

I want to elaborate my protocol a little bit more. Our protocol is different from others in the sense that:

I) we are using directory service for server discovery against the common notion of multicast address

II) We are providing fault transparency to user.

Following assumptions are being made in discussion below:

We compared our project to real world and tried to figure out where our protocol will be useful.

i) For a single distributed Chat application, we will have limited number of chat servers like Yahoo.India, Yahoo.US etc..

ii) That is we don't see thousands of chat servers in a single distributed application

iii) This implies in real application, after initially a network has been set then new chat servers will not be entering into the network every minute or so, i.e., after initialization it will become stable network and chat servers will be added slowly and after sometime.

iv) Recovery time of any system in our application will be finite either manually or automatically.

Details about I)

Anyone who sees our protocol will instantly think that we have single point failure in our protocol. But they didn't notice that we use directory service for ONLY entering/exiting the network. Hence when directory service fails then no new chat server will be able to add (or remove) but all the current chat servers interactions will not be hampered. It is like till the time primary chat server recovers, no new user agent will be able to connect to that chat server because user agents use IP address and port to connect to chat server and new user agent has no neighbor information. Hence Directory Service is like DYNAMIC NETWORK CONFIGURATION FILE used at network initialization.

OR

we can always provide backup for that machine.

Details about II)

i) When backup servers take user agent of primary servers then it will not change the seed of primary server's chat rooms when end user enters or leaves a chat room. In this manner, while the primary server recovers, seeds are frozen.

ii)

Freezing seeds enable B.A.a and C.A.b (which were participating in same chat room r1) to still talk to each other even if new user agents enter or leave that chat room. Therefore there are two copies of a single chat room at B and C and their information about A's chatrooms will be inconsistent but end user will not be able to know. But if some user leaves room B.A.R2 and

enters B.A.R1 then only B.A.* nodes will be able to see this user while C.A.* in C.A.R1 will not see this user but still receive this message on their chat room. So if needed, they can add any such user's info (with matchin seed) to A's chat room info on C.

iii) Also note that since all the user agents that went to same backup server will see the consistent chat room information of A.

iv) And most important to see that little inconsistent data in ii) will be there only for the recovery time of A.

v) Note if B.A.a create a new chat room r10 then it will be created by B and multicast address and seed will be provided and handled by B under database of A. This chat room will not be visible to C.A.* but then they never know if r10 exists or not.

I believe this is better than user suddenly gets logged out and sees new users and chatrooms and when A recovers then they get logged out again.