Politecnico di Milano
FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

Corso di Piattaforme Software per la rete – MODULO 2
anno accademico 2013-2014

Prof. William FORNACIARI

TRACCIA DI SOLUZIONE (12 LUGLIO 2013)

Quesito D1 (5)

Nel sistema vengono creati tre processi (P1, P2, P3) con le durate e i Pid indicati in figura. Supponendo che ogni 100ms vengano prese le decisioni di scheduling da parte del sistema operativo, si richiede di:

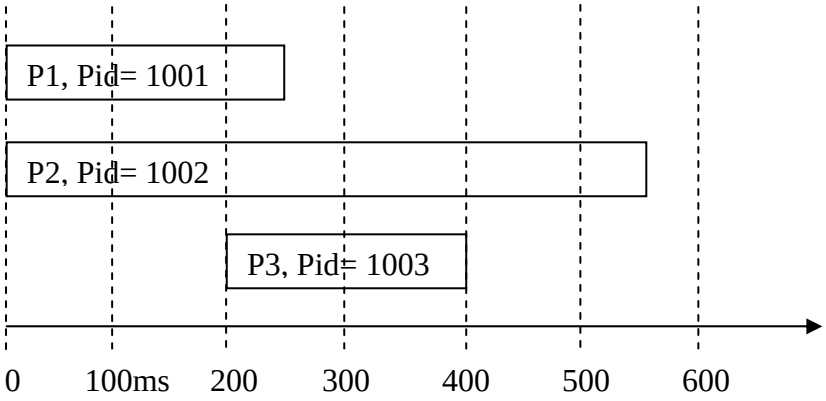
1) Mostrare (anche in modo grafico) l'evoluzione nel tempo dell'occupazione della CPU da parte dei tre processi.
2) Il calcolo della latenza di ogni processo, inteso come la differenza fra il tempo di completamento ed il tempo di arrivo nel sistema da parte di ogni singolo processo

Si valutino i due punti precedenti per ciascuna delle seguenti politiche di scheduling.

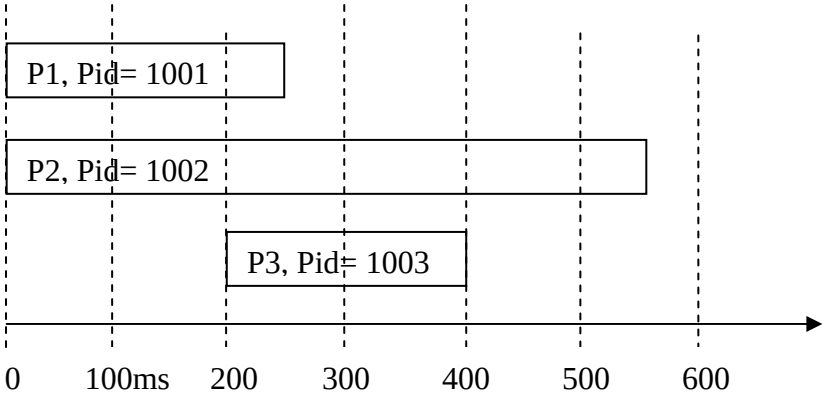
- Round- Robin con prelazione
- Round-Robin senza prelazione
- Shortest Remaining Time

Facoltativo

- Calcolare la latenza media di ciascun algoritmo di scheduling, eventualmente commentando tali valori (ideali) rispetto all'esistenza di un overhead di commutazione.

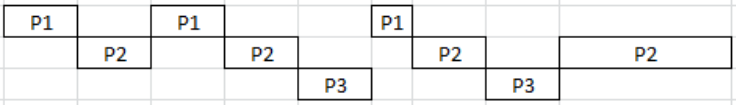


Risposta D1 (10)



Nota: possibile discussione su rilascio o meno della cpu prima dello scadere del quanto di tempo. Nel caso i diagrammi potrebbero avere uno “tempo morto”.

RR with preemption

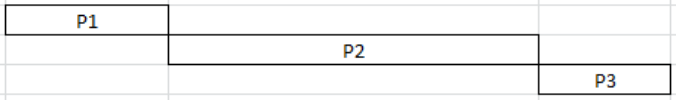


100 200 300 400 500 650 750 1000

P3 quando creato viene messo in fondo alla coda

L1=	550-0=550
L2=	000-0=1000
L3=	50-200=550
Lmedia	(550+1000+550)/3=700

RR no preemption (FIFO)



250 800 1000

L1=	250-0=250
L2=	800-0 = 800
L3=	1000-200=800
Lmedia	(250+800+800)/3=617

Shortest Remaining Time

250 450 1000

L1=	250-0=250
L2=	1000-0=1000
L3=	450-200=250
Lmedia	(250+250+1000)/3=500

Quesito D2 (8)

Dato il seguente sorgente:

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

int flip;
void my_handler_1(int param){
        perror("Work in progress");
}
void my_handler_2(int param){
flip=1;
}
void my_handler_3(int param){
exit(0);
}

int main(int argc, char* argv[]){

        struct sigaction handler_1,handler_2,handler_3,old_handler_1,old_handler_2;
        handler_1.sa_handler=&my_handler_1;
        handler_2.sa_handler=&my_handler_2;
        handler_3.sa_handler=&my_handler_3;
        sigaction(SIGQUIT,&handler_1,&old_handler_1);
        sigaction(SIGUSR1,&handler_2,&old_handler_2);

        while(1){
                fprintf(stderr,"Yawn...\n");
                if (flip){sigaction(SIGCONT,&handler_3,&old_handler_1);}
                flip=0;
                sleep(5);
                }
        return 0;
}
```

Cosa succede se al processo di cui sono dati i sorgenti vengono inviate le seguenti sequenze di segnali nell' ordine dato?

Sequenza 1	**Sequenza 2**	**Sequenza 3**	**Sequenza 4**
QUIT	QUIT	QUIT	KILL
USR1	QUIT	USR1	CHLD
TERM	QUIT	CONT	CONT

Risposta D2

QUIT - il processo stampa "work in progress" dall' handler di SIGQUIT
USR1 - il processo cambia l' handler di SIGCONT
TERM - il processo termina

QUIT - il processo stampa "work in progress" dall' handler di SIGQUIT
QUIT - il processo stampa "work in progress" dall' handler di SIGQUIT
QUIT il processo stampa "work in progress" dall' handler di SIGQUIT

QUIT - il processo stampa "work in progress" dall' handler di SIGQUIT

USR1 - il processo cambia l' handler di SIGCONT
CONT - il processo esce, dall' handler di SIGCONT

KILL - il processo muore
CHLD - nulla
CONT - nulla

Quesito D3 (9)

Quale errore concettuale è presente nel seguente frammento di codice e come è possibile risolverlo? Indicare cosa va eliminato e cosa va aggiunto affinché il programma seguente sia in grado di trasmettere dati.

```
int main(int argc, char *argv[])
{
   int socket_fd;
   struct sockaddr_in name;
   struct hostent* hostinfo;
   if ( (socket_fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
      perror("Socket creation error");
      return -1;
   }
   hostinfo = gethostbyname ("127.0.0.1");
   errorlog(hostinfo,NULL);
   name.sin_addr = *((struct in_addr *) hostinfo->h_addr);
   name.sin_port = htons (9999);
   connect(socket_fd,(struct sockaddr*) &name, sizeof (struct sockaddr_in));

   /*trasmissione dati*/

   return 0;
}
```

Risposta D3

Il codice utilizza un socket UDP tentando di trasmettere dati previa esplicita chiamata alla connect per instaurare una sessione. Questo non è necessario e il codice può essere corretto sia cambiando il tipo di protocollo (cambiare SOCK_DGRAM in SOCK_STREAM) alla creazione del socket, sia eliminando la connect.

Quesito D4

Indicate quali regole iptables scattano e, quante volte, su una macchina configurata con la seguente lista di comandi :

iptables -A INPUT -s 192.168.3.3 -j LOG
iptables -A INPUT -i lo -j ACCEPT
iptables -P FORWARD ACCEPT
iptables -A FORWARD -i lo -j ACCEPT
iptables -A INPUT -s 192.168.3.4/24 -j RATEEST iptables -A FORWARD -d 46.59.1.2 -j ACCEPT iptables -I INPUT 2 -s 192.168.3.3 -j DROP iptables -I INPUT 2 -s 192.168.3.3 -j QUEUE iptables -P OUTPUT ACCEPT iptables -A INPUT -d 192.168.3.42 -j ACCEPT iptables -A INPUT -s 192.168.3.7 -j DROP iptables -A INPUT -s !192.168.3.3 -j ACCEPT iptables -P INPUT DROP iptables -A FORWARD -s 192.168.3.4/24 -j RATEEST iptables -A FORWARD -s 192.168.3.4/24 -j ACCEPT iptables -A FORWARD -p tcp --dport 53 -j LOG

Assumendo che passino i seguenti pacchetti:

1 - 192.168.3.7:12943 -> 192.168.3.42:123, TCP
2 - 192.168.3.42:12943 -> 192.168.3.42:80, TCP
3 - 192.168.3.3:12943 -> 8.8.8.8:53, TCP
4 - 192.168.3.3:12943 -> 192.168.3.4:80, TCP
5 - 46.59.1.2:12943 -> 192.168.3.42:80, TCP

L' host abbia 2 interfacce di rete, una esposta con indirizzo pubblico 131.175.1.2/24, l' altra verso una rete interna con indirizzo 192.168.3.42/24.
Il forwarding dei pacchetti ip è abilitato.

Indicate anche quali regole sono inutili (ovvero quali regole potete rimuovere senza cambiare il comportamento della macchina), indipendentemente dal traffico che transita.

Risposta D4

Riordinando il ruleset, tenendo conto degli inserimenti ordinati con l' opzione -I otteniamo, per l' hook di input:

Policy : DROP (settata dall' ultimo comando relativo a INPUT)
iptables -A INPUT -s 192.168.3.3 -j LOG
iptables -I INPUT 2 -s 192.168.3.3 -j QUEUE
iptables -I INPUT 2 -s 192.168.3.3 -j DROP
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -s 192.168.3.4/24 -j RATEEST
iptables -A INPUT -d 192.168.3.42 -j ACCEPT
iptables -A INPUT -s 192.168.3.7 -j DROP
iptables -A INPUT -s !192.168.3.3 -j ACCEPT

Mentre per quello di forward :
Policy : ACCEPT (settata dal primo comando relativo ad ACCEPT)
iptables -A FORWARD -i lo -j ACCEPT

iptables -A FORWARD -d 46.59.1.2 -j ACCEPT
iptables -A FORWARD -s 192.168.3.4/24 -j RATEEST
iptables -A FORWARD -s 192.168.3.4/24 -j ACCEPT
iptables -A FORWARD -p tcp --dport 53 -j LOG

per l' hook di output abbiamo solo la configurazione della policy ad ACCEPT.

Il pacchetto 1 è destinato alla macchina, sull' interfaccia di rete interna: verrà quindi analizzato dall' hook di INPUT e accettato dalla regola iptables -A INPUT -d 192.168.3.42 -j ACCEPT , senza che ne scattino altre

Il pacchetto 2 è inviato dalla macchina a sé stessa: viene instradato sull' interfaccia di loopback dalla struttura di Netfilter e viene quindi accettato dalla regola iptables -A INPUT -i lo -j ACCEPT

Il pacchetto 3 proviene dalla rete interna e viene instradato verso l' esterno, venendo quindi analizzato dall' hook FORWARD: fa scattare la regola iptables -A FORWARD -s 192.168.3.4/24 -j RATEEST che ne tiene conto nei contatori interni e viene accettato dalla regola successiva. Non viene loggato dall' ultima regola.

Il pacchetto 4 proviene dalla rete interna ed è destinato ad essa: viene quindi analizzato dall' hook FORWARD: il comportamento è analogo al pacchetto 3.

Il pacchetto 5 arriva dall' esterno ed è destinato alla macchina, viene quindi considerato dall' hook di INPUT: viene accettato da iptables -A INPUT -d 192.168.3.42 -j ACCEPT