The Jabber protocol in Cambridge

Tony Finch <fanf2@cam.ac.uk>

University of Cambridge Computing Service

1. Introduction

Jabber is the open standard instant messaging and presence protocol.

Instant messaging (IM) includes the ability to send short messages which pop up on the recipient's screen, one-to-one online chat, and (as an extension) group chat or chat rooms. It fits somewhere between email and the phone in the way it is used.

Presence is an indication of whether you are on-line and how willing you are to talk. In the past a limited form of online presence was available on timesharing systems with commands like ''finger'', but since people have dispersed to workstations this functionality was lost.

These notes aim to provide a technical overview of the Jabber protocol with comments on how we plan to deploy it in Cambridge.

The project originated in October when I was investigating the feasibility of a Jabber service in spare time, inspired by my use of Jabber in collaboration with the IETF, and further motivated by Google Talk. This investigation stalled when I found that I could not do realistic tests without making the project official, so I wrote a proposal at the start of November. The proposal went through two revisions for the Computing Service Senior Management Team and one for the Information Technology Syndicate. Compared to the proposal documents, these notes go into somewhat more detail about the technicalities.

Each section below examines some aspect of the Jabber protocol suite and discusses matters that are relevant to a Cambridge deployment. In some cases I will mention differences between the original core Jabber protocols and the core protocols standardized by the IETF as XMPP; the context for these differences is discussed in more detail in the section on standards.

2. Basic syntax

Jabber is based on XML streams. The protocol data units are ''XML stanzas'', which are complete XML elements within the outermost `<stream/>` element. For example, the following abbreviated stream comprises two stanzas: a `<presence/>` stanza and a `<message/>` stanza.

```
<stream> ... <presence> ... </presence> ... <message> ... </message> ... </stream>
```

Jabber gets a lot of basic functionality for free by using XML:

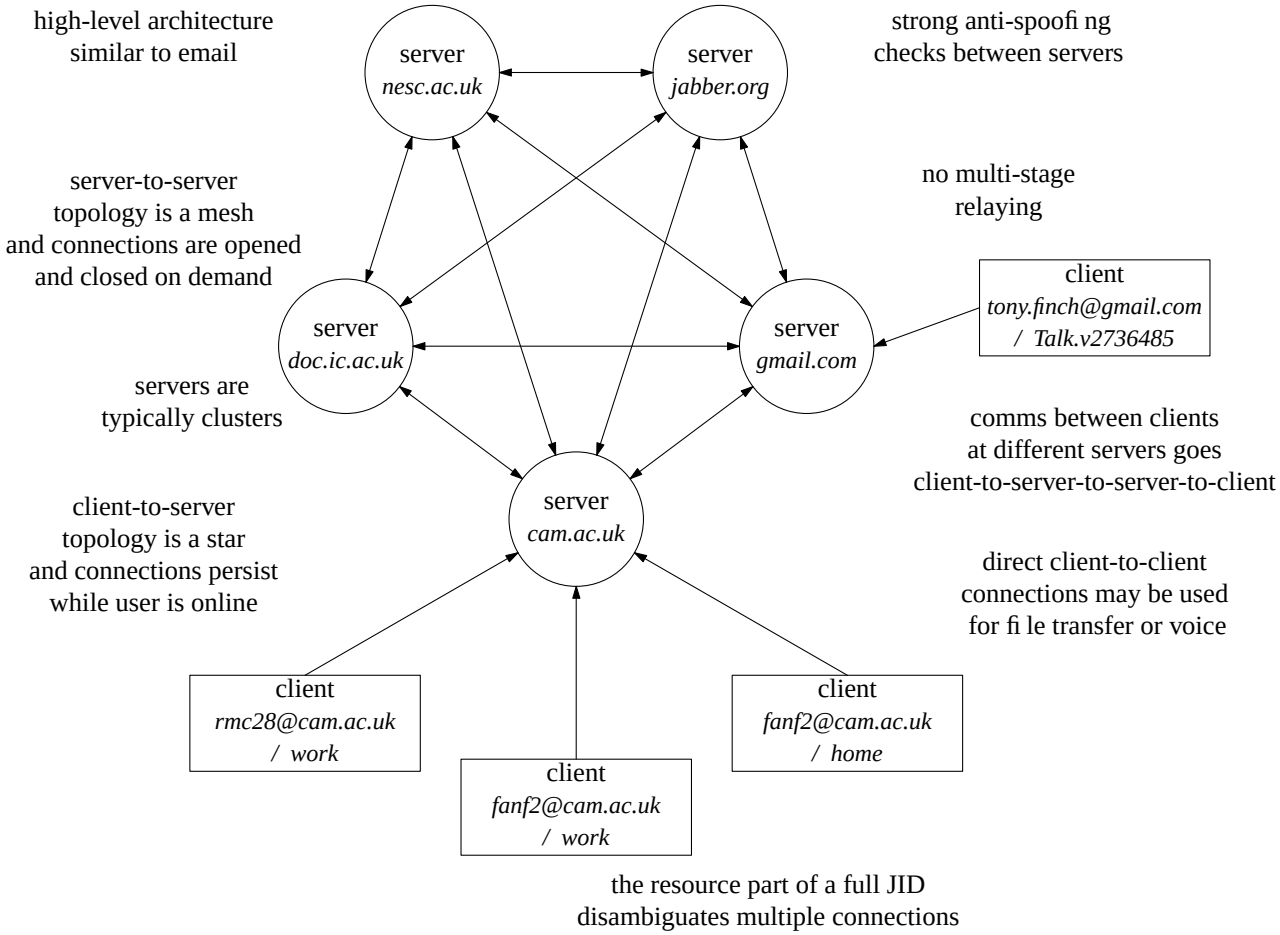
- The syntax of commands and escapes is provided by XML elements and entities.
- XML namespaces prevent extensions from clashing.
- The `xml:lang` attribute specifies the localization of text.
- Easy support for desirable payload types such as XHTML[1] and SOAP[2].

The latter example might be slightly surprising, but although Jabber is designed for IM, its basic protocol is generally useful and can provide better performance than HTTP whilst being considerably simpler than BEEP[3].

3. Topology and addressing

The topology of the global Jabber network is similar to that of Internet email, though somewhat simplified. This is illustrated by the diagram on the following page. Note that inter-domain communication is a fundamental part of Jabber: it is designed to be a global IM system rather than an ''enterprise'' IM system.

These notes were produced for the University of Cambridge Information Technology Syndicate Technical Committee meeting held on the 31st January 2006.



Furthermore, Jabber IDs look like email addresses, in the form ‘‘*username*@*domain*’’. It therefore makes sense for us to set up our Jabber service so that our users’ JIDs are the same as their `@cam` email addresses, to make their online identities as consistent as possible. This is similar to Google Talk’s use of their users’ `@gmail.com` email addresses as JIDs.

In most cases users only deal with these ‘‘bare JIDs’’. There are also ‘‘full JIDs’’ which have an extra ‘‘`/`*resource*’’ part. If a user has multiple concurrent connections, these will have separate resource parts, for example ‘‘`fanf2@cam.ac.uk/work`’’ or ‘‘`fanf2@cam.ac.uk/home`’’. A one-shot message is typically sent to a bare JID, and will be routed to the user’s highest priority connection; whereas an online chat is typically tied to a particular pair of connections by the use of full JIDs. These details are generally handled by client software without exposing much complexity to the user.

JIDs are also used for Jabber entities other than users. For example, a server’s JID is just its domain. Server JIDs may be used for service discovery, which allows any user to browse a server’s facilities, and they may be used for some special server features, for example the server administrator can set the message of the day by sending a message to ‘‘`cam.ac.uk/motd`’’.

3.1. Server components and multi-user chat

Advanced facilities are typically not part of the core server, but are implemented as ‘‘server components’’ that are addressed separately but which depend on the main server for communication with other Jabber entities.

A common example is the multi-user chat facility. (Compare `@lists` in the email world.) The MUC component has a separate domain, for example ‘‘`group.chat.cam.ac.uk`’’, with chat rooms having names like ‘‘*room*`@group.chat .cam.ac.uk`’’. A message to the room’s JID is sent to all its occupants. Each person in a room has a nickname in that room, for example ‘‘*room*`@group.chat.cam.ac.uk/`*nick*’’, which may be used for private messages between users in the room.

The specifications do not require a user's nick to be related to their normal JID, and chat rooms may be ''anonymous'' meaning they do not allow occupants to find out each others' normal JIDs (hence the need for a private message function). We will probably want to restrict facilities like these in our multi-user chat service to minimize the scope for fooling around and causing trouble.

Server components are also used for foreign protocol gateways, which are discussed below.

3.2. DNS matters

In a similar way to email's MX records, Jabber uses the DNS to provide an indirection layer between JIDs and the names of the hosts providing the service. This is based on SRV records, which are effectively a generalized form of MX record that can be used by any protocol. Whereas email only uses MX records for server-to-server connections, Jabber has SRV records for server-to-server and also client-to-server connections; this simplifies client configuration, especially in situations like ours where it is not appropriate to have a server with a hostname of `cam.ac.uk`.

SRV records for c2s were introduced in XMPP and are not yet supported by all clients; older clients will have to be explicitly told the host name `chat.cam.ac.uk`.

We do not yet have any SRV records in the main `cam.ac.uk` zone. The hostmaster team are currently performing some compatibility tests and when these are complete we should be able to deploy the Jabber SRV records. If there is a delay this will affect client autoconfiguration and communication with other Jabber servers.

4.1. Privacy and abuse

It is not possible for a Jabber user to find out anyone's presence information without first ''subscribing'' to it. This is a handshake procedure between the two users in which the target user is sent a subscription request which they may approve or decline. The handshake is typically reciprocated so that in the end there is a bidirectional presence subscription.

Your subscriptions are listed in your ''roster'' (aka ''buddy list''). This is typically the basis of the Jabber user interface: a list of your buddies with indications of their online status.

XMPP added ''privacy lists'' to the basic Jabber roster. Privacy lists can be used to block unwanted communication, including presence indications and subscription requests, as well as instant messages. Users may set up multiple privacy lists which they can subsequently choose between depending on circumstances.

Because this is a new feature it is not yet supported by all software, but I will ensure that we at least have server-side support for it, and it will have some bearing on which client software we choose to support.

We also need appropriate logging in order to deal with complaints. I anticipate that our privacy and abuse policies will be similar to those for email, though we have not yet spent time to finalize them.

4.2. Authentication and Security

Jabber's security features are built on TLS and SASL, the same protocols used in email. (Before XMPP Jabber had its own authentication protocols instead of SASL but there is already good support for the replacement.) We have a number of requirements for authentication in our deployment.

- We should not introduce a new password.
- Any web-based client must use Raven authentication.
- Raven passwords must only be typed into Raven, so a native Jabber client must use some other password.
- Because of the similarity to email it makes sense to use Hermes passwords to authenticate native Jabber clients.

This approach also has the advantage of requiring no additional user administration work. The Chat service can simply use the results of the Hermes user admin scripts.

Some local customization of the Jabber server software will be required to support Raven authentication; however it should be possible to do this in a reasonably modular and maintainable manner because of the extensible framework provided by SASL.

5. Software

Because Jabber is an open standard with a lively community of users, there is a broad variety of software, both open source and commercial, on the server and the client.

I have spent some time evaluating open-source servers in the course of running a Jabber server for my own personal use. The best choice seems to be ejabberd[4]. When the project gets the go-ahead and when the service infrastructure is ready, the first job will be to finalize this decision.

On the client side we intend to provide a web-based client, and to support at least one native client. At the moment Psi[5] is looking like a good candidate, since it works on Linux, Mac OS, and Windows, and the developers are keen to support advanced features like privacy lists and telephony.

During the development phase of the project, the service will be available to early adopters, and we hope to get feedback from them about which clients are preferable. We also have a project wiki[6] where users can directly contribute to documentation, as an alternative to providing suggestions via email.

6. History and Standards

The Jabber open-source project started in early 1999, and by mid-2000 the basic protocols and features were stable, and supported by multiple implementations. In 2001 the Jabber Software Foundation was established to maintain the Jabber specifications through an open standards process. Extensions to the base Jabber protocols are published[7] as ''Jabber Enhancement Proposals'', for example, JEP-45 specifies multi-user chat.

At the same time the IETF ''Instant Messaging and Presence Protocol'' working group was attempting to specify their own IM&P protocol. They were not working from the basis of an already-established protocol, but instead had three competing paper designs, and so failed to reach consensus. However they produced a requirements document and some abstract specifications for gatewaying between different IM&P protocols.

The Jabber community was in favour of getting official IETF blessing for their protocols from the early days, but did not have sufficient momentum to carry this out until later. In 2002, they managed to form the XMPP working group, which was chartered to adapt the Jabber protocols to conform to IETF requirements including those produced by the IMPP working group. The XMPP working group completed its work in 2003, and on the publications of its RFCs in 2004[8,9,10,11] it concluded successfully.

The main improvements between the old Jabber core protocol and XMPP are: better internationalization; better security, through the consistent use of TLS and SASL; and privacy lists.

7. Other IM protocols

Since the early days, Jabber has supported gatewaying to other IM systems – which usually means closed proprietary systems. These gateways are implemented as Jabber server components, named for example `msn.chat.cam.ac.uk`, which act as clients from the point of view of the foreign network. A Jabber user can register with the gateway, after which point their MSN buddies are incorporated into their Jabber roster and they can exchange messages transparently.

Because the foreign networks are closed, this gatewaying is not two-way: MSN users can only communicate with Jabber users who are registered with the gateway, and they cannot join Jabber chat rooms. The other disadvantage is that Jabber users must trust the gateway administrator with their foreign network login credentials.

An alternative is to use a multi-protocol IM client such as Gaim[12] which implements the foreign protocol clients and the roster merging in the same program as the Jabber client.

We intend to provide server-side gateways to the main proprietary IM networks: AIM, MSN, and Yahoo! Note that Google Talk is part of the Jabber network so our users will be able to communicate with Google Talk users with no problems.

8. Telephony

The other protocol worth discussing is SIP, the IETF's voice-over-IP protocol. SIP has a set of IM&P extensions called ''SIMPLE''. Like Jabber, SIMPLE is an open standard; however unlike Jabber it does not have a wide variety of implementations and it has very few users. Furthermore the protocol still lacks many important features such as multi-user chat, file transfer, privacy lists, rich-text messages, etc. and even basic things like one-to-one chat are not yet fully specified. As such, implementations of SIMPLE tend to rely on

proprietary non-interoperable extensions.

In practice, the way that SIP implementations are providing IM features is with a dual-stack approach: using SIP for telephony and Jabber for chat. This is the approach taken by the soft phone programs Gizmo Project[13] and OpenWengo[14].

At the same time, Jabber is being extended with telephony support. The leading example of this is Google Talk. Google's protocol extensions have been published under the name ''Jingle''[15], and they have released an open source implementation which is being incorporated into clients such as Psi.

This is not a promising situation from the interoperability point of view. However, Jingle uses the same streaming media protocols as SIP (i.e. RTP), and the signalling concepts are similar, so it is likely that it will be easy to gateway between Jingle and SIP in the future. There are already multiple open VOIP standards – H.323 and IAX as well as SIP – so another does not make things significantly worse. However there are network-effect advantages to having a single instant messaging network. At the moment it looks like Jabber will be the IM protocol of choice, and VOIP interoperability will rely on gatewaying.

There are also a number of popular closed VOIP applications, notably Skype and Apple iTalk. I believe that iTalk can use Jabber for VOIP signalling as well as IM (although it does not use Jingle) so it fits in well with our plans for the Chat service. In the medium term, we will be able to recommend Jabber clients that support Jingle telephony as an alternative to the problematic Skype.

9. Chat service summary

This section lists the features that we plan to support in the initial version of the University's Jabber service. This list is reasonably modest in order to keep the development time within the target release date of July.

- Core Jabber features, including one-to-one chat, presence, privacy lists, and communication with the public Jabber network.
- Multi-user chat, for subject-related discussion between students and teachers, and to provide ''virtual common rooms'' for departments and colleges.
- Foreign protocol gateways, to make it easier for people to use the Chat service in conjunction with any IM accounts they might already have.
- A web-based client to make it possible to use the service without having to install extra software.
- Documentation and support for the above, plus the most popular native Jabber clients.

This service will be able to support other popular features such as file transfer and telephony, since these typically work direct from client to client rather than requiring special server-side support.

10. Future possibilities

One of the advantages of running our own IM service is that it opens up opportunities for nice integration with other facilities provided by the Computing Service. The following examples are not part of the initial project specification for time reasons, but we have noted them as desirable future projects.

- The Sieve email filtering system on Hermes has a ''notify'' command which provides a hook for instant notification of new email. It would be fairly straightforward to enable Jabber notifications.
- Jabber servers may have a User Directory. We can provide one which provides access to data from the Lookup service.

Alongside email, wikis, discussion fora, and blogs, IM is an important part of a fully-featured virtual learning environment.

References

1. Peter Saint-Andre, *JEP-0071: XHTML-IM.* http://www.jabber.org/jeps/jep-0071.html
2. Fabio Forno and Peter Saint-Andre, *JEP-0072: SOAP Over XMPP.* http://www.jabber.org/jeps/jep-0072.html
3. Marshall T. Rose, "The Blocks Extensible Exchange Protocol Core," RFC 3080 (March 2001). http://www.ietf.org/rfc/rfc3080.txt
4. *ejabberd: A free, distributed, fault-tolerant Jabber server written in Erlang.* http://ejabberd.jabber.ru/
5. *The Psi Jabber client.* http://psi-im.org
6. *The Chat service wiki.* http://wiki.csx.cam.ac.uk/chat/
7. *Jabber Enhancement Proposals.* http://www.jabber.org/jeps/jeplist.shtml
8. Peter Saint-Andre, (ed), "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 3920 (Oct 2004). http://www.ietf.org/rfc/rfc3920.txt
9. Peter Saint-Andre, (ed), "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence," RFC 3921 (Oct 2004). http://www.ietf.org/rfc/rfc3921.txt
10. Peter Saint-Andre, "Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)," RFC 3922 (Oct 2004). http://www.ietf.org/rfc/rfc3922.txt
11. Peter Saint-Andre, "End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)," RFC 3923 (Oct 2004). http://www.ietf.org/rfc/rfc3923.txt
12. *Gaim: A multi-protocol instant messaging (IM) client.* http://gaim.sourceforge.net/
13. *Gizmo Project.* http://www.gizmoproject.com/
14. *OpenWengo.* http://www.openwengo.com/
15. *Jabber Software Foundation Publishes Open VoIP and Multimedia Protocols.* http://www.jabber.org/press/2005-12-15.shtml