INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

A Study and Understanding of Mobile Presence Service with Enhanced Scalability in Social Network Application

Dr. B. R. Prasad Babu
Professor & HOD, Department of Computer Science & Engineering
SEA College of Engg &Technology, Bangalore, India
Savitha R.
M.Tech 4th Sem, Department of Computer Science & Engineering
SEA College of Engineering Technology, Bangalore, India

Abstract:
Mobile presence service is a vital component of a social network applications due to mobile user's presence details such as global positioning system location, network address, and online/offline status are continuously apprise to user's online buddies. If persistent updates occur about presence information and also when servers disperse expansive number of messages scalability problem arises in large scale mobile presence service. In this paper a presence cloud which is scalable server architecture is proposed which support large scale social network application where it searches for the presence of his/her buddies and reveals them about his/her arrival. The performance results in achievement of small constant search latency by using direct search algorithm and one hop caching approach.

Key words*: Presence, cloud, Prevalence, Friend list, Social network service*

1. Introduction

Presence [1] entitled applications such as Facebook, Twitter etc., which is produced by mobile devices and cloud computing [2] nature due to the prevalence of internet [3].Way the members are engaged with their buddies on internet are changed by the social network services [4]. In order to interact with buddies across great distance participants can dispense the live event immediately using their mobile device. Mobile user's presence information details will be maintained by mobile presence service [5]. In cloud computing environment mobile presence service is a vital component of social network application. Presence information tells the detail about mobile user's availability, activity and machine capacity**.** Service does binding of user id to his/her current presence information details. Each individual mobile user has a buddy list which includes details of whom he/she wants to interact with in social network services. When a user does shipment from one level to other, this change is instinctively transmitted to each individual on the buddy list. Server cluster technology increases the search speed and decrease the report time. For example in social network application mobile user logs in through his/her mobile device, the mobile presence services searches and reveals each of them about user's friend list such as instant messaging system [6]. Potential of presence cloud [5] [7] can be examined by using search cost and search satisfaction without impaired neither of them. When a user arrives presence server provoke a number of messages is search cost. Time it takes to examine the arrival of user's buddy list is search satisfaction. To help the users who are present worldwide, the services enhanced by Google [3] [8] and Facebook [3] are proliferated among many servers. Presence server used in large scale social network services to ameliorate the coherence of mobile presence services.

In this section, examine the existing server architecture for buddy list in large scale geographical information Centre. Overloading of buddy search message on presence server leads to scalability problem. Presence cloud disseminates many users' information details among many presence servers on the internet, which is used as a building block of mobile presence services. For efficient buddy list search there is no single point collapse, since servers in presence cloud are organized in quorum [9] based server to server architecture to gain small search delay using directed buddy search algorithm. Caching procedure is used to reduce buddy list search. The potential of three architectures such as presence cloud, mesh [10] based scheme and distributed hash table [11] are examined in terms of search response time and friend notification time.

In section 2, searching information among dispense presence servers by using presence cloud which make use of social network application. Using dispense presence server architecture scalability problem is examined, and define a new problem called buddy list search problem.
The next section 3, contains buddy list search problem in disseminated presence server architecture; mobile presence service finds presence details of mobile users m_{uj} in friend list f_{lj} (m_{uj}) and tells each of them about the presence of m_{uj} and also notifies mobile user online friends if mobile user m_{uj} changes his/her presence status .

2. Proposed System

Aim of proposed system is to design an architecture of disseminate server for coherence request to the system for buddy list search. In this project work a scalable server architecture which provides services to 'n' number of users is presented. And presenting a precise design by improving the thought of peer to peer [12] [13] system while designing presence cloud. There are 3 elements in presence cloud which run across presence servers such as presence cloud server overlay, one hop [14] caching approach, and directed buddy search [15].

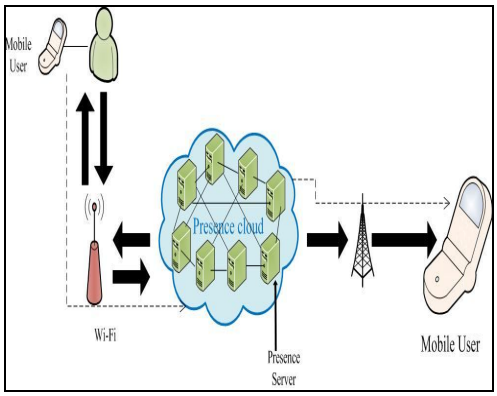
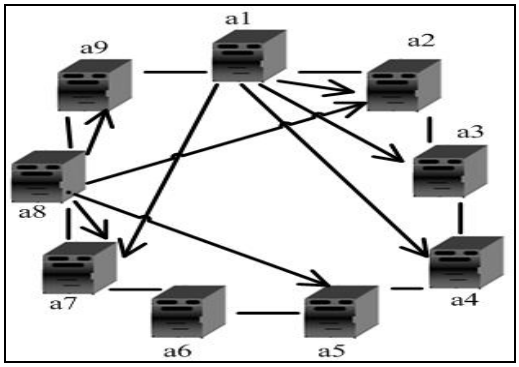


Figure 1. Proposed Architecture for Presence Cloud

- Presence servers which are present in presence cloud, where these presence servers are arranged in quorum based server to server architecture and also load on servers are balance in presence cloud sever overlay.
- All these presence server keeps caches for buddies in order to increase query speed is one hop caching approach.
- Finding small constant search delay results in directed buddy search by decreasing network traffic using one hop search strategy.

Architecture of presence cloud which is the proposed work is shown in Figure1, Using 3G or Wi-Fi services mobile user access the internet and make a data link to the presence cloud. Using secure hash algorithm mobile users are intent to one of the presence servers. To transfer presence information details, the mobile user is authenticated to the mobile presence services and also opens a TCP link. Once path is set up, the mobile user request for the friend list to the presence server which is present in presence cloud. And finally the request is responded by the presence cloud after completing an efficient search of buddy's presence information.

2.1. Presence cloud server overlay



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

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

Figure 2. Presence Cloud server overlay

Presence server nodes are ordered in the form of server to server overlay in presence cloud server overlay and also endow low diameter overlay. Needs two hops to reach from one presence server node to other presence server node is the possession of low diameter and Presence cloud is based on grid quorum system. Size of presence server node is O √m, where m is the number of presence server in mobile presence services. 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 Figure2, grid quorum is set to √9*√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.

2.2. One hop caching

To duplicate the presence information details presence cloud requires caching strategy in order to enhance the efficiency of search operation. In presence cloud for the attached users, presence information details of user list are maintained by presence server node. Duplicating user list by presence server nodes are at most one hop away from itself. When connection is proven by neighbor's cache is updated and also updated periodically with their neighbors. If query accepted by presence server node it is not only respond with matches from cache where user list available by its neighbors. Presence information changes for mobile users when user leaves presence cloud or due to failure. Response from presence server node broad casts its new presence to other neighboring presence server node for updates. Presence information remains constant and up to date throughout the session time of user is ensured by one hop caching strategy.

2.3. Directed buddy search

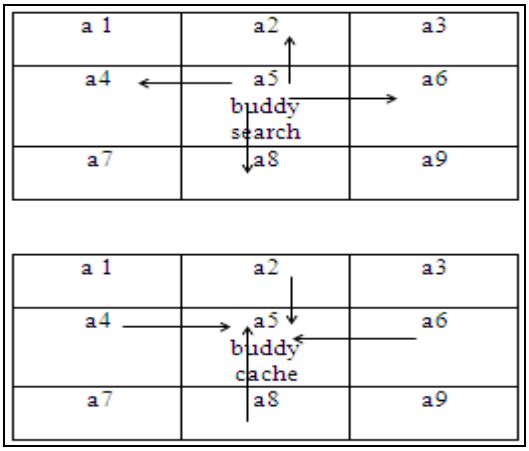


Figure 3. Buddy list search in Presence Cloud

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 neighbors to enhance response time by increasing in finding buddies.

ALGORITHM1. PRESENCE CLOUD MAINTENANCE ALGORITHM

```
1.  //To prove that Presence Server nodes in presence cloud has n presence server records
2.  Definition:-
3.  Presence server record: - contain set of the present presence server record of this presence server node
4.  Presence server record [].link:-the present presence server node in presence server record
5.  Presence server record [].id:- finder of the Precise link in presence server record.
6.  n.id:- finder of Presence Server node
7.  Algorithm:-
8.   s =sizeof (presence server record)
9.   for k = 1 to s do
10. n =presence server record[k].link
11. if n.id ≠presence server record[k].id then
12. fn=find node(n)
13. if fn=nil then
14. presence server record[k].link=rn(n)
15. else
16. presence server record[k].link=fn
17. end if
18. else
19. bf=send message (n)
20. if bfailed= true then
21. presence server record[k].link=rn(n)
22. end if
23. end if
24. end for
```

Algorithm1 is fault tolerance design, where each presence server contains presence server list by using maintenance function. System architecture for social network application using mobile presence services is shown in Figure2, setting overall outlook of application in service layer where user registers for presence cloud and updates his/her profiler. In persistence layer performing CURDS operation such as create, read, update and delete. Presence cloud searches for presence information details such as online/offline, network address, global positioning system location when user joins the network.

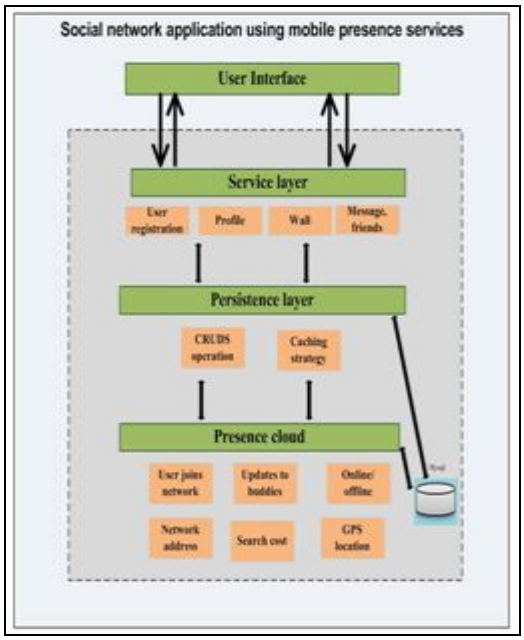


Figure 4. System Architecture for Mobile presence service in social network application

3. Implementation

In Figure 4, the process flowchart of the implementation is shown. The proposed system is implemented in 32 bit Windows operating system with 1 GHz Processor and 1GB RAM .The design environment is selected in java. User registers for presence cloud if it is success then he/she can view the friend list and send a request to the presence server which is present in presence cloud for buddy list search. Presence cloud makes an efficient search of buddy list and returns the result to the user. Then user send request to buddies

using friend id if accepted then added to buddy list, otherwise buddies can ignore the request. Buddies send messages to their friend and also can view message, compose messages for their buddies using friend id and finally navigates to the main page.

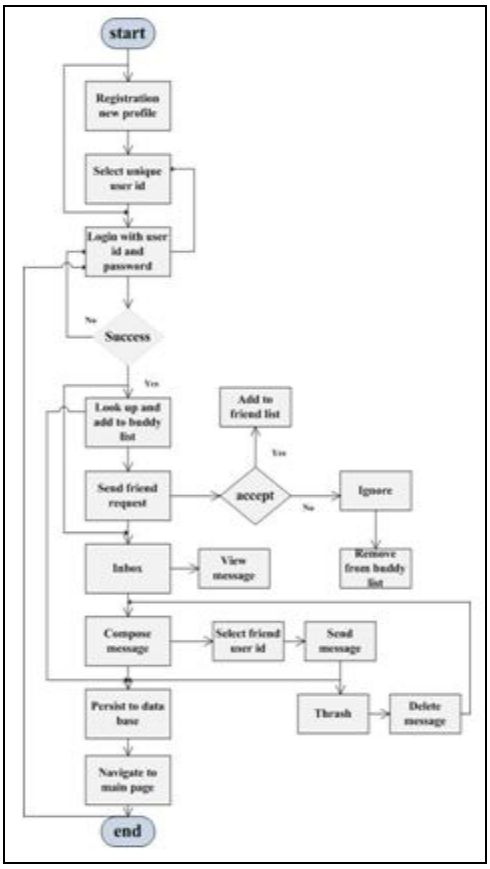


Figure 5. Process flowchart implementation

4. Performance Analysis

In king topology for king data set there is a real internet topology. Internet computation are derived using technique for king data set latency pattern. In Figure5 shows, the brite topology alpha is set to 0.15 and beta is set to 0.2 using Waxman model BRITE topology generator creates AS topology.

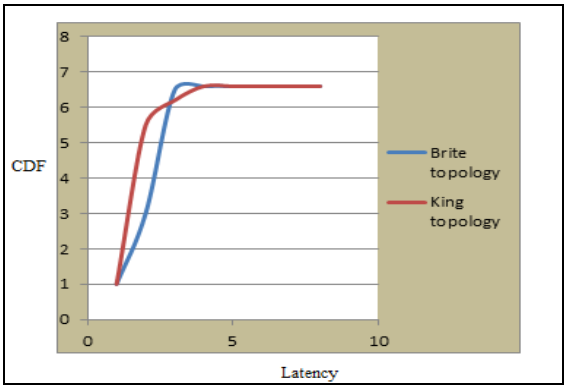


Figure 6. System performance

Performance metrics: -There are three metrics which computes the performance of server architecture shown in Figure5. 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.

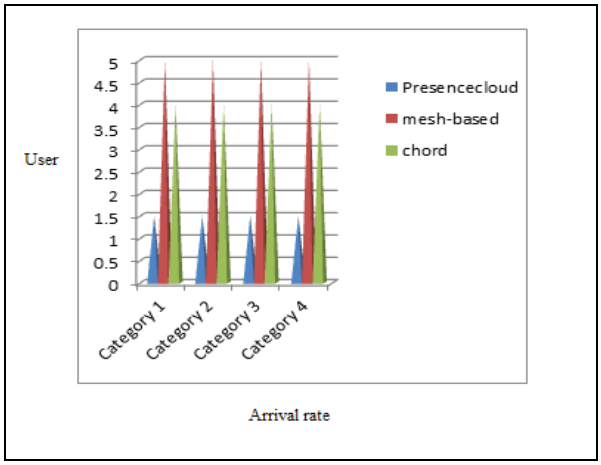


Figure 7. Performance of Presence Cloud

5. Conclusion

In large scale social network services mobile presence services is supported by the scalable server architecture called as presence cloud. Performance improved for mobile presence services and also low search latency is accomplished by presence cloud. Number of buddy search messages increased with user arrival rate. Presence cloud achieved performance gain in terms of search cost.

6. References

1. K.Peternel, L.zebee andA.Kos, “Using Presence information for an effective Collaboration”.
2. ChetanS,Gautam Kumar,K.Dinesh,Mathew K,and Abhimanya M.A,”Cloud computing for Mobile world”.
3. Facebook, http://www.facebook.com.
4. Christian Voigt, Sandra Barker, Sharron King, Kit Macfarlane, Tim Sawyer & Sheila Scutter, “Conceptualising social networking capabilities: Connections, objects, power and affect”, 2010.
5. D.Dhiravida vasantha nayaki and G.Mangayarkarasi,”Presence server for mobile ubiquity services in presence cloud,”March 2014.
6. Instant Messaging and presence protocol IETF working group,http://www.ietf.org/html.charters/impp-charter.html,2012.
7. Chi-Jen Wu, Jan-Ming Ho, *Member, IEEE*, Ming-Syan Chen, *Fellow, IEEE,”* A Scalable Server Architecture for Mobile Presence Services in Social Network Applications”,2013.
8. Google, http://www.google.com.
9. Quorumbasedtechniqueshttp://www.en.wikipidiaorg/wiki/Quorum_(distributed_computing).
10. Jianming zhao, Nianmin yao, Shaobin Cai, and Xiang li, “Tree-Mesh Based P2P Streaming data distribution schema”,2012.
11. Distributedhash tables http://www.cs.cmu.edu/~dga/15-744/S07/lectures /16-dht.pdf.
12. Juuso aleksi lehtinen, “Mobile peer-to-peer over session initiation protocol”, August 4,2008.
13. Kundan Singh and Henning Schulzrinne Department of Computer Science, Columbia University {kns10,hgs}@ cs.columbia.edu, “SIPPEER : A session iniiation protocol (SIP)-baed peer-to-peer internet telephony cllient adaptor” .
14. Michael Piatek, Tomas Isdal, Arvind Krishnamurthy , and Thomas Anderson “One hop Reputations for Peer to Peer File Sharing Workloads”.
15. Brent Hecht, Jaime Teevan , Meredith Ringel Morris, and Dan Liebling, “SearchBuddies: Bringing Search Engines into the Conversation”, 2012