International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
NCICN-2015 Conference Proceedings

Presence Server Architecture Scalability and Security

Vimitha R Vidhya Lakshmi[1*], Anju J[2]
[1*]PG Scholar, Dept. of computer science and engineering
[2]Assistant Professor, Dept. of computer science and engineering
TKM Institute of Technology
Kollam, India

***Abstract*— Presence enabled applications have been growing rapidly. A mobile presence Server contains presence information of all mobile users. Each mobile user has a friend list (buddy list) to which the user's status is broadcasted each time. Server cluster technology is used in most presence services. It is expected that the number of mobile presence service users will increase in the near future. Therefore, a scalable mobile presence server is essential. For this a scalable server-to-server architecture called PresenceCloud is proposed. The buddy list search problem arises when the distributed presence server is overloaded with buddy search messages. Thus quorum based server-to-server architecture is used for efficient buddy list searching. Here one hop caching strategy and Directed buddy Search is used which reduces the number of transmitted messages and search latency respectively. Finally, security is provided by SSL protocol.**

Index Terms*— Mobile cloud computing, Presence cloud, Buddy search, Quorum, SSL.

I. INTRODUCTION

Mobile computing is a technology that allows transmission of data, through a computer, without having to be connected to any fixed physical link. Cloud computing is a new emerging technology which involves deployment of groups of remote servers and software networks that allow centralized data storage and online access to computer services or resources. Mobile Cloud computing is a new platform which combines the above two technologies. Here the heavy lifting tasks such as data processing and data storage takes place outside the mobile devices in the cloud. There are many applications for Mobile Cloud computing, among them one such application is social network applications.

Social network applications/ services like Twitter, Facebook, Google Latitude, Foursquare and Mobile Instant Messaging (MIM) are some examples of presence-enabled applications. A mobile presence server is an essential component of social network services in cloud computing environments. The main function of a mobile presence server is to maintain an updated list of presence information of all mobile users. This presence information includes details like mobile user's location, availability, activity, mood etc. Each mobile user has a friend list or buddy list, which contains the contact information of other users that he/she wants to communicate with. The mobile user's status is broadcasted automatically to each person on the buddy list whenever he/she transits from one status to the other. Therefore, an efficient, scalable and secure server-to-server overlay architecture called PresenceCloud is proposed in-order to improve the efficiency of mobile presence services for large-scale social network services.

Mobile presence services are prone to scalability problems. One such problem is buddy list search problem which can be solved by using this scalable presence server architecture. PresenceCloud organizes presence servers into a quorum-based server-to-server architecture to facilitate efficient buddy list searching. The server overlay and a directed buddy search algorithm is used to achieve small constant search latency and also employs an active caching strategy that substantially reduces the number of messages generated by each search for a list of buddies.

PresenceCloud security is provided by Secure Sockets Layer (SSL) protocol. SSL protocol consists of two phases: handshake phase and data transfer phase. During the handshake phase, the client and server determine secret-key by using a public-key encryption algorithm. During the data transfer phase, both sides use this secret key to encrypt and decrypt data in successive data transmissions.

The next section is about related works, in section III the system model of this proposed system is discussed in detail, section IV is about performance evaluation and finally section V is the conclusion.

II. RELATED WORKS

The Over the last several years Instant Messaging (IM) and Internet chat communication have been growing widely. An overview of the features, functions and system architectures of the three most popular IM networks: AOL Instant Messenger (AIM), Yahoo Messenger (YMSG) and Microsoft Messenger (MSN) have been discussed in [1]. The text messages of all major IM systems are routed through central servers. Thus when the number of users increases to large extent it creates a potential bottleneck at the instant messaging servers. The traffic characteristics of two popular IM systems: AIM and MSN are discussed in [2]. Most IM traffic is due to presence or hints.

The sharing of basic presence information can result in a large volume of traffic as users log on or off frequently, especially for users with large numbers of contacts. The volume is increased when user's sub-states such as "away" and "do not disturb" are shared and the volume is increased to a greater extend if information such as device capabilities, geo-location, mood, activity even the music to which a user is listening is shared. This traffic can be reduced to smaller extent by compressing the XML streams. However, the above suffers from scalability problem like buddy list searching problem.

Presence directory service [3] is an essential component of an IM system. P2Dir maintains an updated list of presence information of all the mobile users. Here the design of P2Dir, a distributed peer-to-peer (P2P) presence directory organizes the DS nodes into a 2-hop P2P overlay for efficient buddy searching using Kelips system. In [4] a scalable server architecture called PresenceCloud is introduced. This quorum based server-to-server architecture is used for efficient buddy list searching. Here one hop caching strategy and Directed buddy Search is used which reduces the number of transmitted messages and search latency respectively. Thus scalability problem can be reduced to a certain extend.

A simple way to improve the presence service using Presence Network Agent is proposed in [5]. A new architecture of intelligent server management framework based on Extensible Messaging and Presence Protocol (XMPP), is proposed in [6].

As wireless communication increases the need for security also increases. The insecurity of wireless networks is discussed in [7]. The security and privacy issues of multimedia services are studied in [8]. In [9], [10] SSL is used as a security protocol for providing secure e-commerce transaction over the web. Many encryption algorithms, authentication procedures and working of SSL is explained here. Two truthful mechanisms are used to provide secure communication in [11]. [12] is about the study of energy consumption characteristics of SSL protocol. Security in Mobile Cloud computing is discussed in [13] by proposing lightweight schemes and algorithms.

[14] is about the secure protection between users and the mobile media cloud. Here secure sharing and watermarking schemes are used to protect user's data in the media cloud.

III. SYSTEM MODEL

A. Overview of PresenceCloud

A mobile user makes a data connection to PresenceCloud via 3G or Wi-Fi services. Then the user is directed to one of the presence server in PresenceCloud by using a secure hash algorithm. Then by opening a TCP connection mobile user request for his/her buddy list searching through this control path. Figure 1 shows the overview of PresenceCloud.

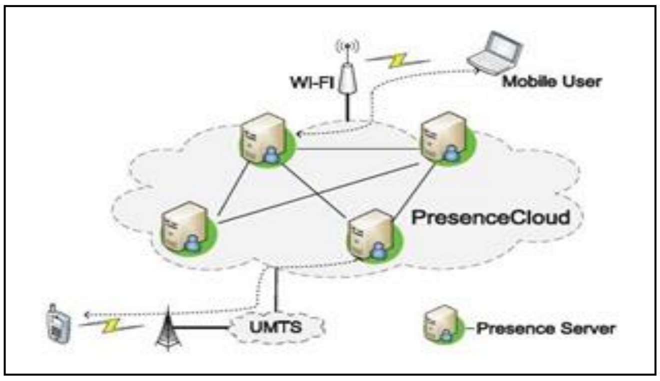

Fig. 1. Overview of PresenceCloud

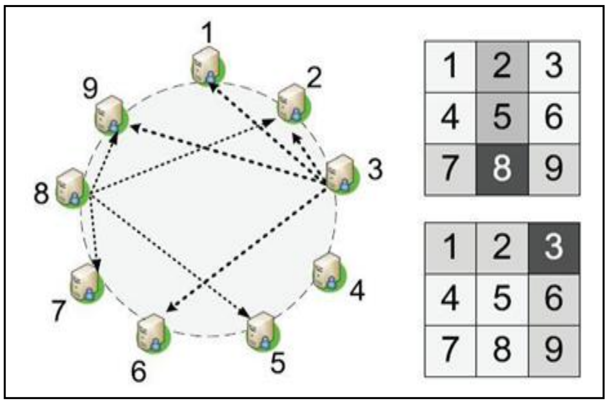

Fig. 2. PresenceCloud server overlay.

B. Design of PresenceCloud

PresenceCloud consists of three key components:

- PresenceCloud server overlay: PresenceCloud server overlay construction algorithm is based on low diameter property which ensures that a presence server (PS) node only needs two hops to search any other presence server nodes. Figure 2 shows the PresenceCloud server overlay. PresenceCloud is based on concept of grid quorum system where a presence server node maintains only a set of presence server nodes called presence server list of size O(√ n) where n is number of presence server nodes in mobile presence services. When a presence server node joins the server overlay it gets an ID in the grid and locates its position in the grid by contacting the root server. The presence server list (pslist) of a node is its corresponding row and column in the grid. In figure 2 we can see that for presence server node 8 its pslist contains the presence server nodes 2, 5, 7 and 9. For presence node 6 its pslist contains presence server nodes 3, 4, 5 and 9. Therefore 8 can reach 6 through presence server node 5 or 9 because 5 and 9 are the presence server nodes that are present in the intersection set of above two pslists. Thus we can say that any presence server node can reach any other presence server node in two hops i.e. route 8→ 5→ 6 or 8→ 9→ 6.
- One-hop caching strategy: In PresenceCloud each presence server maintains a user list which contains the presence information of other user nodes attached to it and is responsible for caching the user list of each node in its presence server list. This improves the efficiency of search operation.
- Directed buddy search: Directed searching technique is used here. One hop caching increases the chances of finding buddies thus minimizes the searching response time.

C. Security

PresenceCloud security is provided by SSL protocol. SSL protocol is used to provide a secure connection to transmit

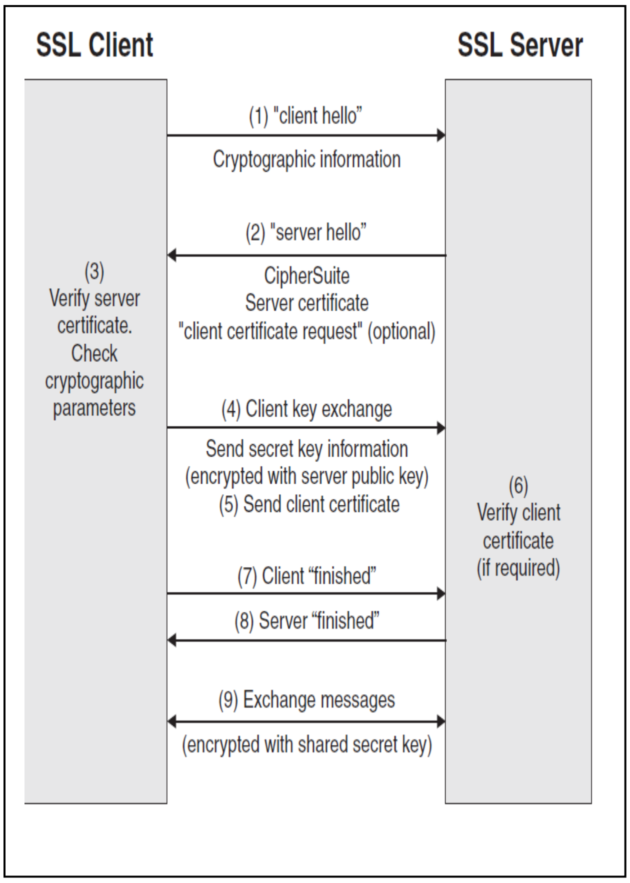
SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3)
Verify server certificate. Check cryptographic parameters
(4) Client key exchange
Send secret key information
(encrypted with server public key)
(5) Send client certificate
(6)
Verify client certificate
(if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Fig. 3. Working of SSL protocol

private data online. The stepwise working of SSL protocol is shown in Figure 3.

Here at handshake phase, first the client sends a hello message to server. This hello message contains cryptographic information like encryption algorithms and message authentication algorithms that are supported by client. This cryptographic information is called cipher suite. As the next step the server selects any one cipher suite and sends back a hello message to client with selected cipher suite. Server then sends server certificate to client in-order to prove its identity along with server's public key. Client certificate request from server is optional. This server certificate is then verified at the client. Certificates are issued by the certificate authority (CA). Certificates are obtained by the server by sending server's public key to CA. Then the certificate fingerprint is created by CA using message digest algorithm. Certificate signature is created by encrypting the fingerprint with CA's private key. The client first uses CA's public key to decrypt the signature in-order to get fingerprint and also computes the certificate's fingerprint independently. Both these fingerprints are checked for matching. Thus client validates server certificate.

The fourth step is that the client sends the secret key to server by encrypting it with server's public key. If server had requested client's certificate then client certificate is send and verified at the server then. The server then decrypts and gets the secret key send by client. As last step of handshake phase the client and server exchange finish message.

Next phase is data transfer phase, in this the message to be exchanged is divided into fragments and each fragment is attached with message authentication code (MACs). Then each fragment is encrypted with the secret key that have already exchanged. The commonly used encryption algorithm is RSA public key algorithm.

D. Algorithms

PresenceCloud Stabilization algorithm: The nodes are mobile nodes which change its position from time to time so a stabilization algorithm is required to maintain nodes in presence server list. Here node failure or wrong connectivity nodes can occur. This is a fault-tolerance algorithm.

Algorithm 1: PresenceCloud Stabilization algorithm

```
/* periodically verify PS node n's pslist */
1.  For every node in pslist
2.  Check whether it's node id and current PS node id
    are   equal
3.  If not equal then find correct connection PS node
4.      If no correct connection PS node
5.      then select a random node
6.      Replace the current PS node id with correct
        connection node id or random node id
7.  Else send Heart Beat message to check node failure
8.  If there is node failure then
9.  a random node is selected to replace the failed node
```

This random node is the node that is in the same row or column of failed or wrong connection PS node.

Algorithm 2: Directed Buddy Search algorithm:

Definition:

B={b1,b2,...,bk}: set of identifiers of user's buddies
b(i): set of buddies that share the same grid ID i
Sj: set of pslist[].id of PS node j

Algorithm

```
1.  A mobile user logins PresenceCloud
2.  Connected to the associated PS node, q
3.  The user then sends a Buddy List Search Message, B
    to q
4.  When q receives B it retrieves each bi from B

5.  Searches its user list and one-hop cache list to
    respond to the coming query.
6.  Then removes the responded buddies from B
7.  If B = nil, the buddy list search operation is done
8.  Otherwise, q should hash each remaining identifier in
    B to obtain a grid ID
9.  Then 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.
```

IV. PERFORMANCE EVALUATION

PresenceCloud can achieve major performance gains in terms of the search cost and search satisfaction even for large number of users. The average buddy searching time for a mobile user is called search latency. The above architecture is simulated by using a NS2 simulator and the following graphs are obtained.

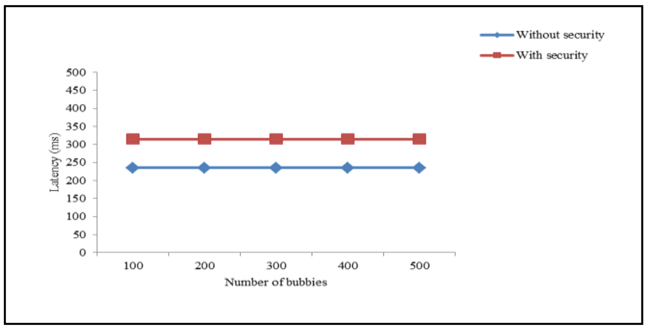

Fig. 4. Searching latency versus number of buddies

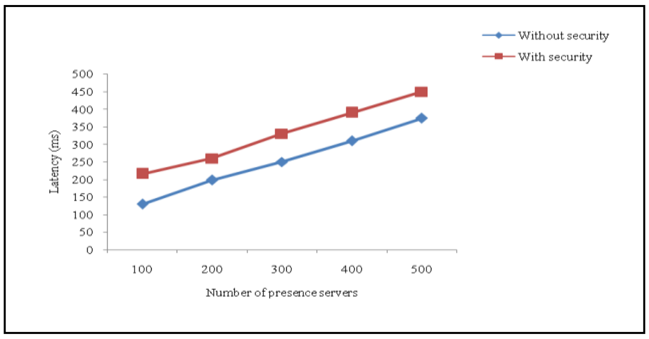

Fig. 5. Searching latency versus number of presence servers

In figure 4 the number of presence server is kept constant and the number of buddies is increased. It is seen that the latency remains the same when the number of buddies increases; this is because the search latency depends on server overlay diameter not the number of buddies. In figure 5, when the number of presence servers are increased the search latency also increases. For the system with security the latency will be higher than that of the system without security. This high in latency is because of key computation, key exchange etc. But this level of search latency will be lower as compared to the earlier technologies which uses mesh- based design and Distributed Hash Tables (DHT).

V. CONCLUSION

A quorum based server to server architecture called Presence Cloud has been presented. This architecture supports mobile presence services and solves the buddy list search problem in large scale social networks. The presented architecture has been secured by SSL protocol. The performance of the above architecture has been evaluated and has shown improved search satisfaction. The number of search messages and search latency has been reduced.

REFERENCES

[1] R. B. Jennings, E. M. Nahum, Olshefski D.P, Saha, D, Zon-Yin Shae, Waters C. ”A study of internet instant messaging and chat protocols” IEEE Network, Page(s): 16 – 21, Volume: 20, 2006.

[2] Z. Xiao, L. Guo, J. Tracey, ”Understanding instant messaging traffic characteristics” Proc. of IEEE ICDCS, Page(s): 51, 2007.

[3] Chi-Jen Wu, Jan-Ming Ho, Ming-Syan Chen, “A Scalable Peer-to-Peer Presence Directory” Institute of Information Science, 2008.

[4] Chi-Jen Wu, Jan-Ming Ho, Ming-Syan Chen, “A scalable server architecture for mobile presence services in social network applications” IEEE Transactions on Mobile Computing, Page(s): 386 - 398, Volume: 12, 2013.

[5] Loreto S., Eriksson G.A., “Presence Network Agent: A Simple Way to Improve the Presence Service”, IEEE Communications, Page(s): 75- 79, Volume: 46, 2008.

[6] Peifeng Si, Chuan Song, Xiang Zhou, “Intelligent server management framework over extensible messaging and presence protocol”, IEEE Communications, Page(s): 128- 136, Volume: 10, 2013.

[7] Sheldon F.T., Weber John Mark, Seong-Moo Yoo, Pan, W.David “The Insecurity of Wireless Networks” IEEE Security and Privacy, Page(s): 54- 61, Volume: 10, 2012.

[8] Kuan Zhang, Xiaohui Liang, Xuemin Shen, Rongxing Lu “Exploiting multimedia services in mobile social networks from security and privacy perspectives” IEEE Communications Magazine, Page(s): 58 – 65, Volume: 52, 2014.

[9] Chou W “Inside SSL: the secure sockets layer protocol” IEEE IT Professional, Page(s): 47- 52, Volume: 4, 2002.

[10] Alfred C. Weaver “Secure Sockets Layer” IEEE Computer, Page(s): 88- 90, Volume: 39, 2006.

[11] Jun Deng, Rongqing Zhang, Lingyang Song, Zhu Han, Bingli Jiao “Truthful Mechanisms for Secure Communication in Wireless Cooperative System” IEEE Transactions on Wireless Communications, Page(s): 4236- 4245, Volume: 12, 2013.

[12] Potlapally, N.R., Ravi S., Raghunathan A., Jha N.K. “A study of the energy consumption characteristics of cryptographic algorithms and security protocols” IEEE Transactions on Mobile Computing, Page(s): 128 – 143, Volume:5, 2006.

[13] Ren Wei, Yu Linchen, Gao Ren, Xiong Feng “Lightweight and compromise resilient storage outsourcing with distributed secure accessibility in mobile cloud computing” Tsinghua Science and Technology, Page(s): 520- 528, Volume: 16, 2011.

[14] Honggang Wang, Shaoen Wu, Min Chen, Wei Wang “Security protection between users and the mobile media cloud” IEEE Communications Magazine, Page(s): 73- 79, Volume: 52, 2014.