

P2P Computing System with Remote Method Invocation over JXTA

Pawel Jurczyk¹, Maciej Golenia¹, Maciej Malawski¹, Dawid Kurzyniec²,
Marian Bubak^{1,3} and Vaidy S. Sunderam²

¹ Institute of Computer Science, AGH, Mickiewicza 30, 30-059 Kraków, Poland

² Emory University, Atlanta, USA

³ Academic Computer Centre – CYFRONET, Nawojki 11,30-950 Kraków, Poland

Abstract. In this paper we present a P2P system with RMI over JXTA which was developed on top of the H2O distributed resource sharing platform. Since RMIX is an underlying communication substrate of H2O, there was a need to adapt RMIX to run over JXTA virtual network. We show that this integration was possible due to extensibility of RMIX which allowed to use JXTA-provided socket implementation. The result of this integration is the fully operational RMI implementation running over JXTA P2P network, where methods can be invoked on remote objects located behind firewalls or NATs. We present results of tests showing that our implementation can be used to connect peers in different LANs that cannot interact directly, while in the case of direct connection the performance is comparable to that of using standard sockets.

Keywords distributed computing, P2P systems, JXTA, remote method invocation, H2O system

1 Introduction

Scientists or companies are looking for new ways of accessing the computational power needed for solving large-scale computing problems. This issue is addressed by Grid systems development which may provide middleware for running applications on distributed resources shared between many institutions. On the other hand, there is also a huge potential computing power located in the millions of computers connected to the Internet. These resources can be used in a Peer-to-Peer (P2P) fashion which was shown on the example of SETI@home [1] project. The common feature of Grid and P2P systems is the resource sharing and distributed nature of the environment. However, it requires large administrative effort for setting up the Grid infrastructure and establishing virtual organization security mechanisms. On the other hand, P2P systems are more suitable for ad-hoc formed collaborations which allow for more dynamic participation than in Grid systems.

Avoiding the administrative burden related to using Grid systems was one of the goals of H2O [2] resource sharing platform. H2O proposes and implements the model, where the roles of resource providers, service deployers and users can be separated. In this model, the providers can independently offer the CPU

power of their machines by running H2O kernels, but the process of deployment (installation) of services (called pluglets) is left to others. This makes the sharing of resources easier for providers, which is important in P2P model. As an important part, H2O uses RMIX, which is an extensible RMI-based communication framework, thus offering the RMI programming model for H2O-based distributed applications.

In order to enhance the H2O with the ability to run in a P2P environment, we are working on integrating H2O with JXTA P2P system. The goal of this work is to use JXTA mechanisms for enabling resource sharing among peers which may be hidden behind NAT or firewalls, and which may dynamically join and leave the P2P network, possibly changing their locations. There are two main tasks needed to approach this goal: (1) enabling communication in P2P environment and (2) resource discovery in P2P network. In our previous paper [3], we presented the basic concepts of our solution. In this paper, we describe the integration of RMIX with JXTA, which brings the RMI programming model to P2P systems and allows running H2O over the virtual JXTA network.

This paper is organized as follows: Firstly, we give the background on RMI as a programming model for distributed computing and RMIX as a specific extension. Next, we overview the P2P technologies and JXTA as a proposed standard. Then we describe our work on integrating RMIX with JXTA and we present the results of preliminary tests.

2 RMI as a successful programming model and RMIX

The diversity of distributed programming approaches results in development of many distributed computing infrastructures. In recent time one can observe a development of systems which map function calls to the network – remote procedure call infrastructure with its XML-RPC [4] implementation. The second group are systems with distributed objects such as CORBA [5] or Java RMI [6].

The RMI model that allows one to use remote objects as if they were instances created locally is very comfortable to use. Although standard Java RMI implementation is based on the Java Remote Method Protocol (JRMP) that is sophisticated and full-featured, it is limited to pure Java systems. There are some other RMI implementations based on other protocols (e.g. RMI-IIOP [7] that enable connectivity with CORBA or JAX-RPC [8] using SOAP/HTTP to provide Web services connectivity) but usually these solutions have decreased functionality. In these circumstances the RMIX project [9] has been initiated.

The main objective of the RMIX communication library is to provide the possibility of using various RMI protocol service providers within a single, RMI paradigm based framework. The RMIX framework is easy to extend and it permits to integrate existing RMI implementations in a very simple way. Moreover, provider modules can be managed on a dynamic basis. Additionally, RMIX features several general-purpose enhancements over Java RMI, including dynamic stubs, SSL support, runtime binding and customizable virtual endpoints that allow the same remote object to be accessed via different protocols such as

SOAP, JRMP, SunRPC. The framework has a possibility of dynamic control of access policies - the interceptor allows or denies method invocation depending on custom, changing criteria. RMIX provides a set of new features in method invocation model providing an asynchronous calls and a one-way calls [10].

What is important from the P2P environment point of view, the RMIX communication library is based on the Java RMI model, which allows to use various socket factories. This mechanism can be used for plugging in custom socket implementations. Moreover, RMIX provides its own socket factory interfaces supporting non-IP addressing type. This makes it possible to move the framework into any type of network, e.g. the P2P overlay with its flat addressing scheme.

3 P2P networks and JXTA P2P network implementation

One of the first P2P applications was Napster [11], used for file sharing purposes and representing a first-generation P2P network. Next generations tried to avoid central servers and provided mechanisms for distributed search (Gnutella [12]), also introducing the concept of peer hubs for more efficient operation (Morpheus [13]).

The idea of exchanging files in a Peer-to-Peer manner inspired people to exchange free CPU cycles and resources. Today there are millions of computers that could be linked and used as one big supercomputer. Good examples are SETI@home [1], GPU Project [14], JNGI [15] or Parabon [16].

There are many libraries allowing to create custom Peer-to-Peer systems but most of them are platform and protocol dependent. They are specialized for specific types of networks for example file sharing. They do not offer all of interesting features in one consistent implementation. That is why JXTA [17] was introduced. It was designed to be a set of open, language independent protocols that allows any connected devices from cell phone, PDA, notebook to big server to communicate with each other as peers [18]. Currently, there are implementations of JXTA in PERL, C and, of course, in Java.

JXTA brings an abstraction of a peer. When a peer connects to the network, it publishes a network interface (JXTA Endpoint) that allows it to communicate with other hosts. Peers can gather information about network, available resources and services. They may also form PeerGroups which gather peers that want to serve a set of same services. Peers may belong to more than one group at a time.

One of the most important features of the JXTA are its communication capabilities. Peers can create communication channels described as pipes, which are used in the JXTA network to send messages between peers. These channels do not need a direct connection between peers. If peers creating a pipe connection are behind a firewall, they just need to use a special peer that automatically helps to propagate messages through firewalls. Pipe messages can carry any type of data, for example text messages, binary data or even Java objects. In the Java implementation of JXTA, pipes are unidirectional. Peer endpoints use any network interfaces that is available, therefore when peers are in the

same subnetwork, pipes may use direct TCP connection that greatly increases efficiency.

JXTA pipes are very useful but they do not offer an abstraction similar to well-known and frequently used sockets. That is why in later versions of JXTA on the top of pipes, socket abstraction was introduced. JXTA Sockets share an identical API with standard Java sockets.

4 Advantages of a P2P system for distributed computing

Distributed resource sharing systems like H2O can draw many advantages from using a P2P RMI framework. This will allow a client from any local network to interact with any other resource from other private networks, hidden behind firewalls or NATs. This is achieved by using flat addressing type in P2P systems. Moreover, object address is independent from host IP address. This gives the possibility of sharing a specified resource using the same address independently from its location. These new features in P2P distributed application framework will yield new possibilities for building global computing systems:

- simplicity in building of distributed application (no need of specialized configuration of routers or firewalls),
- wider distributed application range (users from private networks can participate in any distributed application),
- clients can use the same resource independently from its location,
- ad-hoc collaboration of shared resources - virtual computing groups (using peer groups in P2P network).

RMIX communication library is the best candidate to be moved into the P2P environment. On the other hand, we have presented a JXTA P2P network implementation. JXTA is not only easy to use, but also provides JXTA Sockets which are Java socket abstraction in the P2P environment. Therefore, integration of both systems can be easy and can bring all the advantages presented above. As a result, our distributed application framework will be based on the integration of RMIX with JXTA technology.

5 Integration of RMIX with JXTA

The integration of JXTA and RMIX will be based on the implementation of client socket factory and server socket factory modules for RMIX communication library. Thus, by using JXTA Sockets abstraction in the mentioned socket factories, RMIX framework will be extended to the P2P network.

To denote RMIX endpoint address in the P2P environment we have decided to use the following addressing model:

```
<string endpoint address>[@<group>[(param-1;param-2;...;param-n)]]
```

As can be seen, an address can be a simple string such as *rmixEndpoint*. Then, an address denotes the endpoint in *NetPeerGroup*. Additionally, there are two

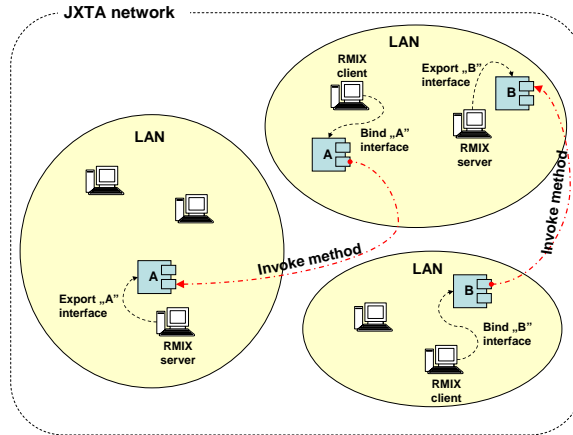
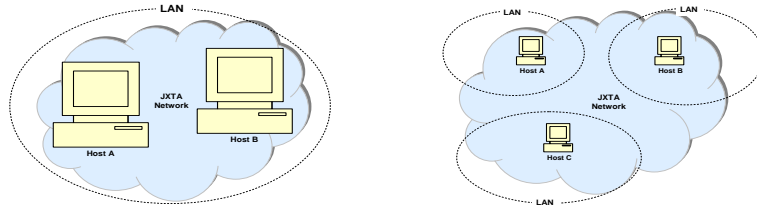


Fig. 1. RMI communication library in P2P environment.

optional parts - the first of them defines a custom group in which endpoint will be created and the second is used to provide additional parameters used while joining the group. This offers flexibility in group security controlling - as the JXTA can use custom implementations of Membership Service. In result, address *myEndpoint@cGroup* denotes an RMI endpoint with address *myEndpoint* that is located in the *cGroup* JXTA group, with no security control. When one would like to use group with security control, a valid address of RMI endpoint can be *myEndpoint@cGroup(groupInterfaceImpl;p1;p2)*. Here, when joining *cGroup*, the custom security control defined in *groupInterfaceImpl* implementation of our *JxtaGroupInterface* interface will be used.

The result of our work is that, when the user decides to use either *JXTAServerSocketFactory* or *JXTAClientSocketFactory*, he or she provides only a string representation of the JXTA address (which fulfills the model introduced above) and JXTA Socket Factories manage all the JXTA functionality of the system. Thus, it automatically connects to the network (becoming a JXTA peer), joins a group (if needed), manages rendezvous status of the current JXTA peer or connects to the JXTA socket specified by provided address. There is no need for additional steps, all the JXTA connectivity is encapsulated inside the RMI and using JXTA Socket Factories is transparent from the user's point of view.

The solution presented in this section gives a possibility of using the RMI communication library in P2P environment (as shown on Fig. 1). In our opinion, it can bring new possibilities for developing distributed applications by providing a simple remote method invocation model that can be used across any firewalls, NATs or private networks.



(a) Configuration that uses only a Local Area Network (b) Configuration that uses the JXTA network spanning several Local Area Networks

Fig. 2. Test configurations of the JXTA Socket Factories

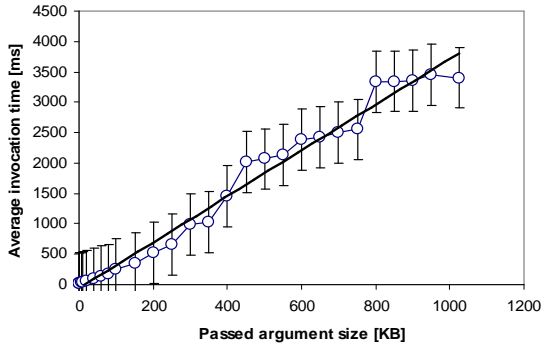
6 Tests of RMIX framework in P2P environment

We have used a simple benchmark to measure the performance of the RMIX communication framework in the P2P environment. We have decided to measure calls of the remote method that takes one attribute being table of bytes and returns a String object. The test application has been developed to allow one to invoke methods for various sizes of parameter. For each table size we called the method 50 times and below we will present the results we obtained. The measurement was performed for two JXTA network configurations, presented in Fig. 2.

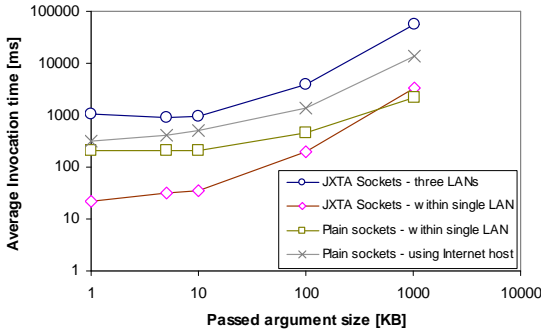
As can be seen, in the first test configuration (refer to Fig. 2(a)) JXTA network connects two computers which are in the same LAN. In this situation JXTA Sockets should use direct connections between peers. The second test configuration assumes that JXTA spans three LANs. In one LAN there is a JXTA rendezvous peer, in other Local Networks there are two other peers - the first of them will be arranged to be the server for our test application and the second will be the client. Additionally, to compare JXTA Socket performance with plain socket performance, we have decided to run our test for RMIX plain socket factories. All the results we have obtained are presented above.

First, we have tested JXTA Socket performance when JXTA was used inside a single LAN (the configuration of Fig. 2(a)). Basing on the results, we have prepared the chart presented in Fig. 3(a). As can be seen, invocation time increases linearly with the increase of the table length that is passed as function attribute.

Next, we have compared results between both configurations presented in Fig. 2. Additionally, in the comparison we have enclosed invocation times when using plain RMIX socket factories in two configurations: the first one is for two hosts inside a single Local Area Network, the second involves a host within a LAN connecting to a server that is somewhere on the Internet. Note that the chart presented in Fig. 3(b) has logarithmic scales; the reason is that the results can be presented more clearly in this way.



(a) Single Local Area Network.



(b) Different network configurations

Fig. 3. Test results of RMIX using JXTA.

7 Summary

In this paper we discussed the role of a P2P-enabled RMIX communication library in the context of distributed resource sharing in the H2O platform. The need for a generic platform which allows running arbitrary code on resources available within a P2P network can be satisfied by building it on H2O with the help of JXTA technology. Since RMIX is an underlying communication substrate of H2O, there is a need to adapt RMIX to run over the JXTA virtual network. We have shown that this integration is possible because of the extensibility of RMIX which allows using the JXTA-provided socket implementation. The result of this integration is the fully operational RMI implementation running over JXTA P2P network, where methods can be invoked on remote objects located behind firewalls or NATs, which is not possible in traditional RMI systems. The results of tests show that our implementation can be used to connect peers in

As one can expect, when passing large size arguments, JXTA Sockets are slower than plain sockets even within a single LAN. The reason is that despite using direct socket communication, there is still overhead induced by pipes. What can be surprising, for smaller data sizes, JXTA Sockets are significantly faster than plain sockets. The reason is the Nagle Algorithm enabled in RMIX when plain sockets factories are used, while JXTA Sockets do not have this time delay. Additionally, the invocation times between peers inside the single LAN are significantly shorter than times measured over the JXTA network spanning three LANs. This is the effect of using direct socket connections between peers in a single LAN. When JXTA spans different LANs, there is no such possibility, thus invocation times are much longer.

diferent LANs that cannot interact directly, while in the case of direct connection the performance is comparable to that of standard sockets.

Our RMI over JXTA implementation is used as a basis for enabling H2O resource sharing in P2P environment. Currently it is possible for H2O kernels to be accessible using JXTA endpoints, thus allowing resource providers, service deployers and users to operate in a P2P environment. Our current work is focused on integration of automatic discovery of H2O kernels with the P2P discovery mechanisms. This will result in a powerful general-purpose distributed computing platform running in a P2P environment.

Acknowledgments This research is partly funded by the EU IST Project Core-GRID and the Polish State Committee for Scientific Research SPUB-M grant. The authors are grateful to Piotr Nowakowski for his remarks.

References

1. SETI@home: Search for extraterrestrial intelligence at home (2003) <http://setiathome.ssl.berkeley.edu/>.
2. Kurzyniec, D., Wrzosek, T., Drzewiecki, D., Sunderam, V.: Towards self-organizing distributed computing frameworks: The H2O approach. *Parallel Processing Letters* **13**(2) (2003) 273–290
3. Jurczyk, P., Golenia, M., Malawski, M., Kurzyniec, D., Bubak, M., Sunderam, V.S.: A system for distributed computing based on H2O and JXTA. In: *Cracow Grid Workshop, CGW'04, December 13–15, 2004, Kraków, Poland* (2005) 257–268
4. Scripting News, Inc.: XML-RPC Home Page (2005) <http://www.xmlrpc.com/>.
5. Object Management Group, Inc.: CORBA (2005) <http://www.corba.org/>.
6. Sun Microsystems, I.: Java Remote Method Invocation (2005) <http://java.sun.com/products/jdk/rmi/>.
7. Sun Microsystems, Inc.: Java RMI over IIOP (2005) <http://java.sun.com/products/rmi-iiop/>.
8. Sun Microsystems, Inc.: Java API for XML-Based RPC (JAX-RPC) (2005) <http://java.sun.com/xml/jaxrpc/>.
9. Kurzyniec, D., Wrzosek, T., Sunderam, V., Słomiński, A.: RMIX: A multiprotocol RMI framework for java. In: *Proc. of the Intl. Parallel and Distributed Processing Symposium (IPDPS'03), Nice, France, IEEE Computer Society* (2003) 140–146
10. Kurzyniec, D., Sunderam, V.S.: Semantic aspects of asynchronous RMI: The RMIX approach. In: *Proc. of 6th International Workshop on Java for Parallel and Distributed Computing, in conjunction with IPDPS 2004, Santa Fe, New Mexico, USA, IEEE Computer Society* (2004)
11. Napster, LLC.: NAPSTER Music Service (2005) <http://www.napster.com>.
12. OSMB, LLC: Gnutella File Sharing Network (2001) <http://www.gnutella.org>.
13. StreamCast Networks, Inc.: Morpheus File Sharing System (2005) <http://www.morpheus.com/>.
14. GPU Team: GPU project (2005) <http://gpu.sourceforge.net/>.
15. Verbeke, J., Nadgir, N., Ruetsch, G., Sharapov, I.: Framework for peer-to-peer distributed computing in a heterogeneous, decentralized environment. *Lecture Notes in Computer Science* **2536** (2002) 1–12
16. Parabon Computation, Inc.: Parabon distributed computing platform (2005) <http://www.parabon.com/index.jsp>.

17. CollabNet, Inc.: JXTA Home Page (2005) <http://www.jxta.org/>.
18. Sun Microsystems: JXTA v2.3.x: Java Programmers Guide. (2005)
http://www.jxta.org/docs/JxtaProgGuide_v2.3.pdf.