Università di Verona
Dipartimento di Informatica

Architetture Multimediali

Garanzia della Qualità del Servizio di flussi simulati mediante NS-2

Davide Quaglia
Giuseppe Di Guglielmo

Scopo di questa esercitazione è l'utilizzo di tecniche per garantire la Qualità del Servizio su rete IP e la verifica del loro funzionamento mediante la simulazione della trasmissione di un flusso multimediale.

1. Problemi con le congestioni

Esempio 1

Provare a simulare lo scenario `congestione2.tcl` in cui due flussi CBR identici a 80 kb/s vengono immessi sullo stesso canale limitato a 100 kb/s.

Verificare il throughput del flusso 1000 al nodo 4 e del flusso 2000 al nodo 5 con lo script `./throughput-flow-F-at-node.pl`. Quale dei due flussi perde più pacchetti ?

Verificare il jitter del flusso 1000 al nodo 4 e del flusso 2000 al nodo 5 con lo script `./jitter-flow-F-at-node.pl`. Quale dei due flussi ha ritardi più irregolari ?

Analizzare l'andamento della dimensione della coda all'inizio del bottleneck. Su quale valore si assesta la sua dimensione (in pacchetti) ?

2. Politiche avanzate di gestione delle code dei router

Finora abbiamo visto la politica “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 contenporaneamente e altrettanto contemporaneamente vi reagiscono abbassando il bitrate per poi rialzarlo contemporaneamente; questa

sincronizzazione nel comportamento delle sorgenti abbassa l'efficienza nell'utilizzazione del bottleneck.

Per questo motivo è stato studiato un'altra tecnica di gestione delle code chiamata *Random Early Discard* (RED) che essenzialmente prevede l'eliminazione di un pacchetto non quando la coda è ormai piena ma secondo una politica probabilistica legata alla stima della dimensione futura della coda calcolata sulla base della sua storia passata. La probabilità che un pacchetto in arrivo venga scartato aumenta con l'aumento di tale lunghezza stimata; quindi se la coda è stata per lo più vuota nel passato tale probabilità sarà bassa mentre se la coda è stata per lo più piena in passato allora la probabilità sarà alta.

La tecnica RED consiste di due aspetti: il calcolo della stima della lunghezza della coda e la decisione sulla perdita di un pacchetto in arrivo.

La stima della lunghezza della coda si ottiene calcolando la media esponenziale che dipende dalla storia passata e dalla dimensione attuale della coda `q`:

```
avg[i] = 0                        se i=0
       = (1-w)avg[i-1] + w*q      se i>0
```

All'arrivo del pacchetto i-esimo esso viene scartato con un probabilità `p(avg[i])` che dipende dalla stima della lunghezza della coda all'istante i-esimo `avg[i]`. Come mostrato in Figura 1, `p(avg[i])` vale 0 quando `avg[i]` è minore di `min_th`, vale 1 quando `avg[i]` è maggiore di `max_th` e varia linearmente da 0 a `max_p` per valori di `avg[i]` tra `min_th` e `max_th`. Ovviamente `max_p` deve essere compreso tra 0 e 1.

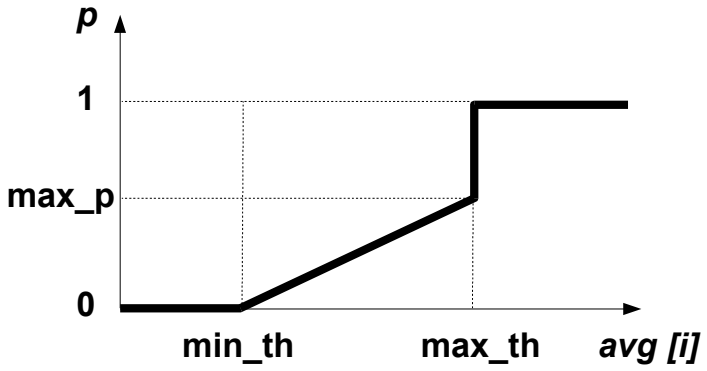

Figura 1. Probabilità di scarto in funzione della lunghezza stimata della coda.

Per utilizzare una coda RED in NS-2 è sufficiente sostituire `droptail` con `RED` nella dichiarazione del link

```
$ns duplex-link $node2 $node3 100kb 20ms RED
```

Per impostare i parametri delle code RED occorre mettere in testa al file tcl:

```
Queue/RED set thres_ 5          # min_th
```

```
Queue/RED set maxthres_ 15      # max_th
Queue/RED set linterm_ 1        # max_p = 1/linterm_
Queue/RED set q_weight_ 0.002   # w
```

Per stampare su file la dimensione in byte del valore istantaneo e stimato della coda RED occorre aggiungere le seguenti istruzioni:

```
set redq [[$ns link $node2 $node3] queue]
set traceq [open congestioneRED.redq w]
$redq trace curq_
$redq trace ave_
$redq attach $traceq
```

Verrà creato un file contenente 3 colonne: la prima indica se si sta riportando il valore istantaneo (lettera "Q") o stimato (lettera "a") della coda, la seconda riporta il tempo e la terza il valore in questione (in byte non in numero di pacchetti). Si può filtrare tale file nei suoi due tipi di valori con il comando `grep` ma è pure utile vedere, a parità di tempo, come il valore stimato insegue quello reale.

Esempio 2

Simulare lo scenario `congestioneRED.tcl` e analizzare throughput e jitter dei due flussi CBR come nell'Esempio 1. Cosa si può notare ora nel comportamento dei due flussi ?

Analizzare l'andamento della dimensione della coda all'inizio del bottleneck. Su quale valore si assesta la sua dimensione (in pacchetti) ?

Analizzare l'andamento del valore istantaneo e stimato della coda leggendo il file `congestioneRED.redq`. Come sono i valori stimato e reale all'inizio della simulazione e man mano che il tempo passa ?

Esercizio: verificare cosa succede con tre flussi CBR concorrenti invece che due.

3. Differentiated Services

L'architettura tradizionale di Internet mira a far sì che tutte le connessioni vengano trattate dalla rete allo stesso modo e quindi ricevano la stessa Qualità del Servizio. Ad esempio nel Capitolo precedente abbiamo visto come questo possa avvenire grazie ad una gestione più sofisticata delle code.

Tuttavia è sorta l'esigenza di poter discriminare tra classi di applicazioni/connessioni e di poter trattare ciascuna connessione secondo la classe di appartenenza. In questo modo un utente disposto a pagare di più può ricevere una Qualità del Servizio migliore in termini di minore e più costante ritardo e maggiore throughput. Questo è particolarmente vero per le applicazioni multimediali.

Per questa ragione il modello Differentiated Services è stato introdotto. Esso prevede di marcare i pacchetti 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. Un modo

comune di fare ciò è usare code RED con parametri diversi a seconda della marcatura dei pacchetti.

NS-2 implementa il modello "Assured Forwarding" di Differentiated Services come descritto nella RFC 2597. Le connessioni possono essere classificate 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 per marcare i pacchetti; tali etichette determinano nei nodi interni alla rete le politiche di scheduling delle code e di scarto dei pacchetti.

In una rete DiffServ occore definire dei nodi aggiuntivi con la funzione di "edge router" mostrati in grigio nella Figura 2 che mostra lo scenario contenuto nel file `diffserv.tcl`.

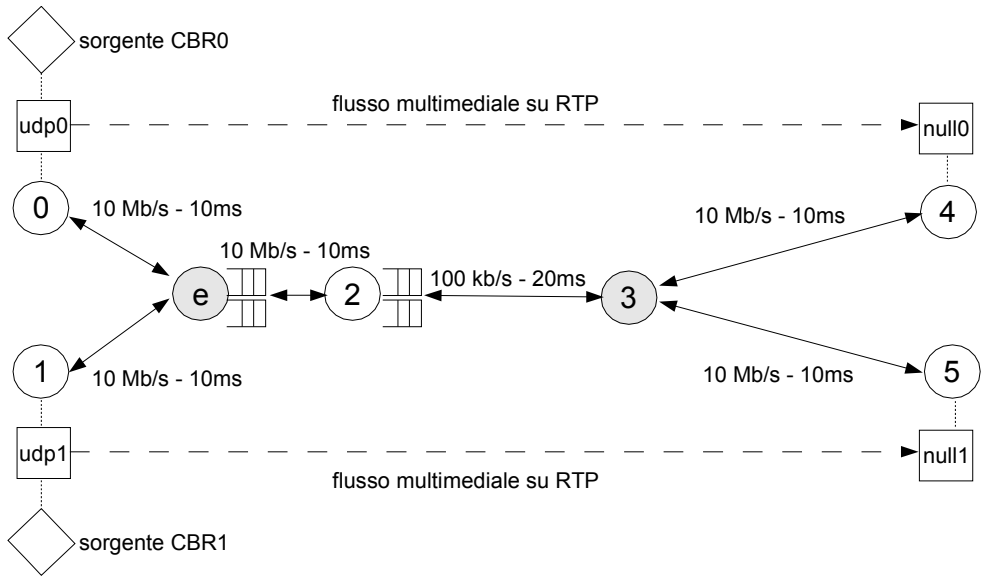

Figura 2. Topologia DiffServ dello scenario `diffserv.tcl`.

Per creare code DiffServ nei nodi interni alla rete occorre usare link semplici e non full-duplex come fatto finora (per creare un link full-duplex si usano due link semplici):

```
$ns simplex-link $edge $node2 10Mb 10ms dsRED/edge
```

Per determinare il numero di code fisiche si usa il comando:

```
set qe2 [[$ns link $edge $node2] queue]
$qe2 set numQueues_ $m
```

dove `$m` può valere da 1 a 4.

Per determinare il numero di code virtuali per ciascuna coda fisica si usa il comando:

```
set qe2 [[$ns link $edge $node2] queue]
$qe2 setNumPrec $n
```

dove `$n` può valere da 1 a 3.

NOTA: per semplificare la trattazione nel seguito considereremo una sola coda virtuale associata a ciascuna coda fisica.

I parametri RED di ciascuna delle `$m x $n` code sono definiti col comando:

```
set qe2 [[$ns link $edge $node2] queue]
$qe2 configQ $queueNum $virtualQueueNum $min_th $max_th $maxp
```

un sesto parametro `$w` può essere fornito altrimenti viene assunto il valore 0.002.

Per definire la dimensione media del pacchetto si usa il comando:

```
set qe2 [[$ns link $edge $node2] queue]
$qe2 meanPktSize 400
```

Per dire all'edge router come marcare i pacchetti occorre usare 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 pesa diversi alle varie code come si vede nel frammento di codice seguente dove la coda 0 verrà servita il 20% del tempo mentre la coda 1 l'80%:

```
set q23 [[$ns link $node2 $node3] queue]
$q23 set numQueues_ 2
$q23 setNumPrec 1
$q23 setSchedularMode WRR
$q23 addQueueWeights 0 2
$q23 addQueueWeights 1 8
```

Esercizio: studiare throughput e jitter dei due flussi CBR dello scenario diffserv.tcl; successivamente modificare i pesi dell'algoritmo WRR in modo che un flusso sia servito con priorità rispetto al secondo e verificare cosa avviene a throughput e jitter.