Scelte implementative nello sviluppo del progetto di Sistemi Operativi

Matteo Barone

17 giugno 2022

1 Scelte implementative

- La struct Transaction ha un campo in piu che conta il numero di volte che una transazione è stata rifiutata e mandata a un amico. Dopo un certo numero di volte la transazione verrà mandata al master che creerà un nuovo nodo

```
typedef struct _transaction{
  pid_t sender;
  pid_t receiver;
  long timestamp;
  int quantity;
  int reward;
  int n_of_hops;
}Transaction;
```

- Lo spazio allocato in Shared Memory per i nodi è il doppio del numero di nodi. Ciò è necessario al fine di aver abbastanza spazio per creare nuovi nodi durante l'esecuzione

```
,all_conf->SO_NODES_NUM * sizeof(Node) *2);
```

- Il file commonipcs.c contiene tutti i metodi necessari per lavorare sulla memoria comune,i semafori e la coda di messaggi. L'unico metodo non legato ai casi precedenti serve a scegliere un nodo casuale

- Tutti gli ID per l'accesso alla memoria sono in una struct apposta e vengono passate ai Nodi e agli User come argomenti dell'argv

- I semafori vengono utilizzati in tre casi, sincronizzare la partenza dei processi,scrivere nel libro mastro e scrivere nella messageQueue

- Il master non necessità di un segnale per la morte di uno User in quanto ogni informazione sugli user è presente in memoria comune compresa una flag che indica se lo user è in vita

```
ledger = (Ledger*)attachSHM(ids.book_id);
ledger->last_id = 0;
all_nodes = (Node*)attachSHM(ids.nodes_id);
all_users = (User*)attachSHM(ids.users_id);
```

- I metodi notifyDieMasterUser e checkMasterTrans controllano rispettivamente se ci sono User ancora vivi e se cè ancora spazio nel libro mastro. In caso tutti gli User siano morti prematuramente il segnale SIGUSR2 mentre se il numero di blocchi massimo per il libro massimo fosse stato raggiunto viene mandato il segnale SIGUSR1. Ambedue i segnali vengono mandati al master che procede a bloccare la simulazione

- Per inviare periodicamente una transazione a un nodo amico, ogni noto fa partire un timer interno con la funziona alarm() e gestisce il segnale SIGALRM nel signalHandlerNode inviando la transazione e resettando il timer

```
case SIGALRM:
  if(curr_pool_size != 0)
    sendToFriend();
  alarm(config_node->SO_SIM_SEC/2);
  break;
```