XGFX: eXtensible Game Framework through XMPP

Aditya Borikar
Information Technology
Pune Institute of Computer Technology
Pune, India

ABSTRACT
XMPP is the open standard for messaging and presence. It stands for Extensible Messaging and Presence Protocol, a set of protocols known for their wide range of application in Instant Messaging through presence, multi-party chat, voice/video calls, and generalized routing of XML data. An idea for a gaming framework has been developed and illustrated in this paper as a prototype for secure decentralized gaming servers trying to obtain easy transmission of data through XML. The concept introduced is meant to operate between client and server enabling data transmission through multiple servers hosting different domain-name similar to emails. XMPP is extended by a set of protocols called XMPP Extension Protocols (XEP). These XEPs allow developers to develop a generic middleware between gaming servers and gaming stations using XEPs such as publish/subscribe (pubsub), group management and Jingle.

General Terms
Gaming Framework, Server-Client Communication, XGFX, XEP, XMPP.

Keywords
XGFX, Jingle, XEP, XMPP, Games, Pubsub

1. INTRODUCTION
This paper introduces an approach to configure gaming servers through XMPP, the highly preferred framework for Instant Messaging. XMPP is a set of protocols which allow asynchronous transmission of data between server-client along with client-client transmission. As online gaming is on high demand, modern gaming servers are developed in order to handle heavy congestion to switch quickly between multiple gaming worlds, establish voice or text channels from player-to-player and server configuration for server-server and server-client communication. Instant Messaging(IM) servers and Gaming Servers(GS) have similarities between them which are discussed in this documentation.

How XMLs serve as the building block of XMPP:

The information exchange in any XMPP implementation is based upon a connection established through XML stream. The smallest unit of information that is transmitted across the connection is called a stanza which is a well formed piece of XML. There are three types of stanzas: Message, Presence and Info/Query.

1) Message: The <message/> stanza is a 'push' mechanism where one entity pushes information to another entity,similar to the communications that occur in a system such as email. All message stanzas will possess a 'to' attribute that specifies the intended recipient of the message, unless the message is being sent to the bare JID of a connected client's account. Upon receiving a message stanza with a 'to' address, a server SHOULD attempt to route or deliver it to the intended recipient.

   <message from="romeo@montague.net"

   to="juliet@capulet.com">

   <body> Take me back !</body> …[Payload]

   </message>

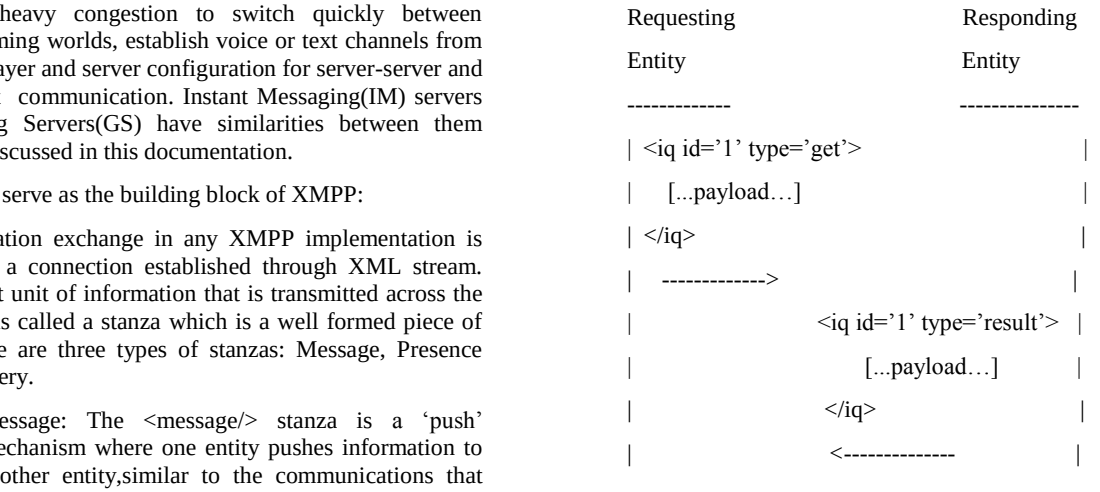
2) Presence : The <presence/> stanza is a specialized 'broadcast' or 'publish-subscribe' mechanism, whereby multiple entities receive information (In this case, network availability information) about an entity to which they have subscribed. Presence information is disclosed only to other entities the user has already approved in order to protect the privacy of XMPP users.

   <presence from="romeo@example.net">

   <status>Available</status>
   …[Payload]

   </presence>

3) Info/Query(IQ) : Info/Query, or IQ, is a 'request-response' mechanism, similar in some ways to the Hypertext Transfer Protocol [HTTP]. The semantics of IQ enable an entity to make a request of, and receive a response from, another entity. The data content of the request and response is defined by the schema or other structural definition associated with the XML namespace that qualifies the direct child element of the IQ element, and the interaction is tracked by the requesting entity through the use of 'id' attribute.



Host resolution on an XMPP network:

The location of an entity can be uniquely identified on a network using Jabber ID(JID). A JID consists of the following attributes.

1) Localpart

2) Domainpart

3) Resourcepart

And is structured as : local@domain/resource

eg : juliet@example.com/balcony

2. XGFX Environment

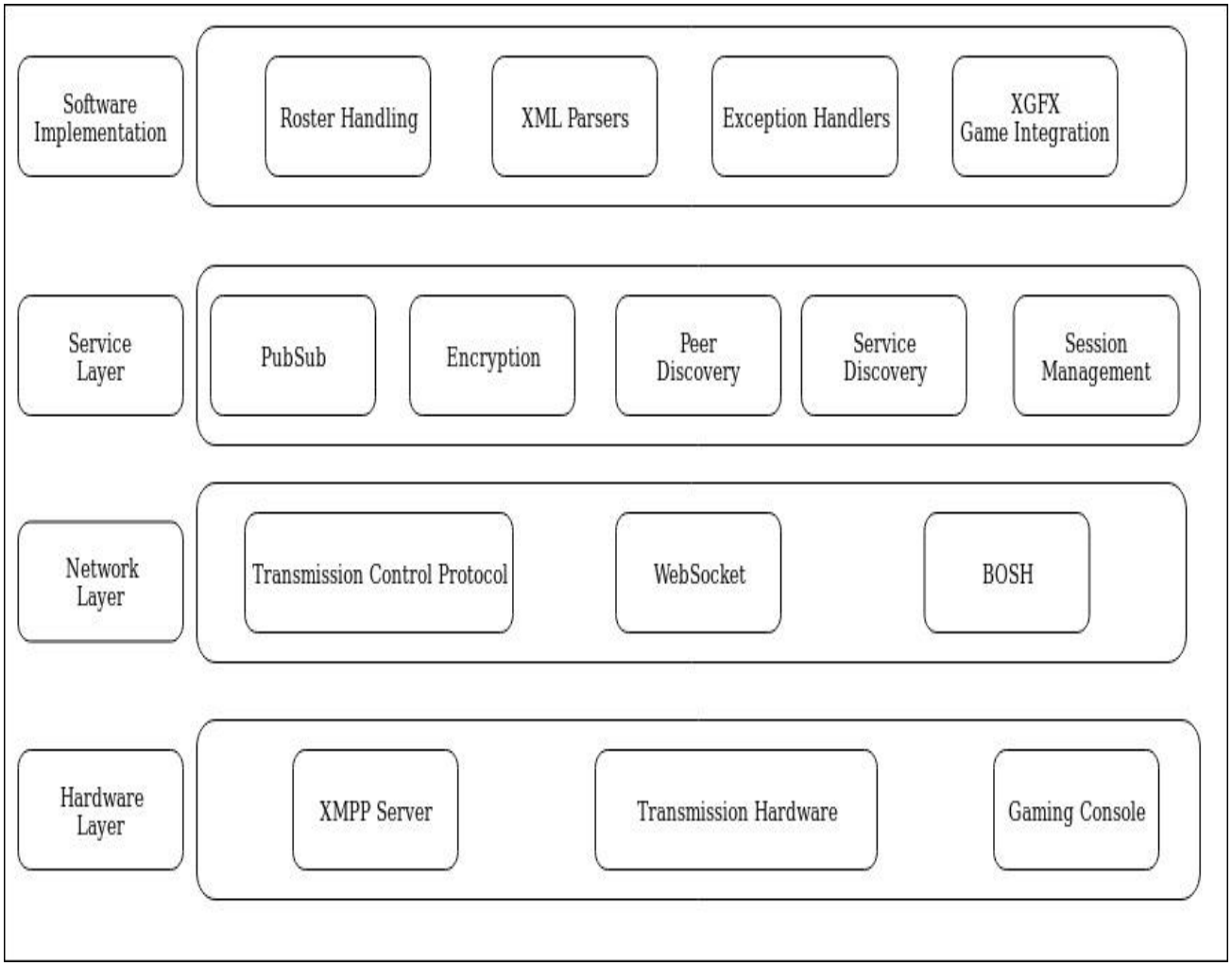


Figure 1 : XGFX Environment

1) Hardware Layer.
Servers will be used to host domain names across the network. XGFX would use a real-time Ejabberd server owned by ProcessOne. Ejabberd makes extensive use of Erlang. The compliance of Ejabberd orders to RFC-6120 along with RFC-6121 which enable establishing successful data transmission across TCP.

The Hardware Layer consists of three major parts:

1. XMPP Server such as Ejabberd, Openfire.
2. Transmission Hardware consisting of Wifi router, LAN cables.

2) Gaming Console which can interpret XMPP stanzas through XMPP client library Smack.
Network Layer.

XGFX would establish connection using Transport Control Protocol(TCP), Websockets or Bidirectional-streams Over Synchronous HTTP(BOSH). An XMPP client can be configured by using Smack. Smack is a client side XMPP library written in Java SE which would enable a gaming console to make use of XMPP framework. Smack is improving as developers are constantly attempting to implement newer and better XEPs.

3) Service Layer.
The service layer consists of XEP implementations which will allow the client to interpret the stanzas received from the network layer. The service layer takes care of end-to-end encryption and Discovery management and contact management through XMPP Roster. Smack-extensions has APIs required to configure the services necessary to implement the XEPs constituting XGFX.

4) Software Implementation.

The game integration with XMPP can be carried out by configuring game clients with Smack. Smack currently establishes connection through TCP and BOSH. It still lacks support for websockets but as desktop gaming environments can be connected to XMPP server using Smack's TCP connection it is currently the best library in the market to implement XMPP in a gaming console.

The Smack's Parser implementation is based upon StAX, best suitable for this pull,streaming parser type mechanism. XML parsing needs to be efficient because the entire XMPP framework is based upon it. So improved parsers directly refer to a better implementation.

Various XEPs serving distinct features involved in games are as follows :

XEP Extension	Description	Usage
XEP-0012	Last Activity	Gamer's online status feature [11]
XEP-0027	Current Jabber OpenPGP Usage	OpenPGP allows data encryption satisfying privacy and security concerns. [12]
XEP-0030	Service Discovery	Allows client to discover presence information and features served by other XMPP entity. [13]
XEP-0045	Multi-User Chat	Will allow the gaming consoles to exchange messages in the context of a room or channel. [14]
XEP-0054	vcard-temp	vcards are meant to be used similar to electronic business cards which can be used to manage gamer's profile. [15]
XEP-0060	Publish-subscribe	This allows XMPP entities to create nodes and publish them onto the network. This is the base for other XEPs such as UserTune and Geolocation.[16]
XEP-0082	XMPP Data and Time Profiles	This is the standardization to numerous date and time formats to avoid anomaly. [17]
XEP-0124	Bidirectional-streams Over Synchronous HTTP (BOSH)	Uses multiple synchronous HTTP request/response pairs without requiring polling and emulates the semantic of TCP connection. [18]
XEP-0166	Jingle	Enables session management that enables voice chat, video chat and file transfer over TCP. [19]
XEP-0172	User Nickname	Allows user to manage game handles for avatars. [20]
XEP-0206	XMPP over BOSH	Allows XEP-0124 to transport XMPP stanzas. [21]
XEP-0288	Bidirectional Server-to-server connection	Allows server-to-server connections to be used to send and receive stanzas in a bidirectional fashion. [22]
XEP-0363	HTTP File Upload	Allows a file to be placed on HTTP server which can later be downloaded again. [23]

3. XMPP IN MARKET

A number of Gaming servers already use XMPP. Some of the games using XMPP Server which have gained considerable popularity are as follows :

Users	Game/ Community	Description	Since
~34 million	Nintendo Switch	Uses Ejabberd for Push Notifications	2019
~1 million	EVE Online	Uses XMPP for in-game chats	2018
~250 million	Fortnite	A cooperative survival game using XMPP for presence, push, whispers and group chat	2017
~16 million	Neverwinter	Using XMPP for chat, presence, group chat	2013
~40 million	Origin	XMPP is used internally for buddy and chat system.	2011
~27 million	League Of Legends	Uses XMPP for chat and game invitations	2009

The Jabber Unity Networking framework has similar to XGFX but is a paid tool. XGFX will show a way to developers all over the globe to integrate Gaming Frameworks with XMPP.

4. PERFORMANCE

The performance results of XMPP server are provided by the ProcessOne XMPP Community (PXC) using Tsung over Ejabberd Server.

About Tsung - It is a tool to perform distributed load testing on an XMPP server and is released under GPLv2 license. It can simulate thousands of virtual users concurrently which creates significant load on the machine. These statistics are published as Tsung Load Testing results. It goes well will TCP, UDP, Web-socket, TLS/SSL (with or without certificate) IPv4 and IPv6. [8]

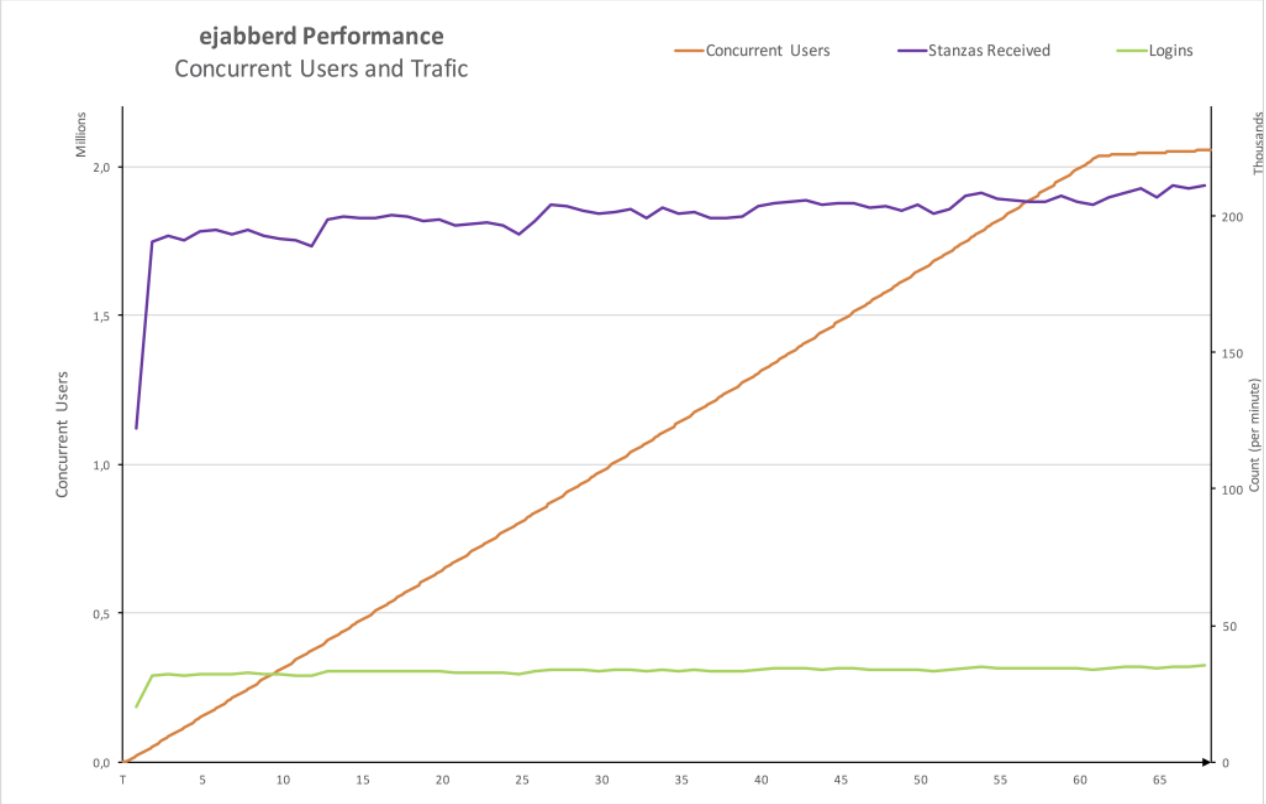
Benchmark shows that we reached 2 million concurrent users after one hour. We were logging in about 33k users per minute, producing session traffic of a bit more than 210k XMPP packets per minute (this includes the stanzas to do the SASL authentication, binding, roster retrieval, etc). Maximum number of concurrent users is reached shortly after the 2 million concurrent users mark, by design in the scenario. At this point, we still connect new users but, as the first users start disconnecting, the number of concurrent users gets stable.[9]

As we try to reproduce common client behavior we setup Tsung to send “keepalive pings” on the connections. Since each session sends one of such whitespace pings each minute, the number of such requests grows proportionately with the number of connected users. And while idle connections consume few resources on the server, it is important to note that in this scale they start to be noticeable. Once you have 2M users, you will be handling 33K/sec of such pings just from idle connections. They are not represented on the graphs, but are an important part of the real life traffic we were generating.[9]

At all time, ejabberd health was fine. Typically, when ejabberd is overloaded, TCP connection establishment time and authentication tend to grow to an unacceptable level. In our case, both of those operations performed very fast during all bench, in under 10 milliseconds. There was almost no errors (the rare occurrences are artefacts of the benchmark process).[9]

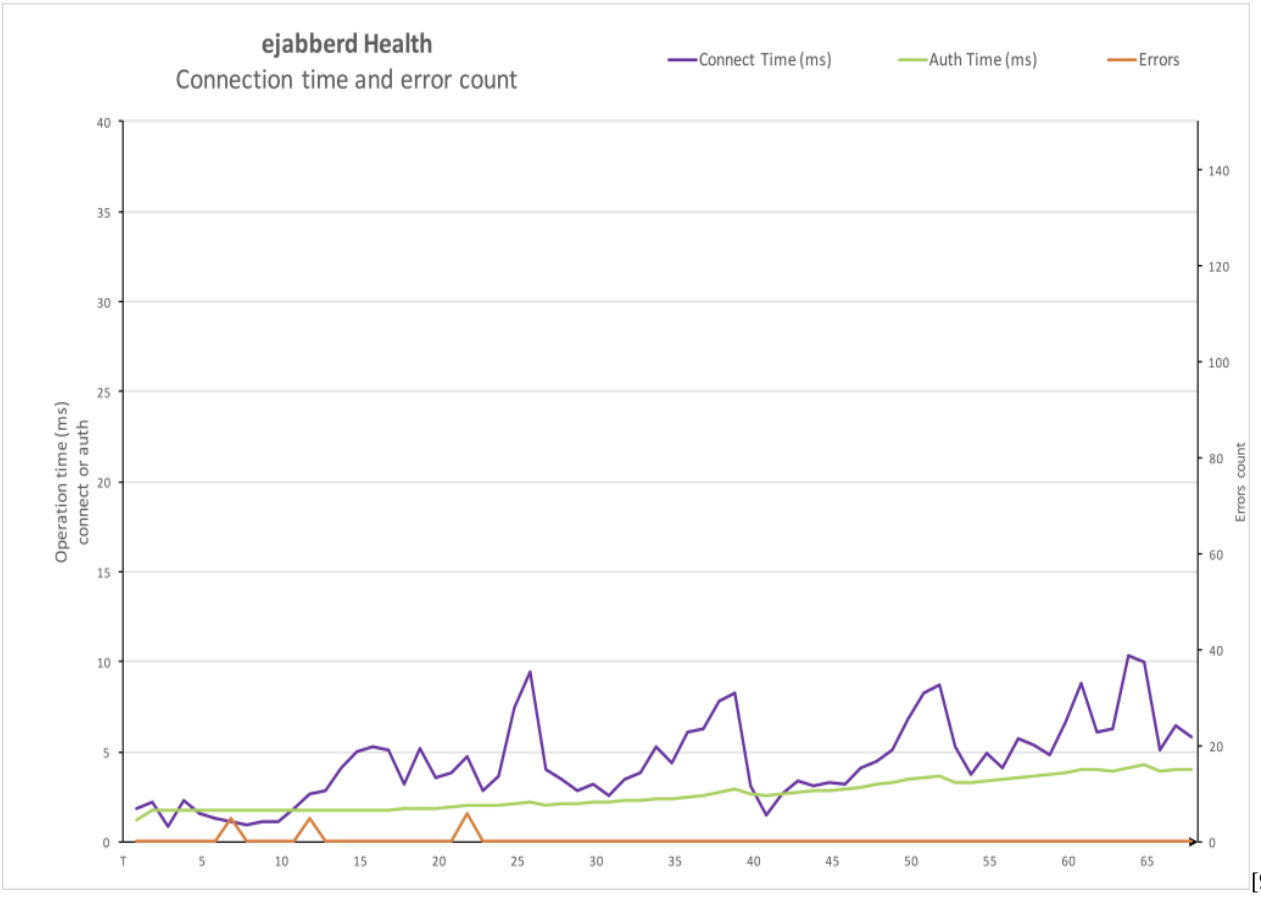
Good health and performance are confirmed by the state of the platform. CPU and memory consumption were totally under control, as shown in the graph. CPU consumption stays far from system limits. Memory grows proportionally to the number of concurrent users.[9]

We also need to mention that values for CPUs are slightly overestimated as seen by the OS, as Erlang schedulers stay a bit of busy waiting when running out of work.[9]



[9]

Figure 2 : Concurrent Users and Traffic



[9]

Figure 3 : Connection time and error count

[9]

Figure 4 : Ejabberd node CPU and memory

5. CONCLUSION

Thus we conclude that XMPP is not merely a set of protocols for defining behavior of XML, and its potential reaches far beyond the horizons of Instant Messaging. As there is a surge in online server-based games, XMPP is a feasible solution because of the similarities between IM servers and Gaming servers. We found out that numerous XMPP (XEPs) allow easy transmission of information through Transmission Control Protocol. We have listed a number of XMPP applications in modern games which show that XMPP can be scaled as per the breadth of Gaming needs. ProcessOne's Ejabberd and IgniteRealtime's Openfire, both XMPP servers are concluded as best taking consideration of their compliance with current XEPs. Hence, we believe that this approach will be of immense help to open-source gaming servers satisfying all major requirements and a significant step towards the development of advanced Game Engines.

6. ACKNOWLEDGEMENT

I am happy to acknowledge the efforts of the entire XMPP community and the founder of XSF(XMPP Standards Foundation) Peter Saint Andre for developing highly scalable and robust XMPP architecture and for RFC 6120 (XMPP Core) and RFC 6121 (XMPP : Instant Messaging and Presence).

7. REFERENCES

[1] Peter Saint Andre , "XMPP : The Definitive Guide, Building Real-time Applications with Jabber Technologies".

[2] Peter Saint Andre, "Extensible Messaging and Presence Protocol (XMPP) : Core".

[3] Peter Saint Andre, "Extensible Messaging and Presence Protocol (XMPP) : Instant Messaging and Presence".

[4] Daniel Schuster, Thomas Springler, Istvan Koren, Markus Endler, "Creating Applications for Real-Time Collaboration with XMPP and Android on Mobile Devices.

[5] Oracle Java Documentation, "Why StAX?" .

[6] XMPP Standards Foundation (XSF), "Online Games using XMPP".

[7] Jive Software, "WhitePaper".

[8] Tsung Erlang Projects, "Documentation".

[9] ProcessOne's Ejabberd, "Documentation".

[10] XMPP Standards Foundation (XSF), "Instant Messaging".

[11] Jeremie Miller, Thomas Muldowney, Peter Saint-Andre, "XEP-0012: Last Activity".

[12] Thomas Muldowney, "XEP-0027: Current Jabber OpenPGP Usage".

[13] Joe Hildebrand, Peter Millard, Ryan Eatmon, Peter Saint-Andre, "XEP-0030: Service Discovery".

[14] Peter Saint-Andre, "XEP-0045: Multi-User Chat".

[15] Peter Saint-Andre, "XEP-0054: vcard-temp".

[16] Peter Millard, Peter Saint-Andre, Ralph Meijer, "XEP-0060: Publish-Subscribe".

[17] Peter Saint-Andre, Tobias Markmann, "XEP-0082: XMPP Date and Time Profiles".

[18] Ian Pateron, Dave Smith, Peter Saint-Andre, Jack Moffitt, Lance Stout, Winfried Tilanus, "XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH)".

[19] Scott Ludwig, Joe Beda, Peter Saint-Andre, Robert McQueen, Sean Egan, Joe Hildebrand, "XEP-0166: Jingle".

[20] Peter Saint-Andre, Valerie Mercier, "XEP-0172: User Nickname".

[21] Ian Pateron, Dave Smith, Peter Saint-Andre, Jack Moffitt, Lance Stout, Winfried Tilanus, "XEP-0206: XMPP Over Bosh".

[22] Philipp Hancke, Dave Crindland, "XEP-0288: Bidirectional Server-to-Server Connections".

[23] Daniel Gultsch, "XEP-0363: HTTP File Upload".