Documentazione progetto reti informatiche

L'applicazione utilizza l'IO multiplexing, e socket di tipo TCP non bloccanti. È stato scelto di usare il protocollo TCP in quanto non si necessita di un'eccessiva velocità di trasmissione ma si necessita di affidabilità. Una volta inizializzata la connessione essa rimane attiva per tutta la sessione, per questo l'overhead dell'handshake TCP non fa diminuire di troppo le prestazioni.

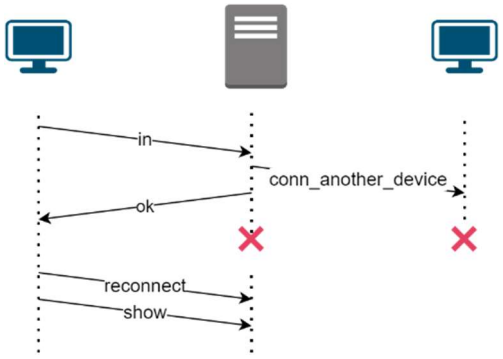
Sia peers che server utilizzano il modulo endpoint.h per interfacciarsi con la rete, il modulo io.h per l'accesso ai file e il modulo utils.h per funzioni di utilità. All'interno del modulo endpoint.h ci sono cinque API: **endpoint** che inizializza il socket di ascolto, gestisce gli input, accetta connessioni, riceve messaggi ed eventualmente manda risposte se richiesto dall'utilizzatore; **connection(port)** che inizializza una connessione TCP con il processo alla porta specificata, **make_request(c, message, need_response)** che invia una richiesta alla connessione specificata, se si necessita di una risposta aspetta che gli venga inviata; **send_file** e **receive_file**.

Ogni volta che si crea una nuova connessione, viene aggiunto un record di tipo **connection_data** ad una lista presente su ogni host. La lista viene dinamicamente aggiornata dalle funzioni **endpoint ()** e **connection()**.

```
struct connection_data{
    int sd;
    int port;
    char username[USERNAME_LENGTH];
    time_t timestamp;
    int logged;
    struct connection_data * next;
} typedef connection_data;
```

Protocollo per la connessione

Il client effettua il login inviando: username, password, porta e vecchio timestamp di logout nel caso in cui il server si fosse disconnesso precedentemente. Il server controlla username e password, se l'utente è online su un altro dispositivo lo disconnette e risponde alla richiesta di login. Se il login va a buon fine il server salva una nuova entry nel file sessions.txt e collega porta e username al socket descriptor che ha effettuato la richiesta. Quando il server si disconnette i peer cercano di riconnettersi ogni volta che gli inviano una richiesta. Durante il logout il client chiude la connessione con il server, esso dal socket descriptor risale all'username connesso e chiude la sessione.



Formato messaggi

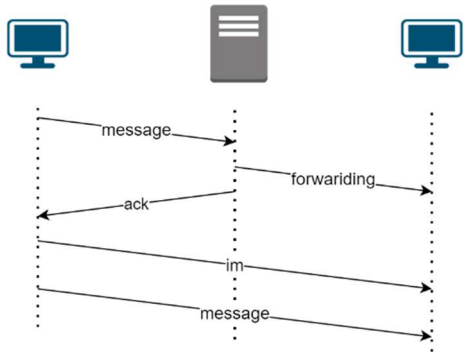
I messaggi inviati sono in formato testo e si suddividono in due gruppi:

- **richieste** che hanno un formato specifico (type|data)
- **risposte** che variano a seconda della richiesta, esse infatti vengono inviate sullo stesso socket e quindi non necessitano di un tipo. I messaggi non hanno una lunghezza predefinita, per questo prima dell'invio effettivo del testo si inviano bit contenenti la dimensione del messaggio in arrivo.

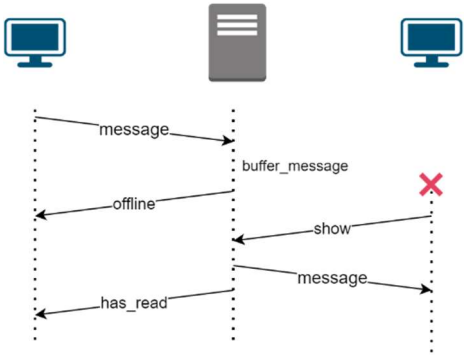
Gestione della chat

Al comando chat si apre una chat singola con un utente. Dopo l'invio di un messaggio, se è il primo di questa sessione all'utente destinatario si invia un messaggio al server che, se online, farà il forwarding dello stesso all'utente richiesto e invierà indietro un ACK contenente la porta a cui è connesso, il mittente invia un messaggio contente le proprie informazioni al ricevitore, altrimenti si limiterà a bufferizzare il messaggio.

Invio messaggio con ricevitore online



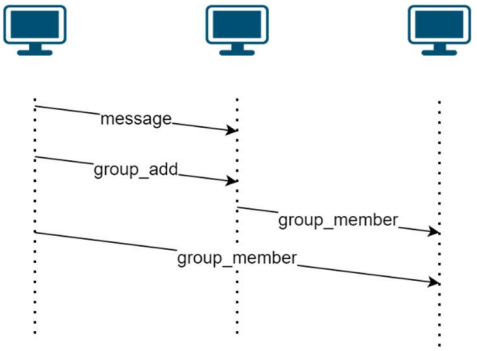
Invio messaggio con ricevitore offline



Gruppi

Colui che aggiunge alla chat un utente invia ai partecipanti alla chat i dati sul nuovo utente da aggiungere, tutti i partecipanti inviano i propri dati al nuovo utente. Se qualche utente facesse parte di un gruppo esso invia una richiesta di uscita con annesso un id del gruppo da cui sta uscendo. Ogni host ha una lista di utenti con cui sta parlando, ogni messaggio viene inviato ad ognuno degli utenti nella lista.

```
struct chat_data{
    char username[USERNAME_LENGTH];
    int online;
    struct chat_data * next;
} typedef chat_data;
```



Protocollo per l'invio dei file

I file possono essere inviati solo se si ha una chat aperta con almeno un utente online e vengono condivisi con tutti i partecipanti ad una chat. Prima dell'invio del file effettivo si invia agli hosts riceventi il nome del file che si vuole inviare, in modo che lo conoscano e che si mettono in attesa di ricezione.