Documentazione scelte progettuali

Politica di integrazione

Un peer che richiede di connettersi al network riceve dal server una coppia di peer già connessi (se presenti) che rappresentano:

1. il peer più “vicino” al peer che si sta connettendo;
2. il neighbor più “lontano” del neighbor del punto 1;

Ho ritenuto che fosse una politica equa in quanto garantisce al nuovo nodo il miglior neighbor disponibile, premia i peer che sono da più tempo nella rete, rimuovendo (nella maggior parte dei casi) il loro neighbor peggiore e, infine, garantisce che nessun peer venga isolato dalla rete.
Il peer che si sta connettendo notificherà in prima persona il proprio arrivo inviando una NEIGHBOR_REQUEST ai neighbor che gli sono stati comunicati dal DS, i quali rimuoveranno dal proprio neighborhood il peer più “lontano” sostituendolo con il peer che li ha contattati. Ho optato per una soluzione che aggiornasse la rete senza l’intervento del DS perchè essendo l’applicazione di natura P2P, ho preferito minimizzare l’infrastruttura del server.
La comunicazione tra un peer e il discovery server avviene, quindi, solo in fase di connessione e disconnessione dal network.

Formato su file

Ho scelto di salvare le entry su file .txt con il seguente formato:

data(dd:mm:yyyy),tipo,quantità

Non ho utilizzato un identificativo e pertanto le entry non sono univoche, ma, poichè i peer utilizzano canali di connessione TCP, non potranno esserci errori di ricezione di pacchetti duplicati.

I dati aggregati vengono salvati su file .txt con il seguente formato:

EventoTipoPeriodo(dd:mm:yyyy-dd:mm:yyyy)=dato_aggregato

Dove *dato_aggregato* vale:

dato_aggregato_variazione = totale_periodo
dato_aggregato_variazione = var_giorno2, var_giorno3 … var_giornoN

Es:

totaleT10:9:2020-12:9:2020=15000
variazioneT10:9:2020-13:9:2020=1000,5530,-5550

Formato dei registers

Vi sono due tipologie di registri: marchiati e non.

- I registri non marchiati contengono entry che sono state aggiunte localmente dal peer con il comando *add*, o al massimo aggiunte in seguito alla disconnessione di uno dei neighbor.
- I registri contrassegnati con il marchio ‘$’ invece, contengono tutte le entry di tutti i peer relative alla data in questione. Pertanto, per il calcolo dei dati aggregati potranno essere utilizzati solo registri marchiati.

Un registro diventa marchiato in seguito all'esecuzione del comando *get* su un periodo che include il register considerato: il peer richiedente accumula localmente le entry con i RES_FLOOD che gli vengono inviati dai vari peer della rete. Nel momento in cui il requester non riceve più RES_FLOOD per un durata pari al timeout allora ha ricevuto tutte le entry che gli servivano dai peer. Al termine del timeout quindi i registri in cui il requester aggiungeva le entry che gli venivano comunicate mano a mano dai peer, saranno registri che contengono tutte le informazioni sulla data in questione e pertanto vengono marchiati.

Protocollo per chiedere entry mancanti

Quando un peer, in seguito a una richiesta *get*, si accorge di non possedere tutti i registri marchiati specificati, fa partire un timer e invia un messaggio ai neighbor:

FLOOD_FOR_ENTRIES-*porta_requeste*r-*lista_registers_mancanti;*

I neighbor che ricevono il flood, estraggono la lista dei registri, salvano in un array *regs_to_send* la porta del mittente del flood e la lista di registri che possiedono e inviano un messaggio TCP RES_FLOOD al peer richiedente. Se un peer possiede la versione $ di un registro, la rimuove dalla richiesta di flood prima di fare il forward, se dopo la rimozione non ci sono più registers allora il peer evita di inoltre il FLOOD.
Il requester che riceve una RES_FLOOD congela il timer, notifica il peer donatore che è pronto a ricevere le entry con un messaggio READY_TO_RECEIVE, a cui il donor risponde con l'invio della dimensione delle entry, e successivamente delle entry vere e proprie. Inoltre visto che i FLOOD che circolano nella rete sono 2, è possibile che un peer riceva due volte lo stesso FLOOD e che possa quindi inviare due RES_FLOOD. Per evitare ciò ho utilizzato una funzione *already_to_serve* che confronta la RES_FLOOD relativa al FLOOD in questione con quelle relative ai FLOOD ricevuti in precedenza e ancora non servite e, in caso di riscontro, evita di inviare la RES_FLOOD e di inoltrare il FLOOD_FOR_ENTRIES .

Comando stop

In seguito all'invio di un comando stop, il peer che sta lasciando la rete contatta i propri neighbors con un messaggio STOP, e delega al secondo dei due neighbor il compito di contattare l'altro per unificare la rete, contattando il primo.

Il primo neighbor riceve: *STOP-0*
Il secondo neighbor riceve: *STOP-porta_primo_neighbor*

Inoltre il secondo neighbor è anche quello che otterrà le entry del peer che sta lasciando la rete. Ho scelto di inviarle a uno solo dei due neighbor perchè altrimenti avrei avuto entry duplicate nella rete, nel caso in cui esse facessero parte di un registro non marchiato. Infine il peer che si disconnette elimina la propria cartella dei registri ma mantiene il file archivio contenente i dati aggregati calcolati perciò, nel momento in cui si connetterà nuovamente alla rete, il peer potrà continuare ad usarlo.