Relazione

Livello trasporto

Ho scelto il protocollo TCP perché per un’applicazione di messaggistica è preferibile l’affidabilità al throughput

Livello applicazione

Ogni pacchetto è composto da un header e da un payload

|  |  |
| --- | --- |
| 1B | 4B |
| Tipo del messaggio | Lunghezza del payload |

* Header:
* Payload: cambia struttura a seconda del tipo del messaggio, questa è documentata nel dettaglio in *global.d/network\_tools/network\_tools.h*

Gestione delle richieste lato server

Il server gestisce le richieste in modo **sequenziale** con *select*, ho fatto questa scelta per evitare la complessità che deriva dalla programmazione concorrente e per facilitare la condivisione delle strutture dati comuni a tutte le connessioni.

Per evitare che il server rimanga in idle attendendo un intero messaggio da un device lento, il contenuto del messaggio viene bufferizzato in una struttura globale e utilizzato solo quando si riceve l’ultimo byte.

Gestione della modalità P2P dei device

Anche per la gestione dei messaggi ricevuti in P2P dai device viene utilizzato lo stesso codice che il server usa per gestire le richieste, in questo modo più messaggi possono essere ricevuti contemporaneamente.

Gestione della connessione al server da parte del device

I messaggi da parte del server sono gestiti in due modi diversi:

* Se non abbiamo mandato alcuna richiesta al server viene fatto un controllo grazie a *select* per controllare se ci sono messaggi da ricevere
* Se abbiamo fatto una richiesta al server il device si **blocca** in attesa della risposta

Ho scelto questo comportamento perché non posso sapere a priori quando il server invia una notifica ma se invio una richiesta non posso continuare l’esecuzione finché non ricevo la risposta (o non si verifica un errore)

Se perde la connessione con il server, il device tenta di riconnettersi periodicamente, senza però interrompere le altre attività