# Enabling Peer-to-Peer Interactions for Scientific Applications on the Grid*

Vincent Matossian and Manish Parashar

The Applied Software Systems Laboratory
Department of Electrical and Computer Engineering
Rutgers University, Piscataway NJ 08855, USA
{vincentm,parashar}@caip.rutgers.edu

**Abstract.** P2P and Grid communities are actively working on deploying and standardizing infrastructure, protocols, and mechanisms, to support decentralized interactions across distributed resources. Such an infrastructure will enable new classes of applications based on continuous, seamless and secure interactions, where the application components, Grid services, resources and data interact as peers. This paper presents the design, implementation and evaluation of a peer-to-peer messaging framework that builds on the JXTA protocols to support the interaction and associated messaging semantics required by these applications.

## 1 Introduction

The emergence of Grid applications coincides with that of Peer-to-Peer (P2P) applications such as Napster or SETI@HOME[1]. This parallel is manifested in the similarities in infrastructural requirements of Grid and P2P systems, such as the underlying decentralized (overlay) network architecture, the dynamic discovery of resources, the aggregation of distributed resources, or the need for system integrity and security guarantees. A key requirement for enabling the application-level interactions on the Grid is a P2P messaging layer that supports the interactions and their associated messaging semantics. The overall objective of this work is to prototype such a messaging framework to support Grid applications in a P2P environment. A number of messaging solutions for P2P/Grid systems have been proposed in recent years. These include Message Oriented Middleware (MOM) systems such as JMS [2], LeSubscribe [3], and IBM MQSeries[4], as well as Grid oriented systems such as GridRPC [5], NaradaBrokering [6], and ICENI[7]. In this paper we present Pawn, a publisher/subscriber messaging substrate that offers interaction services for distributed object management, monitoring and steering, group formation, and collaboration through guaranteed, flexible, and stateful messaging.

* Support for this work was provided by the NSF via grants numbers ACI 9984357 (CAREERS), EIA 0103674 (NGS) and EIA-0120934 (ITR), DOE ASCI/ASAP (Caltech) via grant numbers PC295251 and 1052856.

Pawn combines properties from messaging and P2P messaging on the Grid to provide publisher/subscriber functionalities as well as advanced messaging semantics such as guaranteed message delivery, push, pull, transactions, and request/response interaction modalities, support for synchronous and asynchronous communications, coordination through message ordering, and remote procedure calls. Unlike other publisher/subscriber systems, Pawn focuses on interaction services to support application monitoring and steering, collaboration, and application execution on the Grid. This paper makes three contributions: (1) the definition of messaging requirements for scientific investigation in a P2P Grid environment, (2) the identification and implementation of corresponding services and mechanisms in Pawn, (3) the deployment of the Pawn messaging substrate and its evaluation using a "real-world" Grid application.

The rest of this paper is organized as follows. Section 2 presents the motivating application process and describes the underlying interactions. Section 3 presents the design and implementation of Pawn. Section 4 describes the use of Pawn to enable the interactions require in the motivating application described in section 2. Section 5 presents an experimental evaluation of Pawn.Section 6 concludes the paper.

## 2 Motivating Application: Enabling Autonomic Oil Reservoir Optimization

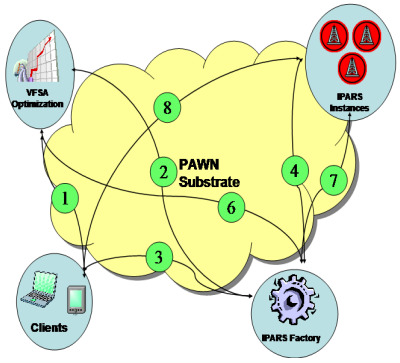


**Fig. 1.** Autonomous optimization in IPARS using VFSA

The research presented in this paper and the Pawn messaging substrate is motivated by the autonomic oil reservoir optimization process on the Grid. The goal of the process is to dynamically optimize the placement and configuration of oil wells to maximize revenue. The overall application scenario is illustrated in Figure 1. The peer components involved include: **(1) Integrated Parallel Accurate Reservoir Simulator (IPARS)**[8] providing sophisticated simulation components that encapsulate complex mathematical models of the physical interaction in the subsurface. **(2) IPARS Factory** responsible for configuring IPARS simulations, executing them on resources on the Grid and managing their execution. **(3) Very Fast Simulated Annealing (VFSA)** an optimization service based on statistical physics **(4) Economic Modeling Service** that uses IPARS simulation outputs and current market parameters (oil prices, costs, etc.) to compute estimated revenues for a particular reservoir configuration. **(5) DISCOVER Middleware** that integrates Globus [9] Grid services (GSI, MDS, GRAM, and

GASS), via the CorbaCoG [10], and DISCOVER remote monitoring, interactive steering, and collaboration services, **(6) DISCOVER Collaborative Portals** providing experts (scientists, engineers) with collaborative access to other peers component. These entities need to dynamically discover and interact with one another as peers to achieve the overall application objectives. The experts use the portals to interact with the DISCOVER middleware and the Globus Grid services to discover and allocate appropriate resource, and to deploy the IPARS Factory, VFSA and Economic model peers ((1)). The IPARS Factory discovers and interacts with the VFSA service peer to configure and initialize it ((2)). The expert interacts with the IPARS Factory and VFSA to define application configuration parameters ((3)). The IPARS Factory then interacts with the DISCOVER middleware to discover and allocate resources and to configure and execute IPARS simulations ((4)). The IPARS simulation now interacts with the Economic model to determine current revenues, and discovers and interacts with the VFSA service when it needs optimization ((5)). VFSA provides IPARS Factory with optimized well information ((6)), which then launches new IPARS simulations ((7)). Experts at anytime can discover and collaboratively monitor and interactively steer IPARS simulations, configure the other services and drive the scientific discovery process ((8)). Once the optimal well parameters are determined, the IPARS Factory configures and deploys a production IPARS run.

## 3 Design and Implementation of Pawn

A conceptual overview of the Pawn framework is presented in Figure 2 and is composed of peers (computing, storage, or user peers), network and interaction services, and mechanisms. These components are layered to represent the requirements stack enabling interactions in a Grid environment. The figure can be read from bottom to top as :"Peers compose messages handled by services through specific interaction modalities". In Pawn, peers can implement one or more services (behaviors); the combination of services implemented by a peer defines its role. Typical roles for a peer are client, application or rendezvous.

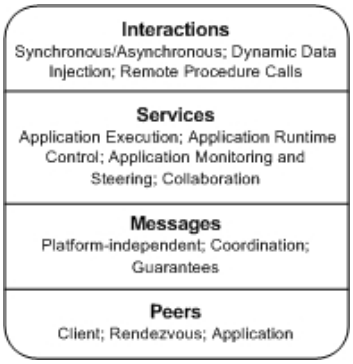


**Fig. 2.** Pawn requirements stack.

A client peer deploys applications on resources and accesses them for interactive monitoring and/or steering. It also collaborates with other peers in the peergroup.An application peer exports its application interfaces and controls to the peergroup. These interfaces are used by other peers to interact with the application. The rendezvous peer distributes and relays messages, and filters them en route to their destination.

Pawn builds on Project JXTA[11], a general-purpose-peer-to-peer framework. JXTA concepts include peer, peergroup, advertisement , module , pipe , rendezvous , and security that is provided by the routing and transport layers at every endpoint. JXTA defines protocols for : (1) discovering peers (Peer Discovery Protocol, PDP), (2) binding virtual end-to-end communication channels between peers (Pipe Binding Protocol, PBP), (3) resolving queries (Peer Resolver Protocol, PRP), (4) obtaining information on a particular peer, such as its available memory or CPU load (Peer Information Protocol, PIP) (5) propagating messages in a peergroup (Rendezvous protocol, RVP), (6) determining and routing from a source to a destination using available transmission protocols (Endpoint Routing Protocol, ERP). Protocols in JXTA define the format of the messages exchanged as well as the behavior adopted on receiving a message.

### 3.1 Pawn Services

A network service is a functionality that can be implemented by a peer and made available to a peergroup. File-sharing or printing are typical examples of network services. In Pawn, network services are application-centric and provide the mechanisms to query, respond, subscribe, or publish information to a peergroup. Pawn offers four key services to enable dynamic collaborations and autonomic interactions in scientific computing environments. These services are: *Application Runtime Control*, *Application Monitoring and Steering*,*Application Execution*, and *Group Communication*.

**Application Execution service [AEX]**: The Application Execution service enables a peer to remotely start, stop, get the status of, or restart an application. This service requires a mechanism to make synchronous and guaranteed remote calls for resource allocation and application deployment (transaction oriented interaction).

**Application Monitoring and Steering service [AMS]**: The Application Monitoring and Steering service handles interactive(i.e. Request/Response) application querying (i.e. PULL) and dynamic setting (i.e. PUSH) of application parameters. It requires support for synchronous and asynchronous communications and guaranteed message delivery and dynamic data injection (e.g. to push information to an application at runtime).

**Application Runtime Control service [ARC]**: The application runtime control service announces the existence of an application to the peergroup, sends application responses, publishes application update messages, and notifies the peergroup of an application termination. This service requires a mechanism to push information to the peergroup and respond to queries (PUSH and Request/Response interaction).

**Collaboration Service [Group communication, Presence]**: The collaboration service defines collaborative tools, group communication and presence. Collaborating peers need to establish direct end-to-end communications through synchronous/asynchronous channels (e.g. for file transfer or text communication), and be able to publish information to the peergroup (Transcation and PULL interactions).

### 3.2 Implementation of Pawn Services

Pawn builds on the current Java implementation of the JXTA protocols. JXTA defines unicast pipes that provide a communication channel between two endpoints, and propagate pipes that can propagate a message to a peergroup. It also define the Resolver Service that sends and receives messages in an asynchronous manner. Recipients of the message can be a specific peer or an entire peergroup. The pipe and resolver service use the available underlying transport protocol (TCP, HTTP, TLS). To realize the four services identified above, Pawn extends the pipe and resolver services to provide stateful and guaranteed messaging. This messaging is then used to enable the key application-level interactions such as synchronous/asynchronous communication, dynamic data injection, and remote procedure calls.

**Stateful Messages**: In Pawn, every message contains a source and destination identifier, a message type, a message identifier, a payload, and a handler tag. The handler tag uniquely identifies the service that will process the message. State is maintained through the payload that contains system/application parameters. These messages are defined in XML to provide platform-independence.

**Message guarantees**: Pawn implements application-level communication guarantees by combining stateful messages and a per-message acknowledgment table maintained at every peer. Guarantees are provided by using message queues (FIFO) to handle incoming and outgoing messages. Every outgoing message that expects a response is flagged in the table as awaiting acknowledgment, this flag is removed once the message is acknowledged. Messages contain a default timeout value representing an upper limit on the estimated response time. If an acknowledgement is not received and/or the timeout value expires, the message is resent, blocking or not blocking the current process depending on the communication type (synchronous or asynchronous). The message identifier is a composition of the destination and sender's unique peer identifiers; it is incremented for every transaction during a session (interval between a peer joining and leaving a peergroup) to provide the application-level message ordering guarantees.

**Synchronous/Asynchronous communication**: Communication in JXTA can be synchronous when using blocking pipes, or asynchronous when using non-blocking pipes or the resolver service. In order to provide reliable messaging, Pawn combines these communication modalities with stateful messaging and guarantee mechanism.

**Dynamic Data Injection**: In Pawn, every peer advertisement contains a pipe advertisement, which uniquely identifies a communication channel to it. This pipe is used by other peers to create an end-to-end channel to send and receive messages dynamically (see figure**??**). Every interacting peer implements a message handler that listens for incoming messages on the peer's input pipe channel; the message payload contains application-sensitive data that can be dynamically passed to the application/service identified by the handler tag field.

**Remote Method Calls (PawnRPC)**: The PawnRPC mechanism provides the low-level constructs for building applications interactions across distributed peers. Using PawnRPC, a peer can invoke a method on a remote peer dynami-

cally by passing its request as an XML message through a pipe. The interfaces of the methods that can be remotely invoked are published as part of the peer advertisement during peer discovery. The XML message is a composition of the destination address, the remote method name, the arguments of the method, and the arguments associated types. Upon receiving an RPC message, a peer locally checks the credentials of the sender, and if the sender is authorized, the peer invokes the appropriate method and returns a response to the requesting peer. The process may be done in a synchronous or asynchronous manner. PawnRPC uses the messaging guarantees to assure delivery ordering, and stateful messages to tolerate failure.

## 4 Enabling autonomic reservoir optimization using Pawn

In this section we describe how the interaction services provided by Pawn are used to support the autonomic oil reservoir process outlined in Figure 1 (Section 2). Every interacting component described in Section 2 is a peer that implements Pawn services. The IPARS Factory, VFSA, and the DISCOVER middleware are Application peers and implement ARC and AEX services. The DISCOVER portals are Client peers and implement AMS and Group communication services. **Peer Deployment:** The Application Execution service in Pawn uses the CorbaCoG kit (and the Globus Grid services it provides) in conjunction with the DISCOVER middleware to provide client peers access to Grid resources and services, and to deploy application peers. **Autonomic Optimization:** Autonomic optimization involves interactions between the IPARS simulation, VFSA service and the IPARS Factory. VFSA provides the IPARS Factory with an initial guess based on the initialization parameters entered by the client. The IPARS factory then configures and spawns an IPARS instance. The simulation output is used by the Economic Model to generate the estimated revenue. This revenue is normalized and then fed back to VFSA, which generates a subsequent guess. The process continues until the terminating condition is reached (e.g. revenue stabilizes). These interactions build on the Application Runtime Control service (ARC) and the PawnRPC mechanism.

Collaboration and Interactive Monitoring and Steering: Users can collaboratively connect to, monitor, and interactively steer the IPARS Factory, VFSA, Economic model and an IPARS simulation instance using the Collaboration Services to communicate with other users, and the Application Monitoring and Steering service (AMS) to interact with the Application Runtime Control service (ARC) implemented by the application peer. An interaction between AMS and ARC is presented on Figure**??**.

## 5 Experimental evaluation of Pawn

Pawn was deployed on 20 hosts (Pentium IV processors at 1.5 GHz with 512 MB of RAM, running Linux RedHat 7.2) located at Rutgers University. The Wide Area measurements were performed using two PCs (a 750 MHz Pentium

III with 256 MB RAM and a 350 MHz Pentium II with 256 MB RAM) deployed on a private ISP. The evaluation consisted of the following experiments:

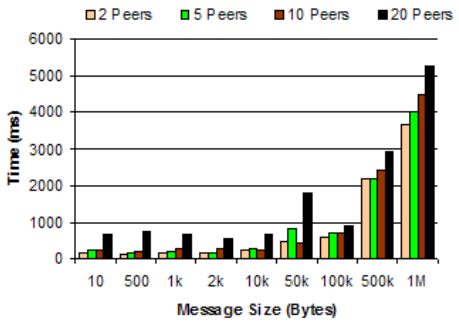


**Fig. 3.** RTT measurement in a LAN

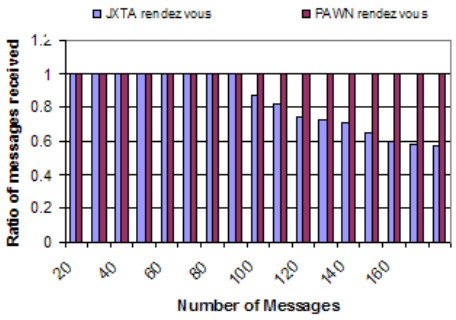


**Fig. 4.** Effectiveness of message queueing

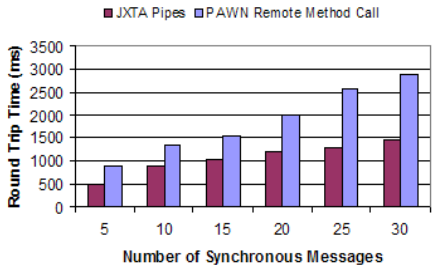


**Fig. 5.** PawnRPC over JXTA pipes

**Round Trip Time communication (RTT) over LAN**: This experiment evaluates the round trip time of messages sent from a peer running the *Application Monitoring and Steering service*, to peers running the *Application Runtime Control service* over a LAN. The message size varies from 10 bytes to 1 Megabyte. The Pawn services (AMS and ARC) build the query and response messages that are then sent using the JXTA Endpoint Routing Protocol. The overall message transfer time is tightly bound to the performance of the JXTA platform, and is likely to improve with the next generation JXTA platform. The results are plotted in Fig.3. Note that the difference in RTT between 2 peers and 20 peers decreases as the message size increases.

**Effectiveness of message queuing:** This experiment compares the behavior of a Pawn rendezvous peer implementing the application-level message queuing to the behavior of a core JXTA rendezvous peer. The number of messages published on the rendezvous peer range from 10 to 500. The ratio of messages received is plotted in Figure 4. It can be seen that the message queuing functionality guarantees that no application-level messages are dropped even under heavy load.

**Overhead of PawnRPC on JXTA pipes**: Figure 5 shows a comparison between PawnRPC and JXTA pipes. Using PawnRPC, a message describing the remote call is marshaled and sent to the remote peer. The remote peer unmarshals the request, processes it before marshaling and sending back a response to the requesting peer. The marshaling, unmarshaling, and invocation add an

overhead on the plain pipe transaction. This overhead remains however less than 50% on average.

## 6 Summary and Conclusions

This paper presented the design, implementation, and evaluation of Pawn, a peer-to-peer messaging substrate that builds on project JXTA to support peer-to-peer interactions for scientific applications on the Grid. Pawn provides a stateful and guaranteed messaging to enable key application-level interactions such as synchronous/asynchronous communication, dynamic data injection, and remote procedure calls. It exports these interaction modalities through services at every step of the scientific investigation process, from application deployment, to interactive monitoring and steering, and group collaboration. The use of Pawn to enable peer-to-peer interactions for an oil reservoir optimization application on the Grid were presented. Pawn is motivated by our conviction that the next generation of scientific and engineering Grid applications will be based on continuous, seamless and secure interactions, where the application components, Grid services, resources (systems, CPUs, instruments, storage) and data (archives, sensors) interact as peers.

## References


1. : SETI@Home. Internet: http://setiathome.ssl.berkeley.edu (1998)
2. Monson-Haefel, R., Chappell, D.: Java Message Service. O'Reilly & Associates, Sebastopol, CA, USA (2000)
3. Fabret, F., Jacobsen, A., Llirbat, F., Pereira, J., Ross, K.A., Shasha, D.: Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems. ACM SIGMOD Record **30** (2001) 115–126
4. : IBM MQSeries. Internet: http://www-3.ibm.com/ software/ts/mqseries/ (2002)
5. Seymour, K., Nakada, H., Matsuoka, S., Dongarra, J., Lee, C., Casanova, H.: Overview of GridRPC: A Remote Procedure Call API for Grid Computing. In Parashar, M., ed.: Proceedings of the Third International Workshop on Grid Computing (GRID 2002), Baltimore,MD, USA, Springer (2002) 274–278
6. Fox, G., Pallickara, S., Rao, X.: A Scaleable Event Infrastructure for Peer to Peer Grids. In: Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande, Seattle, Washington, USA, ACM Press (2002) 66–75
7. Furmento, N., Lee, W., Mayer, A., Newhouse, S., Darlington, J.: ICENI: An Open Grid Service Architecture Implemented with Jini. In: SuperComputing 2002 (SC2002), Baltimore, MD, USA (2002) 10 pages in CDROM.
8. : IPARS: Integrated Parallel Reservoir Simulator. Internet: http://www.ticam.utexas.edu/CSM (2000) Center for Subsurface Modeling, University of Texas at Austin.
9. Foster, I., Kesselman, C.: The Globus Project: A Status Report. In: IPPS/SPDP'98 Heterogeneous Computing Workshop, Orlando, Florida, USA (1998) 4–18
10. Parashar, M., Laszewski, G.V., Verma, S., Gawor, J., Keahey, K., Rehn, H.N.: A CORBA Commodity Grid Kit. Special Issue on Grid Computing Environments, Concurrency and Computation: Practice and Experience **14** (2002) 1057–1074
11. : Project JXTA. Internet: http://www.jxta.org (2001)