Open access Journal International Journal of Emerging Trends in Science and Technology

Proactive Management and Monitoring of Mobile Devices in Social Networking Applications

Authors

TVN Prapulla Chandu[1], A. Vishnuvardhan[2]

[1]M.Tech. Programme, Student of Vasireddy Venkatadri Institute of Technology. Nambur (v), Guntur, Andhra Pradesh.

Email: *chandu2126@gmail.com*

[2]Asst Professor of Vasireddy Venkatadri Institute of Technology, Nambur(v), Guntur, Andhra Pradesh.

Email: *Harshanandhan2k14@gmail.com.*

Abstract

Now-a-days mostly mobile devices using social network applications have a part in mobile software. Social network applications utilize the particular component called mobile presence service. It collects and maintains each and every user's present status details. Some of the details like current status [online/offline], Global location and network address, and also updates the user's online status friends with the details. If presence frequently gets updates, it may lead to a scalability problem in a large-scale mobile presence service because the more number of messages distributed by presence servers. To notify the problem and enables mobile presence services to support large-scale social network applications, we proposed efficient and scalable server architecture, called Presence Cloud. When mobile device using people connect to network, Presence Cloud searches for the presence of his/her friends and notifies them of his/her status. For efficient and good presence searching, Presence Cloud arranges presence servers into a neat architecture, called quorum-based server-to-server architecture. To achieve small constant search latency, It also uses a directed search algorithm and a one-hop caching strategy. The total number of messages generated by the presence server when a user arrives is called the search cost; the time it takes to search for the arriving user's friend list is called the satisfaction level.

1 Introduction

Everyone can use presence-enabled applications through the Internet, mobile devices and cloud computing environments. The examples of these application social network applications/services, worldwide are Face book [3], Twitter [4], Foursquare, Google Latitude, buddy cloud [11] and Mobile Instant Messaging [1] (MIM).The people that participants are using different type of devices to connect with their friends by Social network services on Internet. They collect and notify the details about the presence of friends and their appearances and activities to interact with each other. These are possible by availability of mobile devices which will use the wireless mobile network technologies, social network services to share live happening from a long location to location.

In cloud computing environments, mobile presence service is an essential component of social network services. All mobile users will be maintained by mobile presence service in an up-to-date, which is the function of the mobile presence service. The presence information

includes information about a mobile user's global location, availability, activity, device capability, and preferences. The service must also join the user's ID to his/her current presence details, as well as retrieve and subscribe to changes in the presence details of the user's friends.

In social network services, each mobile user has a friend list chart, typically called a buddy list, which contains the contact details of other users that he/she wants to communicate with each other. The mobile device user's present status is send to all (broadcast) automatically to each person on the buddy list whenever he/she transits from one status to the other. Suppose whenever a user login to social network application through his/her mobile device, then the mobile presence service searches for the logged user's and notifies to every buddy list. Most presence services use server cluster technology to maximize a mobile presence service's search speed and minimize the notification time.

On the Internet currently, more than 500 million people use social network services. Thus, a scalable mobile presence service is deemed essential for future Internet applications. More Internet services have been deployed in paradigms and applications respectively distributed and cloud computing. For example, Google and Face book which they develop the services and spread among as many their distributed servers as possible to support and increase the huge number of users worldwide.

Before introducing the buddy-list search problem in distributed presence architectures in large-scale, we examine the server architectures of existing presence services. When a distributed presence service is overloaded with buddy search messages, then the buddy-list search problem is a scalability problem. The design of Presence Cloud, a scalable server-to-server architecture that can be used as a building block for mobile presence services. To avoid single point of failure, single presence server is not supposed to maintain service-wide global information about all users. Presence Cloud arranges presence servers into a architecture called quorum-based server- to-server architecture to facilitate efficient buddy list searching. The server overlay and a directed buddy search algorithm to achieve small constant search latency; and employs an active caching strategy that substantially reduces the number of messages generated by each search for a list of buddies.

Analyze the performance complexity of Presence Cloud and two other architectures, one is Mesh-based scheme and second is a Distributed Hash Table based scheme. Through the process of the simulation, we compare the performance of the three approaches in terms of the number of messages generated and the search satisfaction which we use to denote the search response time and the buddy notification time.

2 Related Work

Some IM systems, some form of centralized clusters to provide presence services. The AIM, Microsoft MSN and Yahoo Messenger are three most known IM systems.Skype, a popular voice over IP application, utilizes the Global Index (GI) technology to provide a presence service for users. Global Index is a multi tiered network to pology. Where each node collects and maintains full knowledge of all available users. It is difficult to determine GI technology is used exactly in Skype because it is not an open protocol.

Jabber is a well-known deployment of instant messaging technologies based on distributed structure and SMTP protocols has distributed architecture which will be captures by jabber.
By using distributed structures like Jabber, the output is a adaptable network of servers that can be scaled much higher than the enormous, centralized presence services. Recently, there is a growing amount of interest in how to design a peer-to-peer [6] SIP[7] which reduce maintenance costs, and prevent failures.P2PSIP users are organized in a DHT [8] system, which is easy

than in a centralized server and to maintain presence details.

Mobile services integrate the presence services and some mobile devices also support mobile presence services. In a weakly consistent scheme to reduce the number of updating messages in mobile presence services of IP Multimedia Subsystem (IMS). However, it also suffers scalability problem since it uses a central SIP server to perform presence update of mobile users.

3 The Problem Statement

We assume the geographically distributed presence servers to form a server-to-server 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∈ 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 overlay enables its PS nodes to communicate with one another by forwarding messages through other PS nodes in the server overlay. Indicate a collect of the mobile users in a presence service as U = {u1; : : : ; ui; : : : ; um}, where 1 ≤ i≤ mand m is the number of mobile users. A mobile user up communicates with one PS node for search other user's presence information. That PS node will inform to the other mobile users of his/her arrival.

4 Design of Presence Cloud

A new design of mobile presence services is needed to address the buddy list search problem for mobile social network applications. Presence Cloud is used to query the system for buddy list searches. It will construct and maintain distributed server architecture. Presence Cloud consists of 3 most important building blocks that are run across a set of presence servers. The 3 building blocks of Presence Cloud are summarized below:

Presence Cloud server overlay organizes presence servers based on the concept of grid quorum system. So, the server overlay of Presence Cloud has a balanced load property and a two-hop diameter with O (√n) node degrees, where the count of presence servers is indicated by n.

One-hop caching strategy is used to reduce the accelerate query speed and number of transmitted messages. Immediate neighbors which offer a buddies and which will be collected by presence servers that has caches to store.

Directed buddy search is based on the directed planning for search. On average it provides little search latency.

4.1 Presence Cloud Overview

The Presence Cloud is used to construct scalable server architecture for mobile presence services, and can be used to efficiently search the desired buddy lists.

Fig.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 connect and authenticates himself/herself to the mobile presence services, the user is directed to one of Presence Servers in the Presence Cloud by using the Secure Hash Algorithm, such as SHA-1. The mobile user opens a TCP connection to the Presence Server (PS node) for control message transmission. After the control channel is built, the connected PS node receives a request from mobile users for his/her buddy list searching.

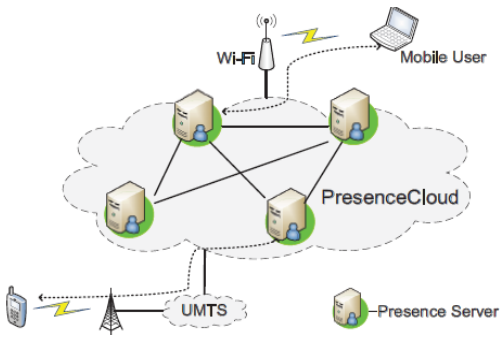


Fig.1. An overview of Presence Cloud

Our Presence Cloud returns the presence detail of the desired buddies to the mobile user after doing an efficient searching operation.

4.2 Presence Cloud Server Overlay

The Presence Cloud server design model algorithm arranges the PS nodes and which provides a good low-diameter overlay property. When the PS nodes into a server-to-server structure. That ensures that a PS node only needs two hops to reach any another PS nodes. The concept of grid quorum system used by our Presence Cloud, where a PS node only maintains a set of PS nodes of size O (√n), where in mobile presence services n is the number of PS nodes. By using a grid quorum system in a Presence Cloud system will construct the PS list, that PS node has a group of PS nodes.

Fig. 2 for n=9. The size of a grid quorum is ⌈ √n⌉ ×⌈ √n⌉ . When a PS node joins the server overlay of Presence Cloud, it receives an ID in the matrix structure, find its point of place in the matrix and obtains its PS list by contacting a root server1. On the ⌈ √n⌉ × ⌈ √n⌉ grid, a PS 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. Fig. 2 illustrates an example of Presence Cloud, in which the grid quorum is set to ⌈ √9⌉ × ⌈ √9⌉ . In the Fig. 2, 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.

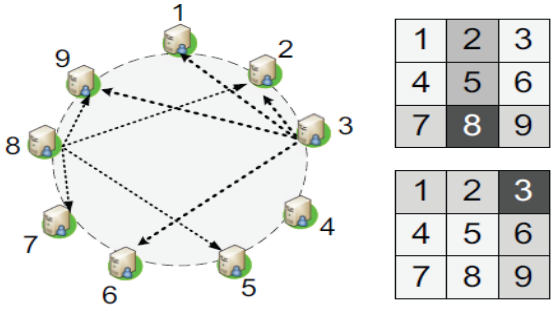


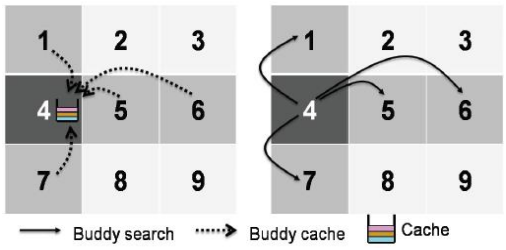
Fig.2. A Perspective of Presence Cloud Server Overlay

4.3 One-hop Caching

The presence of users, the caching strategy to replicate presence information of users. In Presence Cloud, each PS node maintains a user list of presence information of the attached clients [users].Presence Cloud used for caching the client list of each node in its PS list.

At most one hop away from itself the PS nodes replicate the user.

When neighbors establish connections then periodically cache is updated with its neighbors. Therefore, when a query is received by the PS node, it can respond with all matches from its own user list and provide matches from its caches of all neighbors which has user lists. Whenever the user will log off then automatically the Presence Cloud inform to all servers and this information will updated to all user list. This information transmits them by neighboring PS nodes.



4.4 Directed Buddy Search

The buddy list searching algorithm of Presence Cloud [14] coupled with the two-hop overlay and one-hop caching [13] strategy ensures that Presence Cloud can typically provide swift responses for a large number of mobile users. In servers network arrange PS nodes, which will handle queries and reduce the network traffic without impact on the search results by using the one-hop search. Second, the neighbors user lists can be maintained by using the one-hop and by these finding buddies will increase and improve the response time. For saving the cost we have to use piggybacked for broadcasting messages in buddy search.

5 Cost Analysis

Cost analysis of the communication cost of Presence Cloud [14] in terms of the number of messages required to search the buddy information ofa mobile user.
The buddy-list search problem can be solved by a searching algorithm called brute-force [15]. In the mobile presence service brute-force [15] simply finds and identifies all the PS nodes.
The communication cost [16] of searching buddies and replicating presence information can be formulated as Mcost= Q Mesh + RMesh, where R Meshis the communication cost of replicating presence information to all PS nodes. Accordingly, we have M cost =O (n).

In the analysis of Presence Cloud, we assume that all the PS nodes will access all the mobile users. Here, the search cost of analysis denoted as Qp, which is $2 \times(\lceil \sqrt{n} \rceil - 1)$ messages for both searching buddy lists and replicating presence, because these both can be combined into one single message, the replicating cost for communicating, Rp= 0. It is straight forward to know that the communication cost of searching buddies and replicating presence information in Presence Cloud is P cost= $Qp = 2\times ([\sqrt{n}] - 1)$.

6 Performance Evaluation

Our implementation we use Java to write network simulator [17,18] and the related architectures. Some of the architectures are Mesh based, Presence Cloud and Chord-based presence server architecture. To perform tests up to 20,000 users and 2,048 PS nodes the simulator used is packet-level, which make our experiments difficult the data generated by the simulator will fit for small time in RAM.

6.1 Performance Metrics

The following three metrics measure the server architectures performance in the context.

1) Total Searching Messages: This represents during the simulation time the total number of messages transferred between the query initiator and the other PS nodes.
2) Average Searching Messages for each-arrived user: The number of searching messages used for each arrived user.

7 Conclusion

In this paper we presented about the scalability of the server structure and server capable of use PS nodes which will support quires. The server which introduced the buddy-list problem in distributed server architecture of mobile presence. The server notifies and updates the buddy list; when the mobile users are connect to network by social service applications and when they log off. In large-scale social network services Presence Cloud are used has a a scalable mobile presence service

References

1. Instant messaging and presence protocol ietf working group http://www.ietf.org/html.charters/impp-charter.html.
2. Extensible messaging and presence protocol it working grouphttp://www.ietf.org/html.charters/xmpp-charter.html.
3. Facebook, http://www.facebook.com.
4. Twitter, http://twitter.com.
5. Jabber, http://www.jabber.org/.
6. Peer-to-peer session initiation protocol working group,http://www.ietf.org/html.charters/p2psip-charter.html.
7. K. Singh and H. Schulzrinne,"Peer-to-peer internet telephony using sip," Proc. of ACM NOSSDVA, 2005.
8. S. A. Baset, G. Gupta, and H. Schulzrinne,"Openvoip: An open peer-to-peer voip and IM system," Proc. of ACM SIGCOMM, 2008.

9. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler,"Sip: Session initiation protocol," RFC 3261, 2002.
10. Open Mobile Alliance, OMA instant messaging and presence service, 2005.
11. Buddycloud, http://buddycloud.com.
12. Mobile instant messaging, http://en.wikipedia.org/wiki/Mobile instant messaging.
13. M.A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," arXiv preprint ArXiv: 1209.5807, 2012.
14. Extensible Messaging and Presence Protocol IETF Working Group, "http://www.ietf/html.charters/xmppcharter.Service,"2005.
15. Michael T. Goodrich and Roberto Tamassia, Algorithm Design, 2002, John Wiley and Sons, Inc.
16. B. Awerbuch, A. Baratz, and D. Peleg, "Cost-sensitive analysis of communication Protocols", Proc. ACM PODC, pp.177 -187 1990 .
17. M. Law and W. D. Kelton, Simulation modelling and analysis, third ed. New York: McGraw-Hill, 2000.
18. Schmeiser, "Simulation output analysis: A tutorial based on one research thread," presented at the 2004 Winter Simulation Conference, December 5-8, 2004, pp. 162-170.