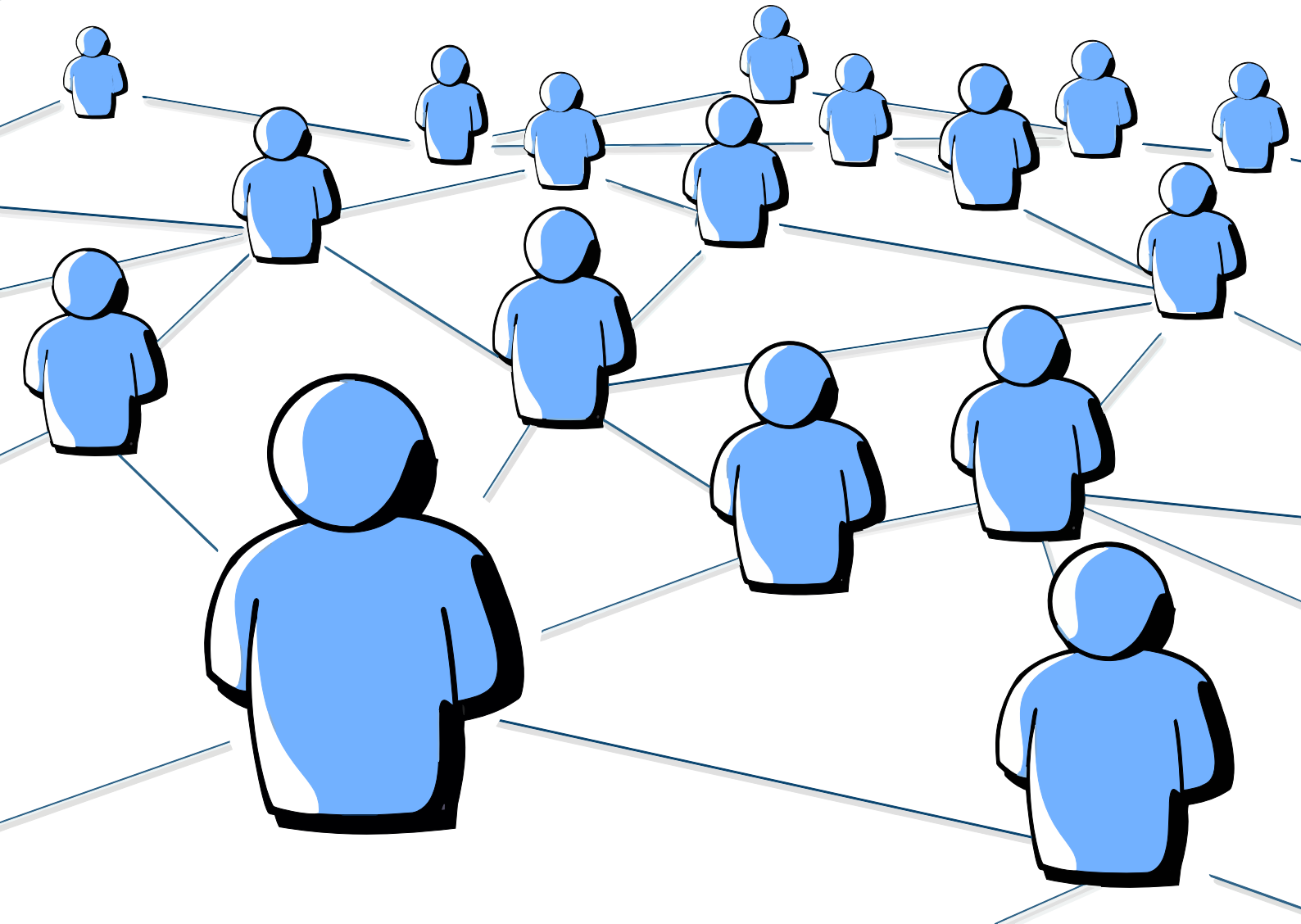
PROGETTO RETI INFORMATICHE 20/21

Gabriele Marino



DOCUMENTAZIONE PROGETTO RETI INFORMATICHE 2020/2021

Questo breve documento fornisce informazioni sulle scelte di progettazione adottate per portare a termine il progettino di Reti Informatiche 2020/2021.

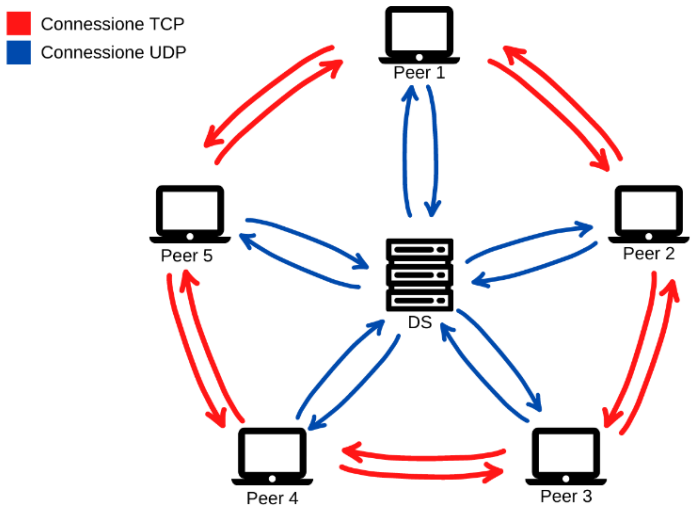


Figura 1: Schema ad anello della rete P2P.

Come mostrato nella figura 1, ho optato per una **struttura ad anello** della rete in cui ogni peer conosce l'indirizzo dei suoi vicini: destro e sinistro. Per vicino sinistro (destro) si intente il peer che sta alla sinistra (destra) di chi guarda lo schema dall'alto.
Come si evince dall'immagine, ho optato per l'uso della connessione TCP per la comunicazione fra peer, ritenendola più idonea allo scopo dell'applicazione sviluppata in quanto si predilige l'affidabilità dei dati scambiati piuttosto che la velocità con cui essi viaggiano nella rete. Peer e DS, come da specifica, usano il protocollo UDP per lo scambio di messaggi.

DISCOVERY SERVER

Il DS ha l'unico scopo di gestire l'indice dei peer che si registrano alla rete, ordinandoli per numero di porta crescente. Il suo schema di funzionamento è **mono processo con socket bloccanti**.
Ogni qual volta che un peer A fa una richiesta di *start,* il DS lo immette nella rete rispondendogli con la porta e l'indirizzo IP dei suoi vicini. Finita questa operazione, è sempre il DS a comunicare ai peer definiti vicini di A, la porta e l'indirizzo IP di A in modo da chiudere l'anello (Richiesta di update).
I formati dei messaggi scambiati fra peer e DS sono:

- **REQ_STR**: inviata dal peer quando esegue il comando *start*
- **REQ_STP**: inviata dal peer quando esegue il comando *stop*
- **REQ_UPD**: inviata dal DS per aggiornare del cambiamento della rete
- **REQ_ESC**: inviata dal DS quando si esegue il comando *esc*

Ogni messaggio ha il suo acknowledgement: ACK_STR, ACK_STP, ACK_UPD, ACK_ESC.
Il DS utilizza un file binario, *lista_peer.bin*, per memorizzare la struttura della rete, e legge e scrive da questo file utilizzando come stampo la seguente struttura dati:

```
struct des_peer
{
    in_port_t   porta;
    uint32_t   IP;
};
```

In fine il DS utilizza il **protocollo binario** per inviare le porte, e **testuale** per le richieste.

PEER

Come per il DS, anche i peer seguono uno schema di funzionamento **mono processo con socket bloccanti**. A differenza del DS, i peer lavorano solo con file testuali quali:

- **reg_GG-MM-AA_PORTA.txt**: file del registro aperto. È sempre unico, perché viene elaborato ogni volta che c'è una chiusura.
- **miei_aggr_PORTA.txt**: file dove vengono compattati i registri chiusi, e contiene i totali registrati dai singoli peer giornalmente.
- **aggr_PORTA.txt**: file dove ogni peer salva gli aggregati che riceve dalla rete, così da elaborarli quando l'utente esegue il comando *get*
- **nome_ultimo_reg_PORTA.txt**: file dove viene salvato il nome dell'ultimo registro aperto.

Il dato aggregato vero e proprio viene scambiato fra i peer usando il **protocollo binario**.
Lo scambio dei messaggi di richiesta viene invece gestito col **protocollo testuale**, inviando prima la dimensione del messaggio e poi il messaggio vero e proprio. I messaggi di richiesta hanno però tutti un format comune:

HEADER GG:MM:AA TIPO TOTALE PORTA

L'Header serve per distinguere la tipologia della richiesta: se si chiede l'aggregato ai vicini o se si inizia una richiesta di flooding (**NGB_REQ**, **FLD_REQ**, **FLD_RPL**, **FLD_WAIT**)

GET

Quando un peer deve eseguire il comando *get,* la prima cosa che fa è verificare se ha gli aggregati richiesti dentro il file *aggr_PORTA.txt*. Per ogni data mancante inizia col chiedere il dato ai vicini (Es. messaggio: <NGB_REQ 1:6:2021 T>). Se i vicini rispondono negativamente (-1), inizia una richiesta di flooding (*figura 2*).

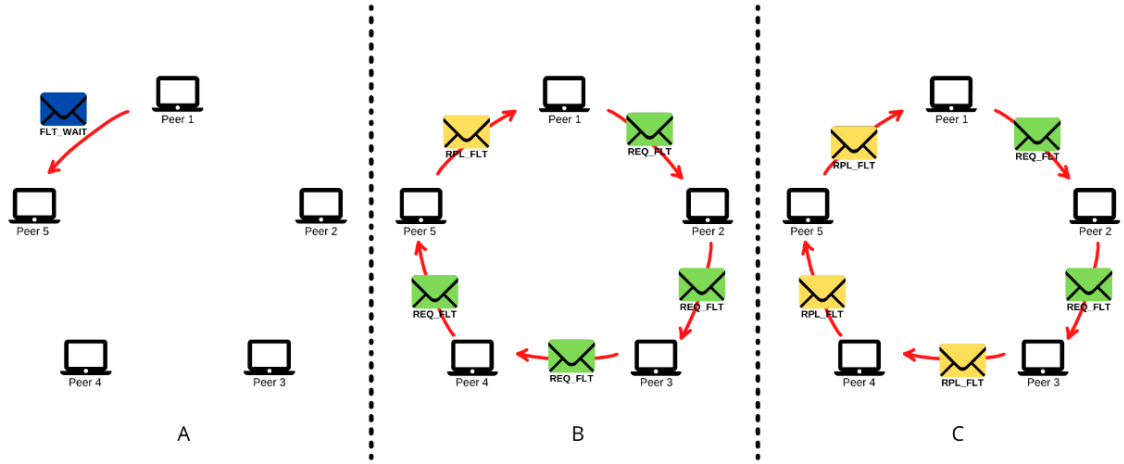


Figura 2: Gestione richiesta di flooding. A) Inizio. B) Caso in cui nessuno ha il dato aggregato. C) Caso in cui peer 3 ha il dato aggregato.

La richiesta di flooding si propaga in un solo verso, quello orario. Il primo passaggio (*figura 2A*) è quello di avvertire il vicino sinistro dell'inizio della richiesta di flooding e quindi aprire una connessione TCP. Ogni peer che riceve un messaggio di tipo FLD_REQ, controlla se per quella data ha il dato aggregato relativo all'intera rete, se non lo ha propaga la richiesta aggiungendo al campo TOTALE del messaggio il proprio totale locale. Se invece il peer ha il dato aggregato propaga un messaggio FLD_RPL che contiene l'aggregato totale. Ogni peer che riceve un messaggio FLD_RPL, prima di propagarlo lo salva nei suoi file se mancante. In questo modo una richiesta di flooding potrebbe giovare a più peer.

CONTRO DELL'APPLICAZIONE

- L'applicazione non predilige la prestazione dell'intero sistema sviluppato, dato che ogni richiesta di flooding si propaga in un solo verso, per un numero considerevole di Host collegati, il ritardo di propagazione per ogni messaggio potrebbe essere elevato.
- Non viene gestita la situazione in cui uno dei componenti si scolleghi forzatamente dalla rete.