Enabling Peer to Peer Grids

Geoffrey Fox Shrideep Pallickara Xi Rao
Community Grid Computing Laboratory, Indiana University,
Suite 224, 501 N. Morton St., IN 47404, USA.
1-812-856-7977 1-812-856-1311 1-812-856-0765
gcf@indiana.edu spallick@indiana.edu xirao@indiana.edu

ABSTRACT
In this paper we propose a peer-to-peer (P2P) grid comprising resources such as relatively static clients, high-end resources and a dynamic collection of multiple P2P subsystems. We investigate the architecture of the messaging and event service that will support such a hybrid environment. We designed a distributed publish-subscribe system NaradaBrokering for XML specified messages. NaradaBrokering interpolates between centralized systems like JMS (Java Message Service) and P2P environments. Here we investigate and present our strategy for the integration of JXTA into NaradaBrokering. The resultant system naturally scales with multiple Peer Groups linked by NaradaBrokering.

Categories and Subject Descriptors
C.4 [**Computer Systems Organization**]: Performance of Systems – *Design studies, Performance attributes*.

General Terms
Performance, Design, Reliability

Keywords
Event distribution systems, middleware, P2P systems, grid computing, JXTA

INTRODUCTION
The peer-to-peer (abbreviated as P2P) style interaction [10] model facilitates sophisticated resource sharing environments between "consenting" peers over the "edges" of the internet; the "disruptive" [11] impact of which has resulted in a slew of powerful applications built around this model. Resources shared could be anything – from CPU cycles, exemplified by SETI@home (extraterrestrial life) [14] and Folding@home (protein folding) [15], to files (Napster and Gnutella [17]). Resources in the form of direct human presence include collaborative systems (Groove [18]) and Instant Messengers (Jabber [16]). Peer "interactions" involves advertising resources, search and subsequent discovery of resources, request for access to these resources, responses to these requests and exchange of messages between peers. An overview of P2P systems and their deployments in distributed computing and collaboration can be found in [9]. Systems tuned towards large-scale P2P systems include *Pastry* [19] from Microsoft, which provides an efficient location and routing substrate for wide-area P2P applications. Pastry provides a self-stabilizing infrastructure that adapts to the arrival, departure and failure of nodes. The JXTA [12] (from *juxta*position) project at Sun Microsystems is another research effort that seeks to provide such large-scale P2P infrastructures. Discussion pertaining to the adoption of event services as a key building block supporting P2P systems can be found in [8,9]. In this paper we propose a peer-to-peer (P2P) grid comprising resources such as relatively static clients, high-end resources and a dynamic collection of multiple P2P subsystems. We investigate the architecture, comprising a distributed brokering system that will support such a hybrid environment. Services can be hosted on such a P2P grid with peer groups managed locally and arranged into a global system supported by core servers. Access to services can then be mediated either by the "broker middleware" or alternatively by direct peer-to-peer (P2P) interactions between machines "on the edge". The relative performance of each approach (which could reflect computer/network cycles as well as the existence of firewalls) would be used in deciding on the implementation to use. P2P approaches best support local dynamic interactions; the distributed broker approach scales best globally but cannot easily manage the rich structure of transient services, which would characterize complex tasks. We use our research system NaradaBrokering as the distributed brokering core to support such a hybrid environment.

There are several attractive features in the P2P model, which motivate the development of such hybrid systems. Deployment of P2P systems is entirely user driven obviating the need for any dedicated management of these systems. Peers expose the resources that they are willing to share and can also specify the security strategy to do so. Driven entirely on demand a resource may be replicated several times; a process that is decentralized and one over

which the original peer that advertised the resource has sometimes little control over. Peers can form groups with the fluid group memberships. In addition P2P systems tend to be very dynamic with peers maintaining an intermittent digital presence. P2P systems incorporate schemes for searching and subsequent discovery of resources. In P2P systems, not every request (search) goes through, and even if it does, there could be zero or more valid responses (discovery). Peers anticipate neither the template that the responses conform to nor the order in which these responses would be received. Furthermore, responses are not identical with each responding peer processing any given request based on the resources at its disposal and it's interpretation of the request. Communication between a requesting peer and responding peers is facilitated by peers en route to these destinations. These intermediate peers are thus made aware of capabilities that exist at other peers. This discovery of services offered by other peers constitutes dynamic real time knowledge propagation. Furthermore, since peer interactions, in most P2P systems, are XML based, peers could be written in any language and can be compiled for any platform.

There are also some issues that need to be addressed while incorporating support for P2P interactions. P2P interactions are self-attenuating with interactions dying out after a certain number of hops. These attenuations in tandem with traces of the peers, which the interactions have passed through, eliminate the continuous echoing problem that result from loops in peer connectivity. However, attenuation of interactions sometimes prevents peers from discovering certain services that are being offered. This results in P2P interactions being very localized. These attenuations thus mean that the P2P world is inevitably fragmented into many small subnets that are not connected. Peers in P2P systems interact directly with each other and sometimes use other peers as intermediaries in interactions. Specialized peers are sometimes deployed to enhance routing characteristics. Nevertheless, sophisticated routing schemes are seldom in place and interactions are primarily through simple forwarding of requests with the propagation range determined by the attenuation indicated in the message. However, attenuations are minimized by the small-world [1,11 pp 203-241] effect in P2P systems.

Efficient brokering environments need to be deployed using a distributed network of brokers to address the issues of scaling, load balancing and failure resiliency. Distributed dynamic publish subscribe is an attractive model for both synchronous [29] and asynchronous communication. Note that this system must support many different patterns including P2P and centralized models. Native NaradaBrokering supports this flexibility but we must also expect that realistic scenarios will require the integration of multiple brokering schemes. NaradaBrokering supports this hybrid case through gateways to the other event worlds. NaradaBrokering supports both JMS and JXTA, which are publish/subscribe environments with very different interaction models. Details pertaining to the JMS integration can be found in [24]. In this paper, we look at both the native NaradaBrokering model and its integration with JXTA. This paper is organized as follows. Section 1 provides an overview of the NaradaBrokering system. Section 2 outlines entry points for NaradaBrokering's support for P2P interactions, while section 3 provides an overview of JXTA. Section 4 describes our strategy of integrating NaradaBrokering and JXTA. Finally, we outline possible Narada-JXTA applications and describe the status of our experiments.

1. NARADABROKERING

NaradaBrokering [3-7,24] is an event brokering system designed to run on a large network of cooperating broker nodes. Communication within NaradaBrokering is asynchronous and the system can be used to support different interactions by encapsulating them in specialized events. Events are central in NaradaBrokering and encapsulate information at various levels as depicted in the figure 1. Clients can create and publish events, specify interests in certain types of events and receive events that conform to specified templates. Client interests are managed and used by the system to compute destinations associated with published events. Clients, once they specify their interests, can disconnect and the system guarantees the delivery of matched events during subsequent reconnects. Clients reconnecting after prolonged disconnects, connect to the local broker instead of the remote broker that it was last attached to. This eliminates bandwidth degradations caused by heavy concentration of clients from disparate geographic locations accessing a certain known remote broker over and over again. The delivery guarantees are met even in the presence of failures.

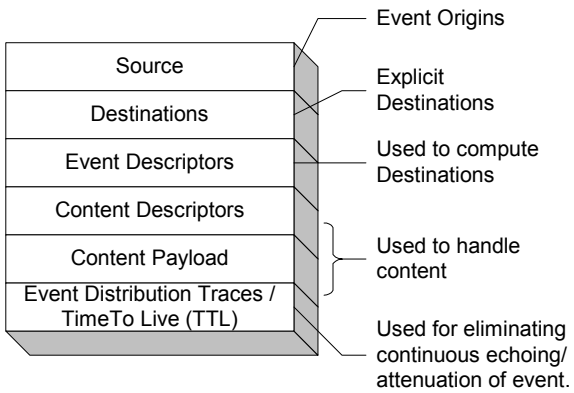

Figure 1: Event in the NaradaBrokering System

1.1 Broker Organization

Uncontrolled broker and connection additions, result in a broker network susceptible to network-partitions and devoid of any logical structure making the creation of efficient broker network maps (BNM) an arduous if not impossible task. The lack of this knowledge hampers development of efficient routing strategies, which exploit the broker topology. Such systems then resort to "flooding" the entire broker network, forcing clients to discard events they are not interested in. To circumvent this, NaradaBrokering incorporates a broker organization protocol, which manages the addition of new brokers and also oversees the initiation of connections between these brokers. The node organization protocol incorporates IP discriminators, cluster size and concurrent connection thresholds at individual brokers in its decision making process to prevent these situations.

In NaradaBrokering we impose a hierarchical structure on the broker network, where a broker is part of a cluster that is part of a super-cluster, which in turn is part of a super-super-cluster and so on. Clusters comprise strongly connected brokers with multiple links to brokers in other clusters, ensuring alternate communication routes during failures. This organization scheme results in "small world networks" [1,2] where the average communication "pathlengths" between brokers increase logarithmically with geometric increases in network size, as opposed to exponential increases in uncontrolled settings. This distributed cluster architecture allows NaradaBrokering to support large heterogeneous client configurations that scale to arbitrary size. Creation of BNMs and the detection of network partitions are easily achieved in this topology. We augment the BNM hosted at individual brokers to reflect the cost associated with traversal over connections, for e.g. intra-cluster communications are faster than inter-cluster communications. The BNM can now be used not only to compute valid paths but also for computing shortest paths. Changes to the network fabric are propagated only to those brokers that have their broker network view altered. Not all changes alter the BNM at a broker and those that do result in updates to the routing caches, containing shortest paths, maintained at individual brokers.

1.2 Dissemination of events

Every event has an implicit or explicit destination list, comprising clients, associated with it. The brokering system as a whole is responsible for computing broker destinations (targets) and ensuring efficient delivery to these targeted brokers en route to the intended client(s). Events as they pass through the broker network are be updated to snapshot its dissemination within the network. The event dissemination traces eliminate continuous echoing and in tandem with the BNM – used for computing shortest paths – at each broker, is used to deploy a near optimal routing solution. The routing is near optimal since for every event the associated targeted set of brokers are usually the only ones involved in disseminations. Furthermore, every broker, either targeted or en route to one, computes the shortest path to reach target destinations while employing only those links and brokers that have not failed or been failure-suspected.

1.3 Failures and Recovery

In NaradaBrokering, stable storages existing in parts of the system are responsible for introducing state into the events. The arrival of events at clients advances the state associated with the corresponding clients. Brokers do not keep track of this state and are responsible for ensuring the most efficient routing. Since the brokers are stateless, they can fail and remain failed forever. The guaranteed delivery scheme within NaradaBrokering does not require

every broker to have access to a stable store or DBMS. The replication scheme is flexible and easily extensible. Stable storages can be added/removed and the replication scheme can be updated. Stable stores can fail but they do need to recover within a finite amount of time. During these failures the clients that are affected are those that were being serviced by the failed storage.

1.4 Support for dynamic topologies

Support for local broker accesses, client roams and stateless brokers provide an environment extremely conducive to dynamic topologies. Brokers and connections could be instantiated dynamically to ensure efficient bandwidth utilizations. These brokers and connections are added to the network fabric in accordance with rules that are dictated by the agents responsible for broker organization. Brokers and connections between brokers can be dynamically instantiated based on the concentration of clients at a geographic location and also based on the content that these clients are interested in. Similarly average pathlengths for communication could be reduced by instantiating connections to optimize clustering coefficients within the broker network. Brokers can be continuously added or fail and the broker network can undulate with these additions and failures of brokers. Clients could then be induced to roam to such dynamically created brokers for optimizing bandwidth utilization.

1.5 JMS Compliance

NaradaBrokering is JMS [21] compliant and provides support not only for JMS clients, but also for replacing single/limited server JMS systems transparently [24] with a distributed NaradaBrokering broker network. Since JMS clients are vendor [22,23] agnostic, this JMS integration has provided NaradaBrokering with access to a plethora of applications built around JMS, while the integrated Narada-JMS solution provides these applications with scaling, availability and dynamic real time load balancing. Among the applications ported to this solution is the Anabas distance education conferencing system [25] and the Online Knowledge Center (OKC) portal [26] being developed at the IU Grid labs.

1.6 Results from the prototype

Figure 3 illustrates some results [4,7] from our initial research where we studied the message delivery time as a function of load. The results are from a system comprising 22 broker processes and 102 clients in the topology outlined in figure 2. Each broker node process is hosted on 1 physical Sun SPARC Ultra-5 machine (128 MB RAM, 333 MHz), with no SPARC Ultra-5 machine hosting two or more broker node processes. The publisher and the *measuring* subscriber reside on the same SPARC Ultra-5 machine. In addition to this there are 100 subscribing client processes, with 5 client processes attached to every other broker node (broker nodes **22** and **21** do not have any other clients besides the publisher and measuring subscriber respectively) within the system. The 100 client node processes all reside on a SPARC Ultra-60 (512 MB RAM, 360 MHz) machine. The run-time environment for all the broker node and client processes is Solaris JVM (JDK 1.2.1, native threads, JIT). The three matching values correspond to the percentages of messages that are delivered to any given subscriber. The 100% case corresponds to systems that would flood the broker network. The system performance improves significantly with increasing selectivity from subscribers. We found that the distributed network scaled well with adequate latency (2 milliseconds per broker hop) unless the system became saturated at very high publish rates.

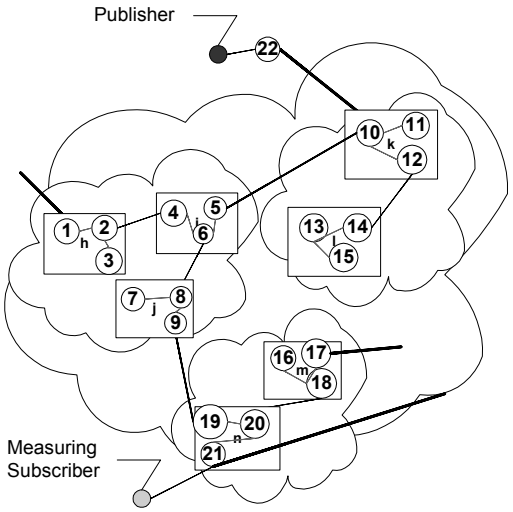

Figure 2: Test Topology

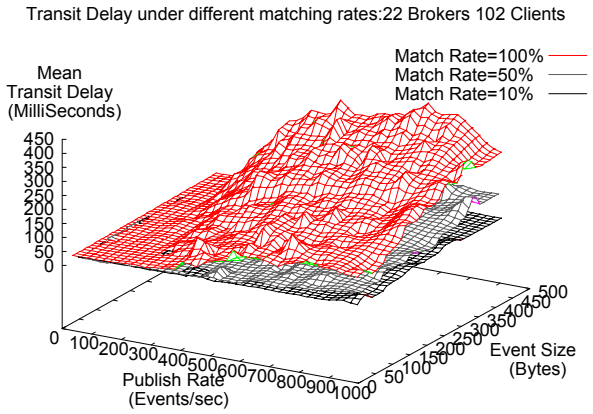

Figure 3: Transit delays for different matching rates

NaradaBrokering's JMS compliant messaging solution also gave comparable performance [24] to SonicMQ except for smaller message payloads at low publish rates. However, we do understand how a production version of the NaradaBrokering system could give Grande performance – about a factor of 3 lower in latency than the prototype. By improving the thread scheduling algorithms and incorporating flow control (needed at high publish rates) into the NaradaBrokering core significant gains in performance can be achieved. Currently we do not intend to incorporate any non-Java modules.

2.0 NARADABROKERING AND P2P INTERACTIONS

Issues in P2P systems pertaining to the discovery of services and intelligent routing can be addressed very well in the NaradaBrokering brokering system. The broker network would be used primarily as a delivery engine, and a pretty efficient one at that, while locating peers and propagating interactions to relevant peers. The most important aspect in P2P systems is the satisfaction of peer requests and discovery of peers and associated resources that could handle these requests. The broker network forwards these requests only to those peers that it believes can handle the requests. Peer interactions in most P2P systems are achieved through XML-based data interchange. XML's data description and encapsulation properties allow for ease of accessing specific elements of data. Individual brokers routing interactions could access relevant elements, cache this information and use it subsequently to achieve the best possible routing characteristics. The brokering system, since it is aware of advertisements, can also act as a hub for search and discovery operations. These advertisements when organized into "queryspaces" allow the integrated system to respond to search operations more efficiently.

Resources in NaradaBrokering are generally within the purview of the broker network. P2P systems replicate resources in an ad hoc fashion, the availability of which is dependent on the peer's active digital presence. Some resources, however, are best managed by the brokering system rather than being left to the discretion of peers who may or may not be present at any given time. An understanding of the network topology and an ability to pin point the existence of peers interested in that resource are paramount for managing the efficient replications of a resource. The distributed broker network, possessing this knowledge, best handles this management of resources while ensuring that these replicated resources are "closer" and "available" at locations with a high interest in that resource. Furthermore, the broker network is also better suited, than a collection of peers, to eliminate race conditions and deadlocks that could exist due to a resource being accessed simultaneously by multiple peers. The broker network can also be responsive to changes in peer concentrations, volumes of peer requests, and resource availability. Brokers and associated interconnections can be dynamically instantiated or purged to compensate for affected routing characteristics.

As mentioned earlier, P2P systems fragment into multiple disconnected sub-systems. NaradaBrokering could also be used to connect islands of peers together. Peers that are not directly connected through the peer network could be indirectly connected through the broker network. Peer interactions and resources in the P2P model are traditionally

unreliable, with interactions being lost or discarded due to peer failures or absences, overloading of peers and queuing thresholds being reached. Guaranteed delivery properties existing in NaradaBrokering can augment peer behavior to provide a notion of reliable peers, interactions and resources. Such an integrated brokering solution would also allow for hybrid interaction schemes to exist alongside each other. Applications could be built around hybrid-clients that would exhibit part peer behavior and part traditional client behavior (e.g. JMS). P2P communications could be then used for traffic where loss of information can be sustained. Similarly, hybrid-clients needing to communicate with each other in a "reliable" fashion could utilize the brokering system's capabilities to achieve that. Sometimes, hybrid-clients can satisfy each other's requests, in which case they would, obviating need for funneling interactions through the broker network. The broker merely serves as an efficient conduit for supporting interaction between different applications (clients, peers or hybrid).

3. JXTA

JXTA is a set of open, generalized protocols to support peer-to-peer interactions and core P2P capabilities such as indexing, file sharing, searching, peer grouping and security. The JXTA peers, and rendezvous peers (specialized routers), rely on a simple forwarding of interactions for disseminations and rely on time-to-live (TTL) indicators and peer traces to attenuate interaction propagations. However JXTA interactions are unreliable, tend to be very localized and are based on simple forwarding. Figure 4 depicts the protocols that comprise the XML encoded JXTA protocol suite. Table 1 outlines the functionality of each protocol layer comprising the JXTA suite [13].

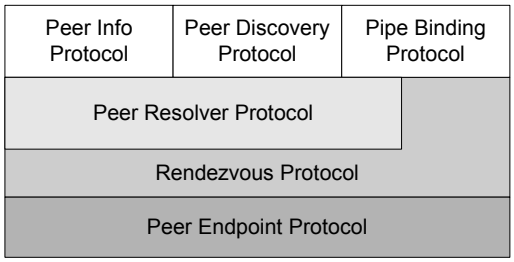

Figure 4: The JXTA protocol suite

Table 1: Functionality of the various JXTA protocols

Protocol Layer	Functionality
Peer Info	Check another peer's status
Peer Discovery	Publish & receive advertisements for a peer group
Peer Binding	Create pipe for communication
Peer Resolver	Send/Receive queries to/from other peers
Rendezvous	Initiate propagations to peers.
Endpoint	Uses network protocols to handle routing

JXTA is independent of transport protocols and can be implemented on top of TCP/IP, HTTP, TLS, Bluetooth, HomePNA, and many other protocols. JXTA provides features such as dynamic discovery and a rich search mechanism while allowing peers to communicate across NAT, DHCP, and firewall boundaries. In JXTA a peer is any node that supports JXTA protocols and could be any digital device. Peers that seek to collaborate could come together to form a peer group. Peers within a peer group can identify each other, agree on group memberships and exchange information with each other. Peers publish the existence of a resource through an advertisement, which is simply an XML document describing the resource. Peers locate other peers, peer groups and properties pertaining to them. Once a peer joins a JXTA group, JXTA's discovery capabilities support queries for services, resources and other peers.

JXTA is programming language independent. Implementation of the core JXTA protocols in Perl 5, Object C, Ruby, Smalltalk and Python are currently underway. It is expected that existing P2P systems would either support JXTA or have bridges initiated to it from JXTA. Support for JXTA would thus enable us to leverage other P2P systems along with applications built around those systems. With regards to bridges to other existing P2P systems, LimeWire (http://www.limewire.com) is investigating shared development between the Gnutella and Project JXTA developer communities. There also proposals pertaining to the management of interactions between Jabber and JXTA peers. NaradaBrokering's support for JXTA in addition to the support for JMS would result in interactions that are robust and dynamic while being supported by a scalable and highly available system. A good example of a dynamic "peer" group is the set of Grid/Web Services [29-34] generated dynamically when a complex task runs – here existing registration/discovery mechanisms are unsuitable. A JXTA like discovery strategy within such a dynamic group combined with NaradaBrokering's JMS mode between groups seems attractive. These "peers" can of course be in "middle tier" – so such a model can be applied in the Internet universe where we have "clusters" (In our analogy JXTA runs galaxies while JMS runs the universe). We intend to investigate this model of dynamic resource management in later papers.

4. JXTA & NARADABROKERING

In our strategy for providing support for P2P interactions within NaradaBrokering, we need to ensure –

- Minimal or zero changes to the NaradaBrokering system core and the associated protocol suites.
- We also make no changes to the JXTA core and the associated protocols. We make additions to the rendezvous layer for integration purposes. Peers do not communicate directly with the NaradaBrokering system and continue to interact with other peers, rendezvous peers and Narada-JXTA proxies (which it sees as a rendezvous peer).
- Furthermore, this integration should entail neither any changes to the peers nor a straitjacketing of the interactions that these peers could have had prior to the integration.

The integration is based on the proxy model, which essentially acts as the bridge between the NaradaBrokering system and JXTA. From the figure outlining the JXTA protocol architecture it is clear that the Narada-JXTA proxy could reside in one of three layers — the peer Resolver, the Rendezvous or the Endpoint layer. The Narada-JXTA proxy, irrespective of the layer it resides in, is responsible for propagation (and receiving) interactions to (and from) the brokering system. It is conceivable that an implementation of the JXTA transport binding mandates handshakes prior to connection set ups and packet exchanges to monitor the state of the connection. These data exchanges are pertinent to the underlying transport implementation and should obviously not be propagated within the brokering system. Thus if the Narada-JXTA proxy were to reside in the Endpoint layer, it would entail an inspection of every data packet prior to propagating it to the brokering system. Furthermore, overheads associated with testing each message prior to forwarding it to the brokering system can add up in the form of queuing delays that might eventually slow the system down. On the other hand if we were to set up the Narada-JXTA proxy inside the Resolver layer we run into the problem of lost interactions. Not all interactions received at the Rendezvous layer are propagated to the Resolver layer. The Resolver layer deals only with the issue and receipt of generic queries to find or search for peers, peer groups, pipes, and other resources. Other peer interactions are not routed to this layer. The Rendezvous layer, which receives all the P2P interactions and does not have the problem of lost interactions, is thus the most appropriate layer for the Narada-JXTA proxy to reside in. The Rendezvous layer has most P2P interactions forwarded to it from the Endpoint Layer. Among the few interactions not forwarded to the Rendezvous layer is the query response, which is handled only within the Endpoint layer of peers en route to the original querying peer. We discuss this issue in more detail in subsequent sections. The routing headers associated with interactions are however accessed within the Endpoint layer. In some cases, which we discuss in the section pertaining to handling queries and responses, the headers need to be processed by the Rendezvous layer to ensure processing within the integrated NaradaBrokering-JXTA system.

The Narada-JXTA proxy, operating inside the JXTA rendezvous layer, serves in a dual role as both a rendezvous peer and as a NaradaBrokering client providing a bridge between NaradaBrokering and JXTA. NaradaBrokering could be viewed as a service by JXTA. The discovery of this service is automatic and instantaneous due to the Narada-JXTA proxy's integration inside the rendezvous layer. Any peer can utilize NaradaBrokering as a service so long as it is connected to a Narada-JXTA proxy. Nevertheless, peers do not know that the broker network is routing some of their interactions. Furthermore, these Narada-JXTA proxies, since they are configured as clients within the NaradaBrokering system, inherit all the guarantees that are provided to NaradaBrokering clients.

4.1 The interaction model

Different JXTA interactions are queued at the queues associated with the relevant layers comprising the JXTA protocol suite [13]. Each layer performs some operations including the addition of additional information. The rendezvous layer processes information arriving at its input queues from the peer-resolving layer and the pipe-binding layer. Since the payload structure associated with different interactions is different we can easily identify the interaction types associated with the payloads. Interactions pertaining to discovery/search or communications within a peer group would be serviced both by JXTA rendezvous peers and also by Narada-JXTA proxies.

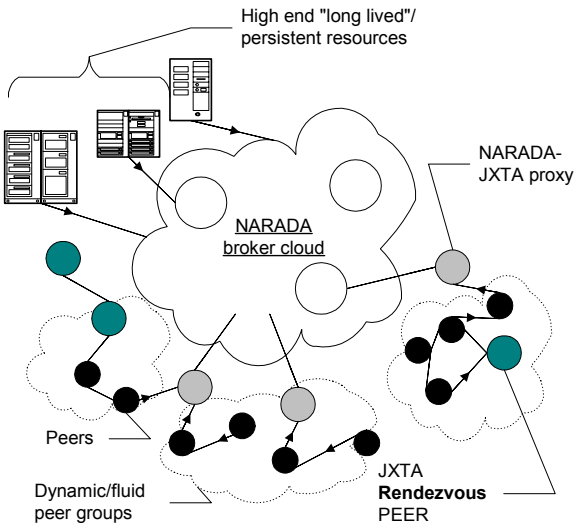

Figure 5: The Narada-JXTA interaction model

Interactions that peers have with the Narada-JXTA proxies are what are routed through the NaradaBrokering system. JXTA peers can continue to interact with each other and of course some of these peers can be connected to pure JXTA rendezvous peers. Peers have multiple routes to reach each other and some of these could include the NaradaBrokering system and some of them need not. Such peers can interact directly with each other during the request/response interactions. Figure 5 outlines the NaradaBrokering JXTA interaction model.

4.2 Interaction Disseminations

Peers can create a peer group; request to be part of a peer group; perform search/request/discovery all with respect to a specific targeted peer group. Peers always issue requests/responses to a specific peer group and sometimes to a specific peer. Peers and peer groups are identified by UUID **[27]** (IETF specification guarantees uniqueness until 3040 A.D.) based identifiers. Every peer generates its own peer id while the peer that created the peer group generates the associated peer group id. Each rendezvous peer keeps track of multiple peer groups through peer group advertisements that it receives. Any given peer group advertisement could of course be received at multiple rendezvous peers. These rendezvous peers are then responsible for forwarding interactions; if it had received an advertisement for the peer group contained in these interactions.

Narada-JXTA proxies are initialized both as rendezvous peers and also as NaradaBrokering clients. During its initialization as a NaradaBrokering client every proxy is assigned a unique connection ID by the NaradaBrokering system, after which the proxy subscribes to a topic identifying itself as a Narada-JXTA proxy. This enables NaradaBrokering to be aware of all the Narada-JXTA proxies that are present in the system. The Narada-JXTA proxy in its role as a rendezvous peer to peers receives –

- Peer group advertisements
- Requests from peers to be part of a certain peer group and responses to these requests
- Messages sent to a certain peer group or a targeted peer
- Queries and responses to these queries

To ensure the efficient dissemination of interactions, it is important to ensure that JXTA interactions that are routed by NaradaBrokering are delivered only to those Narada-JXTA proxies that should receive them. This entails that the Narada-JXTA proxy perform a sequence of operations, based on the interactions that it receives, to ensure selective delivery. The set of operations that the Narada-JXTA proxy performs comprise gleaning relevant information from JXTA's XML encapsulated interactions, constructing an event based on the information gleaned and finally in its role as a NaradaBrokering client subscribing (if it chooses to do so) to a topic to facilitate selective delivery. By subscribing to relevant topics, and creating events targeted to specific topics each proxy ensures that the broker network is not flooded with interactions routed by them. The events constructed by the Narada-JXTA proxies include the entire interaction as the event's payload. Upon receipt at a proxy, this payload is de-serialized and the interaction is propagated as outlined in the proxy's dual role as a rendezvous peer. Events constructed from

interactions need to have a unique identifier associated with them. Advertisements, since, they encapsulate information pertaining to uniquely identifiable resource can use the UUID associated with the advertised resource as the interaction identifier of the constructed event. The interaction type along with the interaction identifier allow us to uniquely identify each event. In the case of JXTA messages the unique interaction identifier is constructed based on the peer id of the peer issuing the message and the timestamp in milliseconds (based on the system-clock at the peer node) associated with the message. We now proceed to outline the sequence of operations associated with different JXTA interactions.

4.2.1 Peer Group Advertisements

When peer group advertisements propagated by a peer are received at a Narada-JXTA proxy, it creates the event depicted in figure 6. The peer group advertisement is the payload contained in this event. The proxy proceeds to initiate a subscription to the peer group with the subscription being registered to the connection that the proxy has into the NaradaBrokering system. This enables NaradaBrokering to identify this Narada-JXTA proxy as a destination when certain interactions are targeted to that specific peer group. NaradaBrokering delivers this peer group advertisement to all the Narada-JXTA proxies within the integrated Narada-JXTA system. Proxies that receive this advertisement do not initiate any actions and the proxy deals with this advertisement just as a JXTA rendezvous peer would.

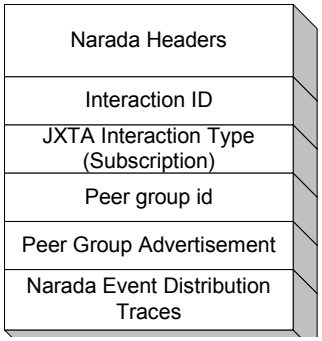

Figure 6: Event for PeerGroup Advertisement

4.2.2 Requests to be part of a peer group, and responses to these requests

When a peer issues a request to be part of a certain peer group, the event constructed by the Narada-JXTA proxy is depicted in figure 7.(a). The advertisement is contained in the payload, and the targeted topic contained in this event is the peer group id. NaradaBrokering thus propagates the event to the proxies that had subscribed to this topic. To ensure that responses to this advertisement are targeted to the proxy forwarding this request, the event also encapsulates its NaradaBrokering connection information within this event. Narada-JXTA proxies receiving this event maintain the JXTA request and the connection associated with the request; to be used during propagation of responses. These Narada-JXTA proxies then behave as a normal rendezvous peer, processing the request as it normally would. This entails forwarding/routing of events en route to peers that are part of the peer group.

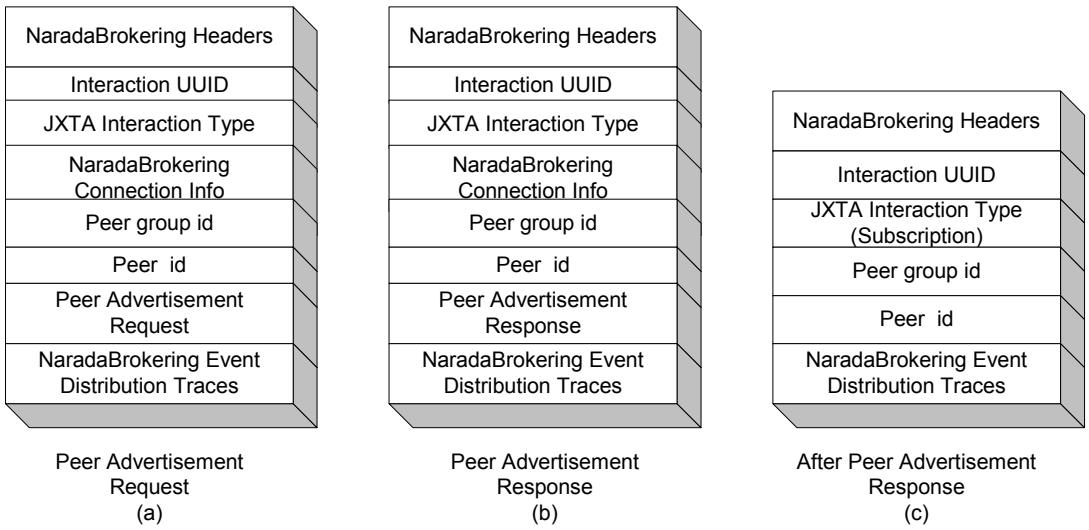

Figure 7: Dealing with request/responses to join peer groups

When responses, initiated by authorized peers, are received at the Narada-JXTA proxy, the proxy checks to see if there was a request associated with it and proceeds to forward the response encapsulated in a NaradaBrokering event depicted in figure 7.(b). The proxy also retrieves the target connection that this response should be sent to and includes it in the event. Upon receipt of this response at the initiating Narada-JXTA proxy, the proxy initiates operations in its role as a rendezvous to ensure propagation of the response to the requesting peer.

The "initiating" Narada-JXTA proxy then proceeds to subscribe to both the peergroup-id and the peer-id. This event is depicted in figure 7.(c), and as can be seen there is no payload contained in this event. The peergroup-id subscription ensures that interactions for the peer group, which the serviced peer will be a part of, are always received at the proxy; this includes advertisements to be part of that peer group. Furthermore, a lot of JXTA interactions are sometimes targeted to specific peers so we also subscribe to the peer-id contained in the received response. When events are sent to a specific peer in a peer group, the NaradaBrokering system routes the event to the Narada-JXTA proxy (or proxies) that can route the event to the targeted peer.

4.2.3 JXTA Messages

When JXTA messages arrive at a Narada-JXTA proxy, the event constructed for routing within the NaradaBrokering system is depicted in figure in figure 8. These are the elements that the NaradaBrokering system will operate on. The rest of the JXTA message needs to be serialized and would be the payload contained within the constructed event.

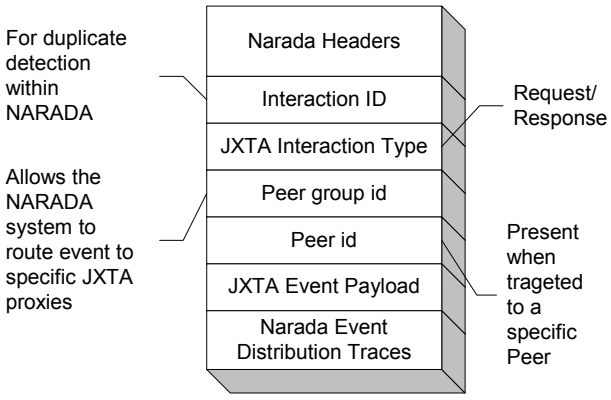

Figure 8: Event for JXTA Messages

NaradaBrokering then routes this event to only those proxies that had subscribed to either the peergroup-id or the peer-id contained in this event. Upon receipt at the Narada-JXTA proxies the payload contained in the event is unmarshalled and the JXTA message is recreated. With the Narada-JXTA proxy now behaving as any other rendezvous peer, propagation from this point on proceeds as dictated by the JXTA reference implementation. NaradaBrokering for its part deals with the efficient routing of the NaradaBrokering events based on the topic and subscriptions (propagated during receipt of peer advertisements at the proxies).

4.2.4 Dealing with queries and query responses

Peers can query for resources and this query is propagated within the system to other peers. Peers that can satisfy the issuing peer's query request respond by constructing appropriate query responses. Of course a query is attenuated if the TTL associated with the interaction is reached and there were no peers that could satisfy the peer's query. Queries maintain peer distribution traces in their headers to ensure that responses that satisfy the query contain path information to ensure that responses can reach the originating peer. A peer adds its trace within this header (the reverse path) in the Endpoint layer. The peer traces are recorded in sequence so that the reverse path can be easily constructed. Figure 9.(a) depicts a query's propagation through various peers. It is clear from the figure that if peer **D** can satisfy peer **A**'s query the peer trace can be used for the response to reach the querying peer **A**. The response generally contains either a resource or a pointer to the resource a peer might have been interested in.

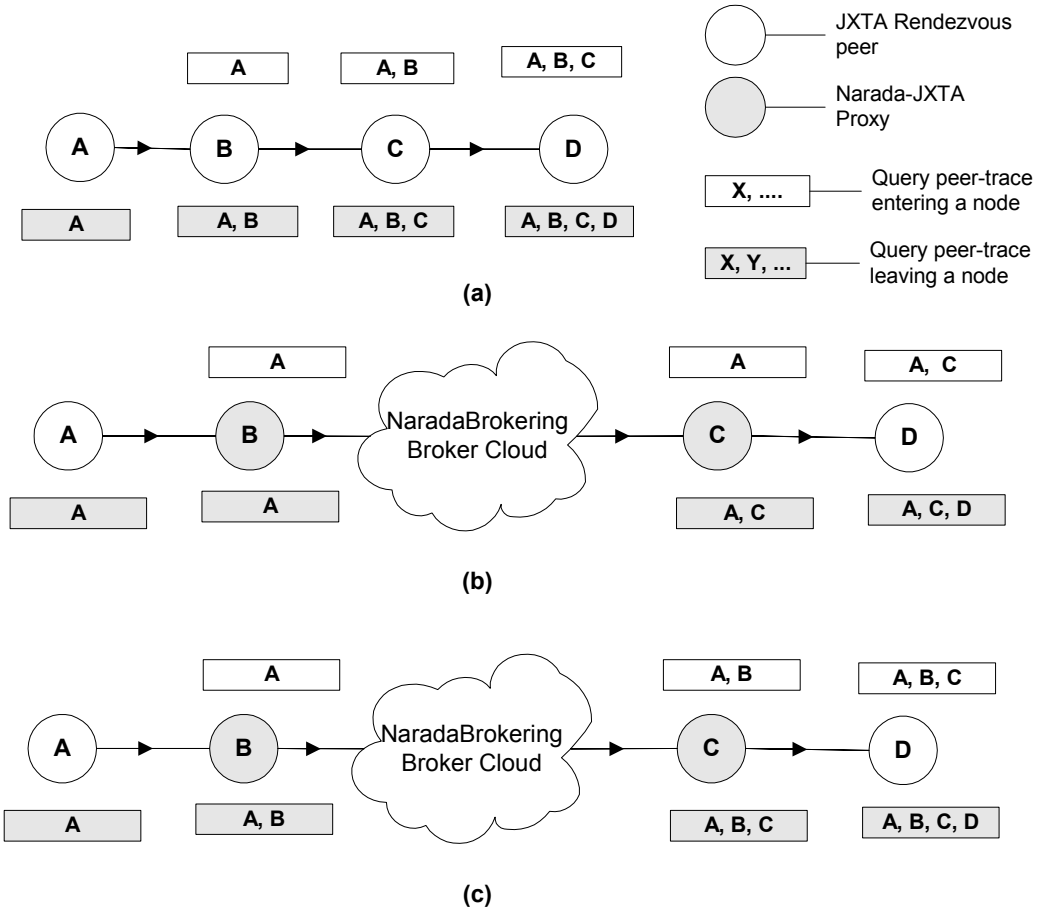

Figure 9: Peer traces in queries under different inter connection scenarios

The peer trace also indicates the last peer that processed the query and propagated it to the peer that is currently processing the query. In our setting there are two issues that we need to deal with, which if not rectified could result in a fragmented P2P system at least as far as queries are concerned. Both of these stem from inconsistent information in peer trace headers associated with queries and responses.

1. For a query propagated by a Narada-JXTA proxy into the NaradaBrokering broker cloud, the trace associated with the query does not include the peer trace of the Narada-JXTA proxy (the proxy also has a dual role as a Rendezvous peer) propagating the query. This is because the trace is updated in the Endpoint layer only after processing associated with the query is complete and the query is ready to be propagated to another peer. Since the interaction is propagated from within the Rendezvous layer (where the proxy resides), the query trace header modification needs to be done prior to this propagation.
2. When the message is received by the Narada-JXTA proxy from the brokering system, the last peer information is invalid. This is because the last peer information would refer to a peer which the Narada-JXTA proxy is not connected to.

The query responses are routed by the Endpoint layers and these responses are not propagated to the Rendezvous layers. There are two issues pertaining to the processing of query responses by the Endpoint layer. First, a query response is not forwarded to the rendezvous layer if the destination ID associated with the query response is not the peer id of the peer. Second, if the reverse path information requires the Endpoint to route the query response to a peer that it is not connected to the query response is discarded.

Figure 9.(b) depicts the peer trace headers associated with query en route to peer **D** through a couple of Narada-JXTA proxies and the NaradaBrokering system. As is clear from the figure based on the information contained in the peer trace header, there is no way for the query response to reach peer **A**. In fact the query response will be discarded by Narada-JXTA proxy **C**'s Endpoint layer because there is insufficient information regarding the peer to

be reached. During the propagation of query responses, which takes place within the Endpoint layer, the reverse path information contained in the query response might lead the Endpoint layer, at a Narada-JXTA proxy, to a peer that it is not aware of. This will result in the query response being discarded by the Endpoint layer. To circumvent this problem, in the peer hosting the Narada-JXTA proxy, we update the Endpoint layer to detect if the interaction being processed is a query response. If it is, the query response is allowed to propagate up to the Rendezvous layer where the Narada-JXTA proxy funnels the message into the brokering system. The brokering system then routes the response to relevant Narada-JXTA proxy where it arrives at the Rendezvous layer. The Rendezvous layer then percolates the query response down to the Endpoint layer where the query-response propagations proceed as outlined in the JXTA specification. Note that since the reverse path information (peer trace of the query) contained in the query response is now consistent the routing proceeds efficiently.

Thus in figure 9.(c) when a query arrives at the Narada-JXTA proxy **B**'s Endpoint layer, the message is forwarded to the Rendezvous layer, which operates on the message headers (usually the Endpoint layer's functionality) and adds its trace to the header associated with the query. Similarly when a response arrives at Narada-JXTA proxy **C**, the response if propagated to the broker network which routes the interaction to peer **B** which percolates the response down to its Endpoint layer from where the response is routed as outlined in JXTA specification.

4.3 The Duplicate detection problem:

Unlike NaradaBrokering clients that interact with only one broker at any given time and never with any other client directly; peers can be connected to multiple peers and rendezvous peers. This situation leads to having the same interaction entering the NaradaBrokering system through multiple points and also due to loops in peer connectivity. Under conditions of increasing loads and high peer concentrations the cumulative effects of these JXTA interactions entering the distributed broker network could entail prohibitive CPU/bandwidth costs. The most crucial thing is to ensure that the NaradaBrokering broker network is not inundated with such duplicate interactions.

JXTA interactions, when they are propagated into the NaradaBrokering system, are encapsulated inside events outlined in earlier sections. Destinations computed for duplicate events are identical; when these events are being routed within the NaradaBrokering network the route they take is more or less the same, i.e. they propagate through the same cluster controllers. Paths traversed by later events (duplicates) en route to final destinations are more or less the same as the event that traversed prior to it. These paths only vary during sudden network changing conditions such as failures of brokers/links and spikes in network usage over sustained durations in which case the paths computed based on network costs would vary. Even in such cases subsequent duplicates continue to traverse along identical computed paths. Having brokers keep track of the events (event id's specifically) they received and discarding duplicates allows us to solve the duplicate detection problem and prevent the network from expending network cycles for routing duplicates. This scheme allows for faster disseminations, with a survival of the fittest scheme, where duplicate events are discarded if they were not the first to arrive at a broker.

4.4 JXTA Applications and NaradaBrokering

JXTA applications that were tested within the integrated Narada-JXTA environment included the *JXTA shell*, which provides a command line interface to JXTA functionality and *myjxta* (also known as InstantP2P), which also includes an instant messaging environment. Work is currently underway to integrate *myjxta* into the Anabas Conferencing system where NaradaBrokering would provide backend support for both JMS and JXTA interactions. In [29] we described the Garnet Message Service (based on the Anabas shared display infrastructure) as a demonstration that JMS and millisecond latency for small messages can support major collaborative functions – Shared Display, Whiteboard, Polls, Text chat among others. The goal of our effort is to explore the use of Garnet with hybrid messaging and bring Garnet and NaradaBrokering functionality to JXTA.

5. NARADABROKERING-JXTA SYSTEMS

The NaradaBrokering JXTA integration service scales naturally since NaradaBrokering provides dynamic "long distance" support while JXTA provides for dynamic localized supports. In the combined global scale infrastructure NaradaBrokering works best for "long lived" and persistent resources that would be efficiently replicated within NaradaBrokering. This integrated model provides efficient search/discovery not only for static activity but also for dynamic activity while allowing JXTA interactions at the edges. The resultant system also scales with multiple JXTA Peer Groups linked by NaradaBrokering, which can dynamically instantiate new brokers to balance the load. As opposed to the simple forwarding of interactions, the intelligent routing in NaradaBrokering in tandem with the

duplicate detection scheme in our solution ensures faster disseminations and improved communication latencies for peers. Furthermore, targeted peer interactions traversing along shortest paths within the broker network obviates the need for a peer to maintain dedicated connections to a lot of peers. Proxies, due to their initialization as clients within NaradaBrokering, inherit all the guarantees accorded to clients within the NaradaBrokering such as guaranteed delivery in the presence of failures and fast disseminations. Discovery of rendezvous peers in JXTA is a slow process. A rendezvous peer generally downloads a list of other rendezvous peers from a server, not all of which would be up and running at that time. We allow for dynamic discovery of Narada-JXTA proxies, which need not be directly aware of each other, but do end up receiving interactions sent to a peer group if they had both received peer group advertisements earlier. The scheme also allows us to connect islands of peers and rendezvous peers together, allowing for a greater and richer set of interactions between these peers.

A typical application of the hybrid Narada/JXTA technology is distance education. This often consists of multiple linked classrooms where the participants in each classroom are individually linked to collaborative environment. Here a peer-to-peer model (such as JXTA) can be used in a classroom to give fast dynamic response to shared input control while the JMS style global NaradaBrokering capability is used to multicast between classrooms. More generally this combination – of globally structured and locally dynamic messaging, scales to support general applications. We can package the JXTA/JMS/NaradaBrokering hybrid as an event or messaging web service whose different modes could correspond to different ports in WSDL [28]. These ports trade off scaling, performance and reliability in a fashion that can be chosen by the user. Alternatively web services communicate via channels and we could use the technology of this paper to flexibly link different services together with dynamic choice of service provider. Other applications we are pursuing include workflow where administrative agents control the traffic between different web services. We have explained in more detail how one can base a P2P Grid architecture on a generalized publish/subscribe mechanism in [8]. NaradaBrokering is currently (like Sonic and others) pure Java in brokers and messaging. We bind XML to Java sockets when working with JXTA. We have extended this model partially and currently NaradaBrokering supports RTP (using UDP) to be used as transport mechanism for audio/video. It would be attractive for NaradaBrokering to use JXTA like flexible transport mechanisms.

6. EXPERIMENTAL SETUP

For comparing JXTA performance in NaradaBrokering we setup the topologies depicted in figures 10 and 11. For each topology we then compare the performance of the pure JXTA environment, the integrated Narada-JXTA system and the pure NaradaBrokering system. To compute communication delays while obviating the need for clock synchronizations and the need to account for clock drifts, the receiver/sender pair is setup on the same machine (Pentium-3, 1 GHz, 256 MB RAM). In all the test cases, a message published by the sender is received at the receiver and the delay is computed. For a given message payload this is done for a sample of messages and we compute the mean delay and the standard deviation associated with the samples. We also measure the Jitter *J*, which is defined as the mean deviation (smoothed absolute value) of the difference *D* in packet spacing at the receiver compared to the sender for a pair of packets. The Jitter J is computed based on the formula – $J = J + (|D(i-1, i)| - J)/16$, where $D(i-1, i)$ corresponds to the difference between the delay for i^{th} packet and the delay for the $(i-1)^{th}$ packet. This is repeated for different payload sizes. For every topology every node (broker or rendezvous peer) involved in the experimental setup is hosted on a different machine (Pentium-3, 1 GHz, 256MB RAM). The run-time environment for all the processes is (JDK-1.3 build Blackdown-1.3.1, Red Hat Linux 7.3). The machines involved in the experimental setup reside on a 100 Mbps LAN.

Figures 12 through 15 depict the mean transit delay for the message samples under the two topologies. Figure 16 is a plot of the jitter values for message samples in Topology-II. We also performed this experiment involving machines at Indiana University (IU), University of Texas (UT) and Florida State University (FSU). The sender/receiver pair was hosted on a machine (Pentium-3, 550MHz, 512MB RAM) at FSU. For each topology half the nodes resided on machines (Pentium-3, 1 GHz, 256MB RAM) at UT and the other half resided on machines (Pentium-3, 1 GHz, 256MB RAM) at IU. Figure 17 depicts the mean transit delay for topology-II involving machines at IU,UT and FSU. Results depicted in figures 12 through 17 indicate the superior performance of the integrated Narada-JXTA system compared to that of the pure JXTA system.

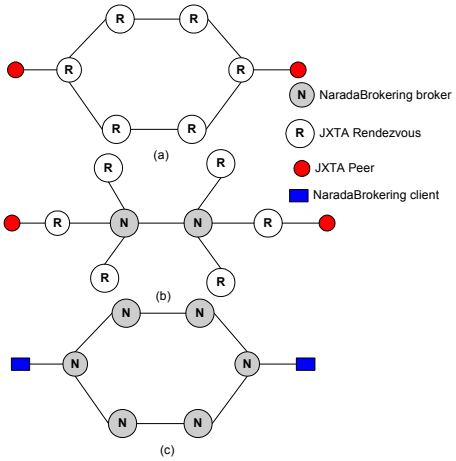

Figure 10: Test Topology-I

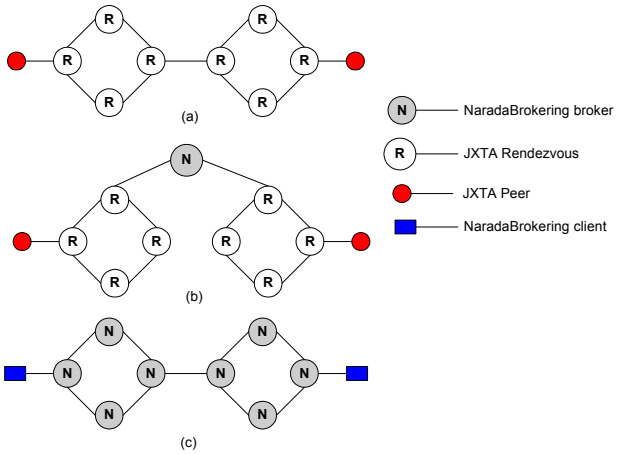

Figure 11: Test topology-II

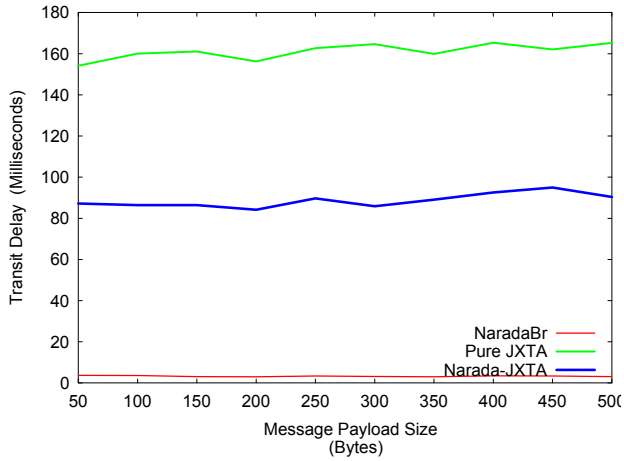

Figure 12: Mean transit delay for message samples in Topology-I (smaller payloads)

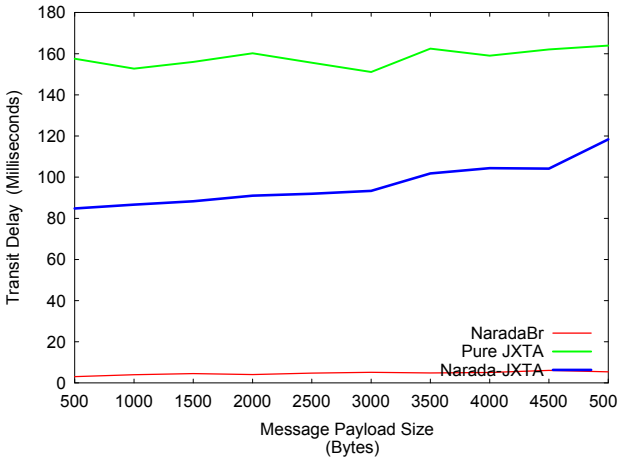

Figure 13: Mean transit delay for message samples in Topology-I (bigger payloads)

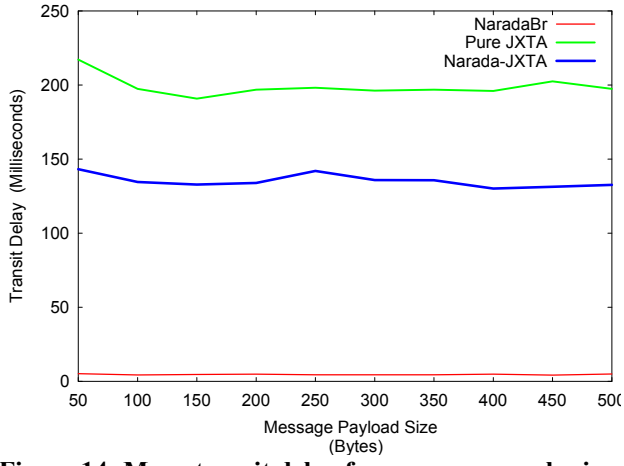

Figure 14: Mean transit delay for message samples in Topology-II (smaller payloads)

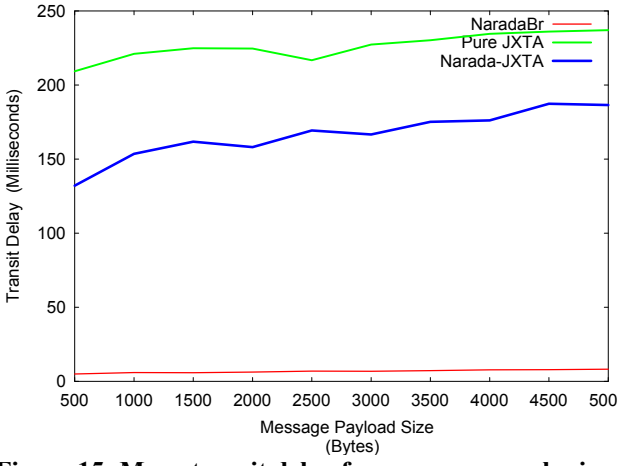

Figure 15: Mean transit delay for message samples in Topology-II (bigger payloads)

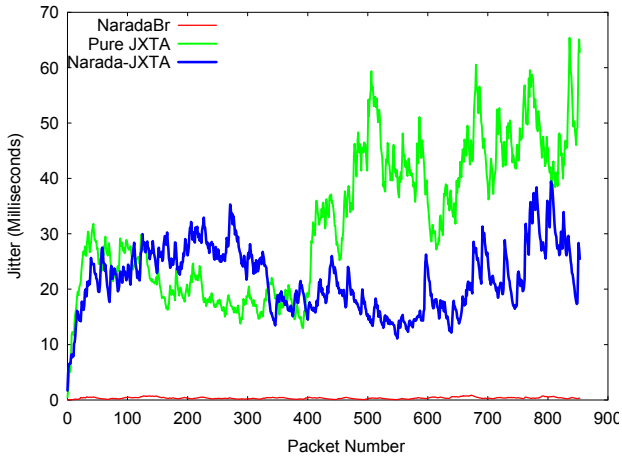

Figure 16: Jitter values for message samples in Topology-II

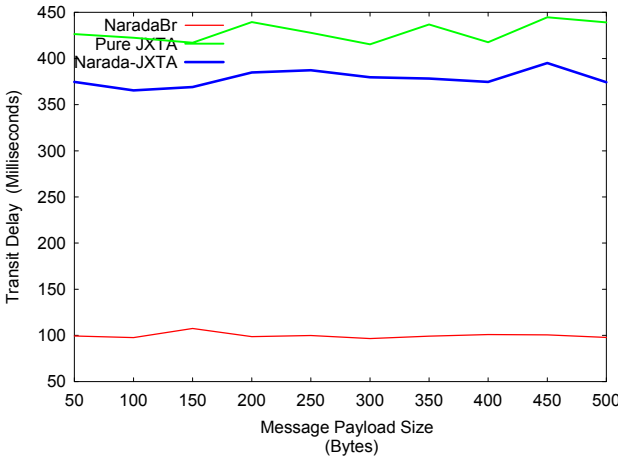

Figure 17: Mean transit delay for message samples in Topology-II (external machines)

7. CONCLUSION

In this paper we presented our strategy for a scaleable P2P grid. In NaradaBrokering we have based support for P2P interactions through JXTA. We also enumerated the benefits that can be accrued, by both NaradaBrokering and P2P systems such as JXTA, through such integrations. The successful integration of NaradaBrokering with JXTA can be combined with our earlier results on JMS [24] to demonstrate that one can both support very different brokering models within the same system and deploy hybrid systems with NaradaBrokering linking different environments. We believe such an environment is appropriate for building scalable (Grande) P2P grids supporting both dynamic local and long-range static resources. Furthermore, it is our belief that this integration, to go along with our existing JMS integration, has added considerable value to NaradaBrokering and that we are well positioned to Web Service "enable" NaradaBrokering.

REFERENCES

1. D.J. Watts and S.H. Strogatz. Collective Dynamics of Small-World Networks. *Nature*. 393:440. 1998.
2. R. Albert, H. Jeong and A. Barabasi. Diameter of the World Wide Web. *Nature* 401:130. 1999.
3. The NaradaBrokering Project at IU's Grid Computing Laboratory http://www.naradabrokering.org
4. Geoffrey Fox and Shrideep Pallickara, An Event Service to Support Grid Computational Environments, to be published in *Concurrency and Computation: Practice and Experience*, Special Issue on Grid Computing Environments.
5. Geoffrey C. Fox and Shrideep Pallickara, An Approach to High Performance Distributed Web Brokering. *ACM Ubiquity* Volume2 Issue 38. November 2001.
6. Pallickara, S., "A Grid Event Service." PhD Syracuse University, 2001.
7. Geoffrey C. Fox and Shrideep Pallickara .The Narada Event Brokering System: Overview and Extensions. Proc. of the *International Conference on Parallel and Distributed Processing Techniques and Applications*. Las Vegas 2002.
8. Geoffrey Fox, Ozgur Balsoy, Shrideep Pallickara, Ahmet Uyar, Dennis Gannon, Aleksander Slominski. Community Grids. Proc. of the *International Conference on Computational Science*. Amsterdam, Netherlands 2002.
9. Geoffrey Fox, "Peer-to-Peer Networks," *Computing in Science & Engineering*, vol. 3, no. 3, May2001.
10. openp2p P2P Web Site from O'Reilly http://www.openp2p.com.
11. "Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology", edited by Andy Oram, O'Reilly Press 2001.
12. Sun Microsystems. The JXTA Project and Peer-to-Peer Technology http://www.jxta.org
13. The JXTA Protocol Specifications. http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html
14. SETI@home Project http://setiathome.ssl.berkeley.edu.
15. Folding@home Project http://www.stanford.edu/group/pandegroup/Cosm
16. Jabber http://www.jabber.org
17. Gnutella. http://gnutella.wego.com
18. Groove Network http://www.groove.net
19. Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. Proceedings of Middleware 2001.
20. PocketJXTA Project: Porting the JXTA-c platform to the PocketPC and similar devices. http://pocketjxta.jxta.org/

21. Mark Happner, Rich Burridge and Rahul Sharma. Sun Microsystems. Java Message Service Specification. 2000. http://java.sun.com/products/jms
22. SonicMQ JMS Server http://www.sonicsoftware.com/
23. The OpenJMS Project http://openjms.exolab.org/
24. Geoffrey C. Fox and Shrideep Pallickara JMS Compliance in the Narada Event Brokering System. Proc. of the *International Conference on Internet Computing*. Las Vegas, 2002.
25. The Anabas Conferencing System. http://www.anabas.com
26. The Online Knowledge Center (OKC) Web Portal http://judi.ucs.indiana.edu/okcportal/index.jsp
27. Paul J. Leach and Rich Salz. Network Working Group. UUIDs and GUIDs. February, 1998.
28. Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/wsdl.
29. Geoffrey Fox et al. Grid Services For Earthquake Science. To appear in *Concurrency & Computation: Practice and Experience*. Special Issue on Grid Computing Environments.
30. Presentation on Web Services by Francesco Curbera of IBM at DoE Components Workshop July 23-25, 2001. Livermore, California. http://www.llnl.gov/CASC/work shops/components2001/viewgraphs/FranciscoCurbera.ppt
31. Frank Laymann, Web Services Flow Language WSFL, http://www.ibm.com/software/solutions/webservices/pdf/ WSFL.pdf
32. Harumi Kuno, Mike Lemon, Alan Karp and Dorothea Beringer, (WSCL – Web Services Conversational Language), “Conversations + Interfaces = Business logic”, http://www.hpl.hp.com/techreports/2001/HPL-20 01127.html
33. Semantic Web from W3C to describe self organizing Intelligence from enhanced web resources. http://www. w3c.org/2001/sw/
34. Berners-Lee, T., Hendler, J., and Lassila, O., "The Semantic Web," Scientific American, May2001.
35. Ken Arnold, Bryan O'Sullivan, Robert Scheifler, Jim Waldo and Ann Wollrath. The Jini Specification. Addison-Wesley. June 1999.