NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids

Shrideep Pallickara and Geoffrey Fox

Community Grid Labs, Indiana University, 501 N. Morton St, Suite 224
Bloomington, IN-47404. USA
{spallick,gcf}@indiana.edu

Abstract. A Peer-to-Peer (P2P) Grid would comprise services that include those of Grids and P2P networks and naturally support environments that have features of both limiting cases. Such a P2P grid integrates the evolving ideas of computational grids, distributed objects, web services, P2P networks and message oriented middleware. In this paper we investigate the architecture, comprising a distributed brokering system that will support such a hybrid environment. Access to services can then be mediated either by the middleware or alternatively by direct P2P interactions between machines.

1 Introduction

The Grid [1-4] has made dramatic progress recently with impressive technology and several large important applications initiated in high-energy physics [5,6], earth science [7,8] and other areas [9,10]. At the same time, there have been equally impressive advances in broadly deployed Internet technology. We can cite the dramatic growth in the use of XML, the "disruptive" impact of peer-to-peer (P2P) approaches [11] that have resulted in a slew of powerful applications, and the more orderly, but still widespread adoption, of a universal Web Service approach to Web based applications [12,13]. There are no crisp definitions of Grids and P2P Networks that allow us to unambiguously discuss their differences and similarities and what it means to integrate them. However these two concepts conjure up stereotype images that can be compared. Taking "extreme" cases, Grids are exemplified by the infrastructure used to allow seamless access to supercomputers and their datasets. P2P technology facilitates sophisticated resource sharing environments between "consenting" peers over the "edges" of the Internet, enabling ad hoc communities of low-end clients to advertise and access resources on communal computers. Each of these examples offers services but they differ in their functionality and style of implementation. The P2P example could involve services to set-up and join peer groups, browse and access files on a peer, or possibly to advertise one's interest in a particular file. The "classic" grid could support job submittal and status services and access to sophisticated data management systems.

Grids typically have structured robust security services while P2P networks can exhibit more intuitive trust mechanisms reminiscent of the "real world". Grids typically offer robust services that scale well in pre-existing hierarchically arranged organizations. P2P networks are often used when a best effort service is needed in a

M. Endler and D. Schmidt (Eds.): Middleware 2003, LNCS 2672, pp. 41–61, 2003.

dynamic poorly structured community. If one needs a particular "hot digital recording", it is not necessary to locate all sources of this, a P2P network needs to search enough plausible resources to ensure that success is statistically guaranteed. On the other hand, a 3D simulation of the universe might need to be carefully scheduled and submitted in a guaranteed fashion to one of the handful of available supercomputers that can support it. There are several attractive features in the P2P model, which motivate the development of hybrid systems. Deployment of P2P systems is entirely user driven, obviating the need for any dedicated management of these systems. Resource discovery and management is an integral part of P2P computing with peers exposing the resources that they are willing to share and the system (sometimes) replicating these resources based on demand. Grids might host different persistent services and they must be able to discover these services and the interfaces they support. Peers can form groups with the fluid group memberships and are thus very relevant for collaboration [14, 15]. This is an area that has been addressed for the Grid in Ref [16] and also in a seminal paper by Foster and collaborators [17] addressing broad support for communities.

A P2P Grid would comprise services that include those of Grids and P2P networks while naturally supporting environments that have features of both limiting cases. We can discuss two examples where such a model is naturally applied. In the High Energy Physics data analysis (e-Science [18]) problem discussed in [19], the initial steps are dominated by the systematic analysis of the accelerator data to produce summary events roughly at the level of sets of particles. This Grid-like step is followed by "physics analysis", which can involve many different studies and much debate between involved physicists regarding the appropriate methods to study the data. Here we see some Grid and some P2P features. As a second example, consider the way one uses the Internet to access information – either news items or multimedia entertainment. Perhaps the large sites like Yahoo, CNN and future digital movie distribution centers have Grid like organization. There are well-defined central repositories and high performance delivery mechanisms involving caching to support access. Security is likely to be strict for premium channels. This structured information is augmented by the P2P mechanisms popularized by Napster with communities sharing MP3 and other treasures in a less organized and controlled fashion. These simple examples suggest that whether for science or commodity communities, information systems should support both Grid and P2P capabilities [20,21].

The proposed P2P grid, which integrates the evolving ideas of computational grids, distributed objects, web services, P2P networks and message oriented middleware, comprises 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 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. Such P2P Grids should seamlessly integrate users to themselves and to resources, which are also linked to each other. We can abstract such environments as a distributed system of "clients" which consist either of "users" or "resources" or proxies thereto. These clients must be linked together in a flexible fault tolerant efficient high performance fashion. The

messaging infrastructure linking clients (both users and resources of course) would provide the backbone for the P2P grid.

The smallest unit of this messaging infrastructure should be able to intelligently process and route messages while working with multiple underlying communication protocols. We refer to this unit as a *broker*, where we avoid the use of the term *server*s to distinguish it clearly from the application servers that would be among the sources/sinks to messages generated within the integrated system. For our purposes (registering, transporting and discovering information), we use the term events/messages interchangeably where events are just messages – typically with time stamps. We may enumerate the following requirements for the messaging infrastructure –

1. *Scaling*: This is of paramount importance considering the number of devices, clients and services that would be aggregated in the P2P grid. The distributed broker network should scale to support the increase in these aggregated entities. However the addition of brokers to aid the scaling should not degrade performance by increasing communication *pathlengths* or ineffective bandwidth utilizations between broker nodes within the system. This calls for efficient organization of the broker network to ensure that the aforementioned degradations along with concomitant problems such as increased communication latencies do not take place.
2. *Efficient disseminations*: The disseminations pertain to routing content, queries, invocations etc. to the relevant destinations in an efficient manner. The routing engine at each broker needs to ensure that the paths traversed within the broker network to reach destinations are along efficient paths that eschew failed broker nodes.
3. *Guaranteed delivery mechanisms*: This is to ensure persistent delivery and reliable transactions within P2P grid realms.
4. *Location independence*: To eliminate bandwidth degradations and bottlenecks stemming from entities accessing a certain known broker over and over again to gain access to services, it must be ensured that any broker within the broker network is just as good as the other. Services and functionality would then be accessible from any point within the broker network.
5. *Support for P2P interactions*: P2P systems tend to be autonomic, obviating the need for dedicated management. P2P systems incorporate sophisticated search and subsequent discovery mechanisms. Support for P2P interactions facilitates access to information resources and services hosted by peers at the "edge" of the network.
6. *Interoperate with other messaging clients*: Enterprises have several systems that are built around messaging. These clients could be based on enterprise vendors such as IBM's MQSeries or Microsoft's MSMQ. Sometimes these would be clients conforming to mature messaging specifications such as the Java Message Service (JMS) [22]. JMS clients, existing in disparate enterprise realms, can utilize the distributed broker network as a JMS provider to communicate with each other.
7. *Communication through proxies and firewalls*: It is inevitable that the realms we try to federate would be protected by firewalls stopping our elegant application channels dead in their tracks. The messaging infrastructure should thus be able to communicate across firewall, DHCP and NAT boundaries. Sometimes communications would also be through authenticating proxies.
8. *Extensible transport framework*: Here we consider the communication subsystem, which provides the messaging between the resources and services. Examining the

growing power of optical networks we see the increasing universal bandwidth that in fact motivates the thin client and server based application model. However the real world also shows slow networks and links(such as dial-ups), leading to a high fraction of dropped packets. We also see some chaos today in the telecom industry which is stunting, somewhat, the rapid deployment of modern "wired' (optical) and wireless networks. We suggest that key to future federating infrastructures will be messaging subsystems that manage the communication between external resources, services and clients to achieve the highest possible system performance and reliability. We suggest this problem is sufficiently hard that we only need solve this problem "once" i.e. that all communication – whether TCP/IP, UDP, RTP (A Transport Protocol for Real-Time Applications) [23], RMI, XML/SOAP [24] or you-name-it be handled by a single messaging or event subsystem.

9. *Ability to monitor the performance of P2P grid realms*: State of the broker network fabric provides a very good indicator of the state of the P2P grid realm. Monitoring the network performance of the connections originating from individual brokers enables us to identify bottlenecks and performance problems, if any, which exist within a P2P grid realm.
10. *Security Infrastructure*: Since it is entirely conceivable that messages (including queries, invocations and responses) would have to traverse over hops where the underlying communication mechanisms are not necessarily secure, a security infrastructure that relies on message level security needs to be in place. Furthermore, the infrastructure should incorporate an authentication and authorization scheme to ensure restricted access to certain services. The infrastructure must also ensure a secure and efficient distribution of keys to ensure access by authorized clients to content encapsulated in encrypted messages.

In this paper we base our investigations on our messaging infrastructure, NaradaBrokering [25-31], which addresses or provides the foundations for the issues discussed above. The remainder of this paper is organized as follows. In section 2.0 we discuss broker network organization, routing of events and support for durable interactions in the NaradaBrokering system. Section 3.0 presents the rationale, and our strategy, to support P2P interactions. Section 4.0 presents an extensible transport framework that addresses the transport issues alluded to earlier. A performance aggregation framework for monitoring and responding to changing network conditions is discussed in Section 5.0. Section 6.0 presents an overview of the message based security framework in the system. Finally, in section 7.0 we present our conclusions and outline future work.

2 NaradaBrokering

To address the issues [31] of scaling, load balancing and failure resiliency, NaradaBrokering is implemented on a network of cooperating brokers. Brokers can run either on separate machines or on clients, whether these clients are associated with users or resources. This network of brokers will need to be dynamic for we need to service the needs of dynamic clients. Communication within NaradaBrokering is asynchronous and the system can be used to support different interactions by encapsulating them in specialized events. 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.

NaradaBrokering goes beyond other operational publish/subscribe systems [32-37] in many (support for JMS, P2P interactions, audio-video conferencing, integrated performance monitoring, communication through firewalls among others) ways. The messaging system must scale over a wide variety of devices – from hand held computers at one end to high performance computers and sensors at the other extreme. We have analyzed the requirements of several Grid services that could be built with this model, including computing and education. Grid Services (including NaradaBrokering) being deployed in the context of Earthquake Science can be found in [29]. NaradaBrokering supports both JMS and JXTA [44] (from juxtaposition), which are publish/subscribe environments with very different interaction models. NaradaBrokering also provides support for legacy RTP clients.

2.1 Broker Organization

Uncontrolled broker and connection additions result in a broker network susceptible to network-partitions and devoid of any logical structure thus making the creation of efficient broker network maps (BNM) an arduous if not impossible task. The lack of this knowledge hampers the 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.

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" [38,39] where the average communication "pathlengths" between brokers increase logarithmically with geometric increases in network size, as opposed to exponential increases in uncontrolled settings. This 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.

2.2 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 destina-

tions (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 updated to snapshot its dissemination within the network. The event dissemination traces eliminate continuous echoing and in tandem with the BNM –computes 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 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 eschewing links and brokers that have failed or have been failure-suspected.

In NaradaBrokering topics could be based on tag-value pairs, Integer and String values. Clients can also specify SQL queries on properties contained in a JMS message. Finally, NaradaBrokering currently incorporates a distributed XML matching engine, which allows clients to specify subscriptions in XPath queries and store advertisements in XML encapsulated events. Real-time XML events are evaluated against the stored XPath subscriptions, while stored XML advertisements are evaluated against a real-time XPath query for discovery purposes.

Figures 2 and 3 illustrate some results [14] 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 1. 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 more than one broker node process. 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 machines involved in the experiment reside on a 100 Mbps network.

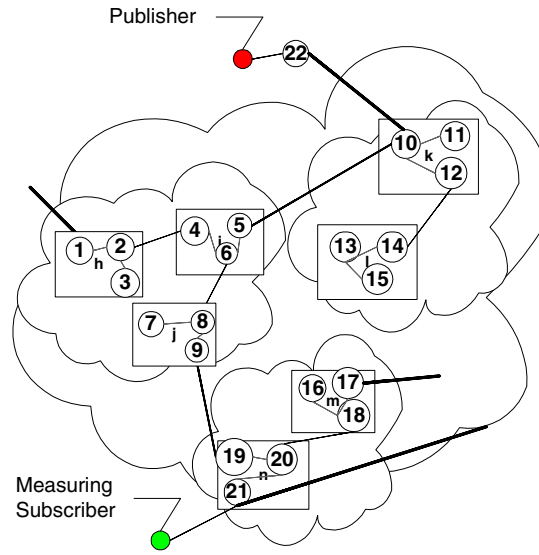

Fig. 1. The NaradaBrokering Test Topology

We measure the latencies at the client under varying conditions of publish rates, event sizes and matching rates. In most systems where events are continually generated a "typical" client is generally interested in only a small subset of these events. This behavior is captured in the matching rate for a given client. Varying the match-

ing rates allows us to perform measurements under conditions of varying selectivity. The 100% case corresponds to systems that would flood the broker network. In systems that resort to flooding (routing a message to every router node) the system performance does not vary with changes in the match rate. Furthermore, in most cases a given message would only be routed to a small set of targeted client nodes.

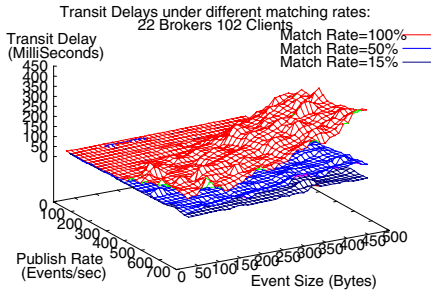

Fig. 2. NaradaBrokering Performance at match rates of 100%, 50% and 15%

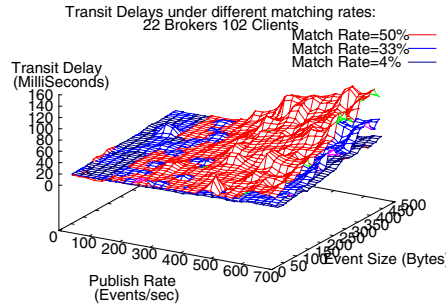

Fig. 3. NaradaBrokering Performance at match rates of 50%, 33% and 4%

As the results demonstrate, the system performance improves significantly with increasing selectivity from subscribers. The distributed broker network scaled well, with adequate latency, unless the system became saturated at very high publish rates.

2.3 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.

2.4 JMS Compliance

NaradaBrokering is JMS compliant and provides support not only for JMS clients, but also for replacing single/limited server JMS systems transparently [28] with a distributed NaradaBrokering broker network. Since JMS clients are vendor agnostic, this JMS integration has provided NaradaBrokering with access to a plethora of applications built around JMS, while the integrated JMS solution provides these applications with scaling, availability and dynamic real time load balancing. Among the applica-

tions ported to this solution are the Anabas distance education conferencing system [40] and the Online Knowledge Center (OKC) portal [41].

2.4.1 JMS Performance Data

To gather performance data, we run an instance of the SonicMQ (version 3.0) [42] broker and NaradaBrokering broker on the same dual CPU (Pentium-3, 1 GHz, 256MB) machine. We then setup 100 subscribers over 10 different JMS TopicConnections on another dual CPU (Pentium-3, 866MHz, 256MB) machine. There is also a *measuring* subscriber and a publisher that are set up on a third dual CPU (Pentium 3, 866MHz, 256MB RAM) machine. The three machines (residing on a 100 Mbps network) have Linux (version 2.2.16) as their operating system. The runtime environment for all the processes is Java 2 JRE (Blackdown-FCS).

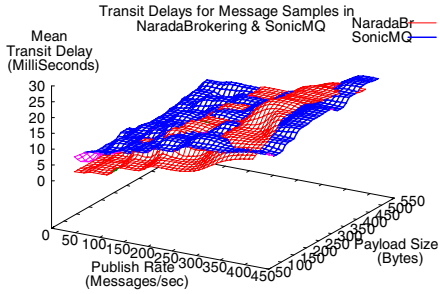

Fig. 4. Transit Delays for messages

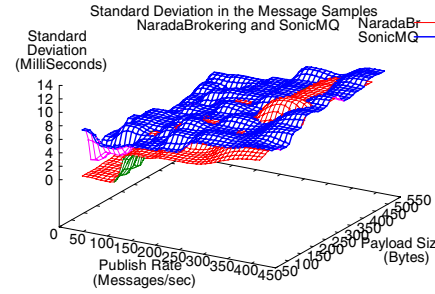

Fig. 5. Standard Deviation for messages

The topic, which the subscribers subscribe to and the publisher publishes to, is the same. We vary the rates at which the publisher publishes messages while varying the payload sizes associated with these messages. We compute the transit delays associated with individual messages and also the standard deviation in the delays (used to compute the mean transit delay) associated with messages in a given test case. Figure 4 depicts the mean transit delays for the measuring subscriber under NaradaBrokering and SonicMQ for high publish rates and smaller payload sizes. Figure 5 depicts the standard deviation associated with message samples under the same conditions.

As can be seen NaradaBrokering compares very well with SonicMQ. Also, the standard deviation associated with message samples in NaradaBrokering were for the most part lower than in SonicMQ. Additional results can be found in [28].

3 Support for P2P Interactions in NaradaBrokering

Issues in P2P systems pertaining to the discovery of services and intelligent routing can be addressed very well in the NaradaBrokering 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 provide easy access to 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.

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. Furthermore, 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. 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 satisfy each other's requests, obviating the need for funneling interactions through the broker network. Systems tuned towards large-scale P2P systems include

Pastry [43] 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 [44] project at Sun Microsystems is another effort to provide such large-scale P2P infrastructures.

3.1 JXTA

JXTA is a set of open, generalized protocols [45] to support P2P 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 dissemination. Time-to-live (TTL) indicators and peer traces attenuate interaction propagations. JXTA interactions are unreliable and tend to be localized. 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.

3.2 JXTA & NaradaBrokering

In our strategy for providing support for P2P interactions within NaradaBrokering, we impose two constraints. First, we make no changes to the JXTA core and the associated protocols. We make additions to the rendezvous layer for integration purposes. Second, 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. 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.

3.2.1 The Interaction Model

Different JXTA interactions are queued at the queues associated with the relevant layers comprising the JXTA protocol suite. 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.

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.

3.2.2 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 [46] (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 and is responsible for forwarding 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 –

1) Peer group advertisements
2) Requests from peers to be part of a certain peer group and responses to these requests
3) Messages sent to a certain peer group or a targeted peer
4) 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 deserialized and the interaction is propagated as outlined in the proxy's dual role as a rendezvous peer. Additional details pertaining to this integration can be found in [27].

3.3 Performance Measurements

For comparing JXTA performance in NaradaBrokering we setup the topologies depicted in Figure 6. We then compare the performance of the pure JXTA environment, the integrated Narada-JXTA system and the native NaradaBrokering system. The rendezvous peers connected to brokers in topology 6.(b) are Narada-JXTA proxies.

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. 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 7 and 8 depict the mean transit delay and standard deviation for the message samples under the different test topologies. These results indicate the superior performance of the integrated NaradaJXTA system compared to that of the pure JXTA system. The results [27] follow the same general pattern for measurements under other test topologies.

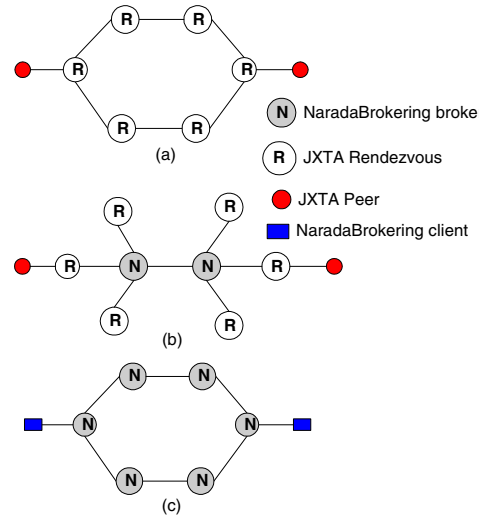

Fig. 6. The JXTA Test Topologies

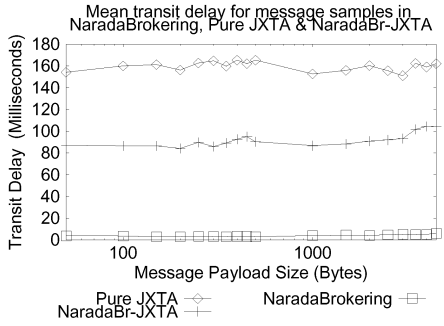

Fig. 7. Mean Transit Delay for samples

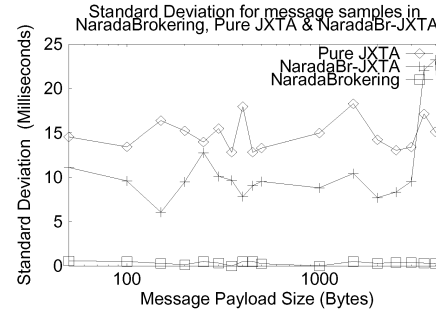

Fig. 8. Standard Deviation for samples

4 NaradaBrokering's Transport Framework

In the distributed NaradaBrokering setting it is expected that when an event traverses an end-to-end *channel* across multiple broker *hops* or *links* the underlying transport protocols deployed for communications would vary. The NaradaBrokering Transport

framework aims to abstract the operations that need to be supported for enabling efficient communications between nodes. These include support for –

1) Easy addition of transport protocols within the framework.
2) Deployments of specialized links to deal with specific data types.
3) Negotiation of the best available communication protocol between two nodes
4) Adaptability in communications by responding to changing network conditions.
5) Accumulating performance data measured by different underlying protocol implementations.

TCP, UDP, Multicast, SSL, HTTP and RTP based implementations of the transport framework are currently available in NaradaBrokering. It is also entirely conceivable that there could be a JXTA link, which will defer communications to the underlying JXTA pipe mechanism. NaradaBrokering can also tunnel through firewalls such as Microsoft's ISA [47] and Checkpoint [48] and proxies such as iPlanet [49]. The user authentication modes supported include Basic, Digest and NTLM. Operations that need to be supported between two communication endpoints are encapsulated within the "link" primitive in the transport framework. The adaptability in communications is achieved by specifying network constraints and conditions under which to migrate to another underlying protocol. For e.g. a UDP link may specify that when the loss rates increase substantially communication should revert to TCP. Though there is support for this adaptability in the transport framework, this feature is not yet implemented in the current release. Figure 9 provides an overview of the NaradaBrokering transport framework.

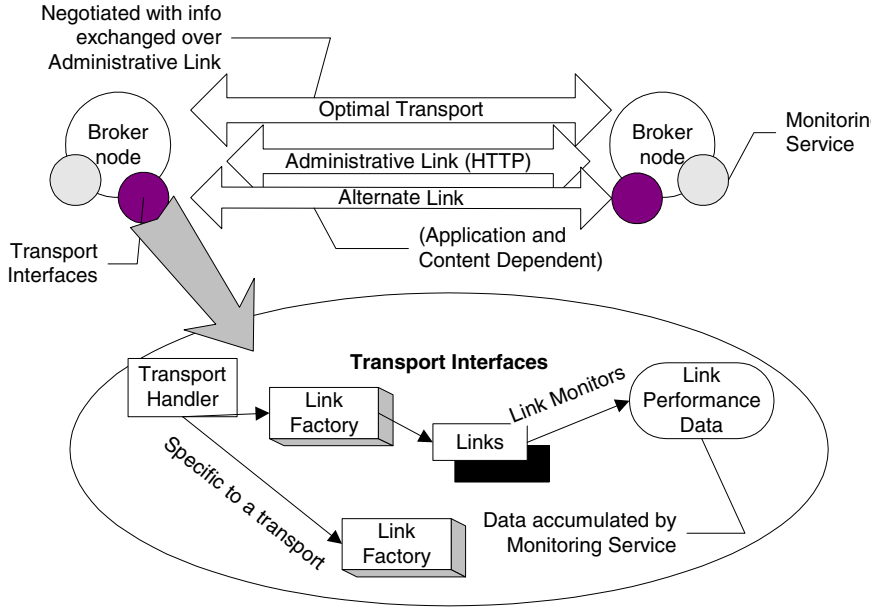

Fig. 9. Transport Framework Overview

A **Link** is an abstraction that hides details pertaining to communications. A Link has features, which allow it to specify a change in the underlying communications and the conditions under which to do so. An implementation of the Link interface can incorporate its own handshaking protocols for setting up communications. The Link also contains methods, which allow for checking the status of the underlying communication mechanism at specified intervals while reporting communication losses to the relevant error handlers within the transport framework. Each implementation of the Link interface can expose and measure a set of performance factors. Measurement of performance factors over a link requires cooperation from the other end-point of the communication link; this particular detail should be handled within the Link implementation itself. How the Link implementation computes round trip delays, jitter factors, bandwidth, loss rates etc. should be within the domain of the implementer. The Link also has methods which enable/disable the measurement of these performance factors. Links expose the performance related information in the **LinkPerformanceData** construct using which it is possible to retrieve information (*type*, *value*, *description*) pertaining to the performance factors being measured.

In the distributed NaradaBrokering setting it is expected that when an event traverses across multiple broker hops it could be sent over multiple communication links. In places where links optimized to deal with the specialized communication needs of the event exist (or can exist) they will be used for communications. While routing events between two NaradaBrokering brokers (that already have a link established between them) it should be possible for the event routing protocol to specify the creation of alternate communication links for disseminations. Support for this feature arises when routing handlers request the deployment of specific transport protocols for routing content, for e.g. a NaradaRTP event router could request that RTP links be used for communication. Sometimes such links will be needed for short durations of time. In such cases one should be able to specify the time for which the link should be kept alive. Expiry of this timer should cause the garbage collection of all resources associated with the link. The *keepalive* time corresponds to the period of inactivity after which the associated link resources must be garbage collected.

All broker locations need not have support for all types of communication links. Information regarding the availability of a specific link type could be encapsulated in an URI. This information could be exchanged along with the information regarding supported link types (at a given node) exchanged over the **AdministrativeLink**, which is different from that of a link in the methods that can be invoked on it. This URI could then possibly be used to dynamically load services. The AdministrativeLink exchanges information regarding the various communication protocols (along with information pertaining to them such as server, port, multicast group etc) that are available at a broker/client node. This is then used to determine the best link to use to communicate with the broker. Communication over the AdministrativeLink will be HTTP based to ensure the best possibility for communications between two nodes. All link implementations need to have an implementation of the **LinkNegotiator** interface. Based on the information returned on the AdministrativeLink, the LinkNegotiators are initialized for the common subset of communications and then deployed to negotiate the transport protocol for communications. The LinkNegotiator determines whether communication is possible over a specified link and also returns metrics that would enable the AdministrativeLink in arriving at a decision regarding the deployment of the best possible link.

All links of a specific communications *type* are managed by a **LinkFactory** instance. The LinkFactory for a particular communications protocol enables communications to and from other nodes over a specific link type. The LinkFactory also controls the intervals at which all its *managed* links check their communication status. Links also allow the specification of constraints (usually on the set of performance factors that it measures) and the link type that the communication must migrate to when those conditions are satisfied. This feature allows a link to revert to an alternate underlying transport protocol when communication degrades or is impossible to achieve. For example, it is conceivable that while communicating using TCP, bandwidth and latency constraints force a switch to UDP communications. The LinkFactory is also used to manage the migration of communication protocols from links of different types. Based on the set of supported communication protocol migrations, which a LinkFactory exposes, adaptive communications between nodes is enabled.

Protocol layers use the **TransportHandler** interface to invoke methods for communications with other NaradaBrokering nodes. LinkFactories are loaded at run-time by the TransportHandler implementation and it is then that TransportHandler interface is passed to the LinkFactory implementation. The reference to the transport handler is passed to every link created by the link factory. This is the reference that is used by individual links to report the availability of data on a link. Individual links use this interface to report data streams that are received over the link, loss of communications and requests to migrate transport protocols if the migration constraint is satisfied. Based on the LinkFactories that are loaded at run-time the transport handler can expose the set of link types (generally corresponding to transport types) that it supports. Transport Handler manages all Link factories and Links. LinkFactories are responsible for the creation of links. Links have methods for sending data (while also indicating the data type). Data received on a communication link is reported to the TransportHandler by invoking the appropriate methods within the interface.

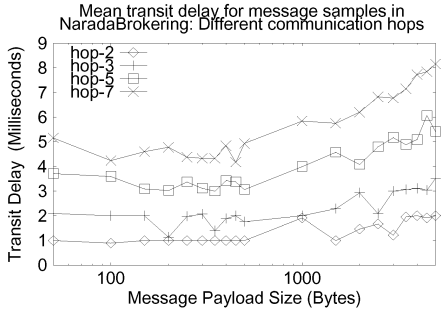

Fig. 10. Transit Delay for message samples

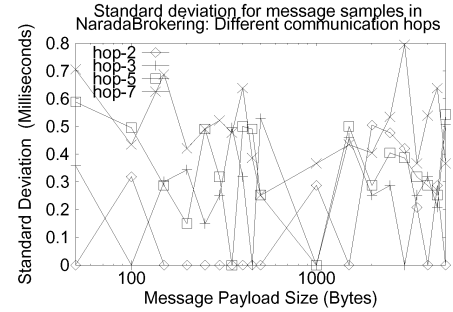

Fig. 11. Standard deviation for samples

4.1 Some Performance Measurements

Figures 10 and 11 depict results for the TCP implementation of the framework. The graphs depict the mean transit delays, and the accompanying standard deviations, for native NaradaBrokering messages traversing through multiple (2, 3, 5 and 7) hops with multiple brokers (1, 2, 4 and 6 respectively) in the path from the sender of the message to the receiver. For each test case the message payload was varied. The tran-

sit delay plotted is the average of the 50 messages that were published for each payload. The sender/receiver pair along with every broker involved in the test cases were hosted on different physical machines (Pentium-3, 1 GHz, 256 MB RAM). The machines reside on a 100 Mbps LAN. The run-time environment for all the processes is JRE-1.3 build Blackdown-1.3.1, Red Hat Linux 7.3

The average delay per inter-node (broker-broker, broker-client) hop was around 500-700 microseconds. The standard deviation varies from 0 microseconds for 50 byte messages traversing a hop to 800 microseconds over 7 hops.

5 Performance Monitoring and Aggregation

The performance monitoring scheme within the distributed broker network needs to have two important characteristics. First, it should be able to work with different transport protocols with no straitjacketing of the performance factors being measured. The Link and LinkPerformanceData primitives that abstract transport details and performance data respectively, as outlined in the preceding section, ensure the ability to work with unlimited performance factors over different transport protocols. Different nodes, with different types of links originating from them, can end up measuring a different set of performance factors. Second, the scheme should be to federate with other network measurement services such as the network weather service (NWS) [50]. An added feature would be to allow administrators to monitor specific realms or domains.

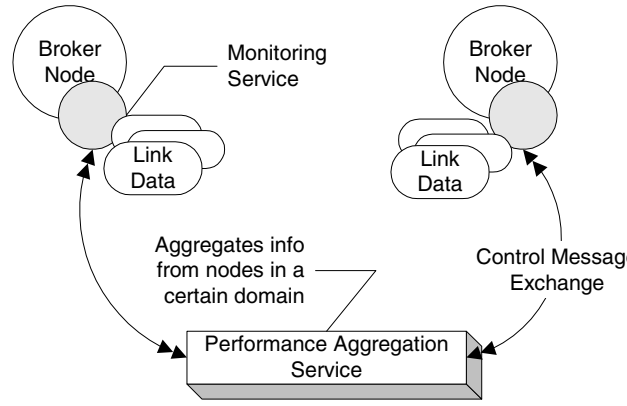

Fig. 12. Performance Aggregation Overview

Every broker in NaradaBrokering incorporates a monitoring service (as shown in Figure 12) that monitors the state of the links originating from the broker node. Metrics computed and reported over individual links, originating from a broker node, include *bandwidth*, *jitter*, *transit delays*, *loss rates* and *system throughputs*. Factors are measured in a non-intrusive way so as to ensure that the measurements do not further degrade the metrics being measured in the first place. Factors such as bandwidth measurements, which can pollute other metrics being measured, are measured at lesser frequencies. Furthermore, once a link is deemed to be at the extreme ends of the performance spectrum (either very good or very bad) the measurement of certain factors are turned off while others are measured at a far lower frequency. Each link can measure different set of parameters. So the set of parameters being measured would be extensible and flexible. The monitoring service that runs at every node encapsulates performance data gathered from each link in an XML structure. The moni-

toring service then reports this data to a performance aggregator node, which aggregates information from monitoring services running at other nodes.

Performance aggregators monitor the state of the network fabric at certain realms; the aggregators themselves may exchange information with each other to provide a state of the integrated network realm. The performance aggregators exchange information with the monitoring services pertaining to the measurement and reporting of performance factors. For example, the aggregator can instruct the monitoring service running at a broker node to stop (or modify the intervals between) the measurement of certain factors. Similarly, an aggregator may instruct the monitoring service to report only certain performance factors and that too, only if the factors have varied by the amount (absolute value or a percentage) specified in it's request.

Information accumulated within the aggregators is accessible to administrators via a portlet residing in a portal such as Apache Jetspeed [51]. Note that, since the information returned to the aggregators in encapsulated in an XML structure, it is very easy to incorporate results gathered from another network monitoring service such as NWS. All that needs to be done is to have a proxy, residing at a NWS node that encapsulates the monitored data into an XML structure. The aggregated XML performance data (from the monitoring service at each node and other third-party services) would be mined to generate information, which would then be used to achieve to certain objectives.

(a) *The ability to identify, circumvent, project and prevent system bottlenecks:* Different transports would reveal this in different ways. As system performance degrades UDP loss rates may increase, TCP latencies increase. Similarly as available bandwidths decrease the overheads associated with TCP error correction and in order delivery may become unacceptable for certain applications.

(b) *To aid routing algorithms:* Costs associated with link traversals in BNM's would be updated to reflect the state of the fabric and the traversal times associated with links in certain realms. Routes computed based on this information would then reveal "true" faster routes.

(c) *To be used for Dynamic topologies to address both (a) and (b):* The aggregated performance information would be used to identify locations to upgrade the network fabric of the messaging infrastructure. This upgrade would involve brokers/connections be instantiated/purged dynamically to assuage system bottlenecks and to facilitate better routing characteristics. Although multicasting and bandwidth reservation protocols such as RSVP [52] and ST-II [53] can help in better utilizing the network they require support at the router level, more conceited effort is need at higher levels, and dynamic topologies coupled with efficient routing protocols can help in the efficient utilization of network resources.

(d) *To determine the best available broker to connect to:* Based on the aggregated information it should be possible to determine the best broker that a client can connect to within a certain realm. Scaling algorithms, such as the one derived from item (c), would benefit greatly from this strategy by incorporating newly added broker nodes (which would be the best available ones) into the routing solution.

(e) *Threshold notifications:* Administrators can specify thresholds, which when reached by specific monitored factors, results in notifications being sent to them.

6 Security Framework

Since it is entirely conceivable that messages (including queries, invocations and responses) would have to traverse over hops where the underlying communication mechanisms are not necessarily secure, a security infrastructure that relies on message level security needs to be in place. The security framework in NaradaBrokering tries to address the following issues

1. *Authentication:* Confirm whether a user is really who he says he is.
2. *Authorization:* Identify if the user is authorized to receive certain events
3. *Key distribution:* Based on the authentication and authorization, distribute keys, which ensure that only the valid clients are able to decrypt encrypted data.
4. *Digital Signing:* Have the ability to verify the source of the event and whether the source is authorized to publish events conforming to the specified template.
5. *Communication Protocol Independence:* Have the ability to work over normal communication channels. Communications need not to be over unencrypted links.
6. *End-to-End integrity:* Ensure that the only place where the unencrypted event is seen at the authorized publisher of the event and the authenticated (and authorized) subscribers to the event.
7. *Detection of security compromise:* Check whether the publisher's signature is a valid one. This approach would be similar to the Certificate Revocation Lists (CRL) scheme.
8. *Qualities of Servic*e *detecting compromise:* Clients may be asked to answer questions to verify its authenticity at regular intervals to facilitate detection of compromise.
9. *Response to security compromise:* This would involve invalidating certain signatures and discarding the use of certain keys for encrypted communications.

In our approach we secure messages independently of any transport level security. This provides a fine-grained security structure suitable for distributed systems and multiple security roles. For example, parts of the message may be encrypted differently, allowing users with different access privileges to access different parts of the message. Basic security operations such as authentication should be performed in a mechanism-independent way, with specific mechanisms (Kerberos [54], PKI) plugged into specific applications. The message level security framework allows us to deploy communication links where data is not encrypted. Furthermore, this scheme also ensures that no node/unauthorized-entity ever sees the unencrypted message. In our strategy we incorporate schemes to detect and respond to security compromises while also dealing with various attack scenarios.

Security specifications for Web Services [55, 56] are just starting to emerge, but generally follow the same approach: the message creator adds a signed XML message containing security statements to the SOAP envelope. The message consumer must be able to check these statements and the associated signature before deciding if it can execute the request. Legion (http://www.cs.virginia.edu/ ~legion/) is a long-standing research project for building a "virtual computer" out of distributed objects running on various computing resources. Legion objects communicate within a secure messaging framework [57] with an abstract authentication/identity system that may use either PKI or Kerberos. Legion also defines an access control policy on objects. Additional details pertaining to the NaradaBrokering security infrastructure can be found in [58].

7 Conclusions and Future Work

This paper outlined an extensible messaging framework that, we propose, would be appropriate to host P2P grids. Our results demonstrate that the framework can indeed be deployed for both synchronous and asynchronous applications while incorporating performance-functionality trade-offs for different scenarios (centralized, distributed and peer-to-peer mode). We believe we are now well positioned to incorporate support, within the messaging infrastructure, for Web/Grid Services.

We have recently incorporated an XML matching engine within the distributed brokering framework. This allows us to facilitate richer discovery mechanisms. Trade-offs in performance versus functionality inherent in such matching engines is a critical area that needs to be researched further. Another area that we intend to investigate is the model of dynamic resource management. A good example of a dynamic peer group is the set of Grid/Web Services [59, 60] generated dynamically when a complex task runs – here existing registration/discovery mechanisms are unsuitable. A P2P like discovery strategy within such a dynamic group combined with NaradaBrokering's JMS mode between groups seems attractive. We have also begun investigations into the management of distributed lightweight XML databases using P2P search and discovery mechanisms. Another area amenable to immediate investigation and research is the federation of services in multiple grid realms.

Bibliography

1. The Grid Forum http://www.gridforum.org
2. GridForum Grid Computing Environment working group(http://www.computingportals.org) and survey of existing grid portal projects. http://www.computingportals.org/
3. "The Grid: Blueprint for a New Computing Infrastructure", Ian Foster and Carl Kesselman (Eds.), Morgan-Kaufman, 1998. See especially D. Gannon, and A. Grimshaw, "Object-Based Approaches", pp. 205-236, of this book.
4. Globus Grid Project http://www.globus.org
5. GriPhyN Particle Physics Grid Project Site, http://www.griphyn.org/
6. International Virtual Data Grid Laboratory at http://www.ivdgl.org/
7. NEES Earthquake Engineering Grid, http://www.neesgrid.org/
8. SCEC Earthquake Science Grid, http://www.scec.org
9. W. Johnston, D. Gannon, B. Nitzberg, A. Woo, B. Thigpen, L. Tanner, "Computing and Data Grids for Science and Engineering," Proceedings of Super Computing 2000.
10. DoE Fusion Grid at http://www.fusiongrid.org
11. Oram, A. (eds) 2001. Peer-To-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly, CA 95472.
12. Web Services Description Language (WSDL) 1.1 http://www.w3c.org/TR/wsdl
13. Definition of Web Services and Components http://www.stencilgroup.com/ideas_scope_200106wsdefined.html#whatare
14. Geoffrey Fox and Shrideep Pallickara, An Event Service to Support Grid Computational Environments. Concurrency and Computation: Practice and Experience. Volume 14(13-15) pp 1097-1129.
15. Fox, G. Report on Architecture and Implementation of a Collaborative Computing and Education Portal. http://aspen.csit.fsu.edu/collabtools/updatejuly01/erdcgarnet.pdf. 2001.
16. V. Mann and M. Parashar, Middleware Support for Global Access to Integrated Computational Collaboratories, Proc. of the 10th IEEE symposium on High Performance Distributed Computing (HPDC-10), CA, August 2001.

17. Ian Foster, Carl Kesselman, Steven Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations http://www.globus.org/research/papers/anatomy.pdf
18. Kingdom e-Science Activity http://www.escience-grid.org.uk/
19. Julian Bunn and Harvey Newman. Chapter on *Data Intensive Grids for High Energy Physics* in Grid Computing: Making the Global Infrastructure a Reality. Editors Berman, Fox and Hey. John Wiley. April 2003.
20. Hasan Bulut et al. An Architecture for e-Science and its Implications. Proceedings of the *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2002)* July 17 2002.
21. Geoffrey Fox, Ozgur Balsoy, Shrideep Pallickara, Ahmet Uyar, Dennis Gannon, and Aleksander Slominski, "Community Grids" invited talk at *International Conference on Computational Science*, April, 2002, Netherlands.
22. Java Message Service Specification". Mark Happner, Rich Burridge and Rahul Sharma. Sun Microsystems. 2000. http://java.sun.com/products/jms.
23. RTP: A Transport Protocol for Real-Time Applications (IETF RFC 1889) http://www.ietf.org/rfc/rfc1889.txt.
24. XML based messaging and protocol specifications SOAP. http://www.w3.org/2000/xp/.
25. The NaradaBrokering System http://www.naradabrokering.org
26. Geoffrey Fox and Shrideep Pallickara. "The Narada Event Brokering System: Overview and Extensions". Proceedings of the *International Conference on Parallel and Distributed Processing Techniques and Applications*, June 2002. pp 353-359.
27. Geoffrey Fox, Shrideep Pallickara and Xi Rao. "A Scaleable Event Infrastructure for Peer to Peer Grids". *Proceedings of ACM Java Grande ISCOPE Conference 2002*. Seattle, Washington. November 2002.
28. Geoffrey Fox and Shrideep Pallickara. "JMS Compliance in the Narada Event Brokering System". Proceedings of the *International Conference on Internet Computing*. June 2002. pp 391-402.
29. "Grid Services For Earthquake Science". Geoffrey Fox et al. *Concurrency & Computation: Practice and Experience*. 14(6-7): 371-393 (2002).
30. Hasan Bulut, Geoffrey Fox, Shrideep Pallickara, Ahmet Uyar and Wenjun Wu. "*Integration of NaradaBrokering and Audio/Video Conferencing as a Web Service*". Proceedings of the IASTED International Conference on Communications, Internet, and Information Technology, November, 2002, in St.Thomas, US Virgin Islands.
31. Geoffrey Fox and Shrideep Pallickara "An Approach to High Performance Distributed Web Brokering", *ACM Ubiquity* Volume2 Issue 38. November 2001.
32. Gurudutt Banavar, et al. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems.In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, Austin, Texas, May 1999.
33. Bill Segall and David Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings AUUG97*, pages 243–255, Australia, 1997.
34. Fiorano Corporation. A Guide to Understanding the Pluggable, Scalable Connection Management (SCM) Architecture - White Paper. Technical report, http://www.fiorano.com/products/fmq5 scm wp.htm, 2000.
35. Talarian Corporation. Smartsockets: Everything you need to know about middleware: Mission critical interprocess communication. Technical report, URL: http://www.talarian.com/products/smartsockets, 2000.
36. TIBCO Corporation. TIB/Rendezvous White Paper. Technical report, URL: http://www.rv.tibco.com/whitepaper.html, 1999.
37. The Object Management Group (OMG). OMG's CORBA Event Service. URL: http://www.omg.org/.
38. D.J. Watts and S.H. Strogatz. "Collective Dynamics of Small-World Networks". *Nature*. 393:440. 1998.

39. R. Albert, H. Jeong and A. Barabasi. “Diameter of the World Wide Web”. *Nature* 401:130. 1999.
40. The Anabas Conferencing System. http://www.anabas.com
41. The Online Knowledge Center (OKC) Web Portal http://ptlportal.ucs.indiana.edu
42. SonicMQ JMS Server http://www.sonicsoftware.com/
43. Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. Proceedings of Middleware 2001.
44. Sun Microsystems. The JXTA Project and Peer-to-Peer Technology http://www.jxta.org
45. The JXTA Protocol Specifications. http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html
46. Paul J. Leach and Rich Salz. Network Working Group. UUIDs and GUIDs. February, 1998.
47. Microsoft Internet Security and Acceleration (ISA) Server. http://www.microsoft.com/isaserver/
48. Checkpoint Technologies. http://www.checkpoint.com/
49. iPlanet. http://www.iplanet.com/
50. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing Rich Wolski, Neil Spring, and Jim Hayes, Journal of Future Generation Computing Systems,Volume 15, Numbers 5-6, pp. 757-768, October, 1999
51. Apache Jetspeed. http://jakarta.apache.org/jetspeed/site/index.html
52. Zhang, L. et al. “ReSource ReserVation Protocol (RSVP) – Functional Specification”, Internet Draft, March 1994.
53. Topolcic, C., “Experimental Internet Stream Protocol: Version 2 (ST-II)”, Internet RFC 1190, October 1990.
54. J. Steiner, C. Neuman, and J. Schiller. “Kerberos: An Authentication Service For Open Networked Systems”. In *Proceedings of the Winter 1988 USENIX Conference*.
55. B.Atkinson, et al. “Web Services Security (WS-Security) Version 1.0 05 April 2002,” Available from http://www-106.ibm.com/developerworks/webservices/library/ws-secure/.
56. “Assertions and Protocol for the OASIS Security Assertion Markup Language,” P. Hallam-Baker and E. Maler, eds. Available from http://www.oasis-open.org/committees/security/docs/ cs-sstc-core-01.pdf.
57. Adam Ferrari et al. "A Flexible Security System for Metacomputing Environments". (HPCN Europe 99), pp 370-380. April 1999
58. Pallickara et. al. A Security Framework for Distributed Brokering Systems available at http://www.naradabrokering.org
59. Semantic Web from W3C to describe self organizing Intelligence from enhanced web resources. http://www. w3c.org/2001/sw/
60. Berners-Lee, T., Hendler, J., and Lassila, O., "The Semantic Web," Scientific American, May2001.