XMPP in Instant Messengers

Neeraj Vashistha[1] Akshay Khilari[2] S.N. Shelke[3]
[3]Assistant Professor
[1,2,3]Department of Computer Engineering
[1,2,3]Sinhgad Academy of Engineering, SPPU, Pune.

Abstract— Instant Messengers(IM) are the most commonly used interacting method to communicate with people worldwide. IM presence is felt in multicast (chat rooms) and one-to-one message exchange between different kind of client entities. IM using eXtensible Messaging and Presence Protocol (XMPP) as the base protocol is able to provide near real-time text-based communication through the use of XML. Social media networks like gtalk, ejabberd, facebook employ XMPP. XMPP can be implemented with servers as well as without servers. XMPP is not only used in IM but also collaborative services like sensor networks, multiplayer games and Internet-of-Things (IoT) as they require real-time event publication and distribution. In this paper we briefly describe XMPP and its usage in IM and also discuss the market share of XMPP.

Key words: XMPP, Social Media Networks, Collaborative Services, Internet of things, Near Real-time communication

I. HISTORY OF XMPP AND LESSONS LEARNT

As discussed in [1], based on TCP, DNS and SSL and built as a client-server architecture XMPP prophesied for simple distributed technology where privacy and security would be given priority but this was not possible if PGP (pretty good privacy) keys were not in place for each user or a digital certificate for each user. Thus the desire to keep the technology as simple as possible was not attainable easily. Following were the issues associated with XMPP.

1) XMPP is optimized for handling a large number of small XML based messages but has a difficulty in sending/parsing large chunks of video /audio.
2) For meshes like ad hoc and mobile network, deploying a technology on TCP and DNS is challenging thus there is a need to rethink about its deployment and dependencies on other technology.
3) XMPP as an open technology strives to be extensible and modular along with it providing anyone to run a server instance to create a decentralized communication system where the network availability information or presence is available throughout the system.

The early XMPP implementation by its inventor Jeremie Miller has all those problems as listed above in points 1,2,3 which are now resolved quite successfully and are discussed in great depth in [1] and much of the undiscussed work of serverless XMPP is implemented in XMPP based SMS gateway [10], XMPP implementation and security at cloud [11], XMPP as a service integration scheme [9], distributed chat service [8].

II. XMPP ARCHITECTURE

XMPP uses a decentralized client-server architecture for instant messaging [6], presence [6], [3] and real-time communication. The XMPP architecture as in [7] used for instant messaging typically consists of hundreds of XMPP servers like ejabberd and XMPP clients which run software like Pidgin, Adium using the standard XMPP protocol. The main advantage of using a decentralized architecture is that it allows client developers to focus on user experience and server developers to focus on reliability and scalability. The servers incorporate important security features such as authentication, channel encryption.

The following are the components of the XMPP architecture:

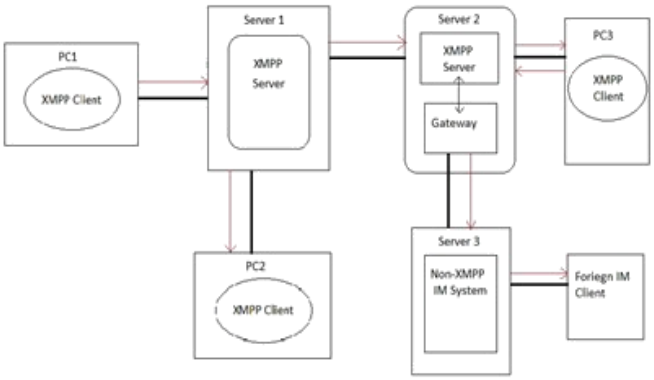

Fig. 1: XMPP Architecture

A. Server:

It provides important security features such as authentication, encryption and is used to manage connections from authorized clients, servers and other entities.

B. Client:

Clients are the end user which initiates streams and communicates with the server using client software like ejabberd.

C. Gateway:

A gateway is a server-side service and its primary function is to translate the XMPP streams into protocols used by a non-XMPP messaging system and vice versa. Gateways to SMS (Short Message Service), IRC (Internet Relay Chat), email are some examples.

III. XMPP BASICS

XMPP uses a client-server architecture but it is highly decentralized. XMPP core [5], specify client can only communicate with another client when it is authenticated with server and unless authenticated, a client cannot inject messages on a network.

However a serverless messaging [4] can be established with zero network configuration on local (or even wide area) network using the _presence._tcp DNS SRV service type.

Once discovered by any server or on a serverless network, the client can communicate by negotiating XML streams among themselves and exchanging structured data using XMPP <message/> and <iq/> (information and query) stanzas.

On a serverless (or ad hoc WAN) messaging, a chat client can advertise its presence by invoking a DNS-based Service Discovery (DNS-SD) daemon (RFC 6763) and multicast DNS (mDNS) (RFC 6762) by publishing the DNS records on multicast DNS address 244.0.0.251 and listen for multicast DNS queries.

```
workstation7.  local. A 10.2.1.2
tim@workstation7._presence._tcp.local. SRV
5356 workstation7.local.
_presence._tcp.local. PTR
tim@workstation7._presence._tcp.local
```

Here 'A' record at IP 10.2.1.2 specifies 'workstation7' listening for connection, 'SRV' record (RFC 2732[3]) maps the presence status of instance 'tim@workstation7' to machine 'workstation7.local' on port 5356 and 'PTR'(pointer) record (RFC 2317[4]) points that service of type 'presence 'exist over TCP. Along with above information DNS TXT record (RFC 1464[5]) is also published by a chat client. this information is typically key value based and is published using mDNS daemon.

```
txtvers=1
1st=Tim
email=tim@mail.com
hash=sha-1
id=117
last=Petric
msg=Hanging out
nick=timP
node=http://www.mdgn.com
phsh=a3839614e1aa56b3fe7bcf78523f519e02358913
port.p2pj=5356
status=avail
vc=CA!
ver=QjolPKapkwPSDYstT/WM75uBl47=
```

The same functionality is applicable to a client-server distributed architecture, when client joins the network and advertises its presence using the _presence._tcp DNS SRV service type. For example, a client(John) would be able to discover any number of local presence services like time@workstation7 at IP address 10.2.1.3 on port 5356. On discovering this, client can initiate a chat by opening a XML stream.

```
<?xml version='1.0'?>
<stream:stream
xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
   from='john@comp-PG-1'
   to='tim@workstation7'
   version='1.0'>
```

On receiving a request from John, Tim respond with a response stream header.

```
 <?xml version='1.0'?>
<stream:stream
xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'
      from='tim@workstation7'
      to='john@comp-PG-1'
      version='1.0'>
```

There are three types of XML stanza which are exchanged between clients, Message, presence, and iq.

Message stanzas consists of to and from along with attribute, type and id. Payloads include subject and body. In Server messaging, the from address is stamped by server to avoid spoofing but in case of serverless messaging it is ought to be included by the client. For example, John may now send a message to Tim in the following way.

```
<message          from='john@comp-PG-1'
                  to='tim@workstation7'
                  type='chat'
        <body>Hello, Tim how are you?</body>
        <subject>Query</subject>
</message>
```

Presence stanza is a specialised publish-subscribe method to publish network availability and status messages of clients to only those clients which are authorized to receive i.e. subscribed clients. When a client comes into network (online) and leaves the network (offline) presence is advertised. This XML stanza provides presence status in roster, it is the responsibility of server (in Server Messaging Scenario) to handle the task of notifying other clients in the roster.

```
<presence         from='tim@workstation7'>
        <show>Available</show>
        <status>Hanging out</status>
</presence>
```

The iq stanza (information query) in XMPP are method for requesting and responding information in structured format through the use of of id (or jid) attribute. The payload, id associates the stanza and helps to track the request and response between client entities. In case of failure, iq-error is returned.

```
<iq       from='tim@workstation7'
          id='117'
          to='tim@mail.com'
          type='get'>
          <query xmlns='jabber:iq:roster'/>
</iq>
```

The response is iq-result having a payload, item element for each client in the roster.

```
<iq       from='tim@mail.com'
          id=117
          to='tim@workstation7'
          type='result'>
          <query xmlns='jabber:iq:roster'>
                    <item jid='steve@mail.com'/>
                    <item jid='philipp@mail.com'/>
          </query>
</iq>
```

The iq-set is then used to update new contacts in client roster through new transaction.

```
<iq       from='tim@workstation7'
          id=117
          to='tim@mail.com'
          type='set'>
          <query xmlns='jabber:iq:roster'>
                    <item id='steve@mail.com'/>
          </query>
</iq>
```

An acknowledgement through iq-result is submitted by server to client.

```
<iq       from='tim@mail.com'
          id=117
          to='tim@workstation7'
          type ='result'/>
```


IV. MARKET DOMINANCE

Although there are various proprietary and open source protocols for IM and presence applications, XMPP (Jingle, WFP) is one of the open source and standardised protocol. Other open source and recognised protocols are IMPP, IRC, MTProto, RetroShare, SIP (MSRP, SIMPLE), TextSecure, Tox, Zephyr.

Some of the proprietary IM protocol for near real time messaging and presence are MSNP, OSCAR (TOC), Skype, YMSG.

According to a report[6], the number of XMPP servers deployed worldwide are approximately 7264, of which the share of ejabberd server reports to be around 36%, google servers employing XMPP are around 14%, the other main players using XMPP servers are Openfire (13%), jabberd14 (13%), jabberd2 (6%), and Prosody (3%).

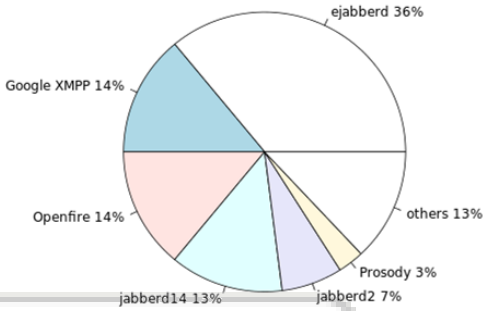

Fig. 2: XMPP Market Share

V. XMPP AND SIMPLE

SIMPLE is SIP (Session Initiation Protocol) for Instant Messaging and Presence Leveraging Extension (RFC 3428[7]) is an alternative to XMPP. SIMPLE is an extension for SIP which serves well for voice and media message transport along with the presence. SIP is basically a signaling technology which intends to transport both presence and data. XML-based XMPP apostle argues well, as XMPP is more versatile, extensible and adaptable it is very much suited for handling IM (Instant Messages) rather than a signaling technology such as SIMPLE. This argument has proven its grounds as the growth of XMPP in social networking and other wide range of application is tremendous. SIMPLE (RFC 3428[7]) applies SIP for presence exchange, sending short-message and managing near real-time sessions and is found in SIP softphones and SIP hardphones. Thus to list out a few advantages of SIMPLE

1) SIP software stack support VoIP (Voice over Internet Protocol) (RFC 6405[8]), thus SIMPLE has the capability to fulfill todays need of deploying VoIP with no need to employing any extensions.
2) Along with authorization and authentication, SIMPLE since built on top of SIP, can even employ SigComp (RFC 3320[9]), which is Signal Compression.
3) Since IP Multimedia Subsystem(IMS) uses SIMPLE, the majority of network including mobiles, cables and fixed network are based on SIP.

VI. XMPP ADVANTAGES AND LIMITATIONS

The main advantages of using XMPP in an Instant Messenger are:

1) As the name suggest, XMPP is highly extensible and new add-ons can be easily being incorporated in an existing protocol [1], XMPP Extension Protocols (XEPs), an open standard manages existing extensions.
2) Deployment of XMPP stack [2] on a thin client like mobile devices [10], [11], sensors is reasonable as XMPP is lightweight and in comparison to SIMPLE, XMPP requires a few number of RFC to be used to create a functional XMPP server.
3) In comparison to proprietary protocols, XMPP is highly decentralised. Proprietary protocols extensively control the specification and exchange of messages along with it, only those clients that implement these protocols can communicate with servers.
4) Through the use of gateways [10] where federation mechanism can be implemented, XMPP allows communicating with proprietary protocols.

A few limitations faced while setting up a XMPP server for exchange of XML data are:

1) Large overhead data in XMPP causes inter-server traffic, this data mainly comprise of redundant presence status, which is multicast to all the clients in a roster.
2) The use of unavoidable external protocols like HTTP for file transfer, Jingle for voice and video etc. These external protocols are used as binary form of data cannot be directly being transmitted. Marshalling and demarshalling for encrypted messages is done via embedding using encoding techniques like Off-the-Record Messaging (OTR), base64.

VII. CONCLUSION

In this paper we briefly described XMPP, the architecture of XMPP, which explained working with clients and server. Section XMPP Basics, showed how XMPP is used to communicate with other clients on a server or serverless network and described configurations which are need to establish a successful communication. In the same section, we discussed XML stanzas which described message, presence and iq (information query), the structured XMPP data. XMPP and SIMPLE both dominated the market and are extensively adopted by open community, although the use of proprietary protocols exits, the process of federation for internetworking with standardised protocols is widespread. Though XMPP has some limitations but these will be resolved in the forthcoming years and will be standardised and a more versatile and extensible protocol for Instant Messengers(IM) is possible.

REFERENCES

[1] Peter Saint-Andre, "XMPP: lessons learned from ten years of XML messaging", in IEEE Multimedia, vol. 47, no. 4, pp.92 -96 2009
[2] Peter Saint-Andre, "Streaming XML with Jabber/XMPP", in IEEE Internet Comp., vol.9, no. 5, pp.82 -89 2005
[3] Joe Hildebrand, Peter Millard, Ryan Eatmon, Peter Saint-Andre, "XEP-0030: Service Discovery" in XMPP Standard Foundation, June 2008
[4] Peter Saint-Andre, "XEP-0174: Serverless Messaging" in XMPP Standard Foundation, Nov 2008
[5] Peter Saint-Andre, Extensible Messaging and Presence Protocol (XMPP): Core, IETF proposed standard, RFC 6120, Mar 2011, www.ietf.org/rfc/rfc6120.txt.

[6] Peter Saint-Andre, Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, IETF proposed standard, RFC 6121, Mar 2011, www.ietf.org/rfc/rfc6121.txt.
[7] P. Saint-Andre, K. Smith, and R. Tronçon, XMPP: The Definitive Guide, O'Reilly, 2009.
[8] Robert, N.L.; Macker, J.; Millar, D.; William, C.R.; Taylor, I., "XO: XMPP overlay service for distributed chat," in Military Communications Conference, 2010 - MILCOM 2010 , vol., no., pp.1116-1121, Oct. 31 2010-Nov. 3 2010
[9] Feng-Cheng Chang; Duen-Kai Chen, "The Design of an XMPP-based Service Integration Scheme," in IIH-MSP, 2011 Seventh International Conference on, vol., no., pp.33-36, 14-16 Oct. 2011
[10] Xingchen Lu; Weimin Lei; Wei Zhang, "The Design and Implementation of XMPP-Based SMS Gateway," in Computational Intelligence, Communication Systems and Networks (CICSyN), 2012 Fourth International Conference on, vol., no., pp.145-148, 24-26 July 2012
[11] Grubitzsch, P.; Schuster, D., "Hosting and Discovery of Distributed Mobile Services in an XMPP Cloud," in Mobile Services (MS), 2014 IEEE International Conference on, vol., no., pp.47-54, June 27 2014-July 2 2014

