Instant Messaging in Java

The Jabber Protocols

IAIN SHIGEOKA

MANNING

Greenwich
(74° w. long.)

For electronic information and ordering of this and other Manning books, go to www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact:

Special Sales Department
Manning Publications Co.
209 Bruce Park Avenue Fax: (203) 661-9018
Greenwich, CT 06830 email: orders@manning.com

Manning Publications Co.
209 Bruce Park Avenue
Greenwich, CT 06830

Copyeditor: Lois Patterson
Typesetter: Shan Young
Cover designer: Leslie Haimes

ISBN 1-930110-46-4

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10 - VHG 05 04 03 02

contents

Instant messaging (IM) systems using the Java programming language are poised to become a major part of both consumer and enterprise networking, and will play a core communication role similar to email.

Messaging of course has always been a core feature of the Internet. For example, one of the first and still most pervasive Internet technologies is email. It remains an Internet killer app. However, we all know that Internet communication can be even more interesting and powerful than "plain old email." We should be able to better exploit it as an inexpensive medium for transferring data almost instantaneously. And that is the aim of this book.

Most of the examples and IM issues discussed here are from the point of view of an enterprise developer interested in creating systems for medium or large businesses. This is primarily a bias on my part, as my projects tend to fall into this category. However, the same issues that face enterprises also will be important to any other developer who wants to create reliable, secure systems.

To make the discussion concrete, we'll implement an IM system in Java that complies with the Jabber protocols (which you can find at www.jabber.org). By examining the Jabber protocols in particular, we'll get a real sense for what makes a working IM system tick and why I think they may very well be the best ones to implement in the fragmented field of IM. Finally, the Jabber protocols have suffered from a lack of documentation that I hope this book begins to address.

In this chapter, we'll examine IM systems in general and Jabber in particular. The discussion will be largely nontechnical so we can concentrate on why and where we need IM. Discussion of the technical aspects of the Jabber protocols will occupy the remainder of the book and explain the *how* of Jabber. We will develop a basic Jabber server and client in Java throughout chapters 3–8 to help make the concepts concrete. The chapter closes with a look at the benefits and drawbacks associated with Jabber IM.

1.1 Background on messaging

The idea of instant messaging has been around for a long time. All of the visible IM features like one-on-one chat and group chats existed in other Internet applications long before IM entered the scene. For example, the classic Unix `talk` application allowed users to chat over the network years before IM ever appeared, and group chats have been carried out on Internet Relay Chat (IRC) systems almost as long as `talk` has been around.

The innovation in today's IM systems is the packaging of these separate systems into a managed messaging platform. Jabber takes this further and establishes a *universal messaging address* and the concept of *presence* that can be applied to that address for an even simpler communication experience. A common address and presence are relatively simple technologies that have existed in other forms. However, it's the packaging of the concepts into a single, easy to use, cohesive messaging system that has really caught on. The need for ease of use has increased as more people (many of whom have less technical knowledge or inclination than earlier Internet users) join the network community.

IM addresses are an extension of the well-established email addresses almost everyone has today. However, your email address can be used only to receive email. In contrast, IM ties many message types together using a single IM address (figure 1.1). In an IM system, you can receive a message, chat, or groupchat[1] with a person using one IM address.

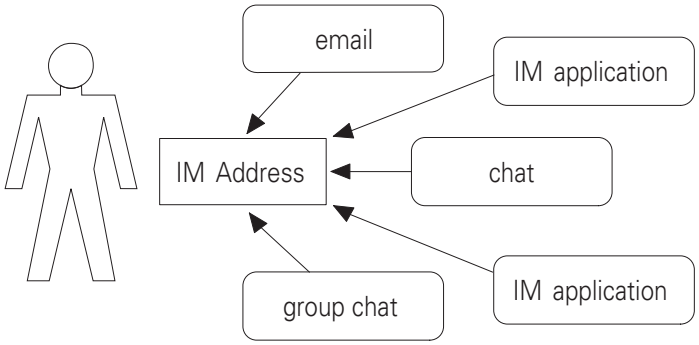

Figure 1.1 IM addresses serve as universal communication destinations. They aggregate many different messaging sources into a single user message endpoint.

Chat and group chat are conversational, and require you to be online at the same time as the person you are chatting with. To make the online rendezvous simple, your IM presence will constantly inform other users when you are online and available for chatting (figure 1.2). IM presence makes communication through IM

[1] IM messages are like memos or emails and tend to be complete like a letter. Chat and groupchat allows you to quickly exchange short text messages in a conversation. Each individual chat or groupchat message is like a sentence taken out of a telephone conversation; it doesn't make sense unless you know what has been said before. The difference between chat and groupchat is that chat is a one-on-one conversation while groupchat is a chat among more than two people.

systems similar to striking up conversations around the water cooler. You can easily see who is available to talk, and casually start conversations.

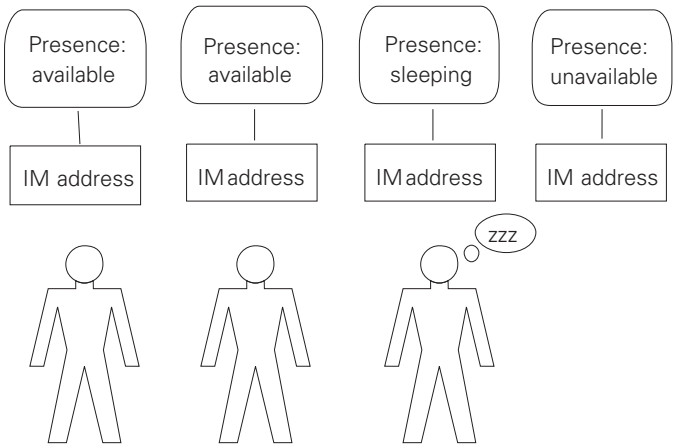

Figure 1.2 Presence shows at a glance who you can communicate with, and the best type of messaging to use (chat, messages, etc).

Older chat systems like Unix `talk` work similarly to the phone system in the sense that they lack presence. When attempting to chat, you have to make blind calls to the person with whom you want to converse, hoping he or she is available to answer the call. Unlike the phone system, though, most people are not available to talk online as much as they are with the phone so the chance of finding someone "at home" online is greatly reduced.

The simplicity and integration of IM systems first caught on with the consumer market. The most successful of these consumer systems is AIM (AOL Instant Messenger) that introduced IM to the mainstream consumer and encouraged its large membership to embrace the technology. The consumer market for IM continues to grow and many opportunities exist for the enterprising developer.

Despite the large consumer market for IM, most of the development community is interested in the opportunities for applying IM technologies in the enterprise (table 1.1). First, enterprises live or die by their ability to communicate within the company and with partners and customers. IM provides new communication channels that are well-suited to many messaging tasks.

1.1.1 A brief history of Jabber

The Jabber project began in early 1998 as the brainchild of Jeremie Miller. The project quickly grew and evolved. It garnered wide public attention when it was discussed on the popular developer discussion website Slashdot (www.Slashdot.org) in January 1999.

The core Jabber protocols matured and the 1.0 release of the open source reference Jabber server was released in May 2000. The core Jabber protocols that were implemented in the 1.0 release of the reference server have remained relatively unchanged to this day.

From its beginnings, the Jabber development community has tried to create IM standards and encourage interoperability between IM systems. These cooperative efforts are in direct contrast to the behavior of other popular IM providers that actively work to keep their systems proprietary and isolated from other IM networks.

As part of the Jabber IM standards effort, in June 2000 the Jabber community submitted the Jabber protocols as a Request for Comments (RFC) to the Internet Engineering Task Force (IETF)[6] as part of its Instant Messaging and Presence Protocol (IMPP) standard. The IETF maintains some of the most important Internet standards including those for email and Internet addresses. Unfortunately, the IMPP effort bogged down and as of this writing appears to be going nowhere fast. Jabber has also tried to participate in other standardization efforts like IMUnified (www.imunified.org) and the Presence and Availability Management Forum (www.pamforum.org) with varying degrees of success.[7]

In May 2001 the Jabber community (www.jabber.org) and Jabber Inc. (www.jabber.com) created the Jabber Software Foundation (foundation.jabber.org).[8] The Jabber Software Foundation is an organization similar to the successful Apache Foundation. Its charter clearly shows the Jabber community's dedication to open standards and interoperability.

---

[6] The IETF website is at www.ietf.org.

[7] Jabber's success at creating Internet standards for IM have mirrored that of the IM community in general, which is to say its success has been "zero." I'm not sure if this is an indication of the immaturity and proprietary nature of IM today or a factor inherent to the IM community or its technology. I would hazard a guess that it's the former, but a cynic would claim the latter. In any case, standards are desperately needed and work continues within the Jabber community to promote and push for standardization.

JABBER SOFTWARE FOUNDATION CHARTER

The Jabber Software Foundation shall provide direct organizational assistance and indirect technical assistance to the Jabber Community in carrying out its mission. Direct organizational assistance will include brand management, logistical support, legal assistance, press relations, and communication facilities. Indirect technical assistance will include project management, protocol specification, standards activities, documentation support, and site development. The Jabber Software Foundation will not make technical decisions for Jabber but will help the Jabber Community to make technical decisions more effectively. All activities and proceedings will be openly accessible and published. Jabber Software Foundation Announcement: http://jabber.org/?oid=1309.

The Jabber Software Foundation is still in its infancy but I have great expectations for it. I'm particularly interested in using the Foundation to clarify and formalize the existing Jabber standards into a form that will allow Jabber technology to be rapidly adopted by the wider development community. This book is an effort toward that goal. This book at least begins to address the need for better documentation.

As Jabber moves forward, much work remains to be done. Further standardization work is still under way. In addition, new standards are being proposed to extend the Jabber protocols beyond simple IM to meet the wider needs for electronic messaging of all forms. These changes include better security, and support for enterprise requirements such as transactions, quality of service, and delivery guarantees.

1.1.2 Goals of the Jabber project

In many ways, the Jabber project's goal is simply to build a better IM system supporting real-time presence and messaging. From my discussions with other Jabber developers, the underlying goals seem to really be about defining what "better" means. For most Jabber developers, better means:

[8] There is a tight link between the Jabber community and Jabber Inc. (usually called Jabber.com or jc). Jabber Inc. employs most of the "core" members of the Jabber community and developers including Jeremie Miller, the creator of Jabber. However Jabber Inc. has been earnest in its efforts to keep the Jabber community and Jabber standards open. The Foundation is primarily an effort to formalize Jabber Inc.'s commitment to open Jabber standards and provide a legal entity to represent the Jabber community's interests (especially when they don't necessarily match that of Jabber Inc.). Many of the board members of the Foundation are Jabber Inc. employees; however there are enough outsiders on the board to keep Jabber commercial interests balanced against that of the community.

- *A completely open system and standards*—Unlike other systems, Jabber will be completely open, thus allowing anyone to create Jabber implementations at no cost and with no strings attached.
- *Open, XML-based technology*—Extensible Markup Language (XML is an enabling technology with many benefits with respect to flexibility, availability, and ease of use. The use of XML helps to "future proof" the Jabber protocols. At the same time, XML is a well-known technology that people are interested in using. There is also an extremely wide selection of tools for modeling, analyzing, programming, and authoring XML.
- *Interoperability with other* IM *systems*—The value of a communication system increases with the number of people with whom you can communicate. Optimally, a user of an IM system should be able to IM with any other user regardless of the underlying IM system. Jabber will maximize its value by working with as many other IM systems as possible.
- *Simple protocols*—Simple protocols are easier to design and implement. In addition, for most Jabber developers, simple things have an inherent appeal that we strive for constantly.
- *Simplifying the responsibilities of clients whenever possible*—There will be many clients and few servers in any Jabber system. It makes sense to design the Jabber system so that writing clients is as simple as possible. In addition, clients may run on resource-constrained systems so reducing their needs increases the possible types of clients.
- *Control over the system*—No organization, group, or service provider should be able to control the system. Jabber's design allows anyone to create a Jabber server and run their own Jabber network as they see fit.

These unwritten goals seem to be commonly held among Jabber community members. They represent an open, self-reliant community that sees the benefits in sharing technology and knowledge whether you are an open-source or commercial developer. From these basic goals, the Jabber community has produced a feature-rich IM framework.

1.2 What is Jabber?

Jabber IM means different things to different people. End users typically associate Jabber with the Jabber IM system as a whole, just as they consider the Web to mean the entire web system including web servers, web clients and the protocols and data structures that power the World Wide Web. Developers often confuse the Jabber

open source reference implementation of a Jabber server called jabberd[9] for the Jabber protocols that it supports. For the purposes of this book, we'll restrict the discussion of Jabber IM to the Jabber protocols and messaging model.

One of Jabber's most discussed advantages is its truly open nature. The Jabber protocols and data formats are all documented either directly in the Jabber standards documents or as source code in the freely available Jabber reference server. This is in stark contrast to all other major IM systems that use proprietary standards.

Jabber's advantages don't end there. Its use of XML-based data formats exploits the popularity and extensibility of XML. In addition, Jabber uses a simple, distributed client/server architecture. The combination of the simplicity of the basic client/server communication model with the scalability of distributed servers makes it well-suited to the dynamic environments that IM systems will be used in the future.

Let's break down Jabber technology to get an overview of how it works.

1.2.1 Jabber's XML-based data formats

If Jabber's open standards are its primary political and business advantage, its use of XML as its standard data format is its crucial technical advantage. XML is a World Wide Web Consortium (W3C) standard that defines a standard, generic data format for documents. Its primary technical advantages are its simplicity, extensibility, and a compromise between easy human and machine readability.

All Jabber communications involve the exchange of Jabber packets, each being a well-formed XML fragment. These XML fragments can be considered XML subdocuments within the Jabber communication stream. For example, to send a message to iain@shigeoka.com, you would send a Jabber packet that looks like this:

```
<message to='iain@shigeoka.com'>
 <subject>How are you today?</subject>
 <body>Just thought I'd drop a line and see what you're up to.</body>
</message>
```

Even if you don't know anything about XML you can probably figure out what each part of this XML fragment does.[10] If you are XML-savvy, Jabber defines all its packets in standard XML Document Type Definitions (DTD) and uses XML namespaces extensively to define both its own standard packets, and provide

[9] The `Jabberd` reference server development is continuing along two separate paths. `Jabberd` will continue to improve and expand for the foreseeable future. It provides the definitive reference implementation of the current Jabber protocols that we'll cover in this book. A revised set of Jabber protocols, referred to as Jabber Next Generation (JNG), is in development. A new reference server named `Jabberd` will be written to implement these protocols when they are settled upon.

The strong interest in XML and Java has created a huge market of tools, books, programming libraries, and third-party experts (consultants, contract programming firms, etc.). This market legitimizes both XML and Java, and helps to ensure that anything based on Java and XML will be compatible with future technologies. It appears that we'll be using XML for all sorts of programming problems for a some time to come. However, even if it doesn't work out that way, enough people are committed to XML that new technologies will almost certainly continue to offer XML support or conversion tools.

The use of XML in the Jabber protocols is an excellent start in creating an accessible, flexible IM system. However XML alone is not enough to make Jabber truly interesting. The complementary breakthrough with Jabber is its simple, scalable architecture.

1.2.2 Jabber's simple architecture

The Jabber messaging model follows the well-understood client/server architecture. Jabber clients only communicate with Jabber servers in their Jabber domain.[12] Jabber domains break the entire IM universe into separate zones of control, each supported and managed by a separate Jabber server. This is in contrast with most IM systems that use one centralized server for the whole IM system.[13] In Jabber, messages pass from the sender's client, to the sender's server, to the recipient's server,[14] to the recipient's client.

Jabber is client/server

The basic Jabber communication model follows the simple and well-understood client/server architecture model. In client/server systems, the client displays information to the end user and handles user requests. Information is passed from the client to a server that offers well-defined services (figure 1.5).

[12] This statement is an over-simplification. Normal Jabber messaging occurs through the server. However, clients are always free to send data directly to each other through any means at their disposal. How to do so, though, is not covered by the Jabber standards. Jabber does provide an out-of-band packet to help clients arrange for these direct, client-to-client communications. However the client-to-client exchange does not occur within the Jabber system.

[13] A central server is the IM system architecture for services like AIM, Microsoft Messenger, and Yahoo! Messenger.

[14] If the sender and receiver both use the same Jabber server, this step is not necessary.

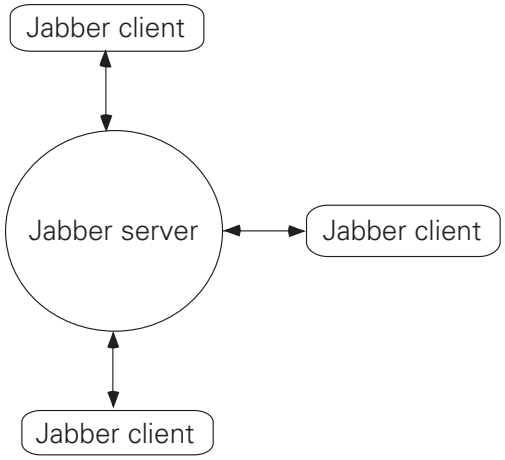

Figure 1.5
Jabber client/server architecture: clients use one connection and only communicate directly with their server.

In Jabber, the client/server model is heavily weighted to favor the creation of simple clients. Most of the processing and IM logic is carried out on the server. Minimal Jabber client responsibilities give Jabber client developers the most flexibility in creating Jabber IM clients that fit the needs of users. Embedded and other limited resource environments can support simple Jabber clients. Heavyweight Jabber clients can concentrate on the user interface and ease of use issues. Finally, the simplicity of creating Jabber clients encourages people to write more clients on different platforms using different programming languages, helping to spread Jabber access far and wide.

Unlike distributed systems, such as those based on peer-to-peer technology,[15] the simple Jabber client/server architecture provides an excellent opportunity to implement centralized control of Jabber domains and provides opportunities for enforcing quality of service guarantees. This is especially important for enterprises that may need to enforce corporate communication policies. In addition, as IM systems are used for mission-critical messaging beyond direct user communication, it will become increasingly important to ensure certain levels of quality of service can be met.

Jabber distributed servers build the Jabber network

Just as the current email system allows separate, distributed email servers to manage email domains, Jabber servers manage a Jabber domain. Like email, Jabber domains are defined by an Internet domain name. So a Jabber server that manages

[15] Peer-to-peer describes a family of technologies that allows network systems to be created without the use of a central server.

users can communicate with other Jabber servers. These same advantages have been the primary reasons why the current Internet email system is so pervasive and maintains such a high degree of cohesiveness.

In many ways, Jabber's simple, client/server architecture with distributed servers is old technology. There is nothing really innovative about Jabber's messaging model. In my opinion, this is a major strength rather than a weakness. IM's major innovation is the addition of presence to communication systems. Jabber's innovation is the use of an open XML data format for the data being sent in the communication system. Adding any more innovations at the same time would have probably resulted in a much less elegant and easy to build system.

1.2.3 Jabber's four core concepts

Four central Jabber concepts form the basis for Jabber systems. Before we can look at the Jabber protocols, we must understand these concepts as they exist in Jabber. They're straightforward but it is important for us all to be working with the same set of messaging concepts before setting off into Jabber's details. These four central concepts are:

- Jabber domains
- Users and resources
- Jabber IDs
- Presence

Jabber systems organized into networks and domains

The Jabber universe is broken down into several logical sets and subsets of entities. From the largest to the smallest, we have a Jabber:

- *Network*—All Jabber domains that exchange messages. A network must contain at least one domain.
- *Domain*—A subset of the network containing all entities that handle or belong to a domain. Jabber domains provide local control over parts of the Jabber network while still communicating with users outside of the Jabber domain. A domain is defined by:
  - A valid Internet domain name address.
  - The server that handles connections to that address.
- *Server*—A logical entity that manages a Jabber domain.
- *User*—An entity representing a *logical* message delivery endpoint. Jabber data packets are usually addressed to users, but are always delivered to a resource. Users are managed on the server with user accounts.

- *Resource*—An entity representing a *particular* message delivery endpoint for a user. All Jabber data packets are delivered to resources. Jabber clients play the role of Jabber resources.

A minimal Jabber network is composed of a single Jabber server handling one Jabber domain, and the Jabber clients that use that server, as shown in figure 1.7. Jabber servers can exchange messages using the `jabber:server` protocols. When servers do this, they are federating their Jabber domains to expand the Jabber network.[16]

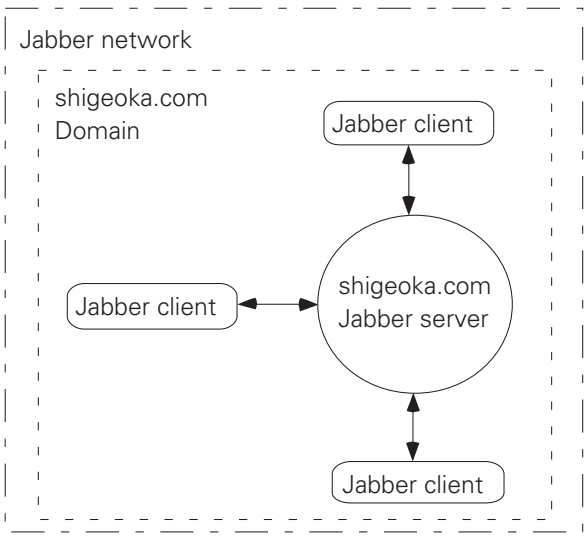

Figure 1.7
A minimal Jabber network with a single domain managed by a Jabber server and zero or more clients.

Server federations are the primary method for Jabber networks to grow. Many Jabber servers are configured to federate with any other Jabber server on Internet. This essentially creates a single Internet Jabber network containing thousands of users.

It is also possible to bridge Jabber systems by using the client protocols to exchange messages between domains as shown in figure 1.8. I'll refer to this process as *bridging* to differentiate it from the privileged server-side access that server federation requires.

[16] Jabber servers find each other by opening connections to the domain name address of other servers (see chapter 9 for more details). Jabber server administrators can play some "DNS tricks" such as allowing multiple physical machines to act as the same logical Jabber server. This trick, commonly referred to as *round-robin DNS*, creates *server farms* in advanced Jabber installations. Jabber servers are also free to develop their own proprietary server-to-server connections in addition to (or as a replacement for) support for standard Jabber server-to-server connections if needs demand.

In jabberd, the Jabber server reference implementation written in C, a hybrid bridging solution is provided by server components known as transports. They have privileged Jabber server access but typically use client protocols to access foreign systems.

For example, one of the most popular and most controversial is an AIM transport. The AIM transport lets Jabber users transparently send messages to AIM users and track AIM users presence, an important capability when most IM users are currently using AIM. However you should be careful about creating transports or bridges as it may violate the usage policies for foreign services. AOL in particular has been resistant to AIM transports.

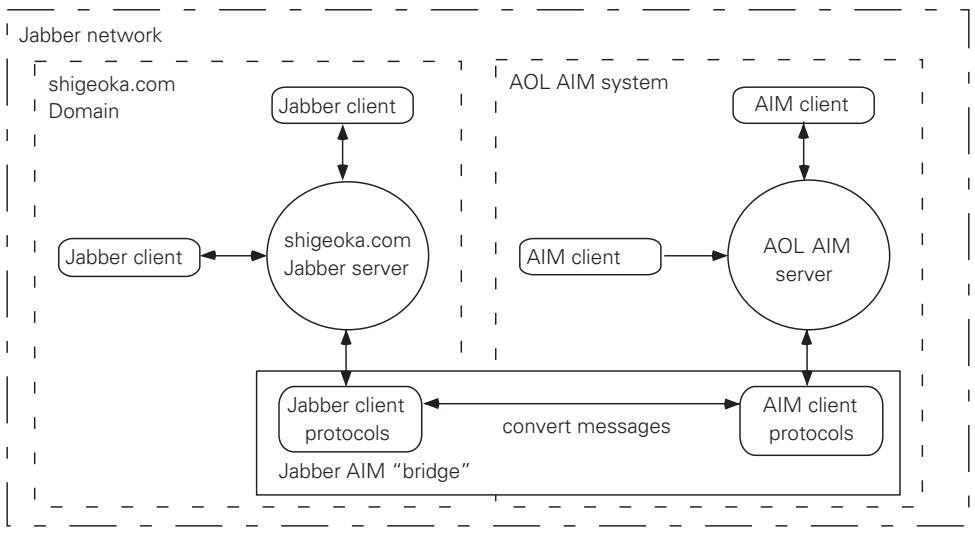

Figure 1.8 Jabber clients or server components known as transports can bridge IM systems by serving as bridges. Here a Jabber AIM bridge acts on behalf of Jabber users to deliver AOL AIM messages.

For example, imagine you have a small business with an intranet local-area network (LAN) that is isolated from the Internet. If you set up a Jabber server on that network and use Jabber clients on your workstations, you have created a companywide Jabber network. You can also create isolated Jabber networks on the Internet or other large networks by preventing your Jabber server from connecting to other Jabber servers. Your clients will only connect with your server, and your server can't connect with any other Jabber servers, creating a separate Jabber network.

The last major protocols for our Jabber IM system are rosters and user presence. Our tour begins by describing what rosters are, and how they work with the presence protocols to maintain a systemwide user presence. We'll cover the roster protocol in detail and finish with the modified Java Jabber server and client software that supports rosters and user presence.

8.1 Roster protocol: presence's missing piece

The story of the Jabber roster began in chapter 5 with the presence protocols. The presence protocols allow us to do two things: update a user's presence status and manage presence subscriptions. Presence updates are straightforward, and they are covered in chapter 5 where presence is used to manage groupchat group membership. Presence updates can also be used to update the user's presence status.

The major difference in updating a user's presence status versus groupchat is controlling where the presence updates go. In groupchat, presence updates are sent to the groupchat server, and from there, to group members.

It is possible for every user to send their presence to all other users. However, if we sent the presence updates between every user on the Jabber system, servers and clients would quickly be swamped, leaving no resources for messaging! In addition, users need to control access to their presence updates for privacy and ease of use.

To address these issues, Jabber has a concept of a presence subscription. As the name implies, a presence subscription determines the subscribers who wish to receive presence updates from presence publishers. Subscribers must request a subscription from the publisher, and the publisher has the option of accepting or refusing a subscription. Each user must manage both their publisher and subscriber presence relationships.

To organize and manage subscriptions for each user, Jabber has defined a standard data structure known as the Jabber roster. The Jabber roster[1] is a list of other users identified by their Jabber ID. We'll call these users *roster subscribers* even though their presence subscription relationship can be as presence subscribers, publishers, or both for the roster owner. Storing all subscription information for each user in their roster helps to simplify the job of the client developer and in some cases the server developer.

For the client developer, having every subscription relationship described in the roster provides them with a single, authoritative source of presence information. For the server developer, when carrying out presence related tasks, you only

[1] The Jabber roster is better known by its AOL name, the "buddy list."