A New Approach of Instant Message Extended from Short Message Service with Jabber

Heng-Te Henry Chu

Dept. of Product Development, Mobitai Communications Corporation, Taichung, Taiwan

henry_chu@mobitai.com.tw

Yi-Hung Huang*

*Dept. of Information Networking Technology,

Hsiuping Institute of Technology, Da-Li, Taichung, Taiwan

ehhwang@mail.hit.edu.tw

Wen-Shiung Chen**

**VIP-CCLab., Dept. of Electrical Engineering, National Chi Nan University, Pu-Li, Nan-Tou, 545 Taiwan

wschen@ncnu.edu.tw

Abstract

Nowadays, short message service has been very successful in carrier business. Instant message service also gains popularity through Internet. To bridge them is becoming a new promising niche because people really enjoy getting messages instantly, anywhere and anytime. However, there is still not a common, unified and open standard to communicate with each other. Hence, in this paper we propose a solution, based on XML-based protocol, Jabber, to simplify interconnections among Internet instant message systems and short message systems. Currently, several providers, such as AOL, MSN, Yahoo, and ICQ dominate Internet instant message systems via proprietary protocols. On the other hand, every carrier operator has been trying to refine its own homemade interface to short message application services with the intention to cover the complexity and security of the communication with its own short message service center. Since Jabber is an XML-based protocol, which is in human-readable format, it can be used to cover different underlying protocols and present a unified and easier interface to message applications and services. Obviously, it is much easier than SS7 or any other protocols of proprietary or binary data formats, to achieve message exchange and interoperability among different systems, such as GSM, PHS, CDMA and 3G. An XML-based protocol is also purely an IP-based solution, which comes out to be much cheaper than a telecommunication-based solution to adopt and upgrade. Furthermore, an IP-based protocol can be easily translated to and from any other IP-based protocol by way of modern programming languages.

Keywords : SMS, SMPP, IM, Jabber, XML

1. Introduction

Enterprises and consumers have discovered that instant message service [1][2] is more cost-effective than e-mail. People may rely on instant messages to improve their work efficiency. Businesses and applications may count on instant message services for faster message flow processing. In reality, mobile handsets with short message services [6] are ideal tools for people to get messages anywhere and/or anytime, instantly.

A short message service center (SMSC) provides an old-fashioned protocol, i.e., short message peer-to -peer protocol (SMPP) [6], for IP-based applications to communicate with. The SMPP speaks in binary data format, instead of human-readable text formats. However, the SMPP is so peculiar to most application developers that Internet content providers (ICP) view it as a technical barrier to join. On the other hand, major Internet instant message service providers (e.g., AOL, MSN, Yahoo, ICQ, and so on) are apt to use their own proprietary interface to secure their investment. Certainly, those proprietary protocols block interoperability and also raise another barrier to ICP.

Since XML is in human-readable text format, it is open, flexible, portable, and simple to create and read [5]. Beneficially, most modern programming languages, e.g., Java, begin to support XML parsing and processing [10]. Jabber [3][4] is an open-source, XML-based protocol. Accordingly, Jabber is capable of being open to non-Jabber systems in nature. With a plug-in component, called Jabber transport, a Jabber server may communicate with a non-Jabber system. Similarly, a Jabber client may talk to a non-Jabber community as well as to another Jabber client. Hence, the Jabber transport may cover up the complexity of SMPP, and still exposes the same XML-based interface

to outside world for simplicity. Based on XML, interoperability between different proprietary interfaces (or instant message service systems) is easy and foreseeable to achieve.

2. Jabber Architecture

The basic Jabber communication model follows the well-understood and simple client/server architecture, as shown in Fig. 1. Unlike those peer-to-peer approaches, Jabber encourages to implement centralized control and enforce communication policies. On the one hand, a Jabber server enforces those control policies. On the other hand, the Jabber protocol keeps its client as thin as possible, for sake of easy development and joint.

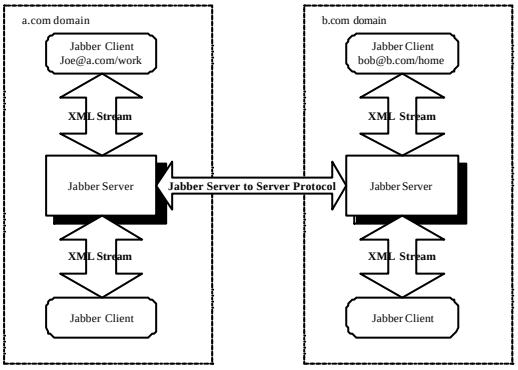

Fig. 1. The Jabber client/server architecture.

Like e-mail, a Jabber domains is defined by an Internet domain name. The Jabber servers manage the Jabber domains. The Jabber addressing format is similar to e-mail address, like User@domain/resource, except the resource part. The domain part guides how to relay messages from one domain to another. The resource part indicates a particular message delivery endpoint for a user, e.g., Joe@xyz.com/work or Joe@xyz.com/mobile, as shown in Fig. 2. All Jabber data are delivered to resources. As shown in Fig. 3, a Jabber server takes charge of parsing incoming XML streams and routing outgoing XML packets to the best and/or preferred client's resource available for a user. An XML packet contains valid XML subdocuments.

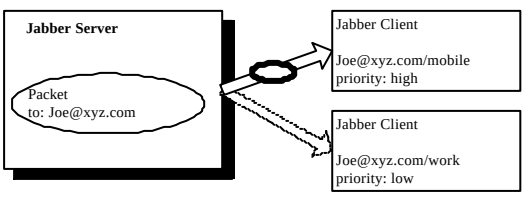

Fig. 2. Routing to the best and available client.

It is worth noting that instant messaging occurs across space and time. In other words, it means how to determine where to deliver across network, and when to reach as soon as recipients becoming available. Fig. 3 shows that the Jabber server designs a packet queue to store and forward the packets from the XML parser. Instead of defining its own queuing mechanism, Jabber allows to make use of existing message queue technologies or products.

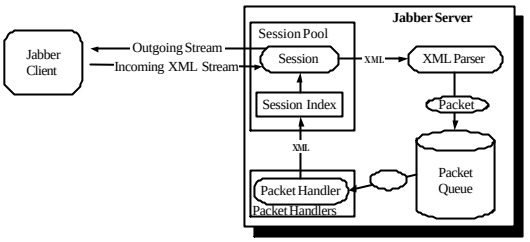

Fig. 3. Basic functional modules of the Jabber server.

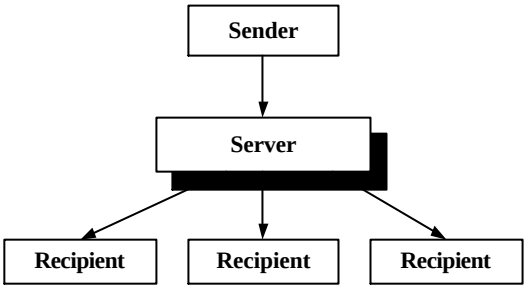

Fig. 4. The Jabber server arbitrates all presence exchanges.

Jabber relies on user presence information to determine best target for a user. Instead of having every user sending their presence to other users, Jabber comes out with the concept of presence subscription, as shown in Fig. 4.

Subscribers must send a request to the publisher. The publisher may accept or refuse to reveal its presence update for privacy concern. Each user must manage its own publisher and subscriber presence relationships. As shown in Fig. 4, the Jabber server hosts, organizes and maintains user subscriptions, and arbitrates all presence exchanges via the so-called roster. Substantially, a roster is similar to a "buddy list." Each user account has only one corresponding roster, but many sessions, each with its own presence status (e.g., your PC is off, but your mobile handset is at hand). However, people just care if they can send messages to another person, regardless of what client or device to be. Jabber will rely on user presence for message routing and store-and-forward delivery, in order to reach the best available client.

Since Jabber is an open and packet-based design, it uses modules, called transports, which act as a bridge between Jabber and those foreign non-Jabber messaging systems. The Jabber transport acts as the Jabber server plug-in, translating packets between different instant message systems, in order to provide seamless access to both, as shown in Fig. 5.

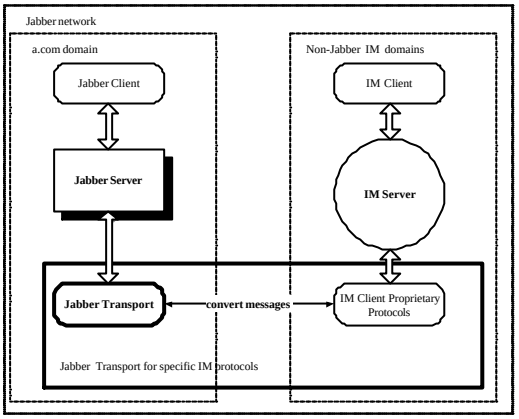

Fig. 5. The Jabber transport.

Although XML is in human-readable text format, the content itself may be secured independently. However, regarding to security concern, Jabber is open to adopt existing security technologies and products on the market. As shown in Table 1, for example, Jabber may adopt LDAP, Kerebos, or Java Authentication and Authorization Service (JAAS), for authentication and authorization. Content itself may be secured by way of message digest for integrity, digital signature for non-repudiation, encryption for confidentiality. Furthermore, concerning with the security of communication channel, the Jabber server may utilize SSL to secure its TCP communications with clients as well.

Table 1. Security Concerns and Solutions.

Security Concern	Description	Existing Technology
Authentication	Checking who it is	LDAP, JAAS, Kerebos, …
Authorization	Checking what its access rights to be	LDAP, JAAS, ...
Integrity	Checking if data remain intact	Message digest
Non-repudiation	Ensuring the source of data	Digital signature
Confidentiality	Ensuring data to reveal to only right entities	Encryption

3. Short Message Peer to Peer Protocol

For brevity, we focus only on its support with GSM technology and TCP/IP network. The SMPP is based on the exchange of request and response protocol data units (PDUs) between the external short message entity (ESME) and the SMSC over an underlying TCP/IP network connection. Here, the ESME may be a ticket system, an application for headline news, or an advertisement broadcast, as shown in Fig. 6.

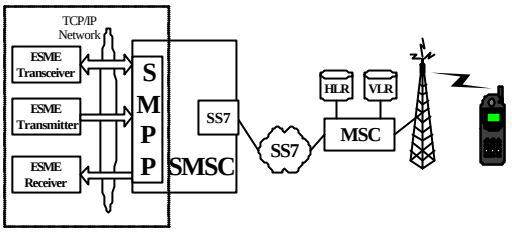

Fig. 6. The SMPP interface.

The SMPP session may be defined in terms of the following possible states:

- **OPEN** (Connected and Bind Pending)
  An ESME has established a network connection to the SMSC but has not yet issued a bind request.
- **BOUND_TX**
  A connected ESME has requested to bind as an ESME transmitter
- **BOUND_RX**
  A connected ESME has requested to bind as an ESME receiver.
- **BOUND_TRX**
  A connected ESME has requested to bind as an ESME Transceiver. An ESME bound as a transceiver supports the complete set of operations supported by a Transmitter ESME and a receiver ESME.
- **CLOSED (Unbound and Disconnected)**
  An ESME has unbound from the SMSC and has closed the network connection. The SMSC may also unbind from the ESME.

An SMPP session always begins with sending a bind request for authentication first from ESME to SMSC, before transferring any message. It finally ends up with a unbind request to close the session, as shown in Figure 7 and Figure 8.

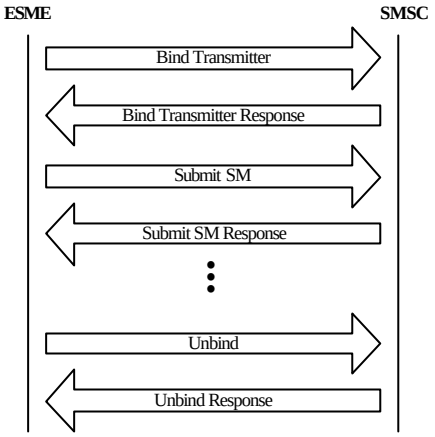

Figure 7. Typical SMPP session sequence - ESME transmitter.

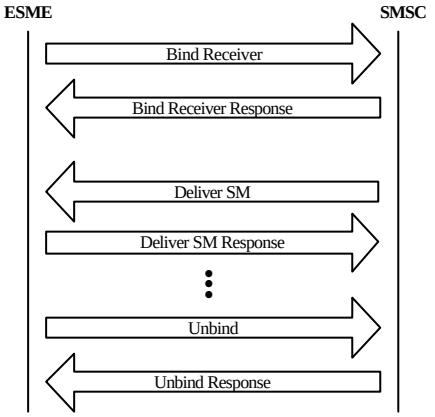

Figure 8. Typical SMPP session sequence - ESME receiver.

As shown in Table 2, command length, ID, status, and sequence number are with a type of 4-octet unsigned (big-endian) integer. The PDU body may contain mandatory and/or optional parameters corresponding to command ID field, defined by the SMPP protocol. Moreover, a GSM short message contains up to 160 7-bit characters or 140 8-bit octets [7][8]. The 8-bit data are in UCS-2 [9], 16-bit encoding. A conversion between Big-5 and UCS-2 is required for implementation. Thus, the SMPP PDU is much harder than XML to compose.

Table 2. An overview of the SMPP PDU format.

SMPP PDU				
PDU Header (Mandatory)				Body (Optional)
Command Length	Command ID	Command Status	Sequence Number	PDU Body
4 Octets	Length = (Command Length value - 4) octets			

The SMPP uses sequence number to match request and response packets. The sending side takes charge of filling in a packet with sequence number in a unique and sequential manner. The receiving side must reply with the same sequence number in its response packet. The sequence number can distinguish session information at all. Sequence numbers of a user may be interleaved by numbers of another users.

Since handsets always receive messages after a certain amount of delay, the ESME may need to know actual delivery statuses. The SMSC can answer it via a very practical message type: delivery receipt, which tells (via delivery PDU) the final status: delivered, expired, rejected, undeliverable, unknown, deleted or accepted.

Typically, the SMSC supports three message modes:

- Store and Forward
- Datagram
- Transaction mode

The conventional approach stores the message in a SMSC storage area before forwarding the message for delivery to the recipient. With this model, the message remains securely stored until all delivery attempts have been made by the SMSC. This mode of messaging is commonly referred to as "store and forward."

The datagram mode emulates the datagram paradigm, like UDP datagram. This mode focuses on high message throughput without the associated secure storage and retry guarantees of store-and-forward message mode. In this mode, the ESME does not receive any delivery acknowledgement.

The transaction mode is designed for applications that involve real-time messaging without the need for long term SMSC storage. The ESME requires a synchronous end-to-end delivery outcome. However, this mode could cause serious performance degradation while system load is heavy. Domestic carrier operators are not willing to open datagram and transaction modes, due to reliability and performance concerns.

4. Existing Proprietary Interfaces

Currently, every domestic carrier operator opens one's own proprietary interfaces to ICP for short message value-added services, as shown in Fig. 9. Those proprietary interfaces can be mainly categorized into the following three types:

1. HTTP + Query String, e.g.,

   http://a.com?ID=123&MSG=hello&…

2. HTTP POST method + XML body, e.g.,

```
<?xml version="1.0" encoding="Big5"?>
  <sms>
    <ID>123</ID>
    <MSG>Hello</MSG>
    …
  </sms>
```

3. TCP socket + Home-made PDU

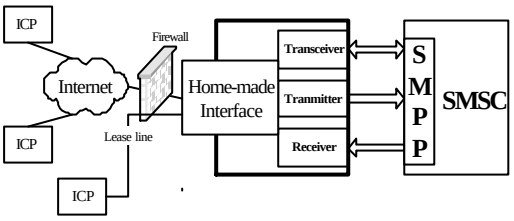

Figure 9. Proprietary Interface Overview.

The first type is easy and straightforward at first glance. However, it introduces some problems, such as security, version control, and parsing fields. Obviously, it is vulnerable to attack. To change the query string formats causes tedious modifications on both operator and ICP sides. Meanwhile, the common delimiter, &, may cause the conflicts while paring and filtering the query string [12].

The second type resolves the disadvantages of the first type. Instead of appending data to URL, it uses the HTTP POST method to convey enclosed data to remote server [11]. Business complexity can be hidden inside the context. It keeps HTTP URL intact and clear. However, its user-defined XML tags and formats are still totally proprietary. Other carrier operators cannot recognize them at all. Thus, an ICP still has to deal with different XML in order to speak in a right language with a right operator. The third type actually seems to take place first. It derived from the SMPP directly in certain degree. Inevitably, it comes along with the difficulty as much as the SMPP.

However, major instant message service players are still resistant to proceed for an open interface. They intentionally use their own proprietary interface to grape their territory and exclude any free interconnection, for sake of securing their investments. Interconnections become legal issues and require to paying for permission first.

5. The Proposed Interconnection

The Jabber transport plays a key role to realize the bridge. The Jabber transport is appointed by zero or multiple unique Jabber IDs, e.g., 0987654321@operator.com.tw. Sending messages to those Jabber IDs cause them to be handled by this transport component. Then, the transport will pass those messages to a converter component to translate data from the XML format into the SMPP PDU protocol data units. The converter will pass the SMPP PDU to the ESME component. The ESME component will submit them to the SMSC for final delivery and vice versa, as shown in Fig. 10.

These three components, Jabber transport, converter, and ESME, can cohere together inside a single application, or run independently among different machines or applications, as long as their data path and throughput being fine.

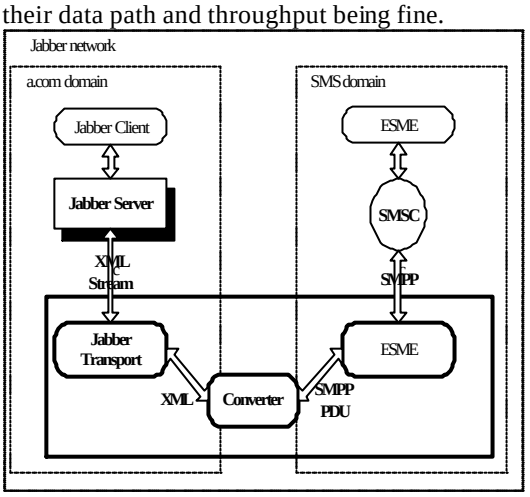

Figure 10. Interconnection Overview

The start of a Jabber or SMPP session needs to open a TCP connection first through its corresponding server port. The scenario is straightforward and is described as follows:

1. Connect port (e.g. 5222) of Jabber server.
2. Send an opening <stream:stream> tag containing server address
3. Wait for the server's<stream:stream> reply and record the stream's session ID
4. Use Jabber authentication protocol to login
5. Follow Jabber protocols to send packets. Jabber server will route them to appropriate recipients
6. Send a closing </stream:stream> tag to close the stream
7. Close the network connection.

On the other side, assuming that SMSC has created client accounts and granted access rights to port X already, the scenario is similar and is described as follows:

1. Connect SMSC on port X
2. Compose and send both SMPP bind_transmitter and bind_receiver PDUs to SMSC.
3. Wait for bind responses.
4. Use submit_sm PDUs to pack and send messages to SMSC for further delivery. Then wait for corresponding submit_sm responses. Or, wait for deliver_sm PDUs from SMSC for receiving messages, and then, acknowledge SMSC with deliver_sm response PDUs.
5. Send Unbind PDUs and close the connection.

Once the interconnections to both protocol stacks get ready, conversion between XML and binary SMPP PDU is the remaining task to plug in.

While data moving toward mobile network, the actual delivery may reach real handsets or another ESME applications since the SMSC is capable of intercepting a mobile number and re-route messages to a predefined ESME. However, an interception of the messages originated from subscribers of foreign network requires an advanced feature, so-called virtual number capability. Technically, the SMSC runs a virtual HLR inside, which will answer and guide SS7 network how to route messages to a virtual mobile number. The virtual mobile number does not map to any existing handset. Instead, it is bound with a pre-defined ESME application by the SMSC. The application is constantly listening to any incoming messages for further processing, as shown in Figure 11.

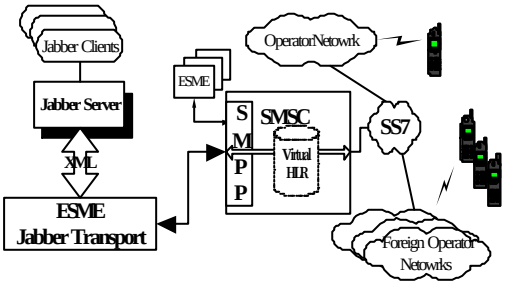

Figure 11. SMSC with virtual number capability

Therefore, the SMSC with virtual number capability will be capable of receiving and forwarding messages originated from mobile subscribers of all operators to pre-defined ESME applications. However, the virtual number capability does not matter with sending message from an application to any operator network.

By way of the SMSC virtual number, the Jabber server will save itself from interconnection with all operators, since it may receive message originated from all operators via the SMSC of a single operator.

Furthermore, as shown in Figure 12, carrier operators may adopt the Jabber XML-based protocol as an IP-based solution to interconnect with each other, instead of the conventional SS7 protocol. Such a unified environment will make everyone to talk each other in a much cheaper and easier way.

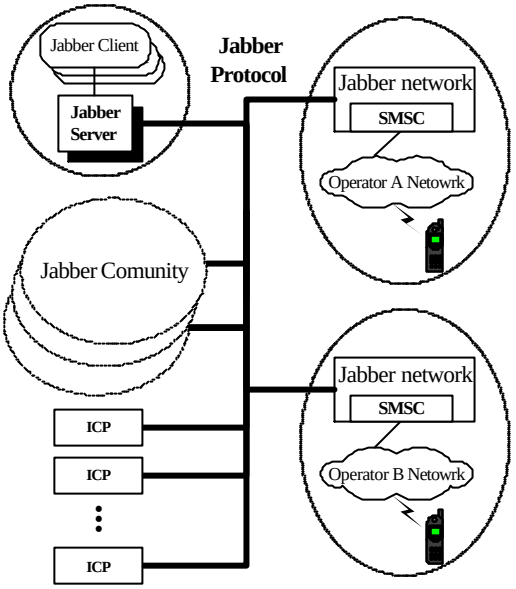

Figure 12. Speaking in one protocol and reaching everywhere.

6. Conclusion and Future Work

Telecommunication industry and IP-based network technology progress so quickly that message traffic volume will boom for sure. Operators and ICP can gain much more revenue, and users can enjoy more for sure. By adopting existing proven XML-based technologies, we can easily achieve to unify interfaces and cover up many different hybrid and proprietary systems. However, there are still certain works to do in developing and applying the Jabber protocol to instant message service. Some issues, such as the guaranteed quality of service and mission-critical message flow support, are critical in the design and implementation of Jabber protocol. Quality of service concerns the problems including:

1) Time to send and receive,
2) Delivery ordering,
3) Delivery priorities, particularly while system load is heavy.

References

[1] M. Day, J. Rosenberg and H. Sugano, "A model for presence and instant messaging," Internet RFC 2778, Feb. 2000.

[2] M. Day, S. Aggarwal, G. Mohr and J. Vincent, "Instant messaging/presence protocol requirements," Internet RFC 2779, Feb. 2000.

[3] Jabber software foundation official web site, http://www.jabber.org

[4] I. Shigeoka, "Instant Messaging in Java, The Jabber Protocols," Manning Publications, 2002.

[5] W3c XML standards, http://www.w3.org/XML/.

[6] http://smsforum.net/doc/public/Spec/SMPP_v3_4_Issue1_2.pdf, October 12, 1999.

[7] ETSI TS 100 900 V7.2.0 (1999-07), Digital cellular telecommunications system (phase 2+), alphabets and language-specific information (GSM 03.38 version 7.2.0 release) 1998.

[8] ETSI TS 100 901 V7.4.0 (1999-12), Digital cellular telecommunications system (phase 2+), technical realization of the short message Service (SMS) (GSM 03.40 version 7.4.0 release) 1998.

[9] The Unicode Standard - http://www.unicode.org.

[10] Java Programming - http://java.sun.com.

[11] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, Tim Berners-Lee – "Hypertext Transfer Protocol -- HTTP/1.1", Internet RFC 2616, June 1999

[12] Tim Berners-Lee, Roy T. Fielding, Larry Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", Internet RFC 2396, August 1998