A Jabber service for the University of Cambridge SMT/nnn

Tony Finch, fanf2@cam.ac.uk

18th November 2005

Jabber/XMPP 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.

“Presence” is an indication of whether you are on-line and how willing you are to receive instant messages.

XMPP is the name of the core Jabber protocols as standardized by the IETF. (See the last section of this document for information about the history of the Jabber specifications.)

Why the University should have a Jabber service

IM is an increasingly popular Internet application, particularly amongst younger users (which probably includes those at postgraduate level) where it is to some extent displacing email. IM occupies a space somewhere between email and the telephone: it has the textual nature of the former, and the synchronous nature of the latter. However the presence aspect of it is not something we currently have.

If people in the University are going to use IM, a University IM service will help in a number of ways:

- We can provide a stronger link between a user's IM ID and their real-world identity, by making use of the CRS.
- We can provide links between the IM service and our other services, such as the Directory and Hermes.
- We will be able to monitor usage of the system in ways that can help in the event of abuse or harassment.

These are all explained in more detail below, in the “Plan of work” section.

Why Jabber, as opposed to some other IM system? The popular IM services are all closed and centrally controlled, which means we cannot run a service based on the AOL or MSN IM systems. Older forms of open IM have problems that make them unsuitable. The Unix `talk` program would only be appropriate if everyone used CUS all the time. IRC has a weak model of identity and is difficult to operate across multiple domains. Neither of them have good internationalization, or any useful presence features.

The Jabber specifications and development process are open, so there is plenty of software to choose from: multiple server and client implementations, commercial and open source, in multiple programming languages. Anyone can deploy a Jabber server, which may or may not federate with other Jabber servers or the public Jabber network as a whole. As such, it is particularly popular for internal corporate chat services. The most high-profile public Jabber deployment however is probably Google Talk (which does not yet federate with other Jabber servers, though they promise it will). Thus we can be confident that Jabber has plenty of momentum.

Usage scenarios

Obviously the users will determine what Jabber is used for in practice, since it doesn't impose any ideas of its own. However experience in other contexts suggests that some of the following might be popular.

- A supplement to face-to-face supervisions, e.g. to sort out timetabling or for students to ask quick questions.
- Subject-related group chat between students, or a virtual common room for staff.
- As the messaging component of a virtual learning environment, in conjunction with other modern communication tools such as wikis and blogs.
- Using the presence features to indicate your location (e.g. college or department), to avoid phone tag or wasted trips to empty offices.
- Collaboration between physically-separated researchers, within and outside the University.
- A side-band for phone calls for communicating things that are awkward to spell out, such as URLs.
- Moderating telephone conference calls, by using a Jabber group chat to decide who gets to speak next.

Jabber protocol outline

All entities in Jabber have a Jabber ID, including users, chat rooms, servers, and other components. A bare Jabber ID looks like an email address, for example my JID might be `<fanf2@cam.ac.uk>`, and has a similar meaning: the user `fanf2` at the server `cam.ac.uk`. Similarly a chat room JID might be `<ucam-chat@group.chat.cam.ac.uk>`. JIDs can also have a “resource” part: in the case of users, resources are used to distinguish concurrent connections to the server, such as `<fanf2@cam.ac.uk/work>` and `<fanf2@cam.ac.uk/home>`; for multi-user chat, resources are used to identify each user's connection to the chat room, for example `<ucam-chat@group.chat.cam.ac.uk/fanf2>`.

At its lowest level the protocol is based on XML streams, which provide the basis for Jabber's extensibility. Security in the current version of Jabber is based on TLS and SASL, as used in the email protocols. (This implies that Raven authentication cannot be used, since it cannot be implemented as a SASL mechanism.)

The high-level architecture is also similar to email, but much simplified. The domain part of a JID identifies a server (which may of course be clustered). Users talk to servers, with a star topology around each server; and servers talk to each other, with a mesh topology across the Internet. The client-to-server and server-to-server protocols are basically the same but differ in details of authentication and symmetry.

Jabber servers are located using SRV records in the DNS. These are similar to MX records in email, but generalized. This means that the `cam.ac.uk` in a JID doesn't identify the server directly: the actual server may be called `chat.cam.ac.uk`, similar to the email case where the server for email to `cam.ac.uk` is actually `mx.cam.ac.uk`.

As well as routing messages between local users or to and from remote servers, a Jabber server also maintains some per-user state. This includes users' presence status, their “rosters” and privacy lists. Your roster is the list of the other Jabber users whose presence you wish to monitor (a “buddy list” in AIM terminology) and those who are permitted to monitor your presence. Your privacy list allows you to block messages from people.

In order to be able to monitor someone's presence, you must go through a subscription process. When you add someone to your roster, your Jabber client sends a presence subscription request, which the recipient may decline, approve, or reciprocate — presence subscriptions are frequently

bidirectional, but do not have to be. Once your subscription is approved your Jabber client will receive presence notifications from the other user.

Jabber allows you to block individual JIDs, such as <fanf2@cam.ac.uk> for messages directly from me, or <ucam-chat@group.chat.cam.ac.uk/fanf2> for things I say in ucam-chat. You can also organize your roster into groups, and use the groups to define blocking rules. Or you can block according to presence subscription status, including anyone not on your roster; for example you might want to block messages from people who have not gone through a subscription handshake with you. You can separately block messages, presence notifications, subscriptions, and so forth. Furthermore you can keep several privacy lists and use different ones at different times.

Jabber is less vulnerable to spoofing than email, because it has built-in symmetry and mutual authentication in its server-to-server protocol that email lacks. This means that (with any luck) instant messaging spam ("spim") should be somewhat less vexatious — or perhaps easier to block — than email spam. Existing anti-spam technologies (such as blacklists of compromised hosts) can easily be applied to spim when it becomes a problem.

Suggested deployment plan

The name of the service

I propose that our users' JIDs will be the same as their @cam email addresses.

I have used "chat.cam.ac.uk" in the examples above. The reasons for this suggestion include the fact that it is mnemonic, and it provides a nice name for the multi-user chat component, group.chat.cam.ac.uk. (The protocol constrains the MUC component to be named with a subdomain of the server's name.)

It is relatively common for Jabber services to be named like jabber.cam.ac.uk and muc.jabber.cam.ac.uk, but we tend not to name services after the protocols they implement, and this choice makes it harder to come up with a brief non-jargon name for the multi-user chat component.

Another possibility is to expose the fact that it is sharing infrastructure with Hermes, and give it names under hermes.cam.ac.uk. However this makes it harder to separate the services in the future, if we decide to do so, and it makes the names longer.

User administration

The server will need some user-admin support — provisioning and cancelling users, password changes, etc. Rather than duplicating effort and proliferating passwords, it seems sensible to use Hermes's infrastructure. This seems reasonable given that email and IM are both forms of messaging and JIDs and email addresses look the same. This implies that password changes will be done through webmail.hermes.cam.ac.uk, not the usual Jabber mechanism.

In order to deal with the differences between the @cam and @hermes user lists, the Jabber user list will be the intersection of the two — i.e. shared Hermes accounts will be excluded, as will those who have @cam addresses but not Hermes accounts. However it will be blind to email forwarding, such as those with Hermes accounts who redirect their @cam email somewhere other than Hermes.

Provisioning users is relatively simple, since the server will automatically create the necessary state the first time the user logs in. (Many Jabber servers are "open" in that they allow anyone to turn up and register an account; this is unacceptable for our server but means that implementing the

infrastructure is easier!).

This plan should mean that there is no significant extra staff time needed to handle user admin of this new service. However that does not of course include support load, such as explaining to users that they change their Jabber password using Hermes webmail.

Hardware

The Jabber service will be run as a component of Hermes, sharing its support infrastructure including user administration scripts (discussed above), automated installation scripts, the software and configuration management repository, and the backup system. These functions all run on the Hermes admin machine. The additional load imposed by the Jabber service will not be significant, since these are all lightweight tasks, and Jabber does not have very much per-user data to back up.

The service itself needs two additional computers, a live machine and a hot spare. This will be sufficient for the full service including multi-user chat. The software can be distributed across multiple machines if in the future the load on the service requires it.

Plan of work

I already have a prototype `jabberd` setup, with scripts to create the password file as described above, and a simple extension to the `jabberd` software so that it can read password files in the format used on Hermes.

The next stage is to publish records in the DNS so that I can do meaningful client-to-server and server-to-server testing. I will also make the service available to CS staff at this stage.

One area that needs attention is server logging, for statistics gathering and in case of abuse or harassment. This may require an add-on software component. We will also have to decide an appropriate privacy policy.

I plan to do the early testing with self-signed TLS certificates. Before deployment we will have to purchase properly signed certificates. This is one of the areas where the specification and implementations are not in sync, so I will need to do some interoperability testing to make sure we buy the right thing!

We will also need to do some work on deciding which client software we can recommend, and this should be installed and supported on the PWF. We will need to document the service, including how to configure common client software.

At this stage we will have a service that supports basic user-to-user chat.

The feature that is going to attract the most users is multi-user chat. This is implemented as an add-on server component. I will need to become familiar with this software and the MUC specification, and work out a sensible policy for managing the chat-room name space.

Longer term possibilities

Further in the future there are opportunities for integrating Jabber with our other services. For example, the email filtering facility on Hermes has a hook for instant notification of new email. It would be relatively easy to hang something on the hook which sends notifications via Jabber.

There are specifications for extended information about a Jabber user in the form of a vCard, which is a format compatible with most address book applications. A decent implementation would be based on hooking Jabber and the Directory together.

Although the Jabber service will be aimed at users running Jabber clients on their computers and perhaps PWF machines, it may be worth providing a web interface too.

We may wish to provide gatewaying of Jabber messages to email or the University pager service, for when users are not on-line.

Client software

There is a list of implementations at
`http://www.jabber.org/software/clients.shtml`.

I have used Psi, which is open source and multi-platform
(`http://psi-im.org/psi.affinix.com/`).

`iChat` in Mac OS X 10.4 has Jabber support.

GAIM is both multi-platform and multi-protocol (it supports closed IM protocols in addition to Jabber).
(`http://gaim.sourceforge.net/`)

Standards and specifications

Jabber development started in 1998 and took off strongly after the initial announcement in January 1999. In 2001 the Jabber Software Foundation was formed as the venue for developing and refining the Jabber protocols and extensions. The JSF has an open process, similar to the IETF or open source development.

The basic protocols were standardized by the IETF in 2002-2004 under the name XMPP, the extensible messaging and presence protocol. RFC 3920 defines the protocol's core, a framework based on XML “stanzas”. RFC 3921 defines how instant messaging and presence are implemented on top of the core. XMPP is basically the same as the earlier Jabber protocols, but has enhanced security based on TLS and SASL, built-in privacy enhancements, better internationalization, and extensible error handling. Functionality beyond these base documents is defined by the JSF in documents called “Jabber enhancement proposals”, for example, JEP-0045 defines multi-user conferencing.

I should note that there is another IM&P protocol called SIMPLE being developed by the IETF. It is an extension to SIP, and at the moment it is neither finished nor simple (it is suffering from the telco specification development model, somewhat divorced from implementation). At the same time, Jabber is being extended with IP telephony functions, as in Google Talk. It is likely that SIP and Jabber will coexist, and interoperate through gateways.