Global Journal of Computer Sciences: Theory and Research

Volume 06, Issue 1, (2016) 10-17

www.gjcs.eu

Extensible messaging and presence protocol's adaptation to business applications

Halil Arslan*, Cumhuriyet University, Sivas Vocational School, 58140 Imaret/Sivas Merkez/Sivas, Turkey.
Sinan Tuncel, Computer Engineering, Sakarya University, 54050 Serdivan/Sakarya,Turkey.
Osman Gun, Detaysoft, 34692 Uskudar, Istanbul, Turkey.

Suggested Citation:
Arslan, H., Tuncel, S. & Gun, O (2016). Extensible messaging and presence protocol's adaptation to business applications. *Global Journal of Counseling and Guidance in Schools: Current Perspectives*. *6*(1), 10-17.

Received April 15, 2016; revised May 10, 2016; accepted June 05, 2016.
Selection and peer review under responsibility of Prof. Dr. Dogan Ibrahim, Near East University, Cyprus.

Abstract

Communication protocols are used for data communication between different parties in computer networks. Research on improvement and generalization of the usage areas of these protocols keeps going. One of these protocols, Extensible Messaging and Presence Protocol (XMPP), draws the attention with the advantages of providing an open communication infrastructure for real-time transmission and usage of XML format along with packet definition, which is also available in other protocols. In this study, a new model was developed making the use of XMPP advantages. The developed model can interoperate with the software using other protocols and is intended for efficient, location-independent communication of people and business applications with each other. To verify the constructed model, a new enterprise application including an instant messaging service was developed that is able to interoperate with business applications. A new area was added to the usage areas of XMPP that contributes to the generalization of the protocol.

Keywords: XMPP, business application, instant messaging, presence management

* ADDRESS FOR CORRESPONDENCE: **Halil Arslan,** Cumhuriyet University, Sivas Vocational School, 58140 İmaret/Sivas Merkez/Sivas, Turkey. *E-mail address:* harslan@cumhuriyet.edu.tr / Tel.: +90-536-412-6484

1. Introduction

Instant messaging (IM) is an IP-based application that can be used with various devices, providing easy communication among users. Currently the widely known form of IM has voice and video attachment support along with peer-to-peer instant textual messaging. Instant messaging services have spread out with the ICQ since 1996 [1]. By the success of this technology, many firms have begun to work on similar products such as AOL, Yahoo and Live Messenger. Each of these applications is dependent on a specific network protocol that is operated by the developer firm itself. As a result, only the devices or software that use the same protocols, can communicate with each other. This problem led to the idea of development of decentralized instant messaging network and protocol. Extensible messaging and presence protocol [2], called Jabber in 1999 and currently known as XMPP, was defined with RFC 3920 and RFC 3921 which were published by Internet Engineering Task Force (IETF) and it is still being improved.

XMPP can construct autonomous networks between two servers, two clients or services that connect to each other and provide communication between pairs [3]. With XMPP protocol, users in the special chat networks can communicate with each other in the extent their services allow. This lets users to be able to communicate with the users in their own contact list via a product in the instant messaging market that has XMPP support.

When the previous studies have been examined, it can be seen that XMPP protocol has constituted a basis for many of them. But no XMPP extension that depends on the efficiency of the business processes, has been encountered. Pankaj et. al. [4] presented the theoretical basics of business applications, proposed suitable business applications and emphasized that peer-to-peer communication approach can provide suitable infrastructures in the context of these applications. Ragavan et. al. [5] showed that XMPP protocol can provide an important infrastructure for real-time industrial business applications by developing an application that has the ability of real time data gathering and controls various industrial applications using XMPP protocol. Bønes et. al. [6] studied the XMPP protocol to use it along with instant messaging service at the health sector and introduced risk analysis for mobile platforms. Sun et. al. [7] presented a hybrid encryption algorithm for the enterprise users that depends on the XMPP-based Bouncy Castle encryption library and applied this algorithm to the instant messaging phases. Gomes et. al. [8] presented the XMPP-based content management architecture and proposed a new model to provide user location information by adding a GPS node to the IQ packets. Lübke et. al. [9] presented a location-based group management service in the XMPP infrastructure for mobile software developers. Towards this goal, they presented a client-server architecture that is compatible with XMPP protocol. These studies show that this protocol will continually be improved for IP-based communication protocols and will provide significant benefits for enterprise applications.

The issue that is intended to solve with this study is the adaptation of XMPP protocol to the enterprise structures and integration of this protocol to the business applications. Besides instant communication (textual, auditory or visual) of company workers (stationary or mobile), an example model that can satisfy the business application requirements (to-do lists etc.) with maximum efficiency on the basis of internal communication, has been designed. While the model is being designed it was aimed to have a model that can carry the external stakeholders (firms, customers, sellers, project staff etc.) on it by taking the situations that are inconsistent with the firm policies into account such as the integration of public messaging applications to the business applications and enterprise user pools. As a result of the study, an enterprise software infrastructure model that interoperates with business applications, and includes an XMPP-based instant messaging service accepted as a standard by many applications (gtalk, skype, one team etc.) was constructed. This study will contribute to develop a business process extension for the XMPP standard.

The sections of the study are organized as follows: In section 2, it has been mentioned about the structure of XMPP protocol. In section 3, the model architecture intended for integration of business applications and XMPP protocol, is introduced. In section 4, the application user interface is shown and the results are presented. In the final section the results of the developed model are evaluated and future studies are explained.

2. XMPP

XMPP is a real-time communication protocol that uses XML packet format for data transmission between clients [2]. Basically, XMPP provides a suitable communication environment for XML-based data transmission that can be considered real-time between clients. As in web servers and e-mail services, XMPP has a decentralized client-server architecture. As shown in Figure 1, while everyone can manage his/her own XMPP server's security and scalability with decentralized client-server architecture, clients can be distributedly developed focusing only on the user experiences.

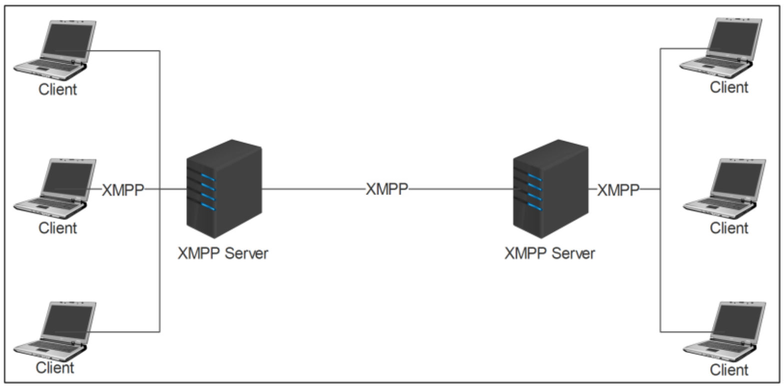


Figure 1. XMPP client-server architecture

2.1. *XMPP Addressing*

Every entity on the XMPP network should have a unique address called JID (jabber identity). A JID has a structure of a local name, a domain name and a resource such as user@domain/resource. User domain is used to identify a unique user in the related domain name as in the e-mail account case. The domain, which has to exist in every XMPP identity, is used to identify the subnet which belongs to that entity in the XMPP network. And the resource is used to identify which client/resource the user used in case he/she can have connection using different clients. A user can create similar sessions concurrently over different resources or it can be prevented by request [1, 3, 10].

2.2. *XMPP Signaling and Packet Structures*

When a long-term TCP connection is established between client and server, the data flow continues with XML format after the <stream:stream> tag and ends with </stream:stream> tag. This structure can be considered as <html>…</html> web structure. In XMPP, there are three basic XML packet format definitions as message (<message>), presence (<presence>) and info/query (<iq>) [2].

Message packet (<message>) can be used to carry every kind of custom information between clients as it is basically used in message transmission between clients for instant messaging systems. The defined message packet types are normal, chat, groupchat, error and handline. Normal message is the message to be sent to the offline user, chat is the message that is directly sent to the online user, and groupchat is the message to be sent to a chat room. Error message type is the response packet that returns in some cases such as the correspondent user can not be discovered in the XMPP network after the message is sent. Headline messages are the messages that are automatically generated and used for information such as the RSS feeds which are used as web page feeds. A sample message packet structure is shown in Figure 2.

```
<message type='chat' id='e9bf4' from='c1@domain/resource' to='c2@domain' xmlns='jabber:client'>
     <body>Hi</body>
</message>
```

Figure 2. Sample message packet

The presence packet (<presence>) is the packet structure that is used to manage the status of the user in an XMPP network such as online, busy and offline. Thus it is ensured that the users can check each other's availability. The presence packets of a client are distributed only with the other users on the contact list which are approved by that client. A sample presence packet structure is shown in Figure 3.

```
<presence xmlns='jabber:client'>
     <show>Will return</show>
     <priority>0</priority>
     <status>Away</status>
</presence>
```

Figure 3. A sample presence packet

If a packet that is transmitted between client and server is not a message or presence packet, then it is configured as an Info/Query (<iq>) packet. Info/query packets are usually treated as request/response packets. The extensibility of the protocol is provided by specializing this packet structure. A sample info/query packet structure is shown in Figure 4.

```
<iq from='client1@domain.com/resource' id='d3079' type='get' xmlns='jabber:client'>
     <query xmlns='jabber:iq:roster' />
</iq>
```

Figure 4. A sample info/query packet

2.3. XMPP Security and Session

XMPP provides Transport Layer Security (TLS) encryption support by default. Clients can initialize a TLS session when the TLS support is supplied by the XMPP server. Therefore all data flow between the client and server is encrypted using TLS. After this phase, client can realize authentication via Simple Authentication and Security Layers (SASL) protocol by choosing one of the security mechanisms such as md5, plain or token [10]. The sample packet structures of declaration of the authentication mechanisms by the server and declaration of user identity by the client using md5 mechanism in return are shown in Figure 5.

```
Server:
<stream:features>
 <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
     <mechanism>INTERNAL</mechanism>
     <mechanism>DIGEST-MD5</mechanism>
     <mechanism>PLAIN</mechanism>
 </mechanisms>
 <auth xmlns='http://jabber.org/features/iq-auth'/>
</stream:features>
```

```
Client:
<auth mechanism='DIGEST-MD5' xmlns='...'>
  Y2xpZW50MSQ1QTgw....00NEYwQUMxQzdCMjA
</auth>
```

Figure 5. XMPP authentication phases

3. Model Design

The objective of this study is to develop an XMPP-based business applications module. A To-Do List management application, which is needed by many enterprise firms, is implemented as a practice of the objective. The companies require serious infrastructures to define the jobs that will be tracked by their workers and to track the jobs themselves. But the applications which cannot be distributedly used in real-time, lead to the situations where they are only used by specific people and cannot be reflected to the whole firm. In this study, it is assumed that a To-Do List management application, integrated with the daily instant messaging applications, will be a solution suitable with the utilization habits for the people spending most of their time with computers. The layer architecture of the study is shown in Figure 6.

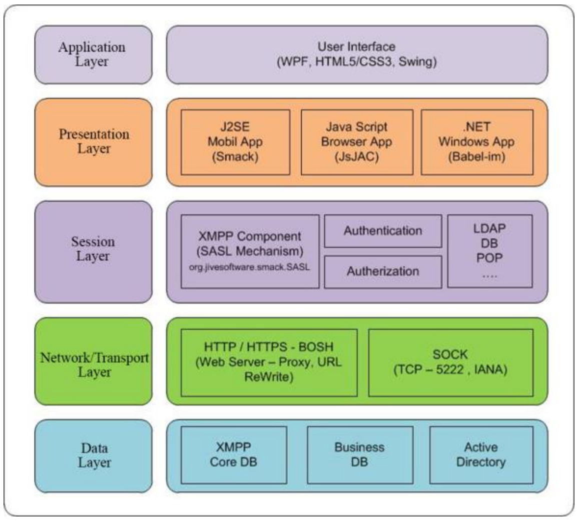

Application Layer
User Interface (WPF, HTML5/CSS3, Swing)
Presentation Layer
J2SE Mobil App (Smack)
Java Script Browser App (JsJAC)
.NET Windows App (Babel-im)
Session Layer
XMPP Component (SASL Mechanism) org.jivesoftware.smack.SASL
Authentication
Autherization
LDAP DB POP ....
Network/Transport Layer
HTTP / HTTPS - BOSH (Web Server – Proxy, URL ReWrite)
SOCK (TCP – 5222 , IANA)
Data Layer
XMPP Core DB
Business DB
Active Directory

Figure 6. Application layer model

The model shown in Figure 6 consists of 5 basic layers. In the data layer, the database including the XMPP core components required by the application, the business application database, and the database that stores the users are defined. The network/transport layer used for client-server communication where the default TCP port 5222 of XMPP and http/https connections are created, is defined in the communication and network layer. The session layer where the XMPP authentication mechanisms are executed, is located in the session layer. The layer, where the XMPP Application Programming Interfaces (APIs) used by the client applications are defined, is located in the presentation layer and the layer, where the client interfaces are developed, is located in the application layer. The network configuration of the application developed in the basis of the layer model is shown in Figure 7.

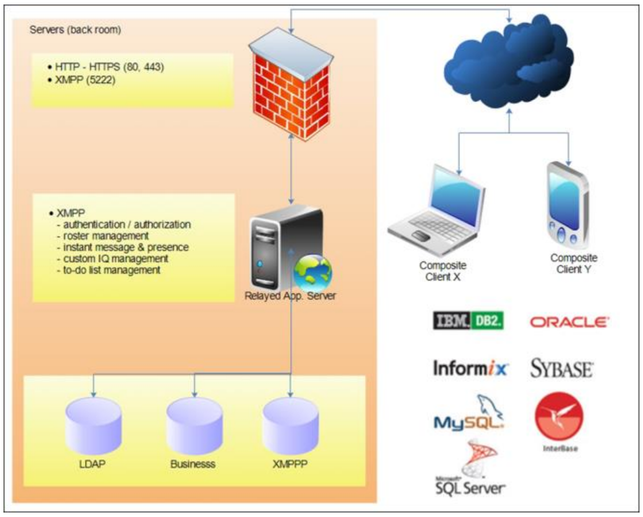


Figure 7. Application network configuration

XMPP server uses TCP 5222, http 80 and https 443 ports in the application network configuration. At this point, along with standard instant messaging processes, XMPP server also implements to-do list management functionality using custom info/query (IQ) packets as shown in Figure 8 and Figure 9.

```
<iq from='client1@domain.com/resource' id='36a42' type='get' xmlns='jabber:client'>
     <todocommand xmlns=todolist:iq:customiq:gettodolist' />
</iq>
```

Figure 8. Custom to-do list request packet

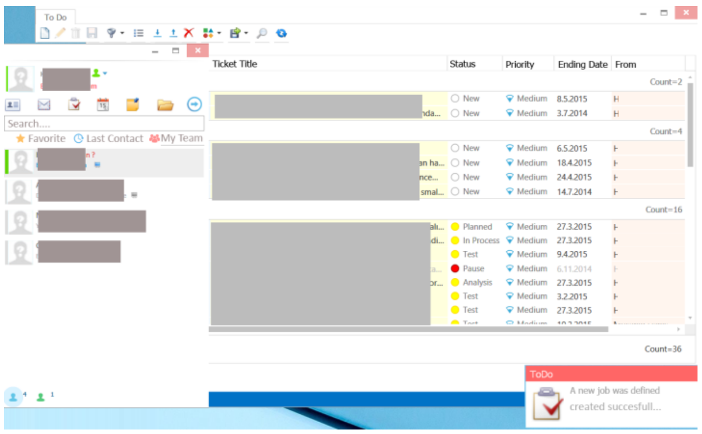
```
<iq from='client1@domain.com/resource' id='36a42' type='set' xmlns='jabber:client'>
     <todocommand xmlns=todolist:iq:customiq:settodolist'>
          <project>project ID</project>
          <title>ticket title</title>
          <persons>
               <to>person id</to> <!-- responder person -->
               <cc>person id</cc> <!-- related person 1 -->
               <cc>...</cc>       <!-- related person n -->
          </persons>
          ...                     <!-- ticket attributes -->
     </todocommand>
</iq>
```

Figure 9. Custom to-do definition packet

When the XMPP server has the GET type info/query packet as shown in Figure 8, it returns the to-do list which resides in the to-do list database of the client who made the request. The client can define a new job to the personnel in the contact list (firm authorized personnel list) using the SET type info/query packet. The new job defined by the client is linked and distributed to the staff between the <persons> tag as a real-time custom <message type=headline> packet. Thus the staff to whom the job was linked, can be aware of the changes in their to-do lists. All the request and response packets in a to-do list management application were defined using similar packet structures.

4. Application and Tests

In the development of the designed architectural model, open-source Java-based OpenFire 3.9.3 was used as the application server [11]. OpenFire is an XMPP server application that supports lots of additional protocols along with XMPP standard libraries and it has a wide developer support. The user interface of the developed application is shown in Figure 10.



Ticket Title	Status	Priority	Ending Date	From
				Count=2
[redacted]	New	Medium	8.5.2015	H
[redacted] nda...	New	Medium	3.7.2014	H
				Count=4
[redacted]	New	Medium	6.5.2015	H
[redacted] an ha...	New	Medium	18.4.2015	H
[redacted] nce...	New	Medium	24.4.2015	H
[redacted] smal...	New	Medium	14.7.2014	H
				Count=16
[redacted] ali...	Planned	Medium	27.3.2015	H
[redacted] di...	In Process	Medium	27.3.2015	H
[redacted]	Test	Medium	9.4.2015	H
[redacted] a...	Pause	Medium	6.11.2014	H
[redacted] or...	Analysis	Medium	27.3.2015	H
[redacted]	Test	Medium	3.2.2015	H
[redacted]	Test	Medium	27.3.2015	H
				Count=36

ToDo
A new job was defined
created succesfull...

Figure 10. Screenshot of the application user interface

The sample screenshot demonstrates the main screen, the declaration of the new defined job (with the subject: "A new job was defined") to the linked staff, and the to-do list all together.

Two computers, configured as a client and a server, were used as a test environment. The client has an 8 Mbps bandwidth ADSL internet connection while the server has a 5 Mbps bandwidth metro internet connection. 10, 50, 100, 250 and 500 users have been defined respectively as client1, client2,... using the Smack API [11] on the OpenFire. After these users have initialized a session on the XMPP server, it has been ensured that they change their status randomly between 10-30 sec. when they are at each other's contact list. In addition, every user sends an empty chat-type message packet randomly in 5-10 sec. intervals. While the server runs on the defined user loads, the client defines a new job for each case and the response time of the arrival of this new defined job to the related user is measured. This process is repeated 10 times for each case and the average time is calculated. Response time of the system for different loads is shown in Figure 11.

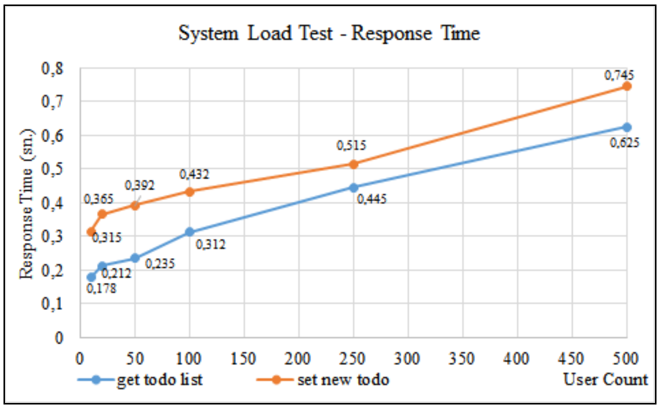


Figure 11. Application load test results

When the load-test results are examined with these parameters, it can be seen that the business application response time is close to the real-time communication values when there are 500 online users. When the number of online users exceed 500, load distribution can be stabilized by adding new servers.

5. Conclusion

With this study, a model that can efficiently satisfy the business application requirements along with the instant communication of company workers, is designed using the extensible structure of XMPP protocol. A software has been developed to verify the designed model and the performance of the model and the software was observed. The adaptation of the XMPP protocol to the enterprise structures and the integration of this protocol with the business applications were satisfied using the proposed model.

The improvements and the custom XML packet configurations can be widened by saving them to the XMPP Standards Foundation as an additional XMPP protocol. They can also be presented to the researchers that work on similar studies with the purpose of contributing to the development of a business process extension for XMPP standard.

Acknowledgements

This study is supported by TÜBİTAK in the context of the TEYDEB (7130452) project.

References

[1]Shigeoka I., Instant Messaging in Java, *Manning Publications Co, 2002.*

[2]Saint-Andre P., Extensible Messaging and Presence Protocol (XMPP): Core, *IETF RFC 3920, 2004*.

[3]Moffitt J., Professional XMPP Programming with JavaScript and jQuery, *Wiley Publishing Inc., Indianapolis, Indiana, 2010*.

[4]Pankaj P., Hyde M., and Rodger J. A., P2P Business Applications: Future and Directions, *Communications and Network, 2012, 4, pp 248 – 260*.

[5]Ragavana S. V., Kusnantoa I. K., Ganapathyb V., Service Oriented Framework for Industrial Automation Systems, *Procedia Engineering 2012, 41, pp 716 – 723*.

[6]Bønes E., Hasvold P., Henriksen E., Strandenæs T., Risk analysis of information security in a mobile instant messaging and presence system for healthcare, *International Journal of Medical Informatics, 2007, 76, 677 – 687*.

[7]Sun X., Du Z., Chen R., A Secure Cross-platform Mobile IM System for Enterprise Applications, *International Conference on Uncertainty Reasoning and Knowledge Engineering, 2011, pp 158 – 161*.

[8]Gomes D., Gonçalvesy J. M., Santosy R. O., Aguiar R., XMPP based Context Management Architecture, *IEEE Globecom 2010 Workshop on Enabling the Future Service-Oriented Internet, 2010, pp 1372 – 1377*.

[9]Lübke R., Schuster D., Schill A., MobilisGroups: Location-based Group Formation in Mobile Social Networks, *Second IEEE Workshop on Pervasive Collaboration and Social Networking, 2011, pp 502 – 507*.

[10]Saint-Andre P., Smith K., Tronçon R., XMPP: The Definitive Guide Building Real-Time Applications with Jabber Technologies, *Published by O'Reilly Media Inc., 2009*.

[11]Ignite Realtime: Openfire XMPP Server. 2014-11-06, Available: http://www.igniterealtime.org.