

Il simulatore di rete ns2

A.A. 2007/2008

ns2 – Network Simulator ver. 2

- Simulatore di reti di telecomunicazioni
  - sviluppato presso *University of Southern California - Information Sciences Institute (ISI)*
  - software di simulazione di pubblico dominio
  - http://www.isi.edu/nsnam/ns/
- Motore di simulazione (*engine*): linguaggio **C++**
- Interazione utente-simulatore: linguaggio **OTcl**
  - Object-oriented Tool Command Language
  - Linguaggio di *scripting*
    - natura interpretata non compilata
    - alta efficienza a livello di utente nello sviluppo del codice

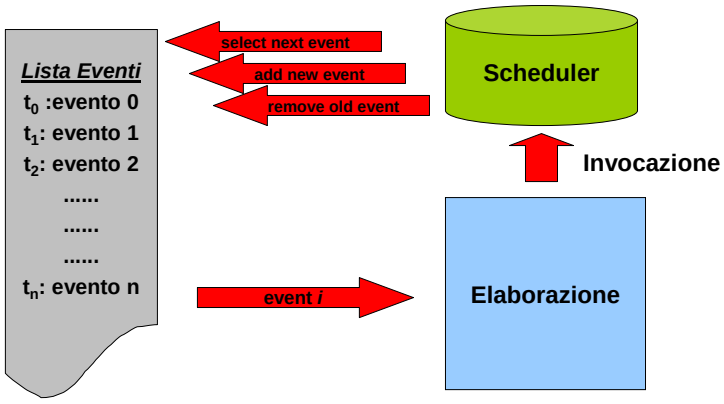
Simulatore ad eventi tempo-discreto

- L'istante in cui si presenta un evento attiva l'esecuzione di azioni specificate dal programma
- Gli eventi sono caratterizzati dall'istante di occorrenza ed inseriti in una **lista degli eventi**
- Il simulatore agisce sulla lista eventi
  - inserisce un nuovo evento nella lista
  - processa l'evento attuale estraendolo dalla lista
  - esegue le azioni associate all'evento
  - elimina l'evento processato
- L'accesso alla lista è gestito da uno **scheduler**

3

Gestione degli eventi

- La gestione degli eventi coinvolge l'unità di elaborazione, lo scheduler e la lista eventi



4

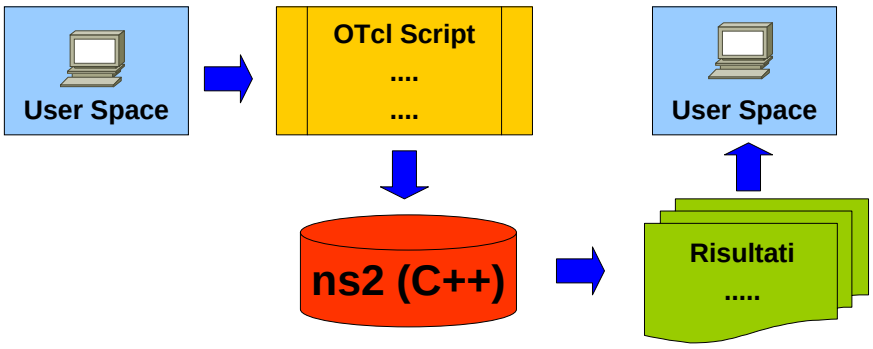
Linguaggio OTcl

- Permette di descrivere tutte le operazioni fondamentali per la simulazione
- Attraverso opportuni comandi consente l'impostazione del set-up:
  - Topologia di rete,
  - caratteristiche dei link,
  - tipo di traffico,
  - protocolli supportati,...
- Elaborazione e presentazione dei risultati di una simulazione

5

La simulazione con script OTcl

- OTcl è il mezzo con il quale l'utente può simulare il funzionamento di una rete reale



6

Caratteristiche e comandi Otcl: esempi (1)

- Creare, inizializzare ed eliminare una variabile:
  - set <nome_variabile> <valore>
  - unset <nome_variabile>

- Chiamata ***per riferimento*** e ***per valore***
  - per riferimento: variabile
  - per valore: $variabile

```
% set x 1

% puts “x vale $x”
i vale 3
```

7

Caratteristiche e comandi Otcl: esempi (2)

- Cicli FOR

```
for {set i 0} {$i < 100} {incr i} {
  set v($i) “elemento n. $i dell’array v”
}
```

- Cicli WHILE

```
while {$i < 100} {
  set v($i) “elemento n. $i dell’array v”
  incr i
}
```

- Istruzioni IF

```
if {$i < 10} {
  puts “i è minore di 10”
}
```

8

Le procedure OTcl

- Sono un componente fondamentale di Otcl
  - facilitano la programmazione
  - permettono l'esecuzione di azioni ricorsive
  - caratterizzano il programma in blocchi logici

- Esempio:

```
proc add {first second} {
  set sum [expr $first + $second]
  return $sum
}

set x 2
set y [add $x 6]
```

9

Documentazione su Tcl/Tk

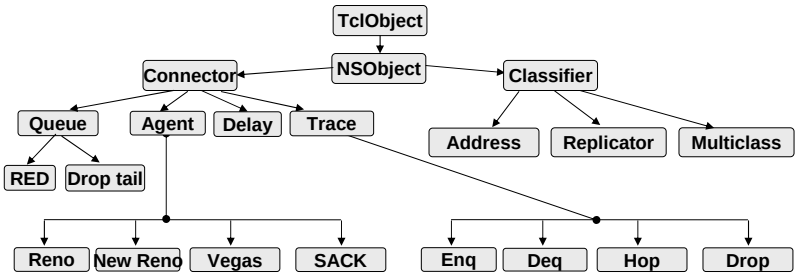
- Sito web ufficiale
  http://www.tcl.tk/doc/

- Tutorial in italiano
  http://monitor.deis.unical.it/ant/it/tcl/index.html

10

Il modello di rete in ns2

- Il modello di rete è realizzato interconnettendo diversi componenti denominati *oggetti ns*
- Gli oggetti *ns2* sono costruiti a partire da una struttura gerarchica di classi *C++*
- Tutti gli oggetti sono derivabili dalla classe *NSObject* che è la base per tutte le classi



11

Le classi di ns2

- ns2 presenta un ampia varietà di classi utili per la definizione di svariati scenari di rete
- ns2 è uno strumento in continua evoluzione
  - permette la definizione di nuove classi per la caratterizzazione di nuovi componenti di rete, collegamenti, protocolli, ecc,...
- ns2 definisce alcune classi “standard” ampiamente sufficienti per lo scopo del corso
- La descrizione completa di tutte le funzionalità è reperibile sul manuale ufficiale
  - http://www.isi.edu/nsnam/ns/ns-documentation.html

12

Le classi standard di ns2

- Le classi standard utili per implementare un modello di rete per lo studio del protocollo TCP sono:
  - SIMULATOR
  - NODE
  - LINK
  - AGENT
  - ERROR MODEL
  - TRACE
  - MONITOR
- Parametri di default definiti in
  `<ns-2 directory>/tcl/lib/ns-default.tcl`

13

La classe SIMULATOR

- Classe fondamentale di ns2
- Permette di istanziare il motore di simulazione
- Primo oggetto creato da uno script OTcl

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

- I metodi della classe *simulator* permettono di
  - configurare la simulazione
  - schedulare gli eventi
- 3 categorie di metodi
  - configurazione della topologia (creazione dei nodi, dei link,...)
  - monitoraggio dell'evoluzione temporale di oggetti (trace)
  - configurazione dello scheduler

14

La classe NODE: generalità

- Classe implementata in Otcl
- Implementano le funzionalità del protocollo IP
  - definizione dell'indirizzamento
  - routing (shortest path)
  - consegna delle unità informative al protocollo di trasporto
- Due tipi:
  - *Unicast Node*: per gestire pacchetti con un solo mittente ed un solo destinatario
  - *Multicast Node*: per gestire pacchetti con un solo mittente e più destinatari

15

La classe NODE: comandi

- Creazione dell'oggetto NODE:

```
set node1 [$ns node]
```

- Ritorno dell'identificatore di nodo

```
$node1 id
```

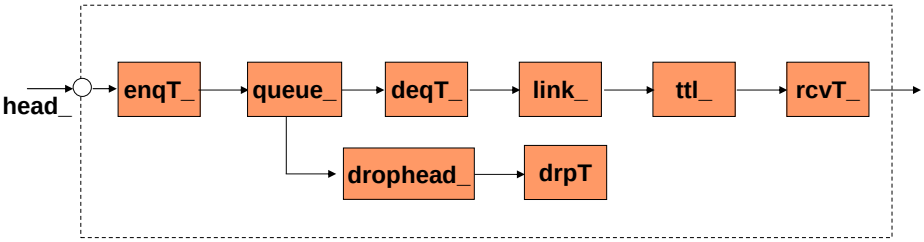
- Creazione di N oggetti NODE

```
for {set i 0} {$i < N} {incr i} {
  set node$i [$ns node]
}
```

16

La classe LINK: generalità

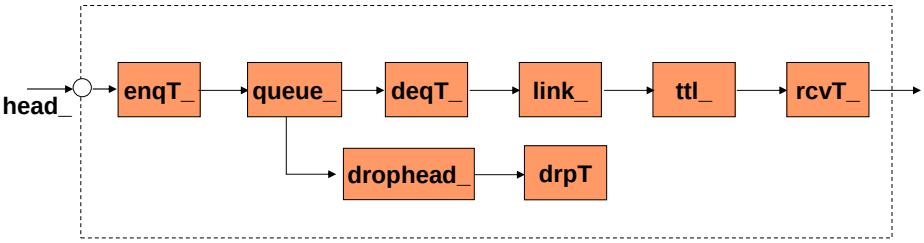
- Oggetto che permette i collegamenti fra i nodi



- head_ puntatore all'ingresso
- queue_ puntatore alla coda del link (*in ns2 le code si trovano nei link*)
- link_ puntatore all'oggetto che descrive il link in termini di ritardo di propagazione, capacità,...
- ttl_ puntatore all'oggetto adibito alla gestione del TTL
- drophead_ puntatore all'ingresso di un oggetto per gestire lo scarto dei pacchetti sul link

17

La classe LINK: generalità



- enqT_ puntatore all'oggetto che processa i pacchetti che accedono all'oggetto queue_
- deqT_ puntatore all'oggetto che registra i pacchetti che lasciano l'oggetto queue_
- drpT_ puntatore all'oggetto che registra i pacchetti che vengono scartati dall'oggetto queue_
- rcvT_ puntatore all'oggetto che registra i pacchetti ricevuti dal nodo collegato attraverso Link

18

La classe LINK: creazione dell'oggetto

- Esempio:

```
set ns [ new Simulator ]
set node1 [ $ns node ]
set node2 [ $ns node ]
set node3 [ $ns node ]
set node4 [ $ns node ]

$ns simplex-link $node1 $node2 10Mb 1ms DropTail
$ns duplex-link $node3 $node4 100Mb 50ms DropTail
```

invocazione della classe simulator

creazione di 4 oggetti NODE

Creazione di 2 collegamenti fra i nodi specificandone il tipo, la capacità, il ritardo e la politica di coda all'interfaccia fra il nodo ed il link d'uscita

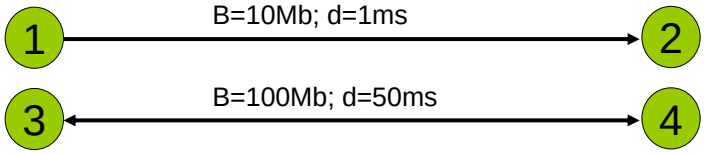
19

La classe LINK: creazione dell'oggetto

- Esempio:

```
set ns [ new Simulator ]
set node1 [ $ns node ]
set node2 [ $ns node ]
set node3 [ $ns node ]
set node4 [ $ns node ]

$ns simplex-link $node1 $node2 10Mb 1ms DropTail
$ns duplex-link $node3 $node4 100Mb 50ms DropTail
```



20

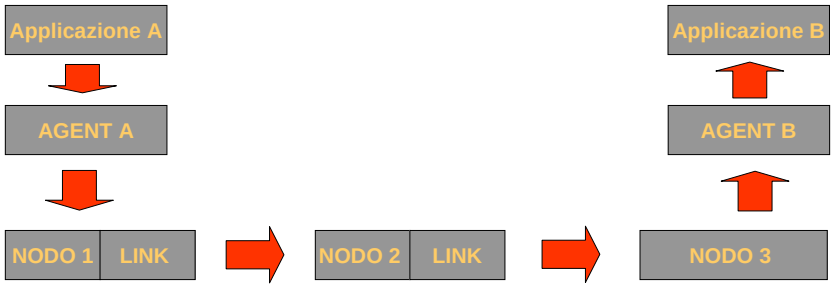
La classe LINK: alcuni comandi

- ***queue-limit***: massima dimensione della coda
  `$ns queue-limit <node0> <node1> <queue_limit>`
- ***$link up***: attiva un collegamento
- ***$link down***: disabilita un collegamento
- ***$link up?***: interrogazione sullo stato di un link

21

La classe AGENT: generalità

- Gli agenti sono gli elementi dove si realizza:
  - la generazione delle unità informative che devono essere trasmesse (es.:applicazione che invia dati)
  - la rimozione delle stesse dopo la ricezione (es.: applicazione che riceve i dati)



22

La classe AGENT: caratteristiche

- L'agente non riceve realmente dati dall'applicazione ma solo la dimensione dell'unità informativa (bytes)
- Diversi tipi di agenti per gestire diversi tipi di protocolli di trasporto (es. TCP, UDP,…)
- Per ogni protocollo di trasporto è definito:
  - Un agente ***trasmettitore***
  - Un agente ***ricevitore***
- Al nodo ricevitore i pacchetti vengono "scartati" dall'agente ricevitore (libera la memoria associata alla struttura dati del pacchetto)
  - AGENT NULL (scarto)
  - AGENT TCPSink (scarto + invio di ACK)

23

La classe AGENT: agent UDP

- Riceve unità dati dall'applicazione di dimensione variabile
- Genera pacchetti di dimensione massima specificata dalla variabile membro `packetSize_`

Esempio:

```
set udp_agent [new Agent/UDP]
$ns attach-agent $node1 $udp_agent
$udp_agent set packetSize_ 512
set null_agent [new Agent/Null]
$ns attach-agent $node2 $null_agent
$ns connect $udp_agent $null_agent
```

24

La classe AGENT: agent TCP

- Due categorie possibili:
  - Agenti TCP unidirezionali
    - Agenti trasmettitori
    - Agenti ricevitori
  - Agenti TCP bidirezionali
- Possibilità di impostare diverse versioni del TCP
  - Tahoe, Reno, NewReno, Vegas etc.
- La classe Agent/TCP è caratterizzata da numerose variabili membro
  - Senza nessuna specifica ➔ valore di default
  - Possibilità di modificare i valori di default

```
Agent/TCP set <member_variable> <value>
```

25

Agent/TCP: variabili membro e significato

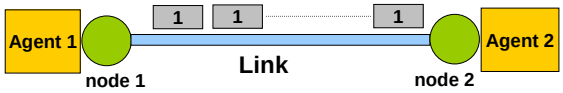
- **window**: dimensione massima della finestra di congestione/ricezione
- **packetsize**: dimensione in byte del pacchetto trasmesso
- **tcpTick**: granularità temporale nella stima del RTT
- **maxrto**: valore massimo per il RTO
- **dupacks**: contatore degli ack duplicati
- **Ack**: il valore più alto di ACK ricevuto
- **cwnd**: dimensione della finestra di congestione (in pacchetti)
- **sstresh**: valore della soglia di slow start
- **rtt**: campione del RTT
- **srtt**: valor medio del campione di RTT
- **rttvar**: varianza del campione di RTT
- **maxseq**: massimo numero di sequenza trasmesso
- ....

26

La classe AGENT: utilizzo

- Esempio:

```
set ns [ new Simulator ]
set node1 [ $ns node ]
set node2 [ $ns node ]
$ns simplex-link $node1 $node2 100Mb 2ms DropTail
set agent1 [ new Agent /UDP ]
set agent2 [ new Agent / Null ]
$ns attach-agent $node1 $agent1
$ns attach-agent $node2 $agent2
$ns connect $agent1 $agent2
$agent1 set fid_ 1
```
(assegna un identificativo al flusso dei pacchetti)



27

Agent TCP bidirezionali

- Classe: *Agent/TCP/FullTcp*
- Permette di simulare il *three-way handshake*;
- Gestisce lo scambio di pacchetti SYN/FIN
- Trasferisce dati in entrambe le direzioni simultaneamente
- Necessita di:
  - oggetti *duplex-link*
  - indicazione esplicita dell'agente TX e dell'agente RX
  - dichiarazione di stato d'ascolto per una richiesta di connessione ($agent_rx listen)

28

La classe APPLICATION: generalità

- Ha lo scopo di emulare le applicazioni più comuni e le caratteristiche di profili di traffico noti
- Le unità dati generate da oggetti application sono passati all'oggetto agent tramite funzioni di interfaccia
- Nei sistemi reali è la *socket* che realizza l'interfacciamento con la suite protocollare TCP/IP
- Anche in ns2 si hanno funzioni di interfacciamento (*API-Application Programming Interface*) che interagiscono con funzioni della classe agent

29

La classe APPLICATION: simulatore di applicazioni

- Possibilità di:
  - generare traffico tipico di applicazioni del mondo reale (FTP, Telnet, WEB)
  - Configurare le caratteristiche dell'applicazione
- In ns2 l'applicazione non genera veri dati ma solo delle dimensioni di file (es.: numero di byte inviati per trasferire un file)
- I byte generati vengono incapsulati in segmenti TCP o pacchetti UDP dall'agente corrispondente
  - inserimento delle intestazioni TCP/UDP ed IP

30

Simulatore di applicazioni: esempio (FTP)

```
set ns [new Simulator]
set node1 [$ns node]
set node2 [$ns node]
$ns duplex-link $node1 $node2 1Mb 1ms DropTail
set agent1 [new Agent/TCP]
set agent2 [new Agent/TCPSink]
$ns attach-agent $node1 $agent1
$ns attach-agent $node2 $agent2
$ns connect $agent1 $agent2
set application1 [new Application/FTP]
$application1 attach-agent $agent1
$ns at 0.0 "$application1 start"
```

31

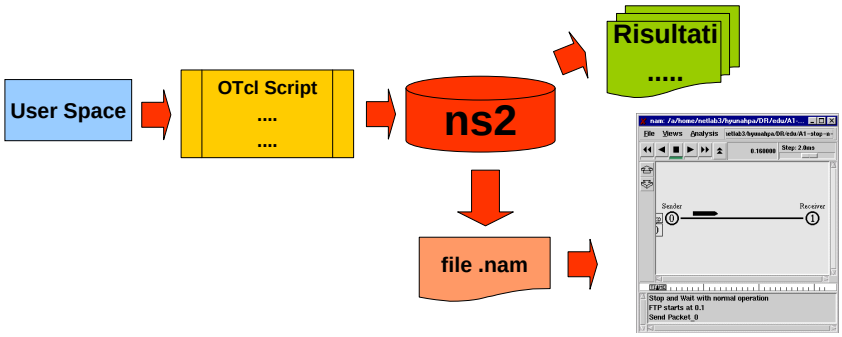
La classe APPLICATION: generatori di traffico

- In ns2 sono implementate 4 classi derivate dalla classe TrafficGenerator:
  - ***EXPOO_Traffic***. generazione di traffico “On/Off ” con distribuzione exp dei tempi di permanenza in ogni stato
  - ***POO_Traffic:*** tempi di permanenza in ciascuno stato distribuiti secondo una distribuzione di Pareto (*bursty*)
  - ***CBR_Traffic:*** generazione di traffico a *rate* costante, con pacchetti di dimensione fissa
  - ***TrafficTrace:*** generazione di traffico effettivo ottenuto da misurazioni sulla rete

32

ns2 tool: NAM-Network Animator

- Il NAM è un tool di animazione appositamente implementato per ns2 che fornisce un'interfaccia grafica per la gestione della simulazione
- L'insieme degli eventi viene registrato in un file di testo (.nam), processato poi dal NAM il quale produce un'animazione della simulazione
- http://www.isi.edu/nsnam/nam/index.html



33

Utilizzo di ns2

- Lab. INFOMEC
  - avviare i PC in modalità Linux
  - accedere alla linea di comando (applicazione Terminal)
  - eseguire lo script di configurazione delle variabili d'ambiente
    `source` _____
  - creare uno script nome.tcl con un editor di testo
  - eseguirlo con ns da linea di comando
    `ns nome.tcl`

- Installazione di ns2 sul proprio PC
  - istruzioni per l'installazione su diverse piattaforme: http://www.isi.edu/nsnam/ns/ns-build.html
  - ricordarsi di impostare le variabili d'ambiente

34

Esercitazione 1

- Leggere le sezioni da I a VI del tutorial di Marc Greis

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

  ed eseguire gli script in esse contenuti

35

Esercitazione 2

- Scrivere ed eseguire uno script che simuli quanto segue:
  - topologia a stella comprendente 4 nodi edge (N1, N2, N3 e N4) e un nodo core (N5)
  - ciascun nodo edge è connesso fisicamente al nodo core tramite un collegamento full-duplex, con ritardo di prop. pari a 10 ms
  - su ciascun nodo edge sono presenti tre sorgenti CBR su UDP che inviano dati a ciascuno degli altri tre nodi edge
  - ogni sorgente genera 200 pacchetti al secondo
  - la dimensione dei pacchetti generati e l'istante in cui le sorgenti iniziano a trasmettere dipendono dalla destinazione:

Destinazione	Dimensione (byte)	Inizio trasmissione
1	100	0.2
2	200	0.4
3	300	0.6
4	400	0.8

  - la capacità di ciascun collegamento deve essere tale da garantire massima utilizzazione della banda senza perdite di pacchetti

36

Esercitazione 3

- Leggere la sezione VIII del tutorial di Marc Greis ed eseguire lo script in essa contenuto, prestando attenzione a quanto segue:
  - come si definisce una sorgente di traffico ON/OFF con periodi di ON e OFF distribuiti esponenzialmente
  - come si definisce e come si richiama una procedura Tcl
  - come si valutano le espressioni numeriche tramite il comando “expr”
  - come si misurano i dati e li si riporta su un file di testo
- Lo script genera file di testo in uscita, ciascuno dei quali contiene la banda misurata di una sorgente in funzione del tempo in un formato a doppia colonna.
- I dati vengono automaticamente riportati su un grafico tramite xgraph, lanciato direttamente dallo script di simulazione
- Gli stessi grafici possono essere realizzati utilizzando software migliori di xgraph, come gnuplot o excel

37

Esercitazione 4

- Modificare lo script usato nell’esercitazione 3 cambiando la frequenza di aggiornamento dei dati nel file di uscita. Per questo occorre modificare il valore del parametro “time” nella procedura “record”. Impostarlo a 2, 1, 0.5 e 0.1 ed eseguire di nuovo la simulazione in ciascun caso. Commentare i grafici risultanti.
- Si può notare che eseguendo lo stesso script più volte senza modificare alcun parametro, anche se le sorgenti sono casuali il grafico non cambia. Questo è dovuto al fatto che il seme utilizzato dal generatore casuale è fisso. Per modificare il seme e impostarlo al valore intero “NEW_SEED” occorre aggiungere all’inizio dello script quanto segue:

```
global defaultRNG
$defaultRNG seed NEW_SEED
```

  Per i dettagli: http://www.isi.edu/nsnam/ns/doc/node266.html
- Eseguire 5 simulazioni con un valore diverso del seme e riportare i grafici ottenuti.

38