Jabber, E-mail and Beyond

Ralph Meijer and Peter Saint-Andre

May 2, 2005

Keywords

Application architecture, Data interchange, Distributed Systems, E-mail, Enterprise applications, Finance, Fragment, Internet, Interoperability, Middleware, SOAP, SVG, Unicode, Web Services, XML, XML-RPC, Atom, Messaging, Presence, Publish-Subscribe, Presence, RSS, Streaming, Syndication

1 Introduction

The proliferation of spam, viruses and other malware threatens e-mail, with SMTP [RFC2821] as its core protocol, to come to grinding halt. Many have tried to address these issues, with varying amounts of success, but so far, the outlook is grim. Jabber started as a solution to the problem of having multiple clients for different instant messaging networks running simultaneously. Jabber shares some similarities with e-mail in how they came into existence and how it is architected, although they grew from two totally different starting points.

This paper provides a brief introduction to the concepts of streaming XML and XML-based communications as found in Jabber/XMPP technologies, how it solves problems found in e-mail as it is used today, and how it goes beyond that.

2 Streams and Stanzas

Jeremie Miller thought up and built the jabberd instant messaging and presence server, using a protocol based on *XML Streams* and publicly released it in early 1999. Since then, many more implementations have been written, the core XML streaming protocol has been formalized by the IETF under the name Extensible Messaging and Presence Protocol (XMPP), and Jabber/XMPP technologies have been extended far beyond the realm of instant messaging to encompass everything from network management systems to online gaming networks to financial trading applications.

As defined in RFC 3920 [RFC3920] (the core XMPP specification), an XML stream is a container for the exchange of XML [XML] elements between any two entities over a network. Those who are accustomed to thinking of XML in a document-centric manner may wish to view an XML stream as an open-ended XML document that is built up over time by sending XML elements over the stream, with the root `<stream/>` element serving as the document entity for the stream. However, this perspective is a convenience only; XMPP does not deal in documents but in XML streams and XML stanzas (the first-level child elements that are sent over those streams).

While peer-to-peer implementations of XMPP exist, most implementations follow a client-server model that is familiar from e-mail. When a client connects to a server, it opens a stream to the server and the server opens a stream to the client (resulting in two streams, one in each direction). After negotiation of various stream parameters (including channel encryption and appropriate authentication), each entity is free to send an unbounded number of XML stanzas over the stream. If a client addresses a stanza to a non-local entity, its server will negotiate a server-to-server stream with the foreign domain, the send the stanza over that server-to-server stream for delivery to the non-local entity. As a result, a client can send stanzas to any network-addressable entity.

XMPP defines three core stanza types, each with different semantics:

1. The `<message/>` stanza is a "push" mechanism whereby one entity pushes information to another entity, similar to the communications that occur in a system such as email.

2. The `<presence/>` stanza is a basic broadcast or "publish-subscribe" mechanism, whereby multiple entities receive information about an entity to which they have subscribed (in particular, information about an entity's network availability).

3. The `<iq/>` (Info/Query) stanza is a request-response mechanism, similar in some ways to HTTP, that enables an entity to make a request of and receive a response from another entity.

To date, these core stanza types have proven sufficient for a wide variety of applications. The key is that the stanza types provide the delivery semantics or *transport layer* for near-real-time communications, whereas the content of any given stanza is specified by its child elements, which may be qualified by any XML namespace. Thus the content of an XML stanza is not a MIME type or an attachment but pure XML, and XMPP can be used to exchange any data that can be represented in XML, enabling development of a wide variety of applications.

3 Core Services

As can be guessed from the preceding description of XML streams and XML stanzas, the core functions of an XMPP server are to manage streams and to route stanzas. Most XML streams are negotiated over long-lived TCP [RFC793] connections and XML stanzas are simply first-level child elements sent over the stream.

Managing XML streams involves more than just maintaining TCP connections. RFC 3920 mandates that a server implementation must support Transport Layer Security (TLS) [RFC2246] for channel encryption, Simple Authentication and Security Layer (SASL) [RFC2222] for authentication, DNS SRV records [RFC2782] for port lookups, various profiles of stringprep [RFC3454] for addresses that can contain any Unicode character, UTF-8 [RFC3629] for fully internationalized stream content, and an XMPP-specific protocol for binding a resource to the stream for network addressing purposes. Servers must also stamp XML stanzas with a validated "from" address to prevent address spoofing, enforce various stanza delivery rules, and comply with some restrictions on XML usage in XMPP (such as prohibitions on comments, processing instructions, and DTD subsets).

4 Comparing with E-mail

As a message delivery platform, Jabber shares the basic structure of interconnected servers and client connecting to one server to send and receive messages. In contrast, XMPP uses the same protocol for sending and receiving data. No polling is required, because clients maintain a connection to their server as long as it is running. As e-mail uses the store and forward principle for delivering messages, end-to-end latency is usually measured in minutes, hours. It is also common that messages cross more than two servers between end points. In Jabber, there are usually no intermediate servers and messages are delivered in near-real-time, ie. measured in seconds or less. If a delivery fails, because a server is not reachable, or a client has gone offline, resending is not tried. Instead the originator immediately gets an error message back.

The use of TLS and SASL, between clients and servers, but also between servers, combined with strict address checks, make it hard to abuse Jabber for spam and virus distribution. On top of that, the XMPP-based instant messaging and presence servers provide more core services (as described in RFC 3921 [RFC3921]), among which user-defined block lists and allow lists for communication between other entities on the network.

Besides messaging, Jabber/XMPP brings presence to the table. Sometimes underestimated, presence is of great value in communication. Knowing that your communication partner is available makes sure you know that your messages have a high probability of being read right as you send them. It is what makes messaging feel instant. Also it is a hint to the server where to send messages. Jabber allows multiple, simultaneous client connections to a

server for the same user. For example, someone could connect from his desktop machine *and* his handheld device. Walking from the desktop would make the presence there to become 'away', but as the handheld device is still available, messages are automatically routed there.

5 Key XMPP Extensions

XMPP is an open wire protocol, and because of that, there exists a multitude of implementations of both servers in clients. Commercial or open source, in any imaginable language and for just about any platform, including handheld devices. The use of XML has helped the Jabber/XMPP community develop a large number of extensions to the core protocol defined in RFC 3920. Extensions for basic IM and presence features such as contact lists are specified in RFC 3921. The non-profit Jabber Software Foundation (JSF), which contributed XMPP to the IETF, continues to define more advanced XMPP extensions through its series of Jabber Enhancement Proposals (JEPs). Some of the key extensions include the following documents published in the JEP series:

- JEP-0030 [JEP0030]: Service Discovery – a robust protocol for determining the features supported by other entities on an XMPP network.
- JEP-0115 [JEP0115]: Entity Capabilities – a real-time profile of JEP-0030 [JEP0030] for advertising capability changes via presence.
- JEP-0004 [JEP0004]: Data Forms – a flexible protocol for forms-handling via XMPP, mainly used in workflow applications and for dynamic configuration.
- JEP-0096 [JEP0096]: File Transfer – a protocol for transferring files from one XMPP entity to another based on stream initiation (JEP-0095 [JEP0095]) and several bytestreaming extensions (JEP-0047 [JEP0047], JEP-0065 [JEP0065]).
- JEP-0071 [JEP0071]: XHTML-IM – a W3C-reviewed protocol for exchanging XHTML-formatted messages between XMPP entities.

XML not being a very good container for binary data, Jabber enables end points to negotiate data streams that take place out of band. This could be simple file streams or video streams, in a peer-to-peer fashion, via proxies or from another point in the network, using the best protocols for that purpose. This frees the server of processing large quantities of data (including storage). Combined with service discovery, this also gives users a choice in what data they are receiving (cost, convenience) and tailored to what the currently used client or device can handle (capabilities).

The JEP series also defines XMPP extensions for a wide range of additional features, including XML-RPC [JEP0009] and SOAP [JEP0072] bindings, in-band registration [JEP0077], HTTP binding instead of TCP [JEP0124], Multi-User Chat [JEP0045]) and reliable message delivery [JEP0079].

6 Beyond Messaging

Mailing lists are a good example of a system that uses publish-subscribe as its core design pattern is mailing lists. A mailing list usually revolves around a certain *topic* and has a list of *subscribers*. Every time someone wants *publish* some message, it sends an e-mail to this list. Every subscriber will get a copy of this message (*notification*.

JEP-0060 [JEP0060] builds on this concept and defines a generalized framework for publish-subscribe functionality, that can be used to deploy content syndication, extended presence, and event notification services. A publish-subscribe *service* keeps the list of topics and their subscribers, and sends out notifications on behalf of the publisher. This allows entities to publish information without having the burden of manually distributing it to all interested parties, and at the same time prevents the unsolicited broadcast of information to all contacts.

Extended presence [JEP0119] can gives hints about what a contact is feeling (user mood), doing (user activity), listening to (user tune) and where he is (geolocation [JEP0080]. Although user mood and user tune may seem frivolous, what someone is currently doing and where, combined with availability information (normal presence), can be very valuable in deciding

if and how to communicate with the contact. User activity includes things like on-the-phone, having-a-meeting or in-transit. In business settings this could be used to decide which of a selection of qualified people can be addressed to solve a problem. The location of your people or vehicles (Jabber is also very suitable for communication with non-human entities) has already been proven to be essential.

Content syndication using publish-subscribe is much more efficient than present-day polling of RSS or Atom feeds. Why not directly publish your news items to a publish-subscribe topic, and let your readership be instantly notified? Note that subscribers would not need to be directly subscribed, but via publish-subscribe repeaters, for maximum efficiency gains. Examples of this are Mimír, and PubSub.com.

Of course, every owner of a publish-subscribe topic has full control over who can subscribe and publish to that topic. A subscription may be configurable to control how and when notifications are sent. This also enables content-based subscriptions, besides the usual topic-based subscription model. Also, topics can be grouped into *collections*, so that entities can be notified of events from different topic at once, or of the addition of new topics.

7 The Future

While Jabber/XMPP technologies have been growing in usefulness and popularity since they were first released in January 1999, IETF approval of the core protocols in October 2004 has led to significant new implementations (Sun, Apple), major new deployments (U.S. Government), and renewed activity by open-source projects and commercial software developers alike.

What does the future hold for XMPP? It is always difficult to know how a particular technology will be applied, but several trends seem clear:

- XMPP technologies (especially the pub-sub extension) will be used in more and more applications – whenever presence information, event notification, or real-time content delivery is needed.
- Increasing adoption will lead to convergence on a handful of implementations for particular platforms and fewer protocol extensions over time as more deployments come to depend on XMPP technologies.
- Software tools vendors will make it much easier for developers to write XMPP front ends (e.g., in J2ME and Flash) and server-side services (e.g., in Python and Java).
- Responding to pressure from financial services firms and perhaps a major XMPP-based entrant into the market, the traditional consumer IM services will begin to offer federated access to their systems from XMPP deployments.

Also, we can see e-mail being overtaken by IM. Both because of the popularity of more direct communication, and because of the growing dislike of the problems surrounding the use of e-mail as it exists today. E-mail is not limited to the use of SMTP, POP and IMAP. Instead, it could very well be reimplemented on top of XMPP.

References

[JEP0009] Adams, DJ. JEP-0009: Jabber-RPC. Jabber Software Foundation, December 2002 (Cited in section 5.)

[XML] Bray, Tim, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler and Francois Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). World Wide Web Consortium, February 2004 (Cited in section 2.)

[RFC2246] Dierks, Tim and Christopher Allen. RFC 2246: The TLS Protocol, Version 1.0. Internet Engineering Task Force, January 1999 (Cited in section 3.)

[JEP0004] Eatmon, Ryan, Joe Hildebrand, Jeremie Miller, Thomas Muldowney and Peter Saint-Andre. JEP-0004: Data Forms. Jabber Software Foundation, November 2004 (Cited in section 5.)

[JEP0072] Forno, Fabio and Peter Saint-Andre. JEP-0072: SOAP Over XMPP. Jabber Software Foundation, April 2005 (Cited in section 5.)

[RFC2782] Gulbrandsen, Arnt, Paul Vixie and Levon Esibov. RFC 2782: A DNS RR for specifying the location of services (DNS SRV). Internet Engineering Task Force, February 2000 (Cited in section 3.)

[JEP0080] Hildebrand, Joe and Peter Saint-Andre. JEP-0080: User Geolocation. Jabber Software Foundation, October 2004 (Cited in section 6.)

[JEP0115] Hildebrand, Joe and Peter Saint-Andre. JEP-0115: Entity Capabilities. Jabber Software Foundation, November 2004 (Cited in section 5.)

[JEP0030] Hildebrand, Joe, Peter Millard, Ryan Eatmon and Peter Saint-Andre. JEP-0030: Service Discovery. Jabber Software Foundation, April 2005 (Cited in section 5.)

[RFC3454] Hoffman, Paul and Marc Blanchet. RFC 3454: Preparation of Internationalized Strings ("stringprep"). Internet Engineering Task Force, December 2002 (Cited in section 3.)

[JEP0047] Karneges, Justin. JEP-0047: In-Band Bytestreams. Jabber Software Foundation, December 2003 (Cited in section 5.)

[RFC2821] Klensin, J. RFC 2821: Simple Mail Transfer Protocol. Internet Engineering Task Force, April 2001 (Cited in section 1.)

[JEP0060] Millard, Peter, Peter Saint-Andre and Ralph Meijer. JEP-0060: Publish-Subscribe. Jabber Software Foundation, March 2005 (Cited in section 6.)

[JEP0079] Miller, Matthew and Peter Saint-Andre. JEP-0079: Advanced Message Processing. Jabber Software Foundation, October 2004 (Cited in section 5.)

[JEP0095] Muldowney, Thomas, Matthew Miller and Ryan Eatmon. JEP-0095: Stream Initiation. Jabber Software Foundation, April 2004 (Cited in section 5.)

[JEP0096] Muldowney, Thomas, Matthew Miller and Ryan Eatmon. JEP-0096: File Transfer. Jabber Software Foundation, April 2004 (Cited in section 5.)

[RFC2222] Myers, John. RFC 2222: Simple Authentication and Security Layer (SASL). Internet Engineering Task Force, October 1997 (Cited in section 3.)

[RFC793] Postel, Jon. RFC 793: Transmission Control Protocol (TCP). Internet Engineering Task Force, September 1981 (Cited in section 3.)

[JEP0077] Saint-Andre, Peter. JEP-0077: In-Band Registration. Jabber Software Foundation, August 2004 (Cited in section 5.)

[JEP0071] Saint-Andre, Peter. JEP-0071: XHTML-IM. Jabber Software Foundation, September 2004 (Cited in section 5.)

[RFC3920] Saint-Andre, Peter. RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core. Internet Engineering Task Force, October 2004 (Cited in section 2.)

[RFC3921] Saint-Andre, Peter. RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. Internet Engineering Task Force, October 2004 (Cited in section 4.)

[JEP0119] Saint-Andre, Peter. JEP-0119: Extended Presence Protocol Suite. Jabber Software Foundation, March 2005 (Cited in section 6.)

[JEP0045] Saint-Andre, Peter. JEP-0045: Multi-User Chat. Jabber Software Foundation, April 2005 (Cited in section 5.)

[JEP0065] Smith, Dave, Matthew Miller and Peter Saint-Andre. JEP-0065: SOCKS5 Bytestreams. Jabber Software Foundation, November 2004 (Cited in section 5.)

[JEP0124] Smith, Dave, Peter Saint-Andre and Ian Paterson. JEP-0124: HTTP Binding. Jabber Software Foundation, March 2005 (Cited in section 5.)

[RFC3629] Yergeau, Francois. RFC 3629: UTF-8, a transformation format of ISO 10646. Internet Engineering Task Force, November 2003 (Cited in section 3.)