A Scalable Server Architecture for Mobile Presence Services in Social Network Applications

A. Radha Krishna*
K. Chandra Sekharaiah**

Abstract

The use of Social network is becoming gradually more popular on mobile devices. The important component of a social network application is a mobile presence service because it maintains each mobile user's presence information, such as the current status (online/offline), GPS location and network address, and also updates the user's online friends with the information repeatedly. If presence updates occur frequently, the enormous number of messages distributed by presence servers may lead to a scalability problem in a large-scale mobile presence service. We propose an efficient and scalable server architecture, called PresenceCloud, to address the problem, which enables mobile presence services to support large-scale social network applications. PresenceCloud searches for the presence of his/her friends and notifies them of his/her arrival when a mobile user joins a network. For efficient presence searching presenceCloud categorizes presence servers into a quorum-based server-to-server design .Directed search algorithm and a one-hop caching strategy to achieve small constant search latency is also controls by it.The performance of PresenceCloud we scrutinize in terms of the search cost and search satisfaction level. The search cost is characterized as the total number of messages generated by the presence server when a user arrives; and search approval level is defined as the time it takes to search for the arriving user's friend list. The results of simulations demonstrate that PresenceCloud achieves performance gains in the search cost without compromising search satisfaction.

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

Introduction

Motivation

Mainly the enthusiasm for doing this project is an interest in undertaking a challenging project in an interesting area of research (Networking). The prospect to learn about a new area of computing not covered in lectures.

Problem Definition

The reason of the ubiquity of the Internet, mobile devices and cloud computing environments can afford presence-enabled applications, *i.e.*, social network applications/services, worldwide. Facebook [1], Twitter [2], Foursquare [3], Google Latitude [4], Buddy-Cloud [5] and Mobile Instant Messaging (MIM) [6], are examples of presence-enabled applications that have developed quickly in the last decade. Social –Networks services are altering the way in which participants connects to their friendsand develop the information about the status of participants including their appearances and activities to interact with their friends. Social network services enable participants to share live experiences instantly across great distances with the social networks like Facebook,twitteretc by the consumption of wireless mobile network technologies. Mobile devices will become more powerful, sensing and media capture devices in the future and we can expect that the social networks services will be the next generation of mobile Internet applications.

Aamobile presence service an important component of social network services in cloud computing environments and to maintain an up-to-date list of presence information of all mobile users is the key function . The presence information consist of details about a mobile user's location, availability, activity, device capability, and preferences. The service must

A. Radha Krishna*
Vasjrs2004@gmail.com

K. Chandra Sekharaiah**
chandrasekharaiahk@gamil.com

also connect the user's ID to his/her current presence information, as well as repossess and subscribe to changes in the occurrence information of the user's friends.

Each mobile user has a friend list, typically called a buddy list (contact information of other user) in social network services. The mobile user's status is broadcast automatically to each person on the buddy list whenever he/she transits from one status to the other.

It is expected that the number of mobile presence service users will increase significantly in the near future due to the development of social network applications and mobile network capacity. Thus, a scalable mobile presence service is considered essential for future Internet applications.

Distributed paradigms as well as cloud computing applications have been organized by many Internet services in the last decade, .So we explore the relationship between distributed presence servers and server network topologies on the Internet, and intend an efficient and scalable server-to-server overlay architecture called Presence Cloud to develop the efficiency of mobile presence services for large-scale social network services.

Distributed presence architectures in large-scale geographically data centers first we examine the server architectures of existing presence services, and introduce the buddy-list search problem . The *buddy-list search problem* is a scalability problem thatoccurs when a distributed presence service is teeming with buddy search messages.

Objective of the Paper

In this project, the main objective is to propose a Presence Cloud is scalable server-to-server architecture , that can be used as a building block for mobile presence services. The validation 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.

Presence Cloud organizes presence servers into a quorum-based server-to-server architecture to make easy efficient buddy list searching. To accomplish small constant search latency it also influences the server overlay and a directed buddy search algorithm; and employs an active caching strategy that reduces the number of messages generated by each search for a list of buddies. Presence Cloud and two other architectures, a Mesh-based scheme and a Distributed Hash Table (DHT)-based scheme We analyzed the presentation complexity .

Through duplications, we also compare the performance of the three approaches in terms of the number of messages generated and the search response time and the buddy notification time. The results demonstrate that Presence Cloud achieves major performance gains in terms of reducing the number of messages without sacrificing search satisfaction thus can support a large-scale social network service distributed among thousands of servers on the Internet.

The role of this paper is threefold.

- First, the original architecture for mobile presence services is Presence Cloud.
- The second role is that we consider the scalability problems of distributed presence server architectures, and characterize a new problem called the buddy-list search problem.
- Finally we consider the concert complexity of Presence- Cloud and different designs of distributed architectures, and evaluate them empirically to demonstrate the advantages of Presence Cloud.

Literature Survey

A study of internet instant messaging and chat protocols

Instant messaging (IM) and network chat communication have seen an immense rise in popularity over the last several years. This scrutiny helps bridge this gap by providing an overview of the available features, functions, system architectures, and protocol specifications of the three most popular network IM protocols: AOL Instant Messenger, Yahoo! Messenger, and Microsoft Messenger.

The technical characteristics of commercial Internet IM and chat protocols, due to the closed proprietary nature of these systems. We have presented a taxonomy of different feature and functions supported by most common systems, namely, AOL Instant Messenger (AIM), Yahoo Messenger (YMSG), and MSN Messenger (MSN). AIM sup- ports the most features and thus is the most complex network IM protocol. This may be a result of the fact that AIM has the largest user base of the three systems.

Perceptive instant messaging traffic characteristics

Due to its quick response time Instant messaging (IM) has become increasingly popular, its ease of use, and option of multitasking. It is estimated that there are several millions of instant messaging users who use IM for various purposes: simple requests and responses, scheduling face to face meetings, or just to check the availability of colleagues and friends and this leads relatively small traffic volum . All major instant messaging systems route text messages through central servers this facilitating firewall traversal and gives IM companies more control, at the IMservers it creates a potential bottleneck. This is especially so for large instant messaging operators with tens of millions of users and during flash crowd events.

Traces due to privacy concerns is another reason in getting access to instant messaging. AOL Instant Messenger (AIM) and MSN/Windows Live Messenger we analyze the traffic of two popular instant messaging systems. We found that most instant messaging traffic is due to presence, hints, or other extraneous traffic.During the overload the IM server can protect the instantaneous nature of the communication by dropping extraneous traffic. We also found that the social network of IM users does not follow a power law distribution. It can be characterized by a Weibull distribution. Our analysis lean-to light on instant messaging system design and optimization and provides a scientific basis for instant messaging workload generation. In the future, we plan to extend the scope of our study by analyzing traces from other user population.

A token-bucket based notification traffic control mechanism for IMS presence service

The service to inform about the specified state of another user is called as Presence service. The next generation applications Instant messaging, push- to-talk and web2.0 are became the key enable for the Presence service.Heavy signaling load on IP multimedia subsystem (IMS) network are caused by the notification traffic of presence service . A Token-bucket based Notification Traffic Control (TNTC) mechanism, which is an application layer solution deployed at the presence server. The aim of TNTC is Upgrading valid access probability while controlling the notification traffic. Extensive replications established that TNTC can effectively control notification traffic and perform better than the existing schemes in terms of valid access probability and update arrival rate.

TNTC which is a token-bucket based notification traffic control mechanism for IMS presence service. The different notification traffic control policy depending on the user class to improve the QoS of presence service.

Extensible messaging and presence protocol (xmpp): Instant messaging and presence describes instant messaging (im), the most common application of xmpp

First effort in creating an open standard for instant messaging and presence information is the Extensible Messaging and Presence Protocol (XMPP). XMPP was introduced by the Jabber Software Foundation (JSF) and formalized in the IETF. Numerous extensions called Jabber Enhancement Proposals (JEP) have evolved through subsequent work. The XMPP/Jabber technology has achieved big success, especially since the IETF approval of the core protocols. . It led to significant implementations, major deployments and renewed activity by open-source projects and commercial software developers.

The XMPP/Jabber still has to face challenge from competing technologies, such as SIMPLE. More effort on the gateway development to interoperate with other IM systems.

Peer-to-peer internet telephony using sip

P2P systems inherently have high scalability, robustness and fault tolerance because of there is no centralized server and the network self-organizes itself. This is achieved at the cost of higher latency for locating the resources of interest in the P2P overlay network. Internet telephony can be seen as an application of P2P architecture where the participants form a self-organizing and communicate with other participants.Based on the IP telephony system ,a pure P2P architecture for the Session Initiation Protocol (SIP) is proposed.The P2P-SIP architecture supports basic user registration,call set up ,offline message delivary, voice/video mail ,multi-party conferencing, firewalls,Network Address Transalation (NAT) traversal and security.

The pure P2P architecture for SIP telephony provides reliability ,scalability and interoperability with existing SIP infrastructure.The cost5 of increased call setup latency is the advantage. By using Chord as the underlying DHT we can propose various design alternatives.

More simulations may not add any research value to the existing simulation results if the implementation is based on the Chord. Large scale application level multicast conferencing using P2P, distributed reputation system for peers, and PSTN interworking related issues such as authentication and accounting are advanced services need more work. The load on public super-nodes is reduced by allowing an internal node inside a firewall and NAT to become a super-node.

Problem statement for sip/simple

The number of contributions between domains quickly becomes an issue. This document examines the scaling issue and concludes that additional optimizations are necessary. The traffic up to the 50% can reduce by these calculations show that the current suggested optimizations although effective in some circumstances and there is still a very high volume traffic occur in SIP deployments of presence. In order to make the deployment of SIP presence more effective additional optimizations are needed.

OpenVoIP: An open peer-to-peer VoIP and IM system:

This demo presents OpenVoIP, a 500 node open peer-to-peer VoIP and IM system running on Planet Lab. The three key aspects of OpenVoIP's design are its ability to use any DHT or unstructured peer-to-peer protocol for directory service, the use of intermediate peers with unrestricted connectivity to relay signaling and media traffic between peers behind NAT and firewalls, and a diagnostic system integrated with Google maps for graphical monitoring. The demo will show these aspects of OpenVoIP and provide approaching on issues and related problems in building such a system.

A weakly consistent scheme for IMS presence service

Presence service for Universal Mobile Tele-communications System (UMTS) is offered by IP Multimedia Core Network Subsystem (IMS).In IMS, the updated presence information is done by the presence server responsible for notifying an authorized watcher .If any upadetes occurs the presence server will generate many notifications and it consists of weakly schema to reduce the notification traffic and delayed timer is defined for to control the notification rate.

Failover and load sharing in SIP telephony

For high service availability and scalability to the relatively new IP telephony context some of the exisiting web server redundancy techniques are applied.

In this various failover and load sharing methods for registration and call routing server based on the Session Initiation protocols (SIP) are compared.

The SIP server failover techniques based on the clients, DNS (Domain Name Service), database replication and IP address takeover, and the load sharing techniques using DNS, SIP identifiers, network address translators and servers with same IP addresses.

We implemented the failover mechanism by using the SIP proxy and registration server and the open source MySQL database.

Network co-location of the servers does not required for the DNS SRV to do redundancy.DNS itself is replicated and Combining DNS with the identifier-based load sharing can scale to large user base.

More work to do failover and load sharing in the middle of the call without breaking the session is need

Call stateful services such as voicemail, conferencing and PSTN interworking. It is difficult for the individual server failover to detect and recovery of wide area path outage. Instead of statically configuring the redundant servers, it will be useful if the servers can automatically discover and configure other available servers on the Internet.

Chord: A scalable peer-to-peer lookup service for internet

- Peer -to-peer applications is to efficiently locate the node is a fundamental problem that deals with that stores a particular data item. Chord, a distributed lookup protocol that addresses this problem.
- Data location can be easily implemented on top of Chord by associating a key with each data item, and storing the key/data item pair at the node to which the key maps.
- Chord can answer queries even if the system is continuously changing and easily adapts the nodes joining and leaving of the system.
- Chord is scalable.
- The Chord protocol solves this challenging problem in decentralized manner.
- Only O(log n) messages are needed for updating the routing information for nodes leaving and joining.
- Simplicity, provable correctness, provable performance and parallel node arrivals and departures are the attractive features of Chord.

Experimental results confirm that Chord scales well with the number of nodes, recovers from large numbers of simultaneous node failures and joins, and answers most lookups correctly even during recovery.

Chord will be a valuable component for peer-to-peer, large-scale distributed applications, large-scale distributed computing platforms.

Algorithm 2.1PresenceCloud Stabilization algorithm

```
1: /* periodically verify PS node n's pslist*/
2: Definition:
3: pslist: set of the current PS list of this PS node, n
4: pslist[].connection: the current PS node in pslist
5: pslist[].id: identifier of the correct connection in pslist
6: node.id: identifier of PS node node
7: Algorithm:
8: r •! Sizeof(pslist)
9: for i = 1 to r do
10: node •! pslist[i].connection
11: if node:id 8= pslist[i].id then
12: /* ask node to refresh n's PS list entries */
13: findnode •! Find CorrectPSNode(node)
14: if findnode= nil then
15: pslist[i].connection •! RandomNode(node)
16: else
17: pslist[i].connection •! findnode
18: end if
19: else
20: /* send a heartbeat message */
21: bfailed •! SendHeartbeatmsg(node)
22: if bfailed= true then
23: pslist[i].connection •! RandomNode(node)
24: end if
25: end if
26: end for
```

Algorithm 2.2 Buddy Search Algorithm

For each buddy list searching operation, the directed buddy search of PresenceCloud retrieves the presence information _ of the queried buddy list at most onehop.

Proof: This is a direct consequence of Lemma 2 and

Lemma 3. Before presenting the directed buddy search algorithm, lets revisit some terminologies which will be used in the algorithm.

B = {b1; b2; : : : ; bk}: set of identifiers of user's buddies

B(i): Buddy List Search Message be sent to PS node i

b(i): set of buddies that shared the same grid ID i

Sj: set of pslist[]:id of PS node j

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 bi 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 8= 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 "Sj. Here PS node j is the intersection node of Sq)"Sg. And sends the new *B*(*j*) to PS node *j*.

Following, we describe an example of directed buddy search in PresenceCloud. When a PS node 4 receives a Buddy List Search Message, *B* = *{*1*;* 2*;* 3*;* 4*;* 5*;* 6*;* 7*;* 8*;* 9*}*, from a mobile user, PS node 4 first searches its local *user list* and the buddy cache, and then it responds these.

Existing System

In this we describe previous researches on presence services, and survey the presence service of existing systems. To provide presence services Well known commercial IM systems influence some form of centralized clusters. Jennings III *et al.* presented a taxonomy of different features and functions supported by the three most popular IM systems, AIM, Microsoft MSN and Yahoo! Messenger. The authors also provided an overview of the system architectures and observed that the systems use client-server-based architectures. Skype, a popular voice over IP application, GI is multi-tiered network architecture. Since Skype is not an open protocol, it is difficult to determine how GI technology is used exactly. Moreover, Xiao *et al.* analyzed the traffic of MSN and AIM system. They found that the presence information is one of most messaging traffic in instant messaging systems. In, authors shown that the largest message traffic in existing presence services is buddy NOTIFY messages.

Proposed System

To remove the centralized server, reduce maintenance costs, and prevent failures in server-based SIP deployment the P2PSIP has been proposed.P2PSIP clients are organized in a DHT system to maintain presence information. The presence service architectures of Jabber and P2PSIP are distributed, the *buddy-list search problem* we defined later also could affect such distributed systems.

It is noted that few articles in discuss the scalability issues of the distributed presence server architecture. Saint Andre analyzes the traffic generated as a result of presence information between users of inter-domains that support the XMPP. Houri*et al.* Show that the amount of presence traffic in SIMPLE can be extremely heavy, and they analyze the effect of a large presence system on the memory and CPU loading. Those works in study related problems and developing an initial set of guidelines for optimizing inter-domain presence traffic and present a DHT-based presence server architecture.

Recently, presence services are also incorporated into mobile services. For example, 3GPP has defined the integration of presence service into its specification in UMTS. It is based on SIP protocol, and uses SIMPLE to manage presence information. Recently, some mobile devices also support mobile presence services. For example, the Wireless Village consortium and was united into Open Mobile Alliance (OMA) IMPS in 2005 developed the Instant Messaging and Presence Services (IMPS). In, Chen *et al.* proposed 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. In IMS-based presence service, authors presented the server scalability and distributed management issues.

Conclusion

In this paper, we have presented PresenceCloud, a scalableserver architecture that supports mobile presence servicesin large-scale social network services. We have shown that enhancesthe performance of mobile presence services by PresenceCloud accomplishes low search latency. In addition,we discussed the scalability problem in server architecturedesigns, and introduced the buddy-list search problem,which is a scalability problem in the distributed serverarchitecture of mobile presence services. Through a simple Mathematical model, we show that considerably with the userarrival rate and the number of presence servers bythe total number ofbuddy search messages increases. The resultsof simulations demonstrate that PresenceCloud achievesmajor performance gains in terms of the search cost andsearch satisfaction. Overall, PresenceCloud is shown to bea scalable mobile presence service in large-scale socialnetwork services.

References

1. Facebook, http://www.facebook.com.
2. Twitter, http://twitter.com.
3. Foursquare http://www.foursquare.com.
4. Google latitude, http://www.google.com/intl/enus/latitude/intro.html.
5. Buddycloud, http://buddycloud.com.
6. Mobile instant messaging, http://en.wikipedia.org/wiki/Mobile instant messaging.
7. 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.
8. Gobalindex, http://www.skype.com/intl/en-us/support/user-guides/ p2pexplained/.
9. Z. Xiao, L. Guo, and J. Tracey, "Understanding instant messaging traffic characteristics," *Proc. of IEEE ICDCS*, 2007.
10. C. Chi, R. Hao, D. Wang, and Z.-Z. Cao, "Ims presence server: Traffic analysis and performance modelling," *Proc. of IEEE ICNP*, 2008.
11. Instant messaging and presence protocol ietf working group http://www.ietf.org/html.charters/impp-charter.html.
12. Extensible messaging and presence protocol ietf working group.http://www.ietf.org/html.charters/xmpp-charter.html.
13. Sip for instant messaging and presence leveraging extensions ietf working group. http://www.ietf.org/html.charters/simplecharter.html.
14. P. Saint-Andre., "Extensible messaging and presence protocol (xmpp): Instant messaging and presence describes instant messaging (im), the most common application of xmpp," *RFC 3921*, 2004.
15. B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, "Session initiation protocol (sip) extension for instant messaging," *RFC 3428*, 2002.
16. Jabber, http://www.jabber.org/.