 1040 [0 0 0 0] ----- [2:1 9:1 32 0] [1 0] 0 0
s 60.314477381 _2_ AGT --- 802 tcp 1040 [0 0 0 0] ----- [2:1 9:1 32 0] [2 0] 0 0
```

- **perl:** linguaggio di scripting che permette una facile ricerca e estrazione dei dati dai file di trace



Rappresentare i risultati

- Una volta filtrati i file di tracce è possibile costruire dei grafici con degli strumenti di plotting
 - gnuplot
 - xgraph
- Esempio





Outline

- Architettura del Network Simulator NS2
- Utilizzo di NS2
- Esempi di utilizzo del simulatore (protocolli UDP e TCP)



Esempio1.

Creazione di due nodi.

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Execute nam on the trace
    file
    exec nam -a out.nam &
    exit 0
}
```

```
#Create two nodes
set n0 [$ns node]
set n1 [$ns node]
#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms
DropTail
#Call the finish procedure after 5 seconds
of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```




Esempio2.

Creazione traffico CBR su link UDP

```
#Create a UDP agent and attach  
it to node n0  
set udp0 [new Agent/UDP]  
$ns attach-agent $n0 $udp0
```

```
# Create a CBR traffic source  
and attach it to udp0  
set          cbr0          [new  
Application/Traffic/CBR]  
$cbr0 set packetSize_ 500  
$cbr0 set interval_ 0.005  
$cbr0 attach-agent $udp0
```

```
#Create a Null agent (a traffic  
sink) and attach it to node n1  
set null0 [new Agent/Null]  
$ns attach-agent $n1 $null0
```

```
#Connect the traffic source with the  
traffic sink
```

```
$ns connect $udp0 $null0
```

```
#Schedule events for the CBR agent
```

```
$ns at 0.5 "$cbr0 start"
```

```
$ns at 4.5 "$cbr0 stop"
```

```
#Call the finish procedure after 5  
seconds of simulation time
```

```
$ns at 5.0 "finish"
```

```
#Run the simulation
```

```
$ns run
```



Esempio3.

Creazione traffico FTP su link TCP

```
#Create a TCP agent and attach  
it to node n0  
set tcp [new Agent/TCP]  
$tcp set class_ 2  
$ns attach-agent $n0 $tcp
```

```
set sink [new Agent/TCPSink]  
$ns attach-agent $n1 $sink  
$ns connect $tcp $sink  
$tcp set fid_ 1
```

```
#Setup a FTP over TCP  
connection  
set ftp [new Application/FTP]  
$ftp attach-agent $tcp  
$ftp set type_ FTP
```

```
#Schedule events for the CBR agent  
$ns at 0.5 "$ftp start"  
$ns at 4.5 "$ftp stop"
```

```
#Call the finish procedure after 5  
seconds of simulation time  
$ns at 5.0 "finish"  
#Run the simulation  
$ns run
```



Esempio4. 1/2

4 nodi con link UDP

```
#Create a simulator object
set ns [new Simulator]
#Define different colors for data
flows
$ns color 1 Blue
$ns color 2 Red
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Execute nam on the
trace file
    exec nam -a out.nam &
    exit 0
}
#Create four nodes
```

```
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

```
#Create links between the nodes
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms 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
```

```
#Monitor the queue for the link between node
2 and node 3
$ns queue-limit $n2 $n3 10
$ns duplex-link-op $n2 $n3 queuePos 0.5
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$ns attach-agent $n0 $udp0
```



Esempio4. 2/2

4 nodi con link UDP

```
# Create a CBR traffic source and
attach it to udp0
set          cbr0          [new
Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

```
#Create a UDP agent and attach it
to node n1
set udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1
# Create a CBR traffic source and
attach it to udp1
set          cbr1          [new
Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
```

```
#Create a Null agent (a traffic sink) and attach
it to node n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
#Connect the traffic sources with the traffic
sink
$ns connect $udp0 $null0
$ns connect $udp1 $null0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of
simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```



Esempio5.

4 nodi con link TCP e UDP

```
#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

```
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
```

```
#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
#Schedule events for the CBR and FTP
agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
#Detach tcp and sink agents (not really
necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns
detach-agent $n3 $sink"
#Call the finish procedure after 5 seconds of
simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```



Esempio6. 1/2

Più nodi con algoritmo di routing e guasto su link

```
#Create a simulator object
set ns [new Simulator]
#Tell the simulator to use dynamic routing
$ns rtproto DV
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Execute nam on the trace file
    exec nam -a out.nam &
    exit 0
}
#Create seven nodes
for {set i 0} {$i < 7} {incr i} {
    set n($i) [$ns node]
}
```

```
#Create links between the nodes
for {set i 0} {$i < 7} {incr i} {
    $ns duplex-link $n($i) $n([expr
($i+1)%7]) 1Mb 10ms DropTail
}
$ns duplex-link-op $n(0) $n(1) orient right-right-
down
$ns duplex-link-op $n(1) $n(2) orient down
$ns duplex-link-op $n(2) $n(3) orient down
$ns duplex-link-op $n(3) $n(4) orient left
$ns duplex-link-op $n(4) $n(5) orient left-up
$ns duplex-link-op $n(5) $n(6) orient up
$ns duplex-link-op $n(6) $n(0) orient right-right-up

#Create a UDP agent and attach it to node n(0)
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
```



Esempio6. 2/2

Più nodi con guasto su link

```
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n(3)
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent and the network dynamics
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```



Esempio7.

Utilizzo di xgraph

```
#Create a simulator object
set ns [new Simulator]
#Open the output files
set f0 [open out0.tr w]
set f1 [open out1.tr w]
set f2 [open out2.tr w]
#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#Connect the nodes
$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail
```

.....
.....

```
#Define a 'finish' procedure
proc finish {} {
    global f0 f1 f2
    #Close the output files
    close $f0
    close $f1
    close $f2
    #Call xgraph to display the results
    exec xgraph out0.tr out1.tr
    out2.tr -geometry 800x400 &
    exit 0
}
```




Esempi

- Script con diverse tecniche di congestion control (Tahoe, New Reno, Vegas)



Domande?