VOL. 10, NO. 16, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

MANAGEMENT AND ANALYTIC OF SOCIAL NETWORK APPLICATION WITH MOBILE BASED MEMORY DATABASE AND DYNAMIC QUERYING

G. Dheepa and V. Vijayakaveri
Department of Information Technology, Sathyabama University, Sholinganallur, Chennai, Tamilnadu, India
E-Mail: dheepag.1986@gmail.com

ABSTRACT

Social network applications are flattering progressively more familiar on mobile network. A mobile presence application is an imperative constituent of a social network application because it manages each mobile user's occurrence information, such as the current standing (online/offline), GPS locality and network address, and also updates the user's network friends with the information frequently. If updates occur habitually, the enormous number of messages distributed by servers may lead to a scalability problem in a large-scale mobile presence service. By avoiding the problem, we put forward efficient and scalable server structural design, called PresenceCloud; this enables mobile presence services to support large-scale social network applications. When a mobile user comes under a network, Presence Cloud searches for the presence of his/her friends and notifies them of his/her advent. Presence Cloud organizes quorum-based server-to-server architecture to improve searching performance. A directed search algorithm and a one-hop caching strategy to accomplish small constant search latency. We are going to analyze the performance of Presence Cloud in terms of the search cost and search satisfaction level. The search cost is depending on the total number of messages leaved by the presence server when a user arrives; and search satisfaction level is depending on the time takes to search for the arriving user's friend list. The results of simulations exhibit that Presence Cloud achieves performance gains in the search cost without compromising search satisfaction.

Keywords: mobile presence services, social networks, distributed presence servers, cloud computing.

1. INTRODUCTION

Mobile devices and cloud computing environments can provide presence-enabled applications. Face book [1], Twitter [2], Foursquare [3], Google Latitude [4], buddy cloud [5] and Mobile Instant Messaging (MIM) [6], are the examples of Presence-enabled applications. Social network services are changing in which participants engage with their friends on the network. They make use of the information about the status of participants includes appearances and actions to interrelate with their friends. In addition, to the wide accessibility of mobile devices (e.g., Smartphone's) that utilizes wireless mobile network technologies, social network services facilitate participants to share live experiences instantly across the network. For example, Face book receives more than 25 billion shared items every month and Twitter receives more than 55 million tweets each day. Hence, the social network services will be the next trend of mobile Internet applications mobile presence service is an crucial constituent of social network services in cloud computing environments. A mobile presence service is to maintain an up-to-date list of occurrence information of all mobile users. It includes details about a mobile user's position, accessibility, motion, device potential, and preferences. The service also binds the users ID to his/her current information, as well as recovers and donates to changes in the presence information of the user's close members. In social network air force, each mobile user has a friend list, called a buddy list, which contains the contact information of other users of friends list. The mobile user's status is transmitting automatically to each friend on the buddy list. For example, a mobile user logs into a social network application, the mobile presence service searches each and everyone on the user's buddy list. To maximize a search speed and minimize the announcement time, nearly all presence services utilize server cluster technology [7]. The number of mobile presence service users will increase considerably in the near future.

The scalable mobile presence service is necessary for the prospect Internet applications. Many Internet services have been deployed in disseminated paradigms in addition to cloud computing applications. For example, the services implemented by Google and Face book are broaden among as many distributed servers as achievable to sustain the enormous number of users worldwide. We travel around the affiliation between distributed presence servers (PS) and server network topologies on the Internet, and also propose a well-organized and scalable server-to-server superimpose architecture called Presence Cloud to improve the efficiency of mobile presence services for large-scale social network services. First, observe the server architectures of existing presence services. The buddy-list search problem in scattered presence architectures in large-scale geographically data centers. When a distributed presence service is congested with buddy search messages means buddy list search problem arises. A scalable server-to-server architecture is used as

an element for mobile presence services. To avoid single point of malfunction, PresenceCloud organizes presence servers into a quorum-based server-to-server architecture to smooth the progress of proficient buddy list searching. Directed buddy search algorithm to attain small steady search latency; and employs an active caching strategy that considerably reduces the number of messages generated by each search for a list of buddies. Recital complication of Presence Cloud and two other architectures, like a Distributed Hash Table (DHT)-based scheme and a Mesh- based scheme are going too analyzed. The results make obvious that Presence- Cloud achieves foremost performance gains in terms of reducing the number of messages without sacrificing search satisfaction. The donation of this paper through our mathematical calculation, the scalability trouble under the distributed server architectures of mobile presence services is calculated. Finally, we analyze the recital complication of Presence Cloud and different designs of distributed architectures.

2. THE PROBLEM STATEMENT

Problem statement is the system model and buddy- list search problem. Geographically distributed presence servers is used to form a server-to-server overlay network, G = (V, E), where V is the set of the Presence Server nodes, and E is a collection of controlled pairs of V. Each PS node ni € V represents a Presence Server and an element of E is a pair (ni, nj) € E with ni, nj € V. Because the pair is ordered, (nj, ni) € E is not equivalent to (ni, nj) € E. So, the edge (ni; nj) is called an outgoing edge of ni, and an incoming edge of nj. The server superimposes enables PS nodes to communicate with one another by forwarding messages through other PS nodes in the server overlay. A set of the mobile users in a presence service as U={u1 ,... ; ui , .. . ; um }, where 1<=i<=m and m is the number of mobile users. A mobile user ui mingle with one PS node for search other user's presence information and to notify the other mobile users of his/her arrival. When a mobile user ui changes his/her presence standing, 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 associates. The Buddy-List Search Problem is then defined as designing server architecture of mobile presence service such that the costs of searching and notification in communication.

3. ANALYSIS AN ARCHITECTURE FOR MOBILE PRESENCE SERVICE

In a naive architecture of mobile presence services predictable rate of messages generated to search for buddies of newly arrived user. Each mobile user can link and disappear the presence service randomly, and each PS node only knows those mobile users directly attached to it. We also assume the prospect for a mobile user to attach to a PS node to be uniform. Let's denote the average inward rate of mobile users in a mobile presence service. Here we have to tell about architecture of mobile presence services and the problem of designing presence servers. Each PS node has infinite service capacity. Hence, $\mu = \frac{\lambda}{n}$ is the average rate of mobile users attaching node, the n is distinguished the number of PS nodes in a mobile presence examination. Where h denote the prospect of having all users in the buddy list of ui to be attaching to the same PS node as ui.

Thus, $$h = \prod_{|B_i|} \frac{1}{n} = n^{-|B_i|}.$$

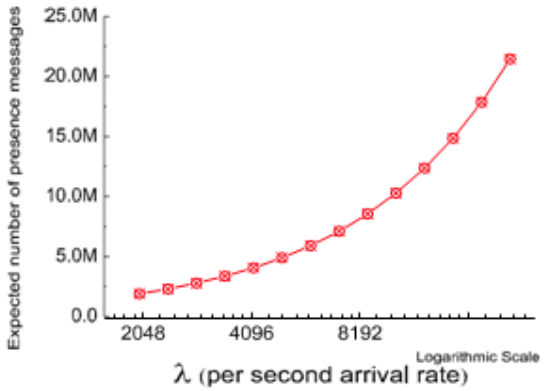
The anticipated number of hunt messages generated by PS node in unit time then

$$(n-1) \times (1-h) \times \mu.$$

Reasonable size of set B_i an $n \geq 100$, consider the expected number Q of messages generated by the n PS nodes per piece time, we have

$$\begin{aligned} Q &= n \times (n-1) \times (1-h) \times \mu \\ &= n \times (n-1) \times (1-h) \times \frac{\lambda}{n} \\ &\simeq (n-1) \times \lambda. \end{aligned}$$

The number of PS nodes increase, both the total communication and total CPU processing overhead also increase. Increases substantially, a major rear-ender on the system transparency. However, a scalable presence system should be able to support more than 20, 000 mobile user logins per second. In the following Figure 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. The figure shows 1,000 PS nodes, then the average arrival rate of mobile user's increases from 2, 000 to 21,000. From the results of study, the quantity of buddy searching messages increases with increasing average arrival rate of mobile users. This plot most important for to design scalable server architecture.



(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

Figure-1.

3. DESIGNOF RESENCECLOUD

The previous algorithms are used to find the fixed object searching problem in distributed systems for astonish back-up. Mobile environment in social network is complementary undependable. Mobile presence services is to address the buddy-list investigate predicament, in exacting the instruct of mobile social network applications. This is worn to generate and maintain distributed server architecture and used to proficiently query the system for buddy list searches. Presence Cloud having a three main mechanism and this should run across a set of presence servers. By the intend of Presence Cloud, we must follow the thoughts of P2P systems and present a design for mobile presence services. There are three chief mechanisms.

- **PresenceCloud server overlay** have a presence servers based on the idea of grid quorum system in a network. Server overlay of Presence Cloud has a balanced load property and a two-hop diameter with $O(\sqrt{n})$ node degrees.
- **One-hop caching strategy** is used to decrease the number of transmitted messages and pick up the rapidity query speed.
- **Directed buddy search** is depends on the directed search algorithm. It could depend on a one-hop search; it gives a product of small constant search latency on average.

3.1. Overview of presence cloud

PresenceCloud is used to erect scalable server architecture for mobile presence services, also the services used to proficiently search the most wanted buddy lists. The list manages an unrefined suggestion of Presence Cloud in following Figure make by using network.

The mobile network, and provide a data connection to Presence Cloud via 3G or Wi-Fi services. Mobile user joins a cloud network and authenticates himself/herself to the mobile presence service in social network, the determinately directed to one of Presence Servers in the Presence Cloud by using the Secure Hash Algorithm (SHA). Network Person of mobile can opens a TCP connection to the Presence Server (PS node) for control meaning distribution, principally for the happening in advancement. The control waterway is conformist, demand of mobile user to associated PS node for his/her buddy list. This Cloud do well-organized searching operation and return the presence information of the desired buddies to the mobile we are departing to explain three components of Presence- Cloud.

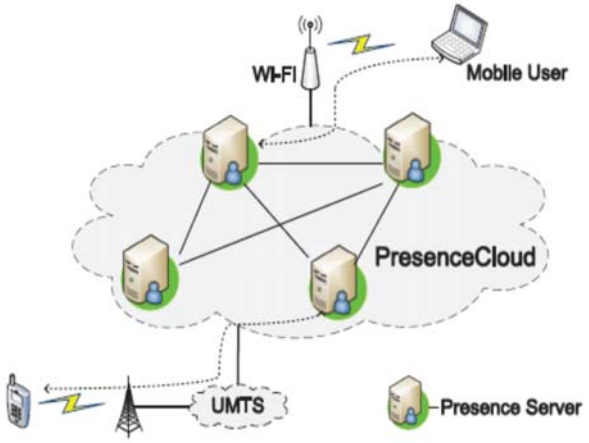


Figure-2. An overview of Presence cloud.

3.2. PresenceCloud server overlay

The PresenceCloud server overlay construction organizes the PS nodes into server-to-server approaches. The low-diameter chattels make sure that a PS node needs to reach any node of presence server in social network. Node of PS can only maintains a set of PS nodes of size $O(\sqrt{n})$, the number of PS nodes denotes n in mobile armed forces. Nodes join a apparent demonstration PS record, Using a grid quorum system, The extent of a grid quorum is $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil$. In this grid quaram presence server node with a grid ID can pick one column and one row of entries and these entries will become its PS list in a Presence Cloud server overlay chattels. Presence cloud node can have details of PS list can be equation $O(\sqrt{n})$. Presence cloud with grid quaram can have a result of each PS node have to reach any PS node almost two hops.

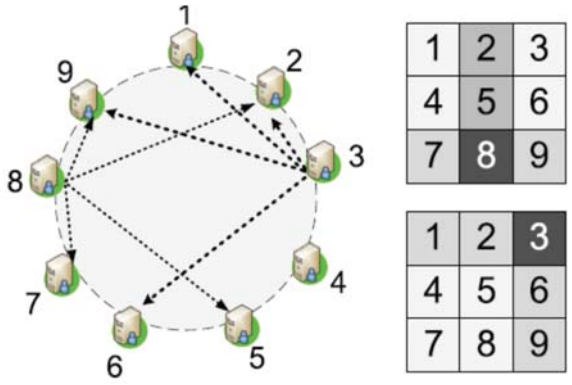


Figure-3. A perspective of PrenceCloud server overlay.

3.2.1 Presence cloud stabilization algorithm

Presence cloud algorithm is fault tolerance intend. Presence server node can have Stabilization development periodically contacts existing PS nodes to sustain the list of presence server. When a new PS node joins, it can be added to PS list by contacting a root. Node of n detects failed PS nodes starting PS list of network.

Algorithm

Presence cloud stabilization algorithm.

```
/* periodically verify PS node n's pslist */
Definition:
pslist: set of the current PS list of this PS node, n
pslist[].connection: the current PS node in pslist
pslist[].id: identifier of the correct connection in pslist
node.id: identifier of PS node node
Algorithm:
r ← Sizeof(pslist)
for i = 1 to r do
    node ← pslist[i].connection
    if node.id ≠ pslist[i].id then
        /* ask node to refresh n's PS list entries */
        findnode ← Find_CorrectPSNode(node)
        if findnode = nil then
            pslist[i].connection ← RandomNode(node)
        else
            pslist[i].connection ← findnode
        end if
    else
        /* send a heartbeat message */
        bfailed ← SendHeartbeatmsg(node)
        if bfailed = true then
            pslist[i].connection ← RandomNode(node)
        end if
    end if
end for
```

3.3 One-hop caching

To develop the effectiveness of the investigate operation; PresenceCloud desires a caching stratagem to duplicate presence message of users from the network. The people of Presence network can be updated by the expensive mechanism for scattered agreement of users. Server of presence node having a user list of presence information of the attached users. PS nodes duplicate the user list at most one hop missing preliminary itself. It invent an association the cache is updated. Presence server node receives a query, it can respond to all of its neighbors. When a mobile user changes its presence in sequence, PS nodes for being paid reorganized promptly. The one-hop caching stratagem ensures that the user's presence information could remain mostly up-to- date and consistent throughout the session time of the user.

3.4 Directed buddy search

Buddy search algorithm is to improve a searching performance in the presence cloud of managing. Searching of buddy information can minimize probing reaction time in the mobile presence services of social application. Presence Cloud searching algorithm is attached with the two-hop overlay and one-hop caching strategy ensures that PresenceCloud can typically provide swift responses for a large number of users.

Directed buddy search algorithm:

1. A mobile user logins PresenceCloud and decides the associated PS node, q.
2. The user sends a Buddy List Search Message, B to the PS node q.
3. When the PS node q receives a B, then retrieves each b_i from B and searches its user list and one-hop cache to respond to the coming query. And removes the responded buddies from B.
4. If $B = nil$, the buddy list search operation is done.
5. Otherwise, if $B \neq nil$, the PS node q should hash each remaining identifier in B to obtain a grid ID, respectively.
6. Then, the PS node q aggregates these $b^{(g)}$ to become a new $B^{(j)}$, for each $g \in S_j$. Here, PS node j is the intersection node of $S_q \cap S_g$. And sends the new $B^{(j)}$ to PS node j.

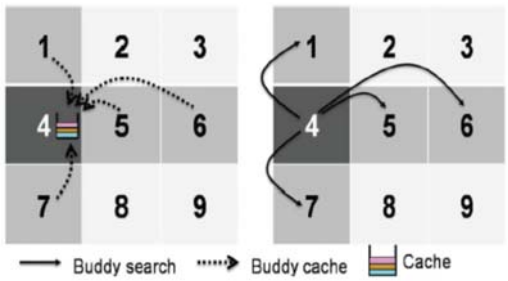


Figure-4. An example of buddy list searching operations in PresenceCloud.

4. COST ANALYSIS

The user has to endow with cost psychoanalysis of the communication cost of Presence Cloud and number of messages requisite to search the buddy information. The buddy-list search problem can be solved by a brute-force search algorithm, and this algorithm simply searches all the PS nodes in the mobile presence service in internet. Mesh-based design denotes that the algorithm replicates all the presence information at each PS node. The search cost can be denote by QMesh. On the other hand, the system needs n-1 messages to replicate a user's presence information to all PS nodes. The communiqué cost of searching buddies and replicating presence information can be formulated as Mcost = QM mesh + Rmesh, where RM esh is the communication cost of replicating presence information to all nodes of presence server. We have Mcost = O(n). Cost of searching Presence Cloud is denoted as QP, which is denoted as $2 \times (\lceil \sqrt{n} \rceil - 1)$ messages for both searching buddy lists and replicating presence information.

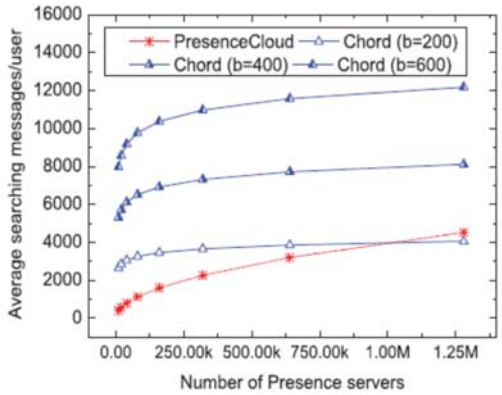


Figure-5. Average searching messages versus very large number of PS nodes.

The replication of presence cloud can be denoted as $P_{cost} = Q_p = 2 \times (\lceil \sqrt{n} \rceil - 1)$. Presence Cloud of a PS node is to search a buddy list and replicates presence information. The buddy list is located on different PS nodes of a presence cloud environment. When all mobile users are distributing equally among the PS nodes and this should be worst case $P_{cost} = 4 \times (\lceil \sqrt{n} \rceil - 1)$.

The search cost of the DHT-based presence architecture is very highly expensive. 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). Note that in some DHT, data replicating increase both the node workload and the maintenance complexity of networking. 2) All mobile users are uniformly distributed in all PS nodes. Our analysis is based on the chord. This note has to maintain the login neighbors in the presence cloud. The buddy list has the information about the presence friends list. The data present in the buddy list can be search one by one. The total search complexity of DHT is equal to $D_{cost} = O(b \times \log n)$.

5. PERFORMANCE EVALUATION

The accomplishment of network simulator and the related architectures like Mesh- based PresenceCloud and Chord-based presence server architecture. Packet-level simulator allows performing tests up to 20, 000 users and 2,048 PS nodes. We relate two physical topologies to simulate Internet networks.

a) **King-topology:** It should be a real Internet topology From the King data set. Delay matrix of king data set is derived from Internet measurements using techniques.
b) **Brite-topology:** AS topology generated the BRITE topology generator [36] using the Waxman model where alpha and Beta. In addition, HS (size of one side of the plane) is set to 1,000 and LS (size of one side of a high-level square) is set to 100. The Brite-topology having 1, 000 nodes.

The computer-generated topology spaces every PS node at position on the King-topology or the Brite-topology, selected consistently at arbitrary. Simulations involve networks of less than 2, 048 PS nodes, we use a pair wise latency matrix derived from measuring the inter-PS node latencies. The average delay 77.4 milliseconds of King-topology and ninety six. Two milliseconds in the Brite- topology. The number of users is set to be 20, 000.

6. SIMULATION RESULTS

The user has to first weigh up and compare the three server architectures by taking into account the total buddy searching messages metric. Instantiated a server network of 256 PS nodes in our simulator, then a number of experiments to investigate the effect of scalability of PS nodes on involved searching messages. More specifically, we varied the user arrival rate is 100 to 8, 000 per second to explore the relation between user arrival rate and the total searching messages. The number of buddies is set to 100. Figure-6 depicts the total number of searching message transmissions during simulation time (1, 800 seconds) under various user arrivals rare.

The following figure shows that the total number of messages increases as per the arrival increases.

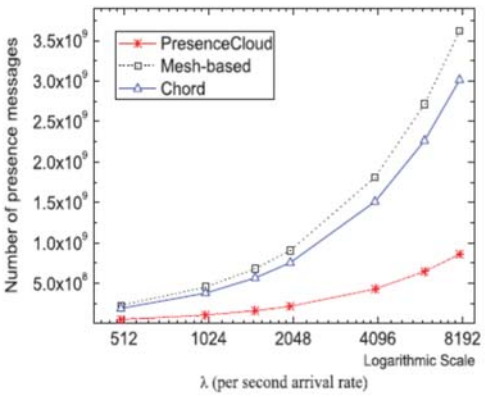


Figure-6. The total message transmissions during simulation time (1,800 s). (The x-axis of this figure is in logarithmic scale).

7. CONCLUTIONS

In this paper, we have represented Presence Cloud, architecture of scalable server that supports mobile presence services in large-scale social network services. PesenceCloud have low search latency and improve the performance of mobile presence services. We discussed in server architecture how the scalable problem arises and introduced the buddy-list search problem, this will be a scalability problem in the distributed server architecture of mobile presence services. In a simple mathematical model, the total number of buddy search messages increases significantly with the user arrival rate and the number of presence servers. Presence Cloud achieves most important performance gains in terms of the search cost and search satisfaction. In general, Presence Cloud is shown to be a scalable mobile presence service in large- scale social network services.

REFERENCES

[1] R.B. Jennings, E.M. Nahum, D.P. Olshefski, D. Saha, Z.-Y. Shae and C. Waters. 2012. A Study of Internet Instant Messaging and Chat Protocols. IEEE Network. 20(6): 16-21, July/August. 2006. Guides/p2pexplained.

[2] Z. Xiao, L. Guo and J. Tracey. 2007. Understanding Instant Messaging Traffic Characteristics. Proc. IEEE 27th Int'l Conf. Distributed Computing Systems (ICDCS).

[3] C. Chi, R. Hao, D. Wang, and Z.-Z. Cao. 2008. IMS Presence Server: Traffic Analysis and Performance Modeling. Proc. IEEE Int'l Conf. Network Protocols (ICNP).

[4] 2012. Instant Messaging and Presence Protocol IETF Working Group, rter.html.

[5] K.P. Gummadi, S. Saroiu and S.D. Gribble. 2002. King: Estimating Latency between Arbitrary Internet End Hosts. Proc. Second ACM SIGCOMM Workshop Internet measurement (IMW).

[6] A. Medina, A. Lakhina, I. Matta and J. Byers. 2001. BRITE: An Approach to Universal Topology Generation. Proc. ACM Ninth Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecomm. Systems (MASCOTS).

[7] R. Cox, A. Muthitacharoen, and R.T. Morris. 2002. Serving DNS Using a Peer-to-Peer Lookup Service. Proc. First Int'l Workshop Peer-to- Peer Systems (IPTPS).

[8] V. Ramasubramanian and E.G. Sirer. 2004. Beehive: 0(1) Lookup Performance for Power-Law Query Distributions in Peer-to-Peer Overlays. Proc. USENIX First Conf. Symp. Networked Systems Design and Implementation (NSDI).

[9] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan. 2003. Chord: A Scalable Peer-to-Peer Lookup Service for Internet. IEEE/ACM Tran. Networking. 11(1): 17-32.

[10] X. Chen, S. Ren, H. Wang and X. Zhang. 2005. SCOPE: Scalable Consistency Maintenance in Structured P2P Systems. Proc. IEEE INFOCOM.