# Mobile Services
# XMPP Chat supporting SIP

Christoph Kieslich, Hannes Markschläger
Manuel Lindorfer, Michael Schöllhammer

January 2013

## 1 Requirements

The goal of this project was to implement a XMPP chat which also supports the communication to SIP users. In order to provide this functionality there needs to be a conversion from either XMPP to SIP or SIP to XMPP messages. This conversion step is performed by a SIP-XMPP-Server which acts as a gateway between both protocols.

The following aspects have to be taken into account:

- **XMPP to XMPP:** Message exchange between two XMPP users
- **XMPP to SIP:** Message exchange between a XMPP and a SIP client
- **SIP to XMPP:** Message exchange between a SIP and a XMPP client

# 2 XMPP basics

The following chapter gibes an overview over the basics of XMPP, the communication based on XMPP and the addressing scheme.

## 2.1 What is XMPP?

XMPP stands for *Extensible Messaging and Presence Protocol* and is simply a message-oriented communication protocol based on XML. XMPP was developed by the *Jabber open-source community* in the late 90s for the main purpose of near-realtime instant messaging, including both presence information and contact list maintenance. Since it's development XMPP has been used for a lot of different applications, including publish-subscribe systems, signalling for VoIP, smart grids of social networking services, just to give a few examples.

The advantages XMPP offers can be summarized as following:

- **Decentralization:** The architecture of the XMPP network is similar to email; anyone can run their own XMPP server and there is no central master server.
- **Open standards:** The Internet Engineering Task Force has formalized XMPP as an approved instant messaging and presence technology under the name of XMPP. No royalties are required to implement support of these specifications and their development is not tied to a single vendor.
- **History:** Since XMPP technologies have been in use since 1999, multiple implementations of the XMPP standards exist for clients, servers, components, and code libraries.
- **Security:** XMPP servers can be isolated from the public XMPP network (e.g., on a company intranet), and strong security (via SASL and TLS) has been built into the core XMPP specifications.
- **Flexibility:** Custom functionality can be built on top of XMPP; to maintain interoperability, common extensions are managed by the XMPP Standards Foundation.

A drawback of XMPP on the the other side is that it's in-band binary data transfer is quite inefficient. Because XMPP is not yet encoded as Efficient XML Interchange but as a long-lived XML stream, binary data must be first base64 encoded before it can be transmitted in-band. Therefore any significant amount of binary data (eġ; file transfers) is best transmitted out-of-band, using in-band messages to coordinate.

## 2.2 XMPP Transport

The original and *native* transport protocol for XMPP is Transmission Control Protocol (TCP), using open-ended XML streams over long-lived TCP connections. As an alternative to the TCP transport, the XMPP community has also developed an HTTP transport for web clients as well as users behind restricted firewalls. In the original specification, XMPP could use HTTP in two ways: polling[16] and binding.

## 2.3 Decentralization and addressing

The XMPP network uses a client–server architecture (clients do not talk directly to one another). However, it is decentralized—by design, there is no central authoritative server, as there is with services such as AOL Instant Messenger or Windows Live Messenger. Some confusion often arises on this point as there is a public XMPP server being run at jabber.org, to which a large number of users subscribe. However, anyone may run their own XMPP server on their own domain.

Every user on the network has a unique ID. To avoid requiring a central server to maintain a list of IDs, the JID is structured like an email address with a username and a domain name (or IP address) for the server where that user resides, separated by an at sign (@), such as username@example.com.

# 3 Message exchange

The following chapter describes the basic principles of how the communication between different clients is ensured by either the clients or the SIP/XMPP-Server.

## 3.1 XMPP to XMPP

The client is able to send messages not only to SIP users via the gateway but also to normal XMPP users. As of now the client lacks the ability to add new contacts, therefore one has to use another client like *pidgin* to add XMPP users to his contact list. Once you have contacts in your list, those will appear in the XMPP client.

## 3.2 XMPP to SIP

In order to send a message from a XMPP to a SIP client, the message is not sent (as by default) to the twattle.net server, but instead to a interface at the SIP-XMPP-Server. This interface provides a method (see listing below) which simply extracts the neccessary information out of the XMPP message and forwards the message to the SIP destination.

Listing 1: Forwarding of XMPP Messages to SIP

```
/**
 * Forward message to SIP
 *
 * @param msg the msg
 */
private void forwardMessage(Message msg) {
  String to = m_relation_in.get(msg.getTo().split("@")[0]);
  Server.handleXMPPMessage(msg.getFrom(),to,msg.getBody());
}
```

## 3.3 SIP to XMPP

When a SIP client sends a message to a XMPP client, the SIP-XMPP-Server again extracts the neccessary information out of the SIP-message. A new XMPP message is created and forwarded to the corresponding SIP user over the twattle.net server. The listing below shows the information extraction and creation and forwarding of the new message.

Listing 2: Forwarding of XMPP Messages to SIP

```
/**
 * Handle sip message and forwards it to the xmpp user
 *
 * @param from the sip user
 * @param to the xmpp user
 * @param msg the message
 */
public void handleSIPMessage(String from, String to, String msg) {
  String fromUser =m_relation_out.get(from);
  Connection conn = m_connections.get(fromUser);
  try {
    to+="@twattle.net";//append domain
    Message message = new Message(to, Message.Type.chat);
    message.setBody(msg);
    conn.sendPacket(message);//send message to client
    m_logArea.append("message sent to: "+to);
  } catch (Exception e) {
    e.printStackTrace();
    m_logArea.append(e.getMessage());
  }
}
```

## 4 Components and technologies

The following itemization gives a short overview over the components and technologies used in our system.

- **Android XMPP Client**
- **twattle.net Server:** For handling the communication between two XMPP clients
- **XMPP-SIP-Gateway:** For handling the communication between either a XMPP and SIP or a SIP and XMPP client.
- **Used technologies**
  - XMPP as message protocol
  - Java smack library for the XMPP-SIP gateway
  - aSmack (Android) for the Android XMPP client

  i

## 5 Appendix