Università di Pisa Dipartimento di Informatica Corso di Laurea in Informatica

Progetto di Laboratorio di Sistemi Operativi

Sessione estiva a.a. 2013-14

CANDIDATO: MARIO VITI.
MATRICOLA: 496105
CORSO: B

INDICE.

INTRODUZIONE.
Il progetto consiste nell'implementazione di un protocollo di trasmissione criptato OOB (out of band).
L'obbiettivo è inviare un numero segreto (secret) ad un server.Il secret non è comunicato in chiaro ma viene ricavato dall'analisi di dati apparentemente insignificanti, come le dimensioni di un pacchetto dati inviato vuoto o come nel nostro caso dal tempo di attesa che intercorre fra due comunicazioni successive.

IMPLEMENTAZIONE.
Dei processi (clients) comunicano attraverso socket a p processi (server) (1 < p < k , k=#server totali) un intero generato casualmente di 8 byte denominato ID. Ogni secret msec (secret=1,.,3000) il client invia il suo ID ad un server casuale fra i p sclelti.
Successivamente un processo supervisor ha il compito di stimare il secret comunicando privatamente con tutti i k server.

PROTOCOLLO SUPERVISOR-SERVER.
Supervisor e server sono processi distinti e comunicano attraverso una pipe senza nome dedicata per ogni server.
Un messaggio consta di ID e secret.
Dato che la comunicazione è privata per ogni messaggio leggono e scrivono un numero convenzionalmente stabilito di byte dal buffer della pipe.
Ogni buffer viene processato con un cast per accedere ai campi ID e secret.

PROTOCOLLO SERVER-CLIENT.
Il protocollo server client è pubblico ed è definito dalla specifica:
un client invia a un server 8 byte in network byte order.

STRUTTURA.
La struttura è rappresentata in fig.1
supervisor.
Il supervisor è strutturato:
multiprocess(k server).
multithreaded(k receiver + 1 gestore stampa).

thread main (supervisor).
Il ciclo principale del main thread consiste nel lancio di k server
e di k thread receiver e di un thread printer. Dopo questo ciclo si sospende in attesa della terminazione dei receiver con una chiamata a phtread_join
per ogni tid dei k thread reciever.

thread receiver (supervisor).
Il thread receiver -i legge in maniera bloccante dall'estremità di lettura della pipe anonima condivisa con il server -i. I dati ottenuti comprendono due valori: ID e secret.
Una volta ricevuti i dati il thread receiver aggiorna una CLIENT_list dove salva ID e minimo secret stimato.

thread printer (supervisor)
Il thread printer effettua il blocco e rimane in attesa del segnale sigint.
Al primo sigint: stampa su stderr il contenuto di CLIENT_list.
Al secondo sigint: se entro un secondo dal primo sigint, stampa su stdout e termina i k server mettendo in moto l'effetto Domino*, dopodichè termina.

server.
Il server è strutturato:multithreaded (uno per client)

thread main (server).
Il ciclo principale del thread main(server -i) consiste nella preparazione di una socket con binding all'indirizzo OOB-server-i.
I file descriptor ritornati dalla accept vengono passati ad un thread client_server dedicato al client appena connesso alla socket.

thread client_server (server).
Ogni client viene servito da un thread dedicato che stima il tempo che intercorre fra due invii dello stesso client al server -i.

client.
Strutturato single process.
Il client genera ID da 8 byte e il secret da 4byte pseudocasualmente.
Si connette a p server (1=<p<k). A questi invia ID per un numero w (w>3k) di volte in ordine casuale.

Domino*.
Il server -i terminando chiude l'estremità di scrittura della pipe condivisa con il thread receiver -i del supervisor.
il thread receiver -i legge eof dalla pipe chiusa e termina.
il thread main del supervisor viene svegliato dalla join e termina.

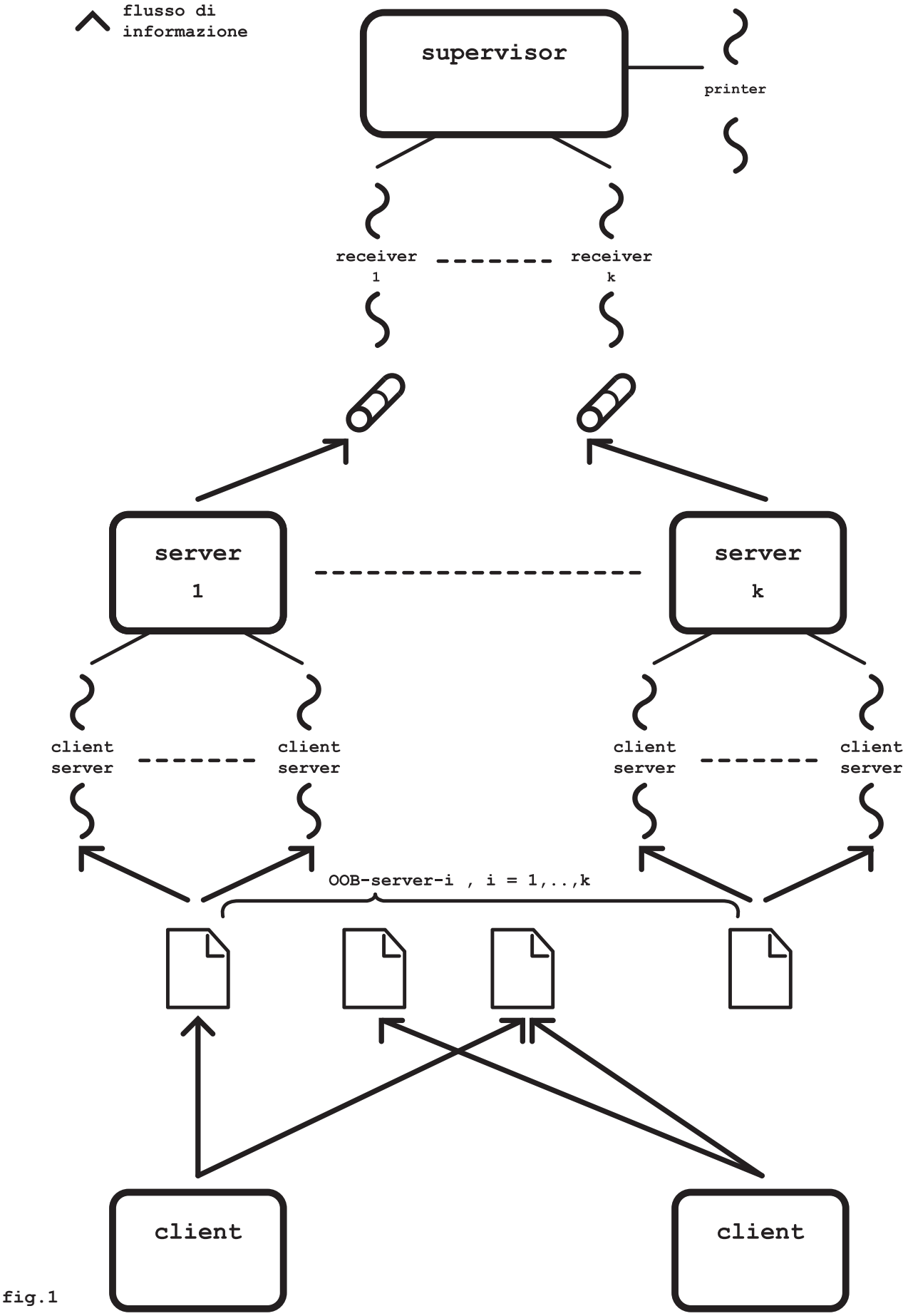

flusso di informazione
supervisor
printer
receiver 1
receiver k
server 1
server k
client server
client server
client server
client server
OOB-server-i , i = 1,..,k
client
client

fig.1

STRATEGIA.
Ogni server dovrà approssimare la propria stima come il minimo tempo intercorso fra due comunicazioniconsecutive (t_{i+1}- t_i) con lo stesso client.
Il supervisor calcolerà il secret come il minore fra i secret stimati da ogni server per quel particolare client.

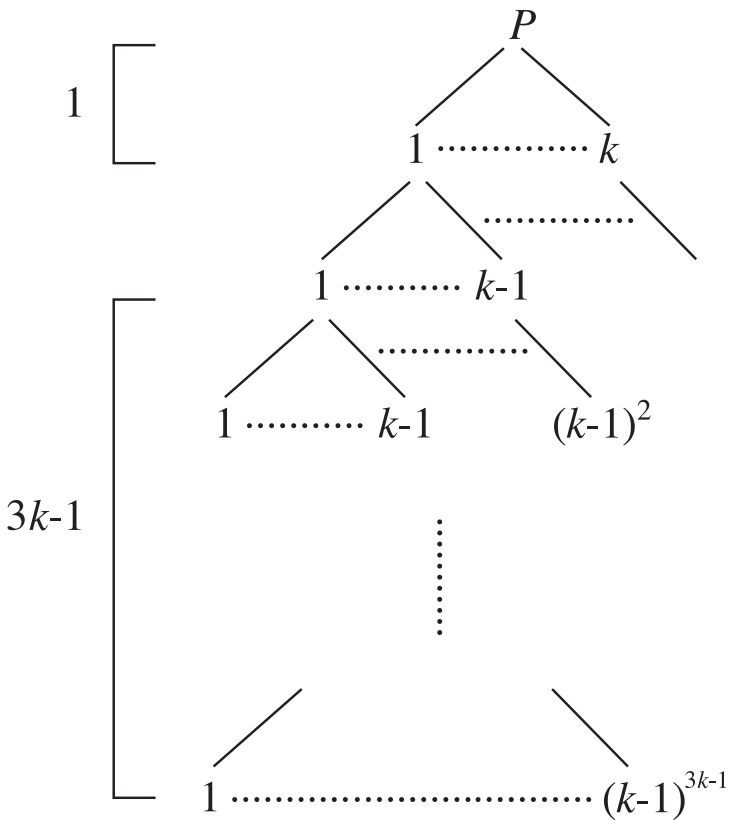
$$stima_{jl} = \min\left\{t^l_{i+1} - t^l_i\right\} \quad , \quad i = 1,.....,\#inv ii_l \quad , \quad j = 1,.....,\#server \quad l = 1,.....,\#clients$$

$$stima_{idl} = \min\left\{stima_{jl}\right\} \quad , \quad j = 1,.....,\#server \quad , \quad l = 1,.....,\#clients$$

EFFICACIA.
th.
Questa strategia è affidabile ma ha una percentuale di fallimento pari a 4.79%.

DIMOSTRAZIONE EFFICACIA.
Il caso favorevole in cui il superviosor stima esattamente un secret è quando un server viene contattato consecutivamente dopo secret msec dallo stesso client.
Tutte le altre stime saranno multipli interi positivi di secret, ovvero: n*secret > secret , n>1 .
Nel caso favorevole il minimo sarà il secret.
Nei casi sfavorevoli la stima sarà un multiplo di secret: n secret , n>1.
Possiamo modellare i casi sfavorevoli con gli alberi di decisione.
Abbiamo k casi possibili e 3k decisioni.
La prima diramazione ha k possibilià.
I successivi hanno k-1 possibilità per 3k-1 decisioni.
Ogni cammino è un caso "sfavorevole", il numero di foglie dell'albero è il numero di tutti i casi sfavorevoli.
L'albero è suddivisibile in k alberi (k-1)-arii di altezza 3k-1.

Dim



$$altezza(albero\ radicato\ in\ P) \setminus P = 3k\text{-}1 = \log_{k\text{-}1}\#foglie \quad => \quad \#foglie = (k\text{-}1)^{3k\text{-}1}$$

$$casi\ sfavorevoli = \#foglie\ di\ (albero\ radicato\ in\ P) = k(k\text{-}1)^{3k\text{-}1}$$

$$P(stima_{idl} = secret_l) = \frac{k^k - k(k\text{-}1)^{3k\text{-}1}}{k^k} \quad , \quad l = 1,.....,\#clients$$

$$\lim_{k->\infty} \frac{k^k - k(k\text{-}1)^{3k\text{-}1}}{k^k} \sim 1\text{-}0.0479 \quad \square$$

Campionamento.

La previsione viene rispettata dallo script
di test su un campione di 100 client.

```
-------------------------statistiche--------------------------

 Numero stime corrette entro 25msec di 100 clients è: 96
 Percentuale stime corrette: 96 %
 Errore medio nella stima di secret entro 25msec: 7 / 96
 Errore medio globale nella stima di secret : 6512 / 100

-------------------------statistiche--------------------------
```

OSSERVAZIONI.

Endian Awareness.
Le funzioni standard per la traduzione di interi in netowrk byte order esistono per gli interi a 32 bit.
Per un itero a 64 bit occorre una funzione ad hoc.

b: puntatore a byte.

bigendian.

```
  b+0 b+1 b+2 b+3 b+4 b+5 b+6 b+7
 |---|---|---|---|---|---|---|---|
   1   2   3   4   5   6   7   8
```

littlendian.

```
  b+7 b+6 b+5 b+4 b+3 b+2 b+1 b+0
 |---|---|---|---|---|---|---|---|
   1   2   3   4   5   6   7   8
```

Il netowrk byte order nel protocollo TCP adotta la convenzione dell'ordinamento bigendian.
Un modo che per la maggior parte dei casi funziona è di testare l'endianess tramite le funzioni standard.
La htonl restituisce un int in network byte order. Per una macchina che adotta il bigendian come ordine di salvataggio dei byte, la htnl è la funzione identità.
Per cui se una macchina è little endian, basterà verificare se l'intero dato come argomento della htnl
è diverso da quello restituito (deve essere significativo, per esempio diverso da 0).
In questo caso inverto l'ordine dei byte prima dell'invio.(versione implementata).
CAVEAT.
Ci sono dei casi in cui questa strategia non funziona: middleendian.
In questo caso bisgna forzare l'endianess con l'ordine di stampa.
L'ordine di stampa è lo stesso con il quale un intero viene "naturalmente" rappresentato.
In una stringa la cifra (nella base scelta) più significativa è quella puntata da b+0, l'ordine di stampa segue indirizzi crescenti.
Potremmo dire che la stampa è bigendian, o meglio lo è nella rappresentazione dei dati.
Stampare un numero forza l'endianess a rispettare l'ordine di rappresentazione.
La stampa è quindi endian aware e prescinde dall'endianess. In base a questo si può creare una funzione che sia universalmente endian aware.

Sigtimedwait.
La funzione sigtimedwait fa parte delle funzioni realtime POSIX, non è quindi obbligatorio che un sistema operativo fornisca un'imlementazione di questa funzione per potersi dichiarare POSIX-compliant.
Si avverte qundi che in quei sistemi che non hanno fornito le implementazioni delle funzioni realtime che il programma potrebbe non funzionare (Maco).