Streaming XML with Jabber/XMPP

Peter Saint-Andre and Ralph Meijer

May 1, 2005

Keywords

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

1 Introduction

The phrase "streaming XML" may strike longtime XML and SGML afficionados as an oxymoron. Isn't XML all about stable (if extensible) containers for documents and data?

It was, until Jeremie Miller developed the idea of XML streams in 1998 and built a working implementation in the jabberd instant messaging and presence server that he released 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.

This paper provides a brief introduction to the concepts of streaming XML and XML-based communications as found in Jabber/XMPP technologies.

2 Streams and Stanzas

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 email. 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-addressible 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. Since most XML streams are negotiated over long-lived TCP [RFC793] connections and XML stanzas are simply first-level child elements sent over the stream, it could be said that a minimal XMPP implementation needs to do only two things: (1) maintain TCP connections and (2) incrementally parse XML. In practice, however, while it's easy to write a basic XMPP implementation, it is much harder to write a really good implementation. The reasons will become clear as we look at the services expected of an XMPP server or client.

Managing XML streams involves more than just maintaining TCP connections. For instance, 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 (no spam from bill@microsoft.com over XMPP!), 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).

XMPP-based instant messaging and presence servers provide even more core services (as described in RFC 3921 [RFC3921]), including IM session management, storage of contact lists, implementation of a rather complicated state machine related to presence subscriptions (i.e., control over access to a user's availability information), and enforcement of user-defined block lists and allow lists for communication with other entities on the network.

That may sound complex, and it is. Thankfully, a guiding principle of the Jabber philosophy since 1999 has been to keep most of the complexity at the server. This has made it much easier to write clients, bots, components, and other entities that connect to servers. One result has been that there is a veritable plethora of Jabber clients. Another has been that libraries for writing clients and client-like entities (such as bots) exist for just about every major programming language (and some minor ones, too). When it comes to experimenting with XMPP and building Jabber applications, developers have a great deal of options.

4 XMPP for Instant Messaging and Presence

The first and still dominant application of XMPP is instant messaging and presence. The IM and presence implementations, best known under the name "Jabber", are often called "the Linux of instant messaging" because they provide a fully open alternative to closed, proprietary IM services such as ICQ, AIM, MSN Messenger, and Yahoo! Instant Messenger. Since the first Jabber code was released in January 1999, millions of end users have downloaded one of the many Jabber clients, hundreds of thousands of system administrators have installed Jabber servers, and thousands of developers have contributed code or written custom Jabber extensions. The result is a thriving open network with tens of thousands of servers, and thousands more private deployments at companies,

universities, non-profit organizations, and government agencies worldwide.

Because XMPP is an open wire protocol (recently standardized by the IETF) rather than a single codebase, multiple implementations and licensing schemes are encouraged. So far there are half a dozen open-source server implementations in C, Java, Python, and Erlang, as well as closed-source implementations produced by software vendors such as Jabber Inc., Antepo, Winfessor, and Sun Microsystems. There are open-source, freeware, shareware, and commercial clients for common (Windows, MacOS, Linux) and not-so-common (Amiga) personal computing operating systems, handheld devices running PalmOS and WinCE, and cellphone platforms like Symbian and J2ME. There are Jabber code libraries for C, C++, C#, COM, Delphi, Erlang, Flash, Java, JavaScript, Mono, Objective-C, Perl, PHP, Python, Ruby, Tcl, and more. Most server implementations are quite modular, so it is relatively easy to write server-side components for custom functionality.

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). Popular IM and presence extensions include multi-user chat (a la IRC), file transfer, XHTML-formatted messages, and "extended presence" functionality such as publishing one's avatar or current song.

Some of these extensions may seem frivolous, and IM has the reputation for being a medium that's appropriate for nothing more weighty than teen chat. Yet it would be a mistake to downplay the impact that presence-enabled, near-real-time messaging can have in an organization. Here are some examples from existing XMPP deployments:

- Many of the world's leading financial services firms consider their XMPP-based messaging systems to be more mission-critical than their phone systems.
- Jabber technologies provide a crucial presence and communications infrastructure for over 65 emergency management agencies in the Washington, D.C., area.
- XMPP is the basis for presence-driven services such as push-to-talk and push-to-video at several large mobile telephony providers.
- A large healthcare provider uses Jabber IM systems to enable rapid communication between patients, nurses, and medical experts.

As we can see, IM is not just for teens anymore.

5 XMPP Beyond IM

But XMPP is much more than IM. The semantics of the `<message/>`, `<presence/>`, and `<iq/>` stanzas provide a generalized communication layer, making it possible to develop and deploy a wide range of presence-enabled applications. Consider a few examples:

- A partnership among some of the world's largest foreign exchange trading banks is transitioning its $100-billion-a-day spot dealing system to an XMPP foundation for improved extensibility and integration with back-end systems.
- A content syndication company provides a free service that enables near-real-time delivery of information from millions of RSS and Atom feeds, using the XMPP publish-subscribe extension to solve the problem of continuous polling for feed updates.
- Using standard XMPP protocols and several custom extensions, a network management company ships software that enables a system administrator to monitor the availability, uptime, and other vital statistics of routers and other deployed infrastructure.
- A university research project uses XMPP and the Jabber-RPC protocol extension to simulate catastrophic network failures on the Internet.
- A telematics company offers an integrated hardware/software solution that uses XMPP presence to track fleet vehicles

and thus reduce losses due to theft and vandalism.

- A commercial software company uses the SOAP XMPP binding both to provide a web services interface to native XMPP services and to enable interaction between XMPP entities and HTTP-accessible web services.
- An open-source project sponsored by a major technical institute is building a distributed whiteboarding system that passes snippets of SVG over XMPP in order to synchronize multiple displays.

6 Key XMPP Extensions

Core XMPP functionality is specified in RFCs 3920 and 3921, published by the Internet Engineering Task Force. But the Jabber/XMPP community continues to innovate in the realm of XMPP protocol extensions. Some of the key extensions include the following documents published in the Jabber Software Foundation's 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-0045 [JEP0045]: Multi-User Chat – a set of protocols for participating in and administering multi-user chat rooms, similar to Internet Relay Chat but with stronger security.
- 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.
- JEP-0124 [JEP0124]: HTTP Binding – a binding of XMPP to HTTP rather than TCP, mainly used for devices that cannot maintain persistent TCP connections to a server.
- JEP-0060 [JEP0060]: Publish-Subscribe – a generalized framework for publish-subscribe functionality, mainly used to deploy content syndication, extended presence, and event notification services.

The JEP series also defines XMPP extensions for a wide range of additional features, including XML-RPC and SOAP bindings (JEP-0009 [JEP0009] and JEP-0072 [JEP0072]), in-band registration (JEP-0077 [JEP0077]), extended presence (JEP-0119 [JEP0119]), geolocation (JEP-0080 [JEP0080]), and reliable message delivery (JEP-0079 [JEP0079]).

7 The Future of XMPP

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.

No one claims that XMPP technologies will achieve world domination next Thursday, or ever. Different technologies are appropriate for different requirements, and most technologies eventually become so stale that someone decides to design a replacement. But as life becomes faster and faster, more developers, organizations, and service providers decide to deploy XMPP technologies because they provide near-real-time delivery of structured data and presence information. And that's reason enough to think that streaming XML was not such a bad idea after all.

References

[JEP0009] Adams, DJ. JEP-0009: Jabber-RPC. Jabber Software Foundation, December 2002 (Cited in section 6.)

[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 6.)

[JEP0072] Forno, Fabio and Peter Saint-Andre. JEP-0072: SOAP Over XMPP. Jabber Software Foundation, April 2005 (Cited in section 6.)

[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 6.)

[JEP0030] Hildebrand, Joe, Peter Millard, Ryan Eatmon and Peter Saint-Andre. JEP-0030: Service Discovery. Jabber Software Foundation, April 2005 (Cited in section 6.)

[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 6.)

[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 6.)

[JEP0095] Muldowney, Thomas, Matthew Miller and Ryan Eatmon. JEP-0095: Stream Initiation. Jabber Software Foundation, April 2004 (Cited in section 6.)

[JEP0096] Muldowney, Thomas, Matthew Miller and Ryan Eatmon. JEP-0096: File Transfer. Jabber Software Foundation, April 2004 (Cited in section 6.)

[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 6.)

[JEP0071] Saint-Andre, Peter. JEP-0071: XHTML-IM. Jabber Software Foundation, September 2004 (Cited in section 6.)

[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 3.)

[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 6.)

[JEP0065] Smith, Dave, Matthew Miller and Peter Saint-Andre. JEP-0065: SOCKS5 Bytestreams. Jabber Software Foundation, November 2004 (Cited in section 6.)

[JEP0124] Smith, Dave, Peter Saint-Andre and Ian Paterson. JEP-0124: HTTP Binding. Jabber Software Foundation, March 2005 (Cited in section 6.)

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