On the Discovery of Topics in Distributed Publish/Subscribe systems

Shrideep Pallickara, Geoffrey Fox and Harshawardhan Gadgil
(spallick, gcf hgadgil)@indiana.edu
Community Grid Labs, Indiana University

1 Introduction

Messaging is a fundamental primitive in distributed systems. Entities communicate with each other through the exchange of messages, which can encapsulate information of interest such as application data, errors and faults, system conditions, search and discovery of resources. Messaging infrastructures can be based on several distinct paradigms viz. publish/subscribe systems, queuing systems, and peer-to-peer systems among others. Our work focuses on a systems based on the publish/subscribe paradigm.

The publish/subscribe paradigm is a powerful one and one in which there is a clear decoupling of the message producer and consumer roles that interacting entities/services might have. The routing of messages from the publisher to the subscriber is within the purview of the message oriented middleware (MOM), which is responsible for routing the right content from the producer to the right consumers. In publish/subscribe systems a subscriber registers its interest in events by subscribing to topics. In its simplest form these topics are typically "/" separated Strings, these have sometimes also been referred to as *subjects*. When a publisher issues events on a specific topic the middleware substrate routes the events to the subscribers that have registered an interest in this topic.

There are several standards/specifications that define the exchanges, control messages and communications within the publish/subscribe paradigm. This facilitates the development of interoperable systems based on conformance to these aforementioned specifications. Here we note three such specifications. First is the Java Message Service (JMS) [1] specification from Sun which defines a set of Java interfaces that enables the development of pub/sub applications. One of the excellent features of JMS is the ability that clients have of replacing the underlying provider with little or perhaps no change to the applications. Second, is the WS-Eventing [2] specification which enables web services to leverage the pub/sub paradigm; here, a *sink* service registers its interest (subscribes) in certain events with another *source* service (the publisher). When an event occurs the source routes the notification describing the occurrence to the registered sinks that had matching subscriptions. Finally there is WS-Notification [3], which is part of the Web Service Resource Framework (WSRF) [4] . WSRF is a realignment of the dominant Open Grid Service Infrastructure [5, 6] to be more in line with the emerging consensus [7] within the Web Services community. WS-Notification is sophisticated specification designed to meet the demands of modern Grid systems. WSN comprises WS-BaseNotification, WS-BrokeredNotification and WS-Topics.

In each of these systems interactions proceed with the assumption that the entity knows what the topic that it needs to subscribe to. Here we note that the WS-Notification specification incorporates the concept of topic spaces which facilitates the organization and categorization of topics. A topic space typically contains multiple trees of topics (some or all of which would be based on hierarchical topics). There could be multiple topic spaces each of which has its own namespace. The work that we present here can be used in tandem with the WS-Topics, and can be used by entities leveraging JMS, WS-Eventing or WS-Notification specifications.

In this paper we present a framework for the discovery of topics in publish/subscribe systems. This scheme facilitates the creation, advertisement and authorized discovery of topics by entities within the system. The discovery process is a distributed process and is resilient to failures that might take place within the system. In this paper we also include empirical results from our experiments related to the implementation of our scheme. We have performed these experiments in the context of our system NaradaBrokering [1-7] which is a distributed messaging infrastructure based on the publish/subscribe paradigm.

2 Brief Overview of NaradaBrokering

NaradaBrokering [8-11] is an open-source, distributed messaging infrastructure based on the publish/subscribe paradigm. The smallest unit of this distributed messaging substrate intelligently processes and routes messages, while working with multiple underlying communication protocols. We refer to this unit as a *broker*. The broker network in NaradaBrokering is based on hierarchical, cluster-based structure [8]. This cluster-based architecture allows NaradaBrokering to support large heterogeneous

client configurations. The routing of events within the substrate is very efficient [11] 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.

The substrate incorporates support for both JMS and the WS-Eventing specification. Work is currently underway on incorporating support for the WS-Notification suite of specifications. The NaradaBrokering substrate also incorporates support for WS-ReliableMessaging [12] and WS-Reliability [13] that facilitate reliable messaging between Web Services. Subscription formats supported within the substrate include "/" separate Strings, Integers, <tag, value> pairs, regular expressions, XPath and SQL queries. In NaradaBrokering entities can also specify constraints on the qualities of service (QoS) related to the delivery of events. The QoS pertain to the reliable delivery, playbacks, order, duplicate elimination, global timing services, security and size of the published events and their encapsulated payloads. Additional information regarding NaradaBrokering can be found in Refs [8-11].

3 Topic Creation request

When an entity is interested in creating a topic, it needs to create a topic creation request. A topic creation request generally includes information pertaining to the topic. Among the information encapsulated within the topic creation request are –

[1] Information regarding the creator and possibly the institution that the creator belongs to
[2] Information regarding the lifetime of this topic
[3] Information regarding the topic – topic descriptors.
[4] The topic synopsis type
[5] The placeholder topic synopsis information – This information is modified by the TDN as we will describe subsequently.

We refer to information regarding the topic as topic descriptors. Topic descriptors provide comprehensive information regarding the topic. This topic description information could be based on "/" separated strings organizing data in a hierarchical fashion. Topic descriptors could also be based on text which describes the data and keywords which facilitate easier descriptive information. In some cases topic descriptors could also be based on set of properties on which SQL queries can be specified. The descriptive information could also be organized based on an XML document on which XPath or XQuery queries could be specified. Finally, all of these topic descriptor formats could be searched based on regular expressions.

Topics provide a synopsis pertaining to the content of an event. In their simplest form this synopsis is based on "/" separated Strings. There are however systems based on the content-based variant of publish/subscribe systems where this synopses can take more complex forms such as a series of comma-separated <tag, value> pairs, XML Documents or a set of Properties. In its creation request the topic creator also needs to specify the synopsis type. Finally, in some cases the topic creator will include place holder information. The TDN adds information to this placeholder synopsis information. We will discuss this issue in a later section. For security purposes the entity might also include its credentials in this request.

4 Topic Discovery Nodes (TDN)

In our system TDNs are specialized nodes that keep track of topics contained within the system. There could be more than one TDN within the system. The information maintained within individual TDNs may overlap each other. However, we do not mandate that the information at individual TDNs be exact replicas of each other. Furthermore, some of the TDNs will be accessible without the need to present credentials, while some TDNs may base their responses on the credentials presented within the system. Similarly it is entirely conceivable that one might setup a TDN that manages discovery of topics created by entities within a certain administrative realm. The TDN serves three functions

a) As a repository of the topic information
b) Addition of system wide unique information to the topic synopsis
c) Signing credentials for the creator which would be used as a provenance regarding the topic ownership. This in turn would then be used to delegate authorizations regarding various publish/subscribe functions.

In order to receive entity interactions pertaining to topic discovery TDNs within the system subscribe to one or more topics of the form Services/Discovery/Topics, Services/Discovery/TopicDiscoveryNode, Services/Discovery/TopicDiscoveryNode/TDN-ID. The TDN-ID corresponds to the unique identifier associated with a specific TDN and facilitates targeted interactions with a given TDN.

4.1 Locating a TDN

Prior to propagating a topic creation request within the system the entity needs to locate a TDN. To locate a TDN the entity issues a TDN discovery request to the topic Services/Discovery/TopicDiscoveryNode. The

entity also includes its ID in this request. The entity must have already subscribed to Entity/EID where EID corresponds to the entity identifier associated with the entity in question. The entity may also include its credentials along with this request.

There could be specialized nodes which respond only to topic creation requests of a certain type. These TDNs might choose to store information regarding topics from a given institution or creator. The TDN may also choose to store information based on the topic description information contained in these requests.

4.1.1 Processing a discovery request

Upon receipt of this topic creation request a TDN may respond based on the credentials and the discovery response policy configuration associated with it. When a TDN does indeed choose to respond to such a discovery request it includes information regarding its identifier the TDN-ID which facilitates targeted interactions through messages published to the topic Services/Discovery/TopicDiscoveryNode/TDN-ID. The TDN might also include its credentials in its response.

4.1.2 Processing discovery responses

An entity may receive several responses to its TDN discovery request. Upon receipt of these responses the entity chooses one of the TDNs to communicate with. This choice would be based on the response times or the credentials associated with the responding TDN.

5 Topic Creation and Advertisements

In this section we outline issues related to the creation and advertisement of a topic.

5.1 Issuing the topic creation request

The client then proceeds to issue the topic creation request to the selected TDN. The entity can target this request to the TDN through the broker network by sending the request to the topic Services/Discovery/TopicDiscoveryNode/TDN-ID where TDN-ID corresponds to the identifier associated with the TDN. For security purposes the entity may sign the request and attach its credentials along with the message.

5.2 Processing a topic creation request

Upon receipt of a topic creation request the TDN generates a new UUID. This UUID will now be part of the topic synopsis. Once the TDN adds this UUID into the structure of the topic synopsis, the topic synopsis associated with the topic creation request is now unique. It must be noted that in some cases this qualifier will replace the synopsis originally in place. The topic synopsis structure associated with the topic will now be unique throughout the system. Next, the TDN takes all the information previously specified in the topic creation request, along with the modified template synopsis structure and proceeds to generate a hash (using SHA-256) and proceeds to sign this.

The reason we have UUID generation at the TDN is related to a security issue. Since the topic UUIDs can easily be discovered in the system, a malicious user can present this information to the TDN and request the TDN to issue credentials regarding the provenance of this topic. A given TDN has of course no way of determining if this UUID was generated by the entity in question or if the entity has simply gleaned this information through a previous discovery process and now wishes to claim possession of the topic. Our scheme for authorizations regarding subscriptions and publishing relies on delegated credentials and the vulnerability mentioned above would defeat our scheme. On the other hand UUID generation at the TDN prevents a malicious user from claiming some other topic as theirs.

The signature, the topic information supplied and the modified topic synopsis structure constitutes the topic advertisement. The topic advertisement is then encapsulated in a topic creation response and issued back to the entity through the broker network by publishing the topic creation response on Entity/EID.

5.3 Creating a topic advertisement

Upon receipt of the topic creation response, the entity gleans the topic advertisement. The entity then proceeds to advertise this information by issuing the advertisement to the topic Services/Discovery/Topics. This message is then delivered by the broker network to those TDNs that previously subscribed to it.

A TDN does not maintain any information regarding the topic creation that it serviced. Information about topics is stored based on the topic advertisement propagations within the system. TDNs maintain information regarding topic advertisements, regardless of whether these advertisements were created based on information provided by some other TDN. TDNs keep track of the topic advertisements that were advertised by entities. Of course different TDNs may have policies regarding the type of information that it stores and also regarding the provenance of these advertisements.

6 Topic Discovery

To discover topics an entity issues discovery requests. Upon receipt of responses to the discovery request, the entity is aware of constructing the right topic subscription requests.

6.1 Issuing a topic discovery request

An entity issues a topic discovery request for discovering topics. The request encapsulates a search query. This query could be a simple character String, an SQL query on certain properties, an XPath or XQuery query and finally even a regular expression. The discovery request would be sent to all TDNs within the broker network by issuing the discovery request to the topic Services/Discovery/Topics or to a specialized TDN by sending it to Services/Discovery/Topics/TDN-ID where TDN-ID corresponds to the identifier associated with a specific TDN. In most cases the best way to issue a discovery request would be to first do an asynchronous location of TDNs and then issue a discovery request to a subset of the available TDNs.

6.2 Processing a topic discovery request

Upon receipt of this discovery request at a TDN, a TDN will evaluate this query against the set of stored advertisements. Different TDNs may have different schemes for evaluating the search query on the set of stored advertisements. The advertisements could be searched based on their creation times, expiration times, duration of validity. Depending on the strategies deployed at various TDNs the response times for receipt of advertisements may vary.

6.3 Processing the topic discovery responses

The topic discovery responses encapsulate multiple topic advertisements. These advertisements, as previously articulated, contain topic synopsis information which can facilitate publishing/subscribing to topics. The topics that an entity subscribes to is based on the information contained in the received advertisements. It is possible that an entity may receive the same advertisement from multiple TDNs in which case one might have a duplicate list.

6.4 Responding to unavailability of TDNs

If none of the TDNs respond, an alternative option would be to flush the topic discovery request through the broker network and have individual entities respond to the request.

7 Security and Fault Tolerant Aspects of this approach

We first discuss the security issues pertinent to the discovery, subscription and issuing events conforming to a given template. Every aspect of the system from discovery of TDNs, advertisements, topic discovery, subscription and publication to topics can be secure. Every entity in the system has credentials associated with them. This is generally in the form of a certificate issued through some out-of-band methods. The certificates identify entities and TDNs. Upon receipt of topic creation requests the TDN signs the topic advertisement. TDNs within the system are the only ones authorized to sign topic advertisements.

The creator of a topic can, in turn, authorize users by signing their credentials to publish or subscribe to the topic that it created. The entity needs to present and demonstrate possession of the right credentials to be authorized for certain actions. TDNs will not respond to topic creation requests, or topic advertisement disseminations, if these actions are not accompanied by the right credentials. Similarly, a broker will simply check to see if the entity has the right credentials prior to disseminating its publish/subscribe actions within the broker network.

We now proceed to enumerate the fault tolerant aspects of our approach.

1. TDN Failures can occur, requests/responses to topic creation requests can be lost
2. Entity failures can occur at any time. Since the advertisements remain dormant until the entity issues an explicit advertisement dissemination request, so state is maintained during the creation process at any TDN within the system. This in turn implies that no garbage collection operations need to be performed within the system.
3. A TDN need not be active at all times. In fact the discovery scheme will work even if none of the TDNs are available online. In this case the discovery request will be propagated through the broker network
    a. Entities that are currently present and are topic creators can respond to this request. Of course there would be a TTL associated with these requests. Note that this scheme is akin to P2P requests.
4. Topic Advertisements /Discovery requests-responses can be lost en route to TDNs and entities alike.

8 Performance Measurements

We tested our system with two topologies. Table 1 summarizes the machines used for testing and their configuration.

Machine	Machine Specification	JVM Version

Complexity Indianapolis, IN, USA	SunOS complexity 5.9 Generic_112233-03 sun4u sparc SUNW,Sun-Fire-880	Java HotSpot(TM) Client VM (build 1.4.2-beta-b19, mixed mode)
Local Machine for testing, CGL, Bloomington, IN	Pentium 4, 1.6 GHz, 1 GB RAM, Win XP Professional Edition	Java HotSpot(TM) Client VM (build 1.4.2_06-b03, mixed mode)

Table 1 Machine Configuration

In the first topology, the Topic Discovery Node (TDN) and the Clients were placed on separate machines (Refer Figure 1). Figure 2 shows the topic creation time in this topology. This time includes the following steps. (1) Entity sends a TDN discovery request, (2) TDN sends an encrypted TDN Dsicovery response, (3) Entity sends an encrypted Topic Creation request, (4) TDN creates a Topic Advertisement, (5) TDN encrypts and sends the Topic Advertisement to the Entity, (6) Entity computes the digest of the Topic Advertisement, signs it with its private key and broadcasts a packet containing the Topic Advertisement, its X.509 Certificate and the signed digest over a pre-defined topic to all TDNs. Figure 3 shows the distribution of time spent in various activities in the TDN. For this topology, we also timed the process of discovering a topic. Figure 4 shows the time required when 1, 10 and 100 topics were discovered. Note that these experiments were carried out for string topic synopsis types only. Ref [11] contains a detailed discussion on other types of topic synopsis matching such as tag-value pairs, integers and regular expressions among others.

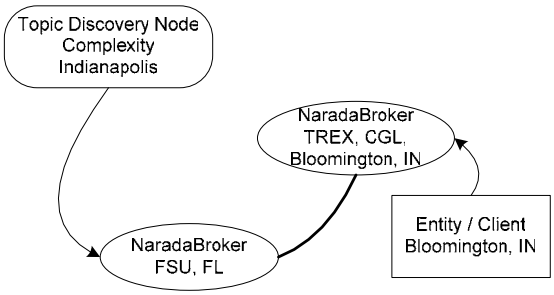

Figure 1 Wide Area Topology

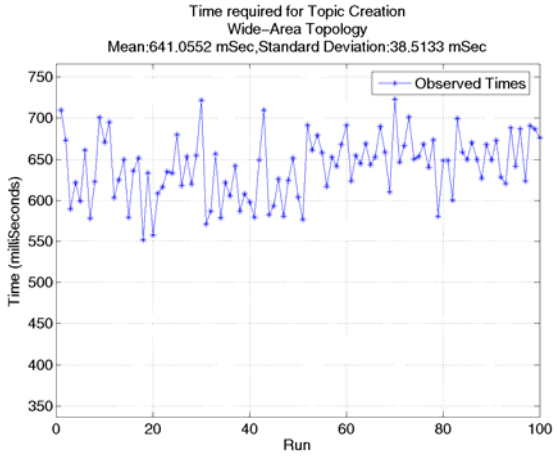

Figure 2 Topic Creation (Wide Area Topology)

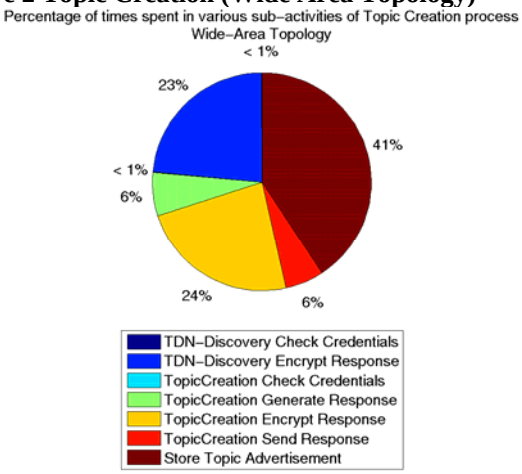

Figure 3 Time spent in Topic Creation sub-activities (Wide Area Topology)

Note that in Figure 4 the discovery time increases as more topics satisfying the specified criteria are discovered. Since the Topic Discovery responses are also encrypted, we conclude that the increase in time for overall topic discovery is because of the increase in time for encrypting bigger response packet.

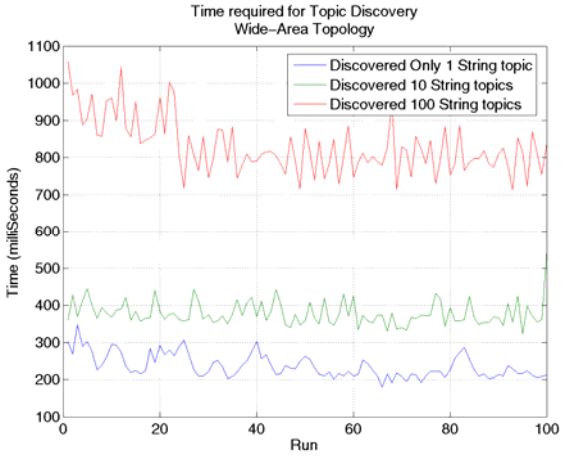

Figure 4 Topic Discovery (Wide Area Topology), String topics

Table 2 lists the metrics associated with wide area topology.

	All values in MilliSeconds				
	Mean	**Std. Dev.**	**Max**	**Min**	**Std. Error**
Topic Creation	641.06	38.51	723.18	551.63	3.85
Topic Discovery					
Discovering 1 Topic	236.86	32.01	348.88	178.95	3.20
Discovering 10 Topics	378.33	33.42	548.11	32..68	3.34
Discovering 100 Topics	829.69	73.61	1057.50	712.84	7.36

Table 2 Metrics for Topic Creation and Discovery (Wide Area Topology)

In our second topology (Figure 5), we ran the TDN and the clients on the same machine. Figure 6 - Figure 8 show the corresponding results.

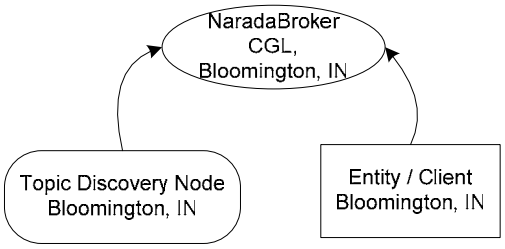

Figure 5 Local Machine Topology

Our scheme relies heavily on digital certificates and we observed that a majority of percentage of processing time of the TDN is spent in encrypting the messages. The encryption procedure is a two step process. In the first step a random SecretKey is generated and the TDN response is encrypted using this secret key. This secret key is then encrypted using the public key of the entity (topic creator or topic discovery client) in question. Further all topic advertisements are signed by the topic creator. A signed topic advertisement contains 3 parts, namely, the Topic Advertisement, the X.509 certificate of the creator and the signed digest of the topic advertisement.

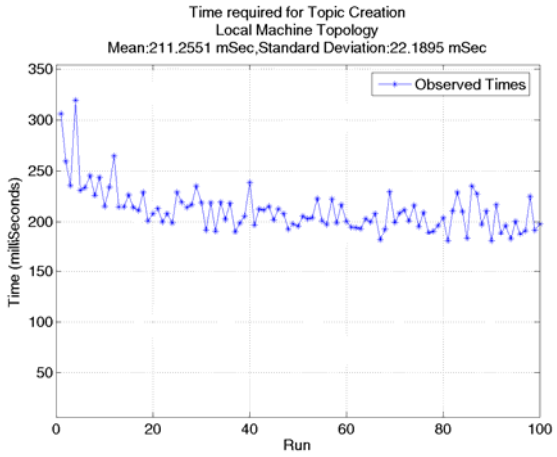

Figure 6 Topic Creation (Local Machine Topology)

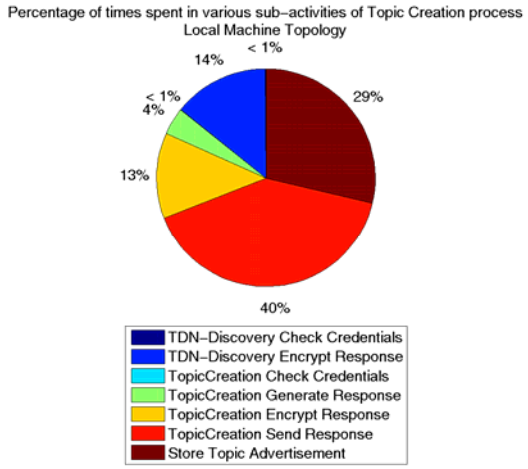

Figure 7 Time spent in Topic Creation sub-activities (Local Machine Topology)

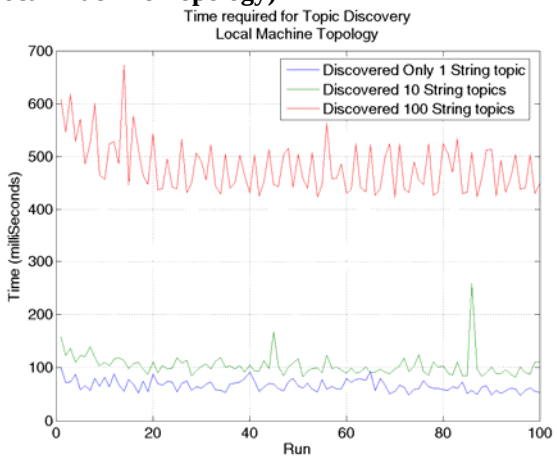

Figure 8 Topic Discovery (Local Machine Topology) , String Topics

Table 3 summarizes the metrics associated with the process in the local area topology.

	All values in MilliSeconds				
	Mean	**Std. Dev.**	**Max.**	**Min.**	**Std. Error**
Topic Creation	211.26	22.19	319.50	180.35	2.22
Topic Discovery					
Discovering 1 Topic	65.86	10.74	99.38	47.26	1.07
Discovering 10 Topics	104.25	21.49	258.50	81.29	2.15
Discovering 100 Topics	480.72	49.89	672.75	422.67	4.99

Table 3 Metrics for Topic Creation and Discovery (Local Machine Topology)

9 Conclusions

In this paper we presented our scheme to discover topics in distributed publish/subscribe settings. We also included performance measurements related to our implementation. There are a few advantages related to our approach which we enumerate below.

a) No singe point of failure: Any component within the system can fail. The scheme does not rely on any component being available at all times. This is more clearly outlined in the previous section.

b) Asynchronous discovery of topics: The discovery of topics is not time bound nor do we make any assumptions regarding the response times associated with these responses.

c) Discovery of topics can be based on a variety of search formats such as Regular Expressions, SQL queries or XPath Queries. It should be noted that neither the TDN which signed the advertisement nor the entity which initiated its creations needs to be present in the system for an entity to discover the topic advertisement.

d) Every interaction within the system can be secured and require demonstrating the possession of right credentials before any actions can proceed. The created topic advertisement can itself be secured by encrypting the advertisement with a symmetric key and securing this advertisement key with the creator's public key.

References

[1] Mark Happner, Rich Burridge and Rahul Sharma. Sun Microsystems. Java Message Service Specification. 2000. http://java.sun.com/products/jms

[2] Web Services Eventing. Microsoft, IBM & BEA. http://ftpna2.bea.com/pub/downloads/WS-Eventing.pdf

[3] Web Services Notification (WS-Notification). IBM, Globus, Akamai et al. http://www-106.ibm.com/developerworks/library/specification/ws-notification/

[4] [wsrf] The Web Services Resource Framework. http://www.globus.org/wsrf/

[5] [ogsa] I. Foster, C. Kesselman, J. Nick, S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration." Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002. Available from http://www.globus.org/research/papers/ogsa.pdf.

[6] [ogsi] The Open Grid Services Infrastructure (OGSI). http://www.gridforum.org/Meetings/ggf7/drafts/draft-ggf-ogsi-gridservice-23_2003-02-17.pdf

[7] [ws-arch] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture." W3C Working Group Note 11 February 2004. Available from http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

[8] Shrideep Pallickara and Geoffrey Fox. NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware-2003.

[9] Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil. Building Messaging Substrates for Web and Grid Applications. (To appear) in the Special Issue on Scientific Applications of Grid Computing in the Philosophical Transactions of the Royal Society of London 2005.

[10] Geoffrey Fox and Shrideep Pallickara. Deploying the NaradaBrokering Substrate in Aiding Efficient Web & Grid Service Interactions. Invited paper for Special Issue of the Proceedings of the IEEE on Grid Computing. Vol 93, No 3. pp 564-577. March 2005.

[11] Shrideep Pallickara and Geoffrey Fox. On the Matching Of Events in Distributed Brokering Systems. Proceedings of IEEE ITCC Conference on Information Technology. April 2004. pp 68-76 Volume II.

[12] Web Services Reliable Messaging Protocol (WS-ReliableMessaging) ftp://www6.software.ibm.com/software/developer/library/ws-reliablemessaging200403.pdf

[13] Web Services Reliable Messaging TC WS-Reliability. http://www.oasis-open.org/committees/download.php/5155/WS-Reliability-2004-01-26.pdf