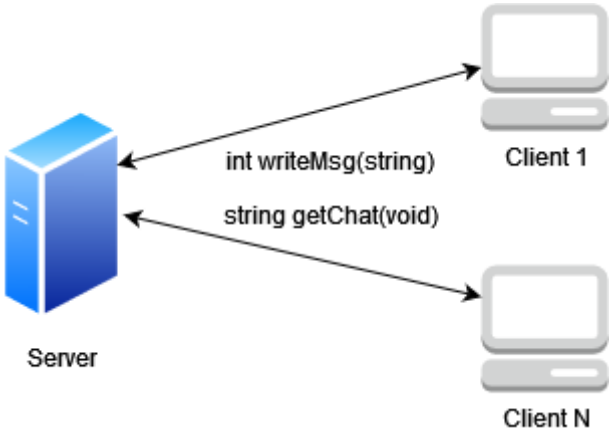
Armand Daussà Baize

RPC-based-chat

Introducció

Per a la realització d'aquesta pràctica es demana implementar una aplicació de client-servidor mitjançant el protocol de RPCs. El sistema consistirà en un servidor que essencialment atendrà dues peticions:



Funcionament

Per a simplificar la implementació de la pràctica, s'utilitza una eina anomenada *rpcgen* que ens permet generar el codi *boilerplate* que necessitarem per a la comunicació entre el client i el servidor.

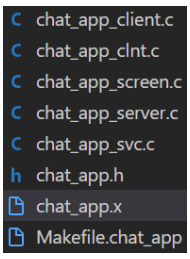
Per començar crearem un fitxer 'chat_app.x' que contindrà les següents línies:

```
program CHAT_APP{
    version VERSION_1 {
        int writeMsg(string)=1;
        string getChat(void)=2;
    } = 1;

}=0x12341111;
```

Aquí, indicarem quin és el nom de programa, la versió del programa, i dins de la versió, indicarem les dues funcions que implementarà el servidor.

Ara que ja tenim establert aquest fitxer, només caldrà executar la comanda 'rpcgen -a chat_app.x' i tot l'esquelete de codi serà generat automàticament.



Ara ja podem editar el client i el servidor al nostre gust per a que funcioni de la manera que desitjem.

Interfície

Per a proporcionar de l'usuari d'una interfície farem servir la llibreria 'ncurses.h' la qual proporciona mètodes que ens permeten pintar pantalles a la terminal. En aquest cas, ens serà molt convenient ja que el que buscarem serà tenir una pantalla per anar agafant el input del usuari i una altra per anar mostrant el chat a temps real.

Pel que fa a la meva implementació, he optat per a definir una secció on l'usuari introduirà els missatges, i separadament, una secció que mostrarà el xat actualitzat en tot moment. Gràcies a que fem servir ncurses podem tenir un thread que s'encarregui de demanar la nova versió del xat cada segon mentre que el thread principal està pendent del que l'usuari escriu per pantalla per a enviar el missatge.

Limitacions

Pel que fa a les limitacions del sistema, podem destacar una d'important. El sistema es comporta de manera que el client està preguntant constantment al servidor per el valor més actualitzat del xat, de manera que aquest (el servidor) està sent bombardejat constantment per peticions les quals augmenten linealment amb la quantitat de clients que estan connectats. En els casos d'us que hem provat per a aquesta pràctica hi havia 3 clients connectats al mateix temps com a màxim, però és evident que a mesura que més clients es connectin el servidor patirà més les conseqüències de rebre tantes peticions i el sistema funcionarà malament.

Solució

És evident que si, per posar un exemple, WhatsApp funciones de la mateixa manera que funciona el nostre xat RPC, ara mateix no estaríem fent servir la App. La solució al problema plantejat implica que de la banda del servidor hi hagi lògica implementada de tal manera que el servidor sigui l'encarregat de notificar el client en el moment de rebre un nou missatge. D'aquesta forma el xat només s'actualitza en el moment de rebre un missatge (com sembla més evident).

Conclusions

Fins ara, mai havíem plantejat que les funcions dels programes que implementàvem residissin en remot. Sempre havíem importat les llibreries que necessitàvem i havíem cridat les funcions localment. RPC ens permet deixar de banda aquesta noció de que tot el codi ha de residir en local i optimitzar espai implementant funcions en remot. Potser la xarxa es converteix en el nostre coll d'ampolla però les funcions remotes poden residir en servidors dedicats amb més potencia que la màquina del client i el temps de comunicació a través de la xarxa no es comparable amb el temps de càlcul que necessitaria un client amb pocs recursos per executar determinades funcions.