The Network Simulator NS2

casalicchio@ing.uniroma2.it

http://www.ce.uniroma2.it/courses/MMI/

http://www.uniroma2.it/didattica/MMI

- Introduzione
  - Documentazione e sorgenti
  - Architettura
  - Tcl/Otcl
- Esempio
- Componenti principali

Documentazione

- Questa lezione è basata sul materiale seguente
- http://www.isi.edu/nsnam/ns/
  - Marc Greis's tutorial
  - "NS for Beginners" by Altman and Jimenez
  - Ns Manual
  - “NS by Example” by J.Chung and M.Claypool http://nile.wpi.edu/NS/
  - Varie presentazioni e tutorial “ns workshops and presentations”
- Tcl/tk, http://www.tcl.tk/
- Otcl,

http://bmrc.berkeley.edu/research/cmt/cmtdoc/otcl/index.html

Installazione
http://www.isi.edu/nsnam/ns/ns-build.html

All-in-one (Consigliata – Linux/Windows/Mac):

- Tcl release 8.4.18 (required component)
- Tk release 8.4.18 (required component)
- Otcl release 1.13 (required component)
- TclCL release 1.19 (required component)
- Ns release 2.33 (required component)
- Nam release 1.13 (optional component)
- Xgraph version 12 (optional component)
- CWeb version 3.4g (optional component)
- SGB version 1.0 (?) (optional component, builds sgblib for all UNIX type platforms)
- Gt-itm gt-itm and sgb2ns 1.1 (optional component)
- Zlib version 1.2.3 (optional, but required should Nam be used)

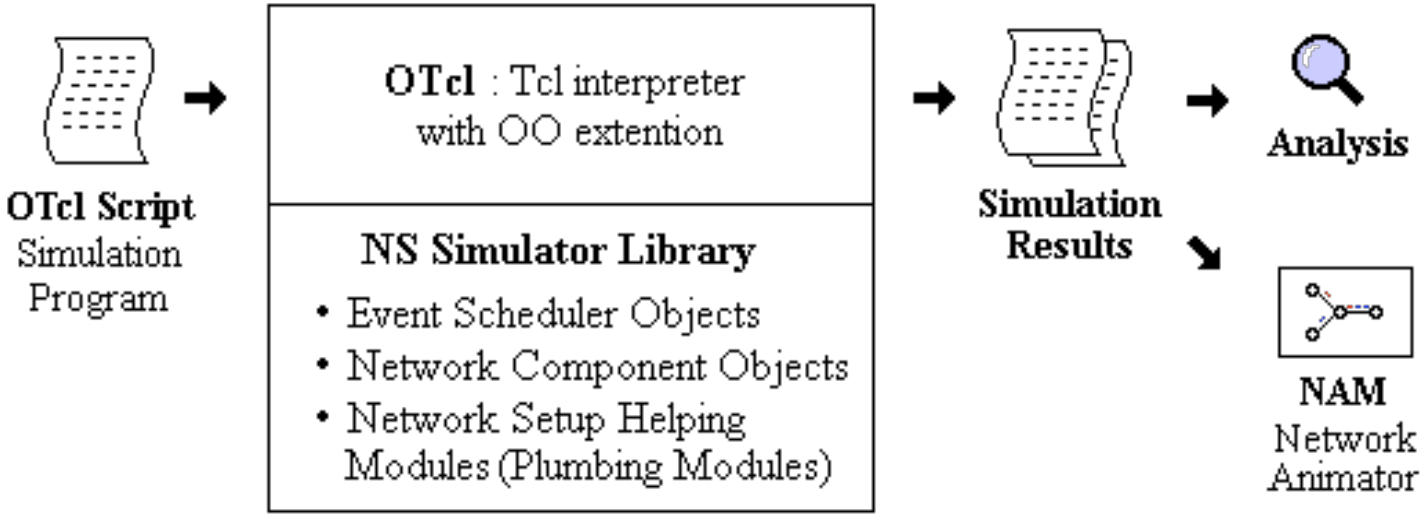
Oppure…

Installazione e configurazione separata dei singoli componenti

Cos'è NS2

- **Discrete-event simulator**
  - Sviluppato all' UC Berkeley
- Simulazione a **livello di pacchetto**
- Modellazione dal livello **Data Link** a livello **Applicazione**
  - protocolli livello MAC (per simulazioni LAN)
  - Algoritmi di routing: Dijkstra, etc…
  - Meccanismi di gestione delle code dei router: Drop Tail, 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,
- Open source (vasta comunità di sviluppatori ed utenti)
- Basato su
  - **C++** per simulation engine e modelli
  - **Tcl/OTcl (Object Tool Command Language)** per configurazione scenari e logica di simulazione

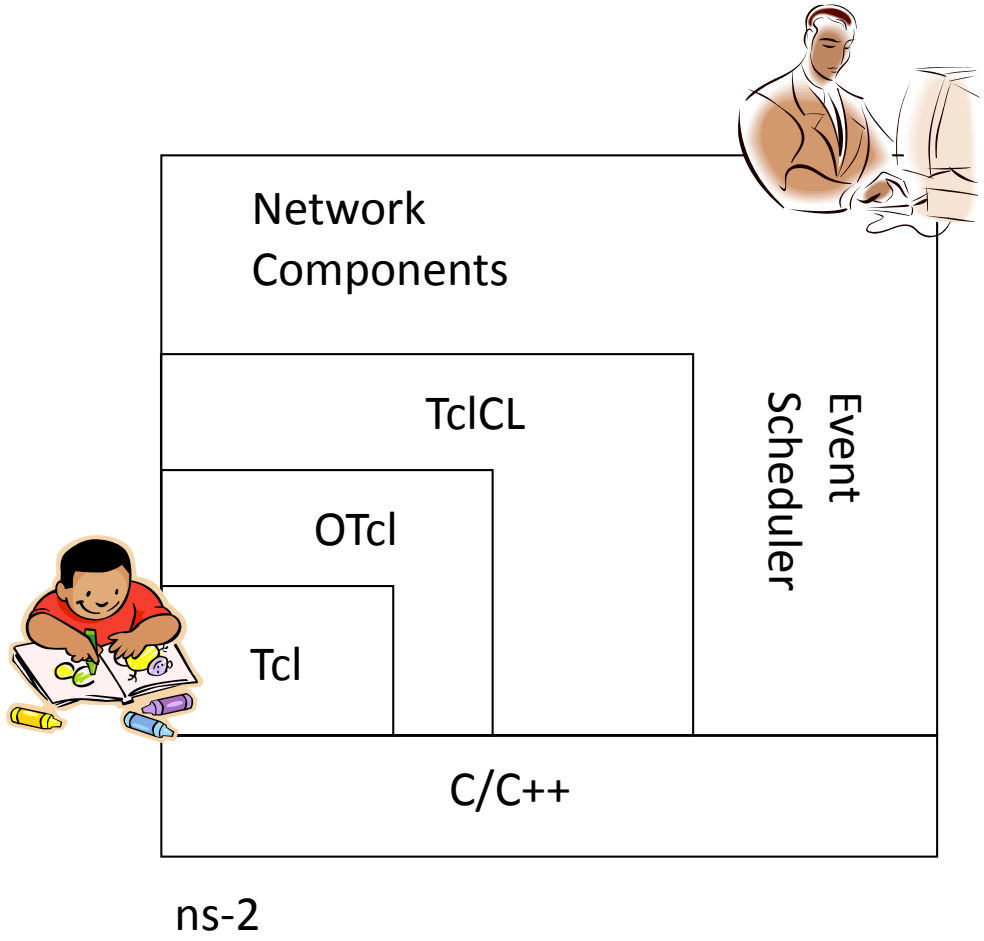
Vista utente: concetti fondamentali



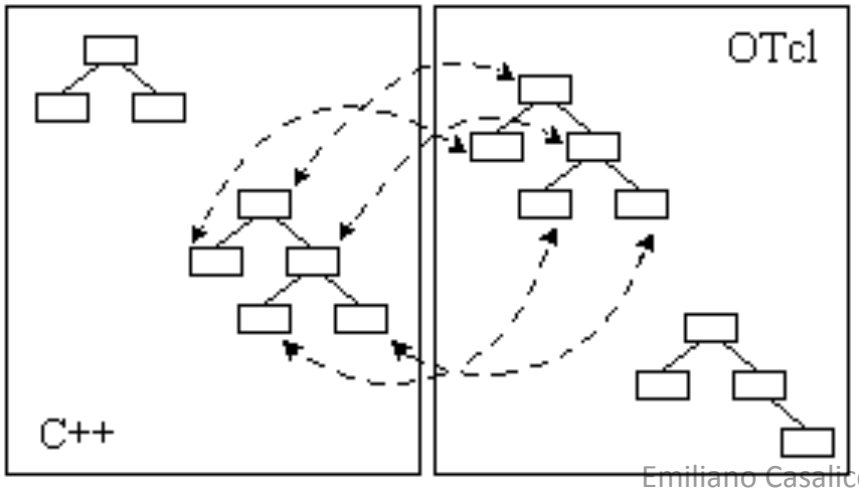
- Otcl per setup ed esecuzione simulazione
  - inizializzazione event scheduler,
  - Setup topologia utilizzando “network object” e “plumbing function”
  - Specificare inizio e fine generazione traffico (eventi)
- **Evento** = (**packetID**, **Schedule Time**, **Obj Pointer**)
  - L’**Obj** è un network object che gestirà il pacchetto
- L’**Event Scheduler**
  - Mantiene il tempo simulato
  - Estrae **(fire)** tutti gli eventi (pacchetti) che occorrono al tempo **t** invocando i relativi componenti di rete (network object/component)
- I componenti di rete comunicano scambiandosi pacchetti

Architettura

- Scheduler e Network components scritti in C++
- Corrispodenza (OTcl linkage - TclCL) tra oggetti C++ e OTcl
  - Solo per gli oggetti che modellano network component



ns-2



Hello World - Interactive Mode

```
swallow 71% ns
% set ns [new Simulator]
_o3
% $ns at 1 “puts \“Hello World!\””
1
% $ns at 1.5 “exit”
2
% $ns run
Hello World!
swallow 72%
```

Hello World - Batch Mode

simple.tcl

```
set ns [new Simulator]
$ns at 1 "puts \"Hello World!\""
$ns at 1.5 "exit"
$ns run
```

```
swallow 74% ns simple.tcl
Hello World!
swallow 75%
```

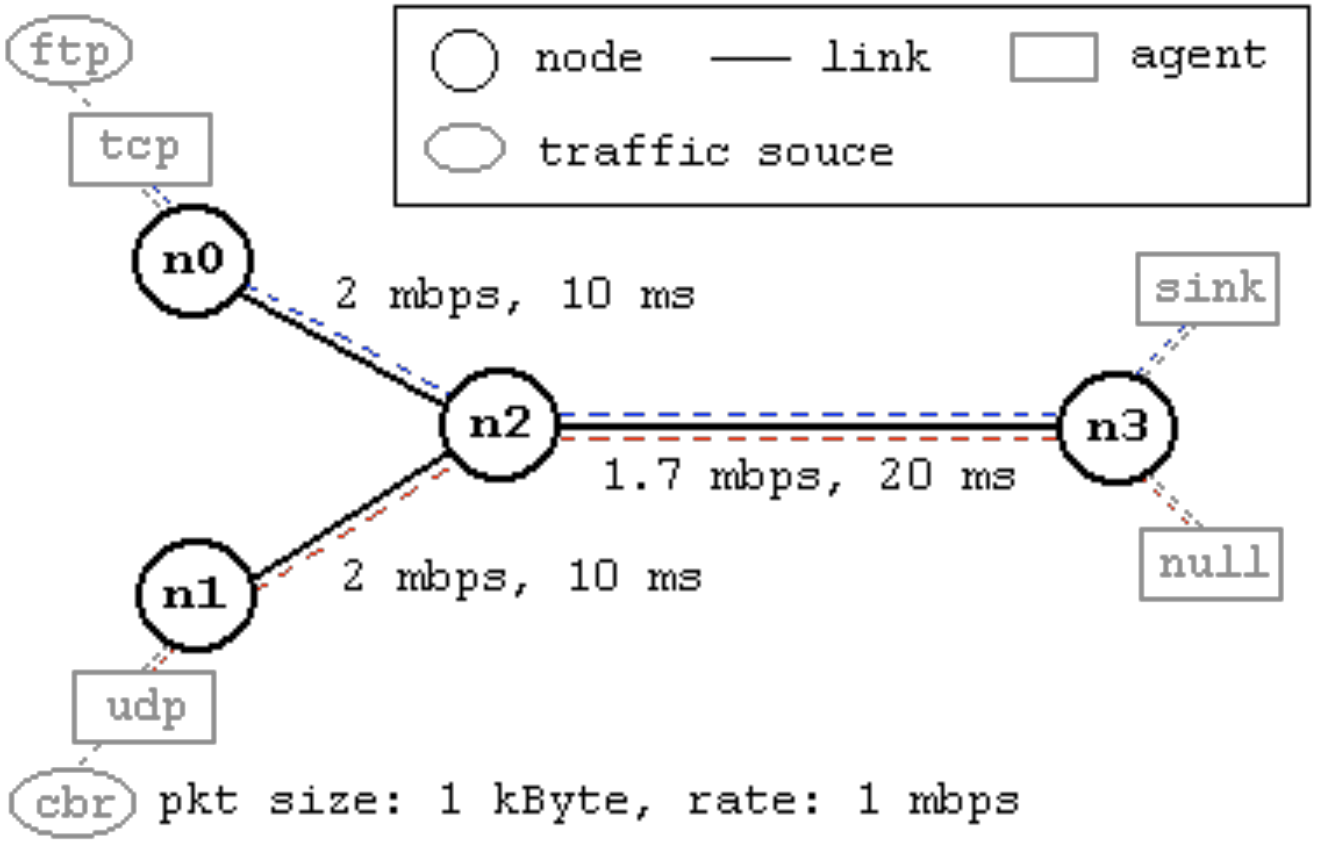
Basic tcl

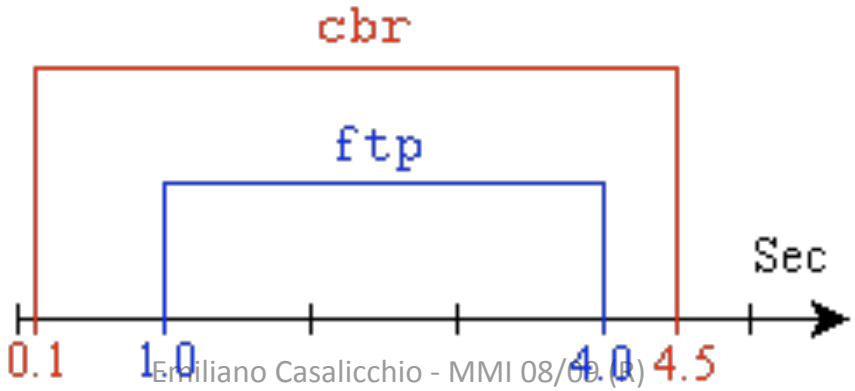
```
proc test { a b } {
   set c [expr $a + $b]
   set d [expr [expr $a - $b] * $c]
   for {set k 0} {$k < 10} {incr k} {
     if {$k < 5} {
       puts "k < 5, pow = [expr pow($d, $k)]"
     } else {
       puts "k >= 5, mod = [expr $d % $k]"
     }
   }
}

test 43 27

%oppure

set aa 43
set bb 27
Test $aa $bb
```

Output

```
k < 5, pow = 1.0
k < 5, pow = 1120.0
k < 5, pow = 1254400.0
k < 5, pow = 1404928000.0
k < 5, pow = 1573519360000.0
k >= 5, mod = 0
k >= 5, mod = 4
k >= 5, mod = 0
k >= 5, mod = 0
k >= 5, mod = 4
```

Esempio (ex-simple.tcl)





Instance simulation object

- **set *ns* [new Simulator]:** generates an NS simulator object instance, and assigns it to variable ***ns***. To instatiate a Simulator object means:
  - Initialize the packet format (ignore this for now)
  - Create a scheduler (default is **calendar scheduler**)
  - Select the default address format (ignore this for now)
- The "Simulator" object has member functions to:
  - Create compound objects such as nodes and links (described later)
  - Connect network component objects created (ex. attach-agent)
  - Set network component parameters (mostly for compound objects)
  - Create connections between agents (ex. make connection between a "tcp" and "sink")
  - Specify NAM display options
- The "Simulator" object member function implementations are located in the "ns-2/tcl/lib/ns-lib.tcl" file.

NAM setup

- **$ns color *fid color*:**
  - is to set color of the packets for a flow specified by the flow id (f***id***).
  - is for the NAM display, and has no effect on the actual simulation.
- **$ns namtrace-all *file-descriptor***:
  - tells the simulator to record simulation traces in NAM input format.
  - gives the file name that the trace will be written to later by the command **$ns flush-trace**.
  - Similarly, the member function trace-all is for recording the simulation trace in a general format.

Network object setup

- **proc *finish* {}**:
  - function that specify post-simulation processes
  - is called after the simulation is over by the command **$ns at 5.0 "finish"**
- **set *n0* [$*ns* node]**:
  - creates a node. A node in NS is compound object made of address and port classifiers
  - Users can create a node by separately creating an address and a port classifier objects and connecting them together.
  - To see how a node is created, look at the files: "ns-2/tcl/libs/ns-lib.tcl" and "ns-2/tcl/libs/ns-node.tcl".

- **$ns duplex-link *node1 node2 bandwidth delay queue-type***:
  - creates two simplex links of specified bandwidth and delay, and connects the two specified nodes.
  - In NS, the output queue of a node is implemented as a part of a link, therefore users should specify the queue-type when creating links.
  - In the above simulation script, DropTail queue is used.
  - Link source codes can be found in "ns-2/tcl/libs/ns-lib.tcl" and "ns-2/tcl/libs/ns-link.tcl" files.
  - the user can insert error modules in a link component to simulate a lossy link

- **$ns queue-limit *node1 node2 number***:
  - sets the queue limit of the two simplex links that connect node1 and node2 to the number specified.
  - take a look at "ns-2/tcl/libs/ns-lib.tcl" and "ns-2/tcl/libs/ns-link.tcl", or NS documentation for more information.
- **$ns duplex-link-op *node1 node2 ...***:
  - Effect the NAM animation showing flows of packets
  - To see the effects of these lines, users can comment these lines out and try the simulation.

Setup Traffic Agents

- **set *tcp* [new *Agent/TCP*]**:
  - creates a TCP agent. But in general, users can create any agent or traffic sources in this way.
  - Agents and traffic sources are in fact basic objects (not compound objects), mostly implemented in C++ and linked to OTcl. Therefore, there are no specific Simulator object member functions that create these object instances.
  - To create agents or traffic sources, a user should know the class names these objects (Agent/TCP, Agnet/TCPSink, Application/FTP and so on).
  - This information can be found in the NS documentation or partly in this documentation. But one shortcut is to look at the "ns-2/tcl/libs/ns-default.tcl" file. This file contains the default configurable parameter value settings for available network objects.

- **$*ns* attach-agent *node agent***:
  - attaches an agent object created to a node object. Actually, what this function does is call the attach member function of specified node, which attaches the given agent to itself. Therefore, a user can do the same thing by, for example, $n0 attach $tcp. Similarly, each agent object has a member function attach-agent that attaches a traffic source object to itself.

- **$*ns* connect *agent1 agent2***:
  - to establish a logical network connection between two agents. This line establishes a network connection by setting the destination address to each others' network and port address pair.

Scenario setup

- **$ns at *time* "*string*"**:
  - makes the scheduler (scheduler_ is the variable that points the scheduler object created by [new Scheduler] command at the beginning of the script) to schedule the execution of the specified string at given simulation time.
  - For example, **$ns at 0.1 "$cbr start"** will make the scheduler call a **start** member function of the CBR traffic source object, which starts the CBR to transmit data.
- **In NS, usually a traffic source does not transmit actual data, but it notifies the underlying agent that it has some amount of data to transmit, and the agent, just knowing how much of the data to transfer, creates packets and sends them.**