

Simulatore di rete NS2

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

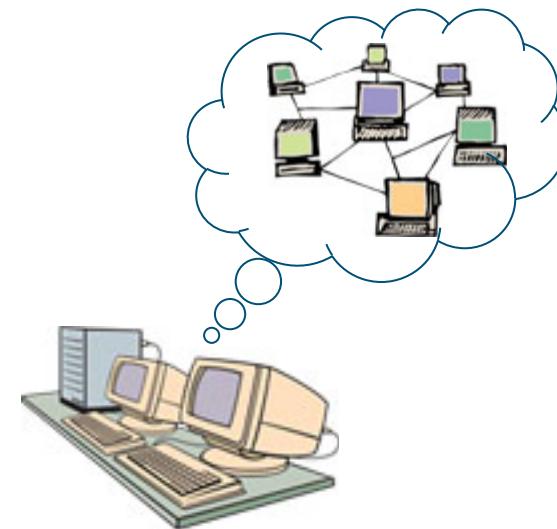
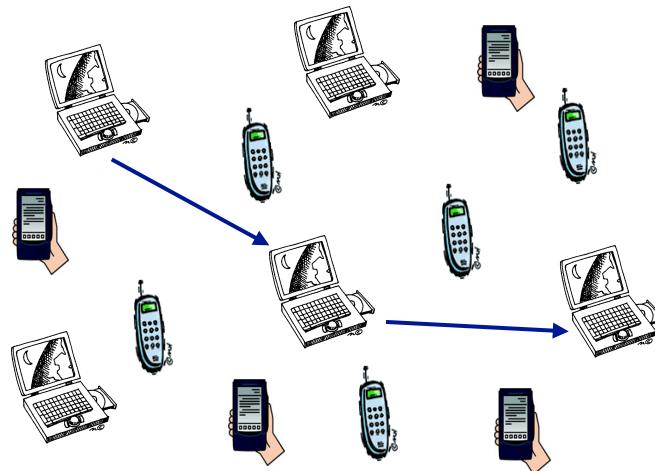


Outline

- Introduzione alla simulazione di rete
- Architettura del Network Simulator 2
- Utilizzo di NS-2



Introduzione alla simulazione



- Cosa è un simulatore di rete
 - Strumento software per la modellazione dei protocolli di rete (wired e wireless)
- Scopo:
 - Ricostruire un sistema che evolve come il sistema reale secondo alcuni aspetti, basandosi su un modello



Quando simulare

- Verifica di soluzioni analitiche
- Studio e sperimentazione delle interazioni interne di un sistema complesso (per es. TCP in sistemi wired e wireless)
- Valutazione delle prestazioni di un sistema prima della costruzione del prototipo
- Largamente diffuso in ambienti di ricerca
 - progettazione di nuovi protocolli
 - analisi del traffico
 - confronto tra protocolli
- Uso della simulazione per scopi didattici (comprendere meglio un sistema)



Perchè simulare

- Una sola workstation è necessaria per eseguire una simulazione
- Il simulatore permette di esaminare facilmente una ampia *varietà di scenari* in un tempo relativamente breve
- E' possibile simulare *topologie* di rete *complesse*, difficili e costose da realizzare in un test-bed
 - Ad-hoc o sensor networks di larga scala
 - Mobilità dei nodi
- Facile testare l'impatto di *modifiche* nei protocolli simulati



Pro e contro della simulazione

- Pro
 - Verifica del funzionamento di un nuovo sistema prima della costruzione del prototipo
 - Possibilità di eseguire un facile debugging del protocollo simulato
 - Possibilità di analizzare la scalabilità di un sistema
 - Identificazione delle vulnerabilità del sistema
 - Flessibilità nello studio del comportamento del sistema
- Contro
 - La creazione del modello e la sua validazione richiedono una comprensione dello strumento di simulazione
 - Non è possibile catturare svariati aspetti del sistema simulato (es. in NS2, prestazioni locali ai singoli nodi)



Outline

- Introduzione alla simulazione di rete
- Architettura del Network Simulator 2
- Utilizzo di NS-2



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/>



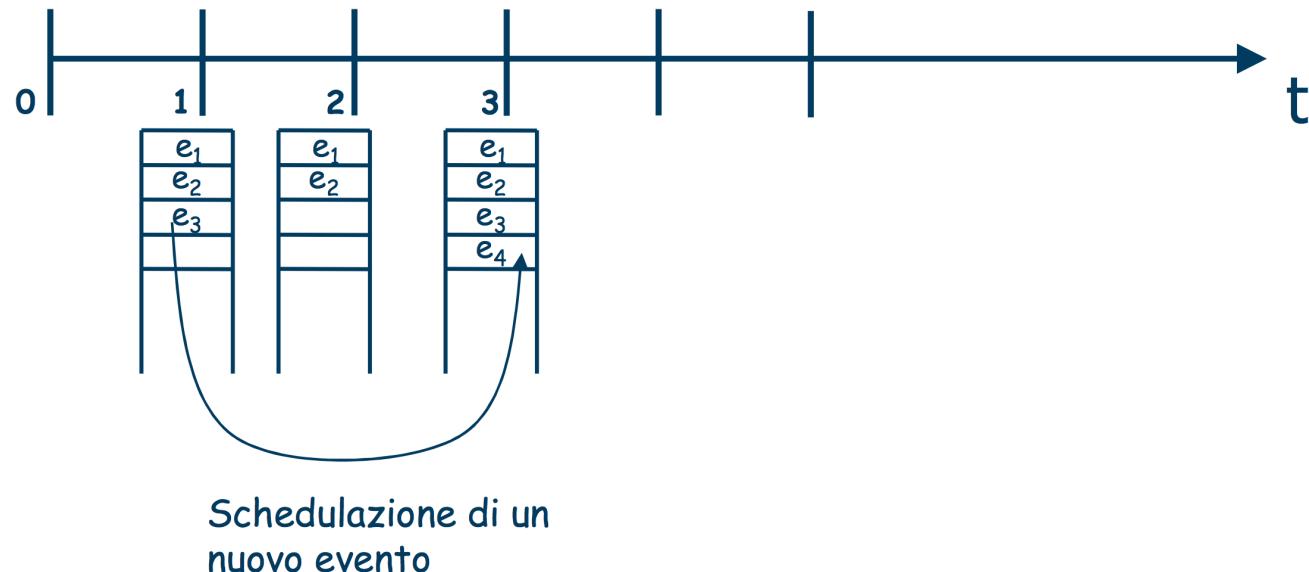
Network Simulator NS2

- 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, UPD (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/>
- Risorsa molto utile: <http://nile.wpi.edu/NS/>



Implementazione di NS2

- Simulatore di rete a eventi discreti
 - L'avanzare del tempo dipende dalla temporizzazione degli eventi, che sono gestiti da uno scheduler
 - Gli eventi rappresentano l'evoluzione temporale del sistema
 - Il sistema evolve in istanti appartenenti ad un insieme discreto
 - Il modello del tempo è continuo





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

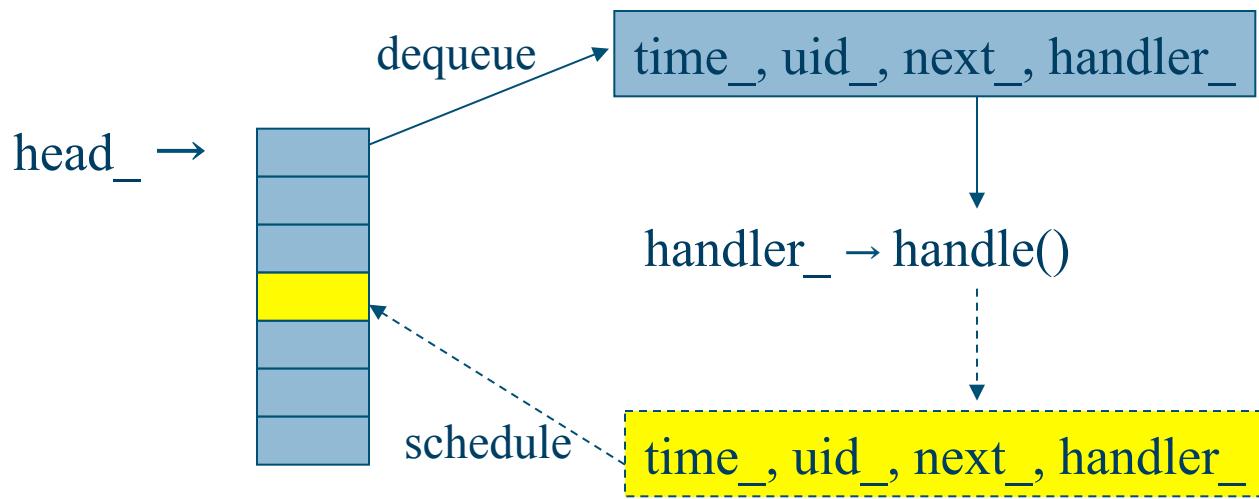
t=1.01:

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



Implementazione di NS2

- 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





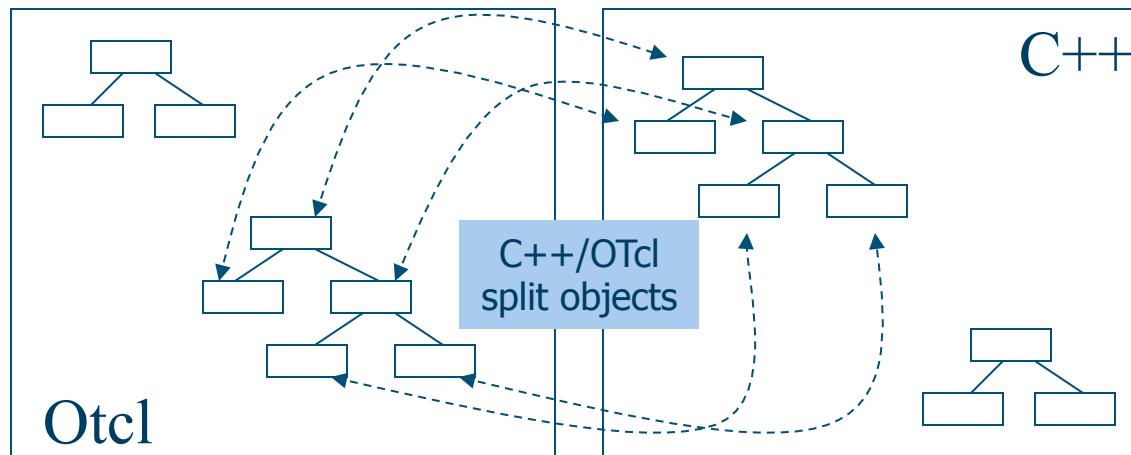
Network Simulator NS2

- 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)



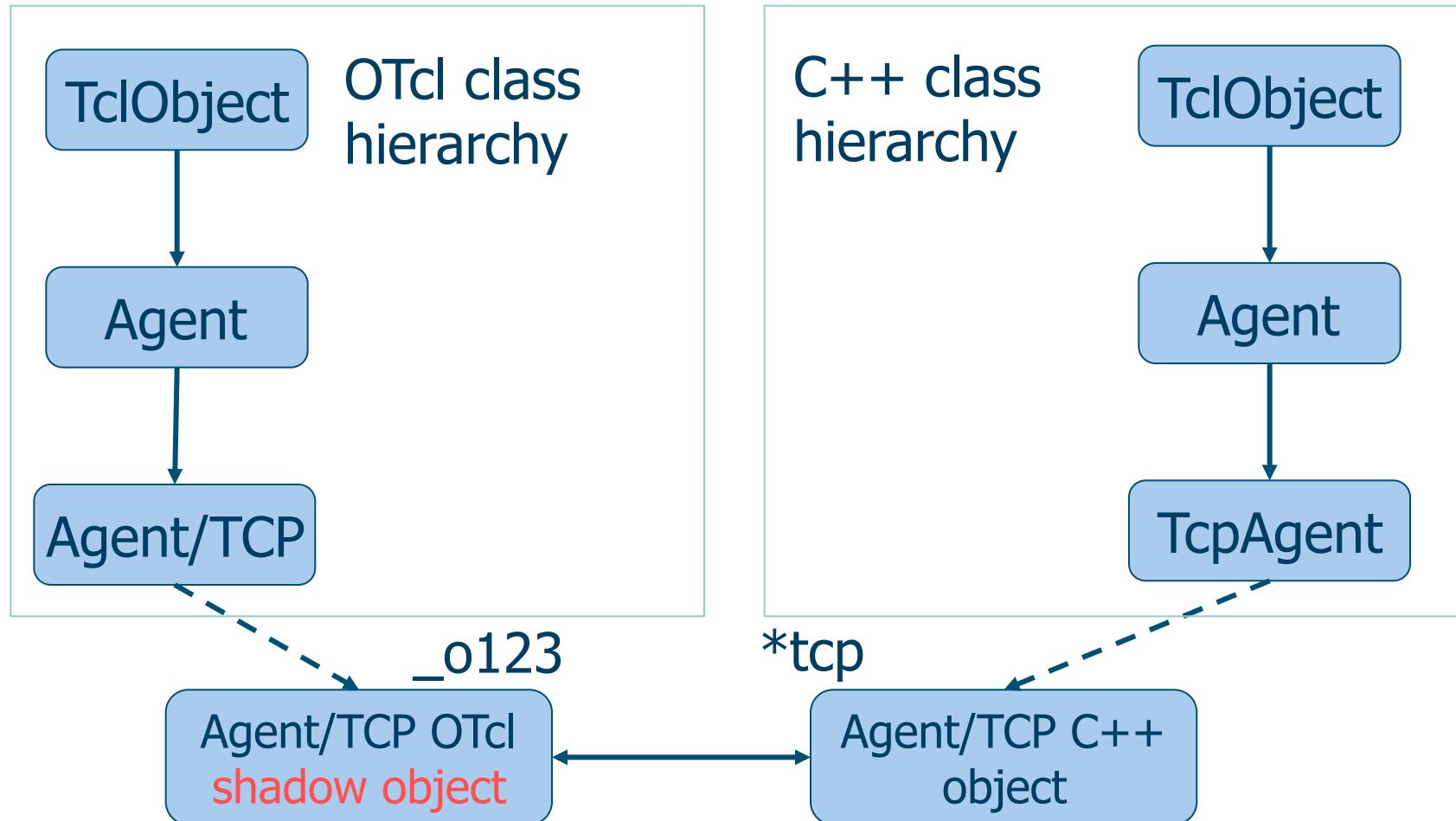
NS2: struttura orientata agli oggetti

- 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





Architettura di NS2

 →
OTcl Script
Simulation
Program

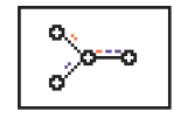
OTcl : Tcl interpreter
with OO extention

NS Simulator Library

- Event Scheduler Objects
- Network Component Objects
- Network Setup Helping Modules (Plumbing Modules)

→
 →
Analysis

Simulation
Results



NAM
Network
Animator

- 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

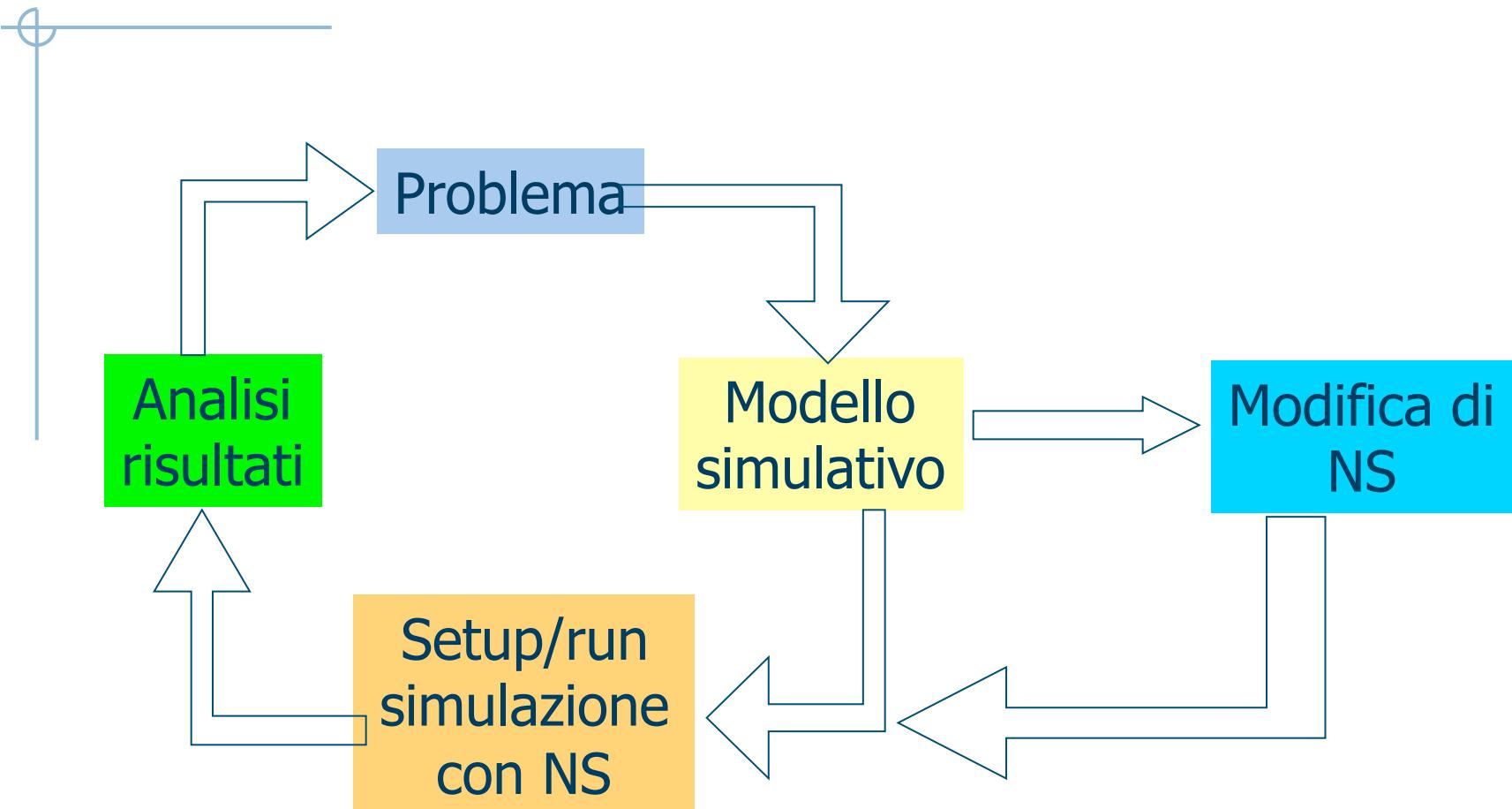


Outline

- Introduzione alla simulazione di rete
- Architettura del Network Simulator 2
- Utilizzo di NS-2



Usare NS2: fasi





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

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



Definizione della topologia

- Creazione degli oggetti di base

- Simulatore

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

(creazione scheduler)

(riferito come \$ns)

- Nodi

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

(node è metodo di Simulator)

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

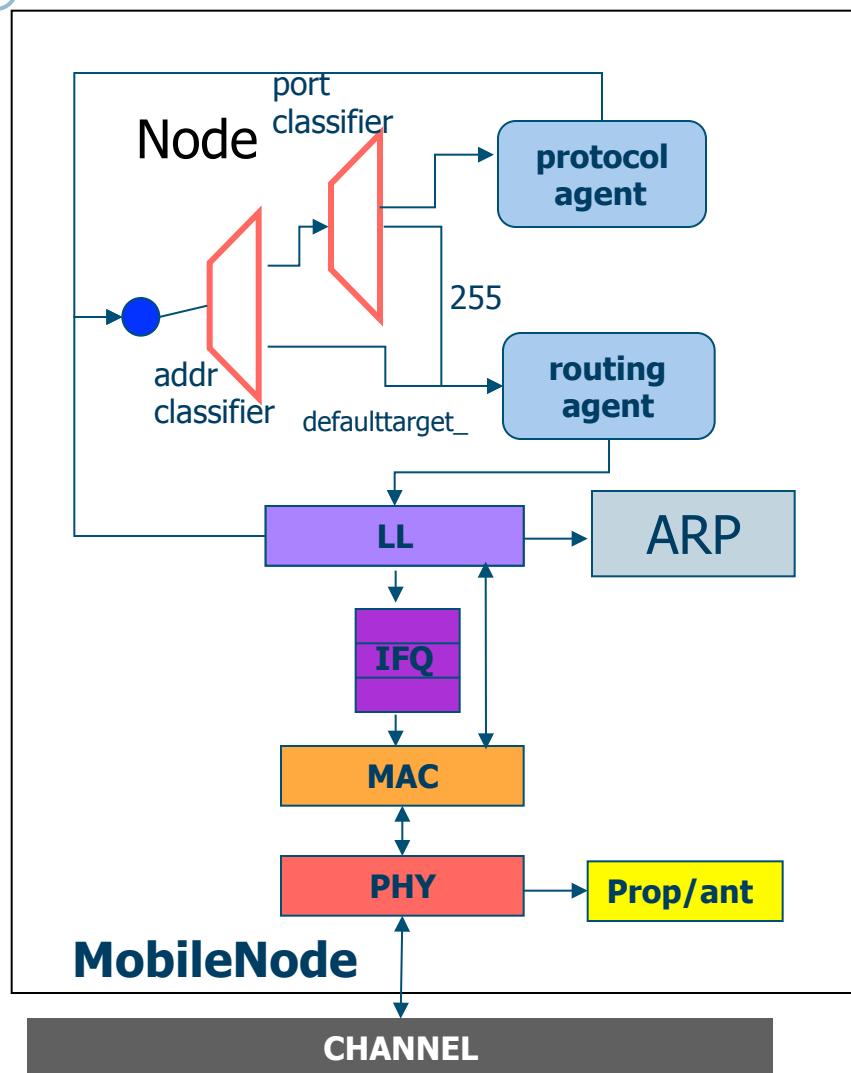
(riferite come \$n0, \$n1)



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



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

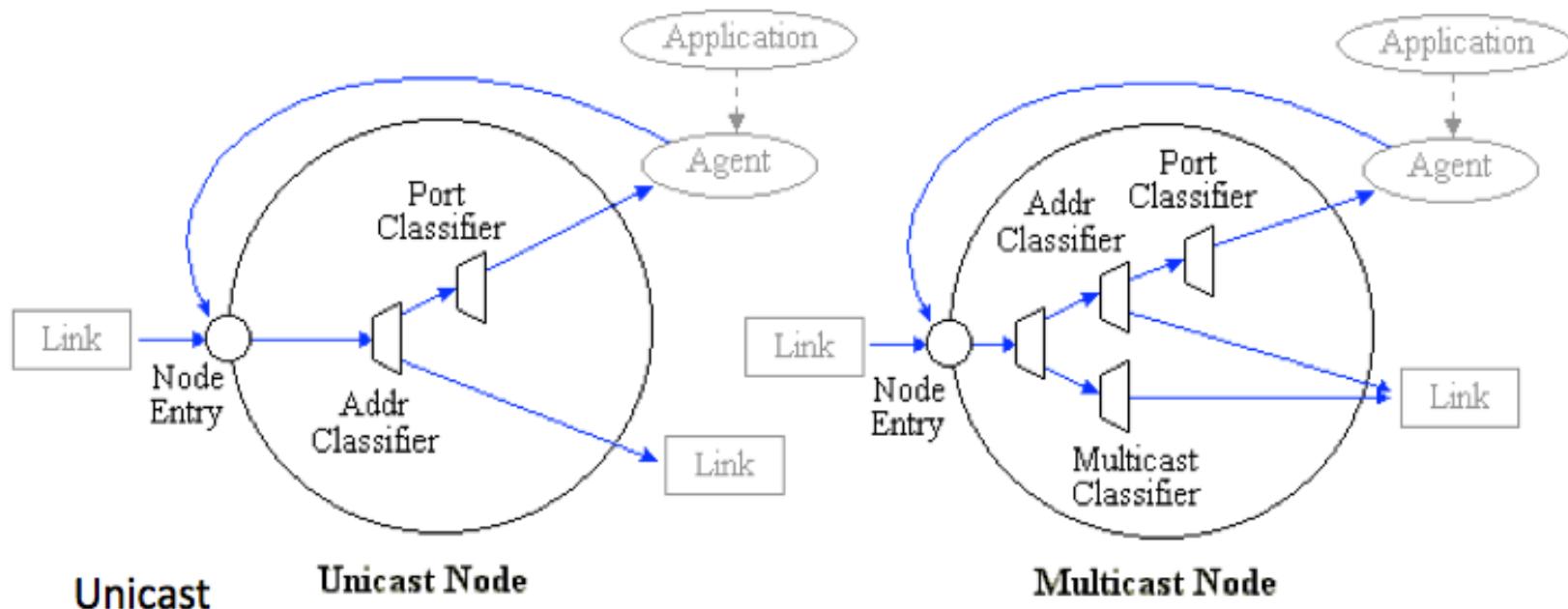


Configurazione e inizializzazione di una rete wireless

```
#-----  
# --- 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  
#-----
```



Nodi e routing



Unicast Unicast Node

Multicast Multicast Node

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

Multicast

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



Definizione di agenti e applicazioni

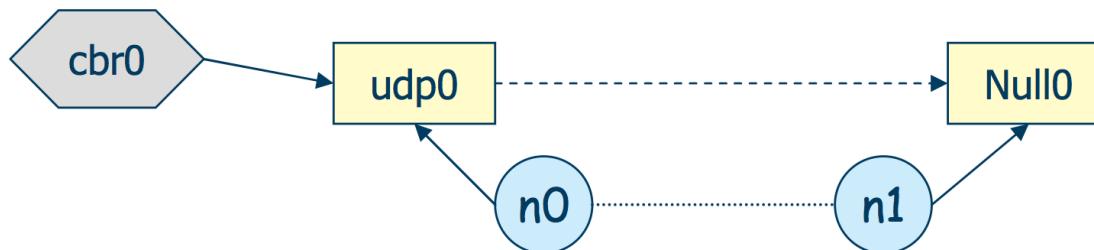
- Agenti (entità che rappresentano il livello di trasporto)

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



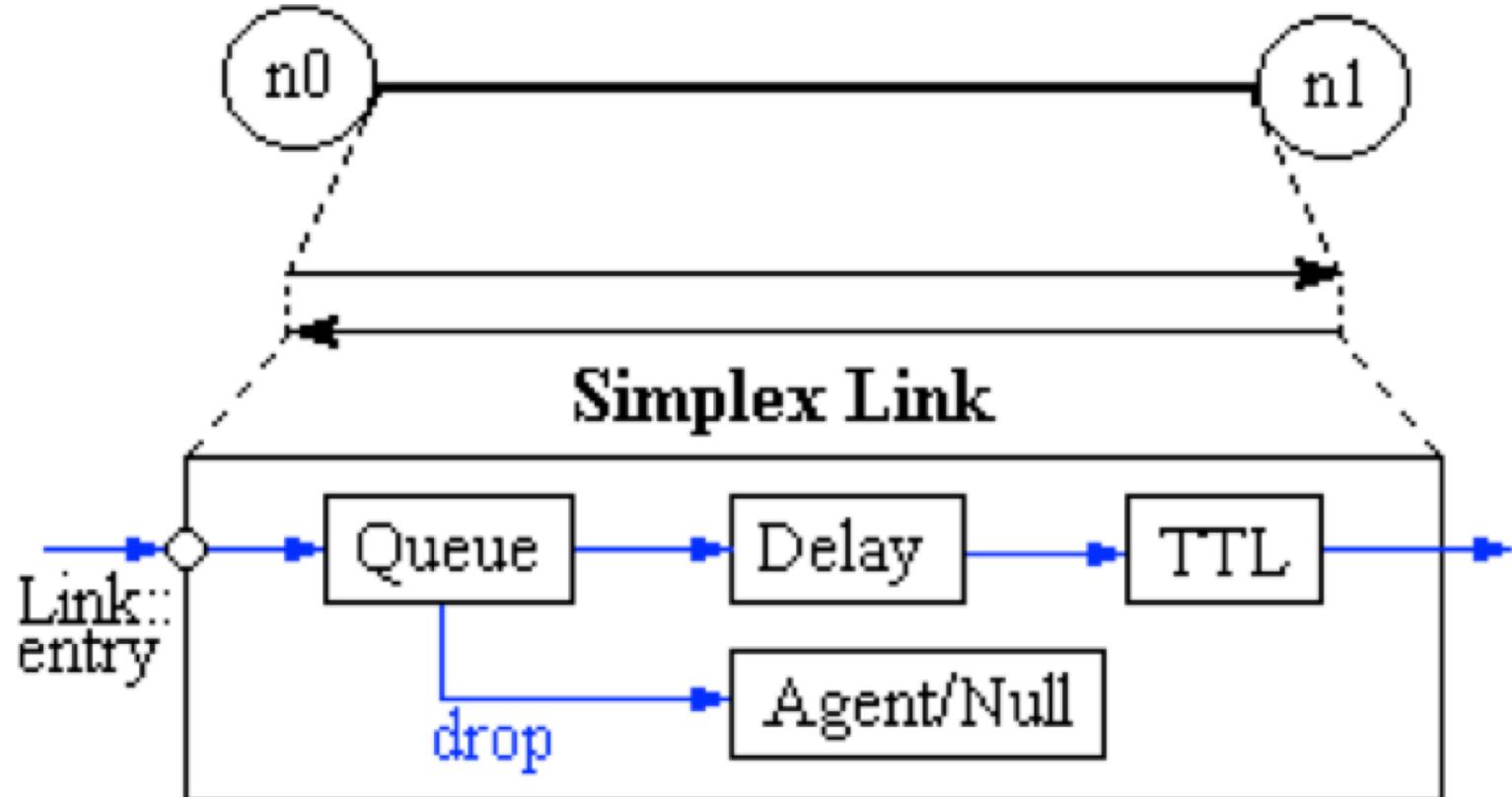
- Applicazioni (entità che generano il traffico)

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





Link

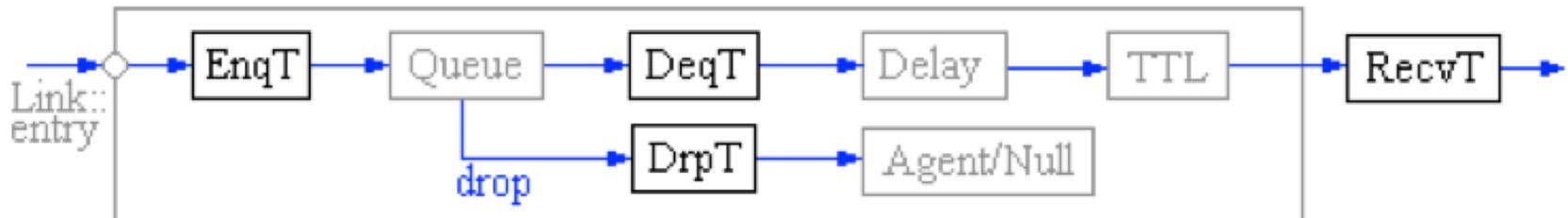


```
$ns duplex-link $n0 $n1 2Mb 10ms DropTail
```



Link

Link with Trace Objects



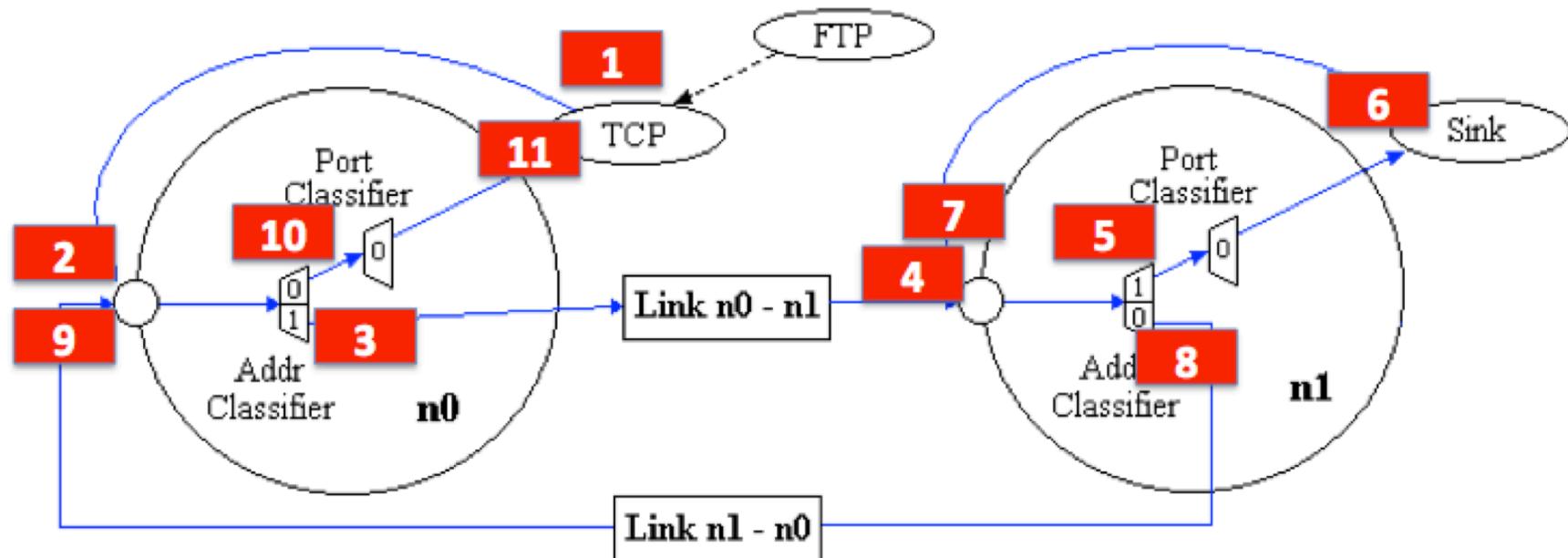
```
set trace_nam [open out.nam w]
$ns namtrace-all $trace_nam
```

```
set trace_file [open out.tr w]
$ns trace-all $trace_file
```

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



Packet flow





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"
```



Eseguire la simulazione

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

```
ns mio_script.tcl
```



Trace dei risultati

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)
- : dequeue (at queue)
d : drop (at queue)

src_addr : node.port (3.0)
dst_addr : node.port (0.0)

```
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
```



Esempio 1: 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
```



Esempio 2: 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
```



Esempio 3: creazione traffico FTP su link TCP

```
#Create a TCP agent and attach $ftp set type_ FTP  
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
```

```
#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
```



Esempio 4 (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 n0 [$ns node]
```

```
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
```



Esempio 4 (2/2)

```
# 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
```



Esempio 5: 4 nodi con link UDP e TCP

```
#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"
#Call the finish procedure after 5 seconds of
simulation time
$ns at 5.0 "finish"
#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"
#Run the simulation
$ns run
```



Esempio 6 (1/2): 6 nodi con routing e guasto sui 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
```



Esempio 6 (2/2)

```
# 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 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
```

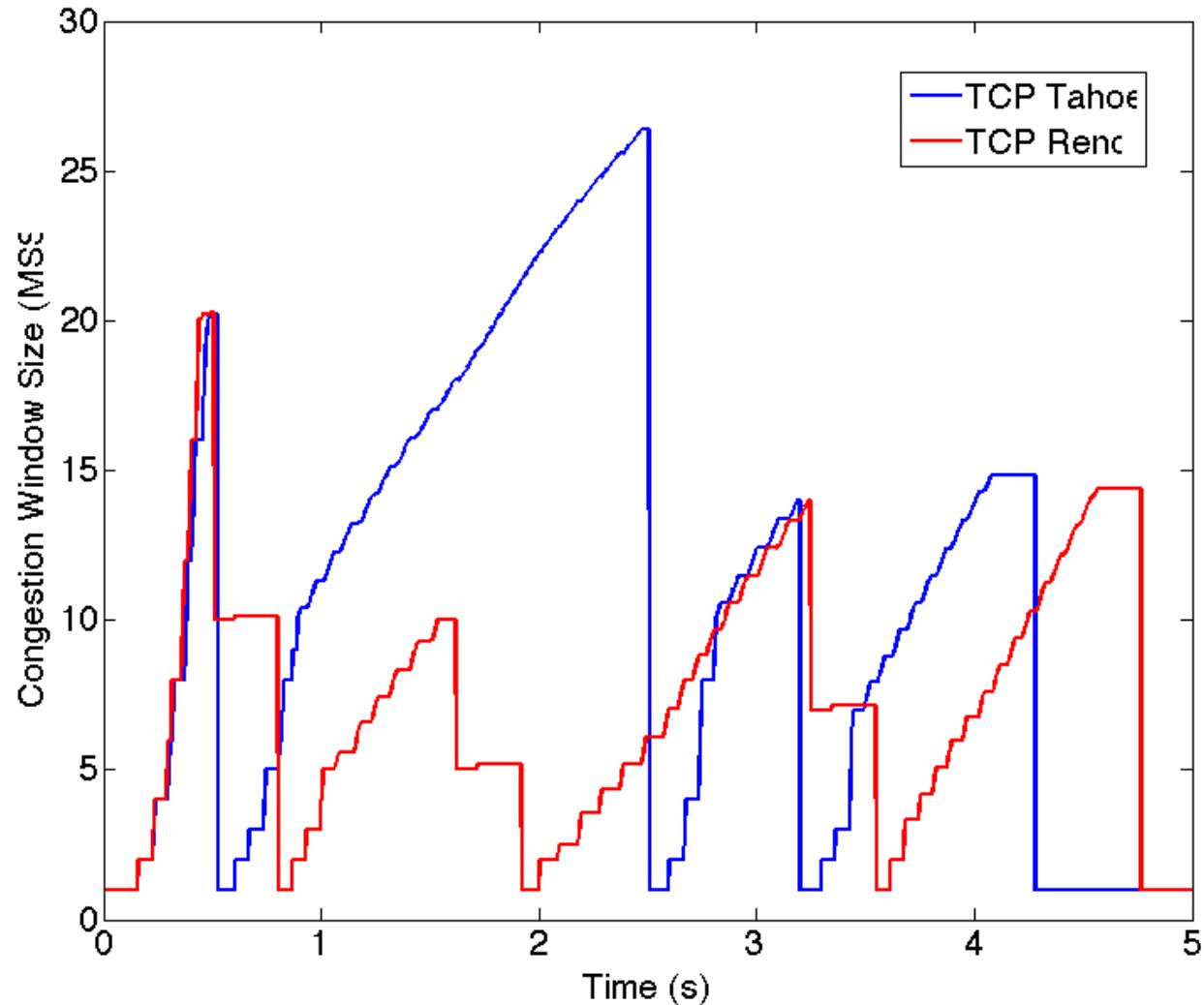


Esempi

- Script con differenti controlli di congestione del TCP
 - Tahoe vs Reno

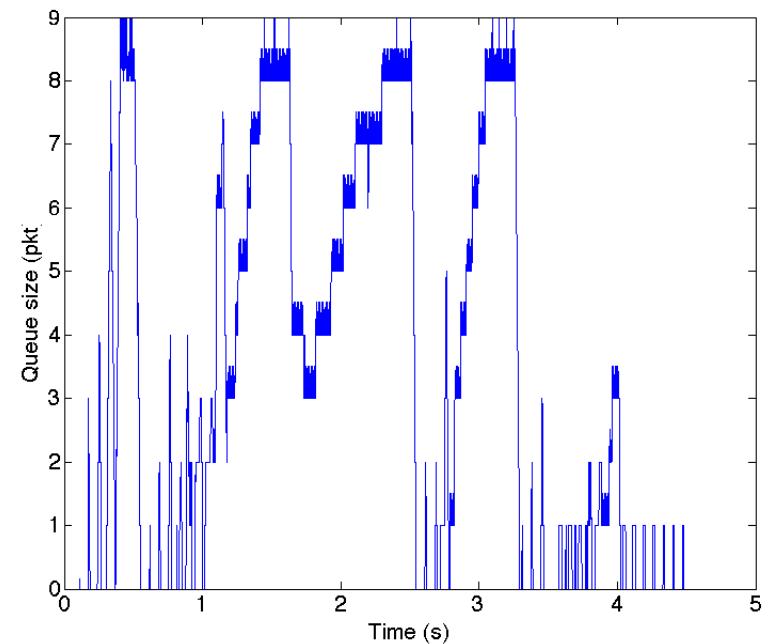
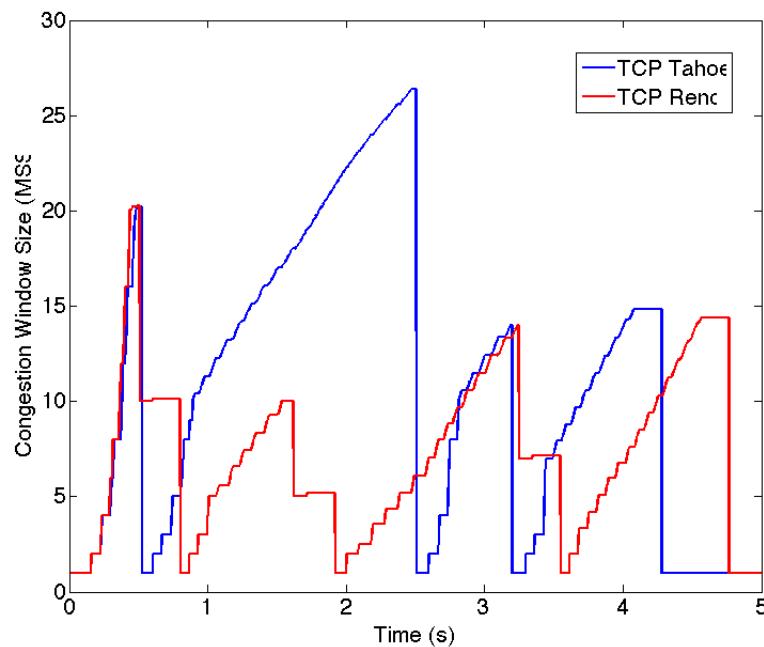


Esempi: Tahoe vs Reno





Esempi: Tahoe vs Reno





Domande?