Università degli Studi del Sannio
Facoltà di Ingegneria
CDL in Ingegneria Informatica
Corso di Misure su Reti di Calcolatori

Simulazione di tecniche di scheduling di un router per QoS applicate al VoIP

Prof. De Vito Luca

Guadagno Potito 393/65
Pagliarulo Luana 393/66
Parente Cosimo 393/80
Zardo Pietro Giuseppe 393/67

a.a. 2006-2007

Sommario

1. Introduzione

Lo scopo del progetto è la simulazione via software del comportamento di una rete VoIP in funzione delle diverse condizioni di traffico di rete e delle diverse tecniche di scheduling applicate nei router per la QoS.
Per lo sviluppo del progetto è stato necessario operare con diversi software specifici, sia per la definizione della simulazione da compiere che per la sua esecuzione, sia per l'interpretazione dei risultati provenienti dalla stessa.

L'elaborato è organizzato in modo da descrivere in prima istanza i tools impiegati per la simulazione della rete e l'elaborazione dei risultati: NS (Simulatore di reti basate su IP) , Nscript (Tool di sviluppo grafico per NS), NAM (Modulo per la generazione di animazioni delle simulazioni) e Tracegraph (Software per la generazione di grafici inerenti la simulazione).

Successivamente l'architettura generale della rete VoIP oggetto della simulazione, specificando la topologia, le tipologie di traffico e le politiche di scheduling adottate nei router.

E per terminare è illustrato il confronto tra le diverse configurazioni dei ruoter circa le prestazioni relative al VoIP tramite una serie di grafici esplicativi e le conclusioni finali.

2. Ambiente di lavoro

2.1 *Introduzione ad NS*

NS è un simulatore di rete IP-based, nato come variante del simulatore di rete *REAL* e implementato dall'Università di Berkeley nel 1989. Nel 1995 lo sviluppo fu supportato anche dal gruppo DARPA (una collaborazione tra USC/ISI, Xerox PARC, LBNL e Università di Berkeley) attraverso il progetto VINT (Virtual InterNetwork Testbed), e numerosi continuano ad essere ancora oggi i contributi di altre ricerche, da ACIRI a UCB Daedalus fino a Sun Microsystem, data la natura opensource del simulatore.

NS utilizza come linguaggio di programmazione il *Tcl* (Tool Command Language), per descrivere tutte le operazioni necessarie alla definizione di una simulazione.
Mediante comandi Tcl è infatti possibile impostare lo scenario di rete, oggetto dello studio di simulazione (la topologia della rete, le proprietà dei collegamenti, i protocolli da usare, ecc.) e la sua evoluzione temporale (la composizione del traffico, la generazione e la terminazione delle sessioni, eventuali modifiche della topologia della rete), elaborare i dati acquisiti ed ottenere i risultati cercati.
Inoltre, sempre attraverso comandi Tcl, è possibile effettuare operazioni come la gestione dei file o le operazioni di ingresso/uscita.
NS è stato costruito semplicemente aggiungendo le funzioni specifiche di un simulatore di reti ai servizi di base offerti da Tcl (e da OTcl, che è la versione Tcl orientata alla programmazione di oggetti).
Il motore di simulazione (*engine*) di NS è scritto interamente in C/C++ e utilizza un interprete Tcl/OTcl come interfaccia utente.
Questo software di simulazione è di pubblico dominio e può essere scaricato gratuitamente all'indirizzo *http://www.isi.edu/nsnam/ns/* .

Tcl/OTcl è un linguaggio di scripting (disponibile sia in ambiente Unix/Linux che Windows) in cui le istruzioni vengono interpretate ed eseguite una linea alla volta. Questa è l'interfaccia attraverso cui l'utente descrive il modello di simulazione, indicando topologia della rete, sorgenti di traffico, attivazione e così via.
I comandi possono essere digitati uno dopo l'altro dal prompt del simulatore oppure raccolti in un file di testo usato come input dal programma.

Il software simula il funzionamento della rete descritta nel file di configurazione della simulazione e produce l'insieme dei file di risultati.

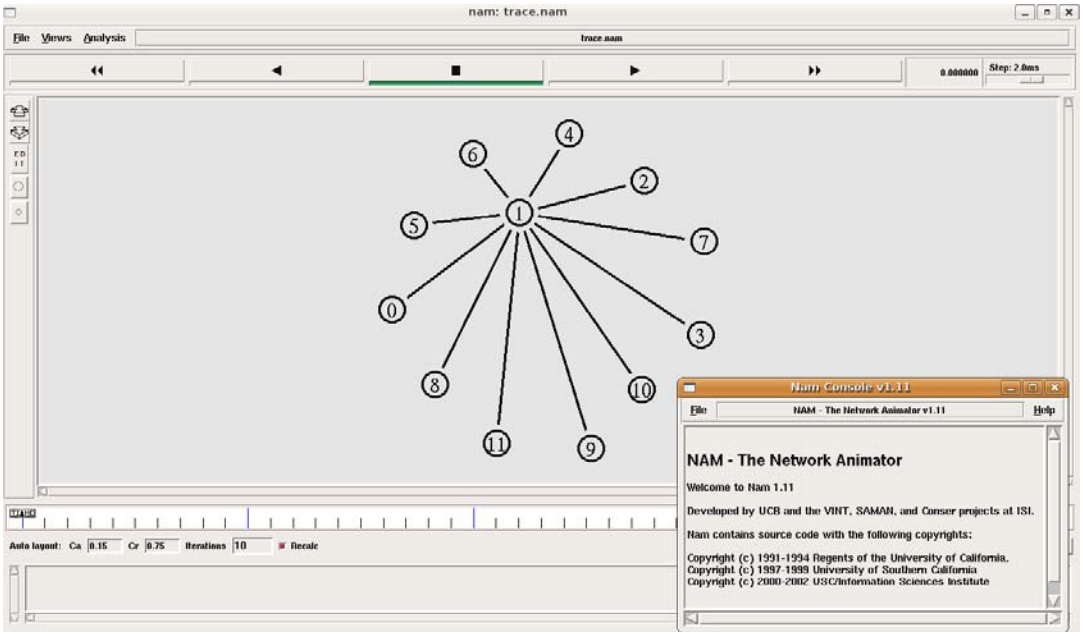
Tra le diverse caratteristiche di NS2 è possibile impostare il simulatore in modo da generare un file di testo (tipicamente con estensione *.nam*) in cui vengono registrati tutti gli eventi che avvengono durante la simulazione, ed in particolare gli istanti di generazione e ricezione di ogni singolo pacchetto ed il loro percorso dettagliato nella rete.

Questo file può essere successivamente processato da un programma dotato di interfaccia grafica, denominato *NAM* (Network Animator Module) che permette di ricostruire la simulazione attraverso un'animazione.

In questo modo possiamo avere un riscontro visivo del funzionamento della rete e dell'interazione fra i vari componenti; inoltre nel corso dell'animazione è possibile realizzare dei fermo-immagine, aumentare o diminuire la velocità di riproduzione e, "cliccando" sui vari elementi grafici che compongono la simulazione, ottenere informazioni specifiche (ad esempio, da un collegamento si ottiene il numero dei pacchetti persi su quel link).

2.2 *NAM "Network Animator Module"*

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



Lanciando nam viene visualizzata una console. E' possibile avere animazioni multiple sotto la stessa istanza nam. Vi è una barra dei menu che offre il menù 'File' e 'Help'.
Il menu 'File' offrè il commando 'New' per la creazione di una topologia di rete ns usando il nam editor, un commando 'Open' che permette di caricare esistenti tracefiles, poi un commando 'WinList' che mostra un pop-up con i nomi di tutti i tracefile correntemente aperti, un comando 'Quit' che permette di uscire da nam.
Il menu help contiene un "pop-up help screen" ed un comando che mostra la versione ed informazioni sul copyright.

Una volta che un tracefile è stato caricato in nam apparirà una finestra di animazione. Una configurazione di rete può essere salvata e stampata tramite i relativi comandi.

Il menù 'Views' ha 4 bottoni:

* New view button: crea una nuova view della stessa animazione che può essere navigata e ridimensionata. Tutte le viste sono animate in modo sincrono.
* Show monitors checkbox: Se selezionata, mostra un pannello in basso alla finestra principale, in cui i vari monitor definiti saranno visualizzati.
* Show autolayout checkbox: se selezionata, mostra un pannello in basso nella finestra principale, che contiene input box e un bottone per un layout automatico. Questo pannello non viene abilitato quando si usano link orientati.

* Show annotation checkbox: se selezionata, mostra una listbox in basso alla finestra principale, con l'elenco delle annotazioni in ordine cronologico.

Sotto la barra dei menu, vi è una barra di controllo che contiene 6 bottoni, un'etichetta, ed un piccolo scrollbar.

- Button 1 («) - Rewind. Quando cliccato il tempo di animazione viene portato indietro.
- Button 2 (<) – Play indietro. Quando cliccato l'animazione viene eseguita all'indietro.
- Button 3 (||) - Stop. Quando cliccato l'animazione viene sospesa.
- Button 4 (>) – Play avanti. Quando cliccato l'animazione viene eseguita in avanti
- Button 5 (») – Fast. Quando cliccato l'animazione viene spostata in avanti.
- Button 6 ([]) – L'animazione viene fermata.

Time label – mostra il tempo di animazione corrente (i.e., tempo di simulazion).

Rate Slider – indica la velocità di aggiornamento dello schermo. La percentuale corrente è mostrata nell'etichetta sopra lo slider.

Il codice dello script mostrato in seguito può essere eseguito dal simulatore NAM.

Esempio di script Tcl

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0 }

#Create two nodes
set n0 [$ns node]
set n1 [$ns node]

#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#Create a CBR agent and attach it to node n0
set cbr0 [new Agent/CBR]
$ns attach-agent $n0 $cbr0
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005

#Create a Null agent (a traffic sink) and attach it to node n1
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0

#Connect the traffic source with the traffic sink
$ns connect $cbr0 $null0

#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Per effettuare una simulazione con NS occorre:

- *descrivere lo scenario della simulazione* (nodi, link, sorgenti di traffico, ecc.)
- *eseguire la simulazione*
- *visualizzare i risultati*

La descrizione dello scenario della simulazione avviene mediante uno script Tcl/Otcl, gli oggetti OTcl utilizzati nello script sono collegati ad oggetti descritti in C++ nel software di simulazione. In questa fase si specifica la topologia della rete, individuando le caratteristiche dei diversi elementi della rete e le interazioni tra essi. Inoltre si descrive anche l'evoluzione dei diversi eventi che avvengono in una rete, quali per esempio l'invio di dati da una particolare sorgente o la caduta di un nodo.
Nella fase di simulazione avviene l'esecuzione dello scenario descritto. Ed è proprio durante questa fase che vengono creati i file che immagazzinano le informazioni circa l'evoluzione della simulazione.
In particolare il tracefile, che tiene informazioni relative al processamento di ogni singolo pacchetto; i monitor file, che contengono informazioni più specifiche circa le statistiche ed il conteggio dei pacchetti; nam file, che descrive l'evoluzione degli eventi che avvengono durante la simulazione.
La visualizzazione dei risultati può essere ottenuta in modi differenti in base allo scopo che ci si prefigge, come attraverso la lettura dei file creati nella fase di simulazione. C'e la possibilità di analizzare il tracefile tramite appositi tool quali "TraceGraph" generando automaticamente una serie di grafici predefiniti; la possibilità di analizzare i monitor file attraverso l'uso di Xgraph per graficare informazioni specifiche; ed in fine si può visualizzare un' animazione della simulazione attraverso l'uso di "NAM" e gli appositi file.

2.1 *Elementi del Modello di Rete*

2.1.1 Simulator

Ogni script OTcl inizia con la creazione di una variabile di tipo Simulator e una volta creata viene utilizzata facendo precedere il nome dal simbolo $. Essa mette a disposizione una seria di metodi che consentono di specificare tutti gli elementi della simulazione, quali nodi, link, ed altro ancora.

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

2.1.2 Nodi

I nodi insieme ai link sono gli elementi base per la creazione di una topologia di rete. Possono essere di tipo unicast e multicast a seconda del numero di nodi verso cui inviano dati. Sono impiegati per connettere insieme protocolli ed applicazioni, configurandosi come punto sorgente o destinazione per la simulazione di applicazioni.
I nodi sono oggetti gestiti da Simulator e sono creati in questo modo: 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.

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

2.1.3 Link

Come nella realtà, i nodi sono connessi da link. Ogni link è caratterizzato da ritardo, ampiezza di banda, errori, e perdita di paccheti.
Ci sono due tipi :

- simplex-link (link unidirezionale)

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

- duplex-link (link bi-direzionale)

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

In definitiva rappresentano una connessione point-to-point, come quella tra router o switch e dispositivi di rete.
I parametri da specificare per un link sono:

<*node1,node2*>: I nodi agli estremi del link.

<*bandwidht*>: La capacità del link.

Es. 10Mb (10 Mb/s), 1.2kb (1.2 kb/s), 1.5e6 (1.5 Mb/s)

<*delay*>: Il ritardo di propagazione del link.

Es. 13s (13 secondi),1.34ms (1.34 ms)

<*queue_type*>: Il metodo di gestione della coda del link

Sono disponibili le seguenti tecniche: DropTail (una gestione di tipo FIFO), FQ (fair queuing), SFQ (stochastic fair queuing), RED (random early detection), DDR (deficit round-robin).

Inoltre è possibile stabilire le dimensioni delle code (in pacchetti):

```
$ns simplex-link <node0> <node1> <bandwidht> <delay> <queue_type>
$ns queue-limit $n0 $n1 10
```

2.1.4 Network Agent

Un agente di rete rappresenta un end-point dove sono generati ed utilizzati i pacchetti e rappresenta il protocollo di comunicazione utilizzato nella rete. Sono presenti diversi tipi di agenti uno per ogni diverso protocollo. Una coppia di agenti rappresentano il mittente ed il destinatario in una connessione su uno specifico protocollo.

I principali tipi di agenti sono:

- TCP: Modella un mittente TCP (invia un pacchetto ed attende un ack).
- UDP:Modella un mittente UDP (invia pacchetti senza attendere ack).
- TCPSink: Modella un destinatario TCP (risponde con un ack ad ogni pacchetto ricevuto).
- NULL: Modella un destinatario UDP (accetta pacchetti senza rispondere).

Un agente per operare deve essere necessariamente connesso ad un nodo, inoltre esso può operare unicamente con un altro agente. Non è consentito avere un agente ricevente con due mittenti o viceversa, quindi la corrispondenza è sempre uno a uno anche tra gli stessi nodi.

```
set udp0 [new Agent/UDP]
new Simulator attach-agent $n0 $udp0

set null0 [new Agent/Null]
new Simulator attach-agent $n1 $null0
```

Dopo essere stati istanziati, gli agenti, possono essere connessi tra di loro.

```
new Simulator connect $udp0 $null0
```

In definitiva tramite una connessione tra due agenti è possibile passare dati tra le applicazioni.

2.1.5 Network Application

Le “Network Application” modellano il traffico di dati sulla rete, ci sono diversi modelli di applicazione a seconda del tipo di applicazione di rete da simulare. Alcuni esempi sono:

- *Application/Traffic:* per generare pattern di traffico specifici.
- *Application/Telnet:* genera traffico simulante una sessione telnet.
- *Application/FTP:* genera traffico simulando una sessione FTP.

Altri modelli possono essere specificate dall’utente tramite classi C++. Ognuna delle applicazioni deve essere collegata ad uno specifico agente che simuli l’appropriato protocollo di comunicazione.

```
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp0
```

Di particolare interesse è l’ “Application/Traffic” che si suddivide in quattro sottoapplicazioni ognuna con caratteristiche di traffico differenti:

- *Traffic/Exponential:* Genera traffico in un ciclo on-off, il tempo dei periodi on-off è determinata da una distribuzione esponenziale.
- *Traffic/Pareto:* Genera traffico in un ciclo on-off, con una distribuzione Pareto.
- *Traffic/CBR:* Genera pacchetti continuamente con un bit rate costante ed è possibile specificare la dimensione dei pacchetti, il bit rate e l’intervallo tra questi.

- *Traffic/Trace:* Genera traffico basandosi sui dati contenuti in un trace file.

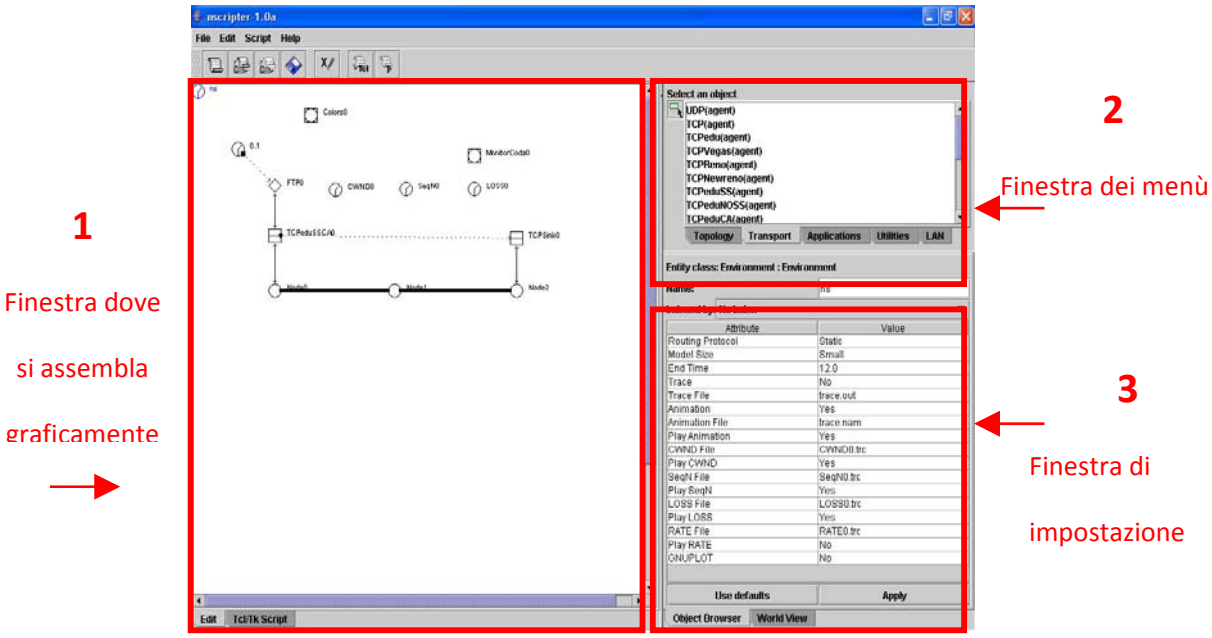
```
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set rate_ 200k
$cbr1 interval_ 50ms
$cbr1 maxpkts_ 2000
```

2.2 *NSCRIPT*

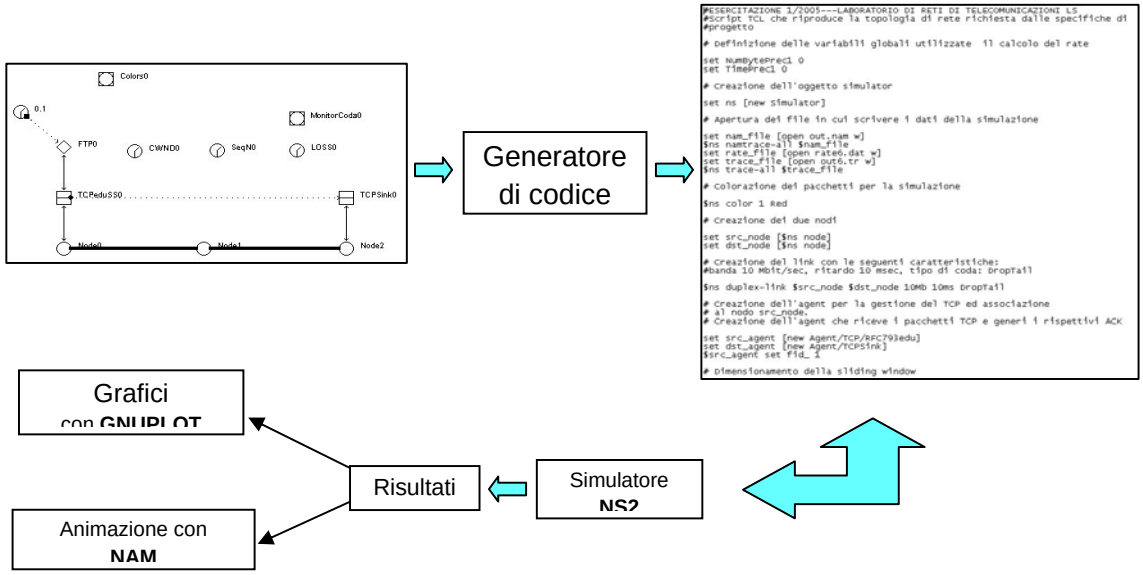
Nscript è un'interfaccia grafica costruita interamente con il linguaggio di programmazione Java ed è di supporto all'uso di NS2, in quanto serve per tradurre in codice Tcl i grafici delle topologie di rete.

L'interfaccia è costituita di quattro parti:

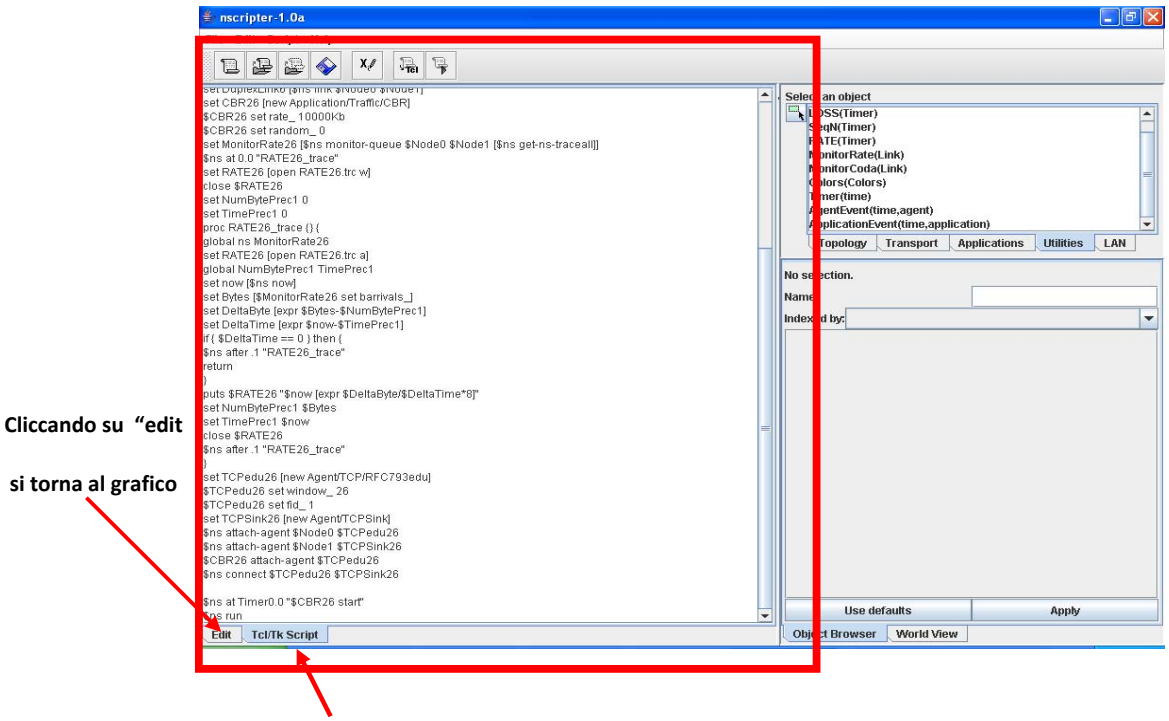
1. Finestra dove si assembla la rete da simulare;
2. Finestra dei menù, dove si scelgono
   - Topology: nodi e i link (duplex-link e simplex-link);
   - Transport: agent Trasport (TCP, TCP reno,..);
   - Applications: (FTP, CBR,...);
   - Utilities: dispositivi per il monitoraggio della finestra di trasmissione, delle perdite, del numero di sequenza;
   - Lan: in cui si trovano elementi per le reti Lan.
3. Finestra dove si settano i vari parametri degli oggetti che costituiscono la rete.
4. Pagina di scrittura dello script.tcl



2.2.1 Dal grafico della topologia ai risultati



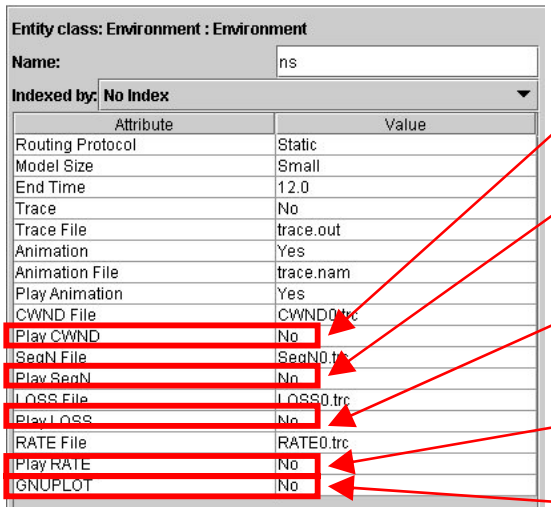
Per eseguire il settaggio dei parametri di simulazione, bisogna selezionare l'elemento "ns" in alto a sinistra nel grafico e compaiono tutti i parametri modificabili per quell'elemento.



Cliccando su "Tcl/tk script" si apre la pagina dello script.tcl

2.2.2 Interattività

Dopo le modifiche, a discrezione dell'utente, oltre al NAM possono essere avviati in automatico anche i grafici in Xgraph dei parametri da monitorare e/o Gnuplot che può essere utilizzato in alternativa a Xgraph.



Questo comando permette l'apertura in automatico della Finestra di Trasmissione graficata tramite Xgraph (per default è selezionato NO)

Questo comando permette l'apertura in automatico del Numero di Sequenza graficata tramite Xgraph (per default è selezionato NO)

Questo comando permette l'apertura in automatico delle Perdite sul link graficate tramite Xgraph (per default è selezionato NO)

Questo comando permette l'apertura in automatico del Rate graficato tramite Xgraph (per default è selezionato NO)

Questo comando permette l'apertura in automatico di Gnuplot (per default è selezionato NO)

2.3 *TRACEGRAPH*

Per l'analisi dei dati della simulazione e la realizzazione di grafici è stato utilizzato un software dedicato a questo scopo: TraceGraph.

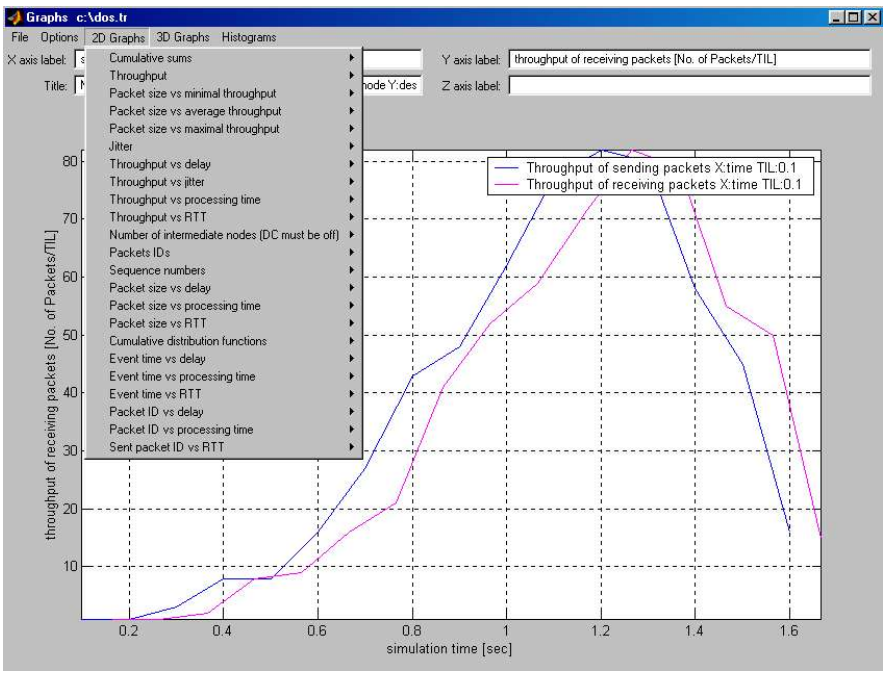
TraceGraph è un analizzatore dei trace file di Network Simulator ns2. Esso supporta i seguenti formati di trace file: wired, satellite, wireless, new trace, wired-cum-wireless.

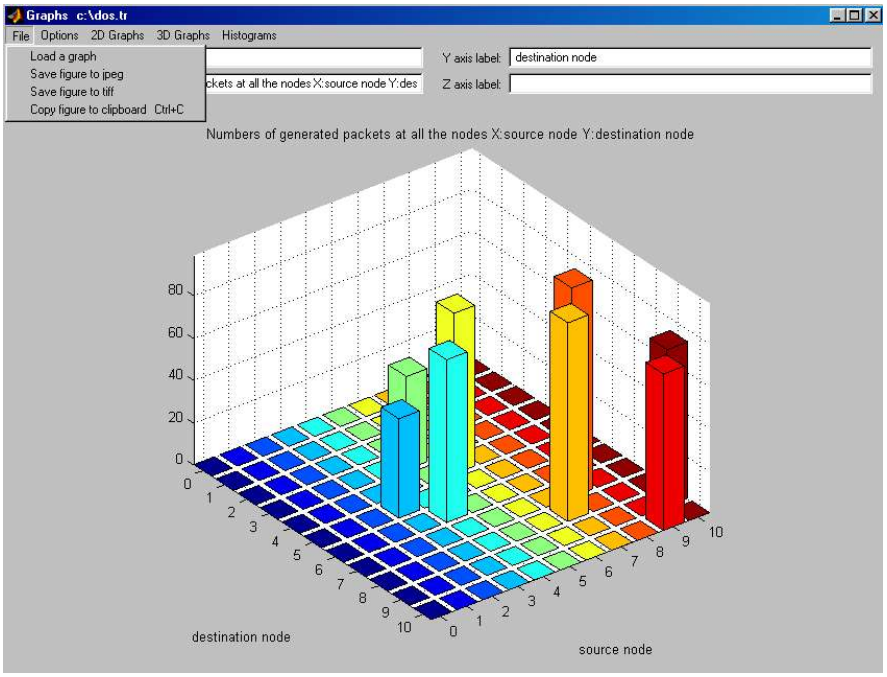
Consente di realizzare grafici in 2D, 3D ed istogrammi su molte delle misure di rete di interesse, in particolare :

ritardo, jitter, tempo di esecuzione, RTT, throughput.

Inoltre permette di effettuare una prima elaborazione dei dati generando grafici in funzione dei nodi di una rete, del flusso di dati tra nodi, in relazione ad eventi durante l'arco temporale. Per l'esecuzione del software sono necessarie le librerie di Matlab 6.0. Si può reperire presso il link "http://www.tracegraph.com/".

Di seguito sono riportati alcuni screenshots del software.




Graphs c:\dos.tr
File Options 2D Graphs 3D Graphs Histograms
Load a graph
Save figure to jpeg
Save figure to tiff
Copy figure to clipboard Ctrl+C
Y axis label: destination node
Z axis label:
Numbers of generated packets at all the nodes X:source node Y:destination node
destination node
source node

3. Definizione del Progetto

Il progetto ha come obiettivo la simulazione di una rete VoIP in funzione delle diverse condizioni di traffico di rete e delle diverse tecniche di scheduling applicate nei router per la QoS, illustrandone il comportamento in termini di pacchetti persi e scartati nelle code, e ritardo.

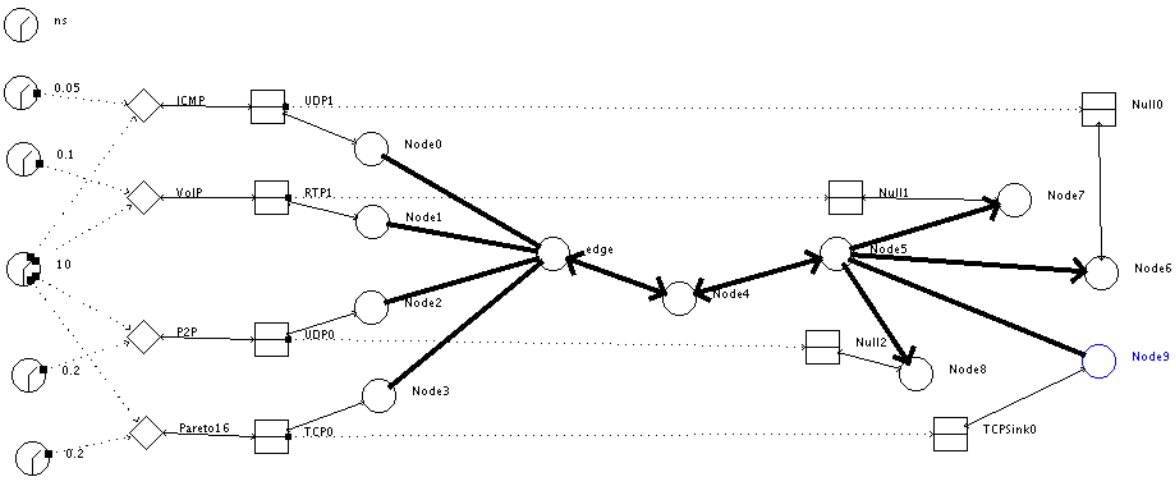
3.1 *Architettura generale*

Il progetto, come detto in precedenza, prevede la simulazione del comportamento del traffico VoIP in relazione ad altre tipologie di traffico. Per questo scopo si è pensato ad una architettura di rete che oltre al traffico VoIP, preveda anche altre tipologie di traffico di comune impiego, quali: HTTP, ICMP e P2P.
Per quanto riguarda la topologia di rete, si è immaginato uno scenario in cui le diverse sorgenti di traffico fossero presenti su una LAN connesse ad un router ADSL.

Particolare attenzione, inoltre, è stata posta nella implementazione della QoS.
Una architettura tradizionale, infatti, non distingue tra i vari flussi di traffico in arrivo, trattando tutte le connessioni della rete allo stesso modo; per ottenere una diversa Qualità del Servizio abbiamo bisogno di una gestione più sofisticata delle code.
L'esigenza di poter discriminare tra classi di applicazioni/connessioni e di poter trattare ciascuna connessione secondo la classe di appartenenza, in modo che, ad esempio, un utente disposto a pagare di più può ricevere una Qualità del Servizio migliore in termini di minore e più costante latenza e maggiore throughput (soprattutto nel caso di applicazioni multimediali), ha richiesto l'uso di un modello Differentiated Services.
Esso prevede di marcare i pacchetti secondo una priorità fissata all'entrata della rete concordemente al livello di Qualità del Servizio che si vuole fornire al flusso corrispondente; all'interno della rete tali pacchetti saranno trattati diversamente a seconda di come sono stati marcati.
NS-2 permette di classificare le connessioni in 4 classi differenti e ciascuna classe possiede una propria coda fisica. All'interno di ciascun flusso è possibile definire 3 sotto-

classi dotate di diversa “drop precedence”; per far ciò ad ogni coda fisica sono associate al massimo 3 code virtuali. In totale quindi sono disponibili un massimo di 12 combinazioni diverse a cui corrispondono altrettante etichette disponibili per marcare i pacchetti; tali etichette determinano nei nodi interni alla rete le politiche di scheduling delle code e, quindi, di scarto dei pacchetti.

Per creare code DiffServ nei nodi occorre usare link semplici e non full-duplex (per creare un link full-duplex si usano due link semplici).



La rete su cui abbiamo lavorato ha la topologia mostrata nella figura seguente: ci sono 4 sorgenti di traffico (nodi 0,1,2,3) e 4 destinatari (nodi 6,7,8,9).

Le sorgenti sono:

- Generatore di traffico HTTP su protocollo TCP (Nodo0)
- Generatore di traffico VOIP su protocollo UDP (Nodo1)
- Generatore di pacchetti ICMP (Nodo2)
- Generatore di traffico P2P su protocollo UDP (Nodo3)

Per la simulazione di un router si utilizzano due nodi che fungono da nodo *edge* e nodo *core*.

Il nodo *edge* (Nodo10) ha il compito di assegnare ad ogni pacchetto proveniente da una data sorgente una "*label*", etichetta, che permette di identificare un pacchetto in funzione del flusso di appartenenza.

Il nodo *core* (Nodo4) ha il compito di gestire le code utilizzando le varie politiche di scheduling inserite in fase di progettazione.

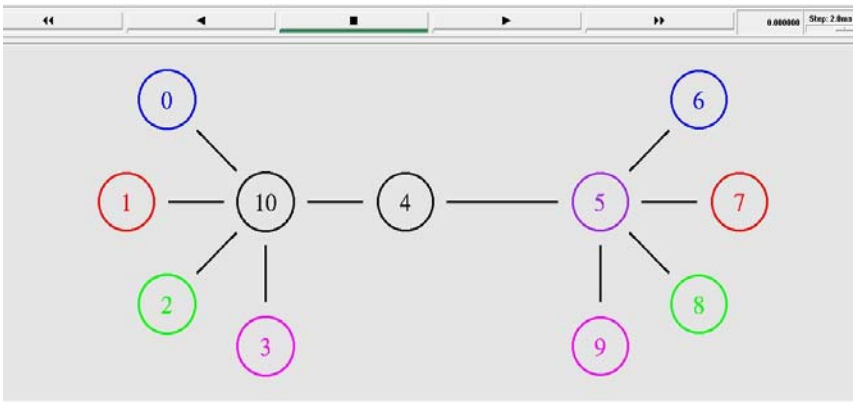
Per l'inoltro dei pacchetti verso le rispettive destinazioni si utilizza un nodo con la funzione di *switch* (Nodo5).

Il Nodo6 risponde ai pacchetti HTTP inviando pacchetti di risposta "*acknowledgement*".

I restanti nodi Nodo7 Nodo8 Nodo9 rappresentano le stazioni di destinazione.

3.2 *Parametri di configurazione dei link*

Per ogni coppia di nodi è stato necessario specificare uno o più link a seconda delle esigenze. In particolare tra i nodi di confine e quelli che si occupano dell' instradamento (10-router ,5 -switch) sono presenti dei link di tipo duplex mentre nei restanti casi dei link di tipo simplex. Questi ultimi sono stati adottati perchè richiesti dalla implementazione del modulo dei Differentiated Services di ns, per la gestione della QoS.



Per ognuno dei link è stata specificata una diversa dimensione ed una diversa politica di gestione della coda.

L'ampiezza di banda dei link simula una configurazione in cui si abbia una LAN connessa tramite un router ad una connessione ADSL a 1.2 Mbit.

3.2.1 Simulazione 1 – Politica di accodamento DropTail

```
$ns duplex-link $node0 $edge 1Mb 10ms DropTail
$ns duplex-link $node1 $edge 1Mb 10ms DropTail
$ns duplex-link $node2 $edge 1Mb 10ms DropTail
$ns duplex-link $node3 $edge 1.2Mb 10ms DropTail
# bottleneck link
$ns duplex-link $node5 $node6 1Mb 10ms DropTail
$ns duplex-link $node5 $node7 1Mb 10ms DropTail
$ns duplex-link $node5 $node8 1Mb 10ms DropTail
$ns duplex-link $node5 $node9 1Mb 10ms DropTail


$ns simplex-link $edge $node4 100Mb 10ms DropTail
$ns simplex-link $node4 $edge 100Mb 10ms DropTail
# bottleneck link
$ns simplex-link $node4 $node5 1.2Mb 20ms DropTail
$ns simplex-link $node5 $node4 1.2Mb 20ms DropTail
$ns queue-limit $node4 $node5 25
```

3.2.2 Simulazione 2 – 3 – 4 Politica di accodamento RandomEarlyDetection

```
$ns duplex-link $node0 $edge 1Mb 10ms DropTail
$ns duplex-link $node1 $edge 1Mb 10ms DropTail
$ns duplex-link $node2 $edge 1Mb 10ms DropTail
$ns duplex-link $node3 $edge 1.2Mb 10ms DropTail
# bottleneck link
$ns duplex-link $node5 $node6 1Mb 10ms DropTail
$ns duplex-link $node5 $node7 1Mb 10ms DropTail
$ns duplex-link $node5 $node8 1Mb 10ms DropTail
$ns duplex-link $node5 $node9 1Mb 10ms DropTail


$ns simplex-link $edge $node4 100Mb 10ms dsRED/edge
$ns simplex-link $node4 $edge 100Mb 10ms dsRED/core
# bottleneck link
$ns simplex-link $node4 $node5 1.2Mb 20ms dsRED/core
$ns simplex-link $node5 $node4 1.2Mb 20ms dsRED/edge
```

RoundRobin

```
set q23 [[$ns link $node4
$node5] queue]
$q23 meanPktSize 400
$q23 set numQueues_ 4
$q23 setNumPrec 2
$q23 addPHBEntry 11 1 0
$q23 addPHBEntry 00 0 0
$q23 addPHBEntry 22 2 0
$q23 addPHBEntry 33 3 0
$q23 addPHBEntry 44 0 1
$q23 configQ 0 0 20 40 0.02
$q23 configQ 2 0 20 40 0.02
$q23 configQ 1 0 10 20 0.10
$q23 configQ 3 0 10 20 0.10
```

WeightedRoundRobin

```
set q23 [[$ns link $node4
$node5] queue]
$q23 meanPktSize 400
$q23 set numQueues_ 4
$q23 setNumPrec 2
$q23 setSchedularMode WRR
$q23 addQueueWeights 0 2
$q23 addQueueWeights 1 3
$q23 addQueueWeights 2 4
$q23 addQueueWeights 3 1
$q23 addPHBEntry 11 1 0
$q23 addPHBEntry 00 0 0
$q23 addPHBEntry 22 2 0
$q23 addPHBEntry 33 3 0
$q23 addPHBEntry 44 0 1
$q23 configQ 0 0 20 40 0.02
$q23 configQ 2 0 20 40 0.02
$q23 configQ 1 0 10 20 0.10
$q23 configQ 3 0 10 20 0.10
```

Priority

```
set q23 [[$ns link $node4
$node5] queue]
$q23 meanPktSize 400
$q23 set numQueues_ 4
$q23 setNumPrec 2
$q23 setSchedularMode PRI
# riga per il PRI
$q23 addQueueRate 0 350Kb
$q23 addQueueRate 1 100Kb
$q23 addQueueRate 2 50Kb
$q23 addQueueRate 3 700Kb
$q23 addPHBEntry 11 1 0
$q23 addPHBEntry 00 0 0
$q23 addPHBEntry 22 2 0
$q23 addPHBEntry 33 3 0
$q23 addPHBEntry 44 0 1
$q23 configQ 0 0 20 40 0.02
$q23 configQ 2 0 20 40 0.02
$q23 configQ 1 0 10 20 0.10
$q23 configQ 3 0 10 20 0.10
```

3.3 *Specifiche delle sorgenti di traffico*

La simulazione prevede 4 tipologie di applicazioni che generano altrettante tipologie di traffico di rete. Ognuna di esse è associata ad una particolare coppia di nodi, al fine di una migliore rappresentazione.

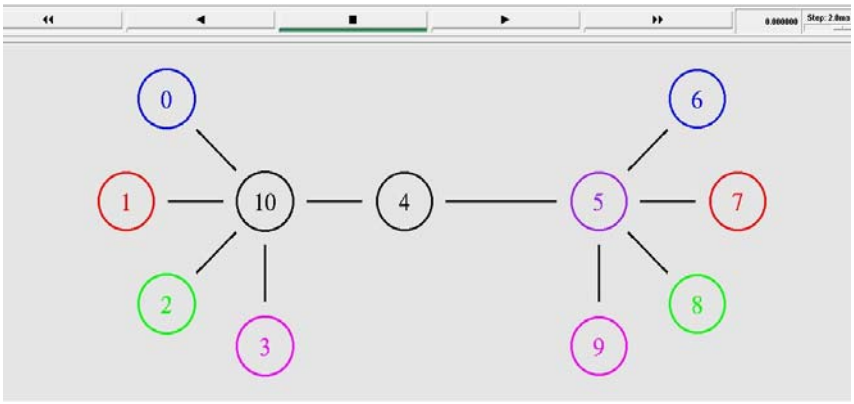
Per ognuna delle tipologie di traffico da simulare è stato specificato il protocollo di comunicazione, la dimensione dei pacchetti e la modalità di generazione degli stessi.

- ***Sorgente 0 – Destinazione 6 - HTTP su TCP – Pareto Distribution:***
  Simula una normale connessione HTTP che invia quantità di pacchetti variabili ad intervalli di tempo variabile.

- ***Sorgente 1 – Destinazione 7 - VOIP su UDP – 218 byte – 20ms:***
  Simula una connessione di tipo VoIP in cui si ha la generazione di pacchetti UDP da 218 byte ogni 20 ms. Ogni pacchetto è costituito da: 160 byte di dati vocali, 20 di intestazione IP, 8 di UDP, 16 di RTP, 14 Ethernet.

- ***Sorgente 2 – Destinazione 8 – ICMP su UDP – 38 byte – 50ms:***
  Simula traffico di controllo realizzato tramite l'invio di pacchetti UDP da 38 byte ogni 50 ms.

- ***Sorgente 3 – Destinazione 9 - PeerToPeer su UDP – 1500 byte – 10ms:***
  Simula traffico di rete di tipo P2P con l'intento di saturare la banda del canale.



3.4 *Politiche di Scheduling*

I pacchetti vengono marcati in modo differente a seconda dell'architettura utilizzata (Diffsev, Intserv, ecc.). Ogni architettura definisce infatti un campo specifico in un header, che identifica il tipo di paccheto. I nodi interni di rete classificano i pacchetti analizzando solamente questo campo. Per migliorare le prestazioni per quanto riguarda latenza e jitter è possibile intervenire sulla dimensione dei pacchetti in modo da evitare che pacchetti di grosse dimensioni e bassa priorità siano causa ritardi eccessivi al traffico prioritario.

Il *queue scheduling* rappresenta un insieme di algoritmi e meccanismi usati per controllare il trasferimento dei pacchetti dalle code di input a quelle di output nei router. Gli obiettivi sono quelli di condividere la banda in modo equo evitando la *starvation* di alcuni utenti, garantire i parametri di QoS come banda e latenza se specificati e ridurre il jitter. Gli algoritmi usati più frequentemente per realizzare questi obiettivi sono:

- *First-In First-Out (FIFO)*: la coda più semplice, il primo arrivato è il primo ad uscire. Tutti i pacchetti ricevono lo stesso trattamento. Funziona bene grazie alla sua semplicità in assenza di congestioni.
- *Priority (P)*: le code vengono servite dalla più bassa (0) alla più alta (3) e viene assegnata una quantità di banda massima ad ogni flusso; il router garantirà tale banda scartando pacchetti che superano la capacità assegnata.
- *Round Robin (RR):* sono presenti varie code. L'algoritmo base serve in ordine tutte le code, un pacchetto per volta.
- *Weighted Round Robin (WRR)*: Nella versione *weighted* è possibile specificare il numero di pacchetti serviti per le singole code ad ogni turno. Garantisce priorità e non provoca *starvation*. Non viene considerata la dimensione dei pacchetti, quindi possono verificarsi lunghe attese per pacchetti di piccole dimensioni. Non può quindi garantire limiti di banda o latenza.

Esistono anche altri metodi di gestione delle code, ad esempio variazioni agli algoritmi indicati, realizzazioni proprietarie di soluzioni generiche e nuove soluzioni che non sono generalmente presenti nei router.

Il *congestion control* è l'insieme dei meccanismi usati per prevenire ed eliminare le congestioni nelle reti. Vi sono tecniche reattive che tendono a diminuire l'intensità del traffico che attraversa la rete, e tecniche proattive che cercano di evitare il formarsi di congestioni. I principali approcci al problema utilizzati sono:

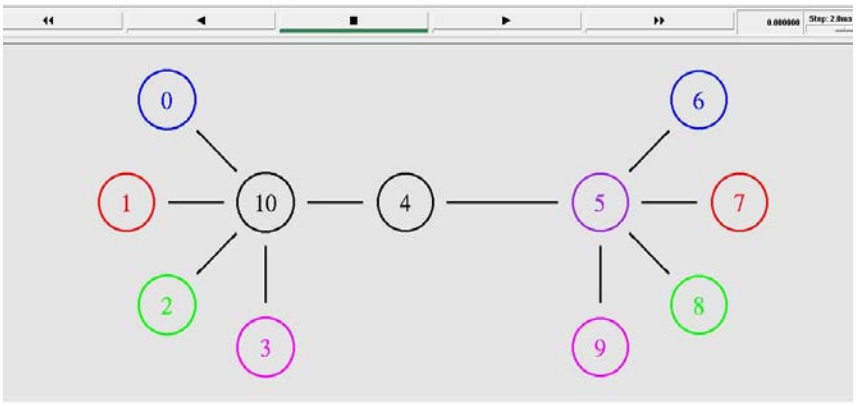
- *Tail Drop*: vengono eliminati i pacchetti ricevuti se la coda di output è piena.
- *Random Early Detection (RED)*: i pacchetti vengono scartati con una data probabilità in modo da evitare la formazione di congestioni. Quando le code sono lontane dalla saturazione e si è quindi in condizioni di traffico sostenibile dalla rete nessun pacchetto viene scartato. Se il numero di pacchetti in coda supera una soglia specificata il meccanismo RED comincia a scartare alcuni pacchetti in modo casuale; la probabilità che hanno i pacchetti di essere scartati aumenta con la percentuale di riempimento della coda, giungendo a scartare tutti i pacchetti in arrivo quando il buffer è pieno.

Le politiche di schedulino presentate sono state tutte implementate nella simulazione.

Per quanto riguarda il congestion control, invece, vanno distinti due casi, in funzione della tecnica di queue scheduling adottata. Nel caso in cui viene adottata la tecnica FIFO la congestione viene gestita su tutti i link tramite il "DropTail", negli altri casi la configurazione adottata è la seguente.

DropTail per i link "*Link0-10, Link1-10, Link2-10, Link3-10, Link5-6, Link5-7, Link5-8, Link5-9*"

Random-Early-Detection per i link "*Link10-4, Link4-10, Link4-5, Link5-4*".



É stato necessario adottare questa distinzione perchè nel secondo caso (non FIFO) non era possibile adottare una politica ti tipo "DropTail".

4. Analisi simulativa

In questo capitolo descriveremo lo studio simulativo che abbiamo condotto utilizzando ns (netSim il modulo che abbiamo implementato). Verificheremo dapprima il funzionamento della rete e mostreremo gli script utilizzati, i risultati delle varie sessioni di simulazione e le conclusioni tratte.

4.1 *Introduzione*

Il progetto si compone di 4 script in OTcl ciascuno relativo ad un caso analizzato. In particolare abbiamo simulato la rete utilizzando le diverse politiche di scheduling "RR", "WRR" e "Priority" e di accodamento “DropTail”, “RandomEarlyDetection”.
Ogni script OTcl si compone di 4 parti comuni:

Innanzitutto viene istanziato il simulatore, vengono aperti i file per raccogliere le informazioni di trace e per la simulazione con NAM.

```
set ns [new Simulator]
$ns set cosim_type 0

#### Open files for statistics #####
set namFile [open diffserv.nam w]
set traceFile [open diffserv.tr w]

$ns namtrace-all $namFile
$ns trace-all $traceFile
```

Poi viene costituita la rete oggetto della simulazione. Quindi vengono creati i nodi e i link bidirezionali e vengono definite le specifiche. In questa stessa fase vengono istanziate le applicazioni e gli agenti di rete attaccati sui nodi.
Poi una apposita procedura fa partire la simulazione che consiste nell’avviare progressivamente ogni applicazione associata al relativo nodo, in modo da immettere flussi di traffico diversi uno per volta.

```
set node0 [$ns node]
$node0 color "blue"
set node1 [$ns node]
$node1 color "red"
set node2 [$ns node]
$node2 color "green"
set node3 [$ns node]
$node3 color "magenta"
set node4 [$ns node]
$node4 color "black"
set node5 [$ns node]
$node5 color "purple"
set node6 [$ns node]
$node6 color "blue"
set node7 [$ns node]
$node7 color "red"
set node8 [$ns node]
$node8 color "green"
set node9 [$ns node]
$node9 color "magenta"
set edge [$ns node]
$edge color "black"

# seguono link le application e gli agent collegati
ai nodi relativi ai diversi casi analizzati
```

Poi abbiamo la procedura finish che termina la simulazione, preoccupandosi di chiudere i file di trace e avviare il network animator.

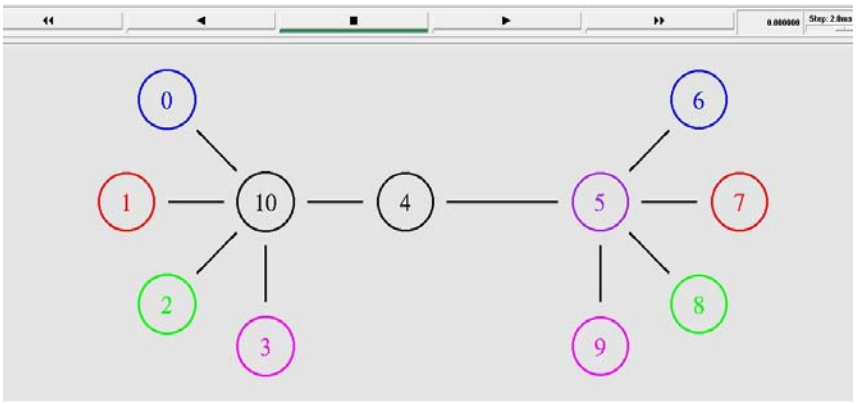
```
proc finish {} {
      global ns namFile traceFile
      $ns flush-trace
      close $namFile
      close $traceFile
      exec nam diffserv.nam &
      exit 0

}
```

Per ultimo ci sono le istruzioni che fanno partire la simulazione registrando gli eventi nello scheduler.

```
$ns at 0.1 "$cbr1 start"
$ns at 0.2 "$cbr3 start"
$ns at 0.05 "$cbr2 start"
$ns at 10.0 "$cbr2 stop"
$ns at 10.0 "$cbr1 stop"
$ns at 10.0 "$cbr3 stop"
$ns at 10.0 "finish"
$ns run
```

4.2 *Lo scenario di simulazione*

La rete di telecomunicazioni su cui abbiamo lavorato ha la topologia mostrata nella figura seguente: ci sono 4 sorgenti di traffico (nodi 0,1,2,3) e 4 destinatari (nodi 6,7,8,9).



La prima sorgente di traffico (nodo 0 -> nodo 6) simula traffico HTTP su TCP; in questo caso abbiamo utilizzato un generatore di traffico “pareto” messo a disposizione da NS (EXPOO_Traffic).

Questa classe fornisce traffico in accordo ad un modello On/Off con tempi di permanenza in ciascuno stato, distribuiti secondo una distribuzione di Pareto. Nello stato On i pacchetti sono trasmessi a rate costante con tempi di interpartenza deterministici e dimensione fissa dei pacchetti, mentre nello stato Off non viene trasmesso nessun pacchetto.

Le variabili configurabili dall’utente sono:

rate: $par($i) set rate_ [expr 64*4]k

packetsize: $par($i) set packetsize_ 210

burst_time: $par($i) set burst_time_ 50ms
idle_time_: $par($i) set idle_time_ 2
indicatore di flusso : $tcp set fid_ 1.

La seconda sorgente di traffico (nodo 1 -> nodo 7) genera traffico VOIP su UDP; in particolare abbiamo utilizzato la classe CBR, come mostrano i comandi, che genera traffico con bit-rate costante con pacchetti di dimensione di 218 byte ad intervalli regolari di 0.02s, indicatore di flusso "2";

La terza sorgente di traffico (nodo 2 -> nodo 8) genera traffico di controllo ICMP su UDP; anche in questo caso è stata utilizzata la classe CBR che genera traffico con bit-rate costante con pacchetti di dimensione di 38 byte ad intervalli regolari di 0.05s, indicatore di flusso 3.

La quarta sorgente di traffico (nodo 3 -> nodo 9) simula traffico P2P su UDP; ancora una volta è stata utilizzata la classe CBR che genera traffico con bit-rate costante con pacchetti di dimensione di 1500 byte ad intervalli regolari di 0.01s, indicatore di flusso 4.

I comandi Tcl che servono a stabilire questa topologia di rete (e che fanno parte degli script di simulazione mostrati interamente nell'Appendice A) sono i seguenti:

```
# CBR 1
# TRAFFICO VOIP
# on Ethernet the max IP packet size is 1500 byte
#-----------------------------------------------------------
set udp1 [new Agent/UDP]
$udp1 set packetSize_ 1500
$udp1 set fid_ 2
set null1 [new Agent/Null]
$ns attach-agent $node1 $udp1
$ns attach-agent $node7 $null1
$ns connect $udp1 $null1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 218
$cbr1 set interval_ 0.02
$cbr1 attach-agent $udp1
#-----------------------------------------------------------
# CBR 2
# TRAFFICO DI CONTROLLO
# on Ethernet the max IP packet size is 1500 byte
#-----------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 3
set null2 [new Agent/Null]
$ns attach-agent $node2 $udp2
$ns attach-agent $node8 $null2
$ns connect $udp2 $null2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 38
$cbr2 set interval_ 0.05
$cbr2 attach-agent $udp2
#-----------------------------------------------------------
#Setup a TCP connection node 0
# TRAFFICO HTTP
#-----------------------------------------------------------
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $node0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $node6 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

# create 16 Pareto On/Off source traffic
  for {set i 0} { $i < 16} {incr i} {
              set par($i) [new Application/Traffic/Pareto]
          $par($i) set rate_ [expr 64*4]k
          $par($i) set packetsize_ 210
          $par($i) set burst_time_ 50ms
          $par($i) set idle_time_ 2
          $par($i) attach-agent $tcp
          $ns at 0.2 "$par($i) start"
  }
#-----------------------------------------------------------
```

```
# CBR 3
# TRAFFICO P2P
# on Ethernet the max IP packet size is 1500 byte
#---------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 4
set null2 [new Agent/Null]

$ns attach-agent $node3 $udp2
$ns attach-agent $node9 $null2
$ns connect $udp2 $null2

set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 1500
$cbr3 set interval_ 0.01
$cbr3 attach-agent $udp2
#---------------------------------------------------------
```

Nella prima simulazione abbiamo utilizzato una politica di scheduling di tipo “droptail” che scarta un pacchetto in arrivo quando questo trova la coda piena. Tale politica se da un lato è facile da implementare, dall'altro ha diversi difetti:

- le sorgenti perdono pacchetti quando ormai la coda è piena determinando un aumento del ritardo end-to-end e del jitter con effetti negativi sulle applicazioni multimediali e in particolare su quelle interattive;
- tra due flussi aventi lo stesso bitrate ma lunghezza media del burst differente, viene penalizzato quello con lunghezza media del burst più alta;
- tutti i flussi TCP che mandano dati ad una coda piena subiscono perdita di pacchetti contemporaneamente e reagiscono abbassando il bitrate per poi rialzarlo contemporaneamente; questa sincronizzazione nel comportamento delle sorgenti abbassa l'efficienza nell'utilizzazione del *bottleneck*.

```
# bottleneck link
$ns simplex-link $node4 $node5 1.2Mb 20ms DropTail
$ns simplex-link-op $node4 $node5 orient right
$ns simplex-link $node5 $node4 1.2Mb 20ms DropTail
$ns duplex-link-op $node4 $node5 queuePos 0.5
$ns queue-limit $node4 $node5 25
```

Per le altre simulazioni RR, WRR e Priority, occorre specificare come l'edge router deve marcare i pacchetti, tramite il comando addPolicyEntry.
In particolare, col comando:

```
set qe2 [[$ns link $edge $node2] queue]
$qe2 addPolicyEntry [$node0 id] [$node4 id] Null 11
$qe2 addPolicerEntry Null 11
```

Si dice di marcare tutti i pacchetti diretti dal nodo 0 al nodo 4 con l'etichetta "11".

Per associare i pacchetti marcati con una certa etichetta ad una certa coda si usa il comando:

```
set qe2 [[$ns link $edge $node2] queue]
$qe2 addPHBEntry $etichetta $queueNum $virtualQueueNum
```

Per default le code vengono servite in Round Robin (turno circolare). Se si vuole servire una coda con priorità maggiore rispetto ad un'altra occorre usare il metodo Weighted Round Robin (WRR) che assegna pesi diversi alle varie code, come si vede nel frammento di codice seguente, dove la coda 0 (HTTP) verrà servita il 20% del tempo, quella 1 (VoIP) il 30%, quella 2 il 40% (ICMP) e la coda 3 (P2P) il restante 10%:

```
set q45 [[$ns link $node4 $node5] queue]
$q45 set numQueues_ 4
$q45 setNumPrec 2
$q45 setSchedularMode WRR
$q45 addQueueWeights 0 2
$q45 addQueueWeights 1 3
$q45 addQueueWeights 2 4
$q45 addQueueWeights 3 1
```

Un altro metodo che si può utilizzare è il Priority che prioritizza le code dalla più bassa alla più alta e di assegnare ai vari flussi di dati una massima banda, che verrà garantita dal router core.

Nel codice sottostante assegnamo, su un totale di 1.2 Mb di banda disponibile, bande massime alle code pari a 50Kb per la coda 0 (ICMP), 100Kb per la 1 (VoIP), 350Kb per la coda 2 (HTTP), 700Kb per la coda 3 (P2P):

```
set q45 [[$ns link $node4 $node5] queue]
$q45 meanPktSize 400
$q45 set numQueues_ 4
$q45 setNumPrec 2
$q45 setSchedularMode PRI
# riga per il PRI
$q45 addQueueRate 2 350Kb
$q45 addQueueRate 1 100Kb
$q45 addQueueRate 0 50Kb
$q45 addQueueRate 3 700Kb
```

4.3 *Parametri della simulazione*

All'istante "0.05" facciamo partire la sorgente "cbr2"

All'istante "0.1" facciamo partire la sorgente "cbr1"

All'istante "0.2" facciamo partire la sorgente "cbr3" e HTTP

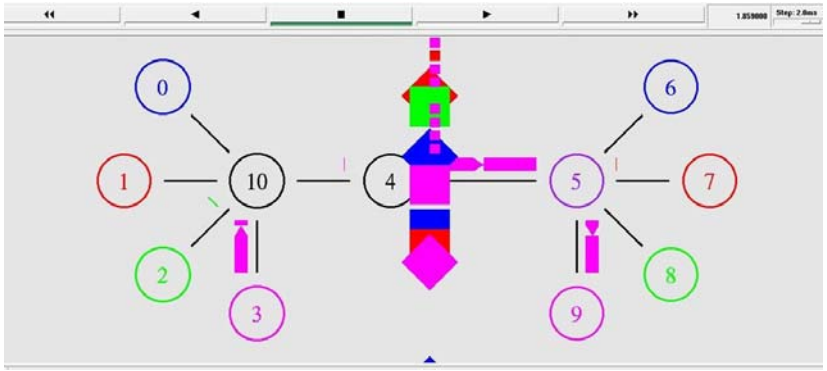
La simulazione dura complessivamente 10 secondi.

Allo start i link sono liberi; man mano che le applicazioni cominciano a generare traffico, i link cominciano a saturarsi; il traffico CBR3, è stato impostato in modo da saturare il link “bottleneck” (nodo 4 -> nodo 5) in modo da verificare cosa succede in caso di carico eccessivo della rete. A questo punto si sono potuti studiare i comportamenti delle varie politiche di scheduling e come i pacchetti vengono scartati in base ai flussi e al riempimento delle code del router core.

Le simulazioni sono state analizzate con il software Tracegraph, che ha permesso di ottenere dei grafici dettagliati a partire dai file di traccia generati da NS.

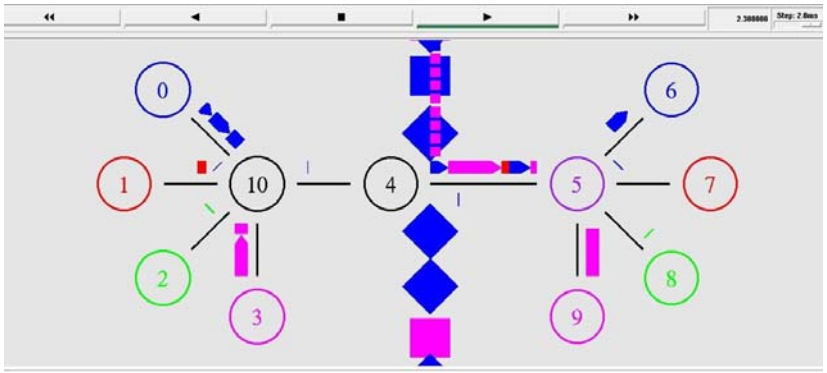
5. Screenshots

5.1 *Politica di accodamento. Confronto DropTail – RandomEarlyDetection*



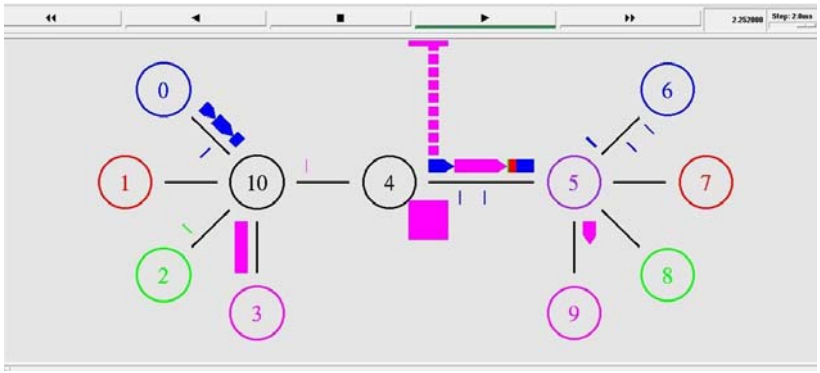
Drop-Tail

Scarta qualsiasi tipo di pacchetto quando la coda è piena.



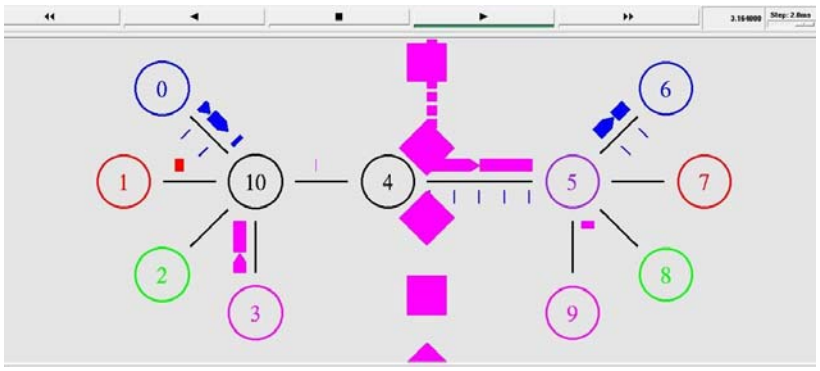
RandomEarlyDetection – RoundRobin

Inoltrando i pacchetti in maniera circolare, scarta quelli che provengono dalle sorgenti che inviano di più.



RandomEarlyDetection – WeightedRoundRobin

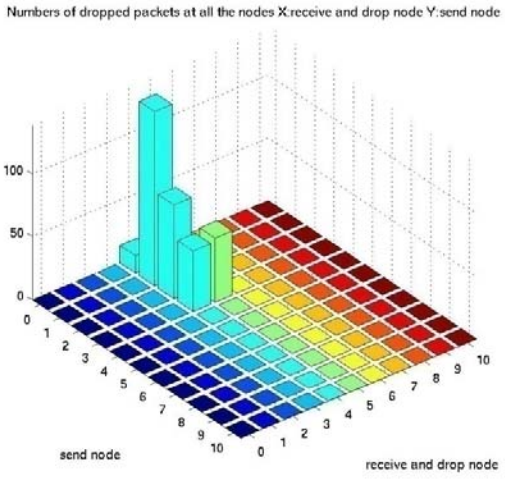
Servendo le code in base alla priorità assegnata, vengono scartati i pacchetti provenienti dalle sorgenti con tempo di utilizzo del canale inferiore.



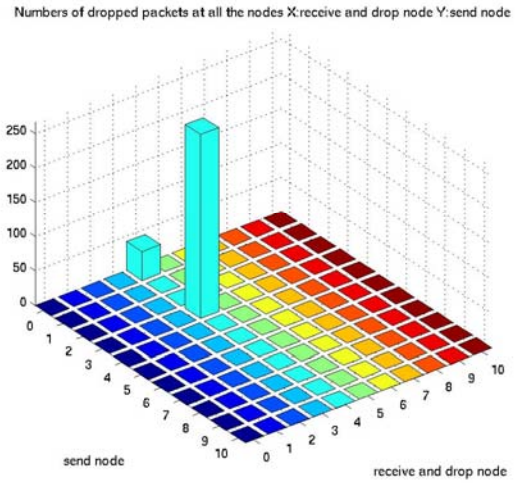
RandomEarlyDetection – Priority

Servendo le code in base all'ordine ed alla larghezza di banda massima, vengono scartati i pacchetti provenienti dalle sorgenti con priorità più bassa.

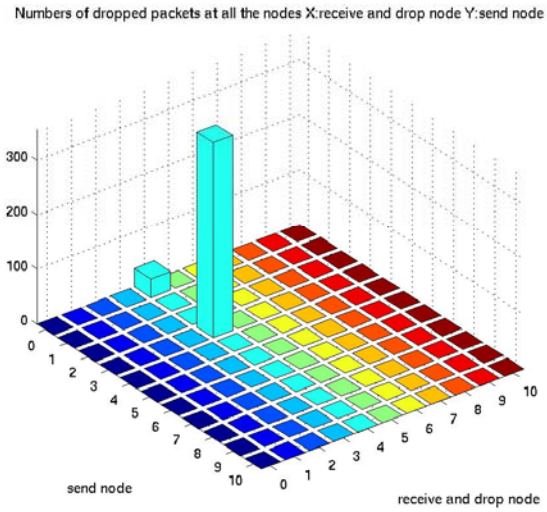
5.2 *Pacchetti scartati* ***DropTail – RoundRobin – WeightedRoundRobin - Priority***



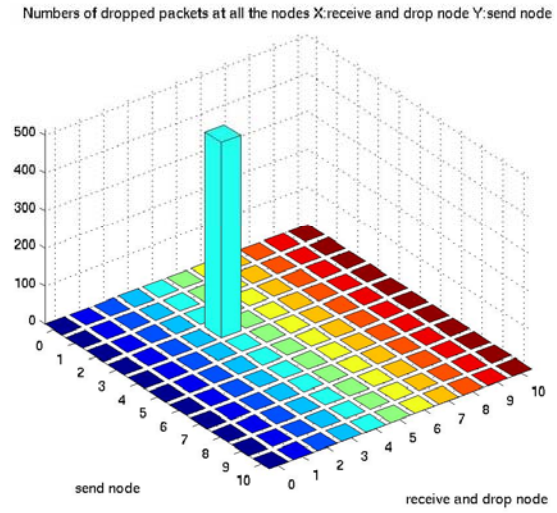
DropTail



RoundRobin

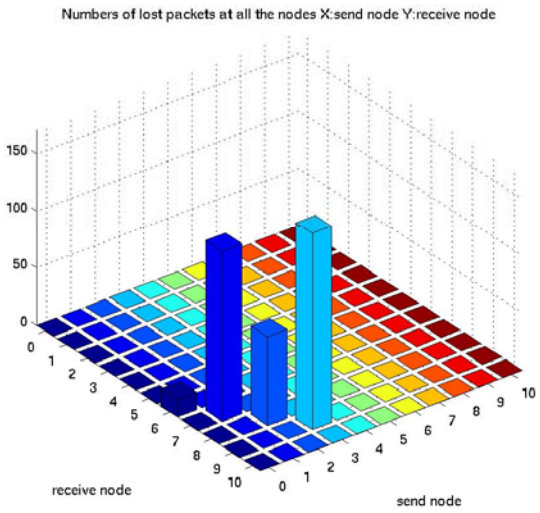


WeightedRoundRobin

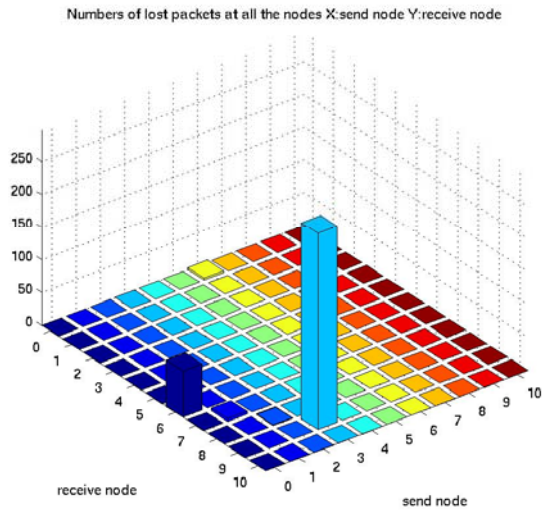


Priority

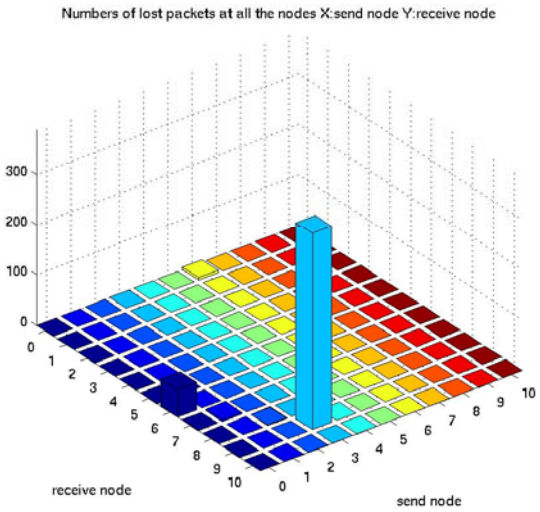
5.3 *Pacchetti persi* ***DropTail - RoundRobin – WeightedRoundRobun – Priority***



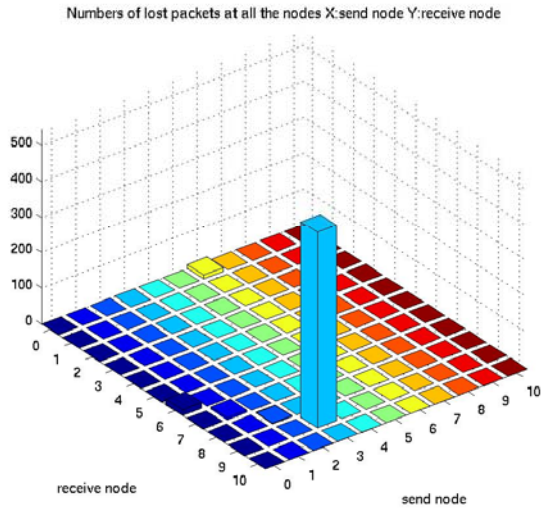
DropTail



RoundRobin

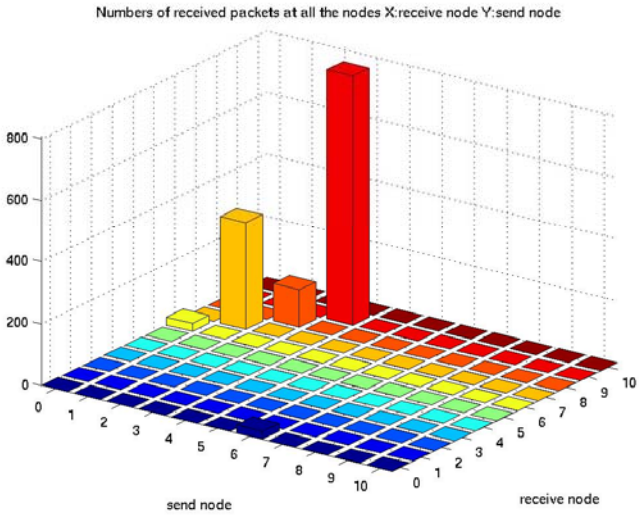


WeightedRoundRobin

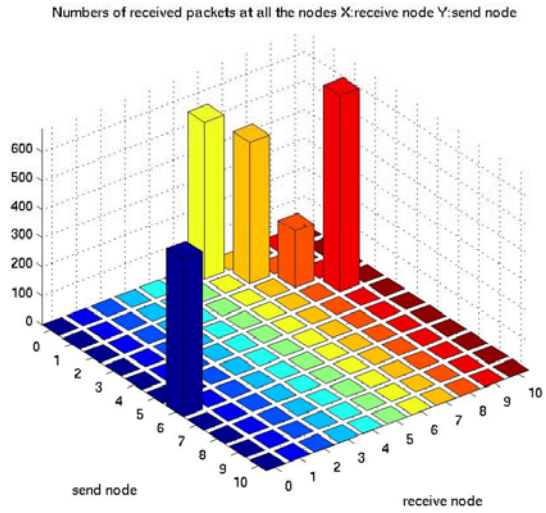


Priority

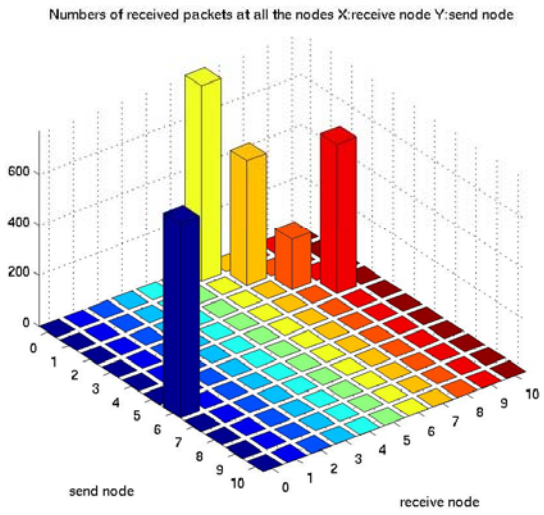
5.4 *Pacchetti ricevuti* ***DropTail - RoundRobin – WeightedRoundRobun – Priority***



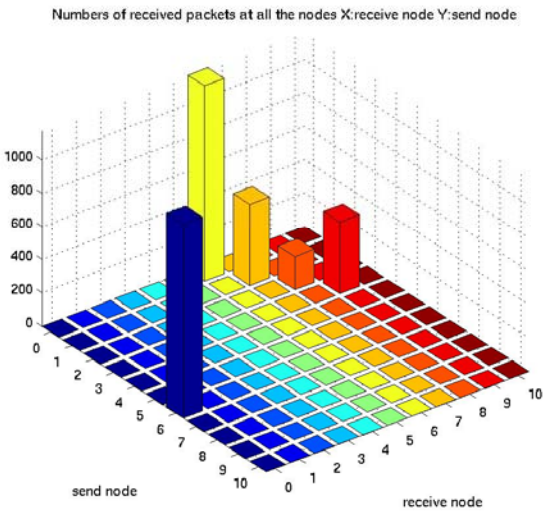
DropTail



RoundRobin

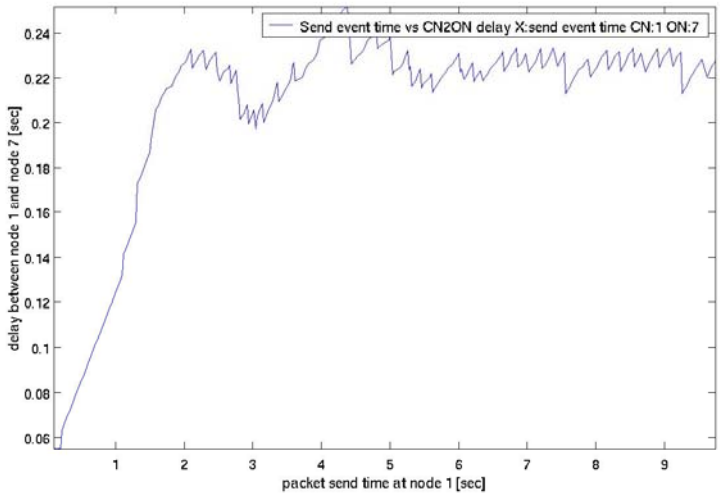


WeightedRoundRobin

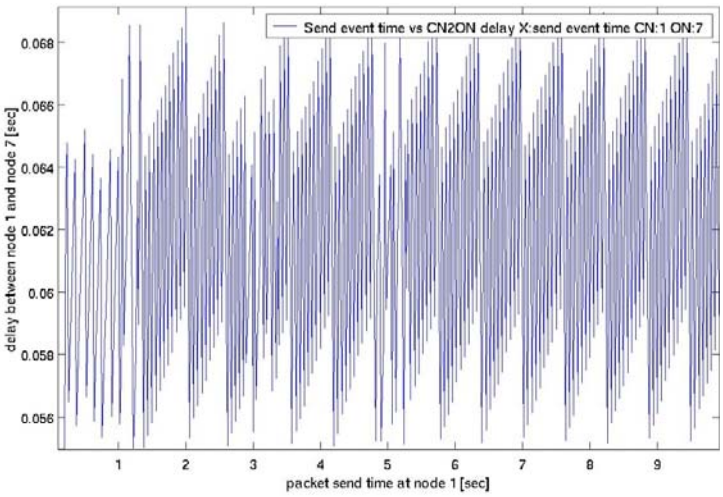


Priority

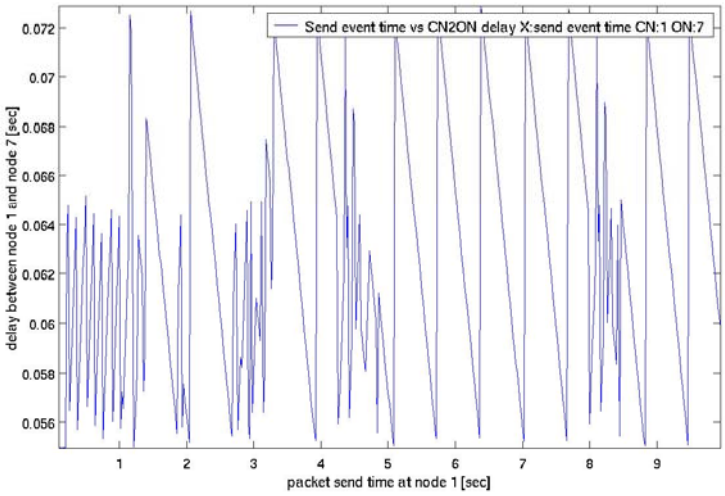
5.5 *Ritardo VoIP DropTail - RoundRobin – WeightedRoundRobun – Priority*



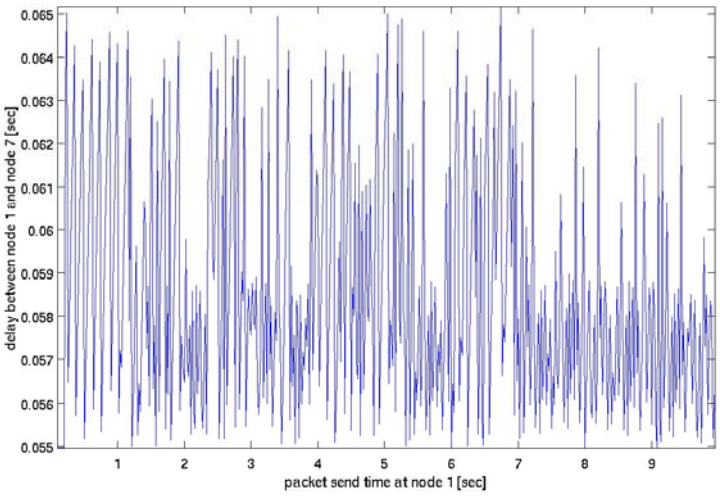
DropTail



RoundRobin

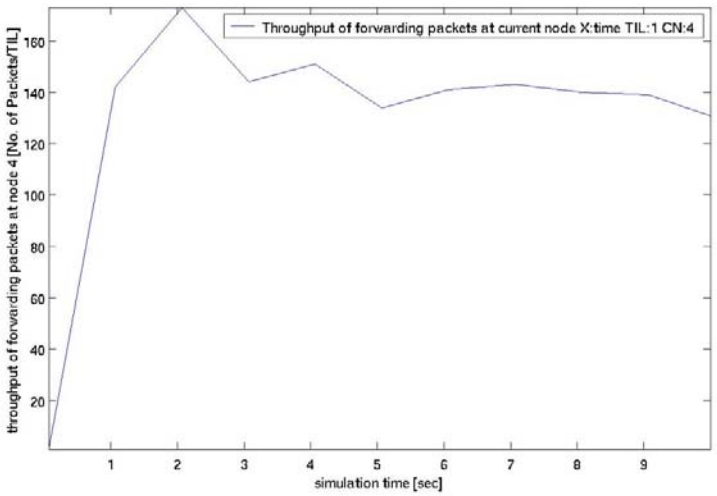


WeightedRoundRobin

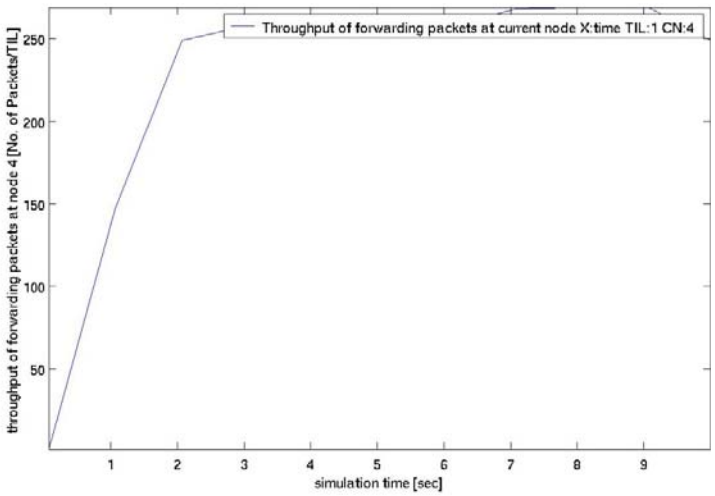


Priority

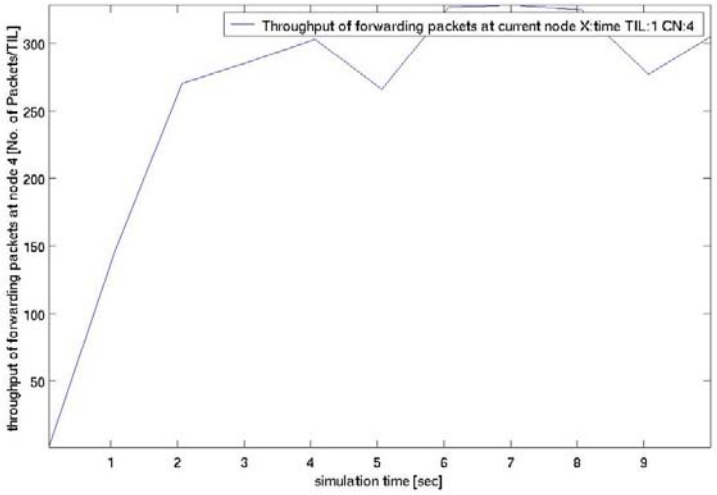
5.6 *Throughput "Node4"* ***DropTail - RoundRobin – WeightedRoundRobun – Priority***



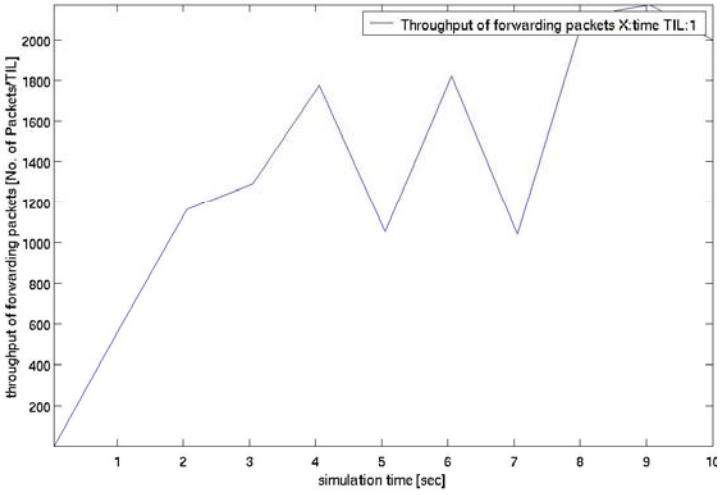
DropTail



RoundRobin

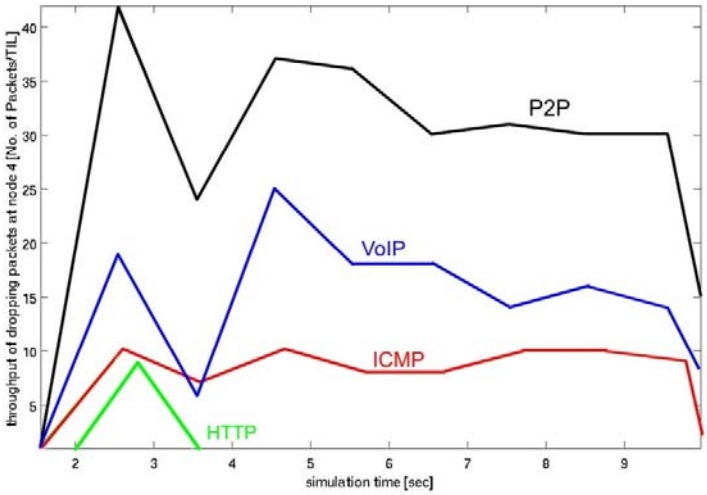


WeightedRoundRobin

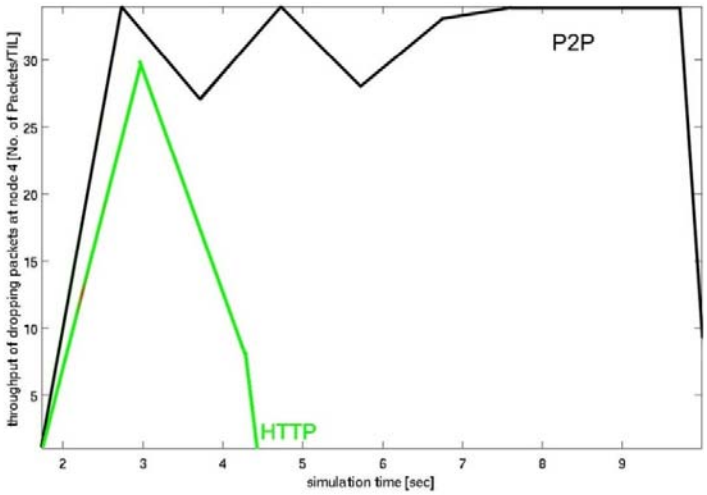


Priority

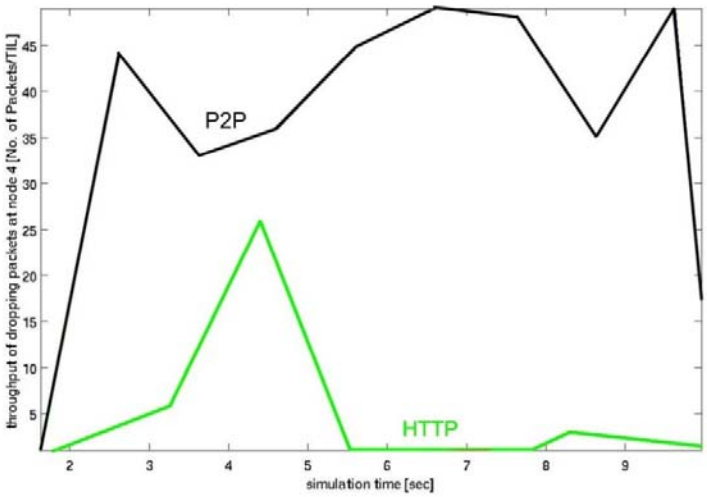
5.7 *Scarto di pacchetti in base al flusso in funzione del tempo*



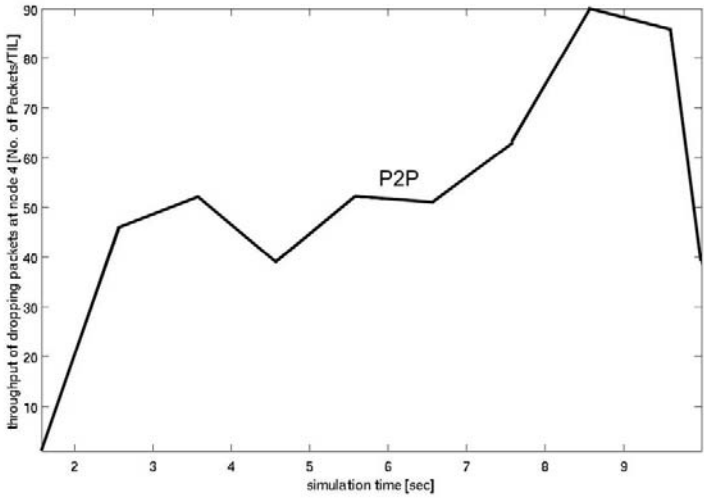
DropTail



RoundRobin



WeightedRoundRobin



Priority

6. Conclusioni

Con questo progetto abbiamo potuto testare le potenzialità dei tool utilizzati (NS e Tracegraph in primis), e come essi possono essere utilizzati efficientemente per ottenere informazioni di simulazione di una topologia particolare di rete.

NS, in particolare, consente di simulare topologie di reti complesse composte da nodi e link di varia natura e di comportamento differente, permettendo di impostare tutti i parametri necessari a vari livelli di profondità dell'architettura OSI.

Tracegraph risulta estremamente utile per quanto riguarda l'analisi dei dati, perché permette di poter visualizzare su grafici tutte le informazioni che possono essere di interesse nel misurare i parametri tipici di una rete di calcolatori (jitter, throughput, latenza, RTT, ecc.), senza la necessità di dover aggiungere ulteriore codice di scripting, come invece sarebbe necessario utilizzando Xgraph o Gnuplot.

Per quanto concerne le simulazioni realizzate, abbiamo constatato che per il traffico VoIP l'utilizzo della qualità del servizio nelle politiche di scheduling dei router permette di ottenere dei grandi vantaggi, tra i quali il disaccoppiamento tra la latenza dei pacchetti VoIP e il livello di congestione della rete.

In particolare la DropTail non è adatta perché, come mostrato dai grafici precedenti, c'è un'alta quantità di pacchetti VoIP scartati dal router quando la coda è piena.

La politica Round Robin migliora la situazione perché garantisce un livello minimo di servizio a tutti i pacchetti, ma la latenza dei pacchetti VoIP rimane comunque alta e variabile in base al traffico di rete.

La Weighted Round Robin è adatta nel caso si voglia massimizzare il throughput della rete, poiché assicura un livello di servizio sufficiente al VoIP grazie al tempo di servizio impostato, a scapito però della latenza, che sebbene potrebbe essere

sufficiente per una buona comunicazione audio, non può arrivare ai livelli della Priority.

La Priority, infine, è paragonabile alla WRR con la differenza che riesce a garantire una banda minima ai vari flussi, cosa che garantisce una bassa latenza ai pacchetti VoIP.

In definitiva, le politiche migliori risultano la WRR e la PRI, che possono essere utilizzare in base alle esigenze dell'utente: se si preferisce maggiore throughtput si adotterà la prima, se invece si vuole assicurare una comunicazione VoIP ottimale si dovrà utilizzare la seconda.

7. Appendice A

7.1 *Codice 1° Simulazione*

```
set ns [new Simulator]
$ns set cosim_type 0

#### Open files for statistics #####
set namFile [open diffserv.nam w]
set traceFile [open diffserv.tr w]
$ns namtrace-all $namFile
$ns trace-all $traceFile


proc finish {} {
      #global ns namFile qsize qbw qlost traceFile
      global namFile traceFile
      #$ns flush-trace
      close $namFile
      close $traceFile
      exec nam diffserv.nam &
      exit 0

}

set node0 [$ns node]
$node0 color "blue"
set node1 [$ns node]
$node1 color "red"
set node2 [$ns node]
$node2 color "green"
set node3 [$ns node]
$node3 color "magenta"
set node4 [$ns node]
$node4 color "black"
set node5 [$ns node]
$node5 color "purple"
set node6 [$ns node]
$node6 color "blue"
set node7 [$ns node]
$node7 color "red"
set node8 [$ns node]
$node8 color "green"
set node9 [$ns node]
$node9 color "magenta"
set edge [$ns node]
$edge color "black"


$ns duplex-link $node0 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node0 $edge orient right-down
#$ns duplex-link-op $node0 $edge queuePos 0.5

$ns duplex-link $node1 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node1 $edge orient right
```

```
#$ns duplex-link-op $node1 $edge queuePos 0.5

$ns duplex-link $node2 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node2 $edge orient right-up

$ns duplex-link $node3 $edge 1.2Mb 10ms DropTail
$ns duplex-link-op $node3 $edge orient up
#$ns duplex-link-op $node3 $edge queuePos 0.5

$ns simplex-link $edge $node4 100Mb 10ms DropTail
$ns simplex-link-op $edge $node4 orient right
$ns simplex-link $node4 $edge 100Mb 10ms DropTail


# bottleneck link
$ns simplex-link $node4 $node5 1.2Mb 20ms DropTail
$ns simplex-link-op $node4 $node5 orient right
$ns simplex-link $node5 $node4 1.2Mb 20ms DropTail
$ns simplex-link-op $node4 $node5 queuePos 0.5
$ns queue-limit $node4 $node5 25

$ns duplex-link $node5 $node6 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node6 orient right-up
$ns duplex-link $node5 $node7 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node7 orient right
$ns duplex-link $node5 $node8 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node8 orient right-down
$ns duplex-link $node5 $node9 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node9 orient down



# CBR 1
# TRAFFICO VOIP
# on Ethernet the max IP packet size is 1500 byte
#-------------------------------------------------------------
set udp1 [new Agent/UDP]
$udp1 set packetSize_ 1500
$udp1 set fid_ 2
set null1 [new Agent/Null]

$ns attach-agent $node1 $udp1
$ns attach-agent $node7 $null1
$ns connect $udp1 $null1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 218
$cbr1 set interval_ 0.02
$cbr1 attach-agent $udp1
#-------------------------------------------------------------

# CBR 2
# TRAFFICO DI CONTROLLO
# on Ethernet the max IP packet size is 1500 byte
#-------------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 3
set null2 [new Agent/Null]
```

```
$ns attach-agent $node2 $udp2
$ns attach-agent $node8 $null2
$ns connect $udp2 $null2

set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 38
$cbr2 set interval_ 0.05
$cbr2 attach-agent $udp2
#---------------------------------------------------------


#Setup a TCP connection node 0
# TRAFFICO HTTP
#---------------------------------------------------------
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $node0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $node6 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

# create 16 Pareto On/Off source traffic
  for {set i 0} { $i < 16} {incr i} {
          set par($i) [new Application/Traffic/Pareto]
          $par($i) set rate_ [expr 64*4]k
          $par($i) set packetsize_ 210
          $par($i) set burst_time_ 50ms
          $par($i) set idle_time_ 2
          $par($i) attach-agent $tcp
          $ns at 0.2 "$par($i) start"
  }
#---------------------------------------------------------


# CBR 3
# TRAFFICO P2P
# on Ethernet the max IP packet size is 1500 byte
#---------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 4
set null2 [new Agent/Null]

$ns attach-agent $node3 $udp2
$ns attach-agent $node9 $null2
$ns connect $udp2 $null2

set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 1500
$cbr3 set interval_ 0.01
$cbr3 attach-agent $udp2
#---------------------------------------------------------


$ns color 1 blue
```

```
$ns color 2 red
$ns color 3 green
$ns color 4 magenta

$ns at 0.1 "$cbr1 start"
$ns at 0.2 "$cbr3 start"
$ns at 0.05 "$cbr2 start"
$ns at 10.0 "$cbr2 stop"
$ns at 10.0 "$cbr1 stop"
$ns at 10.0 "$cbr3 stop"
$ns at 10.0 "finish"
$ns run
```

7.2 *Codice 2° Simulazione*

```
set ns [new Simulator]
$ns set cosim_type 0

#### Open files for statistics #####
set namFile [open diffserv.nam w]
set traceFile [open diffserv.tr w]
$ns namtrace-all $namFile
$ns trace-all $traceFile


proc finish {} {
      global ns namFile qsize qbw qlost traceFile
      $ns flush-trace
      close $namFile
      close $traceFile
      exec nam diffserv.nam &
      exit 0

}

set node0 [$ns node]
$node0 color "blue"
set node1 [$ns node]
$node1 color "red"
set node2 [$ns node]
$node2 color "green"
set node3 [$ns node]
$node3 color "magenta"
set node4 [$ns node]
$node4 color "black"
set node5 [$ns node]
$node5 color "purple"
set node6 [$ns node]
$node6 color "blue"
set node7 [$ns node]
$node7 color "red"
set node8 [$ns node]
$node8 color "green"
set node9 [$ns node]
$node9 color "magenta"
set edge [$ns node]
$edge color "black"
```

```
$ns duplex-link $node0 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node0 $edge orient right-down
#$ns duplex-link-op $node0 $edge queuePos 0.5

$ns duplex-link $node1 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node1 $edge orient right
#$ns duplex-link-op $node1 $edge queuePos 0.5

$ns duplex-link $node2 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node2 $edge orient right-up

$ns duplex-link $node3 $edge 1.2Mb 10ms DropTail
$ns duplex-link-op $node3 $edge orient up
#$ns duplex-link-op $node3 $edge queuePos 0.5

$ns simplex-link $edge $node4 100Mb 10ms dsRED/edge
$ns simplex-link-op $edge $node4 orient right
$ns simplex-link $node4 $edge 100Mb 10ms dsRED/core


# bottleneck link
$ns simplex-link $node4 $node5 1.2Mb 20ms dsRED/core
$ns simplex-link-op $node4 $node5 orient right
$ns simplex-link $node5 $node4 1.2Mb 20ms dsRED/edge
$ns duplex-link-op $node4 $node5 queuePos 0.5
$ns queue-limit $node4 $node5 25

$ns duplex-link $node5 $node6 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node6 orient right-up
$ns duplex-link $node5 $node7 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node7 orient right
$ns duplex-link $node5 $node8 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node8 orient right-down
$ns duplex-link $node5 $node9 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node9 orient down


set qe2 [[$ns link $edge $node4] queue]
$qe2 meanPktSize 400
$qe2 set numQueues_ 4
$qe2 setNumPrec 2
$qe2 addPolicyEntry [$node0 id] [$node6 id] Null 00
$qe2 addPolicyEntry [$node1 id] [$node7 id] Null 11
$qe2 addPolicyEntry [$node2 id] [$node8 id] Null 22
$qe2 addPolicyEntry [$node3 id] [$node9 id] Null 33
$qe2 addPolicyEntry [$node6 id] [$node0 id] Null 44
$qe2 addPolicerEntry Null 44
$qe2 addPolicerEntry Null 33
$qe2 addPolicerEntry Null 22
$qe2 addPolicerEntry Null 11
$qe2 addPolicerEntry Null 00
$qe2 addPHBEntry 11 1 0
$qe2 addPHBEntry 00 0 0
$qe2 addPHBEntry 22 2 0
$qe2 addPHBEntry 33 3 0
$qe2 addPHBEntry 44 0 1
$qe2 configQ 0 0 20 40 0.02
```

```
$qe2 configQ 2 0 20 40 0.02
$qe2 configQ 1 0 10 20 0.10
$qe2 configQ 3 0 10 20 0.10

set q23 [[$ns link $node4 $node5] queue]
$q23 meanPktSize 400
$q23 set numQueues_ 4
$q23 setNumPrec 2
$q23 addPHBEntry 11 1 0
$q23 addPHBEntry 00 0 0
$q23 addPHBEntry 22 2 0
$q23 addPHBEntry 33 3 0
$q23 addPHBEntry 44 0 1
$q23 configQ 0 0 20 40 0.02
$q23 configQ 2 0 20 40 0.02
$q23 configQ 1 0 10 20 0.10
$q23 configQ 3 0 10 20 0.10

set q32 [[$ns link $node5 $node4] queue]
$q32 meanPktSize 400
$q32 set numQueues_ 4
$q32 setNumPrec 2
$q32 addPolicyEntry [$node0 id] [$node6 id] Null 00
$q32 addPolicyEntry [$node1 id] [$node7 id] Null 11
$q32 addPolicyEntry [$node2 id] [$node8 id] Null 22
$q32 addPolicyEntry [$node3 id] [$node9 id] Null 33
$q32 addPolicyEntry [$node6 id] [$node0 id] Null 44
$q32 addPolicerEntry Null 44
$q32 addPolicerEntry Null 33
$q32 addPolicerEntry Null 22
$q32 addPolicerEntry Null 11
$q32 addPolicerEntry Null 00
$q32 addPHBEntry 11 1 0
$q32 addPHBEntry 00 0 0
$q32 addPHBEntry 22 2 0
$q32 addPHBEntry 33 3 0
$q32 addPHBEntry 44 0 1
$q32 configQ 0 0 20 40 0.02
$q32 configQ 1 0 10 20 0.10
$q32 configQ 2 0 20 40 0.02
$q32 configQ 3 0 20 40 0.02

set q2e [[$ns link $node4 $edge] queue]
$q2e meanPktSize 400
$q2e set numQueues_ 4
$q2e setNumPrec 2
$q2e addPHBEntry 11 1 0
$q2e addPHBEntry 00 0 0
$q2e addPHBEntry 22 2 0
$q2e addPHBEntry 33 3 0
$q2e addPHBEntry 44 0 1
$q2e configQ 0 0 20 40 0.02
$q2e configQ 1 0 10 20 0.10
$q2e configQ 2 0 20 40 0.02
$q2e configQ 3 0 20 40 0.02

# CBR 1
# TRAFFICO VOIP
# on Ethernet the max IP packet size is 1500 byte
```

```
#-----------------------------------------------------------
set udp1 [new Agent/UDP]
$udp1 set packetSize_ 1500
$udp1 set fid_ 2
set null1 [new Agent/Null]

$ns attach-agent $node1 $udp1
$ns attach-agent $node7 $null1
$ns connect $udp1 $null1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 218
$cbr1 set interval_ 0.02
$cbr1 attach-agent $udp1
#-----------------------------------------------------------

# CBR 2
# TRAFFICO DI CONTROLLO
# on Ethernet the max IP packet size is 1500 byte
#-----------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 3
set null2 [new Agent/Null]

$ns attach-agent $node2 $udp2
$ns attach-agent $node8 $null2
$ns connect $udp2 $null2

set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 38
$cbr2 set interval_ 0.05
$cbr2 attach-agent $udp2
#-----------------------------------------------------------


#Setup a TCP connection node 0
# TRAFFICO HTTP
#-----------------------------------------------------------
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $node0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $node6 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

# create 16 Pareto On/Off source traffic
  for {set i 0} { $i < 16} {incr i} {
          set par($i) [new Application/Traffic/Pareto]
          $par($i) set rate_ [expr 64*4]k
          $par($i) set packetsize_ 210
          $par($i) set burst_time_ 50ms
          $par($i) set idle_time_ 2
          $par($i) attach-agent $tcp
          $ns at 0.2 "$par($i) start"
  }
```

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


# CBR 3
# TRAFFICO P2P
# on Ethernet the max IP packet size is 1500 byte
#-----------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 4
set null2 [new Agent/Null]

$ns attach-agent $node3 $udp2
$ns attach-agent $node9 $null2
$ns connect $udp2 $null2

set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 1500
$cbr3 set interval_ 0.01
$cbr3 attach-agent $udp2
#-----------------------------------------------------------


$ns color 1 blue
$ns color 2 red
$ns color 3 green
$ns color 4 magenta


$ns at 0.1 "$cbr1 start"
$ns at 0.2 "$cbr3 start"
$ns at 0.05 "$cbr2 start"
$ns at 10.0 "$cbr2 stop"
$ns at 10.0 "$cbr1 stop"
$ns at 10.0 "$cbr3 stop"
$ns at 10.0 "finish"
$ns run
```

7.3 *Codice 3° Simulazione*

```
set ns [new Simulator]
$ns set cosim_type 0

#### Open files for statistics #####
set namFile [open diffserv.nam w]
set traceFile [open diffserv.tr w]
$ns namtrace-all $namFile
$ns trace-all $traceFile



proc finish {} {
```

```
        global ns namFile qsize qbw qlost traceFile
        $ns flush-trace
        close $namFile
        close $traceFile
        exec nam diffserv.nam &
        exit 0

}

set node0 [$ns node]
$node0 color "blue"
set node1 [$ns node]
$node1 color "red"
set node2 [$ns node]
$node2 color "green"
set node3 [$ns node]
$node3 color "magenta"
set node4 [$ns node]
$node4 color "black"
set node5 [$ns node]
$node5 color "purple"
set node6 [$ns node]
$node6 color "blue"
set node7 [$ns node]
$node7 color "red"
set node8 [$ns node]
$node8 color "green"
set node9 [$ns node]
$node9 color "magenta"
set edge [$ns node]
$edge color "black"

$ns duplex-link $node0 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node0 $edge orient right-down
#$ns duplex-link-op $node0 $edge queuePos 0.5

$ns duplex-link $node1 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node1 $edge orient right
#$ns duplex-link-op $node1 $edge queuePos 0.5

$ns duplex-link $node2 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node2 $edge orient right-up

$ns duplex-link $node3 $edge 1.2Mb 10ms DropTail
$ns duplex-link-op $node3 $edge orient up
$ns duplex-link-op $node3 $edge queuePos 0.5

$ns simplex-link $edge $node4 100Mb 10ms dsRED/edge
$ns simplex-link-op $edge $node4 orient right
$ns simplex-link $node4 $edge 100Mb 10ms dsRED/core


# bottleneck link
$ns simplex-link $node4 $node5 1.2Mb 20ms dsRED/core
$ns simplex-link-op $node4 $node5 orient right
$ns simplex-link $node5 $node4 1.2Mb 20ms dsRED/edge
$ns duplex-link-op $node4 $node5 queuePos 0.5
$ns queue-limit $node4 $node5 25
```

```
$ns duplex-link $node5 $node6 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node6 orient right-up
$ns duplex-link $node5 $node7 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node7 orient right
$ns duplex-link $node5 $node8 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node8 orient right-down
$ns duplex-link $node5 $node9 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node9 orient down


set qe2 [[$ns link $edge $node4] queue]
$qe2 meanPktSize 400
$qe2 set numQueues_ 4
$qe2 setNumPrec 2
$qe2 addPolicyEntry [$node0 id] [$node6 id] Null 00
$qe2 addPolicyEntry [$node1 id] [$node7 id] Null 11
$qe2 addPolicyEntry [$node2 id] [$node8 id] Null 22
$qe2 addPolicyEntry [$node3 id] [$node9 id] Null 33
$qe2 addPolicyEntry [$node6 id] [$node0 id] Null 44
$qe2 addPolicerEntry Null 44
$qe2 addPolicerEntry Null 33
$qe2 addPolicerEntry Null 22
$qe2 addPolicerEntry Null 11
$qe2 addPolicerEntry Null 00
$qe2 addPHBEntry 11 1 0
$qe2 addPHBEntry 22 0 0
$qe2 addPHBEntry 00 2 0
$qe2 addPHBEntry 33 3 0
$qe2 addPHBEntry 44 0 1
$qe2 configQ 0 0 20 40 0.02
$qe2 configQ 2 0 20 40 0.02
$qe2 configQ 1 0 10 20 0.10
$qe2 configQ 3 0 10 20 0.10

set q23 [[$ns link $node4 $node5] queue]
$q23 meanPktSize 400
$q23 set numQueues_ 4
$q23 setNumPrec 2
$q23 setSchedularMode WRR
$q23 addQueueWeights 0 2
$q23 addQueueWeights 1 3
$q23 addQueueWeights 2 4
$q23 addQueueWeights 3 1
$q23 addPHBEntry 11 1 0
$q23 addPHBEntry 22 0 0
$q23 addPHBEntry 00 2 0
$q23 addPHBEntry 33 3 0
$q23 addPHBEntry 44 0 1
$q23 configQ 0 0 20 40 0.02
$q23 configQ 2 0 20 40 0.02
$q23 configQ 1 0 10 20 0.10
$q23 configQ 3 0 10 20 0.10

set q32 [[$ns link $node5 $node4] queue]
$q32 meanPktSize 400
$q32 set numQueues_ 4
$q32 setNumPrec 2
$q32 addPolicyEntry [$node0 id] [$node6 id] Null 00
$q32 addPolicyEntry [$node1 id] [$node7 id] Null 11
```

```
$q32 addPolicyEntry [$node2 id] [$node8 id] Null 22
$q32 addPolicyEntry [$node3 id] [$node9 id] Null 33
$q32 addPolicyEntry [$node6 id] [$node0 id] Null 44
$q32 addPolicerEntry Null 44
$q32 addPolicerEntry Null 33
$q32 addPolicerEntry Null 22
$q32 addPolicerEntry Null 11
$q32 addPolicerEntry Null 00
$q32 addPHBEntry 11 1 0
$q32 addPHBEntry 22 0 0
$q32 addPHBEntry 00 2 0
$q32 addPHBEntry 33 3 0
$q32 addPHBEntry 44 0 1
$q32 configQ 0 0 20 40 0.02
$q32 configQ 1 0 10 20 0.10
$q32 configQ 2 0 20 40 0.02
$q32 configQ 3 0 20 40 0.02

set q2e [[$ns link $node4 $edge] queue]
$q2e meanPktSize 400
$q2e set numQueues_ 4
$q2e setNumPrec 2
$q2e addPHBEntry 11 1 0
$q2e addPHBEntry 22 0 0
$q2e addPHBEntry 00 2 0
$q2e addPHBEntry 33 3 0
$q2e addPHBEntry 44 0 1
$q2e configQ 0 0 20 40 0.02
$q2e configQ 1 0 10 20 0.10
$q2e configQ 2 0 20 40 0.02
$q2e configQ 3 0 20 40 0.02

# CBR 1
# TRAFFICO VOIP
# on Ethernet the max IP packet size is 1500 byte
#-------------------------------------------------------------
set udp1 [new Agent/UDP]
$udp1 set packetSize_ 1500
$udp1 set fid_ 2
set null1 [new Agent/Null]

$ns attach-agent $node1 $udp1
$ns attach-agent $node7 $null1
$ns connect $udp1 $null1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 218
$cbr1 set interval_ 0.02
$cbr1 attach-agent $udp1
#-------------------------------------------------------------

# CBR 2
# TRAFFICO DI CONTROLLO
# on Ethernet the max IP packet size is 1500 byte
#-------------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 3
set null2 [new Agent/Null]
```

```
$ns attach-agent $node2 $udp2
$ns attach-agent $node8 $null2
$ns connect $udp2 $null2

set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 38
$cbr2 set interval_ 0.05
$cbr2 attach-agent $udp2
#-------------------------------------------------------


#Setup a TCP connection node 0
# TRAFFICO HTTP
#-------------------------------------------------------
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $node0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $node6 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

# create 16 Pareto On/Off source traffic
  for {set i 0} { $i < 16} {incr i} {
          set par($i) [new Application/Traffic/Pareto]
          $par($i) set rate_ [expr 64*4]k
          $par($i) set packetsize_ 210
          $par($i) set burst_time_ 50ms
          $par($i) set idle_time_ 2
          $par($i) attach-agent $tcp
          $ns at 0.2 "$par($i) start"
  }
#-------------------------------------------------------


# CBR 3
# TRAFFICO P2P
# on Ethernet the max IP packet size is 1500 byte
#-------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 4
set null2 [new Agent/Null]

$ns attach-agent $node3 $udp2
$ns attach-agent $node9 $null2
$ns connect $udp2 $null2

set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 1500
$cbr3 set interval_ 0.01
$cbr3 attach-agent $udp2
#-------------------------------------------------------


$ns color 1 blue
```

```
$ns color 2 red
$ns color 3 green
$ns color 4 magenta

$ns at 0.1 "$cbr1 start"
$ns at 0.2 "$cbr3 start"
$ns at 0.05 "$cbr2 start"
$ns at 10.0 "$cbr2 stop"
$ns at 10.0 "$cbr1 stop"
$ns at 10.0 "$cbr3 stop"
$ns at 10.0 "finish"
$ns run
```

7.4 *Codice 4° Simulazione*

```
set ns [new Simulator]
$ns set cosim_type 0

#### Open files for statistics #####
set namFile [open diffserv.nam w]
set traceFile [open diffserv.tr w]
$ns namtrace-all $namFile
$ns trace-all $traceFile


proc finish {} {
      global ns namFile qsize qbw qlost traceFile
      $ns flush-trace
      close $namFile
      close $traceFile
      exec nam diffserv.nam &
      exit 0

}

set node0 [$ns node]
$node0 color "blue"
set node1 [$ns node]
$node1 color "red"
set node2 [$ns node]
$node2 color "green"
set node3 [$ns node]
$node3 color "magenta"
set node4 [$ns node]
$node4 color "black"
set node5 [$ns node]
$node5 color "purple"
set node6 [$ns node]
$node6 color "blue"
set node7 [$ns node]
$node7 color "red"
set node8 [$ns node]
$node8 color "green"
```

```
set node9 [$ns node]
$node9 color "magenta"
set edge [$ns node]
$edge color "black"

$ns duplex-link $node0 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node0 $edge orient right-down
#$ns duplex-link-op $node0 $edge queuePos 0.5

$ns duplex-link $node1 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node1 $edge orient right
#$ns duplex-link-op $node1 $edge queuePos 0.5

$ns duplex-link $node2 $edge 1Mb 10ms DropTail
$ns duplex-link-op $node2 $edge orient right-up

$ns duplex-link $node3 $edge 1.2Mb 10ms DropTail
$ns duplex-link-op $node3 $edge orient up
$ns duplex-link-op $node3 $edge queuePos 0.5

$ns simplex-link $edge $node4 100Mb 10ms dsRED/edge
$ns simplex-link-op $edge $node4 orient right
$ns simplex-link $node4 $edge 100Mb 10ms dsRED/core


# bottleneck link
$ns simplex-link $node4 $node5 1.2Mb 20ms dsRED/core
$ns simplex-link-op $node4 $node5 orient right
$ns simplex-link $node5 $node4 1.2Mb 20ms dsRED/edge
$ns duplex-link-op $node4 $node5 queuePos 0.5
$ns queue-limit $node4 $node5 25

$ns duplex-link $node5 $node6 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node6 orient right-up
$ns duplex-link $node5 $node7 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node7 orient right
$ns duplex-link $node5 $node8 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node8 orient right-down
$ns duplex-link $node5 $node9 1Mb 10ms DropTail
$ns duplex-link-op $node5 $node9 orient down


set qe2 [[$ns link $edge $node4] queue]
$qe2 meanPktSize 400
$qe2 set numQueues_ 4
$qe2 setNumPrec 2
$qe2 addPolicyEntry [$node0 id] [$node6 id] Null 00
$qe2 addPolicyEntry [$node1 id] [$node7 id] Null 11
$qe2 addPolicyEntry [$node2 id] [$node8 id] Null 22
$qe2 addPolicyEntry [$node3 id] [$node9 id] Null 33
$qe2 addPolicyEntry [$node6 id] [$node0 id] Null 44
$qe2 addPolicerEntry Null 44
$qe2 addPolicerEntry Null 33
$qe2 addPolicerEntry Null 22
$qe2 addPolicerEntry Null 11
$qe2 addPolicerEntry Null 00
$qe2 addPHBEntry 11 1 0
$qe2 addPHBEntry 00 0 0
$qe2 addPHBEntry 22 2 0
```

```
$qe2 addPHBEntry 33 3 0
$qe2 addPHBEntry 44 0 1
$qe2 configQ 0 0 20 40 0.02
$qe2 configQ 2 0 20 40 0.02
$qe2 configQ 1 0 10 20 0.10
$qe2 configQ 3 0 10 20 0.10

set q23 [[$ns link $node4 $node5] queue]
$q23 meanPktSize 400
$q23 set numQueues_ 4
$q23 setNumPrec 2
$q23 setSchedularMode PRI
# riga per il PRI
$q23 addQueueRate 0 350Kb
$q23 addQueueRate 1 100Kb
$q23 addQueueRate 2 50Kb
$q23 addQueueRate 3 700Kb
$q23 addPHBEntry 11 1 0
$q23 addPHBEntry 00 0 0
$q23 addPHBEntry 22 2 0
$q23 addPHBEntry 33 3 0
$q23 addPHBEntry 44 0 1
$q23 configQ 0 0 20 40 0.02
$q23 configQ 2 0 20 40 0.02
$q23 configQ 1 0 10 20 0.10
$q23 configQ 3 0 10 20 0.10

set q32 [[$ns link $node5 $node4] queue]
$q32 meanPktSize 400
$q32 set numQueues_ 4
$q32 setNumPrec 2
$q32 addPolicyEntry [$node0 id] [$node6 id] Null 00
$q32 addPolicyEntry [$node1 id] [$node7 id] Null 11
$q32 addPolicyEntry [$node2 id] [$node8 id] Null 22
$q32 addPolicyEntry [$node3 id] [$node9 id] Null 33
$q32 addPolicyEntry [$node6 id] [$node0 id] Null 44
$q32 addPolicerEntry Null 44
$q32 addPolicerEntry Null 33
$q32 addPolicerEntry Null 22
$q32 addPolicerEntry Null 11
$q32 addPolicerEntry Null 00
$q32 addPHBEntry 11 1 0
$q32 addPHBEntry 00 0 0
$q32 addPHBEntry 22 2 0
$q32 addPHBEntry 33 3 0
$q32 addPHBEntry 44 0 1
$q32 configQ 0 0 20 40 0.02
$q32 configQ 1 0 10 20 0.10
$q32 configQ 2 0 20 40 0.02
$q32 configQ 3 0 20 40 0.02

set q2e [[$ns link $node4 $edge] queue]
$q2e meanPktSize 400
$q2e set numQueues_ 4
$q2e setNumPrec 2
$q2e addPHBEntry 11 1 0
$q2e addPHBEntry 00 0 0
$q2e addPHBEntry 22 2 0
$q2e addPHBEntry 33 3 0
```

```
$q2e addPHBEntry 44 0 1
$q2e configQ 0 0 20 40 0.02
$q2e configQ 1 0 10 20 0.10
$q2e configQ 2 0 20 40 0.02
$q2e configQ 3 0 20 40 0.02

# CBR 1
# TRAFFICO VOIP
# on Ethernet the max IP packet size is 1500 byte
#-------------------------------------------------------
set udp1 [new Agent/UDP]
$udp1 set packetSize_ 1500
$udp1 set fid_ 2
set null1 [new Agent/Null]

$ns attach-agent $node1 $udp1
$ns attach-agent $node7 $null1
$ns connect $udp1 $null1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 218
$cbr1 set interval_ 0.02
$cbr1 attach-agent $udp1
#-------------------------------------------------------

# CBR 2
# TRAFFICO DI CONTROLLO
# on Ethernet the max IP packet size is 1500 byte
#-------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 3
set null2 [new Agent/Null]

$ns attach-agent $node2 $udp2
$ns attach-agent $node8 $null2
$ns connect $udp2 $null2

set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 38
$cbr2 set interval_ 0.05
$cbr2 attach-agent $udp2
#-------------------------------------------------------


#Setup a TCP connection node 0
# TRAFFICO HTTP
#-------------------------------------------------------
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $node0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $node6 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

# create 16 Pareto On/Off source traffic
```

```
  for {set i 0} { $i < 16} {incr i} {
          set par($i) [new Application/Traffic/Pareto]
          $par($i) set rate_ [expr 64*4]k
          $par($i) set packetsize_ 210
          $par($i) set burst_time_ 50ms
          $par($i) set idle_time_ 2
          $par($i) attach-agent $tcp
          $ns at 0.2 "$par($i) start"
  }
#---------------------------------------------------------


# CBR 3
# TRAFFICO P2P
# on Ethernet the max IP packet size is 1500 byte
#---------------------------------------------------------
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1500
$udp2 set fid_ 4
set null2 [new Agent/Null]

$ns attach-agent $node3 $udp2
$ns attach-agent $node9 $null2
$ns connect $udp2 $null2

set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 1500
$cbr3 set interval_ 0.01
$cbr3 attach-agent $udp2
#---------------------------------------------------------


$ns color 1 blue
$ns color 2 red
$ns color 3 green
$ns color 4 magenta
$ns at 0.1 "$cbr1 start"
$ns at 0.2 "$cbr3 start"
$ns at 0.05 "$cbr2 start"
$ns at 10.0 "$cbr2 stop"
$ns at 10.0 "$cbr1 stop"
$ns at 10.0 "$cbr3 stop"
$ns at 10.0 "finish"
$ns run
```