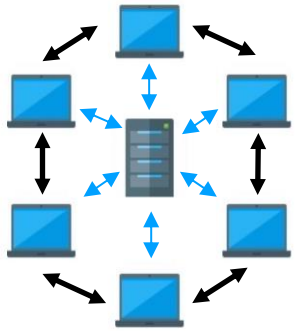
Progetto reti informatiche 2020/2021 - Dario Lauretta

Discovery server

Durante l'avvio, il DS carica dal file *final_entries.txt*, se presente, le entry che erano state salvate precedentemente (tramite esc). Quando si collega il primo peer, il DS gli invia tutte le entry che possiede.
Per gestire la comunicazione con i peer e lo stdin viene utilizzata la funzione select.

Strutture dati Discovery Server

È presente una struttura dati che rappresenta l'insieme dei peer nella rete; in particolare è una lista circolare: cioè ogni elemento punta al successivo, l'ultimo punterà al primo.
I peer vengono identificati dal loro numero di porta e nella coda sono salvati in ordine crescente; di conseguenza ad ogni inserimento di un nuovo peer nella rete, i suoi vicini vengono avvisati e il DS invia i nuovi vicini (cambiati a causa dell'inserimento del nuovo peer).
È inoltre presente una struttura dati, una lista, che viene utilizzata per memorizzare le entry (ricevute dalla add del peer). Non sono presenti duplicati.



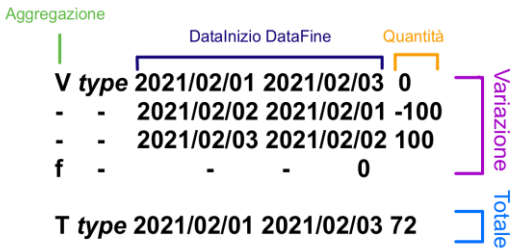
Funzionamento della 'esc'

Il DS contatta tutti i peer con un messaggio "CLOSE", essi rispondono con un messaggio "RESPONSE_CLOSE", chiudono la porta tcp col server e terminano.
Il DS setterà ad 1 eventuali entry aperte e prima di terminare salverà tutte le entry su un file: *final_entries.txt* (con formato uguale a my_entries.txt del peer).

Peer

Inizialmente il peer controlla se sono salvate nel file *my_entries.txt* delle entry da poter caricare in memoria.
Dopo di che il peer effettua la fase di boot con il server utilizzando UDP, dove riceverà i vicini.
Dopo aver chiuso il socket UDP verrà aperto quello TCP dove comunicherà con il server nelle varie fasi. Le comunicazioni con i vicini verranno anch'esse effettuate con TCP.
Anche qui, per gestire la comunicazione con il server, con gli altri peer e con stdin viene utilizzata la funzione select.

File del peer:

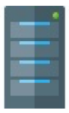


my_aggregazioni.txt

my_entries.txt

my_aggregazioni.txt: file che contiene le aggregazioni effettuate dal peer.
my_entries.txt: file che contiene le entry del peer (e anche quelle degli altri peer).

ADD



4) Salvataggio su struttura dati della entry

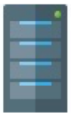
SEND_ENTRY

1) Se l'ora odierna è >= 18, aumentiamo il giorno di 1.

2) Inseriamo la entry (se già esiste aggiorniamo i valori quantità)

3) Invio al server

GET



Requester

VicinoDX

HOW_MANY_ENTRIES

3) Calcola entries nel periodo; se dataFinale è '*' invia anche la data maggiore; se entry non locked, invia -1.

numero_entries

1) Controllare nel file "my_aggregazioni" se presente l'aggregazione.
2) Se non presente, domandare al server numero di entries nel periodo.

4) Calcola le sue e controlla se = al numero ricevuto dal server (se -1 errore).
5) Se sono uguali allora calcoliamo l'aggregazione e la salviamo nel nostro file.
6) Se non sono uguali, chiediamo ai vicini (sx e dx) se hanno l'aggregazione rischiesta con messaggio REQ_DATA. Se rispondono con messaggio SI_DATA riceviamo l'aggregazione, altrimenti NO_DATA.
FLOODING
7) Contatta il vicinoDX ed invia un buffer contenente il periodo e il tipo ed un buffer che serve per concatenare le porte.

8) Se ha entries nel periodo, aggiunge la sua porta nel buffer_conc_porta.
9) Invia i due buffer al suo vicinoDX.

10) Quando i buffer ritornano al Requester (giro completo), esso contatterà tutti i peer presenti nel buffer e si farà inviare le loro entry.
11) Adesso può calcolare l'aggregazione.

CLOSE

Il peer invia tutte le entry che ha in memoria ai suoi vicini e le salva nel file *my_entries.txt* che verranno ricaricate in memoria quando il peer si riattiverà.

Vantaggi e svantaggi dell'implementazione

- Utilizzando una struttura ad anello, la scelta dei vicini e il loro aggiornamento è di semplice implementazione.
- Durante la add il peer invia al DS la entry e questo permette una maggiore flessibilità dei peer, in quanto siamo sicuri che il server avrà sempre tutte le entry; al tempo stesso il server si fa carico di molte trasmissioni e tutto il sistema dipende dal suo corretto funzionamento.
- Dato che il DS mantiene le entry, effettuare il controllo sulle entry durante la get (cioè se il peer le ha tutte) è elementare.
- Alla fine del flooding, quando il buffer ritornerà al peer requester, quest'ultimo contatterà tutti i peer presenti nel buffer e si farà inviare le loro entry; questo metodo non è molto efficiente in quanto più peer potrebbero inviare le stesse entry. Ovviamente non ci saranno duplicati nella lista delle entry, però ci potrebbero essere trasmissioni inutili.