International Journal of Modern Trends in Engineering and Research
www.ijmter.com
e-ISSN No.:2349-9745, Date: 2-4 July, 2015

Scalable Server Architecture for Presence Cloud

Vaishali B. Bondre[1], Varsha N. Wahane[2]
[1] *Department of Information Tech, Terna College of Engg, Navi Mumbai, vaishu.bondre@gmail.com*
[2] *Department of Information Tech, Terna College of Engg, Navi Mumbai, varshasim@gmail.com*

Abstract— Mobile presence service is crucial component of a social network application because it maintains each mobile user's presence information, such as the online/offline, GPS location and network address, and also updates the user's online friends with the information continually. If presence updates occur many times then huge number of messages allocated by presence servers occur scalability problem. So, to deal with this efficient scalable server support called as presence cloud. Presence Cloud arranges presence servers depend on the concept of grid quorum system to fix efficient searching approach. It also manages a directed buddy search and a one-hop caching approach to reach small constant search latency.

Keywords- Social networks, mobile presence services, Presence Cloud, scalable server.

I. INTRODUCTION

Use of Presence-enabled applications is growing with the increasing use of internet. Mobile devices and cloud computing environment provides these applications such as social network applications/services, worldwide. The examples of presence enabled applications are Twitter [1], Facebook [2], Google Latitude [3], and Mobile Instant Messaging (MIM) [4] .These applications have grown rapidly in the last decade. Various new applications are being developed to provide social network services to engage participants with their friends on the Internet. They share the information about the status of participants. So, because of the large use of mobile devices such as Smart phones that utilizes wireless mobile network technologies. Participants can share their live experiences instantly across great distances through social network services. Advantages of mobile devices are increasing day by day will become more powerful. Maintain user information of till today's date, list of presence information is generated and is main functionality of all mobile presence service. The presence information consists of the details about a mobile location of user, user's availability, capability of device, activity. The service should combine the user's ID along with presence information. It can also retrieve and subscribe to modifications in the presence information of the user's friends. In social network services, every mobile user has a list of friends, which is called as buddy list, which includes the contact information of other users that user wants to communicate with. When user switch from one state to another mobile user's status is broadcasted to other user's in its buddy list. Most presence services use server cluster technology [4] to maximize a mobile presence service's search speed and minimize the notification time. Social network services are used by more than 500 million people on the Internet.

Server-to-server architecture [5] is proposed to improve scalability and efficiency of mobile services called Mobile Presence Cloud. First, the server architectures of existing presence services are examined, and introduced the buddy-list search problem in distributed presence architectures in large-scale geographically data centres. When distributed presence server is overloaded with buddy search message at that time scalability problem occurs and it called as buddy-list search problem. Basic building block for mobile presence service is scalable server-to-server architecture. Service-wide global information about all users should not maintain by single presence server in order to

avoid single point of failure. Large-scaled social network service is supported by Presence Cloud. The contribution of this paper consists of three things: First, Presence Cloud is among the architecture for mobile presence services, and it outperform based on distributed hash tables. Secondly, new problem called the buddy-list search problem is defined and scalability problem of distributed presence server architecture.

II. RELEATED WORK

Here we describe the survey on presence service of existing systems. Most IM systems use centralized clusters to provide presence services. Instant messaging (IM) is a type of communications service that enables you to create a kind of private chat room with another individual in order to communicate in real time over the Internet. The authors provided an overview of the system architectures and observed that the systems use client-server-based architectures [4]. Most popular network IM system AOL, YMSG, MSN is discussed.

AIM uses client-server architecture for normal operations but uses a peer-to-peer approach for voice-chat sessions. YMSG also uses client-server architecture for normal operations as well as voice-chat service. YMSG voice traffic is routed through a centralized voice-chat server.MSN also uses a client-server architecture for normal operations and peer-to-peer for voice-chat communication. Most IM systems have mechanisms for maintaining lists of friends. These are typically called buddy lists. Recently, presence services are also integrated into mobile services. For example, 3GPP has defined the integration of presence service into its specification in UMTS. It is based on SIP [6]-[13] protocol, and uses SIMPLE [7] to manage presence information. Recently, some mobile devices also support mobile presence services. For example, the Instant Messaging and Presence Services was developed by the Wireless Village consortium and was united into Open Mobile Alliance (OMA) IMPS [8] in 2005.

The server scalability and distributed management issues in IMS-based presence service. Recently, the IETF [9]-[10] has begun to standardize IM and chat protocols. Two competing standards are being developed: one based on SIMPLE and a second one based on XMPP [11].

Skype, a popular voice over IP application, utilizes the Global Index (GI) technology [12] to provide presence service for users. GI is a multitier network architecture where each node maintains full knowledge of all available users.

All three commercial systems use server clusters for scalability which occur huge number of problem at presence server. So, to deal with this efficient scalable server support called as presence cloud [5]. Presence Cloud arranges presence servers depend on the concept of grid quorum system to fix efficient searching approach. It also manages a directed buddy search and a one-hop caching approach to reach small constant search latency. Overall, Presence Cloud is shown to be a scalable mobile presence service in large-scale social network services.

III. PROBLEM STATEMENT

Here, actually there are several major problems are occurred. In that mainly concentrate on buddy-list search problem. This problem is take place when huge numbers of messages are occurred continuously. By cause of this search problem, time for passing of messages is very slow i.e., time is delayed to reach particular message to the destination. The reason for occurring this type of problem is 'overloaded messages'. i.e., by cause of overloaded messages this buddy-list search problem is occurred. It is sometimes called as 'scalability problem'. Search cost is also called as 'communication cost'. When a user arrives, the total number of messages produced by the presence

server is nothing but search cost. Search satisfaction is nothing but, time it takes to search the user's arriving buddy list. When the buddy-list search problem is occurred, at that situation there is a chance to delay the message passing. By this cause time is delayed. At the same time, there is need to store that message temporarily up to reach to destination. By this cause, here require extra space to store that message in temporary memory. i.e., like cache memory. This is also one of the major problems in already existing system.

IV. PRESENCE CLOUD ARCHITECTURE

Presence Cloud is used to construct scalable server architecture for mobile presence services, and can be used to efficiently search the buddy lists. A simple overview of Presence Cloud in Figure 1. In the mobile Internet, a mobile user can access the Internet and make a data connection to Presence Cloud via 3G or Wi-Fi services. After the mobile user joins and authenticates to the mobile presence service, the mobile user is connected to one of Presence Servers in the Presence Cloud. The mobile user opens a TCP connection to the Presence Server for presence information. After the control channel is established, the mobile user sends a request to the connected PS node for his/her buddy list searching. Our Presence Cloud shall do an efficient searching operation and return the presence information of the desired buddies to the mobile user.

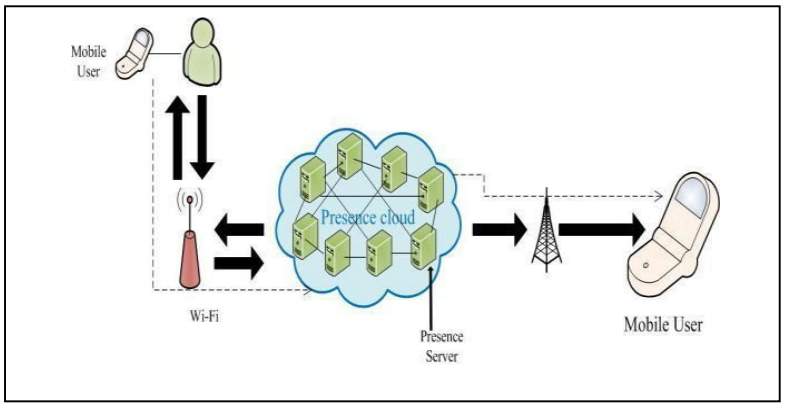

Figure 1. Overview of Presence Cloud

IV. DESIGN OF PRESENCECLOUD

A new design of mobile presence services is needed to solve the buddy list search problem, mostly for the market of mobile social network applications. Presence Cloud is used to form and support a server to server architecture for buddy list searches. Presence Cloud includes three main parts as follows:

5.1. Presence Cloud Server Overlay

Presence Cloud server overlay construction algorithm organizes the PS nodes into a server-to-server overlay, which provides a good low-diameter overlay property. The low diameter property ensures that a PS node only needs two hops to reach any other PS nodes.By using grid quorum system presence server list is built and this presence server list maintains presence server node which has a set of presence server nodes. For example in Figure 2, grid quorum is set to $\sqrt{9}*\sqrt{9}$. Presence server node a8 has a presence server list {a2, a5, a7, a9} and presence server node a1 has presence server list {a2, a3, a4, a7}. Thus presence server node a8 and a1 can built their overlay network according to their presence server list.

a1	**a2**	a3
a4	**a5**	a6
a7	**a8**	**a9**

a1	**a2**	**a3**
a4	a5	a6
a7	a8	a9

Figure 2. Presence Cloud server overlay

Algorithm 1:

Presence Cloud Stabilization Algorithm Basically use for periodic verification of PS node in Plist

Input: Set of the current PSlist of PS node

Output: Establish connection with PS node

for each node **do**

for each document in the corpus C **do**

Check for value of current node and node in plist

if no match found **then** refresh Plist entries

Again check correct node

if returns correct node then establish connection

else nil then pick random node from same column or row of failed node and establish connection

Send heartbeat message

If no connectivity between current node and node in plist i.e node n **then** n will pick random node from same column or row of failed node and establish connection

return connection of PS node with node in PSlist

5.2. One-Hop Caching Strategy

To improve the efficiency of the search operation, Presence Cloud requires a caching strategy to replicate presence information of users. In order to adapt to changes in the presence of users, the caching strategy should be asynchronous and not require expensive mechanisms for distributed agreement. In Presence Cloud, each PS node maintains a user list of presence information of the attached users, and it is responsible for caching the user list of each node in its PS list, in other words, PS nodes only replicate the user list at most one hop away from itself. The cache is updated when neighbors establish connections to it, and periodically updated with its neighbors. Therefore, when a PS node receives a query, it can respond not only with matches from its own user list, but also provide matches from its caches that are the user lists offered by all of its neighbors.

5.3 Directed Buddy Search

We contend that minimizing searching response time is important to mobile presence services. Thus, the buddy list searching algorithm of Presence Cloud coupled with the two-hop overlay and one-hop caching strategy ensures that Presence Cloud can typically provide swift responses for a large number of mobile users. First, by organizing PS nodes in a server-to-server overlay network, we can therefore use one-hop search exactly for queries and thus reduce the network traffic without significant impact on the search results. Second, by capitalizing the one-hop caching that maintains

the user lists of its neighbors, we improve response time by increasing the chances of finding buddies. Clearly, this mechanism both reduces the network traffic and response time. Based on the mechanism, the population of mobile users can be retrieved by a broadcasting operation in any PS node in the mobile presence service. Moreover, the broadcasting message can be piggybacked in a buddy search message for saving the cost broadcasting message can be piggybacked in a buddy search message for saving the cost.

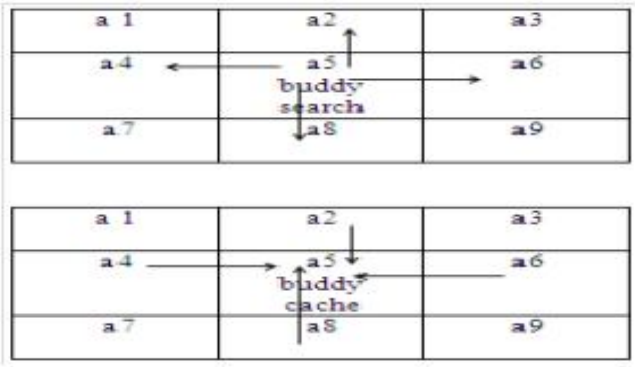

Figure 3. Presence Cloud server overlay

Figure 3. Shows, for mobile presence services it is important to reduce search time. Using two hop overlay and one hop caching strategy presence cloud endow response for large number of mobile users. One hop search used for queries in order to reduce network traffic one hop caching maintains user list of its neighbours to enhance response time by increasing in finding buddies.

Algorithm 2:

Directed Buddy Search Algorithm retrieves presence information of the queried buddy list at one hope

Input: list of identifiers

Output: Set of identifiers belongs to same grid

for user login in Presence Cloud

For PS Node q of that user

Send buddy list search message

If message received then buddy present at one-hop will respond further quires and remove responded buddies

If set of identifiers of user list is nil **then** the buddy list operation is done

Else each remaining identifier is hash to obtain grid ID then PS node q of that user is aggregated with set pf buddies that shares same grid ID

return new identifier after aggregation send to list of identifiers.

V. PERFORMANCE METRICS

There are three metrics which computes the performance of server architecture shown in Figure 4. First, total number of messages transferred between query creator and the presence server in presence cloud is the total search message. Second, average searching messages per arrived user is individual of user arrival prototype. Third, it searches for buddies when mobile user joins the network is average search latency.

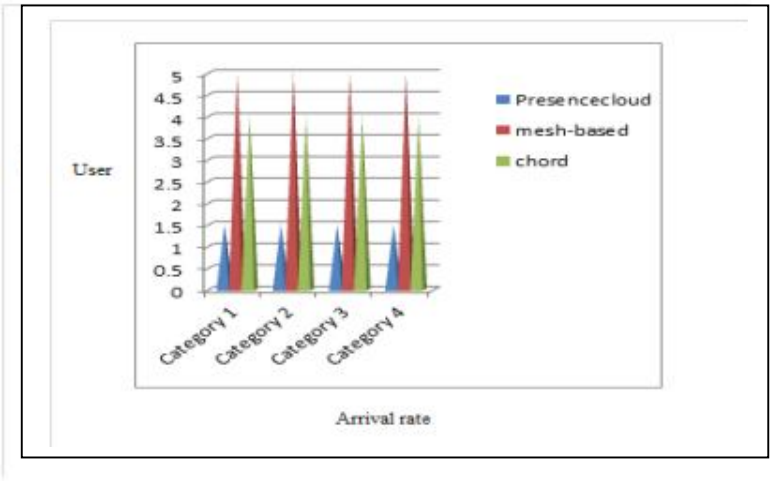

Figure 4. Performance of Presence Cloud

CONCLUSION

In large scale social network services mobile presence services is supported by the scalable server architecture called as presence cloud. In this paper, we discussed the scalability problem in server architecture designs, and introduced the buddy-list search problem, which is a scalability problem in the distributed server architecture of mobile presence services. Presence cloud, that supports the mobile presence services even in large-scale social networks. Presence cloud is used to improve the performance in searching operation. Buddy-list search problem is overcome.

REFERENCES

[1] Twitter, http://twitter.com
[2] Facebook, http://www.facebook.com
[3] Google latitude, http://www.google.com/intl/enus/latitude/intro.html.
[4] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, Z.-Y. Shae, and C. Waters, "A study of internet instant messaging and chat protocols," *IEEE Network*, 2006.
[5] Chi-Jen Wu, Jan-Ming Ho, Member, IEEE, and Ming-Syan Chen, Fellow, IEEE on "A Scalable Server Architecture for Mobile Presence Services in Social Network Applications", 2013.
[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Sip: Session initiation protocol," *RFC 3261*, 2002.
[7] SIP/SIMPLE," IETF Internet draft, 2009. B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, andD. Gurle,"Session initiation protocol (sip) extension for instant messaging," *RFC 3428*, 2002.
[8] Open Mobile Alliance, "OMA Instant Messaging and PresenceService," 2005
[9] Instant messaging and presence protocol ietf working group http://www.ietf.org/html.charters/impp-charter.html
[10] Sip for instant messaging and presence leveraging extensions ietf working group. http://www.ietf.org/html.charters/simplecharter.html
[11] Extensible messaging and presence protocol ietf working group http://www.ietf.org/html.charters/xmpp-charter.html
[12] Gobalindex, http://www.skype.com/intl/enus/support /user-guides/ p2pexplained/.
[13] Sip for instant messaging and presence leveraging extensions working group http://www.ietf.org/html.charters/simplecharter.html

