Università di Bergamo

Dipartimento di Ingegneria dell'Informazione e Metodi Matematici

Laboratorio di Reti

Prof. Fabio Martignon

Università di Bergamo

Dipartimento di Ingegneria dell'Informazione e Metodi Matematici

1 - Introduzione alla Simulazione e a Network Simulator (NS)

Laboratorio di Reti

Informazioni e link

- Sito del corso e Materiale Didattico

  `http://cs.unibg.it/martignon/indexLabReti.html`

- 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/`

Introduzione alla simulazione

- **Cos'è la SIMULAZIONE:**
  - ***La simulazione cerca di costruire un dispositivo sperimentale che si comporti come il sistema reale sotto analisi per alcuni importanti aspetti***
  - **Esempi:**
    - **modellini in scala della superficie esterna di aerei, automobili, treni utilizzati nelle gallerie del vento**
    - **SimCity, Railroad Tycoon, e altri videogame basati sulla riproduzione del funzionamento di un sistema**
    - **simulatori di volo per l'addestramento di piloti**

Introduzione alla simulazione

- **Altri:**
  - **predizione dello sviluppo di ecosistemi dopo un'alterazione artificiale**
  - **verifica di tattiche di investimento in borsa**
  - **previsioni del tempo**
  - **verifica di tattiche di guerra (almeno sono solo simulate!)**
  - **ecc.**

Introduzione alla simulazione

- **Classificazioni:**
- **Esistono molti modi per classificare le simulazioni, tutti mediamente poco utili**
- **Vedremo le differenze con degli esempi in seguito, ma qui iniziamo con qualcosa di base:**
  - **simulazioni deterministiche / casuali:**
    - **le simulazioni deterministiche sono completamente definite dal modello e la loro evoluzione è legata deterministicamente ai parametri d'ingresso;**
    - **le simulazioni casuali sono basate su modelli che includono variabili o processi casuali e necessitano della generazione di variabili casuali; l'evoluzione del modello dipende dai parametri d'ingresso e dalla generazione delle variabili casuali**

Introduzione alla simulazione

- **Classificazioni:**
- **Esempi**
  - **simulazione deterministica:**
    - **si adotta un complesso modello per la descrizione del moto delle bocce sul tavolo di biliardo; note posizione delle bocce, punto-velocità-direzione di impatto con la stecca, si vuol sapere l'esito di un colpo senza risolvere in modo esplicito il modello in modo analitico**
  - **simulazione casuale:**
    - **si consideri una cella GSM con N canali alla quale arrivano delle richieste di connessione secondo un processo di Poisson di tasso λ. Si vuol determinare la probabilità di rifiuto sapendo che con probabilità p le chiamate rifiutate ritentano l'accesso esattamente dopo un tempo pari a T.**

Introduzione alla simulazione

- Classificazioni:
- Tra le simulazioni casuali possiamo distinguere quelle statiche e quelle dinamiche
- simulazioni statiche
  - dette anche simulazioni Monte-Carlo
  - la variabile tempo NON gioca alcun ruolo
  - lo scopo fondamentale è determinare alcune caratteristiche statistiche di una o più variabili casuali
  - di fatto le simulazioni Monte-Carlo implementano misurazioni statistiche su esperimenti ripetuti in modo indipendente

Introduzione alla simulazione

- **simulazioni dinamiche**
  - **dette anche *temporali***
  - **il tempo diventa la variabile principale da legare all'evoluzione del modello**
  - **lo scopo è la raccolta di dati statistici su processi casuali osservati al variare del tempo**
  - **le osservazioni sono critiche perché:**
    - **al contrario delle prove ripetute, non si ha il vantaggio della indipendenza statistica ma occorre tener conto della correlazione delle osservazioni**
    - **non si può sapere a priori se il processo osservato è ergodico (il sistema è stabile o lo si sta osservando in transitorio)**

Introduzione alla simulazione

- La simulazione unita ai linguaggi di programmazione e alla velocità dei moderni elaboratori costituisce uno strumento potente di analisi in grado di risolvere problemi anche complessi
- Ma la simulazione è anche uno strumento che deve essere usato con cura per le seguenti ragioni:
  - non è facile validare i risultati ottenuti
  - la natura statistica dei risultati e la scarsa conoscenza del sistema rende difficile l'analisi dell'output
  - il tempo computazionale può facilmente essere molto elevato
  - non è facile capire come i diversi parametri influenzano il risultato

Modelli e sistemi

- **Sistema:**
  - **è un concetto molto generale che possiamo definire in modo informale come collezione di parti dette componenti che interagiscono tra loro in modo tale che il funzionamento dell'insieme soddisfi certe specifiche**
- **Modello:**
  - **il modello è una rappresentazione del sistema. Tale rappresentazione può assumere varie forme (ad es. quello della replica fisica), ma qui ci si focalizzerà sulla rappresentazione mediante metodi matematici (modello matematico).**

Modelli e sistemi

- **Stato:**
  - lo stato del sistema descrive la condizione istantanea di tutti i suoi componenti
  - allo stato del sistema corrisponde uno stato del modello del sistema, e il modello rappresenta l'evoluzione del sistema mediante la storia dei passaggi di stato
  - lo stato del modello risulta semplificato rispetto allo stato del sistema
  - si parla di *livello di astrazione* del modello ad indicare che alcune caratteristiche dello stato del sistema sono omesse
  - il livello di astrazione è strettamente funzionale alle misure che si vogliono effettuare sul modello
  - il miglior modello è il più semplice che consente di ottenere le misure (prestazionali) desiderate

Modelli e sistemi

- **Variabili**
  - un modello matematico è descritto utilizzando variabili
  - le attività del modello sono descritte come relazioni o funzioni tra variabili
  - variabili di stato:
    - le variabili di stato definiscono in modo completo lo stato del modello e la loro evoluzione definisce l'evoluzione del sistema
  - variabili d'ingresso:
    - le variabili d'ingresso sono parametri da cui dipende il modello e che descrivono sollecitazioni esterne al sistema in esame

Modelli e sistemi

- **variabili d'uscita:**
  - **sono funzione delle variabili di stato e di quello di ingresso e rappresentano le grandezze del modello che si intende misurare**
  - **rappresentano dunque le *sonde* inserite nel modello per la misura**
  - **la soluzione del modello consiste nell'ottenere i valori delle variabili d'uscita**
  - **la soluzione analitica di un modello coinvolge ad esempio metodi matematici di risoluzione di equazioni che descrivono le relazioni tra le variabili**
  - **la soluzione simulata di un modello consiste invece nel riprodurre l'evoluzione del sistema mediante l'evoluzione delle variabili di stato e nella misurazione diretta delle variabili d'uscita**

Simulazione ad eventi discreti

- Alcuni modelli sono caratterizzati dalla proprietà che le variabili di stato cambiano valore solo ad istanti discreti di tempo
- il cambiamento di stato del sistema prende il nome di evento ed è caratterizzato da un istante di occorrenza (un evento non ha durata);
- al contrario l'attività rappresenta una condizione del sistema che perdura per un certo tempo ed è solitamente caratterizzata da un evento di inizio ed un evento di fine
- ad esempio l'inizio e la fine della trasmissione di un pacchetto sono eventi, mentre la trasmissione stessa è un'attività
- la simulazione ad eventi discreti è di fondamentale importanza per le reti di telecomunicazione

Simulazione ad eventi discreti

- **Nella simulazione ad eventi discreti effettuata al calcolatore occorre:**
  - **definire i tipi di eventi che possono verificarsi**
  - **definire per ogni evento le modifiche da apportare allo stato del sistema**
  - **definire una variabile temporale ed ordinare gli eventi in un *calendario* sulla base dell'istante di occorrenza**
  - **definire uno stato iniziale**
  - **scorrere il calendario ed ogni volta che si incontra un evento eseguire le modifiche alle variabili di stato corrispondenti a quell'evento**
  - **effettuare misure sulle variabili d'uscita**

Simulazione ad eventi discreti (riassunto)

- Ingredienti della simulazione ad eventi discreti effettuata al calcolatore:
  - **tipi di eventi che possono verificarsi**
  - **modifiche da apportare allo stato del sistema per ogni evento**
  - **variabile temporale *t* che consenta di ordinare gli eventi in un calendario sulla base dell'istante di occorrenza**
  - **stato iniziale**

- La simulazione (esecuzione del programma) consiste:
  - **nello scorrere il calendario ed ogni volta che si incontra un evento eseguire le modifiche alle variabili di stato corrispondenti a quell'evento**
  - **nell'effettuare "misure" sulle variabili di uscita**

Come fare una simulazione

- Un simulatore è un software
- Si può costruire un simulatore (ad-hoc) scrivendo il software con i normali linguaggi di programmazione (C, C++, Java, ecc.)
- Si possono usare dei software di simulazione che consentono di descrivere il modello simulativo mediante strumenti grafici o linguaggi ad alto livello
- Esistono software per la simulazione di reti di telecomunicazione commerciali molto sofisticati (OPNET è il più noto)
- In questo corso si fa uso di uno strumento freeware: Network Simulator (*ns*)

Network Simulator

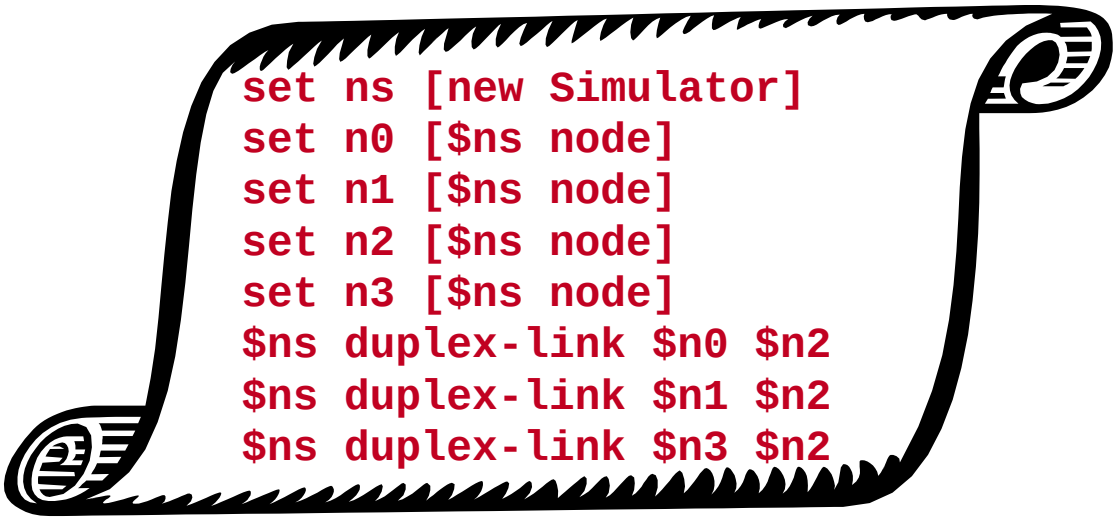
- Si tratta di un simulatore di reti a pacchetto di tipo ad eventi discreti
- è nato e adottato molto spesso per la simulazione di reti IP, ma è utile per lo studio di molti aspetti di base delle reti dati
- è un software FREEWARE
- “non è finito”: è un software in continua evoluzione continuamente aggiornato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

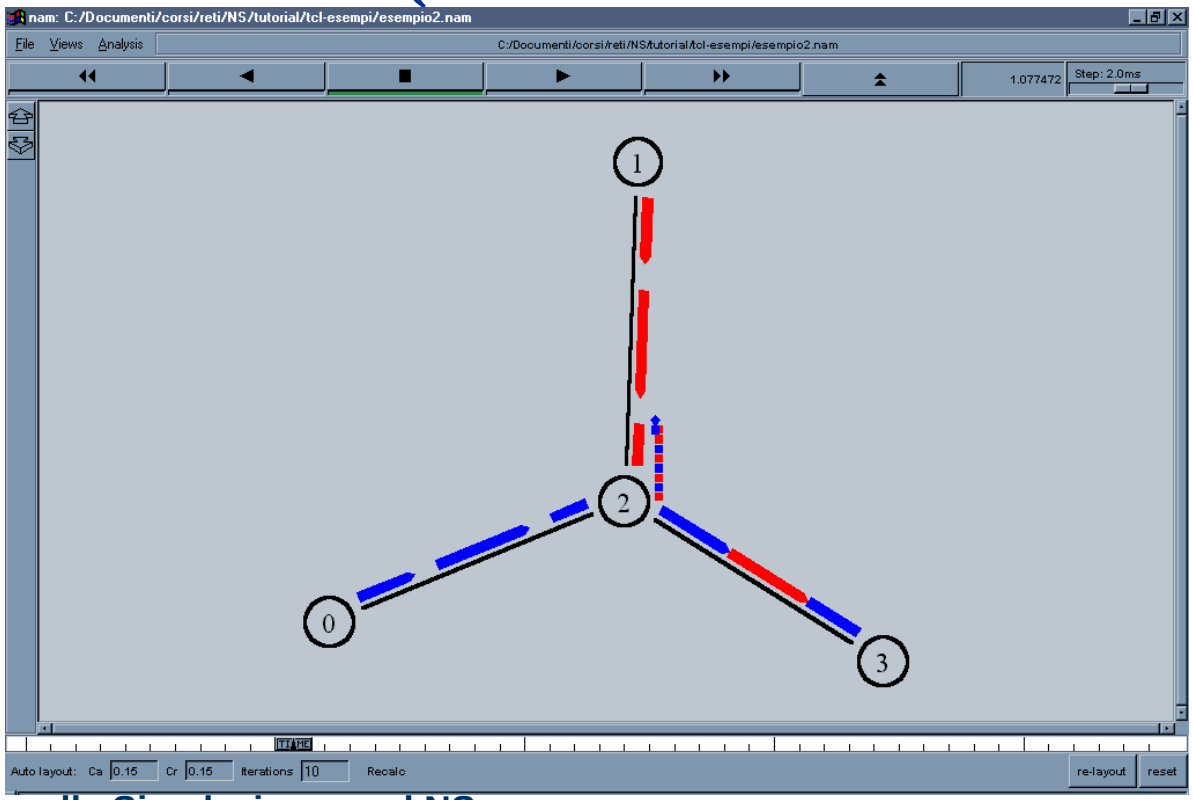
- NS viene fornito nei sorgenti in C++ è può essere compilato per ottenere l'eseguibile
- Esistono gli eseguibili in binario per alcuni sistemi operativi (Linux, Windows, Solaris)

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

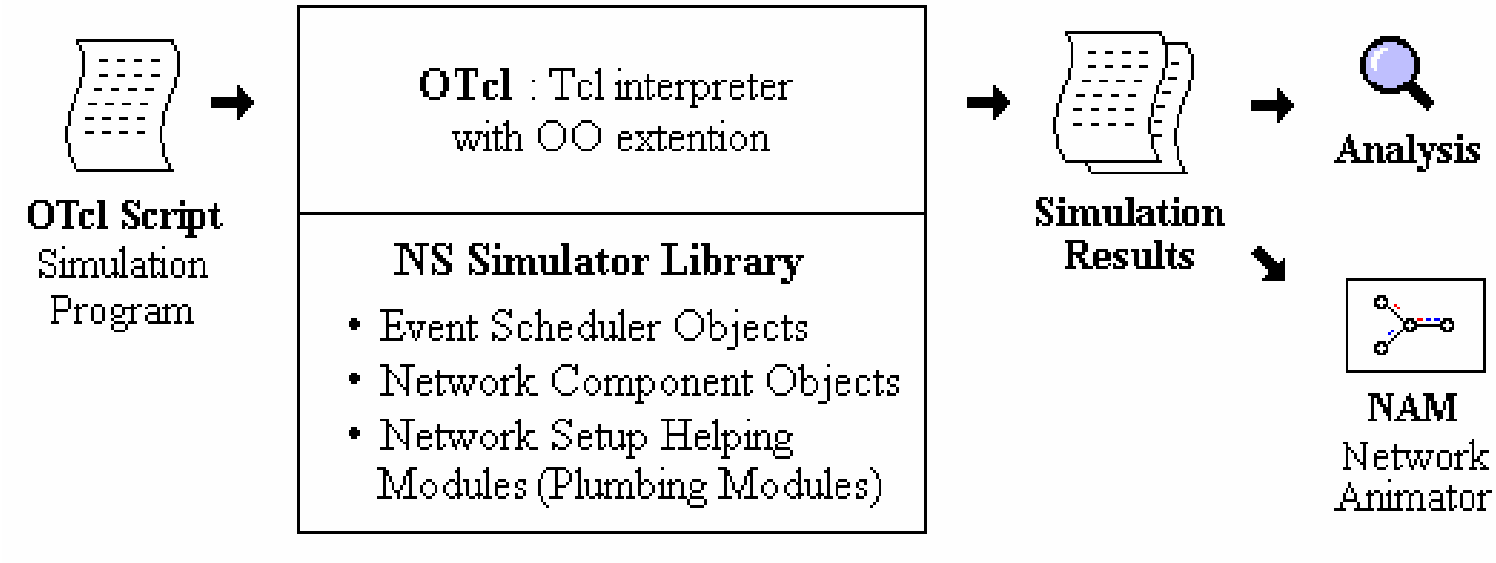
- **Animazione:**
  - **un particolare file di trace generato da NS consente di visualizzare una animazione della simulazione mediante NAM (Network Animator Module)**



Visualizzazione dei risultati

- **Variabili statistiche:**
  - **E’ infine possibile inserire nello script Tcl alcuni comandi che consentono di registrare in un file alcuni 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 in 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 (talora) 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 al 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 e Code

- **Sintassi:**

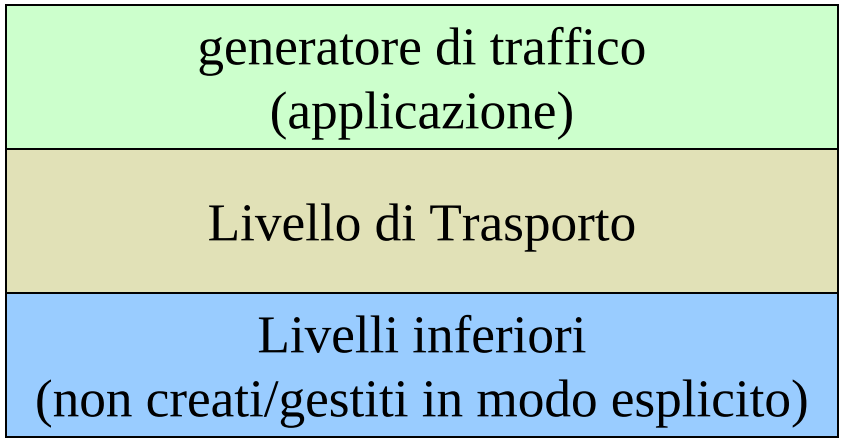
```
$ns simplex-link <node0> <node1> <bandwidth> <delay> <queue_type>
```

- `<bandwidth>` 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 coda con gestione FIFO (First In First Out) in cui viene scartato l'ultimo pacchetto in arrivo al nodo se non c'è più posto nella coda stessa; 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 alle `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 protocolli di livello trasporto definiti in NS sono UDP e 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 sono di diretto interesse per questo corso (ed anche per il corso di *Architetture e Protocolli per Internet*)**
- **Tuttavia, nella prima parte di questo corso faremo uso 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 dell'agente TCP è 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
```

- Suggerisco, negli script TCL, di specificare direttamente il rate (variabile rate_)

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 di dimensione 100 byte, rate 200 kbit/s)
  - la sorgente CBR inizia a trasmettere al tempo 0.5s e finisce al tempo 4.5s
  - la simulazione termina al tempo 5s
  - Visualizzare cosa accade tramite NAM

esercizio1a.tcl

Esercizio 1b - il primo script OTcl

- Con riferimento all'esercizio precedente, modificare lo script tcl per simulare:
  - Nodo 1 comunica con Nodo 2 tramite UDP con traffico CBR (stesso rate e dimensione pacchetti di prima)
  - Nodo 2 comunica con Nodo 1 tramite UDP con traffico CBR
- In entrambi di casi, 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 Esponenziale ON/OFF con questi parametri:
  - tempo medio di ON: 50ms
  - tempo medio di OFF: 100ms
  - rate di trasmissione: 200 kbit/s

esercizio1c.tcl

Esercizio 1d - il primo script OTcl

- **Nello stesso scenario del caso 1a, scrivere uno script TCL che mostri l'andamento nel tempo del ritmo di trasmissione (rate) della sorgente CBR**
- **A tal fine si utilizzino come *sink* degli Agent/LossMonitor**
- **Si misuri un campione della grandezza *rate* ogni 0.1 secondi.**

esercizio1d.tcl

Esercizio 1e - il primo script OTcl

- **Nello stesso scenario del caso 1c, scrivere uno script TCL che mostri l'andamento nel tempo del ritmo di trasmissione (rate) della sorgente Exponential On-Off.**

esercizio1e.tcl

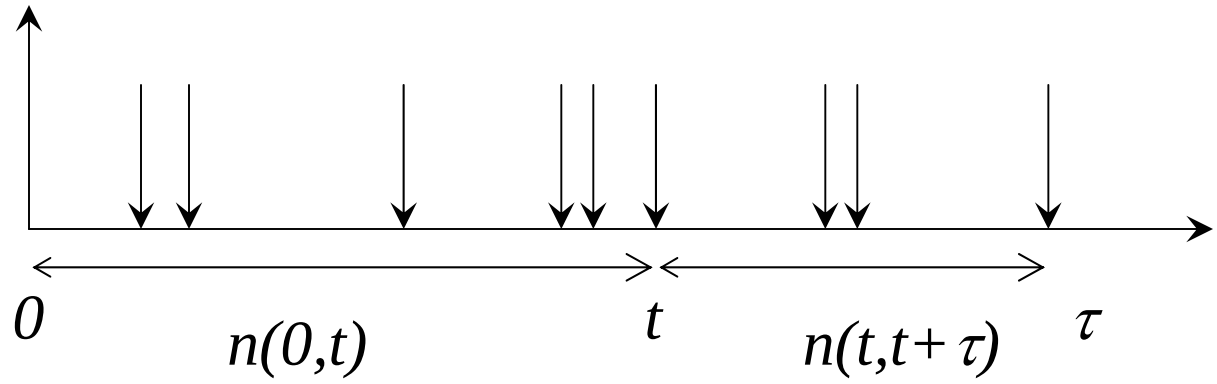
Esercizio 1f - il primo script OTcl

- **Si simuli lo stesso scenario del caso 1a, considerando però una sorgente di traffico Poissoniana, sempre con pacchetti lunghi 100 byte e rate (medio) di trasmissione 200 kbit/s.**

esercizio1f.tcl

Richiamo sulle Sorgenti di Poisson

- **Processo puntuale:**
  - **descrive la posizione di punti su un asse orientato (in generale l'asse temporale!)**
  - **descrizione:**
    - **n(0,t) numero di punti nell'intervallo [0,t]**
    - **n(t,t+τ) numero di punti nell'intervallo [t,t+τ]**



Richiamo sui processi casuali

- **Processo di Poisson:**
  - **1) La probabilità che ci sia un punto di Poisson in un intervallo infinitesimo *dt* è pari a:**

$$P\left[n(t,t+dt)=1\right]=\lambda dt$$

  - **dove il paramtero λ rappresenta la frequenza del processo (in punti per unità di tempo).**

Richiamo sui processi casuali

- **Processo di Poisson:**
  - **2) La probabilità che ci siano più punti in un intervallo infinitesimo dt è nulla**

$$P\left[n(t, t+dt) > 1\right] = 0$$

  - **3) Il numero di punti presenti in intervalli di tempo disgiunti sono variabili causali indipendenti.**

Richiamo sui processi casuali

- **Processo di Poisson:**
  - **La probabilità che vi siano *k* punti di Poisson in un intervallo temporale τ è pari a:**

$$P\left[n(t,t+\tau)=k\right]=\frac{(\lambda\tau)^k}{k!}e^{-\lambda\tau}$$

Rate trasmissivo sorgente Poissoniana Es.1f (campionato ogni 100 ms)