Politecnico di Milano

Dipartimento di Elettronica e Informazione

- 2 -

Introduzione a Network Simulator (NS)

Laboratorio di Reti di Telecomunicazione

Informazioni e link

- Sito dei Laboratori di Reti di Telecomunicazione

  `http://www.elet.polimi.it/ntw/fondamenti/laboratorio`

- Home Page di NS versione 2

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

- Tutorials:
  - tutorial di Marc Greis

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

  - NS by examples

    `http://nile.wpi.edu/NS/`

Network Simulator

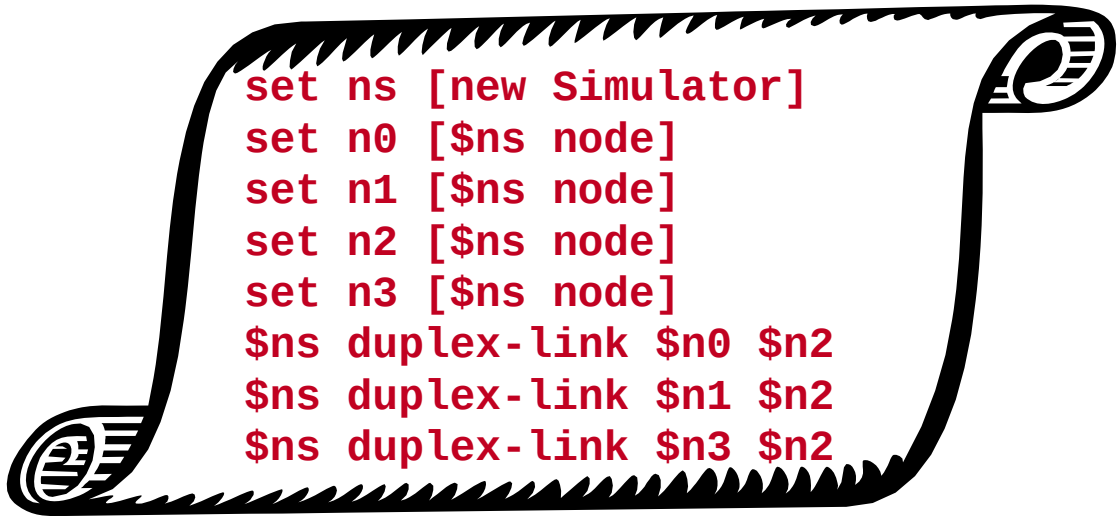
- Si tratta di un simulatore di reti a pacchetto di tipo ad eventi discreti
- è nato e adottato molto spesso per la simulazione di reti IP, ma è utile per lo studio di molti aspetti di base delle reti dati
- è un software FREEWARE
- “non è finito”: è un software in continua evoluzione continuamente aggiornato e modificato da ricercatori e studenti di moltissime università

Network Simulator

- **Per effettuare una simulazione con NS occorre:**
  - **descrivere lo scenario simulativo (nodi, link, sorgenti di traffico, ecc.)**
  - **eseguire la simulazione**
  - **visualizzare i risultati**

Descrizione dello scenario simulativo

- La descrizione dello scenario simulativo avviene mediante uno ***script***
- il linguaggio utilizzato è ***OTcl***, una versione orientata agli oggetti del Tcl (Tool Command Language)
- gli oggetti OTcl utilizzati nello script sono collegati ad oggetti descritti in C++ nel software di simulazione



```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2
$ns duplex-link $n1 $n2
$ns duplex-link $n3 $n2
```

Esecuzione della simulazione

- L'esecuzione della simulazione avviene facendo interpretare lo script OTcl ad NS

```
sh> ns mia-simulazione.tcl
```

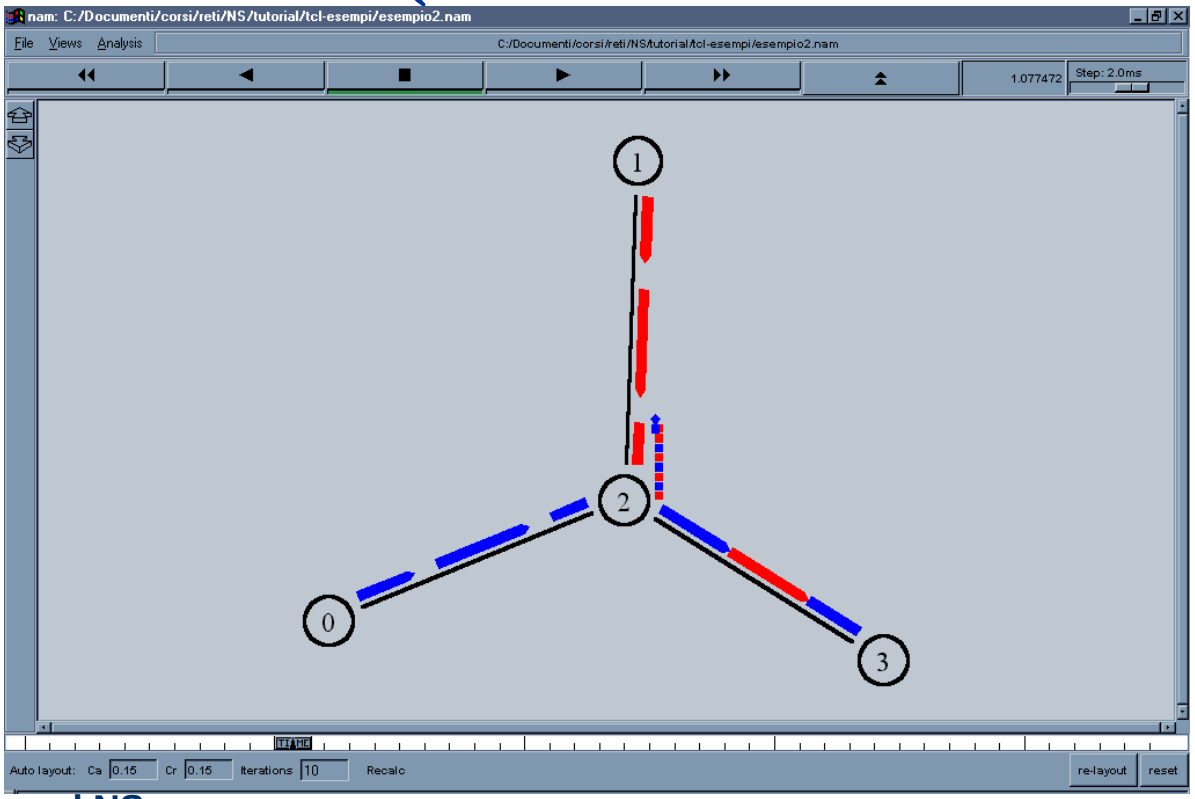
- NS viene fornito nei sorgenti in C++ è può essere compilato per ottenere l'eseguibile
- Esistono gli eseguibili in binario per alcuni sistemi operativi (Linux, Windows, Solaris)

Visualizzazione dei risultati

- La visualizzazione dei risultati può essere ottenuta in molti modi in base allo scopo che ci si prefigge
- File di trace:
  - in modo molto semplice è possibile chiedere al simulatore di generare dei “file di trace” dove vengono registrati tutti gli eventi che si verificano
  - risultati statistici si possono ottenere elaborando offline il file

Visualizzazione dei risultati

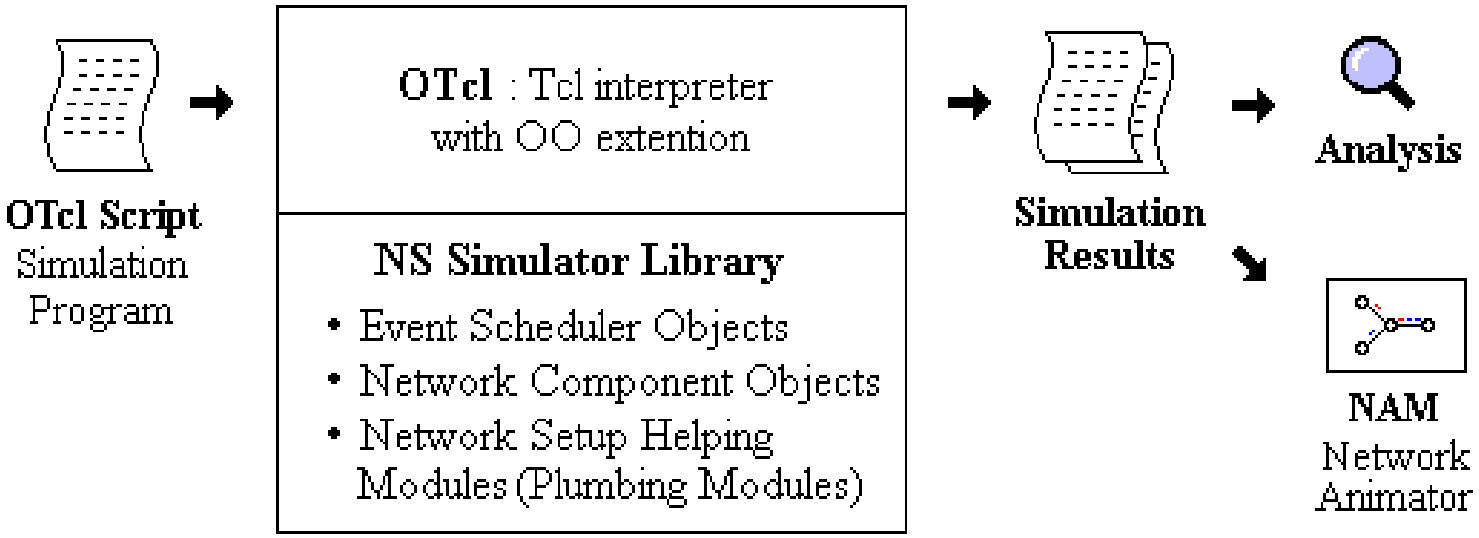
- **Animazione:**
  - **un particolare file di trace generato da NS consente di visualizzare una animazione della simulazione mediante NAM (Network Animator Module)**



Visualizzazione dei risultati

- **Variabili statistiche:**
  - **E’ infine possibile inserire nello script Tcl alcuni comandi che consentono di registrare in un file alcune valori specifici che descrivono le variabili d’uscita desiderate**
  - **non sono purtroppo forniti strumenti generali per questo tipo di approccio (occorre prima prendere familiarità con NS e con OTcl)**

Network Simulator



- **Imparare ad utilizzare NS equivale dunque ad imparare a descrivere gli scenari simulativi mediante OTcl**

Il nostro percorso

- A rigore dovremmo:
  - imparare a programmare OTcl
  - conoscere gli oggetti definiti da NS, le loro variabili e funzioni accessibili tramite script
  - costruire gli strumenti per l'analisi statistica dei risultati
- Noi però
  - adotteremo un approccio semplificato basato su esempi
  - ci serviremo di uno strumento grafico per la descrizione dello scenario che genera lo script OTcl

Oggetti base di NS

- Il primo oggetto base è l'oggetto `Simulator`
- Ogni script OTcl inizia con la creazione di una variabile di tipo `Simulator`

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

- una volta creata la variabile viene utilizzata facendo precedere il nome il simbolo `$`

```
$ns
```

Nodi

- I nodi sono oggetti gestiti da `Simulator` e sono creati in questo modo

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

- la funzione `node` di `Simulator` crea il nuovo oggetto e gli associa un indirizzo interno
- le variabili `$n0` e `$n1` consentono di manipolare i due nodi nello script

Link

- I nodi possono essere collegati da link
- sono definiti due tipi di link
  - simplex-link (link monodirezionale)
  - duplex-link (link bi-direzionale)

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

```
$ns simplex-link $n0 $n1 1Mb 10ms DropTail
```

Link

- **Sintassi:**

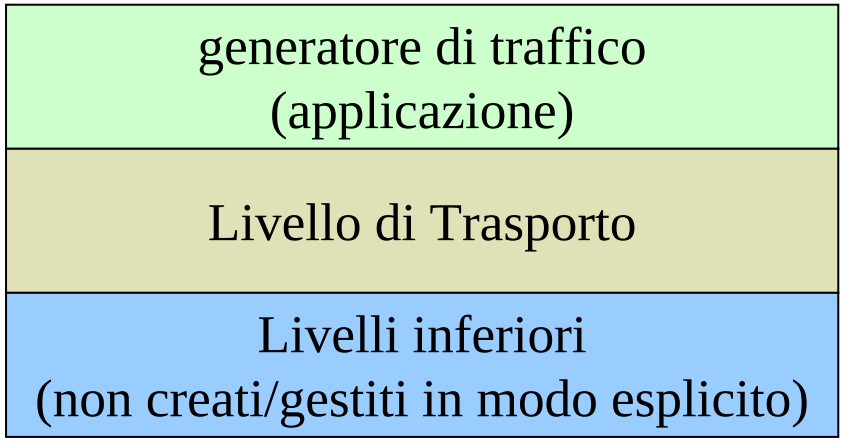
```
$ns simplex-link <node0> <node1> <bandwidht> <delay> <queue_type>
```

- `<bandwidht>` indica la capacità del link
  - `10Mb` (10 Mb/s), `1.2kb` (1.2 kb/s), `1.5e6` (1.5 Mb/s)
- `<delay>` indica il ritardo di propagazione del link
  - `13s` (13 secondi), `1.34ms` (1.34 ms)
- `<queue_type>` indica il metodo di gestione della coda
  - `DropTail` indica una gestione FIFO (First In First Out); esistono anche altre discipline
  - è possibile stabilire le dimensioni della coda (in pacchetti):

```
$ns queue-limit $n0 $n1 10
```

Agenti e Applicazioni

- Dopo aver creato la topologia occorre aggiungere allo scenario simulativo la parte attiva che gestisce il traffico di pacchetti
- In NS questo compito è assegnato agli `Agent` e alla `Application`
- di tipo `Agent` sono le entità che rappresentano il livello di trasporto
- di tipo `Application` sono le entità che generano il traffico



Agenti

- Essendo nato come strumento per il mondo IP i livelli di trasporto definiti in NS sono di tipo UDP e di tipo TCP
- L' `Agent` più semplice è il `Agent/UDP` che modella un livello di trasporto datagram puro lato sender:

```
set UDP0 [new Agent/UDP]
```

- L' `Agent` deve essere collegato ad un nodo mediante la funzione `attach-agent` di `Simulator`:

```
$ns attach-agent $n0 $UDP0
```

Agenti

- Ogni agente deve essere collegato con un altro agente con cui scambia i dati
- Nel caso del sender UDP abbiamo bisogno di un receiver
- Allo scopo possiamo utilizzare un `Agent/Null` che semplicemente riceve i pacchetti e li distrugge
- Oppure un `Agent/LossMonitor` che in più tiene alcune statistiche sui pacchetti ricevuti

```
set Null0 [new Agent/Null]
$ns attach-agent $n1 $Null0
```

- Infine occorre effettuare il collegamento:

```
$ns connect $UDP0 $Null0
```

Agenti

- L’agente UDP svolge funzioni di multiplazione e di segmentazione/riassemblamento
- E’ possibile configurare la dimensione massima del pacchetto (in byte!!!)

```
$UDP0 set packetSize_ 100
```

- e l’identificativo del flusso di pacchetti trasmessi:

```
$UDP0 set fid_ 10
```

Agenti

- Gli agenti TCP che implementano il protocollo in modo completo non sono di diretto interesse per questo corso (lo saranno per il corso di *Infrastrutture e Protocolli per Internet - 2° sem.*)
- Ma noi faremo uso comunque di un agente TCP semplificato (solo lato sender) per studiare le funzioni di controllo di flusso a finestra:

```
set TCPedu0 [new Agent/TCP/RFC793edu]
$ns attach-agent $Node0 $TCPedu0
```

Agenti

- Il ricevitore di dell'agente TCP e' un `Agent/TCPSink`

```
set TCPSink0 [new Agent/TCPSink]
$ns attach-agent $Node1 $TCPSink0
```

- gli agenti sono poi connessi con la solita funzione `connect` di `Simulator`:

```
$ns connect $TCPedu0 $TCPSink0
```

Agenti

- Anche gli agenti TCP possono essere configurati:

```
$TCPedu0 set window_ 2
$TCPedu0 set packetSize_ 100
$TCPedu0 set fid_ 1
```

- nel caso del TCPedu il valore della finestra configurato è fisso mentre per gli altri TCP è solo il valore iniziale

Applicazioni

- La classe `Application` definisce delle sorgenti di traffico
- La più semplice sorgente di traffico è la `Application/Traffic/CBR` che genera pacchetti di lunghezza fissa a ritmo costante

```
set CBR0 [new Application/Traffic/CBR]
```

- La sorgente deve essere collegata ad un `Agent` mediante la funzione `attach-agent`

```
$CBR0 attach-agent $UDP0
```

Applicazioni

- La `Application/Traffic/CBR` può essere configurata:

```
$CBR0 set rate_ 128Kb
$CBR0 set packetSize_ 100
```
(in byte)

- o in alternativa

```
$CBR0 set interval_ 6.25ms
$CBR0 set packetSize_ 100
```

Applicazioni

- L'altra Application di interesse è il `Application/Traffic/Exponential` che definisce una sorgente di traffico ON/OFF
  - periodi di ON e OFF con lunghezza casuale esponenziale negativa
  - durante i periodi di OFF non viene generato alcun pacchetto
  - durante i periodi di ON vengono generati pacchetti a ritmo costante

```
set ExpOnOff0 [new Application/Traffic/Exponential]
```

Applicazioni

- La configurazione di `Application/Traffic/Exponential` usa le seguenti variabili:

```
$ExpOnOff0 set packetSize_ 100
$ExpOnOff0 set burst_time_ 1.2s
$ExpOnOff0 set idle_time_ 1.2s
$ExpOnOff0 set rate_ 100kb
```

- dove:
  - `burst_time_` indica il tempo medio di ON
  - `idle_time_` indica il tempo medio di OFF

Eventi

- Con nodi, link, agenti e applicazioni abbiamo costruito lo scenario simulativo
- ora occorre “animare” lo scenario legandolo alla variabile temporale mediante degli Eventi
- anche se la maggior parte degli eventi sono nascosti all’utilizzatore, occorre comunque inserire dei comandi nello script OTcl
- La riga finale di ogni script OTcl che fa partire la simulazione deve essere:

```
$ns run
```

Eventi

- E’ poi necessario, in generale, inserire degli eventi legati alla generazione di traffico
- Tutte le sorgenti di traffico supportano i comandi (funzioni) di `start` e `stop`
- per fissare lo start e lo stop in determinati istanti di tempo occorre inserire degli eventi:

```
$ns at 0.5 "$CBR0 start"
$ns at 5.0 "$CBR0 stop"
```

- per fermare la simulazione:

```
$ns at 5.5 "exit 0"
```

Trace e NAM

- Per poter ottenere un file di trace utile per l'animazione con NAM basta inserire i comandi:

```
set nf [open file-animazione.nam w]
$ns namtrace-all $nf
```

- il file aperto deve essere chiuso alla fine della simulazione; è conveniente inserire le procedure di fine in una funzione:

```
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exit 0
}
```

- modificando l'evento di chiusura così:

```
$ns at 5.5 ”finish"
```

Esercizio 1a - il primo script OTcl

- Scrivere lo script OTcl per il seguente scenario simulativo:
  - 2 nodi con un link mono-dir. che li collega (capacità 1 Mb/s, ritardo prop. 10 ms)
  - livello di trasporto UDP
  - 1 sorgente di traffico CBR (pacchetti 100 bytes, interarrivo 5 ms)
  - la sorgente CBR inizia a trasmettere al tempo 0.5s e finisce al tempo 4.5s
  - la simulazione termina al tempo 5s

esercizio1a.tcl

Esercizio 1b - il primo script OTcl

- **utilizzare la funzione di segmentazione di UDP settando la massima dimensione del segmento a 50 bytes**

esercizio1b.tcl

Esercizio 1c - il primo script OTcl

- con gli stessi parametri del caso 1a, considerare una sorgente ON/OFF con questi parametri:
  - tempo medio di ON: 50ms
  - tempo medio di OFF: 100ms

esercizio1c.tcl