

Jabber: Mehr als Instant Messaging

von Jochen Topf

Einleitung

Die Welt des Instant Messaging wird dominiert von einige großen Firmen mit ihren proprietären Chat-Systemen: Microsofts MSN, AOLs AIM, ICQ und ein paar andere. Will man jemanden kontaktieren, so muss man nicht nur wissen, welches Netz er benutzt und welche ID er in diesem Netz hat, man muss auch selber eine ID in diesem Netz besitzen. Verbindungen zwischen den Netzen gibt es nicht. Die Situation gleicht der bei E-Mail vor 15 Jahren, als es verschiedene E-Mail-Netze gab, die verschiedene Protokolle und Adressformate benutzten. Eine E-Mail von CompuServe zu einem Fido-Net-Node? Geht nicht. Oder nur mit Tricks über irgendwelche Gateways. Dass es beim Instant Messaging immernoch so aussieht, ist eigentlich ein Anachronismus.

Bereits im Jahre 1998 entwickelte Jeremie Miller das Jabber-Protokoll, das Abhilfe schaffen sollte. Ein offenes, auf XML-Standards basierendes Protokoll für den Austausch von Instant Messaging-Nachrichten. Anders als die proprietären Protokolle setzt es nicht auf eine einzige Zentrale, über die alle Nachrichten gehen, sondern ist verteilt organisiert: Jabber-Clients reden mit „ihrem“ Server, der wiederum mit anderen Servern Verbindung aufnimmt, um die Clients hinter diesen Servern zu erreichen. Jeder kann nicht nur einen eigenen Client haben, sondern auch einen eigenen Server betreiben. Das Netz sieht damit grundsätzlich recht ähnlich aus, wie das Netz der E-Mail-Server. Im Jahre 2004 wurde das Jabber-Protokoll dann von der IETF unter dem Namen XMPP (*Extensible Messaging and Presence Protocol*) standardisiert. In den RFCs 3920 und 3921 sind die grundlegenden Funktionen festgelegt. Seither existieren die Namen Jabber und XMPP nebeneinander. Jabber wird dabei eher für das IM-Netzwerk allgemein verwendet und XMPP bezeichnet die Protokollfamilie.

Die Jabber ID

Basis aller Kommunikation zwischen Endpunkten im XMPP-Netz sind sogenannte Jabber IDs (*JID*). Jeder Client, jeder Server, jede Subkomponente, die einzeln angesprochen werden soll, hat so eine Adresse.

Jabber IDs sehen auf den ersten Blick aus wie E-Mail-Adressen: *jochen@topf.org* zum Beispiel ist meine Jabber ID. In diesem Fall ist *topf.org* die Adresse des Jabber-Servers, der unter dieser Adresse auch direkt angesprochen werden kann. Und *jochen@topf.org* ist eben die Adresse einer Person.

Nimmt man mit mir Kontakt auf, so wird man feststellen, dass mein Client vielleicht die Subadresse *jochen@topf.org/Home* hat. Das */Home* kann im Client eingestellt werden und dient dazu mehrere Clients des gleichen Users (zum Beispiel einen zu Hause und einen bei der Arbeit) zu unterscheiden. Spricht man die Hauptadresse *jochen@topf.org* an, so wird die Nachricht automatisch an den Client ausgeliefert, der die höchste (vom User einstellbare) Priorität hat.

Die Jabber ID wird aber noch für viel mehr als nur für User, Clients und Server verwendet. Einzelne Komponenten im Server können auch Adressen haben. Zum Beispiel gibt es im Server *jabber.org* eine Konferenz-Komponente unter dem Namen *conference.jabber.org*, die für Multiuser-Chats verwendet wird. Und jeder Konferenzraum darunter hat wiederum eine Adresse der Form *raum@conference.jabber.org*. Andere Komponenten sind beispielsweise Gateways zu anderen Netzen oder User-Datenbanken.

Das Jabber-Netzwerk

Alle Jabber-Clients und Server bilden zusammen ein Netzwerk, das sich jeweils passend zusammensetzt. Jeder Client ist dabei nur mit seinem eigenen Server verbunden. Die Adresse des Clients ist daher auch immer von der des Servers abgeleitet. Ein Server kann beliebig viele Clients haben und es gibt auch Server-Software, die sehr viele Clients unterstützt. Auf dem öffentlichen *jabber.org*-Server beispielsweise sind mehr als 200.000 User angemeldet, von denen auch schonmal 10.000 gleichzeitig online sind.

Aufgabe der Server ist es, Nachrichten zwischen den Clients weiterzuleiten. Hängen zwei Clients am gleichen Server, so ist das einfach. Sind die Clients an verschiedenen Servern, so bauen die Server untereinander Verbindungen auf. Dabei reden immer die Server direkt miteinander, deren Clients an einem Gespräch beteiligt sind. Eine Weiterleitung über mehr als zwei Server hinweg, wie zum Beispiel in IRC-Netzwerken, gibt es nicht.

Die Server finden sich über das DNS, da jede Jabber ID immer auch den Servernamen enthält. Dazu werden entweder A-Records verwendet oder spezielle SRV-Records. Ist beispielsweise auf dem Rechner *jabber.example.org* ein Jabber-Server installiert und soll er für die Domain *example.org* zuständig sein und Adressen direkt unter *example.org* verwenden, dann wird ein SRV-Record eingerichtet, der von *example.org* auf *jabber.example.org* zeigt. Es funktioniert also ähnlich wie bei den MX-Records für E-Mail.

Damit man nicht so einfach wie im E-Mail-System Adressen fälschen kann, benutzen XMPP-Server Signaturen oder Callbacks. Bei einem Callback baut dabei ein angesprochener Server rückwärts eine TCP-Verbindung zu dem Server auf, der ihn angesprochen hat. Dabei benutzt er das DNS zur Namensauflösung und kann sich daher sicher sein, dass er den echten Besitzer der Domain erreicht.

Spezielle Dienste, wie Mehrpersonen-Chats und Gateways in andere Netze sind auch in den Servern implementiert, oftmals in sogenannte

Komponenten (*components*) ausgelagert. Clients können daher relativ einfach gebaut sein, sie müssen nur die einfachen Mechanismen unterstützen, alles kompliziertere wird in die Server und Komponenten verlagert.

Das Protokoll

XMPP basiert auf XML. Jede Nachricht zwischen Client und Server oder zwischen zwei Servern ist in ein XML-Fragment verpackt, das neben der Nachricht selber Meta-Informationen, wie Absender und Empfänger, eine Nachrichten-ID und ggf. auch Fehlermeldungen usw. enthält.

Bei XMPP wird zwischen drei Nachrichten-Typen unterschieden: Die eigentlichen Instant Messaging-Nachrichten (*message*), Nachrichten zur Status von Benutzern (*presence*), mit denen man angeben kann, ob man gerade online oder offline ist, bzw. bereit zum Chat oder „away“, und als dritter Typ die Service-Discovery-Nachrichten (*iq*), mit denen man erfahren kann, welche Dienste ein Server oder Client zur Verfügung stellt.

Als Beispiel sei hier eine normale Nachricht gezeigt:

```
<message from="oskar@jabber.org/Psi" to="sabine@jabber.org/Home"
  type="chat" id="aab8a">
  <body>Lust auf Kino heute abend?</body>
</message>
```

Hier schreibt also Oskar (der wohl den Client namens PSI verwendet) an Sabine (die wohl zu Hause ist). Die ID dient zur Zuordnung von Antworten (z.B. Fehlermeldungen), der Typ der Nachricht ist hier „chat“, es geht also um eine laufende Unterhaltung.

Wie man sieht ist der Inhalt der Nachricht ein einfacher Text, stattdessen ist es aber auch möglich hier HTML-Nachrichten zu versenden oder beliebige Andere zusätzliche Daten. Das XML-Format erlaubt es weitere Daten zu ergänzen, die der Server einfach weiterreicht. Sind die beiden Clients dafür ausgelegt, so können sie diese Informationen anzeigen oder sonstwie auswerten. Apple iChat, das auch XMPP benutzt, kann z.B. die gerade abgespielte Musik in Presence-Nachrichten mitliefern, so können die Chat-Partner sehen, was man sich gerade anhört.

Presence-Nachrichten werden bei Status-Änderungen automatisch an alle Leute verschickt, die man in seinem *Roster*, also der Liste von Bekannten hat. (Also das, was andere Netze *buddy list* nennen.) Erst wenn man jemanden in seinem Roster hat, kann er auch sehen, ob man gerade online ist. Spricht man jemanden zum ersten Mal an, so wird

typischerweise auch gleich eine Nachricht an ihn geschickt, die ihn fragt, ob er einen in seinen Roster aufnimmt.

Service Discovery

Ein ganz wichtiges Feature für die Erweiterbarkeit von XMPP ist die sogenannte Service Discovery. Jede Jabber ID kann eine andere JID befragen, welche Dienste sie zur Verfügung stellt und welche Sub-Dienste sie anbietet. Dadurch können sich Clients, Server und andere Komponenten aufeinander einstellen und ihre eigene Funktion anpassen. Benutzt wird dieser flexible Mechanismus für viele sehr verschiedene Dinge:

Will man sich beispielsweise bei einem Jabber-Server registrieren (also eine JID dort bekommen), so fragt der Client erstmal an, ob der Server die Registrierung unterstützt. Erlaubt der Server eine Registrierung, so sagt er dem Client, welche Informationen (wie Name, Passwort) er benötigt. Der Client baut ein passendes Formular zusammen und präsentiert es dem User. Der ganze Vorgang ist also so generisch, dass man problemlos weitere Formularfelder (wie z.B. eine Telefonnummer oder dergl.) abfragen kann, wenn das in diesem speziellen Fall sinnvoll ist.

Den gleichen Service Discovery-Mechanismus benutzt man, wenn man wissen will, welche Multi-User-Chats (*conferences*) ein Server zur Verfügung stellt. In der Regel fragt man den Server, welche Sub-Dienste er anbietet und einer davon ist dann der Konferenz-Dienst mit seiner eigenen Jabber ID. Den nun fragt man nach den dort vorhandenen Konferenzen und er liefert eine entsprechende Liste zurück.

Auch Clients kann man nach ihren Fähigkeiten befragen. Das sind typischerweise unterstützte Erweiterungen wie Voice-Chat oder Filetransfer. Der eigene Client kann sich dann nach den Fähigkeiten der Chat-Partner einrichten und entsprechenden Menu-Funktionen oder Buttons zur Versenden von Files oder dergleichen bereit stellen.

Clients

Um am Jabber-Betrieb teilzunehmen, braucht man einen Client. Davon gibt es inzwischen eine ganze Menge und auch die meisten Multiprotokoll-Clients unterstützen inzwischen XMPP. Apple liefert mit iChat einen entsprechenden Client mit, für Linux und Windows gibt es jede Menge Open Source-Clients. Einer der bekannteren ist „gaim“, ein Multiprotokoll-Client, der sich aus einem Client für AOLs AIM-Netz entwickelt hat. Da Google Talk auch das XMPP-Protokoll benutzt, kann man sich auch dort anmelden und wird damit Teil der Jabber-Welt.

Will man etwas tiefer in die Jabber-Welt einsteigen und etwas mehr als nur die üblichen Zwei- oder Mehrpersonen-Chats, so sollte man sich den Client „PSI“ anschauen (<http://psi-im.org/>). PSI gibt es für Linux, MacOS X und Windows. Es unterstützt nur XMPP, dies aber besser als die meisten anderen Clients: Insbesondere bietet es einen Service Discovery Browser und eine XML-Console über die man sehen kann, wie die Roh-Nachrichten aussehen, die man empfängt und versendet. Die Console erlaubt es auch, beliebige XML-Nachrichten zu versenden; eine unverzichtbare Funktion zum Debugging und um die Innereien von XMPP besser kennenzulernen.

Will man selber aktiv werden und einen Client schreiben, dann kann man auf eine Reihe von Libraries zurückgreifen. Für praktisch alle Sprachen gibt es ausgereifte XMPP-Libraries, die einem die Handarbeit abnehmen. Die grundlegenden Protokoll-Funktionen, wie Verbindungsaufbau und den Versand von Nachrichten unterstützen sie alle. Bei komplexeren Funktionen wie der Service Discovery ist die Unterstützung verschieden gut, was vor allem an der Erweiterbarkeit von XMPP liegt, die es schwierig macht immer alle neuen Funktionen zu unterstützen. Notfalls kann man aber immer auf die XML-Ebene herunter und die speziellen Nachrichten-Formen selbst implementieren.

Server und Komponenten

Gleichzeitig mit der Entwicklung des Jabber-Protokolles wurde auch der Open Source Jabber-Server *jabberd* entwickelt, der heute in mehreren verschiedenen Versionen vorliegt. Er war viele Jahre der am meisten verbreitete Server. In letzter Zeit scheint aber der neu entwickelte *ejabberd* beliebter zu werden. Er ist in der Programmiersprache Erlang entwickelt, die Funktionen zur Skalierbarkeit und Hochverfügbarkeit enthält. Daneben gibt es den in Perl geschriebenen *djabberd* und den Java-Server *Wildfire*. Auch diese Server sind als Open Source-Software erhältlich.

Die Einrichtung der Server ist nicht immer ganz einfach, aber wenn man ein bisschen Erfahrung in solchen Dingen hat, auch nicht besonders schwierig, ein bisschen Wissen über Jabber und seine Begriffe vorausgesetzt.

Viele Server kommen von sich aus schon mit weiteren Komponenten wie Gateways in andere IM-Netze, Userdatenbank und Multi-User-Konferenzen. Weitere Komponenten kann man dazu installieren. Die Komponenten sind dabei teilweise Server-spezifisch, viele liegen aber als externe Software vor, die mit jedem Serverzusammenarbeitet, wenn er entsprechend konfiguriert ist.

Falls gewünscht werden die Komponenten vom Server in eine öffentliche Liste seiner Sub-Funktionen aufgenommen und können dadurch von Clients per Service Discovery entdeckt werden.

Gateways

Leider benutzen noch lange nicht alle IM-Dienste das XMPP-Protokoll. Aber es gibt Gateways zwischen den Systemen. Typischerweise sind diese als Server-Komponenten im XMPP-Server implementiert.

Da die proprietären Protokolle im allgemeinen keine Gateways unterstützen, gibt das Gateway zu dieser Seite hin vor, ein Client zu sein. Das heisst aber, dass man auch eine entsprechende Adresse im jeweiligen Netz haben muss. Diese Adresse zusammen mit dem Passwort muss das Gateway kennen, damit es sich dort „ganz normal“ als Client anmelden kann. Die Adressen von anderen Usern im proprietären Netz setzt es als Jabber-Adressen um, aus der MSN-Adresse *user@hotmail.de* kann dann z.B. *user%hotmail.de@gateway.example.org* werden, wenn das Gateway auf dem Jabber-Server *example.org* läuft. Auch hier wird wieder Service Discovery verwendet, um die Funktion so transparent wie möglich zu machen.

Da die Besitzer der proprietären Netze solche Gateways nicht gerne sehen, gibt es kaum große, allgemein nutzbare Gateways. Stattdessen haben sich viele Betreiber von XMPP-Servern ihre eigenen Gateways eingerichtet, die jeweils nur von einer kleinen Gruppe von Leuten benutzt werden. Für alle großen proprietären IM-Dienste wie AIM, Yahoo, ICQ und MSN, sowie für IRC sind Gateways als Open Source Software verfügbar. Und es gibt auch ein (allerdings natürlich kostenpflichtiges) SMS-Gateway.

Protokoll-Erweiterungen

Inzwischen gibt es über 200 XMPP Extensions (siehe *www.xmpp.org*), die die beiden grundlegenden RFCs ergänzen. Und es wird ständig an neuen Erweiterungen gearbeitet. Die Extensions umfassen einen bunten Strauss von Erweiterungen vom Filetransfer über Verschlüsselung und Voice-Chat bis zum Speichern von XML-Daten auf dem Server.

Natürlich kann sich auch jeder seine eigene Erweiterung basteln, XMPP ist darauf ausgelegt. Dazu ergänzt man die XML-Nachrichten einfach um weitere Komponenten in eigenen XML-Namespaces. Die kommen dem Kern der Nachricht dann nicht in die Quere und werden vom Server einfach nur weitergeleitet. Nur der spezielle Client auf der anderen Seite wertet die Informationen dann aus. Braucht man eigene Funktio-

nen im Server, so kann man diese in der Regel in eine Komponente einbauen, die in den Server eingebunden wird.

Im folgenden will ich, ohne Anspruch auf Vollständigkeit, einige interessante Erweiterungen herausheben:

Jingle

Die ursprünglich von Google für Google Talk entwickelte Voice-Erweiterung *Jingle* erlaubt es in Echtzeit Multimediadaten, also vor allem Sprache und Video auszutauschen. Das IM-Protokoll XMPP wird damit zum Voice- und Video-Chat. Unterstützt wird *Jingle* von Googles eigenem Client und es gibt auch erste Open Source-Clients. Seit der vor kurzem veröffentlichten Version 1.4 unterstützt auch die Telefonanlagen-Software Asterisk *Jingle* und stellt damit die Verbindung in die VoIP-Welt bzw. zur klassischen Telefonie her.

Filetransfer

Praktisch alle Clients unterstützen auch den Filetransfer. Dabei erfolgt nur die Verbindungsaufnahmen und das Aushandeln der Parameter über XMPP, die Files selber werden direkt von Client zu Client übertragen. Sind beide Clients hinter einer Firewall und damit kein direkter Kontakt möglich, so kann automatisch ein SOCKS-Proxy dazwischengeschaltet werden. Viele Jabber-Server bieten so einen Proxy an.

vCards

Zu jeder Jabber ID kann auf dem Server eine sogenannte vCard hinterlegt werden. Das ist eine Art Visitenkarte in einem standardisierten Format, wie sie auch manchmal per E-Mail verschickt oder in Handys verwendet wird. Per vCard kann ein User Informationen über sich im Server ablegen, damit andere User ihn leichter finden können bzw. weitere Kontaktmöglichkeiten oder z.B. ein Bild von ihm erhalten können. Welche Informationen man in seine vCard einträgt bleibt einem natürlich selbst überlassen.

Verschlüsselung

Die Verbindung zwischen Client und Server und zwischen den Servern ist bei XMPP typischerweise per TLS verschlüsselt. Durch diese Hop-to-hop-Verschlüsselung ist es einem Serverbetreiber aber theoretisch möglich, mitzuhören. Dagegen gibt es verschiedene Ansätze für eine Ende-zu-Ende-Verschlüsselung, z.B. per PGP. Derzeit hat sich aber noch

kein Standard durchgesetzt.

Publish-Subscribe

Durch die Verwendung des erweiterbaren XML-basierten Protokolles eignet sich XMPP auch gut für die Weitergabe strukturierter Daten, z.B. denen eines Börsentickers oder von RSS-Feeds. Die Publish-Subscribe-Erweiterung erlaubt es einem Client dem Server mitzuteilen, an welchen Nachrichten er interessiert ist. Schickt ein Client eine Nachricht an den Publish-Subscribe-Mechanismus des Servers stellt dieser fest, wer Nachrichten dieser Art subscribed hat und schickt sie entsprechend weiter.