Scalable Mobile Presence Cloud Authentication Through PGP

Puneet Garg[1]
[1]Department of Computer Science &Engineering,
Ganga Institute of Technology and Management,
Kablana, Jhajjar, Haryana, India

***Abstract*— An efficient and scalable server architecture, called PresenceCloud, which enables mobile presence services to support large-scale social network applications. When a mobile user joins a network, PresenceCloud searches for the presence of his/her friends and notifies them of his/her arrival. PresenceCloud organizes presence servers into a quorum-based server-to-server architecture for efficient presence searching. It also leverages a directed search algorithm and a one-hop caching strategy to achieve small constant search latency. We analyze the performance of PresenceCloud in terms of the search cost and search satisfaction level. The search cost is defined as the total number of messages generated by the presence server when a user arrives; and search satisfaction level is defined as the time it takes to search for the arriving user's friend list. The presence server authentication problem is a security problem in distributed presence services. In centralized presence architectures, it is no presence server authentication problem, since users only connect to an authenticated presence server. In PresenceCloud, however, requires a system that assumes no trust between presence servers, it means that a malicious presence server is possible in PresenceCloud. To address this authentication problem, a simple approach is to apply a centralized authentication server. Every presence server needs to register an authentication server; PresenceCloud could certificate the presence server every time when the presence server joins to PresenceCloud. An alternative solution is PGP web of trust model, which is a decentralized approach.**

Keywords*— Social Networks, Mobile presence services, PGP, PresenceCloud, Distributed presence servers

I. INTRODUCTION

Presence enabled applications can be provided by mobile devices and cloud computing environments, i.e., social network applications/ services, worldwide. From the last decade, presence enabled applications are growing rapidly. Some of the examples are Facebook [1], Twitter [2], Foursquare, Google Latitude, buddycloud [3] and Mobile Instant Messaging (MIM) [4]. An essential component of social network services is mobile presence service in cloud computing environments. It maintains an up-to-date list of presence information of all mobile users. The details of presence information includes about a mobile user‟s location, availability, activity, device capability, and preferences.

For example, when a user logs into a social network application, such as an Instant Messaging system, through his/her mobile device, the mobile presence service searches for and notifies everybody on the user‟s buddy list. Most presence services use server cluster technology [5] to maximize a mobile presence service‟s search speed and minimize the notification time. Here we discuss about three contributions. First, PresenceCloud [13] is one of the pioneering architecture for mobile presence services. To the best of our knowledge, this is the first work that explicitly designs a presence server architecture that significantly outperforms those based distributed hash tables. Internet social network applications and services can also utilize presence cloud that needs to replicate or search for mutable and dynamic data among distributed presence servers.

The second contribution is that we define a new problem called the buddy-list search problem by analyzing the scalability problems of distributed presence server architectures. The scalability problem in the distributed server architectures of mobile presence services is analyzed through our mathematical formulation. Finally, we demonstrate the advantages of PresenceCloud by analyzing the performance complexity of PresenceCloud and different designs of distributed architectures, and evaluate them empirically. The primary abstraction exported by our PresenceCloud is used to construct scalable server architecture for mobile presence services, and can be used to efficiently search the desired buddy lists.

In the mobile Internet, a mobile user can access the Internet and make a data connection to PresenceCloud via 3G or Wi-Fi services. After the mobile user joins and authenticates him/her to the mobile presence service, the mobile user is determinately directed to one of Presence Servers in the PresenceCloud by using the Secure Hash Algorithm, such as SHA-1 [6]. The Pretty Good Privacy Program (PGP) uses a "Web of Trust" model for establishing trust relationships between users. This could allow a user to indirectly and unknowingly trust others that the user does not know. Then mobile user opens a TCP connection to the Presence Server (PS node) for control message transmission, particularly for the presence information.

After the control channel is established, the mobile user sends a request to the connected PS node for his/her buddy list searching. Our PresenceCloud shall do an efficient searching operation and return the presence information of the desired buddies to the mobile user.

II. RELATED WORK

In this section, we describe previous researches on presence services, and survey the presence service of

Existing systems. Several IETF charters [7] have addressed closely related topics and many RFC documents on instant messaging and presence services (IMPS) have been published, e.g., XMPP [8], SIMPLE [9]. Jabber [10] is a well-known deployment of instant messaging technologies based on distributed architectures. It captures the distributed architecture of SMTP protocols. Since Jabber‟s architecture is distributed, the result is a flexible network of servers that can be scaled much higher than the monolithic, centralized presence services. Recently, there is an increase amount of interest in how to design a peer-to-peer SIP [11]. P2P-SIP [12] has been proposed to remove the centralized server, minimize maintenance costs, and keep from happening failures in server-based SIP deployment.

To maintain presence information, P2PSIP clients are organized in a DHT system, rather than in a centralized server. Our research is essentially the paradigm in which we attack our problem. Just as in the electronic marketplace scenario, our agents do not have to rely on centralized mechanisms. Trust information is propagated throughout the system via the interaction of the agents themselves, and the „quality‟ of this information is calculated on the basis of the perceived trustworthiness of recommenders,,. Trust is then revised upon receipt of new recommendations or new experiences. Other related work includes Pretty Good Privacy, which helped inspire the distributed nature of our model. Trust is also being used as a basis for cooperation among autonomous agents in the area of Distributed AI. it involves the addition of perceived risk and utility of committing resources in a cooperative relationship.

III. DESIGN OF PRESENCECLOUD

PresenceCloud is used to construct and maintain distributed server architecture and can be used to efficiently query the system for buddy list searches. PresenceCloud consists of three main components that are run across a set of presence servers. In the design of PresenceCloud, we refine the ideas of P2P systems and present a particular design for mobile presence services. The three key components of PresenceCloud are summarized below:

A. PresenceCloud server overlay

The PresenceCloud server overlay construction organizes the Presence server nodes into a server-to-server overlay, which provides a good low-diameter overlay property. The low-diameter property makes sure that a Presence server node only needs two hops to reach any other PS nodes. Our PresenceCloud is based on the concept of grid quorum

system [15], 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. The size of a grid quorum is $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$.

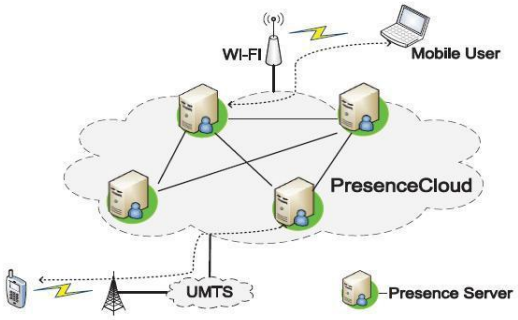

Fig:1 An Overview of Presence Cloud

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.1 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 these entries will become its PS list in a PresenceCloud server overlay. We now show that each PS node in a PresenceCloud System only maintains the PS list of $O(\sqrt{n})$, and the construction of

PresenceCloud using the grid quorum results in each PS node can reach any PS node at most two hops.

B. One-hop caching strategy

PresenceCloud requires a caching strategy to make an exact copy of presence information of users to improve the efficiency of the search operation. 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. In PresenceCloud, each PS node maintains a user list of presence information of the attached users, and it is responsible for caching the user list of each node in its PS list, in other words, PS nodes only replicate the user list at most one hop away from itself. The cache is updated when neighbours establish connections to it, and periodically updated with its neighbours.

Therefore, when a PS node receives a query, it can respond not only with matches from its own user list, but also provide matches from its caches that are the user lists offered by all of its neighbours. Our caching strategy does not require expensive overhead for presence consistency among PS nodes. When a mobile user changes its presence information, either because it leaves PresenceCloud, or due to failure, the responded PS node can disseminate its new presence to other neighbouring PS nodes for getting updated quickly.

Consequently, this one-hop caching strategy ensures that the user‟s presence information could remain mostly up-to date and consistent throughout the session time of the user. More specifically, it should be easy to see that, each PS

node maintains roughly $2(\lceil \sqrt{n} \rceil - 1) \times u$ duplicates of presence information, due to each PS node duplicates its user list at

most one hop away from itself. Here, u is denoted the average number of mobile users in a PS node.

C. Directed Buddy Search

We contend that reducing searching response time is important to mobile presence services. Thus, the buddy list searching algorithm of PresenceCloud coupled with the two-hop overlay and one-hop caching strategy ensures that PresenceCloud 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 neighbours, we improve response time by increasing the chances of finding buddies.

Clearly, this mechanism both reduces the network traffic and response time. Based on the mechanism, the population of mobile users can be retrieved by a broadcasting operation in any PS node in the mobile presence service. Moreover, the broadcasting message can be piggybacked in a buddy search message for saving the cost.

IV.WEB OF TRUST

The web-of-trust model [14] differs greatly from the hierarchical model. The hierarchical model is easily represented with computers as a tree, but the web-of-trust more closely relates to how people determine trust in their own lives. As people go through life and meet new people, they look to those they already trust to help make the determination if they should trust these new people. If people they trust already trust this new person, they are more likely to do the same.

PGP implements this model by allowing people to sign each other‟s keys assigning to each of the keys a level of trust and validity. PGP assumes that a certificate binds a key to a person. That person chooses a name to identify them self, and also provide an email address as a global identifier.

When a user signs another user‟s key, the signing user is asserting that they have verified that the key truly belongs to the listed user and that it is valid. This method of signing keys leads to an ad-hoc network of trust. Because of the shape of the resulting graph, Phil Zimmerman called this system the “Web of Trust”. The system allows users to specify how much trust to place in a signature by indicating how many independent signatures must be placed on a certificate for it to be considered valid.

HOW PGP WORKS

PGP combines some of the best features of both conventional and public key cryptography. PGP is a *hybrid cryptosystem.* When a user encrypts plaintext with PGP, PGP first compresses the plaintext. Data compression saves modem transmission time and disk space and, more importantly, strengthens cryptographic security. Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher. Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis (Files that are too short to compress or which don't compress well aren't compressed).

PGP then creates a *session key*, which is a one-time-only secret key. This key is a random number generated from the random movements of your mouse and the keystrokes you type. This session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext; the result is cipher text. Once the data is encrypted, the session key is then encrypted to the recipient's public key. This public key-encrypted session key is transmitted along with the cipher text to the recipient. Decryption works in the reverse. The recipient's copy of PGP uses his or her private key to recover the temporary session key, which PGP then uses to decrypt the conventionally-encrypted cipher text.

The combination of the two encryption methods combines the convenience of public key encryption with the speed of conventional encryption. Conventional encryption is about 1, 000 times faster than public key encryption. Public key encryption in turn provides a solution to key distribution and data transmission issues. Used together, performance and key distribution are improved without any sacrifice in security.

VI.PGP CERTIFICATE FORMAT

A PGP certificate includes (but is not limited to) the following information:

- The PGP version number — this identifies which version of PGP was used to create the key associated with the certificate.
- The certificate holder's public key — the public portion of your key pair, together with the algorithm of the key: RSA, DH (Diffie-Hellman), or DSA (Digital Signature Algorithm).
- The certificate holder's information — this consists of "identity" information about the user, such as his or her name, user ID, photograph, and so on.
- The digital signature of the certificate owner — also called a *self-signature;* this is the signature using the corresponding private key of the public key associated with the certificate.
- The certificate's validity period — the certificate's start date/ time and expiration date/ time; indicates when the certificate will expire.
- The preferred symmetric encryption algorithm for the key — indicates the encryption algorithm to which the certificate owner prefers to have information encrypted. The supported algorithms are CAST, IDEA or Triple-DES.

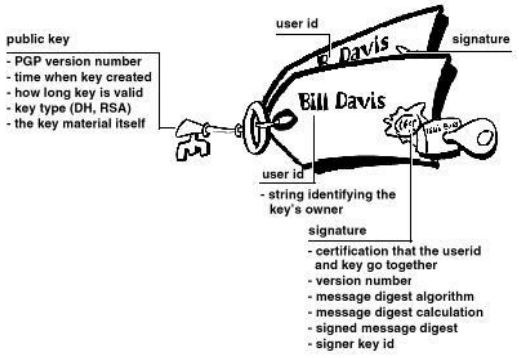

Fig 2. A PGP certificate

You might think of a PGP certificate as a public key with one or more labels tied to it (see *Figure 1-9*). On these 'labels' you'll find information identifying the owner of the key and a signature of the key's owner, which states that the key and the identification go together. (This particular signature is called a *self-signature;* every PGP certificate contains a self-signature). One unique aspect of the PGP certificate format is that a single certificate can contain multiple signatures. Several or many people may sign the key/ identification pair to attest to their own assurance that the public key definitely belongs to the specified owner. If you look on a public certificate server, you may notice that certain certificates, such as that of PGP's creator, Phil Zimmermann, contain many signatures.

Some PGP certificates consist of a public key with several labels, each of which contains a different means of identifying the key's owner (for example, the owner's name and corporate email account, the owner's nickname and home email account, a photograph of the owner — all in one certificate). The list of signatures of each of those identities may differ; signatures attest to the authenticity that one of the labels belongs to the public key, not that all the labels on the key are authentic. (Note that 'authentic' is in the eye of its beholder — signatures are opinions, and different people devote different levels of due diligence in checking authenticity before signing a key).

VII.CONCLUSION

In this paper we have presented a scalable server architecture called PresenceCloud that supports mobile presence services in large scale social network services. The authentication problem is the security problem in distributed presence services and it could be solved through PGP web of trust model, which is a decentralized approach.

REFERENCES

[1] Facebook, http://www.facebook.com, 2012.
[2] Twitter, http://twitter.com, 2012
[3] Buddycloud, http://buddycloud.com, 2012.
[4] Mobile Instant Messaging, http://en.wikipedia.org/wiki/ Mobile_instant_messaging, 2012.
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, vol. 20, no. 6, pp. 16-21, July/Aug. 2006
[6] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," IETF RFC 3174, 2001.
[7] Extensible Messaging and Presence Protocol IETF Working Group, http://www.ietf.org/html.charters/xmpp-charter.html, 2012.
[8] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence Describes Instant Messaging (IM), the Most Common Application of XMPP," IETF RFC 3921, 2004.
[9] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, Session Initiation Protocol (SIP) Extension for Instant Messaging, IETF RFC 3428, 2002.
[10] http://www.jabber.org, 2012.
[11] Peer-to-Peer Session Initiation Protocol IETF Working Group, http://www.ietf.org/html.charters/p2psip-charter.html, 2012.
[12] K. Singh and H. Schulzrinne, "Peer-to-Peer Internet Telephony Using SIP," Proc. ACM Int"l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDVA), 2005.
[13] Chi-Jen Wu . Jan-Ming Ho , Ming-Syan Chen , "A Scalable Server Architecture for Mobile Presence Services in Social Network Applications ," IEEE Trans. Mobile Computing, vol. 12, no. 2, pp. 386-398, Feb. 2013, doi: 10.1109/TMC.2011.263
[14] A. Abdul-Rahman and S. Hailes, "A Distributed Trust Model," Proc. Workshop New Security Paradigms, 1997.
[15] M. Maekawa, "A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems," ACM Trans. Computer Systems, vol. 3, pp. 145-159, 1985.