Secure Peer-to-Peer Networking: The JXTA Example

William Yeager and Joseph Williams

An open-source framework, JXTA can accommodate centralized and decentralized certificate authorities for P2P networking.

The rise of Napster and its technical descendants—Gnutella, Morpheus, and others—has probably caught everyone's attention by now. These products rely on an abbreviated form of distributed computing to connect millions of users, letting them share music, videos, images, and other files. The technology that enables this form of distributed computing is called *peer-to-peer computing* or *peer-to-peer networking*, depending on your view of the technology; most people simply refer to it as P2P.

File sharing is just one aspect of P2P, as the "What is P2P?" sidebar discusses. Companies are developing applications that will use P2P networking to create ad hoc work groups and virtual communities. When married with mobile technologies, such as the Wireless Application Protocol or IEEE 802.11, P2P can also become the foundation for offering location-based services. Such services can support virtual play groups, such as students communicating with each other at a ski resort. Or they can support emergency services, such as an ad hoc computing environment for a hazardous-materials clean-up team. Projects like the Search for Extraterrestrial Intelligence (SETI) At-Home also let P2P networks share computing cycles among computers.

P2P has so captured the business community's imagination that *Fortune* magazine recently featured P2P as one of the four technologies that will shape the Internet's future (Erick Schonfeld, "Post-Napster, Peer-to-Peer Computing Gets Ready for Prime Time," *Fortune*, 25 Oct. 2001).

BASIC PREMISE OF P2P

A P2P application has to deal with several computers (peers) running on different physical machines. These peers can be running a daunting variety or versions of operating systems, and work in varied security, application, and tools environments. Given a collection of such peers, how do you create P2P applications? For starters, you need a way for peers to communicate and a way to uniquely identify each peer in the network. Since most P2P networks are ad hoc, the network must dynamically discover peers and track when peers join or leave the group.

Once you start connecting

What is P2P?

There actually isn't a universally accepted definition of peer-to-peer computing or networking; people have applied the term to a wide range of technologies. In general, P2P describes an environment where computers (hosts) connect to each other in a distributed environment that does not use a centralized control point to route or connect data traffic. In practice, P2P technologies deployed today adopt a network-based computing style that neither excludes nor inherently depends on centralized control points.

Napster wasn't really a pure example of P2P technology, because it relied on centralized servers to connect its user community. A simple example of a P2P implementation comes from the IEEE 802.11b's ad hoc working-group mode. In this mode, computers join a wireless network as peers, and every computer in the network can communicate directly with every other computer in the network without having to first communicate with a centralized server.

these peers, security becomes an issue. Sharing files and other computing resources among peers opens up an enormous potential for hackers, vandals, and thieves to disrupt and subvert the peered computers. In fact, opponents of P2P often cite authentication, authorization, and privacy concerns as the primary roadblocks to the general deployment of P2P networks. Sharing music files in a P2P environment is one thing, but sharing sensitive corporate information is quite another.

JXTA (pronounced "juxta") addresses these security concerns; it is a popular, open-source framework for enabling P2P networking.

JXTA

In April 2001, Sun CEO Bill Joy unveiled Project JXTA, a community development project to create a new programming platform. He wanted this project to solve several problems in modern distributed computing, especially in P2P computing.

Project JXTA has the following objectives, intending to address the shortcomings of current or developing peer-to-peer systems:

- *Interoperability.* Interconnected peers must easily locate and communicate with each other; participate in community-based activities; and offer services to each other seamlessly across different P2P systems and communities.
- *Platform independence.* P2P must be independent of programming languages, such as C or Java; system platforms, such as the Microsoft Windows and Unix operating systems; and networking platforms, such as TCP/IP (transmission control protocol/Internet protocol) or Bluetooth.
- *Ubiquity.* P2P has its broadest reach when it works with any device having a digital heartbeat, including sensors, consumer electronics, PDAs (personal digital assistants), appliances, network routers, desktop computers, data center servers, and storage systems.

JXTA relies heavily on XML (Extensible Markup Language). It also separates its protocols from specific language bindings, which in effect makes it language independent (Piroz Mohseni, "An Overview of JXTA Architecture," http://softwaredev.earthweb.com/java/article/0,,12082_783281,00.html). The JXTA specification does not indicate when or why various computing devices can form a peer group; it simply supplies the infrastructure for forming and managing groups. Once JXTA enables a network of peers, communication among peers occurs via a *pipe* (channel). Messages sent into a pipe go to all peer hosts that are listening to that pipe. The Pipe Binding protocol controls how peers connect and disconnect to a pipe at runtime. In contrast to these multipeer pipes, a point-to-point pipe permits one-way communication between two peers (thus reducing network traffic).

Pipes, peers, and services communicate with each other via advertisements and by using XML. Typical peer advertisements describe information such as name, peer id, properties, and services offered. JXTA has several core advertisement types—peer, peer group, pipe, service, content, endpoint, and user defined.

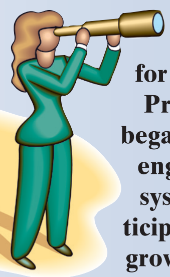
In short, JXTA allows for management of peers individually and as a group. It allows for communication among peers via pipes and also defines the message structure exchanged among peers in advertisements.

There are several other P2P technologies, of course. Many of these have been popularized by the music-sharing phenomena. Groove (http://www.groove.com), Morpheus (http://www.musiccity.com), and Napster (http://www.napster.com) all, to some extent, deploy self-organizing, file-sharing systems. Microsoft also has a P2P technology in the Windows Messenger core that enables its Presence and Instant Messaging platforms. Each of these is a proprietary technology, so this presentation of the open-source JXTA platform provides a generalized vehicle for discussing the P2P security problem.

SECURITY

Before developers can build new enterprise and mission-critical applications, JXTA must address security. Some security questions are quite fundamental: How will a peer-to-peer environment deal with security? Can such networks use the same notions

Learn More about JXTA

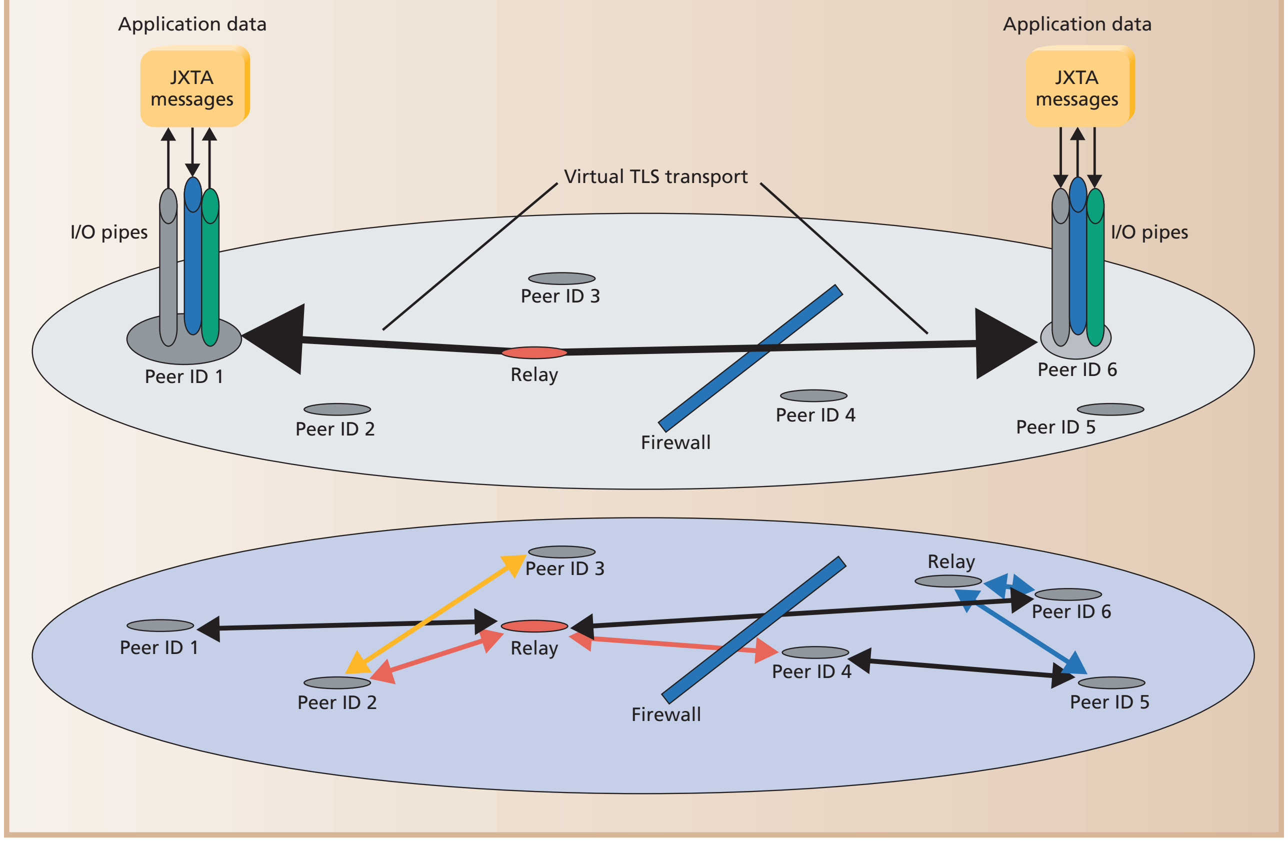


For more information about Project JXTA, visit its Web site at http://www.jxta.org. The site hosts documentation, JXTA source code, sample applications, and technical discussions.

The security project team for Project JXTA has developed a Web-of-trust-based security model for JXTA. This model uses public-key cryptography to let groups of peers communicate securely (http://security.jxta.org for more information).

Project JXTA initially began with a small group of engineers at Sun Microsystems along with the participation of a small but growing number of experts from academia and industry. Sun now distributes JXTA under an open-source license. As an open-source product, JXTA has a community of P2P-interested users helping to codevelop it.

Figure 1. JXTA uses Transport Layer Security.



of trust, authentication, integrity, and nonrepudiation that IT professionals expect from traditional networks? Must users and application developers radically alter their ideas about how to provide security when working with P2P technologies?

The JXTA community is considering a security model that relies on existing trusted technologies, without compromising the strong security that users and enterprises expect from traditional computing approaches. Such an approach is practical as a consequence of three recent choices by the Project JXTA team:

- the adoption of Transport Layer Security, an emerging industry protocol for the secure transport of information;
- exploitation of the JXTA protocols' end-to-end transport independence; and
- use of X509 version 3 digital certificates in a way that neither mandates nor excludes centralized certificate authorities.

Because this approach doesn't require inventing new technologies, plans for applications development and deployment can proceed with minimal delay. Employing established technologies and protocols also translates into a higher level of trust from the outset. It also obviates the need to radically shift the industry's thinking about how to provide security.

Transport Layer Security

JXTA has adopted TLS version 1 to support reliable, private connections between peers. Figure 1 shows an upper layer, the JXTA virtual TLS transport. The lower layer is the JXTA virtual network. Thus, we have the TLS transport multiplexing multiple pipe connections between two peers on the JXTA virtual network. The TLS specification is currently under development by the Internet Engineering Task Force (IETF) network working group. Although TLS and Secure Sockets Layer—a more well-known network security protocol—share several similarities, they weren't designed to be strictly compatible, although most developers don't expect a hardship in supporting both protocols.

The rationale for selecting TLS for use by Project JXTA is identical to that cited by the IETF: cryptographic security—the protocol should be able to establish a trusted connection between two parties.

The protocol's interoperability is also important. Independent programmers should be able to develop applications that can successfully exchange cryptographic parameters

Advanced References for TLS

The Internet Engineering Task Force offers detailed information on TLS at http://www.ietf.org/rfc/rfc2246.txt. In addition, a reference from Eric Rescorla, *SSL and TLS: Designing and Building Secure Systems* (Addison Wesley, 2000) gives further explanations of SSL and TLS.

Because of its widespread popularity and Berkeley-style licensing policies, Pure TLS, an implementation from Claymore Systems (http://www.claymoresystems.com), is the brand of TLS included in JXTA distribution.

without knowledge of one another's code.

Because new public-key and bulk encryption methods will arise over time, the protocol should also be adaptable enough to incorporate them. This extensibility will also prevent the need to create a new protocol (and the attendant weaknesses of doing so), avoiding the need to implement new security libraries.

Any cryptographic protocol should also be relatively efficient. Cryptographic operations, particularly those for public key, tend to be highly CPU intensive. For this reason, the protocol should incorporate schemes to reduce the number of connections and improve network efficiency.

TLS has two layers: the TLS record and the TLS handshake protocols.

Record protocol. At the lowest level, layered on top of a reliable transport protocol like TCP, is the TLS record protocol. It provides connection security in two ways:

- *TLS connections are private.* These connections use symmetric cryptography—triple DES (Digital Encryption Standard), RC4, and other algorithms—for data encryption. TLS generates unique keys for each connection and bases them on a secret negotiated by another protocol (such as the TLS handshake protocol). Applications can also use the record protocol without encryption.
- *TLS connections are reliable.* Message transport includes a message integrity check that uses a hash message authentication code (HMAC) or keyed message authentication code. TLS uses secure hash functions—such as SHA-1 (Secure Hash Algorithm), MD5 (Message Digest 5), and others—for MAC computations.

The connection uses the TLS record protocol to encapsulate various higher-level protocols.

Handshake protocol. One such encapsulated protocol, the TLS handshake protocol, lets the server and client authenticate each other and negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. The TLS handshake protocol provides connection security that has three basic properties:

- The protocol can authenticate the peer's identity using asymmetric (public-key) cryptography algorithms, such as RSA (Rivest-Shamir-Adleman) and the Digital Signature Standard. An application can make this authentication optional, but minimal security usually requires authentication of at least one peer.
- The negotiation of a shared secret is secure. Eavesdroppers cannot pick up this secret, and for any authenticated connection, the secret is unobtainable, even by an attacker who places himself in the middle of the connection.
- The negotiation is reliable. No attacker can modify the negotiation communication without being detected by the communicating parties.

An added advantage of TLS is its independence of application protocol. Implementations can layer high-level protocols on top of TLS transparently. The TLS standard, however, does not specify how to do this layering. For sources of TLS information see the "Advanced References for TLS" sidebar.

End-to-end transport independence of JXTA protocols

At one layer of abstraction, JXTA technology is a set of protocols with each protocol defined by one or more messages exchanged among peers. Each message has a predefined format and can include multiple data fields. In this regard, JXTA is akin to TCP/IP—TCP/IP links Internet nodes together, and JXTA technology connects peer nodes. Because it is merely a set of protocols, TCP/IP is platform-independent; so is JXTA. Moreover, JXTA technology is transport independent and can use TCP/IP as well as other transport standards with no impact on the messages themselves.

Aside from providing an exceptionally flexible architecture for implementation on a wide variety of devices, this transport independence has important consequences for JXTA security. Many users familiar with the Wireless Application Protocol (WAP) are probably also acquainted with the notion of the so-called "WAP gap." This potential security breech exists when messages moving between wireless and wired networks must pass through a gateway and undergo a protocol conversion. This process exposes the clear text of messages and can lead to a security compromise.

JXTA users will not have this exposure, because JXTA messages are independent of the underlying transport and its protocols. Encrypted content remains encrypted, even during the protocol conversions that occur between networks. This is a natural consequence of JXTA's design and requires no special actions by users or programmers.

P2P applications based on JXTA technology can use the strongest encryption algorithms available, without concern that anyone but authorized parties will see clear-text messages related to the transaction.

Digital certificates and certificate authorities

Authentication and nonrepudiation are central concerns in electronic security. Both ensure the identities of parties to a transaction. A highly reliable and widely used scheme for achieving these goals has incorporated digital signatures and public-key cryptography.

Further, to ensure that no party to a transaction is an impersonator, secure systems can use *digital certificates*, documents that bind a user to a public key. At a minimum, certificates contain a public key and a name. They can also contain an expiration date, the name of the certificate authority (CA) that issued the certificate, a serial number, and the certificate issuer's digital signature. A CA's key benefit is that some entity is willing to say that it has unambiguously established the identity of an individual who is using a specific key. Parties corresponding with this individual can then have a secure transaction with the CA to verify the individual's identity.

Digital certificates come from a trusted third party or an X509-version-3-compliant CA willing to vouch for an identity and public-key pair. The CA can also be the individual's employer or a third party specializing in sponsoring unaffiliated parties. CA policies vary, depending on the strength of authentication required by users and the CA's specific functional requirements.

Despite these benefits, the notion of a centralized CA is not always appropriate to the world of peer-to-peer computing and is at times contradictory.

Large numbers of peers could want to conduct a secure transaction without involving a centralized infrastructure. Fortunately, the answer is simple—JXTA lets peers become their own CA, generating their own root certificates, verifying that they are associated with a specific public key. Larger groups have the added flexibility of designating root CAs for a peer group. The group can then use these root CAs as a strong way to authenticate membership and establish nonrepudiation within the peer group.

In JXTA, peers can also form peer groups and designate one member as the group's CA, making its root certificate part of the peer group's binary version of JXTA. Or a group member can transfer this root certificate to a prospective member over a TLS connection with *out-of-band identity verification*, a process that presents the 10 low-order hex digits of a SHA-1 hash of the root certificate to the recipient of the root certificate. The recipient can then contact the owner and ask him to repeat this sequence to verify the just-received root certificate's origin.

Finally, in a method equivalent to security as it is on today's Internet, a peer group can use a well-known CA to issue certificates, and the CA's root certificate will be again part of the peer group's binary.

This simple approach employs existing public-key and digital certificate technologies without conceptual changes, an all-important requirement for quickly gaining trust in Project JXTA's approach to security.

The JXTA framework's key benefit is its ability to provide security via well-understood, trusted approaches. Likewise, JXTA's transport protocol independence lets applications avoid the security breaches imposed by gateways or other network infrastructure.

Project JXTA has now completed J2SE and J2ME MIDP implementations. The latter is for small-footprint devices like mobile phones. A C implementation is nearly completed. These three are edge-to-edge interoperable and thus will provide JXTA P2P functionality across the device space. At JavaOne 2002, held in San Francisco during March, developers demonstrated group instant messaging among mobile phones, laptops, and desktop systems with firewall traversal, using the JXTA protocols. ■

William Yeager *is Project JXTA Chief Technology Officer. Contact him at yeager@sun.com.*

Joseph Williams *is Chief Architect, Americas, for Sun Microsystems' iPlanet Professional Services group. Contact Williams at joseph.williams@sun.com.*

The information presented here represents the authors' opinions and not necessarily those of Sun Microsystems.