Introduzione al simulatore NS2 e valutazione del TCP in ambiente wireless

Gaia Maselli
maselli@di.uniroma1.it

Outline

- 1° parte
  - Introduzione alla simulazione di rete
  - Architettura del Network Simulator NS2
  - Utilizzo di NS2
- 2° parte
  - Valutazione del protocollo TCP in ambiente wireless

Introduzione alla simulazione

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

Quando simulare

- Studio e sperimentazione delle interazioni interne di un sistema complesso (per es. TCP in sistemi wireless)
- Valutazione delle prestazioni di un sistema prima della costruzione del prototipo
- Verifica di soluzioni analitiche
- 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

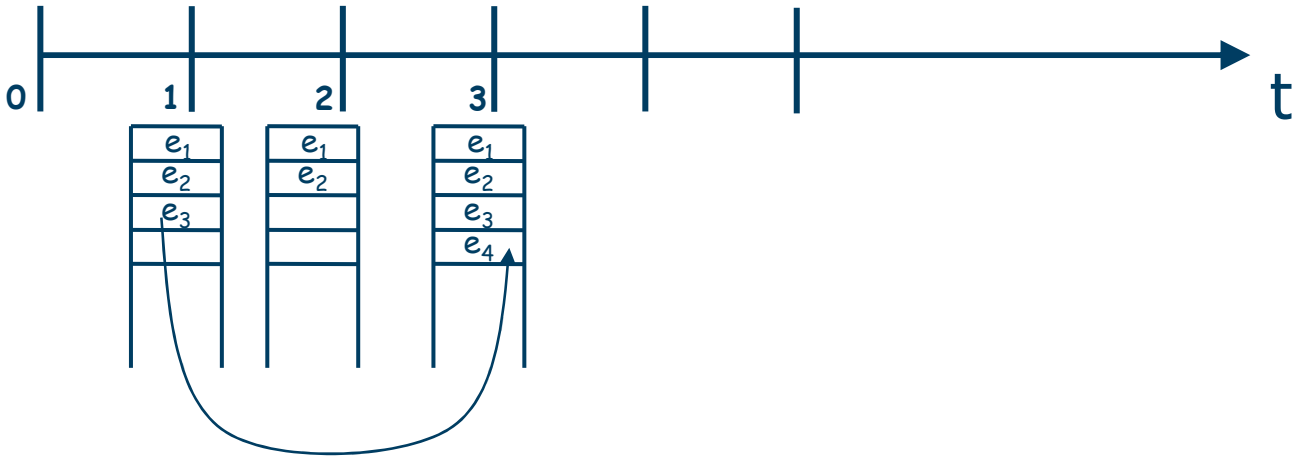
- Contro
  - La creazione del modello e la sua validazione richiedono una comprensione dello strumento di simulazione
  - Non è possibile catturare svariati aspetti del sistema simulato (es. in NS2, prestazioni locali ai singoli nodi)

Outline

- 1° parte
  - Introduzione alla simulazione di rete
  - Architettura del Network Simulator NS2
  - Utilizzo di NS2

Network Simulator NS2

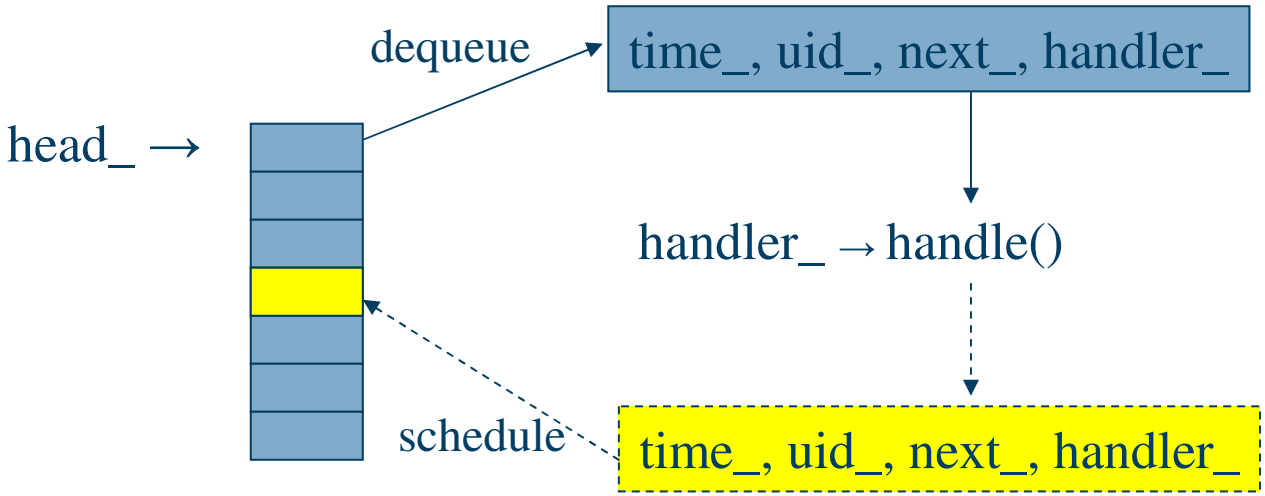
- Simulatore di rete a eventi discreti
  - L'avanzare del tempo dipende dalla temporizzazione degli eventi, che sono gestiti da uno scheduler
  - Gli eventi rappresentano l'evoluzione temporale del sistema in un determinato intervallo di tempo.
  - Il sistema evolve in istanti appartenenti ad un insieme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



Schedulazione di un nuovo evento

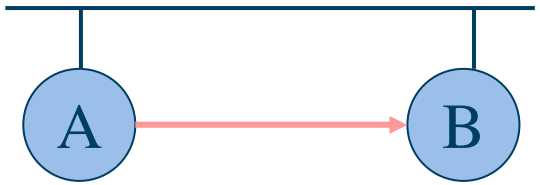
Scheduler degli eventi

- Modellazione ad eventi
  - Lo scheduler
    - Mantiene la lista di eventi che devono essere eseguiti
    - Esrae 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 arbitaria 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

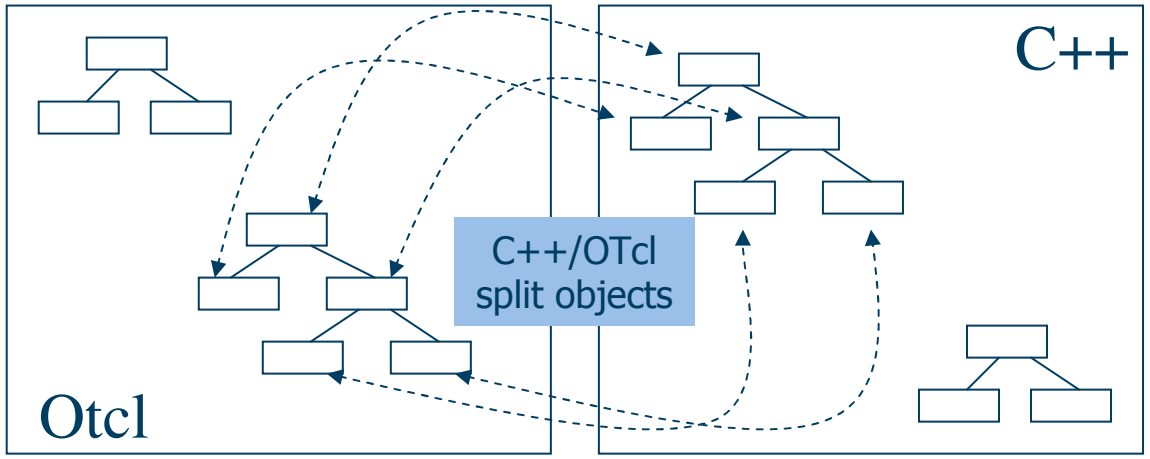
t=1.01:
- NIC di B conclude ricezione
- NIC di B passa il pacchetto all'applicazione

Implementazione di NS2

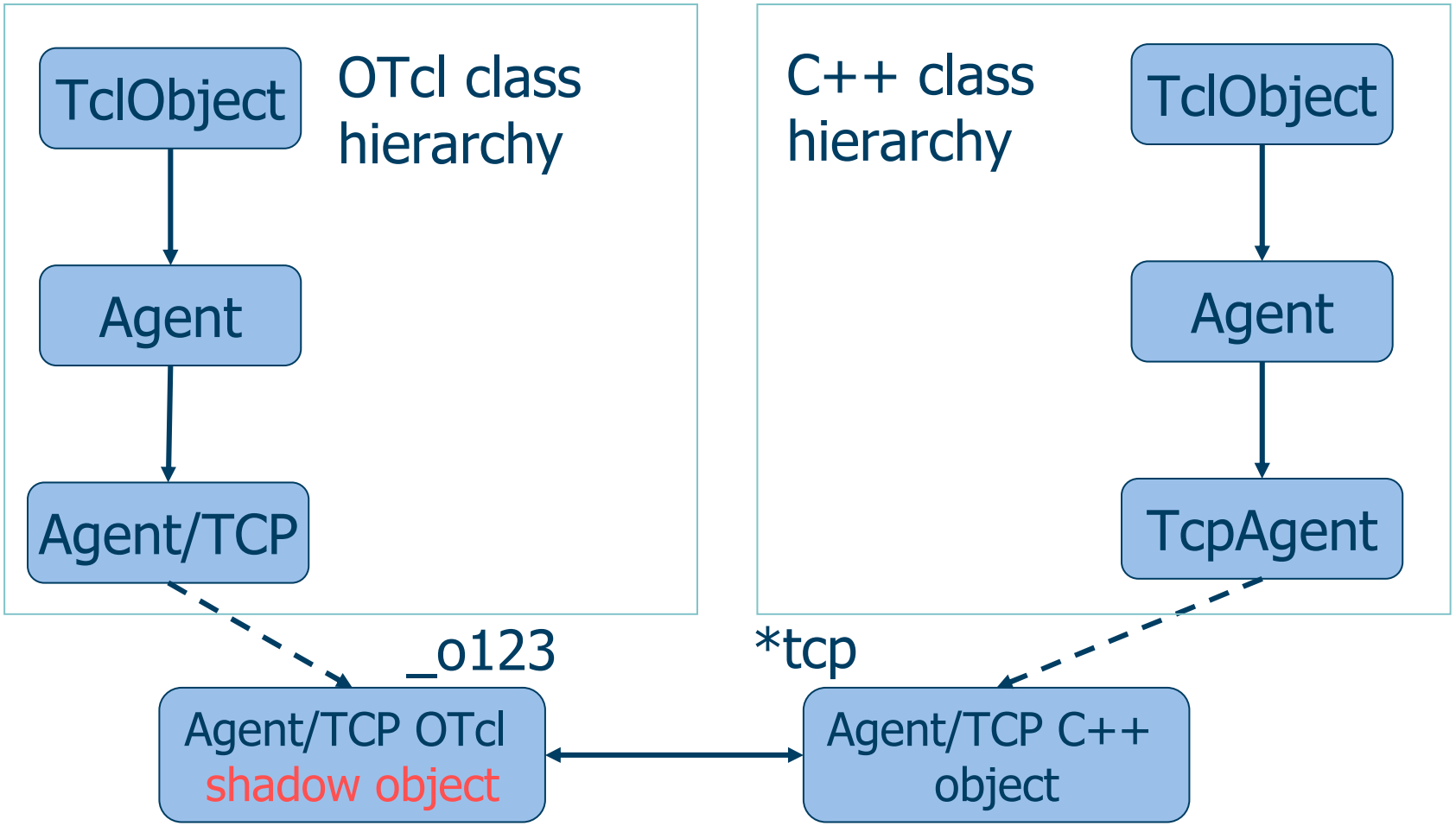
- Codice sorgente di **pubblico dominio**
- In continua evoluzione, aggiornato e modificato da ricercatori e studenti di tutto il mondo
- Piattaforme supportate: Unix, Unix-like, Windows
- Sito ufficiale: http://www.isi.edu/nsnam/ns/
- Approccio **modulare**
- Implementato in ***tcl*** (Tool Command 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



Cosa si può simulare in NS2

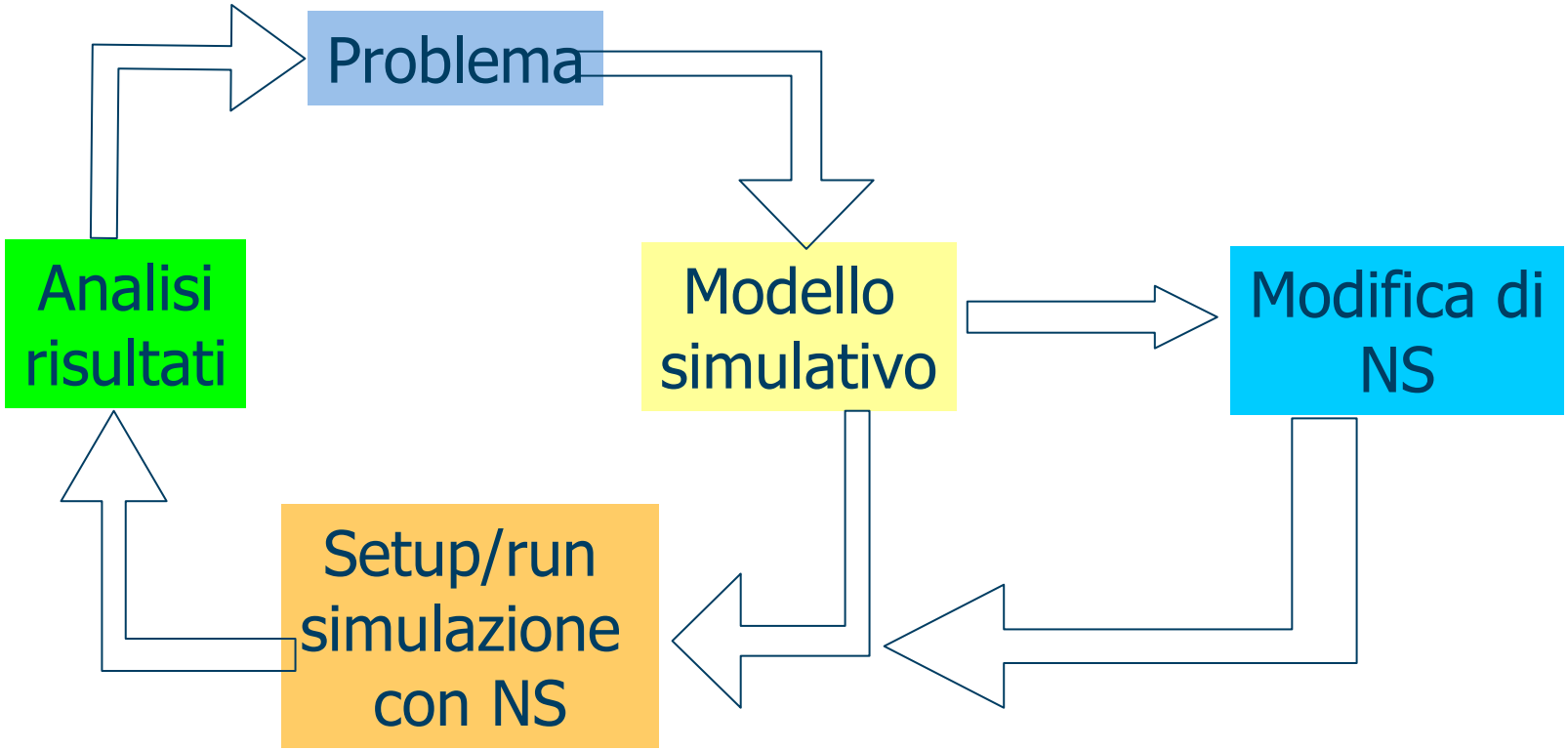
Adatto per la simulazione di protocolli a livello:

- Data-link
  - *Schemi MAC in reti wired/wireless (ad-hoc e sensor networks)*
- Rete
  - *Routing, Multicast*
- Trasporto
  - *TCP Congestion control*
- Applicazione
  - *Caching, Streaming, P2P*

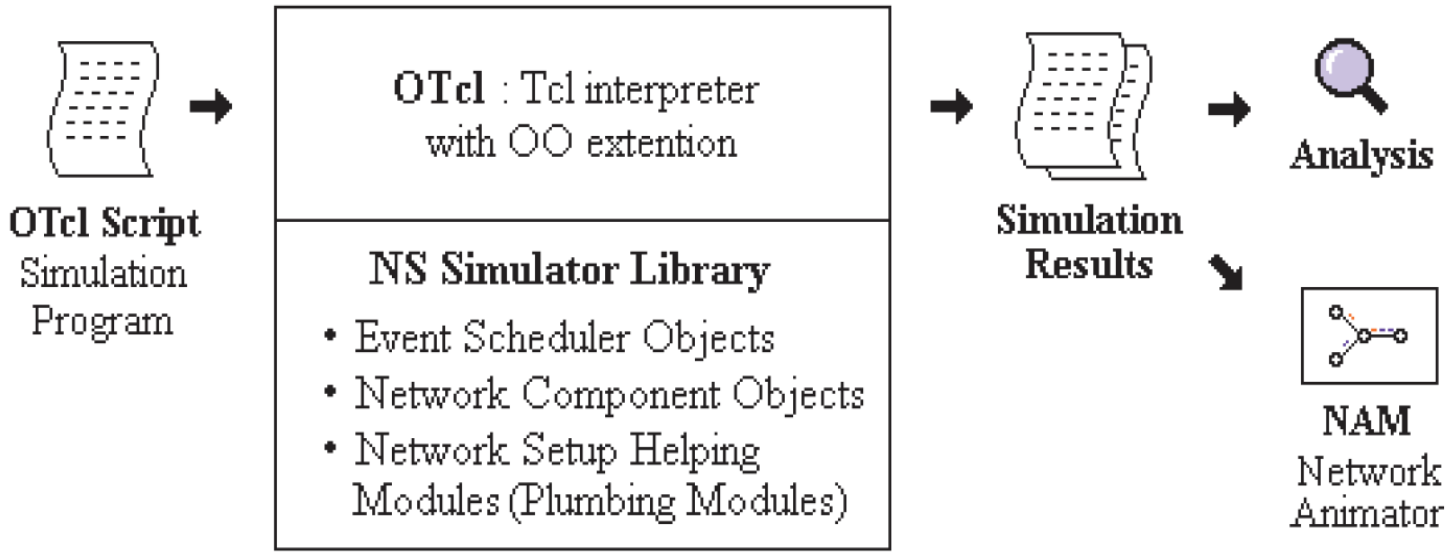
Outline

- 1° parte
  - Introduzione alla simulazione di rete
  - Architettura del Network Simulator NS2
  - Utilizzo di NS2

Usare NS2: fasi



Architettura di NS2



- Dove operare:
  - Tcl: script per costruire il modello di rete che si vuole simulatore
  - C++: per implementare nuovi protocolli è necessario creare o modificare classi C++
- Due tipi di possibile 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

`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 (creazione scheduler)

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

(riferito come `$ns`)

  - Nodi

```
set n0 [$ns node]
set n1 [$ns node]
```

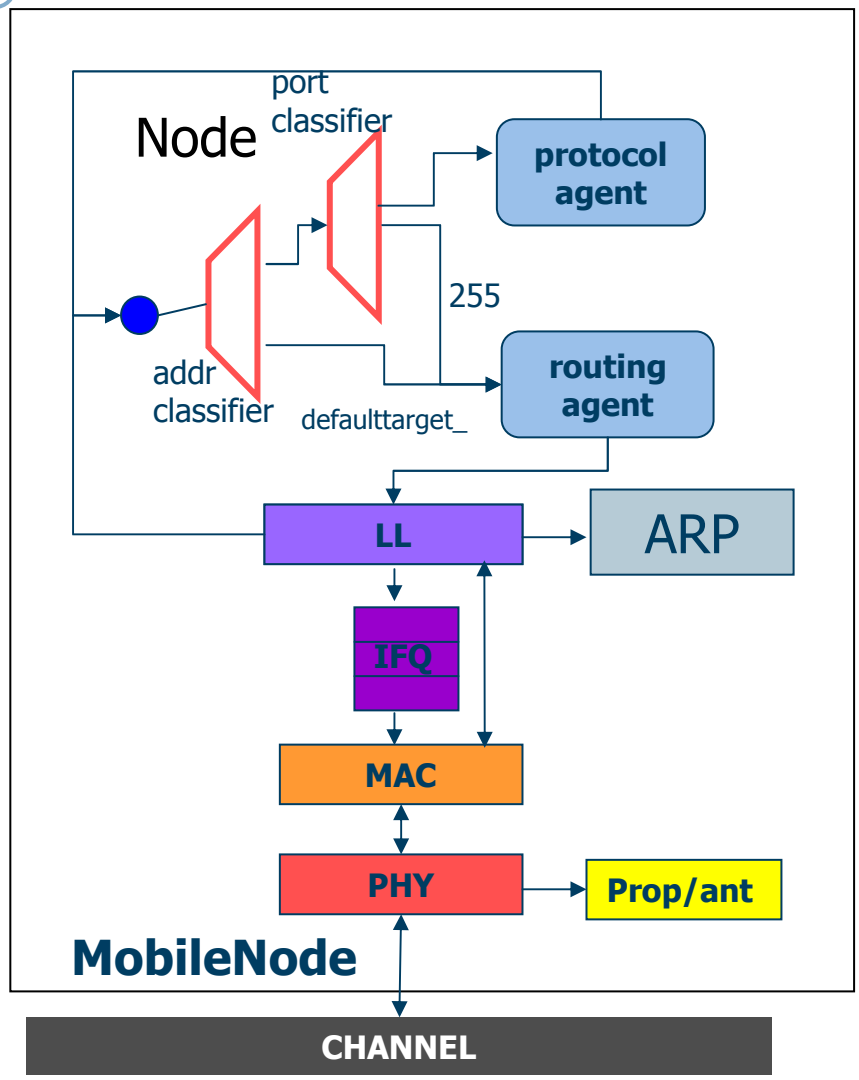
(`node` è metodo di `Simulator`)
(riferite come `$n0, $n1`)

  - Link wireless: parte della definizione del nodo

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

Struttura di un nodo wireless



Classifier: Forwarding

Agent: Protocol Entity

Node Entry

LL

LL: Link layer object

IFQ

IFQ: Interface queue

MAC

MAC: Mac object

PHY

PHY: Net interface

Prop/ant

Radio propagation/ antenna models

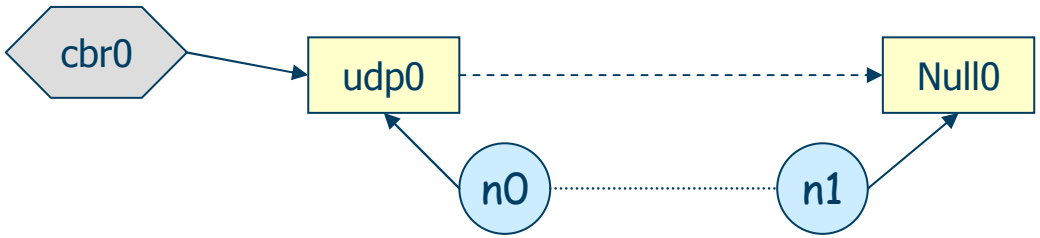
Definizione di agenti e applicazioni

- 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
```



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"
```

Generazione file di trace

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

Pointers to the trace files

Trace files

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

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

Eseguire la simulazione

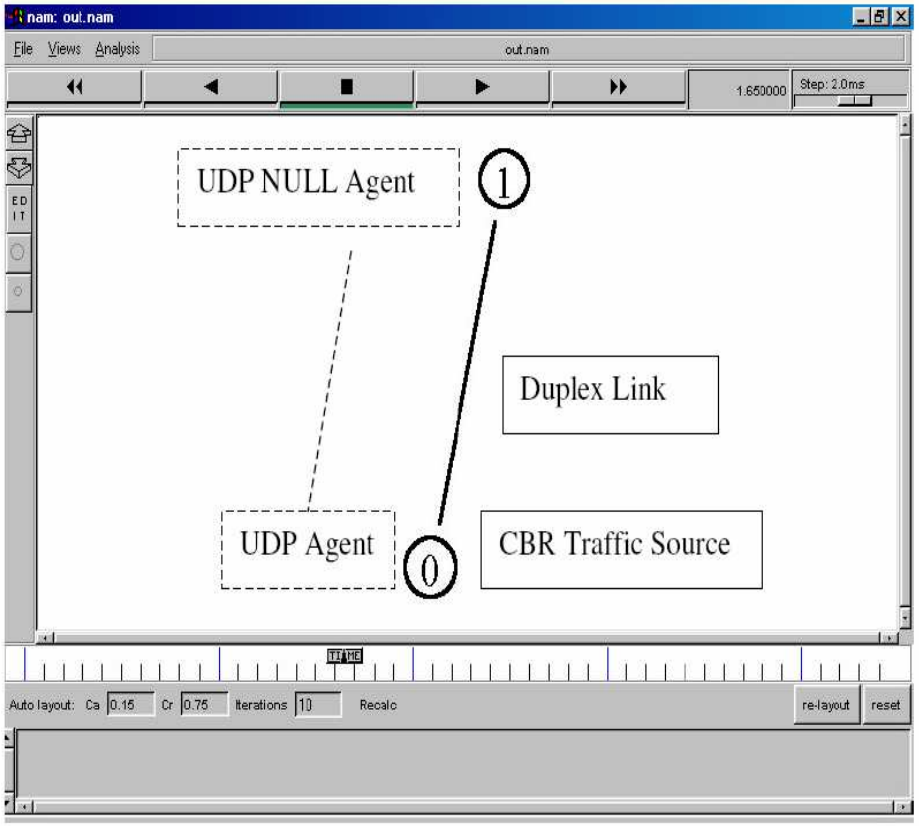
L'esecuzione della simulazione avviene facendo interpretare lo script Otcl a NS

```
ns mio_script.tcl
```

Visualizzare i risultati: NAM

Animazione

- NS genera un file di trace (<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 contente 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`**

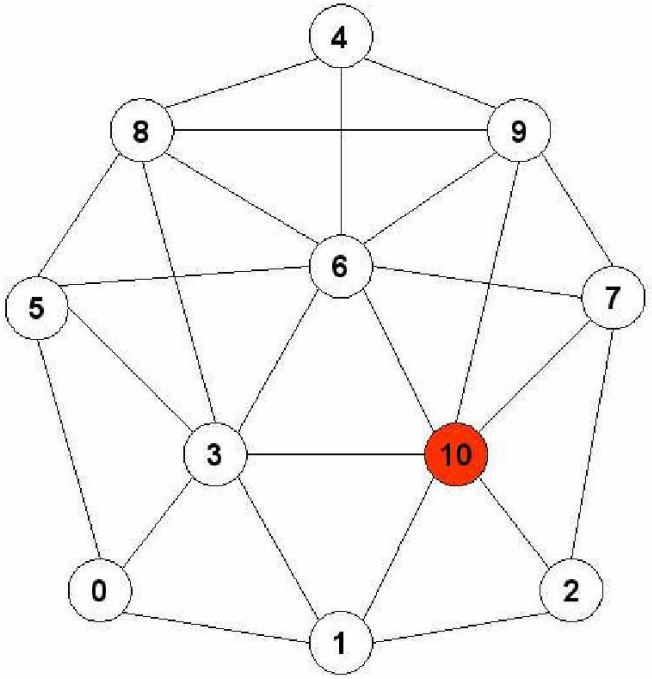
    produce un file contenente:

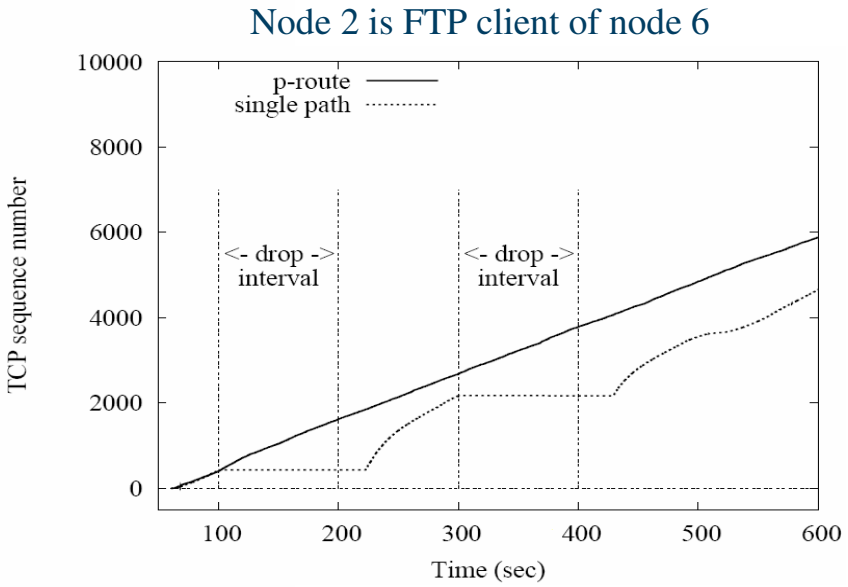
    ```
    s 60.314477381 _2_ AGT  --- 801 tcp 1040 [0 0 0 0] ------- [2:1 9:1 32 0] [1 0] 0 0
    s 60.314477381 _2_ AGT  --- 802 tcp 1040 [0 0 0 0] ------- [2:1 9:1 32 0] [2 0] 0 0
    ```

  - **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 trace è possibile costruire dei grafici con degli strumenti di plotting
  - gnuplot
  - xgraph
- Esempio





Prossima lezione

- Dimostrazione dell'utilizzo di NS2
- Valutazione di varianti TCP in ambiente wireless