Design and Analysis of Multi-Client-Server Messaging System

University of Melbourne

-:May 2018:-
Anubhav Singh <anubhavs@student.unimelb.edu.au>
Ashwin Sajiv <apurushotha@student.unimelb.edu.au>
Sahitya Beru <sberi@student.unimelb.edu.au>
Uttakarsh Tikku <utikku@student.unimelb.edu.au>

1 Introduction

The objective of the project was to build upon the phase-1 of the multi-server system and make the system more robust in presence of failures including client and server crashes, and transient network failure between nodes. The server hierarchy, network model, and protocols were upgraded to improve availability and consistency of the system. Furthermore, the system now allows clients to join and leave, and server to join at any point in time. It guarantees receipt of activity message by all clients that were connected at the time when the message was sent and relative ordering of these messages is maintained when received by other clients. The system continues to maintain requirements that a user can register only once in a system and that user-load is balanced between the servers, albeit in a different manner than phase-1.

The report explains the design of the system including message protocols, network model, and server hierarchy, and provides justification for the chosen models. It also includes analysis of system behavior in case of failures.

2 System Design

The system was designed with Super-peer model at its heart. A super-peer network contains elements of both pure peer-to-peer and centralized model. Allowing for peer-to-peer elements such as autonomy and availability, as well as, benefits of centralized approach in tasks requiring consistency and optimized querying servers for information. In this model, the super-peer acts as a centralized server for their peers, whereas the connection model between the super-peers is that of a peer-to-peer. The super-peers are connected in a mesh network and each normal-peer connect to one of the super-peers.

NOTE: From the client perspective, any server whether peer, super-peer or master-super-peer, provide same functionalities. The clients are able to login to any server but multiple logins are not allowed.

2.1 *Network Design*

As depicted in *Fig. 1*, the super-peers are connected in a mesh network. This allows for added resiliency and reduces the overhead of messages-forwarding in comparison to the tree model in phase-1. Also, given the relatively smaller number of super-peers in the overall system, the cost overhead of mesh-network is less. The super-peer acts as a centralized server for peers and peers only maintain the network connection to one super peer at a time.

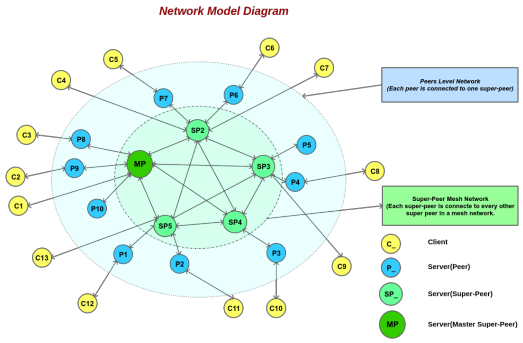

Figure 1: Network Model

2.1.1 *Super-peer*

A super-peer acts as a source of truth for login validation, peer-load information which it sends to its peers via SUPERVISOR_ANNOUNCE. This essentially reduces the message complexity in comparison to model in phase-1 by removing the need to broadcast. The super-peers also collectively ensure that the pending activity messages for a client are delivered whenever the client login again, essentially providing a guarantee that a client receives all activity message which was sent when it was connected to the system.

2.1.2 *Peer*

A peer is only connected to one super-peer at a time and majorly depends on its super-peer for activity broadcasts, registration and login, however, it independently manages login-redirects.

2.1.3 *Master-Super-Peer*

The master-super-peer server acts like a normal super-peer except for functionalities of new user registration and managing number of super-peers in the system.

2.2 *Server states and Transition*

The servers, at any given time, could be in the normal state or transition state. In normal state, a server can be a PEER, SUPERPEER or MASTERSUPERPEER, whereas PEERTOSUPER is the transition state.

MASTERSUPERPEER- This state is set to the very first server that connects to the system or the server that succeeds if the master-super-peer crashes.

SUPERPEER- A server is set to super-peer when the number of super-peer in the system is less than the threshold or in response to the super-peer crash.

PEERTOSUPERPEER- When a peer is given the message, RAISE_TO_SUPERPEER, by the super-peer, it is set to this state until it receives a PEER_RAISED message. During this transition period, the peer gathers all the information required to become a super-peer.

PEER- Every server other than the first server, will have its server state set to this state. If a transition is required, it is changed to one of the other states mentioned above.

2.3 *Functionalities and Protocols*

The system has messaged protocols associated with each of the functionalities. This section discusses the system functionalities and the corresponding protocols.

2.3.1 *Addition of a new server*

When a server starts, it checks if a hostname or port of some other server is given to it. If it does not find any, it takes up the role of master-super-peer. When another server tries to connect, it is given the hostname and port of the first server and sends an AUTHENTICATE message to it. If the secret does not match, an AUTHENTICATION_FAIL message is sent to it. Otherwise, no other message is sent and server two connects to server one successfully. The master-super-peer sends

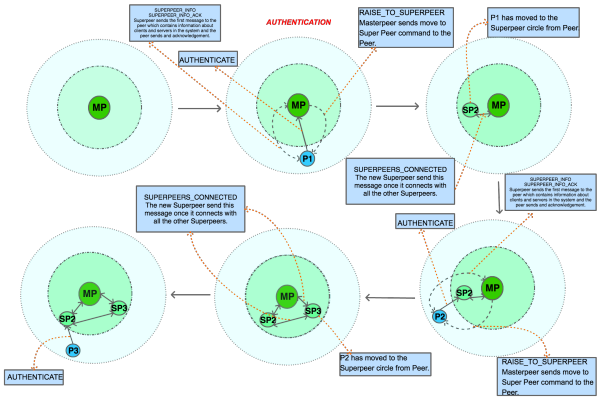

Figure 2: Authentication Flow

RAISE_TO_SUPERPEER to the newly connected peer to make it a super-peer as the number of super-peers in the system is less than 4 including the masters-super-peer. The super-peer that the new peer is connected to (the supervisor), sends RAISING_PEER to every other super-peer in the system with the ID of the new peer. A SUPERPEER_INFO message is sent from that super-peer to the peer in transition which holds the information about all the clients and other super-peers. The peer now sends a SUPERPEER_INFO_ACK to acknowledge the SUPERPEER_INFO message and the super-peer adds it to the super-peers list. Finally, the peer is raised to super-peer and its previous supervisor sends a PEER_RAISED to the peers information. Refer *Fig. 2.*

2.3.2 *Registering a new client*

A client can connect to any of the servers in the system including peers, super-peers, and master-super-peer. Firstly, the client sends a REGISTER message to one of the servers and then waits for a reply back. The server in receipt of the request checks if the user already exists in its local database. If yes, it replies with REGISTER_ATTEMPT_FAILED, else if the server was a normal-peer it forwards the request as REGISTER_REQUEST to its super-peer. If the super-peer also doesnt have the user in its local database. The request is forwarded to the master-super peer (which acts as the master for registration requests) and either replies with REGISTER_FAILED if the user was in the database or broadcasts REGISTER_SUCCESS to all the super-peers. The message flow is same when a client connects to a super-peer, however, in case the client directly connects to master-super-peer, it directly accepts or rejects

the registration request and broadcasts REGISTER_SUCCESS to other super-peers in case of success. Refer *Fig. 3*.

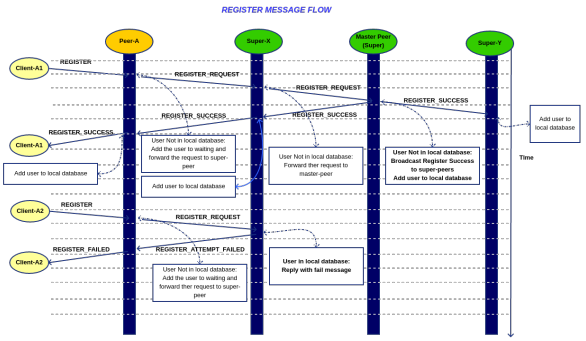

Figure 3: Register Message Flow

2.3.3 *Client Login*

A client can attempt to login at any of the servers with LOGIN message. If the server is a super-peer or master-super-peer and has the user in its database, it validates the user secret and in case of match replies with LOGIN_SUCCESS. After sending login success it broadcasts NEW_LOGIN to other super-peers. Otherwise, if the user is not present or the secret does not match it sends LOGIN_FAILED to the client.

If the server is a peer, it validates the user with its local database and if the user credentials are invalid it replies with LOGIN_FAILED. However, if the user is not in its database or user was successfully verified, it sends a LOGIN_ATTEMPT to its super-peer to validate the user and check for multiple logins. The super-peer performs validation and replies with LOGIN_ATTEMPT_FAILED or LOGIN_ATTEMPT_SUCCESS. In case of LOGIN_ATTEMPT_SUCCESS, it broadcasts NEW_LOGIN to other super-peers.

NEW_LOGIN is used as a mechanism to intimate other super-peers of a newly connected client, and also prevents multiple logins in case user attempts to connect from two different clients with same username and secret at the same time, as shown in *Fig. 4*.

2.3.4 *Client Logout*

The client sends LOGOUT message when it wants to disconnect. The server closes the connection and in case its a peer sends out a USER_DISCONNECTED to its super-peer. The super-peer broadcast the message to all

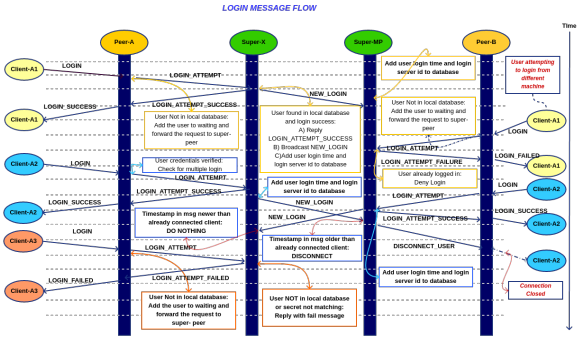

Figure 4: Login Message Flow

other super-peers. USER_DISCONNECTED message is used to communicate logout status and timestamp.

2.3.5 *Redirection*

Redirection can be done for a client as well as a server in this model. Each server has a threshold for connecting servers and clients. If a new connection comes in then the load of the server is compared with the thresholds. If the load is less than the threshold, then the new connection is not redirected. If the load is higher than the minimum threshold and lower than the maximum threshold, the probability for redirection of the new connection is $p = 1 - \{max\_threshold - load\}/\{max\_threshold - min\_threshold\}$. The probability for redirection rises linearly with the increase in load until the maximum threshold is reached and then the probability of redirection becomes 1. Also, the redirection occurs only if eligible servers are available. If no other servers are eligible for redirection, then the connection is not redirected.

2.3.6 *Server Announce*

SUPERPEER_ANNOUNCE: This message is sent between the super-peers and holds the information of load which is shared to every other super-peer in the mesh. The announce message sent by the master is different to that of the other super-peers as it holds the information about the succeeding super-peer if the master crashes.

SUPERVISOR_ANNOUNCE: This message is sent from a super-peer to its connected peers. It holds information about the clients under it and the load which is useful for redirection. It also holds the information of the super-peer that succeeds it in case the super-peer crashes.

SERVER_ANNOUNCE: Sent from a Peer to itss supervisor. Holds information about the

load.

3 Analysis of system under failure models

The system remains available to the clients as long as one of the super-peer remains alive in the system. The impact of failure scenarios has been summarized below.

3.1 *Master super-peer failure*

The master-super-peer randomly selects two super-peers as backup master-super-peers and includes the information in super-peer announce message. In case of master-super-peer failure, the first backup-super-peer immediately takes over as master-super-peer. The other super-peer direct all the request to the backup super-peer. The registration messages which are to be processed by master-super-peer would get impacted, however, for a very short duration. The few register messages which were under processing on Master would not be processed and registration would not complete. In this scenario, the client would need to request for registration again. Since the window of recovery is very small, the impact is construed to be low. The recovery of connected peers and clients remains the same as super-peer failure.

3.2 *Super-peer Failure*

In case a super-peer fails, all its client and peers are disconnected. The peers have a list of backup super-peers and would immediately establish the connection it. During this phase, all the clients remain connected to their peers and the system resume normal operations upon connection to backup.

3.3 *Peer failure*

All the connected-clients of the failed peer gets dropped and it is assumed that they would reconnect to other servers. Its super-peers already have the list of clients which were connected to it, and process the changes as a crash of all the clients. When any of those clients reconnect, it receives the pending messages from the server.

3.4 *Client crash*

The client crash usually does not impact the system except for the scenario where there is a pending message for the client in the system. In this case, the user receives the message when it reconnects. To enable this, each super-peer determines whether there are pending messages for a newly logged in client and then includes it in LOGIN_ATTEMPT_SUCCESS message sent to the peer.

4 Conclusions

The updated system provides extensive improvements over phase -1 of the project. It reduces the message complexity greatly by utilizing the super-peer topology in the network. It offers superior availability to the clients by utilizing the backup super-peers and backup master-super-peers. Therefore, every node in the network is considered to be dispensable. The problem of locking while registering a new user is resolved by using the master-super-peer as the single source of truth and replication between super-peers ensures backup super-peer can immediately replace super-peer in case of a crash. The efficiency of load balancing is improved by the use of a probabilistic model instead of static thresholds. Since pending messages of a client are always delivered after it reconnects, it guarantees that an activity message would eventually be received by all clients connected at the time when it was generated. Message ordering is also maintained.