Research Article Impact Factor: 0.621 ISSN: 2319-507X
Seema Naik, IJPRET, 2014; Volume 2 (9): 638-646 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

SCALABLE MOBILE PRESENCE SERVICE IN LARGE-SCALE SOCIAL NETWORK SERVICES

SEEMA S. NAIK[1], PROF. V. INGOLE[2]

1. M. E. Student, Department of Electronics and Telecommunication, IBSS COE, Maharashtra, India
2. Professor, Department of Electronics and Telecommunication, IBSS COE, Maharashtra, India

Accepted Date: 27/02/2014 ; Published Date: 01/05/2014

Abstract: Now a day's Social network applications have a wide popularity on mobile devices. In mobile social network application mobile presence service is important because it maintains each mobile user's presence information, like current status (online/offline), GPS location and network address, and also updates the user's online friends with the information continually. If presence updates occur frequently, the huge number of messages distributed by presence servers may lead to a scalability problem in a large-scale mobile presence service. To attend the problem, Presence Cloud is an efficient and scalable server architecture, which supports large-scale social network applications [1]. When a mobile user joins a network, Presence Cloud searches for the presence of one's friends and notifies them of one's arrival. Presence Cloud organizes presence servers into a quorum-based server-to-server architecture for efficient presence searching. Presence Cloud analyze the total number of messages generated by the presence server when a user arrives & search satisfaction level. The results of simulations demonstrate that Presence Cloud achieves performance gains in the search cost without compromising search satisfaction [1].

Keywords: Mobile Cloud Computing;, Mobile Presence Services, Distributed Presence Servers, Cloud Computing.

PAPER-QR CODE

Corresponding Author: MS. SEEMA S. NAIK

Access Online On:

www.ijpret.com

How to Cite This Article:

Seema Naik, IJPRET, 2014; Volume 2 (9): 638-646

INTRODUCTION

Mobile devices (e.g., smart phone, tablet pcs, etc) & social networks e.g. Facebook [1], Twitter [2], Foursquare [3], Google Latitude [4], buddy cloud [5] and Mobile Instant Messaging (MIM) [6], are increasingly becoming an essential part of human life as the most effective and convenient communication tools not bounded by time and place. Mobile users accumulate rich experience of various services from mobile applications (e.g., iPhone apps, Google apps, etc).BECAUSE of the ubiquity of the Internet, mobile devices and cloud computing environments can provide presence-enabled applications, i.e., social network applications/services, worldwide. Facebook [4], Twitter [5], Foursquare [6], Google Latitude [7], buddy cloud [8] and Mobile Instant Messaging (MIM) [9], are examples of presence-enabled applications. Social network services are changing the ways in which participants engage with their friends on the Internet. They exploit the information about the status of participants including their appearances and activities to interact with their friends. Together with an explosive growth of the mobile applications and emerging of cloud computing concept, mobile cloud computing (MCC) has been introduced to be a potential technology for mobile services. MCC integrates the cloud computing into the mobile environment and overcomes obstacles related to the performance (e.g., battery life, storage, and bandwidth), environment (e.g., heterogeneity, scalability, and availability), and security (e.g., reliability and privacy) discussed in mobile A mobile presence service is an essential component of social network services in cloud computing environments. The key function of a mobile presence service is to maintain an up-to-date list of presence information of all mobile users. Most of the presence services use server cluster technology for increasing a mobile presence service's search speed and decreasing the notification time, [3].

I. LITERATURE REVIEW

Thus, we explore the relationship between distributed presence servers and server network topologies on the Internet, and propose an efficient and scalable server-to-server overlay architecture called Presence Cloud to improve the efficiency of mobile presence services for large-scale social network services. Each mobile user has a friend list, typically called a buddy list. The buddy-list search problem introduce in distributed presence architectures in large-scale geographically data centers. The underlying principle behind the design of Presence Cloud is to distribute the information of millions of users among thousands of presence servers on the Internet. To avoid single point of failure, no single presence server is supposed to maintain service-wide global information about all users [1].

1. Buddy-list search problem

Assume the geographically distributed presence servers to form a server-to- server overlay network, $G = (V;E)$, where V is the set of the Presence Server (PS) nodes, and E is a collection of ordered pairs of V . Each PS node $ni \in V$ represents a Presence Server and an element of E is a pair $(ni; nj) \in E$ with $ni; nj \in V$. Because the pair is ordered, $(nj ; ni) \in E$ is not equivalent to $(ni; nj) \in E$. So, the edge $(ni; nj)$ is called an outgoing edge of ni, and an incoming edge of nj. The server overlay enables its PS nodes to communicate with one another by forwarding messages through other PS nodes in the server overlay. Suppose a set of the mobile users in a presence service is $U = \{u1; : : : ; ui; : : : ; um\}$, where $1 \leq i \leq m$ and m is the number of mobile users. A mobile user ui connects with one PS node for search other user's presence information, and to notify the other mobile users of his/her arrival. Moreover, we define a *buddy list* as following.

Definition 1. Buddy list, $Bi = \{b1; b2; : : : ; bk\}$ of user $ui \in U$, is defined as a subset of U, where $0 < k \leq |U|$. Furthermore, B is a symmetric relation, *i.e*, $ui \in Bj$ implies $uj \in Bi$.

Problem Statement-Buddy-List Search Problem When a mobile user ui changes his/her presence status, the mobile presence service searches presence information of mobile users in buddy list Bi of ui and notifies each of them of the presence of ui and also notifies ui of these online buddies. The Buddy-List Search Problem is designing a server architecture of mobile presence service such that the costs of searching and notification in communication and storage are minimized.

Analysis of a New Architecture of Mobile

Presence Service:

Let h denote the probability of having all users in

the buddy list of ui to be attaching to the same PS node as ui. It is the probability of having no need to send search messages when ui attaches to a PS node. Thus,

$h = \Pi\, jBij$ 1 $n = n$?$jBij$:

The expected number of search messages generated by this PS node per unit time is then

$(n - 1)$ X $(1 - h)$?_:

For a reasonable size of set Bi (e.g., $|Bi| \geq 3$) and

$n \geq 100$, we consider the expected number Q of messages generated by the n PS nodes per unit time, then we have

$Q = n$?$(n - 1)$?$(1 - h)$?_

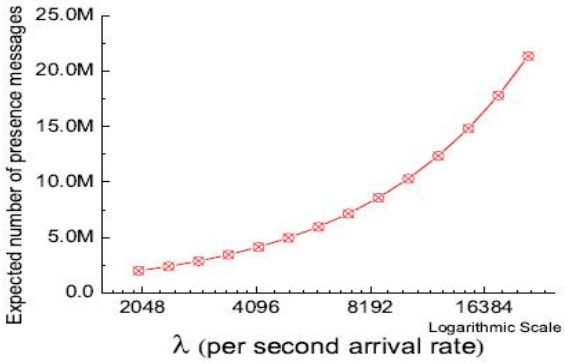
= *n* ?(*n* – 1) ?(1 – *h*) ?_

≃ (*n* – 1) ?_:

Thus, as the number of PS nodes increase, both the total communication and the total CPU processing overhead of presence servers also increase. It also shows that _ is another important parameter having impact on the system overhead. When _ increases substantially, it has a major impact on the system overhead. However, a scalable presence system should be able to support more than 20,000 mobile user logins per second in burst cases as reported by [10].

Fig. 1, we plot statistics for all expected queries transmitted in a mobile presence service to show the increase in number of buddy search messages as lambda increases. Fig. 1(a) shows that in a 1,000 PS nodes system, and the average

arrival rate of mobile users increases from 2,000 to 21,000[1]. From the results of analysis, the number of buddy searching messages increases with increasing average arrival rate of mobile



(a) The average arrival rate of mobile users increases from 2,000 to 21,000 per second in a 1,000 PS nodes mobile presence system

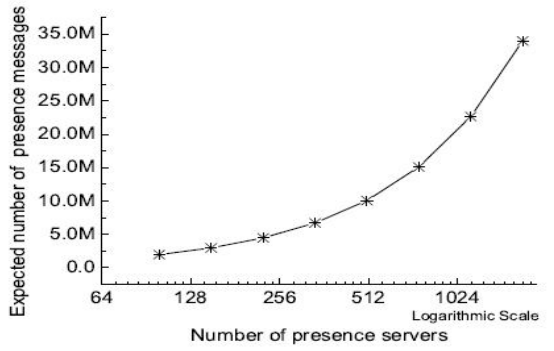
Fig. 1. The analysis of expected total transmissions when searching for buddy lists in a distributed mobile presence service (The *x* axis of both sub figures is in logarithmic scale) users. Fig. 1(b) plots another scenario, the average arrival rate of user joining is 20,000 per second, and the number of PS nodes in a system increases from 100 to 1,700[1]. It shows that the number of total buddy searching messages increases significantly with the number of PS nodes.

II. DESIGN OF PRESENCE CLOUD

A new design of mobile presence services is needed to address the buddylist search problem, especially for the demand of mobile social network applications. PresenceCloud is used to construct and maintain a distributed server architecture and can be used to efficiently query the system for buddy list searches. PresenceCloud consists of three main components as follows

PresenceCloud server overlay of PresenceCloud has a balanced load property and a two-hop diameter with $O(\sqrt{n})$ node degrees, where n is the number of presence servers. _

One-hop caching strategy is used to reduce immediate neighbors. the number of transmitted messages and accelerate query speed. All presence



(b) The number of PS nodes increases from 100 to 1,700 in a mobile presence service with 20,000 per second arrival rate of mobile users

Fig 1. The analysis of expected total transmissions when searching for buddy lists in a distributed mobile presence service (The *x* axis of both sub figures is in logarithmic scale)

servers maintain caches for the buddies offered by their immediate neighbors.

Directed buddy search is based on the directed search strategy. Presence Cloud ensures an one-hop search, it yields a small constant search latency on average.

1.1 Presence Cloud Overview

The primary abstraction exported by our Presence Cloud is used to construct a scalable server architecture for mobile presence services, and can be used to efficiently search the desired buddy lists. We illustrated a simple overview of

Presence Cloud in Fig. 2. In the mobile Internet, a mobile user can access the Internet and make a data connection to Presence Cloud via 3G or Wifi services. After the mobile user joins and authenticates himself/herself to the mobile

presence service, the mobile user is determinately directed to one of Presence Servers in the Presence Cloud by using the Secure Hash

Algorithm, such as SHA-1 [30]. The mobile user opens a TCP connection to the Presence Server

(PS node) for control message transmission, particularly for the presence information. After the ccontrol channel is established, the mobile user sends a request to the connected PS node for his/her buddy list searching.

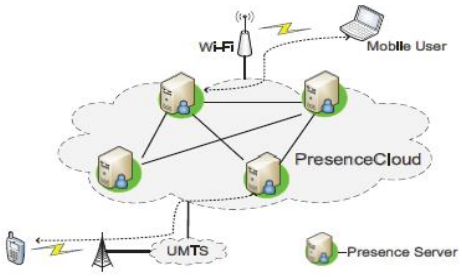


Fig. 2. An overview of PresenceCloud

1.2 PresenceCloud Server Overlay

The PresenceCloud 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. The detailed

description is as follows. Our PresenceCloud is based on the concept of *grid quorum system* [12], where a PS node only maintains a set of PS nodes of size O($\sqrt{n}$), where n is the number of PS nodes in mobile presence services. In a PresenceCloud

system, each PS node has a set of PS nodes, called *PS list*, that constructed by using a grid quorum system, shown in Fig. 3 for n=9. The size of a grid quorum is $\lceil\sqrt{n}\rceil\times\lceil\sqrt{n}\rceil$.When a PS node joins the server overlay of PresenceCloud, it gets an ID in the grid, locates its position in the grid and

obtains its PS list by contacting a root server. On the $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil$ grid, a PS node with a grid ID can pick one column and one row of entries and

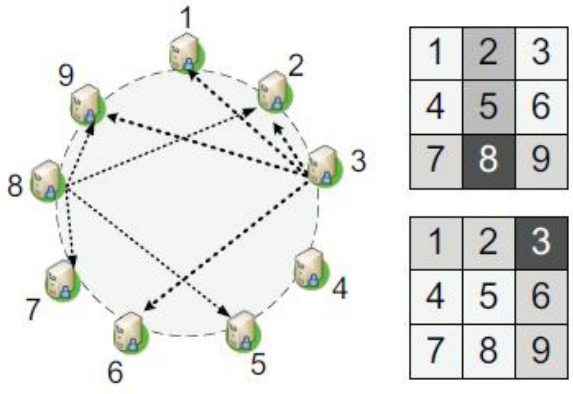


Fig. 3. A perspective of PresenceCloud Server Overlay

these entries will become its PS list in a Presence Cloud server overlay. Fig. 3 illustrates an example of Presence Cloud, in which the grid quorum is set to $\lceil\sqrt{9}\rceil \times \lceil\sqrt{9}\rceil$. In the Fig. 3, the PS node 8 has a PS list {2,5,7,9} and the PS node 3 has a PS list {1,2,6,9}. Thus, the PS node 3 and 8 can construct their overly networks according to their PS lists respectively. We now show that each PS node in a Presence Cloud system only maintains the PS list of size $O(\sqrt{n})$, and the construction of Presence Cloud using the grid quorum results in each PS node can reach any PS node at most two hops.

1.3 One-hop Caching

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. Each presence server in Presence Cloud only maintains the one-hop replicas of presence information of size $O(u \times \sqrt{n})$, where u is denoted the average number of mobile users in a presence server and n is the number of presence servers.

1.4 Directed Buddy Search

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.

IV. COST ANALYSIS

In this section, we provide a cost analysis of the communication cost of PresenceCloud in terms of the number of messages required to search the buddy information of a mobile user. Note that how to reduce the number of inter-server communication messages is the most important

metric in mobile presence service issues. The *buddy-list search problem* can be solved by a brute-force search algorithm, which simply searches all the PS nodes in the mobile presence

service. The communication cost of searching buddies and replicating presence information can be formulated as *Mcost = QMesh +RMesh*, where *RMesh* is the communication cost of replicating

presence information to all PS nodes. However, in Presence Cloud, a PS node not only searches a buddy list and replicates presence information, but also notifies users in the buddy list about the new presence event. Since Presence Cloud must reply every online user on the buddy list individually, it is clear that extra messages must be transmitted. To discuss the search cost of the DHT-based presence architecture. We make the following assumptions to simplify the analysis: 1) the presence information of a mobile user is only stored in one PS node (i.e. no replication). 2) all mobile users are uniformly distributed in all PS nodes. Although our analysis is based on the Chord [13], it can be extended to other DHTs. Note that some replica algorithms [14] are proposed for DHT systems, but these algorithms also increase the complexity of DHT.

V. PERFORMANCE EVALUATION

Within the context of the model, the performance of server architectures can measure using the following three metrics:

1) Total Searching Messages: This represents the total number of messages transferred between the query initiator and the other PS nodes during the simulation time. This is meat and potatoes metric in our experiments, since it is widely regarded to be critical in a mobile presence service that we discussed both in the Section 3 and the Section 5. *2) Average Searching Messages per-arrived user:*

The number of searching messages used per arrived user. Moreover, this metric is independent of user arrival pattern. *3) Average Searching latency:* This represents that average buddy searching time for a joining mobile user. This metric is a critical metric for measuring the search satisfaction of mobile presence services.

III. CONCLUSION

Through a simple mathematical model, we show that the total number of buddy search messages increases substantially with the user arrival rate and the number of presence servers. The results of simulations demonstrate that Presence Cloud achieves major performance gains in terms of the search cost and search satisfaction. the results according to human expectations which was given as the input query.

REFERENCES

1. A Scalable Server Architecture for Mobile Presence Services in Social Network Applications Chi-Jen Wu, Jan-Ming Ho, *Member, IEEE,* Ming-Syan Chen, *Fellow, IEEE*

2. A survey of mobile cloud computing: architecture, applications, and approaches Hoang T. Dinh, Chonho Lee, Dusit Niyato* and Ping Wang School of Computer Engineering, Nanyang Technological University (NTU), Singapore

3. 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.

4. Facebook, http://www.facebook.com.

5. Twitter, http://twitter.com.

6. Foursquare http://www.foursquare.com.

7. Google latitude, http://www.google.com/intl/enus/latitude/intro.html.

8. Buddy cloud, http://buddycloud.com.

9. Mobile instant messaging, http://en.wikipedia.org/wiki/Mobile_instant messaging.

10. A. Houri, E. Aoki, S. Parameswar, T. Rang, , V. Singh, and H. Schulzrinne, "Presence inter domain scaling analysis for sip/simple," *RFC Internet-Draft*, 2009.

11. M. Maekawa, "A *pn* algorithm for mutual exclusion in decentralized systems," *ACM Transactions on Computer Systems*, 1985.

12. Instant messaging and presence protocol ietf working group http://www.ietf.org/html.charters/impp-charter.html.

13. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet," *IEEE/ACM Tran. on Networking*, 2003.

14. X. Chen, S. Ren, H. Wang, and X. Zhang, "Scope: scalable consistency maintenance in structured p2p systems," *Proc. of IEEE INFOCOM*, 2005.