PROGETTO DI RETI INFORMATICHE A.A. 2020-2021

GABRIELE MARINO (604453)

GENERALITÀ SULL'IMPLEMENTAZIONE

Sia il Discovery Server sia il generico peer sono stati implementati come processi multithreaded. In particolare, entrambi i processi presentano un thread per la gestione del terminale e quindi per la gestione dei comandi inseriti dall'utente (*terminalHandler*), e un thread per la gestione della comunicazione di rete (*networkHandler*).

LISTA DEI PEER E IMPLEMENTAZIONE DEL CONCETTO DI NEIGHBOURHOOD

Ogni peer è identificato univocamente dal suo numero di porta. L'insieme dei peer è memorizzato dal Discovery Server tramite una lista ordinata secondo il numero di porta dei peer. Il primo peer della lista è il peer con numero di porta più basso, l'ultimo peer della lista è quello con il numero di porta più alto. I vicini di ciascun peer sono il peer precedente e il peer successivo nella lista. Il primo e l'ultimo peer della lista hanno quindi un solo vicino. I principali vantaggi di questa scelta sono quelli di assicurare che il grafo di neighbourhood dei peer risulti connesso sempre e in ogni caso (a differenza di come potrebbe risultare scegliendo metriche diverse), e di favorire un'estrema facilità implementativa.

GESTIONE DEI COMANDI DI RETE

• (DS) Comando `esc`

Il comando `esc` viene gestito inviando a tutti i peer registrati il messaggio "`SERDOWN`". A seguito della ricezione di questo messaggio i peer terminano la loro esecuzione.

• (Peer) Comando `start`

Il comando `start` viene gestito inviando al DS il messaggio "`BOOTREQ`". Il server risponde inviando al peer appena registrato un messaggio nella forma "`PVS PVD`", dove `PVS` è il numero di porta del vicino sinistro, e `PVD` quello del vicino destro. Il server si preoccupa inoltre di aggiornare i neighbour del peer appena registrato tramite l'invio del messaggio "`UPDTNGB PVS PVD`", dove `PVS` è il numero di porta del nuovo vicino sinistro del peer destinatario, e `PVD` quello del nuovo vicino destro. In seguito alla ricezione di questo messaggio i peer aggiornano le informazioni di contatto dei propri vicini.

• (Peer) Comando `stop`

Il comando `stop` viene gestito preventivamente inviando a uno dei propri vicini tutte le entrate memorizzate dal peer tramite una serie di messaggi nella forma "`ADDENTR anno mese giorno tipo valore`", ciascuno da interpretare nel modo seguente: in data `giorno`/`mese`/`anno`, il peer ha registrato un totale di `valore` dati di tipo `tipo`. In seguito alla ricezione di un messaggio di questo tipo, il peer destinatario memorizza tra i propri dati l'entry così ricevuta, in tal modo evitando la perdita di informazione conseguente allo stop di un peer. Successivamente il peer invia al server il messaggio "`STOPCON`", e quindi termina. In seguito alla ricezione di questo messaggio, il server si preoccupa di aggiornare gli ex-neighour del peer appena disconnesso con un messaggio "`UPDTNGB PVS PVD`", dalla stessa interpretazione di cui sopra.

GESTIONE DELLE RICHIESTE DI AGGREGAZIONE E DEL FLOODING

Al fine del calcolo delle aggregazioni è importante per un peer conoscere il numero totale di dati di un certo tipo registrati su tutta la rete in una certa data. Per chiarezza, indichiamo questo valore con il nome di Risultato Totale Giornaliero (RTG). Nel momento in cui, per il calcolo di un'aggregazione, si ha necessità di conoscere il RTG di una certa data, e nel caso in cui questo non risulti già presente nella memoria del peer, il peer invia un messaggio nella "`REQDATA anno mese giorno tipo`" ai propri vicini, di ovvia

interpretazione. Il peer resta quindi in attesa di messagi di tipo “REPDATA valore”. Il termine valore può anche essere pari a NON_TROVATO, nel caso in cui il risultato desiderato non sia memorizzato neanche dai vicini del peer. Se nessuno dei vicini del peer è a conoscenza del RTG richiesto, il peer inizia un’operazione di flooding. Il flooding avviene inoltrando un messaggio nella forma “FLDFENT porta anno mese giorno tipo” lungo tutta la lista dei peer (il peer invia in generale due messaggi in questa forma, uno al vicino sinistro e uno al vicino destro, che inoltreranno il messaggio rispettivamente a sinistra e a destra nella lista dei peer). In seguito alla ricezione di un messaggio di questo tipo, il peer ricevente risponde eventualmente con un messaggio nella forma “ENTRFND porta_richiedente porta_detentore”, specificando che dal peer di porta porta_detentore sono memorizzate delle entry richieste dal peer di porta porta_richiedente. Questo messaggio ENTRFND è destinato a percorrere, a ritroso, il percorso fatto dal messaggio FLDFENT. Il messaggio FLDFENT viene inoltrato ricorsivamente, un nodo alla volta, fino a raggiungere i peer estremi della lista. Questi, anzichè inoltrare il messaggio FLDFENT, rispondono con l’invio di un messaggio nella forma “ENDLIST porta_richiedente”, destinato anch’essp a percorrere, a ritroso, il percorso fatto dal messaggio FLDFENT, e volto a comunicare al peer di porta porta_richiedente che il flood ha raggiunto l’estremità della lista. Quando il peer che ha iniziato il flood viene a sapere, tramite i messaggi ENDLIST, che l’operazione di flooding si è conclusa, risponde a tutti i messaggi ENTRFND con dei messaggi nella forma “REQENTR anno mese giorno tipo”. I peer che ricevono questo tipo di messaggio rispondono quindi con un messaggio nella forma “REPENTR valore”, dove il campo valore è pari alla somma dei dati del tipo richiesto registrati dal peer nella data richiesta (Totale Giornaliero Individuale). Dopo aver ricevuto tutti i messaggi REPENTR attesi, il peer che ha iniziato il flood può finalmente calcolare il RTG.