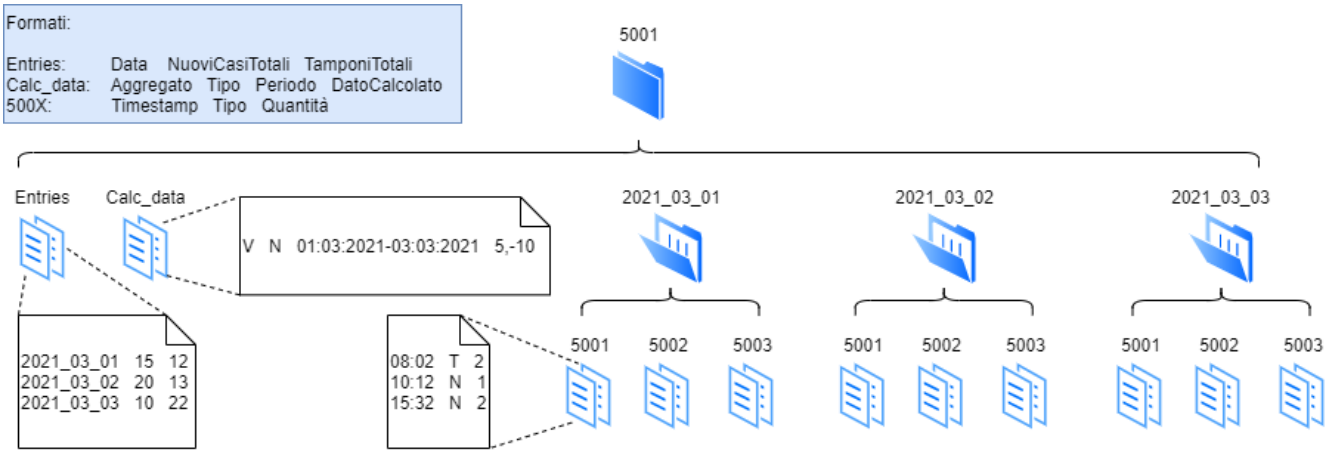
Documentazione Progetto

Tipologia di rete

La rete ha una topologia ad anello: tutti i peer sono collegati ad un peer precedente e ad uno successivo in modo da formare un anello chiuso. Il DS tiene traccia dei peer nella rete mantenendo in memoria una lista circolare bidirezionale. Il tipo di rete ad anello rende semplice l'inserzione e l'eliminazione dei peer nella rete, ma presenta due svantaggi: uno di scalabilità e uno di robustezza. Il primo si ha quando un peer deve richiedere dei dati ad altri peer: la richiesta deve percorrere tutto l'anello, un peer alla volta, facendo aumentare il tempo necessario a soddisfarla proporzionalmente al numero di peer presenti nella rete. Il secondo si ha nel caso in cui un peer si disconnetta dalla rete in modo imprevisto: in questo caso un'eventuale richiesta dei dati da parte di un peer si trova impossibilitata a percorrere l'anello e la richiesta non potrà essere soddisfatta almeno fino a quando il peer disconnesso non riesca a ripristinare la connessione.

Memorizzazione dei dati

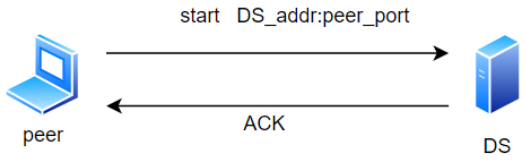
Ogni peer memorizza le *entry* ovvero informazioni in dei *register* ovvero dei file di testo nominati "peer_port.txt" dove *peer_port* è il numero di porta del peer. Ogni riga del *register* contiene una *entry*. Ogni *entry* ha il seguente formato: "*Timestamp* Tipo Quantità", dove *Timestamp* indica un orario nel formato hh:mm, Tipo è un carattere che può essere N (nuovo caso) o T (tampone) e Quantità è un intero positivo che indica il numero di persone a cui la *entry* si riferisce. Ogni *register* è salvato all'interno di una cartella nominata nel formato data "yyyy_mm_dd" in base alla data a cui si riferisce il *register*. Per calcolare un dato aggregato, un peer riceve i *register* necessari dagli altri peer, salvandoli nella cartella che rappresenta la data corrispondente. Quando un peer ha tutte le *entry* di una certa data ne tiene traccia in un file di testo nominato "Entries.txt". Ogni riga del file è nel formato: "data(yyyy_mm_dd) NuoviCasiTotali TamponiTotali" dove NuoviCasiTotali è il numero totale di nuovi casi, TamponiTotali è il numero totale dei tamponi e la data indica a quale giorno si riferiscono i totali sopra citati. Inoltre quando un peer calcola un dato aggregato, questo viene salvato in un file nominato "Calc_data.txt" nel formato "Aggregato Tipo Periodo(dd:mm:yyyy-dd:mm:yyyy) DatoCalcolato" dove Aggregato può essere T se il dato calcolato è un totale o V se è una variazione, Tipo può essere N se è riferito ai nuovi casi o T se è riferito ai tamponi, Periodo indica il periodo al quale si riferisce il dato calcolato e DatoCalcolato può essere un numero rappresentante il totale, oppure, nel caso il dato calcolato sia composto da variazioni, una stringa di numeri separati da virgola rappresentanti le variazioni dei singoli giorni.
Di seguito l'esempio di un peer (5001) collegato ad altri due peer (5002 e 5003) avente tutte le *entry* del periodo 01:03:2021-03:03:2021 che ha già calcolato (e quindi salvato in "Calc_data.txt") la variazione di nuovi casi di tale periodo:



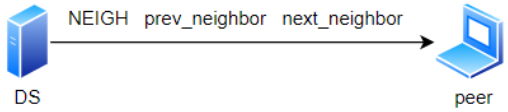
Scambio dei messaggi

start

Il peer invia la richiesta di unirsi al network al DS tramite un messaggio UDP specificando indirizzo IP del DS (DS_addr) e la propria porta (peer_port) finché riceve un ACK:

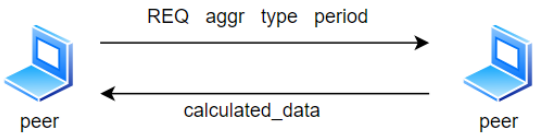


Il DS fornisce il numero di porta dei vicini (prev_neighbor e next_neighbor) al nuovo peer e aggiorna i vicini dei peer adiacenti a quello inserito allo stesso modo.

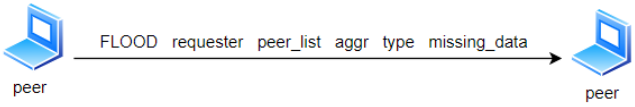


get

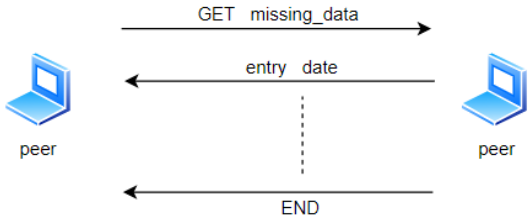
Se un peer non ha le *entry* per calcolare un dato aggregato, invia una richiesta ai vicini specificando il dato aggregato (aggr), il tipo (type) ed il periodo (period) richiesti ricevendo in risposta il dato richiesto (calculated_data):



Se i vicini non hanno il dato richiesto, calculated_data è vuoto e quindi il peer invia un messaggio di *flooding* specificando il proprio numero di porta (requester), il dato aggregato (aggr) ed il tipo (type) richiesti, ed una lista contenente tutte le date di cui mancano le *entry* (missing_data). Il messaggio percorre l'anello ed ogni peer che ha almeno una *entry* mancante aggiungerà il proprio numero di porta nel campo peer_list. Infine, il messaggio giunge di nuovo al *requester*.

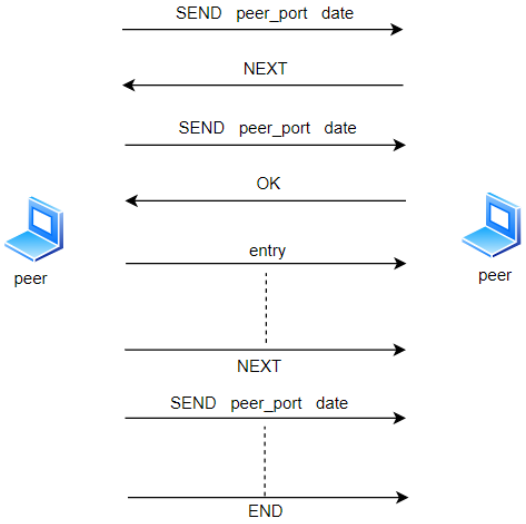


Il *requester* contatta il primo peer della lista (peer_list) inviando la lista delle date di cui non si hanno tutte le *entry* (missing_data) e riceve in risposta tutte le *entry* e la data a cui si riferiscono (date). Una volta ricevute tutte le *entry* riceve END. Poi si contattano allo stesso modo tutti i peer rimanenti nella lista:

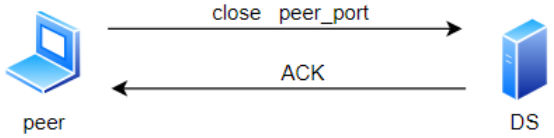


stop

Il peer contatta i vicini inviando il proprio numero di porta (peer_port) e la data (date) del primo *register* da inviare. Il vicino invia in risposta OK se non ha il *register* oppure NEXT se lo ha già. Nel primo caso il peer invia le *entry* del *register* ed invia NEXT una volta che ha finito, nel secondo caso questo passaggio viene saltato. Si ripete il procedimento per tutti i *register* successivi. Finiti i *register*, il peer invia END:



Il peer comunica al DS il proprio numero di porta (peer_port) finché riceve un ACK e termina:



Il DS aggiorna i vicini dei peer adiacenti a quello eliminato come visto in precedenza.

esc

Il DS invia un messaggio di terminazione a tutti i peer e termina: