Obiettivo del Progetto, è la realizzazione di sistema di chat distribuito che sfrutti la logica del Peer2Peer. A noi studenti è stato lasciato il compito di pensare sia al protocollo di comunicazione che all'implementazione delle nostre logiche. L'intero progetto è stato realizzato in Python, e nell'arco di questo elaborato, discuteremmo sulle specifiche che ci erano state richieste e di come abbiamo cercato di soddisfarle.

Scalabilità:

In vista di garantire una alta scalabilità del progetto, che anche all'aumentare della mole di utenti e **messaggi scambiati** garantisse elevati standard di funzionamento, abbiamo deciso di dividere le comunicazioni su tre porte, garantendo un flusso di messaggi che sia sempre moderato, senza sovrapposizioni e colli d'imbuto causati da comunicazioni che hanno scopi differenti.

Ciò è stato necessario poiché abbiamo scelto di non implementare alcuna memoria centrale per il salvataggio delle informazioni dei vari utenti connessi, e per questo è necessario che i tre oracoli (numero che è sempre aumentabile se non riesce più a soddisfare le esigenze dei peer) siano sempre **in comunicazione tra loro**, scambiandosi le informazioni sulla registrazione e la disconnessione dei vari utenti.

Inoltre, avendo implementato un meccanismo che prevede la comunicazione diretta tra due Peer, non è necessario gestire il routing dei messaggi attraverso la rete. Quando un peer vuole comunicare con un altro, gli basta inserire il suo nickname e verrà inviata **una richiesta di ottenere l'indirizzo associato** a quel nickname.

Per evitare che gli oracoli venissero inondati di questo tipo di richieste (chiamate Query), abbiamo attuato due strategie:

- In primis, ad ogni Peer viene attribuito un oracolo predefinito, così da non avere tutti gli utenti che si rivolgano allo stesso per la registrazione e per le query
- Si è deciso poi di assegnare ad ogni utente un set di vicini (tre per la precisione), a cui si rivolge per richiedere gli indirizzi, così da interrogare l'oracolo solo se tutti i vicini hanno dato responso negativo

Abbiamo quindi schematizzato tre tipo di messaggi possibili:

- **MESSAGGI,** con cui intendiamo tutti i messaggi scambiati tra i Peer
- **QUERY,** che sono le interrogazioni che possono avvenire tra un Peer e i suoi vicini o tra un Peer e gli oracoli
- **COMUNICAZIONI,** che comprendono i messaggi inviati in fase di registrazione, o le informazioni scambiate tra i vari oracoli

Abbiamo quindi provveduto ad assegnare una porta per ciascun canale di comunicazione, creando quindi tre socket separate.

Inoltre, per ridurre ulteriormente la possibilità che un Peer invii una query ad uno degli Oracoli, abbiamo scelto di far si che qualora avvenga una comunicazione tra due Peer, essi si registrino vicendevolmente come vicini, così da poter allargare la rete di conoscenze tra Peer.

Chiaramente questo potrebbe portare Peer che comunicano molto a riempirsi eccessivamente di informazioni rischiando quindi di minare alla scalabilità, proprio per questo uno sviluppo futuro potrebbe

essere limitare il numero di vicini registrabili, magari a 10, tenendo traccia anche di quante informazioni (query o messaggi) circolano tra due vicini, così da poter sostituire quelli meno rilevanti.

Un altro sviluppo futuro che potrebbe ottimizzare anche la gestione del carico degli oracoli, sarebbe un check costante sul numero di comunicazioni Peer-Oracolo, cosicché se un Oracolo si accorga di avere associati troppi Peer che 'comunicano' molto, li possa dirottare su un altro oracolo, attuando una sorta di Load Balancer.

In fase di test, siamo riusciti a mantenere attivi oltre 70 peer senza aver nessun problema di congestione.

Tolleranza ai guasti: La scelta di avere più oracoli che mantengono l'elenco dei peer registrati e i loro indirizzi, permette chiaramente di garantire un'elevata ridondanza, cosicché anche al cadere di uno dei tre oracoli, il registro non venga in alcun modo perso.

Inoltre, in vista di favorire la corretta ricezione dei messaggi, abbiamo implementato una logica di ack nel nostro protocollo: Ogni volta che un peer invia un messaggio, si mette in attesa di un ack di ritorno per un tempo prefissato (2 secondi), se non lo riceve allora comunicherà il Timeout all'utente, che potrà così decidere cosa fare.

Il rischio più grande per un'architettura di questo tipo è quello di vedere gli Oracoli cadere, causando l'impossibilità di registrarsi o ottenere gli indirizzi per le comunicazioni. Proprio in vista di scongiurare questo problema, abbiamo ideato la funzione del Realignment, qualora un oracolo cada, al suo riattivarsi invia una richiesta di riallineamento agli altri due, così da ricevere l'intero registro aggiornato e rimettersi in pari. Il registro viene inviato sotto forma di file Json, così da essere facilmente manipolabile e leggero, abbiamo testato con un registro composto da oltre 70 peer, e non ci sono stati problemi di alcun tipo.

Uno sviluppo futuro sarebbe permettere agli oracoli di comunicare autonomamente ad un Peer di doversi trasformare in oracolo, ricevendo il registro e potendo quindi istantaneamente operare.

Sicurezza e Privacy: Quando si crea un protocollo di comunicazione, chiaramente il focus sulla sicurezza è centrale. Nel nostro progetto abbiamo deciso di implementare la criptazione dei messaggi cosicché se vengano intercettati, non sia comunque possibile leggerli.

Nello specifico, abbiamo implementato la **Crittografia RSA,** quando un Peer si registra, genera le proprie chiavi private e pubbliche, quest'ultima viene poi comunicata all'oracolo in fase di registrazione.

Ogni volta che un Peer deve inviare un messaggio, quando richiede l'indirizzo, gli viene fornita anche la chiave pubblica del destinatario, così da poter criptare il messaggio.

Il destinatario potrà allora decriptare il messaggio con la propria chiave privata, e per inviare l'ack di risposta dovrà ottenere la chiave pubblica del mittente originale, qualora non riuscisse a riceverla per qualche motivo, il messaggio non verrà mostrato ma verrà bollato come non sicuro, in quanto interrogando i vicini e gli oracoli, non si è riusciti a identificare il mittente.

Uno sviluppo futuro potrebbe essere implementare un meccanismo di Autenticazione dei Peer, questo non era chiaramente possibile attualmente, in quanto la sessione viene ripristinata una volta che gli oracoli vengono spenti, senza alcun salvataggio dei Peer precedentemente connessi.