

# Esercitazione su Networked Control Systems

Thomas Battisti, Daniele Coati, Davide Quaglia

June 6, 2014

## 1 Esercitazione

### 1.1 Installazione

I passi per installare le librerie SystemC-AMS, SCNSL e l'applicazione utilizzata in questa esercitazione sono qui sotto riportati:

- Scaricare e scompattare il pacchetto *NCS-Diffserv.zip* all'interno della directory `/tmp`.
- Esportare la variabile d'ambiente *LD\_LIBRARY\_PATH* per agganciare la libreria dinamica di SystemC

```
>> export LD_LIBRARY_PATH=/usr/local/systemc-2.3.0/lib-linux64
```

- Dare i permessi di esecuzione agli script `start.sh`, `stats.sh` e `plot.sh` posizionati nella directory `/tmp/CTPL`
- successivamente eseguire

```
>> make clean
>> make
```

- Per avviare la simulazione lanciare lo script `./start.sh`

```
>> ./start.sh
```

- Per rivedere le statistiche globali dell'ultima simulazione

```
>> ./stats.sh
```

- Per rivedere i grafici dell'ultima simulazione

```
>> ./plot.sh
```

Lo script *plot.sh* genera i grafici in `/tmp/CTPL/graphs/` e li visualizza. Una nuova simulazione sovrascrive tali file per cui, se si vuole conservarli per confronto, occorre rinominarli.

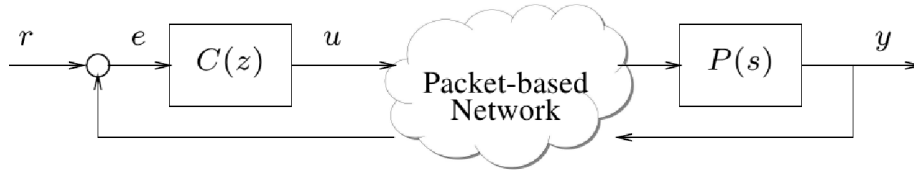


Figure 1: Networked Control System.

## 1.2 Networked control system

Lo schema del networked control system usato in questa esercitazione è riportato in Figura 1. Il controllore digitale  $C(z)$  invia comandi  $u_k$  all'impianto  $P(s)$  costituito da un sistema dinamico continuo. L'obiettivo del controllo è minimizzare l'errore  $e_k$  tra l'uscita  $y_k$  e il riferimento  $r_k$ . Comandi  $u_k$  e misure  $y_k$  sono incapsulati in pacchetti e spediti sulla rete.

## 1.3 Tecniche di modellazione e simulazione

Tutto lo scenario è modellato e simulato tramite SystemC. La rete (si veda la topologia in Figura 4) è implementata mediante la libreria SCNSL che fornisce le primitive per creare nodi di rete collegati tramite link, router, e code che gestiscono la priorità secondo il modello Differentiated Services. Il controllore digitale e l'impianto sono realizzati mediante l'estensione SystemC-AMS per sistemi dinamici (Analog and Mixed Signals - AMS). In particolare, il controllore è stato realizzato mediante il paradigma Timed Data Flow (TDF) che consente di modellare sistemi a segnali campionati mentre l'impianto è stato realizzato con il paradigma Linear Signal Flow (LSF) che consente di modellare sistemi descritti da equazioni differenziali lineari a coefficienti costanti.

## 1.4 Descrizione file

Nella directory `/tmp/CTPL/src/` si trovano i sorgenti del controllore e dell'impianto. In particolare vediamo:

- *ControllerTask.cc* contiene il Task che implementa le funzionalità del controllore (Figura 2). Esso riceve i pacchetti dall'impianto con la misura  $y$  e le statistiche di rete su ritardi e packet loss rate per ogni classe di priorità, calcola il prossimo comando da inviare e la sua priorità (se è attivato il CScheduler). Il calcolo del prossimo comando da inviare e la sua priorità viene effettuato dal modulo *controller\_tdf.cc*, mentre l'oggetto *ControllerIFace.cc* è l'interfaccia che collega i segnali SystemC con i relativi segnali SystemC-AMS TDF.
- *PlantTask.cc* contiene il Task che implementa le funzionalità dell'impianto (Figura 3). Riceve i pacchetti dal controllore con i comandi, calcola le statistiche di rete, applica il comando all'impianto, e infine trasmette l'osservazione e le statistiche di rete al controllore. La dinamica continua dell'impianto è modellata nel modulo *plant\_lsf.cc* mediante SystemC-AMS LSF.

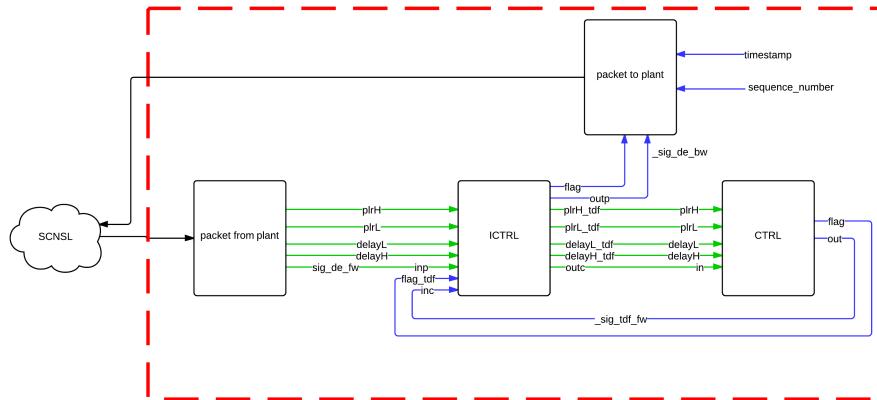


Figure 2: Controllore

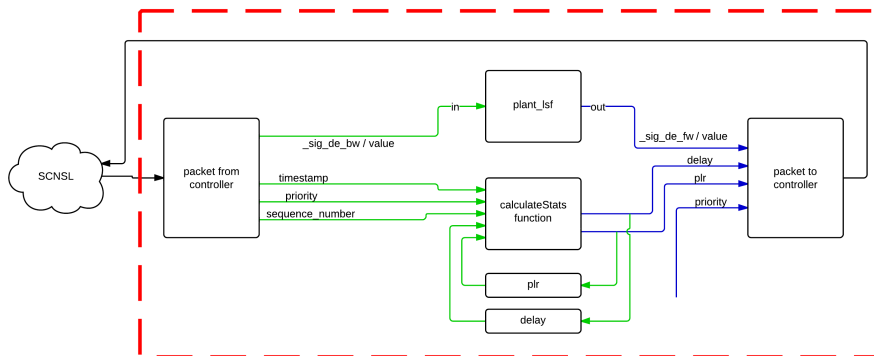


Figure 3: Impianto

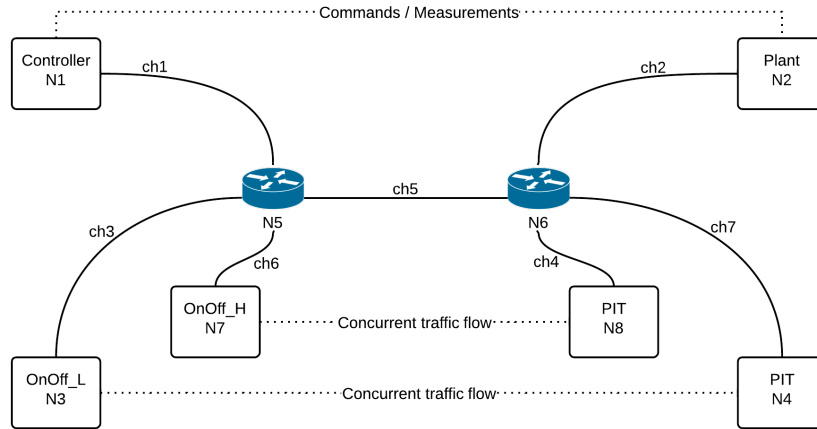


Figure 4: Topologia di rete

## 1.5 Scenario di rete

In Figura 4 è mostrata la topologia della rete modellata nel simulatore SCNSL; essa rappresenta una classica topologia bottleneck con link periferici ad alta capacità dedicati per le varie applicazioni che insistono su un link condiviso in cui possono crearsi *fenomeni di congestione*. Tutti i link sono bidirezionali. Si notino i flussi informativi che condividono lo stesso canale. Il traffico concorrente (task 3 e task 7) è di tipo constant bitrate (CBR) con andamento ON/OFF nella direzione dal controllore all'impianto.

L'idea innovativa che si vuole analizzare in questa esercitazione è quella di usare una rete Differentiated Services (DiffServ) sopra la topologia mostrata al fine di dare priorità ai pacchetti relativi al NCS.

Per semplicità di modello sono stati accorpati in un'unica entità sia gli host che generano e ricevono i dati sia gli edge router che effettuano policing e marcatura dei pacchetti. In particolare vedremo che la politica di marcatura terrà conto delle problematiche di controllo e quindi dovrà essere effettuata direttamente nell'host (sarebbe come dire che l'edge router delega l'host alla marcatura).

I core router (N5 e N6) gestiscono la capacità del bottleneck mediante uno schema *a due priorità* e quindi con due code.

Il task 3 e il task 7 generano traffico concorrente con priorità fissa bassa e alta, rispettivamente.

Nel file *defineList.hh* all'interno della cartella */tmp/CTPL/include* sono riportati i parametri di simulazione usati per questa topologia:

- *PACKET\_SIZE* dimensione del pacchetto in byte.
- *QUEUE\_SIZE* dimensione delle code in byte.
- *DELAY* ritardo in secondi di tutti i link
- *CAPACITY* capacità dei link periferici in bit/sec

- *CAPACITY\_BOTTLENECK* capacità del link condiviso (bottleneck) in bit/sec
- *CONCURRENT\_TRAFFIC\_BITRATE\_H* bitrate del traffico concorrente generato dal task 7 quando si trova nello stato ON (in bit/sec)
- *CONCURRENT\_TRAFFIC\_BITRATE\_L* bitrate del traffico concorrente generato dal task 3 quando si trova nello stato ON (in bit/sec)
- *ON\_L* lunghezza del periodo di ON del task 3 in ms
- *ON\_H* lunghezza del periodo di ON del task 7 in ms
- *OFF\_L* lunghezza del periodo di OFF del task 3 in ms
- *OFF\_H* lunghezza del periodo di OFF del task 7 in ms
- *WITH\_MARKER* attiva l'algoritmo di determinazione dinamica della priorità dei pacchetti del controllore
- *NO\_MARKER\_PRIORITY\_H* per inviare tutti i pacchetti del controllore in alta priorità
- *NO\_MARKER\_PRIORITY\_L* per inviare tutti i pacchetti del controllore in bassa priorità

Il file *main.cc* serve per configurare la rete bottleneck come in Figura 4. Per attivare i task che generano traffico concorrente basterà scommentare la chiamata alla corrispondente funzione `enable()` come mostrato alle righe 509 e 510 del file *main.cc*.

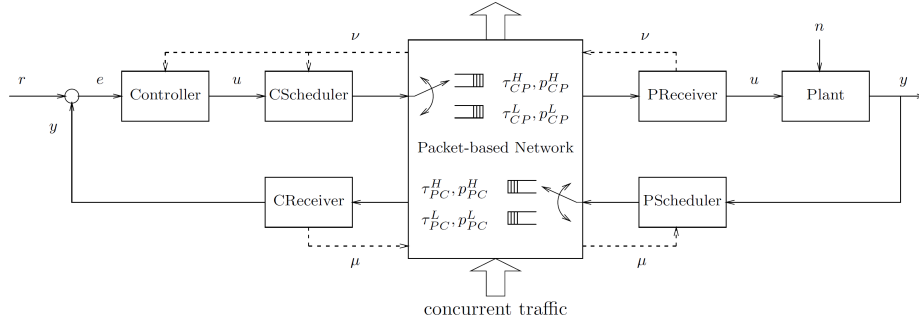


Figure 5: Architettura del DiffServ NCS

## 2 Architettura NCS basata su DiffServ

Il problema della classificazione dei pacchetti si basa su due aspetti:

1. la stima dell'importanza del pacchetto (contenente comando o misura),
2. l'allocazione delle risorse (uso di alta/bassa priorità).

Per valutare "l'importanza" di un pacchetto abbiamo bisogno della conoscenza di un modello dell'impianto, del controllore e dello stato della rete (ritardo di comunicazione e packet loss rate).

Lo schema in Figura 5 presenta l'architettura NCS basata su DiffServ che è composta dei seguenti elementi:

- **CScheduler**: marcatore lato controllore per decidere la politica di invio dei comandi.
- **PScheduler**: marcatore lato impianto per decidere la politica di invio delle misure.
- $\tau_{CP}^H, \tau_{CP}^L, p_{CP}^H, p_{CP}^L$  and  $\tau_{PC}^H, \tau_{PC}^L, p_{PC}^H, p_{PC}^L$ : ritardi di comunicazione e packet loss rates. Questi segnali dipendono dallo stato della rete.
- **CReceiver**: ricevitore lato controllore. Il ricevitore calcola le stime dei parametri di rete  $\mu = \{\hat{\tau}_{PC}^H, \hat{\tau}_{PC}^L, \hat{p}_{PC}^H, \hat{p}_{PC}^L\}$  da inviare al PScheduler.
- **PReceiver**: ricevitore lato impianto. Il ricevitore calcola le stime dei parametri di rete  $\nu = \{\hat{\tau}_{CP}^H, \hat{\tau}_{CP}^L, \hat{p}_{CP}^H, \hat{p}_{CP}^L\}$  da inviare al CScheduler.

### Algoritmo per il CScheduler

La scelta della politica al passo corrente  $\pi_k$  viene decisa dopo aver eseguito i passi:

1. calcolare la stima dell'uscita assumendo la trasmissione del comando usando la priorità  $H$ ,  $\hat{\mathbf{y}}_{get}^H$ , (nella stima si usa  $\hat{\tau}_{CP}^H$ );
2. calcolare la stima dell'uscita assumendo la trasmissione del comando usando la priorità  $L$ ,  $\hat{\mathbf{y}}_{get}^L$ , (nella stima si usa  $\hat{\tau}_{CP}^L$ );
3. calcolare la stima dell'uscita assumendo la perdita del pacchetto,  $\hat{\mathbf{y}}_{lost}$ , (in questo caso si assume di applicare nuovamente l'ultimo comando ricevuto);

4. si pesano le stime con la probabilità di perdita derivata dalle statistiche sul packet loss rate.

$$\begin{aligned}\hat{y}^H(k) &= (1 - \hat{p}_{CP}^H(k))\hat{y}_{get}^H(k) + \hat{p}_{CP}^H(k)\hat{y}_{lost}(k) \\ \hat{y}^L(k) &= (1 - \hat{p}_{CP}^L(k))\hat{y}_{get}^L(k) + \hat{p}_{CP}^L(k)\hat{y}_{lost}(k)\end{aligned}$$

5. si calcola la differenza tra le due stime precedenti

$$\hat{e}(k) = \hat{y}^L(k) - \hat{y}^H(k)$$

6. si utilizza una soglia per decidere la priorità di invio;

```

if  $|\hat{e}(k)| > E$ 
  then  $\pi_k = H$ 
  else  $\pi_k = L$ 

```

dove  $E$  è un parametro deciso dal progettista e dipende dall'applicazione.

### 3 Esercizi

1. Lanciare una simulazione in cui sia il traffico NCS che la sorgente concorrente sono sempre in bassa priorità. Occorre abilitare il task 3 ma non il task 7. Visualizzare l'andamento di packet loss rate e ritardo dei comandi e andamento del tracking error.
  - qual è il ritardo minimo dei comandi?
  - qual è il ritardo massimo dei comandi?
  - qual è la percentuale massima di comandi persi?
  - perché il ritardo dei comandi ad un certo punto inizia a crescere? per quanto tempo cresce?
  - riportare su un foglio la varianza del tracking error e il packet loss rate (PLR) stampati sul terminale dopo la simulazione; serviranno per confrontarli con quelli degli esercizi successivi.
  - modificare la dimensione delle code e verificare come influenza il ritardo dei comandi e la percentuale di perdita dei pacchetti.
  - qual è il bitrate del traffico generato da controllore a impianto considerando che il tempo di campionamento del sistema è  $T_s = 0.02$  secondi e la dimensione dei pacchetti di comando è 56 byte?
  - trovare una capacità del bottleneck in modo tale che non ci siano perdite di pacchetti.
  - modificare il ritardo dei link e verificare come influenza il ritardo dei comandi.
2. Lanciare una simulazione alzando la priorità del traffico NCS mentre la sorgente concorrente rimane sempre in bassa priorità. Per fare questo bisogna abilitare il define `NO_MARKER_PRIORITY_H`. Visualizzare l'andamento di packet loss rate e ritardo dei comandi e andamento del tracking error.

- cosa succede circa l'andamento di ritardo, packet loss rate dei comandi e tracking error?
  - riportare su un foglio la varianza del tracking error e il packet loss rate (PLR) stampati sul terminale dopo la simulazione; confrontarli con quello dell'esercizio 1.
3. Lanciare una simulazione usando il marker intelligente (define *WITH\_MARKER*) con sorgente concorrente in bassa priorità. Visualizzare l'andamento di packet loss rate e ritardo dei comandi e andamento del tracking error.
- cosa succede circa l'andamento di ritardo, packet loss rate dei comandi e tracking error?
  - riportare su un foglio la varianza del tracking error e il packet loss rate (PLR) stampati sul terminale dopo la simulazione; come si posizionano rispetto agli esperimenti 1 e 2?
  - modificare la soglia  $E$  del CScheduler nel file *controller\_tdf.hh* e, per ogni valore di  $E$ , annotare la percentuale di pacchetti in alta/bassa priorità e la varianza del tracking error stampate sul terminale dopo la simulazione; come variano?
4. Lanciare una simulazione usando il marker intelligente (define *WITH\_MARKER*) con entrambe le sorgenti concorrenti in bassa priorità e in alta priorità. Modificare il file *defineList.hh* in modo che il traffico concorrente generi in totale 75600 bit/sec. Visualizzare l'andamento di packet loss rate e ritardo dei comandi e andamento del tracking error.
- cosa succede circa l'andamento di ritardo, packet loss rate dei comandi e tracking error?
  - riportare su un foglio il valore complessivo del tracking error.
  - modificare la soglia  $E$  del CScheduler nel file *controller\_tdf.hh* e, per ogni valore di  $E$ , annotare la percentuale di pacchetti in alta/bassa priorità e la varianza del tracking error stampate sul terminale dopo la simulazione; come variano?