A Survey of P2P Middlewares

Atul Singh and Mads Haahr

Distributed Systems Group, Department of Computer Science. Trinity College, Dublin
Atul.Singh@cs.tcd.ie, Mads.Haahr@cs.tcd.ie,
WWW home page: http://www.dsg.cs.tcd.ie

Abstract. This paper examines a range of existing middlewares available for developing Peer-to-Peer (P2P) applications. P2P middlewares provide software components that can be used for rapidly developing P2P applications. P2P applications developed using P2P middlewares build upon tested components and services and hence tend to be more reliable and bug-free. Current middlewares for P2P have different levels of support and use different approaches, for developing P2P applications. The paper examines and analyses the prominent approaches taken by P2P middlewares. Based on the survey of P2P middlewares, the key concerns that a P2P middleware needs to address are identified. The concerns can be used to evaluate the levels of support for P2P application development provided by existing P2P middlewares. The concerns can be used as a starting point for developing a new P2P middleware.

1 Introduction

The term P2P is used to refer to distributed systems without any central control, where all the nodes (called peers) are equivalent in functionality. In a P2P system, peers can collaborate and communicate with each other without a central authority and infrastructure. This self-organizing nature of P2P helps in reducing the management cost of computer infrastructure. The workload can be distributed across the equivalent peers, especially in the case of systems solving Single Process Multiple Data (SPMD) problems [16], to improve the scalability of the system.

Middleware is a term that is difficult to define. A popular definition given by Bernstein [9] defines middleware as programming interfaces and protocols that sit "in the middle", in a layer above the operating system and networking software and below industry specific applications. A middleware provides services that can be used to rapidly develop and deploy distributed applications. CORBA, J2EE, JMS and COM are some popular middleware solutions.

These existing middleware solutions were not designed for P2P computing. The infrastructure provided by the existing middlewares can be used for synchronous/asynchronous communication between peers, but they do not support P2P specific concerns like overlay network management, message multicast and resource discovery. P2P middlewares address these concerns which were not supported in the older middlewares, and provide services which can be used in the

P2P domain. P2P middleware can be developed using existing lower-level middleware solutions like SOAP or CORBA.

This paper evaluates existing P2P middlewares. Based on the study of the middlewares, the paper identifies the concerns that a P2P middleware needs to address. It evaluates the support provided by the existing P2P middlewares for these concerns. The paper examines five P2P middlewares which are representative of the different directions taken by existing P2P middlewares.

The paper is organized as follows. Section 2 reviews the P2P middlewares. Based on the study of the middlewares, section 3 presents the concerns that a P2P middleware needs to address. It also summarizes how the P2P middlewares reviewed in section 2 address these concerns. Section 4 analyzes the prominent approaches for P2P application development, taken by P2P middlewares and suggests possible improvements. Section 5 presents the conclusions of this paper.

2 P2P Middlewares

P2P applications can be divided into three major categories based on the application domain : parallel computing, content-management and collaborative [16]. The paper examines in detail Jabber a P2P middleware for collaborative systems and JXTA a popular general-purpose P2P middleware which can be also used for developing content-management applications. BOINC (Berkeley Open Infrastructure for Network Computing) [7], [8] is a prominent middleware for developing parallel computing applications. While BOINC is sometimes classified as P2P, applications developed using BOINC can not function in the absence of a central server. Because of this strong control exercised by the central server, the authors feel that it is not appropriate to classify BOINC as a platform for developing P2P applications. The authors have not come across a truly P2P middleware designed specifically for developing parallel computing applications.

The paper also examines in detail Speakeasy, AntHill and Pastry which provide interesting paradigms for developing P2P applications. Speakeasy provides a framework which allows P2P applications to increase their capability (including the ability to access new applications at runtime) by using mobile code. Complex Adaptive Systems (CAS) are adaptive to changing environment and resilient to deviant behavior which makes them an ideal solution for the dynamic environment of P2P. AntHill allows P2P applications to use ideas from the CAS world. Pastry can be used for reliable and efficient delivery of messages to a peer on the overlay network using a Distributed Hash Table (DHT) managed by the peers. Section 2.6 skims over other interesting P2P middlewares that can not be examined in detail because of the constraint of space.

2.1 Jabber

Jabber [3] [6] [10] provides specifications for developing instant messaging (IM) applications. Jabber implementations provide a realization of these specifications. Jabber is intended for IM applications, but the infrastructure provided by it can be used for developing other type of P2P applications.

Peers in Jabber can be uniquely identified by a Jabber Identifier (JID). A JID is of the form [peername@]domain[/resource]. The domain name represents the Jabber server to which the peer connects. Jabber enable any two peer on the Internet to exchange XML documents containing communication messages (message document), resource availability information (presence document) or query/response messages (Info/Query (IQ) documents). The structure of these XML documents is defined by Jabber. All the XML documents can be extended to include extra information within a x tag. IQ(Info/Query) documents provide a simple request/response framework within Jabber. It allows peers to pass XML-formatted queries and responses back and forth. A query tag can be used to extend IQ documents.

Jabber uses a hybrid P2P architecture. The Jabber server is used for authenticating peers and resolving JID to a physical network address. However, peers can exchange data directly by establishing a connection which is brokered using Jabber servers.

2.2 JXTA

JXTA [1], [20] was conceived by Sun Microsystems Inc and has been developed as an open source project. JXTA is a specification for developing P2P applications. The specifications can be implemented as a framework which can be used for P2P application development. The Project JXTA reference implementation [2] is one such framework.

The JXTA protocol specifications are divided into three sections. The first section is called JXTA Core Specifications and must be followed by an implementation to be JXTA compliant. An instance of an application which implements the core JXTA protocol is called a peer. Surprisingly, JXTA specifications do not guarantee interoperability among different JXTA implementations [1].

JXTA specifications define a set of six protocols for P2P applications. JXTA protocols work together to perform services required by a peer. The protocols use XML schema to describe the format of the messages exchanged between peers to perform a service. JXTA is policy agnostic and does not specify how the service provided by a protocol will be implemented. The standardization of protocols ensures interoperability between peers.

Peer Resolver Protocol (PRP) defines the structure of XML request and response messages which are exchanged between peers. The rest of the protocol messages are embedded within a PRP message. The client may use Rendezvous Protocol (RVP) to find peers if the message has to be propagated over the overlay. If a direct connection to a destination peer cannot be found then Endpoint Routing Protocol (ERP) is used to find intermediate hosts which can route the information to the destination peer. A peer can advertise and discover resources and services on the overlay network using Peer Discovery Protocol (PDP). Peer Information protocol (PIP), can be used to enquire the status of a peer. JXTA specifies the rules for exchanging messages using existing transport protocols like HTTP, TCP/IP and TLS. The TLS binding can be used for secure transmission of messages.

The JXTA id is a location independent, unique identifier for resources and services in a JXTA network. In the reference implementation, the id is a 128 bit UUID and is generated by the peer. Resources and services in a JXTA network are described using XML documents called advertisements. JXTA specifies XML schemas which define the structure of the advertisements. JXTA peers can dynamically organize themselves into virtual protected domains called Peer Groups. The specifications also define an abstraction called Pipe which is similar to the socket API. A Pipe creates a virtual communication channel like a socket which can be used to send and receive messages between services or applications.

JXTA applications can enhance their capability (both as a server and a client) using modules. A module is a piece of code that can be dynamically loaded and instantiated at a peer to instantiate a new behavior. A module is represented using advertisements.

The JXTA reference implementation uses a super peer architecture. The super peers are used for searching advertisements and for connecting peers that do not have a direct physical connectivity because of firewall, NAT etc. The super peers organize themselves into a loosely-coupled network.

2.3 AntHill

Anthill is a framework for development and evaluation of P2P applications based on ideas borrowed from multi-agent systems (MAS) and Complex Adaptive Systems (CAS) [17] [18]. Anthill uses terminology from ant colony metaphor. An Anthill consists of a network of interconnected nests. Nests are peer entities which share computational and storage resources. Nests provide services to local applications. An application uses the local nest's Nest Interface to request a service and listens for replies.

Services are implemented by entities called ants. An ant is a piece of mobile code. In response to a request one or more ants may be generated. The ants travel through a society of interconnected nests, and use the computational and storage resources of the nests to process a request. Ants may carry with them the request, result and other data. The ant algorithm may be transmitted along with the ant if the destination is not aware of the algorithm. Ants do not communicate directly with each other. Instead they communicate indirectly by leaving information in the appropriate resource manager on the nest.

In Anthill, emergent behavior manifests as swarm intelligence, whereby the collection of simple ants of limited individual capacity achieves intelligent collective behavior. But developing ant algorithms is more of an art than a science. There is no theoretical framework to predict the emergent behavior of an ant algorithm. An ant algorithm is evaluated in the simulation environment provided with Anthill, to test whether the algorithm is suitable for solving a particular problem. The evaluations can be used to improve (evolve) the ant algorithm. Ant algorithms can be evolved only in the simulation environment.

AntHill's technical framework is implemented using JXTA. AntHill provides a set of API's for application development. It comes with an implementations of a nest. Services can be implemented by developing appropriate ant algorithms.

2.4 SpeakEasy

SpeakEasy [12], [21] (available commercially as ObjE) is a technical framework for developing P2P applications which support ad-hoc collaboration. P2P systems do not require a centralized infrastructure which makes them an ideal choice for spontaneous ad-hoc collaboration. However the current generation of P2P applications like Jabber have not exploited the spontaneous ad hoc collaboration aspect of P2P networking. Current P2P applications are constrained to use only a limited types of shared resources, such as files or processing power. P2P applications developed using SpeakEasy can use new devices and services on the network without having any special knowledge about them. Applications access the devices and services using the mobile-code supplied by them. The mobile code can also be used by the aplication to add the capability to provide a new service.

In SpeakEasy any entity that can be accessed over a network (eg, printer, file URL) is cast as a component. Components provide discrete elements of functionality and are used by applications or other components. All components expose their functionality through four interfaces specified by SpeakEasy. The interfaces are: Data Transfer, Aggregation, Context and Control. Components implement one or all of these four interfaces. The interfaces can be used to exchange data including mobile code, get metadata information about the components and retrieve a user interface which can be used to access the components.

Speakeasy provides a security layer. The security layer ensures that all communications are encrypted and authenticated. Authentication is mutual between client and server. The service provider makes the final decision about access and the decision is made on the machine running the service. The access decision are determined by the semantics of the application. So the security layer calls an application provided security manager.

Discovery of components is not a part of SpeakEasy. SpeakEasy has an application called Casca, which provides framework and user interface for component discovery.

2.5 Pastry and Scribe

Pastry [19] is a decentralized object location and routing sub-system for P2P applications. Pastry provides an API which can be used to develop P2P applications. Each node in the pastry network is assigned a unique node id. When presented with a message and an id, Pastry efficiently routes the message to a peer with the node id that is numerically closest to the key, among all currently live pastry nodes. The expected number of routing steps is $O(log_{2^b} N)$ where N is the number of peers in the network and b is a constant. Eventual delivery of a message is guaranteed in Pastry. Pastry chooses a route which is likely to be "good" with respect to proximity metrics. Pastry expects each application to implement a function which determine the distance of a Pastry peer with a given IP address to itself. When a peer has a choice of addresses to which it can forward a message then it chooses the address which is closer to it.

Pastry is self-organizing. New nodes joining a pastry network send a join message with a key to the network through a boot strap node obtained through external means. The join message is routed to the numerically closest node. All the nodes in the route path of the message respond with their state tables (used for routing messages) and the information is used to populate the state tables of the new node.

Pastry can be used to quickly and deterministically locate an object whose node id is available, but pastry can not be used to multicast messages on the overlay to search for an object matching a search criterion. Scribe [5] [13] is an application level multicast infrastructure build on top of Pastry. It can be used to send messages (including search queries) to all the nodes in a Scribe group.

2.6 Others

XNap [4] is an open source plugin enabled framework for P2P application development. The plugins are developed using XNap API and can be used to develop a P2P application. The plugins are loaded into the XNap client application which provides a GUI based frontend for the application. XNap plugins are available for popular P2P networks like Gnutella and Overnet. Windows P2P Networking [14] from Microsoft is a freely available general-purpose P2P middleware from Microsoft. It is the only middleware which provides support for handling topology related issues such as partitions in the overlay network and removing a peers less useful connections. Groove [11] is a P2P based proprietary collaboration solution. It can also be used for developing collaborative P2P applications. JINI [15] is an architecture for developing distributed systems. It can also be used for developing P2P applications which use mobile code to access services on the network.

3 Concerns

Based on the study of the existing P2P middlewares the authors have identified the concerns that a P2P middleware needs to address. The concerns can be divided into five groups. Table 1 gives an overview of these concerns. The concerns are the use-cases that a P2P middleware needs to address. The concerns can be used as a starting point for developing the architecture of a P2P middleware. The concerns can also be used for a comparative evaluating of existing P2P middlewares. The groups along with the concerns belonging to them are discussed below. Table 2 presents a synoptic view of how some of these concerns are handled by the P2P middlewares discussed earlier.

3.1 Naming

The concerns belonging to this group are: assigning identification to the entities in a P2P system (*identification*) and resolving (*resolution*) an entity's identification to its physical network address. P2P systems assign a network (underlying

Group	Concern
Naming	Identification of entities in a P2P system.
	Resolution of entity id to a physical address.
Overlay Management	Search for a connected peer on the overlay network.
	Handle join requests.
Service Management	Advertise services/resources to share.
	Discovery of shared services/resources.
	Access the service/resources.
Message Routing	Maintain the topology.
	Routing of messages.
Security	Authentication of peers
	Authorization of peers to access a resource/service.
	Secure transmission of messages.

Table 1. The concerns that a P2P middleware needs to address.

physical network) and location independent id to the entities in a system. The id's allow P2P systems to take care of entities whose physical location (IP address) on the network changes with time. The id's are dynamically resolved to determine the current physical location of the entity.

Jabber, JXTA and Pastry have their own rules for assigning id's to entities. Jabber uses the domain server to resolve the physical location of the entity. Pastry maintains a Distributed Hash Table (DHT) which can be used to locate the entity with a given id. JXTA uses a combination of a central server (called rendezvous server) and IP multicast to locate the physical address.

3.2 Overlay Network Management

The concerns in this group are: search (*search*) for an existing overlay network to join and handling of requests to connect to the network (*join*). Peers join an overlay network through another peer which is already connected to the overlay network. Connected peer's address may be a well known information or it can be obtained by an IP multicast. The peer accepting the connection may redirect the incoming peer to a new peer. The connecting peer may also be supplied with information relevant for it.

In hybrid solutions like Jabber peers connect to a well know server to join the overlay network. JXTA uses the combination of a well known server (rendezvous server) and IP multicast to locate the connected peer's address. In pastry the connected peer's address is an information which has to be obtained from out-of-band sources.

The connecting peer in Pastry receives information which it can use to populate its routing table. In Jabber the connecting peer receives presence information about other peers.

3.3 Service/Resource Management

The concerns in this category are: advertising (*advertise*) the services/resources that the peer shares to the overlay network, discovering (*discovery*) the services/resources to use on the network and accessing (*Access*) the service/resources. A peer uses an overlay network to advertise its resources and services and to discover new resources and services which it can access.

P2P systems generally maintain a directory of services and resources. A peer can publish the details of the resources and services it is offering to the directory. The directory could be maintained on one (eg. Jabber), more than one or all the peers (eg. casca). Alternatively the directory can be distributed across some (eg. JXTA) or all the peers in the network. The peers maintaining the directory can be chosen (eg. through an algorithm), or they can be specialized peers responsible for maintaining the directory. Peers can discover the details about the shared services/resources by sending a query message to the peers responsible for maintaining a directory. Peers may also send a multicast message to all the peers on the network to find the details about a service/resource available on the network. The service/resources can be accessed by using mobile code (eg JXTA, SpeakEasy) or by doing a remote process call (RPC). Middlewares using RPC (eg. JXTA, Jabber, pastry) may define the structure and semantics of the messages exchanged. Middlewares using mobile code allow the peers to dynamically extend their capability (both as a client and a server).

3.4 Message Routing

The concerns related to routing are: manage the connection to other peers (*topology*) and routing the messages (*routing*). A peer can directly send the message to the destination peer if the destination peer's address is available. If the destination peer is not accessible or its physical address is not available then the source peer may route the message to one or more known peers on the network which can route the message to the destination peer. The intermediate peers can be chosen from the known peers randomly or by using an algorithm (eg pastry), or they might be well know specialized peers (eg peers in JXTA for routing messages to peers inaccessible because of firewalls).

Peers in decentralized P2P systems maintain connection to other peers in the network. The graph formed by these connections makes the topology of the overlay network. The connections are used to route messages. New connections are established when the old connections fail. The peers for connection may be selected randomly or by using an algorithm (eg pastry).

3.5 Security

The concerns related to security are: authentication of peers to ensure the identity of a peer (*authentication*), checking the authorization of a peer to access a resource/service (*authorization*), and secure transmission of message (*security*). Authentication and authorization can be mutual between the interacting peers (eg SpeakEasy) or a central peer can be used (eg Jabber).

Features	Jabber	Speakeasy	JXTA	Pastry
search	well-known server	through invitation	IP multicast, well-known server	Out of band
join (information received by joining peer)	presence information	resources shared by others	none	other peers state tables
advertise (directory maintained on)	well-known server	all peers in a shared space	well-known server, IP multicast	none
access	messages defined by Jabber	mobile-code	messages defined by JXTA, mobile-code	messages

Table 2. This table describes how some of the concerns are addressed by the P2P middleware described in this paper. AntHill is implemented using JXTA and hence is not presented in this table.

4 Discussion

CAS systems are adaptive to changing environment conditions and are resilient to deviant behavior. Ideas used in CAS systems can be extremely useful in P2P systems which have a dynamic environment with a constant flux of nodes. AntHill builds upon JXTA to provides a mobile-code based approach for developing P2P applications inspired by ideas from CAS.

The ability to extend the ability of the peers so that they can offer and access new services that they were not programmed to is useful for peers especially in ad-hoc collaboration scenarios. Existing P2P middlewares like JXTA, SpeakEasy and Jini provide this ability using mobile code. However mobile code is inherently untrustworthy which limits this approach to trustworthy environments. In untrustworthy environments using a dynamic invocation approach (like DII in CORBA) is more appropriate to access a service the peer was not programmed to access. To the authors knowledge only JXTA supports this approach partially. Module specification advertisement can contain details necessary to invoke a service, written in a low-level middleware like CORBA or SOAP, using a dynamic invocation approach. Peers can use the information in the module-specification advertisement to access the new service dynamically.

Instead of building upon a low-level middleware, P2P middlewares like JXTA and Jabber reinvent the wheel for messaging . JXTA and Jabber specify the format of the messages exchanged between the peers and the transport bindings that can be used to exchange the messages using protocols like HTTP and TCP/IP. Type safe invocation which is standard in most of the low-level middlewares is not provided by these P2P middlewares. Lack of type safe invocations

can lead to bugs that are difficult to track. JXTA and Jabber lack a standard format like CORBA IDL or WSDL for describing services offered by other peers. This makes it difficult to program peers to access services offered by others. A P2P middleware build upon a low-level middleware will benefit a lot by using the mature messaging infrastructure provided by the low-level middlewares.

Most existing P2P middlewares lack support for topology management tasks such as handling partitions and ensuring optimal grouping of peers. To the authors knowledge Windows P2P networking is the only P2P middleware that provides support for detecting and handling overlay network partitions. However none of the existing P2P middlewares provide support for controlling the type of peers which are connected together. This may lead to a suboptimal topology of peers. For example, in a file-sharing application, peers with high bandwidth capacity may be connected together with peers with low bandwidth capacity, which may lead to a degradation in performance. Also peers which are geographically distant (in terms of the underlying network infrastructure) may be connected together, increasing the cost of communication between them.

Interoperability between applications developed using different P2P middlewares will be a major challenge for future P2P middleware developers. JXTA specifications were a step in this direction. By standardizing the structure of the messages exchanged between peers to perform different services, JXTA expected to provide interoperability among peers. But JXTA is still a developing specification and implementing it does not guarantee interoperability among peers.

5 Conclusions

The paper has reviewed the state of the art in P2P middlewares by discussing five prominent P2P middlewares, which give an overview of the state of P2P middlewares. Based on the study of the middlewares, the paper presented the concerns that a P2P middleware needs to address. The paper also discussed the directions that P2P middlewares are taking.

References

1. Jxta v2.0 protocols specification. http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.html.
2. Project jxta 2.0 superpeer virtual network. http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf.
3. What is jabber? http://www.jabber.org/about/overview.php.
4. Xnap. http://www.xnap.org.
5. M. Castro A. Rowstron, A-M. Kermarrec and P. Druschel. Scribe: The design of a large-scale event notification infrastructure. *NGC2001, UCL, London*, 2001.
6. DJ Adams. *Programming Jabber*. O'Reilly and Associates, 2002.
7. D. P. Anderson. Public computing: Reconnecting people to science.
8. D.P. Anderson. Boinc: A system for public-resource computing and storage. *IEEE/ACM International Workshop on Grid Computing*, (5), 2004.
9. Philip A. Bernstein. Middleware: A model for distributed system services. *Communications of the ACM*, (39(2)):86–98, 1996.

10. Cathreine Dodson. Jabber technical white paper. http://xml.coverpages.org/, Aug 2000.
11. James Edwards. *Peer-to-Peer Programming On Groove*. Addison-Wesley, 2002.
12. W. Keith Edwards et. al. Using speakeasy for ad hoc peer to peer collaboration. November 2002.
13. A-M. Kermarrec M. Castro, P. Druschel and A. Rowstron. Scribe: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications)*, 2002.
14. Microsoft. Introduction to windows peer-to-peer networking. Technical report, Microsoft Corporation, 2003.
15. Sun Microsystems. Jini technology architectural overview. Technical report, Sun Microsystems, 1999.
16. Dejan S et. al. Milojicic. Peer-to-peer computing. Technical report, HP Labs, 2002.
17. Alberto Montresor. Anthill: a framework for the design and the analysis of peer-to-peer systems. *Proceedings of the 4th European Research Seminar on Advances in Distributed Systems, Bertinoro, Italy*, May 2001.
18. Hein Meling Ozalp Baboglu and Alberto Montresor. Anthill: A framework for the development of agent based peer-to-peer systems. *ICDCS2002*, 2002.
19. A. Rowstron and P. Druschel. Scalable pastry scalable, decentralized object location and routing for large-scale peer-to-peer systems. *Proceddings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware 2001)*, 2001.
20. Bernard Traversat Scott Oaks and Li Gong. *JXTA In a Nutshell*. O'Reilly and Associates Inc., 2002.
21. Mark W. Newman W Keith Edwards and Jana Z. Sedivy. The case for recombinant computing. Technical report, Xerox Palo Alto Research Center, 2001.