A Self-Organizing Publish/Subscribe Middleware for Dynamic Peer-to-Peer Networks

Markus Oliver Junginger and Yugyung Lee
University of Missouri-Kansas City

Abstract

Peer-to-peer needs new middleware technologies and counterparts to the widely established CORBA, EJB, COM+, and messaging systems products. Specially designed middleware would release the advantages of peer-to-peer networks to a broad spectrum of applications. Network-specific advantages like scalability, fault tolerance, and resource availability could easily be utilized without any concerns about their underlying infrastructure and resources. We address this need with our P2P Messaging System, which focuses on high-performance group communication based on a publish/subscribe model. The P2P Messaging System considers the heterogeneous and dynamic character of peer-to-peer networks by an augmented topology and its supporting features. The P2P Messaging System was evaluated by experimental benchmark tests, and the results provide evidence of its efficiency and scalability.

Great potential lies in peer-to-peer networks as a vast amount of resources are waiting to be utilized. Several years ago, file-sharing applications like Napster and Gnutella impressively demonstrated the possibilities for the first time. Because of their success, peer-to-peer became synonymous with file sharing. Nevertheless, the fundamental peer-to-peer concept is general, and can be an alternative to client/server networks for some applications. Examples are numerous and include content delivery, media streaming, collaboration tools, and knowledge management. Compared to established networks, peer-to-peer solutions have the potential to offer superior scalability, fault tolerance, and an increased number of available resources. Additionally, peer-to-peer solutions may reduce infrastructure costs because they take workload and traffic off servers and onto peers. Thus, using a peer-to-peer network reduces acquisition and running costs for server hardware. This argument gains further weight taking into account the increasing number of end users who own powerful computers connected by high-bandwidth links.

Despite these rich possibilities, peer-to-peer applications beyond file sharing are still rare. The reasons are not only the popularity and omnipresence of Web-based services, but also the lack of peer-to-peer enabling technologies. Companies cannot afford the lengthy and costly development of their own peer-to-peer infrastructure. In times of rapid application development and short time-to-market requirements, middleware products are needed. CORBA, EJB, COM+, and various messaging systems ease the development of distributed applications, but they do not incorporate features needed for decentralized peer-to-peer environments. Although there are no counterparts to these established middleware products yet, there are several efforts being made in this direction. One of the most advanced is Project JXTA,[1] an open-source peer-to-peer platform managed and mostly developed by Sun Microsystems. Unlike first-generation peer-to-peer networks, which are tied to a specific application, JXTA provides a general-purpose platform targeted to meet the requirements of numerous applications. However, JXTA is still in the maturing process and leaves room for improvement: besides changing networking protocols and stability issues, there are also concerns on performance and scalability.

The P2P Messaging System presented here focuses on group communication in which a group comprises peers with a specific common interest within the network. We assume group communication especially relevant for peer-to-peer applications, because the focus shifts from the individual network node toward a group of nodes. The potential of peer-to-peer technology lies in the collaboration of several peers, allowing each peer to profit from others. As collaboration of a group usually requires communication among its members, we see an increasing need for peer-to-peer group communication. Consequently, we believe a middleware tailored to this purpose would improve the development process and peer-to-peer applications themselves significantly. Besides the reduced development time, the product's quality may also profit from the (ideally) advanced reliability, performance, and functionality offered by a middleware product.

[1] *Project JXTA, Available at http://www.jxta.org*

0890-8044/04/$20.00 © 2004 IEEE

Challenges

Designing such peer-to-peer middleware is a complex task. We cannot rely on a static network of dedicated and mostly centralized service providers, which are still the most common approach. Instead, peer-to-peer networks confront us with several new challenges:

- *Shared environment*: Infrastructures tend to become shared platforms for several independent applications that may have contrary requirements and interfere with each other.
- *Scalability*: The large number of nodes within a peer-to-peer network may affect performance, latency, and reliability.
- *Dynamic network*: Nodes are unreliable because they may join or leave the network at any time.
- *Dynamic node characteristics*: Since nodes are autonomous, their characteristics may change. For example, concurrent applications may decrease the available bandwidth of nodes.
- *Network heterogeneity*: Each peer is unique according to its networking and computing power, location in the physical network, and provided services.
- *Quality of service*: Given an unpredictable network, the quality of network-dependent services might not be guaranteed but improved.
- *Security*: The middleware should be resistant to malicious peers. However, this extensive issue does not fall into the scope of this article.

Peer-to-Peer Topology vs. Group Communication

This section points out why the topology of a peer-to-peer network conflicts with requirements of efficient group communication. In a peer-to-peer network, each node is considered a peer that has at least one connection to other peers. We assume application-dependent needs for communication among certain peer groups. For example, there may be a group participating in a videoconference, while another logically independent group is interested in chatting with each other. Although these groups belong to two separate applications, they may share the peer-to-peer network. Each node forwards all messages, but listens to only the messages belonging to its group. Because the peers forming a group are usually spread randomly within the network, efficient group communication is a nontrivial task as the location and number of participants is not known. There are several communication concepts, which are reviewed briefly here. Simple peer-to-peer networks, like Gnutella,[2] forward messages to every peer, while more recent approaches, like Pastry [1], implement routing mechanisms to prevent message flooding. Another method to improve efficiency is to let elected peers perform additional services for other peers, at the cost of losing decentralism aspects. JXTA, for example, uses *rendezvous peers* to cache resource data and propagate messages.

Nevertheless, peer-to-peer topology is not an ideal choice for group communication due to several factors related to the more or less random wiring of nodes. First, latency is usually high, because messages have to traverse a large number of peers that also may not be members of the group to which the message belongs. Second, performance is degraded by slow peers on the route. Third, the bandwidth of each peer is partially consumed by traffic caused by other peers. More centralized solutions with elected service peers may also suffer from scalability restrictions when the number of peers requesting services exceeds the capabilities of the service peer. Other scalability problems apply to networks without routing. Here, bandwidth is consumed by message flooding, which occurs when messages are forwarded to all neighbor peers regardless of whether the message is relevant to the recipients.

[2] *http://www.gnutella.com/*

The P2P Messaging System

Our P2P Messaging System addresses the discussed mismatch between peer-to-peer topology and group communication. The goal is to provide a scalable, robust, and fast middleware suited to a variety of peer-to-peer applications. It supplies message-oriented group communication based on the publish/subscribe model, which is also used by established messaging systems in different domains. In short, this model is based on usually topic-based groups, which consist of publishers and subscribers. Group members can be subscribers, publishers, or both. Messages are sent by publishers and delivered to all subscribers. In our case, a peer can become a member of any number of groups in order to send and receive messages. Further information on the system, and its concepts and algorithms is available in [2].

Although the presented group communication concepts can be used by themselves, the intention is to collaborate with existing peer-to-peer networks like JXTA. This symbiosis allows use of the existing services of peer-to-peer networks (e.g., looking up peers), while providing its publish/subscribe service to peer-to-peer applications. However, the actual message delivery is realized by a separate infrastructure.

Before we go into detail regarding the individual concepts, we briefly illustrate how they relate to the previously presented challenges:

- *Shared environment*: Dynamic creation of overlay networks provides subenvironments, which take care of individual group requirements.
- *Scalability*: The overlay networks are formed using scalable multiring topology, which adjusts dynamically to changes.
- *Dynamic network*: The topology reduces the dependence on single nodes. In addition, new routes evolve dynamically, and backup links compensate node failures and congestions.
- *Dynamic node characteristics*: Nodes relocate within the topology according to their characteristics. This is accompanied by implicit routing that adjusts promptly.
- *Network heterogeneity*: Positions within the topology are assigned according to individual peer capabilities. In this way, less powerful peers do not slow down others, while peers that are more powerful communicate directly with each other.
- *Quality of service*: Applications can influence the delivery process according to individual requirements.

Self-Organizing Overlay Networks

The first fundamental principle is the creation of independent *virtual overlay networks* to overcome the limits of the peer-to-peer network. To take this idea one step further, an independent overlay network is created dynamically for each group. Figure 1 shows a peer-to-peer network, in which the highlighted peers have subscribed to a topic. Among these connected peers, an additional overlay network (highlighted links in the figure) for the group is constructed. In this way, groups do not interfere with each other, and every virtual network can be organized to consider individual resources and requirements of groups. Another advantage of building a new virtual network is the possibility to consider quality of service parame-

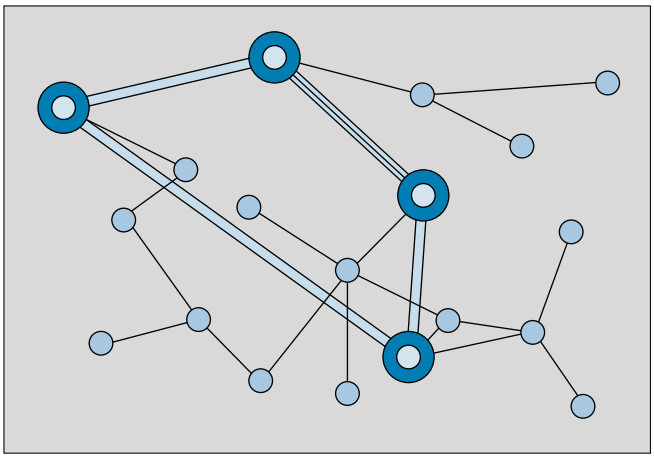

Figure 1. *Peer-to-peer and peer group layers.*

ters. The peer-to-peer network is not replaced, but complemented by multiple group-based overlay networks, which provide an efficient group communication infrastructure.

According to the requirements of peer-to-peer group communication, the multiring topology [3] was designed. There are several reasons why the topology is based on rings. First, rings are scalable in means of bandwidth, because the workload of any node is independent of the total number of nodes. Second, they are easy to manage in a decentralized environment, because a node depends only on its neighbors. Third, they can be made robust easily because of their manageability. Each ring node is connected to two neighbor nodes in order to form a ring. When a node wants to broadcast messages to the other ring nodes, it sends the messages in both directions to its neighbors. Each node receiving messages from a neighbor forwards them to its other neighbor. Internally, arriving messages are put in first in first out (FIFO) message queues until a sender process forwards the messages. In addition, the node keeps track of received messages by storing their unique ID. If a message arrives more than once, the node simply ignores it.

To overcome latency issues that occur in simple rings, the multiring topology forms additional *inner rings* that provide shortcuts to distant ring sections and therefore decrease latency. Inner ring nodes remain also in outer rings (Fig. 2). The formation of inner rings is recursive, as each inner ring may have another inner ring if enough nodes are available. A system parameter q defines the ratio of inner-ring nodes to outer-ring nodes. If this parameter is 10, for example, and the total number of nodes (and most outer ring nodes) is 900, the first inner ring has 90 nodes, and the innermost ring has nine nodes.

Because inner ring nodes forward messages not only to one neighbor, but also to neighbors in inner rings, the bandwidth consumption of these nodes increases. This does not automatically result in negative consequences because we can take advantage of the network heterogeneity. The variety of network nodes with individual capabilities can be exploited by electing nodes with a high bandwidth as inner-ring nodes. Consequently, the mean bandwidth of inner-ring nodes is higher than the ones of outer rings. Because high bandwidth networking systems often imply superior latency, this also suggests a lower mean latency in inner rings. Another effect of the inner ring formation is that powerful nodes communicate directly with each other and are no longer limited by less powerful nodes.

The election process requires keeping track of the bandwidth by measuring the data throughput. Nodes verify their bandwidth periodically as the available bandwidth may vary because of concurrent traffic. After a node has evaluated its bandwidth, the results are compared to the results of neighbor nodes. Assuming a balanced ring with nodes participating in an inner ring every q nodes request the results from its neighbors in both directions within the distance q. This information is gathered by exchanging system messages among the nodes. If a node has superior bandwidth among the evaluated nodes, it is elected as an inner ring node. The membership of inner ring nodes has to be verified frequently. When powerful nodes arrive or the available bandwidth changes due to concurrent network traffic, less capable nodes are abandoned in inner rings. Thus, the inner ring nodes must verify their bandwidth and compare them to others periodically.

To ensure latency scalability, the multiring topology depends on a balanced occurrence of inner ring nodes. If these nodes are unbalanced, the sections that lack inner ring nodes suffer from increased latency. To avoid this situation, the system must balance the inner ring nodes. Each inner ring node determines the distance in the outer ring of its inner ring neighbors. If the distances differ by two or more, the node swaps its position with its neighbor node, which is closer to the more distant inner ring node. This implies that the distances are determined periodically, which is done by exchanging system messages.

Minimizing Dependence on Single Nodes

While a controlled disconnect allows the P2P Messaging System to update links before a node quits, nodes may also quit without warning for any reason. Because this may happen frequently in highly dynamic networks, special precaution must be applied as a node failure may affect other nodes. However, the simplicity of the ring topology allows establishing backup procedures easily. The proposed mechanism is based on backup links, which remain inactive until they are actually needed. Each node has two backup links to the nodes next to its neighbors. When the P2P Messaging System detects broken links to failed nodes, it removes them and activates the backup link. The activated backup link is converted to a regular link, and new backup links are established to restore the ring completely. Backup links are also used when a neighbor node is congested because it did not receive,

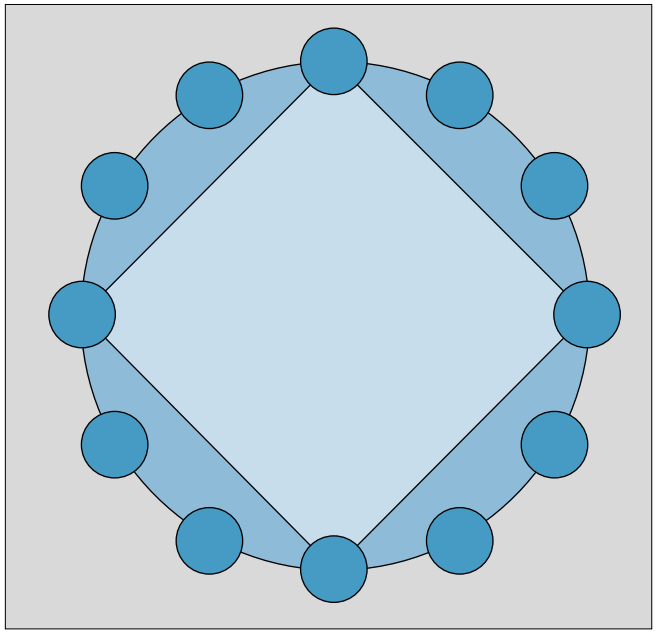

Figure 2. *Outer and inner rings.*

process, or forward the data fast enough. These mechanisms increase robustness and overall throughput.

Implicit Dynamic Routing

Messages may arrive from several sources, especially when inner rings are involved. This improves overall stability because node failures may be compensated for by alternative routes. However, it may also cause wasting network traffic as messages might be delivered to nodes more than once. *Dual mode links* are intended to avoid this scenario without losing the improved stability. The dual mode link mechanism decides whether a message is forwarded immediately or not based on the following policy. The mode of each directed link is considered primary or secondary. Primary links allow immediate sending of messages, while secondary links must first announce the availability of messages. Only if the receiving side needs the announced messages are they actually sent. This simple idea has a global impact on the network as the network dynamically adapts to the message flow by adjusting the mode of its links. Thus, routes that consider efficient message flow implicitly evolve.

Quality of Service

Peer-to-peer networks are unpredictable and may impact the message delivery and thus disrupt applications. *Quality of service (QoS) parameters* can compensate some issues resulting from the dynamic peer-to-peer network, because the application can specify its primary requirements for the message delivery. Despite their importance, peer-to-peer middleware like JXTA barely offer QoS parameters. The P2P Messaging System supports QoS parameters for messages: priority, preservation priority, and expiration time. A key parameter is the priority, which significantly influences the delivery process. The application can further determine messages that should be kept when message queues get full using the preservation priority. Expiration time specifies when a message loses its relevance, and can thus be discarded by the system to reduce network traffic.

Message Protocol

The network abstraction layer allows the P2P Messaging System to use multiple network protocols. For example, peers can use plain TCP adapters for low overhead, or JXTA unicast pipe adapters if NAT traversal and secure communication are required.

On top of the network abstraction layer runs a message-based protocol that takes care of both system and user communication. All exchanged messages have a common header, which is described in Table 1. The minimal header consists of 5 bytes and therefore keeps overhead low.

The message type identifies the message and its content, and falls in either the system (values 0–99) or user category (values 100–254). System category messages are only internally processed and are not forwarded to the user application. Priority and preservation priority are QoS parameters. The Options field allows setting flags to indicate how the message should be forwarded and if the header has extensions. For example, optional extended header fields can be used, which precede the application data. Based on type, length, value (TLV) encoding, the number of extended header fields is unrestricted.

Bits	Field	Description
0..7	Message type	Identifies the message
8..11	Priority	QoS message priority
12..15	Preservation priority	QoS message preservation priority — determines messages to be dropped if queues are full
16..23	Options	Bits 0,1: Forwarding mode Bit 2: Extended length Bit 3: Sender ID and sequence Bit 4: Extended headers fields Bits 5–7: Future use
24..39	Length	Length of message including headers and application data

Table 1. *Header field descriptions.*

Implementation

The P2P Messaging System is implemented in Java to be platform-independent. It is also network-independent, as long as software adapters for networks are available. There is a network adapter for the JXTA peer-to-peer network and an additional implementation that relies on dedicated service providers. The adapter for the JXTA platform collaborates in several ways with the peer-to-peer network. It uses JXTA discovery methods to look up peers in a specific group. If the ring is cut into segments, JXTA's propagate pipes are used to collect peers without neighbors in order to reconnect the ring. The implementation allows the use of JXTA's group authentication facilities to restrict group access. Finally, it benefits from JXTA's pipes as an alternative to the standard TCP links, as these pipes can pass NATs and support security if needed. Despite this extensive collaboration with JXTA, the P2P Messaging System uses its own communication infrastructure with the exception of JXTA pipes.

The P2P Messaging System was compared to the reference implementation of JXTA (version 1.0 released in February 2002). Peer-to-peer networks are highly complex because they are heterogeneous and unlimited in size. In addition, peers are dynamic and unpredictable, as they join and leave frequently. This complexity is difficult to mirror without enormous effort. Instead, we decided to use a local area network to evaluate the two systems. Although this does not reflect the peer-to-peer complexity, it reveals the general efficiency of the systems. The test environment consisted of 13 identical computers, each equipped with a 1.7 GHz Intel Pentium 4 processor, 256 Mbytes RAM, and a 100 Mb Ethernet adapter. The virtual machine was Sun's Java 1.3.1 implementation running in Client Hotspot mode on Microsoft Windows 2000. During the experiments, the message rate was observed depending on the number of receiving peers and message size. One dedicated sender peer constantly sent messages to the group.

The P2P Messaging System constantly provided data rates that were several times higher than those of JXTA's propagate pipe (Fig. 3). The diagram, which is based on the results of 11 receiving peers, reveals that the P2P Messaging System is less dependent on message size and achieves high data rates even with small message sizes. The rate stabilizes with a growing message size because the data rate is limited by the networking hardware. In contrast, JXTA did not reach the limits and always stayed below the network potential. Furthermore, JXTA could not deliver messages at a size of 100,000 bytes.

Figure 4 illustrates the scalability behavior of the two systems during the experiments. Based on the results of testing with three receiving peers, it was measured whether the data

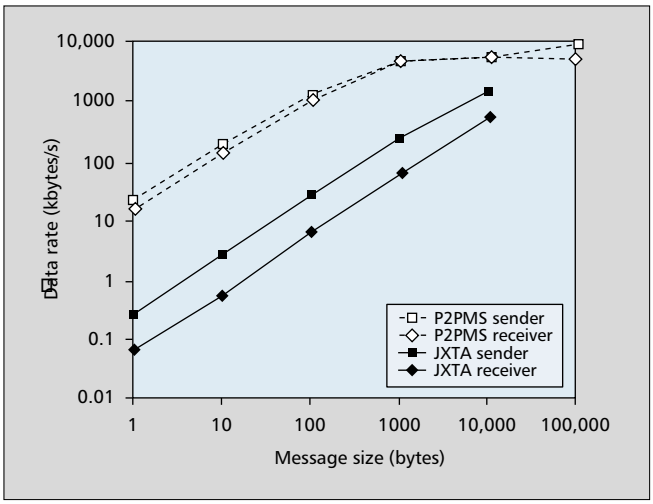

Figure 3. *Data rate depending on message size.*

rate increases or decreases when further peers join the group communication. JXTA's message rate dropped with an increasing number of receivers, which may be a sign for limited scalability. On the other hand, the P2P Messaging System demonstrated its scalable character, as the message rate did not decrease with an increasing number of receivers. This can be explained by the decentralized topology. Unlike systems relying on dedicated servers, the workload is shared among the group participants. Thus, the workload of a typical peer is not affected by growing group sizes. The figure reveals that the results for the P2PMS improved with increasing group size. This result may be unexpected but becomes clearer when one considers that fewer peers cause message collisions because routes are more constant. Further results are presented in [2].

The reason for the contrary results may partially relate to JXTA's propagate pipe principles and implementation issues, which seem to be immature and not yet optimized. This becomes more understandable considering that JXTA is still a new and emerging technology that provides broad functionality. On the contrary, the P2P Messaging System used an infrastructure that was particularly designed and optimized to meet requirements of efficient group communication.

Related Work

While early peer-to-peer research concentrated on file sharing, peer-to-peer is increasingly considered a technology for a variety of applications [4]. Likely, approaches will emerge that combine peer-to-peer networks with group communication. One of these systems [5] merges principles of the distributed lookup service Chord [6] and the content-based notification service Rebeca [7]. Similar systems [8] include Hermes and CovAdv, which run on a simulated peer-to-peer network based on Pastry [1]. Further group communication services that do not directly relate to peer-to-peer systems are discussed in [9].

Scribe [10] and JXTA are more closely related to our P2P Messaging System approach, sharing the notion of explicitly formed groups. Consequently, we discuss these systems in more detail and compare them to the P2P Messaging System.

Scribe from Microsoft Research is a topic-centric publish/subscribe messaging system based on the peer-to-peer platform Pastry [1]. Pastry utilizes routing mechanisms to increase scalability. Each Pastry node stores routing information in a routing table containing information about distant peers and a leaf set containing its direct neighbors. Scribe depends on Pastry to route messages to their destinations. Topics are accessed with a unique ID that consists of the hash of the topic's textual name and its creator's name. Scribe elects the Pastry node whose ID is numerically closest to the topic ID as the topic rendezvous point and root of the multicast tree. Each Scribe subscriber is connected to the root via Pastry connections. A message to be published is forwarded to the multicast tree root using the Pastry connections, or may be sent directly if the root is known to the publisher. Scribe's approach has several disadvantages. Because it relies on Pastry, the group communication is limited by the Peer-to-Peer topology as we pointed out before. Further, the multicasting tree is formed without considering efficiency issues. It imposes a higher workload, especially on the root and the nodes near it, without determining if they are powerful enough. Overloaded nodes may decrease the message delivery efficiency.

JXTA's protocols are available to the public, and are independent of programming languages and platforms. The reference implementation is developed in Java, but other languages like Java Micro Edition, C, Smalltalk, and Perl are supported. In addition to peers, it introduces the concept of rendezvous and relay peers, which supply additional services like forwarding discovery requests and routing. Both rendezvous and relay peers have to be set up explicitly. As JXTA is a general network, it tries to separate peers of a common application into peer groups. Joining a peer group may require an authorization. All resources are described by advertisements, which are based on XML documents published in the network. The communication model is based on message-oriented *pipes* that automatically take care of routing and adjustment to node failures. JXTA supplies several pipe types: unidirectional, bidirectional, propagation, and secure. A propagate pipe is a group communication method that can be used by any number of receivers and senders. It is unreliable, and messages may be lost without any notification. Reliable pipes exist only for unicast.

Although the presented systems provide similar functionality, their implementations differ in many ways. Table 2 gives an overview of their concepts and compares their approaches.

Conclusion

The lack of middleware still complicates developing peer-to-peer applications. After identifying several challenges for

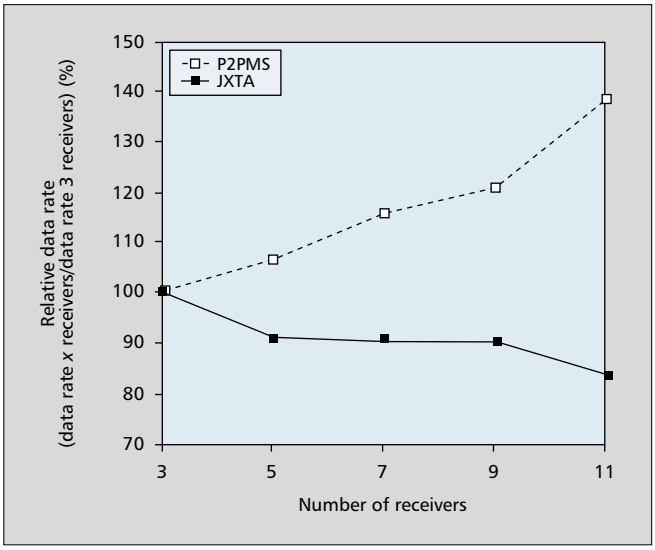

Figure 4. *Scalability: data rate growth/reduction with increasing number of group participants.*

	P2P Messaging System	Pastry/Scribe	JXTA propagate pipe
Group formation	Topic subscription	Topic subscription	Peer group membership, propagate pipe
Node roles	Peers	Peers	Peers, rendezvous peers
Topology	Multi-ring	Tree built on Peer-to-Peer network	Peer-to-peer with dedicated rendezvous peers
Scalability increasing measures	Topology, implicit routing, separation of groups	Topology, routing	Routing, separation of peer groups
Quality of service parameters	Priority, preservation priority, expiration time, delivery mode	None	None
Message filters	Yes	No	No
Multicast messages traverse only nodes in the group	Yes	No	No
Dynamic consideration of individual node characteristics and network heterogeneity	Yes	No	No
Message loss scenarios	Failure of multiple nodes	Failure of a single node	Failure of a single node
Adjustment to node failures	Activation of backup links, restoration of regular links	Local restoration of subscriptions necessary	(Insufficient information available)

■ Table 2. *System comparison.*

peer-to-peer middleware, the P2P Messaging System is presented. Several concepts such as multiring topology, backup links, implicit routing, QoS support, and message protocol are discussed. A self-organizing system is outlined that adapts to a dynamic network. The results of the experimental benchmarks outperformed JXTA, and supported our claims regarding efficiency and scalability. Finally, a comparison revealed that the P2P Messaging System is more suited to the requirements of effective group communication than related systems.

However, some limitations apply to our approach. The infrastructure is complex and consists of several techniques that interrelate with each other. This might not be necessary for all applications. In addition, the maintenance overhead, which is mostly related to the management of inner rings and dual mode links, is not fully analyzed yet. We do not consider the overhead critical, but it might have influence under some circumstances. Lastly, our approach intentionally establishes many links between the network nodes to improve the delivery. This might not be appropriate for all networks and applications. Nevertheless, we are confident that further research and practical tests will help us in refining our concepts.

References

[1] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. IFIP/ACM Int'l. Conf. Distrib. Sys. Platforms (Middleware)*, Heidelberg, Germany, Nov. 2001, pp. 329–50.

[2] M. O. Junginger, "A High Performance Messaging System for Peer-to-Peer Networks," Master's thesis, Univ. of MO-Kansas City, 2003; http://www.junginger.biz/thesis 11

[3] M. O. Junginger and Y. Lee, "The Multi-Ring topology — High-Performance Group Communication in Peer-to-Peer Networks," *Proc. 2nd Int'l. Conf. Peer-to-Peer Comp.*, 2002, pp. 49–56.

[4] A. W. Loo, "The Future of Peer-to-Peer Computing," *Commun. ACM*, vol. 46, no. 9, pp. 57–61.

[5] W.W. Terpstra *et al.*, "A Peer-to-Peer Approach to Content-Based Publish/Subscribe," *2nd Int'l. Wksp. Distrib. Event-Based Sys.*, San Diego, CA, June 8, 2003.

[6] I. Stoica *et al.*, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. Net.*, vol. 11, no. 1, Feb. 2003, pp. 17–32.

[7] G. Mühl, "Large-Scale Content-Based Publish/Subscribe Systems," Ph.D. thesis, Univ. of Technology Darmstadt, 2002.

[8] P. R. Pietzuch and J. Bacon, "Peer-to-Peer Overlay Broker Networks in an Event-Based Middleware," *2nd Int'l. Wksp. Distrib. Event-Based Sys.*, San Diego, CA, June 8, 2003.

[9] A. El-Sayed, V. Roca, and L. Mathy, "A Survey of Proposals for An Alternative Group Communication Service," *IEEE Network*, vol. 17, no. 1, Jan./Feb. 2003, pp. 46–51.

[10] M. Castro *et al.*, "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE JSAC*, vol. 20, no. 8, Oct 2002, pp. 1489–99.

Biographies

MARKUS OLIVER JUNGINGER (markus@junginger.biz) received an M.Sc. degree in computer science from the University of Missouri, Kansas City, and a Dipl.Inf. (FH) degree from the University of Applied Sciences, Augsburg in 2003. Currently, he is a software developer in Munich, Germany. His research interests include decentralized and hybrid networks.

YUGYUNG LEE (leeyu@umkc.edu) is an Assistant Professor at the University of Missouri at Kansas City. She received a B.S. in computer science from the University of Washington in 1990 and a Ph.D. in computer and information sciences from the New Jersey Institute of Technology in 1997. Before joining UMKC, she was working at MCC. Her research interests include distributed intelligent computing and systems, middleware, peer-to-peer networks, pervasive computing, semantic Web, and sensor networks.