On the Secure Creation, Organization and Discovery of Topics in Distributed Publish/Subscribe Systems

Shrideep Pallickara
Community Grids Lab, Indiana University,
Bloomington, IN 47404, USA
E-mail: spallick@indiana.edu

Geoffrey Fox
Community Grids Lab, Indiana University,
Bloomington, IN 47404, USA
E-mail: gcf@indiana.edu

Harshawardhan Gadgil
Community Grids Lab, Indiana University,
Bloomington, IN 47404, USA
E-mail: hgadgil@cs.indiana.edu

Abstract: Publish/Subscribe infrastructures have in the recent years gained significant traction with several specifications such as the Java Message Service, WS-Eventing and WS-Notification trying to capture the essence of publish/subscribe systems and enabling the development of interoperable systems. In this paper we present a scheme for the secure organization and discovery of topics in distributed publish/subscribe systems. The scheme outlined in this paper addresses security related issues such as authorization and provenance in the discovery of the aforementioned topics. We have also included results from our implementation of this scheme to demonstrate the feasibility of this mechanism. We have also included a scheme for the organization and discovery of topics that have hierarchical relationships and those that are part of a collection of topics in dynamic, high-flux publish/subscribe environments. The work that we describe here can be used in systems based on JMS, WS-Eventing or WS-Notification

Keywords: publish/subscribe, topics, topic discovery, authorization.

Reference to this paper should be made as follows: Pallickara, S., Fox, G. and Gadgil, H. (2006) 'On the Secure Creation, Organization and Discovery of Topics in Distributed Publish/Subscribe Systems', *Int. J. High Performance Computing and Networking*, **Vol. X, Nos. YY, pp.ZZ-ZZ**.

Biographical notes:

Shrideep Pallickara received his Masters and PhD from Syracuse University in 1998 and 2001 respectively. He is currently a Post Doctoral Researcher at the Community Grids Lab at Indiana University. His research interests include fault tolerant systems, P2P computing, Grid computing, Web Services and Security.

Geoffrey Fox received his PhD from Cambridge University in 1967. He is a Professor in the Department of Computer Science, School of Informatics and Physics at Indiana University. He is the Director of the Community Grids Lab at Indiana University. He was also the director of the Northeast Parallel Architectures Center at Syracuse University from 1990-2000.

Harshawardhan Gadgil is a Graduate Student at Indiana University, Bloomington. He is currently pursuing his PhD in Computer Science from Indiana University, Bloomington. He holds a B.E. (Bachelor of Engineering) in Computer Engineering from Mumbai University, Mumbai, India and a M.S. in Computer Science from Indiana University. His current research focuses on publish subscribe systems, Web Services and workflow technologies and management issues in large scale distributed systems.

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 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) (Happner et al., 2000) specification from Sun which defines a set of Java interfaces that enables the development of publish/subscribe 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 (Microsoft, et.al. 2004) 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 (IBM et al., 2004), which is part of the Web Service Resource Framework (WSRF) (Foster et. al, 2004). WSRF is a realignment of the dominant Open Grid Service Infrastructure (Foster et al., 2002; Tuecke, et al., 2003) to be more in line with the emerging consensus (Booth, et al., 2004) within the Web Services community. WS-Notification is sophisticated specification designed to meet the demands of modern Grid systems. WS-Notification 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 creation and discovery of topics in distributed 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 (Pallickara and Fox, 2003; Fox et al., 2005; Fox and Pallickara, 2005; Pallickara and Fox, 2004) which is a distributed messaging infrastructure based on the publish/subscribe paradigm.

This paper is an extended version of the paper that appeared in the ACM/IEEE GRID 2005 Workshop and which was subsequently invited for publication in this journal. In the extended version of this article we have added the management of hierarchical and collections of topics. This addresses a critical problem in publish/subscribe systems where there is a very large number of topics, and where it is difficult to manage the relationships that various topics have with each other. As far as we know, this work presents the first solution to the problem of facilitating the secure organization, management and discovery of collections of topics that have dynamic, real-time and continually changing spatial relationships with other topics. Furthermore, the hierarchical and collections of topics that we consider do not comprise a static set of topics but one which is very fluid and continues to evolve over time.

The remainder of this paper is organized as follows. In section 2 we provide a brief overview of the NaradaBrokering substrate which provided the basis for this work. In sections 3, 4, 5 and 6 we include various details pertaining to or topic creation and discovery scheme. Section 7 highlights the fault tolerance and security aspects of this scheme. Section 8 includes details about our scheme

to manage hierarchical and collections of topics. Section 9 includes a discussion of various performance measurements related to our implementation. Section 10 provides an overview of related work in this area. Finally, in section 11 we present our conclusions.

2 BRIEF OVERVIEW OF NARADABROKERING

NaradaBrokering (Pallickara and Fox, 2003; Fox et al., 2005; Fox and Pallickara, 2005; Pallickara and Fox, 2004) 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 (Pallickara and Fox, 2003). This cluster-based architecture allows NaradaBrokering to support large heterogeneous client configurations. The routing of events within the substrate is very efficient (Pallickara and Fox, 2004) 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 (BEA et al., 2004) and WS-Reliability (Fujitsu, 2004) that facilitates 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 (Pallickara and Fox, 2003; Fox et al., 2005; Fox and Pallickara, 2005; Pallickara and Fox, 2004).

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 in Section 5.2. 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, and 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 entity 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 MANAGING THE ORGANIZATION AND DISCOVERY OF TOPICS: HIERARCHICAL AND COLLECTIONS

In publish/subscribe systems there tend to several topics that exist within the system. Some of these topics have hierarchical relationships with each other and in some cases the topics have spatial relationships with other topics simply because they are part of the same application (for example the audio, video and chat streams in a collaborative application). Furthermore, relationships that topics have with each other continually evolve over the lifetime of a system. The management, organization and discovery of topics in such settings is a difficult problem and is exacerbated as the scale of the system increases. These issues tend to be dealt with in an ad-hoc, application specific manner. In this section we provide a framework for managing and organizing these topics. We also provide a

rich discovery scheme which can be leveraged to discover the inter-relationships that topics have with each other. In this section, we deal with the two scenarios that we outlined earlier. The first case pertains to the hierarchical organization of topics, while the second one pertains to managing collections of related topics.

8.1 Managing Hierarchical Topics

In the hierarchical organization of topics we deal with the management of topics that have hierarchical relationships with each other. It should be possible to discover the parent or child topics associated with a given topic. Traditionally, hierarchical organization of topics has been done using "/" separated Strings; here, a topic of the form A/B/C is interpreted to imply that A is the parent of topic B, while topic C is the child of topic B. In such systems, a subscription to the parent topic automatically implies subscriptions to all the child topics. However, subscription to a child topic does not imply subscription to any of the parent topics. Thus, if a user has subscribed to receive scores over the topic Scores/Sports/NBA it will receive scores from all NBA games; however, a user who has subscribed to receive scores for the Indiana Pacers basketball team through a topic Scores/Sports/NBA/IndianaPacers will receive scores related only to the Pacers.

In some cases, an entity may be interested in publishing streams related to a specific player. In this case, the entity may publish the scores on a topic such as Scores/Sports/NBA/IndianaPacers/ONeal. In general in the case of hierarchical topics subscriptions to the child topics provides access to finer granularity of the published content. Furthermore, subscriptions to parent topics typically result in higher traffic as compared to subscription to the child topics.

There are some disadvantages to schemes that are based on hierarchical topics. These include

1. A parent topic has no control over either the breadth, depth or the number of child topics within the topic hierarchy.
2. The topic string itself reveals the hierarchy of the topics. It is thus possible to launch denial of service attacks by simply flooding the system with messages to topics within the hierarchy.
3. Since no one really owns topics or enforces hierarchies, it is generally quite difficult to discover the structure of the topic hierarchy. This also results in the inability to enforce who would be authorized to discover such information.

In the scheme that we propose, we address these disadvantages.

8.1.1 The Topic Creation Request

In the topic creation request, the topic owner includes additional fields to facilitate hierarchical organization of topics. These include

1. *RootTopic*: This is true by default and specifies that the topic in question is the root topic of a topic hierarchy.
2. *Allow_Child_Topics*: This is false by default, and indicates if this node is willing to host child topics.

The other fields are present only if this node authorizes (or is authorized to have) child topics.

3. *MaxDepth*: the number of levels including this one that is allowed within the topic sub-tree that originates at this topic. A value of 0 for this field indicates that there are no restrictions on the depth.
4. *MaxWidth*: the total number of immediate child nodes that any topic within the topic sub-tree can have. A value of 0 for this field indicates that there are no restrictions on the depth.
5. *ParentTopic*: The topic creation request also includes the Topic which would be the parent of the topic in question.

The topic creation request also includes information about entities that are authorized, or barred, from either registering child topics or discovering the topic hierarchy. This is in addition to the information included in the topic creation request about the entities that are authorized to discover them.

8.1.2 Registration of Child Topics

To register a child topic, the owner of the child-topic first needs to discover the immediate-parent topic. Next, it needs to issue a topic creation request which specifies this topic as the parent topic to the topic that will be created at the TDN. Upon receipt of this request, the TDN checks to see if the entity is authorized to perform this operation and if this operation would violate any of the constraints specified by the parent topic any of the successive parent topics within the topic hierarchy. It should be noted that restrictions at the parent nodes take precedence over those specified by child nodes within the topic hierarchy. If the restrictions specified within a given child node violates any of the restrictions specified by the parent, the topic creation request results in a failure. If a child topic's registration breaks the *MaxDepth* restriction imposed by any of its immediate parents the registration request results in an error. Similarly, if the addition of topic breaks the *MaxWidth* restriction at its parent the registration request results in an error. For example if there is a topic T which is set up as a root topic with a *MaxDepth* of 2. If there is a request to register a child topic which specifies the *MaxDepth* as 2 or 0, that request results in an error. This is because it violates the maximum depth request specified in the root topic. If the entity is authorized to register a child topic and no constraints are violated within the topic hierarchy, the TDN generates a UUID that would be part of the topic synopsis as describe earlier. The TDN then also updates the hierarchical tree that it maintains internally to reflect the addition of this child topic.

The lifetime of a child topic is determined by the lifetime of its parent. When a topic expires, all child topics also expire. This is built into the topic advertisement. The expiration time associated with a child topic never exceeds that of any of the parent nodes within its topic hierarchy.

8.1.3 Discovery Operations

A child topic may further restrict its discovery in a manner which is consistent with restrictions specified by the root topic. As a general rule when an entity subscribes to a parent topic it can also request subscriptions to all child topics (or subscriptions to child topics up to a certain depth) within the topic hierarchy. A given topic will be part of exactly one topic hierarchy and has exactly one parent. Entities may also register for change notifications to the underlying topic hierarchy. Addition of child nodes to the topic hierarchy are automatically routed to these registered entities.

It should be noted that the topics themselves do not contain any visible footprints of the topic hierarchy. In the case of simple string topics, for example, every topic is based on a 128-bit UUID. There is thus no "/" separated string which reveals part of the topic hierarchy. Furthermore, if an entity is authorized to discover the topic hierarchy, that entity can discover the entire (or a specific sub-tree) topic hierarchy depending on discovery restrictions. Authorized entities are allowed to discover information about the topic hierarchy of a given topic. The discovery operations can target discovery of the immediate parent nodes or the entire set of child topics or both.

In some cases, if an entity's registration or discovery request is signed by the owner of the appropriate topic owner within the hierarchy, registration and discovery would be possible. Once again all exchanges between the entities are secured using a combination of symmetric and asymmetric key encryptions. In general topics at increasing depths of the topic hierarchy may add entities that are authorized to discover the topics. On the other hand discovery of the root topic within a deep topic hierarchy would probably be restricted to fewer entities.

8.2 Managing Collections of Topics

An application based on the publish/subscribe paradigm typically deals with a large number of topics. As an exemplar we consider a collaborative application that is based on the publish/subscribe paradigm. Further, this application consumes a wide variety of multimedia streams such as audio, video, whiteboards, images, shared displays and text. There is thus a collection of topics that an entity has to subscribe to, in order to ensure seamless operation. Furthermore, the number of topics that an application has to subscribe to is not usually static and can vary substantially over time. Topics may be valid for very short durations and the number of related topics may change continually over time with entities publishing related streams. As an example, consider the scenario where a speaker's audio stream is in English, but someone else would perform the role of real-time interpreter and generate new audio streams in say Greek and somebody else would do the same for Latin; in some cases, there may be multiple streams in Greek. The participants within the collaborative session thus have access to multiple audio streams, depending on their authorizations, if they subscribe to the right topic corresponding to the audio stream of their choice. There thus needs to be a framework which facilitates clients and applications to cope with the high flux inherent in such situations by providing accurate information about the topics that are part of the collection at any given point of time.

As far as we know no system currently incorporates schemes to manage collection of topics. We believe that there would be systems that manage this problem in an ad hoc, application-specific manner. The scheme that we propose provides a generalized framework for the management of collections of topics and to facilitate discovery of topics that part of such collections in dynamic, high flux environments.

A topic collection is a tree-based structure which is used to manage the topics that comprise the collection. A collection may itself be composed of multiple collections. Depending on the topic owner's credentials and accompanying authorizations, a given topic can be registered or deregistered from any collection. A collection is thus amenable to reorganizations, where the constituent topics and collections may be added, removed or reorganized within the collection's structure. Additionally, a given topic or collection may be part of multiple collections concurrently. All nodes within the collection tree are collections with the topics constituting the leaf nodes. An example of the collection tree is depicted in Figure 1.

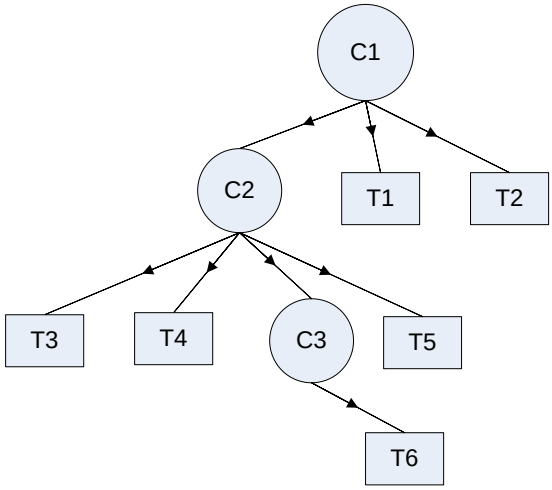

Figure 1 An example collection tree

8.2.1 Creating a Topic Collection

The creation of a collection is very similar to the creation of topic. Here the owner of a collection, issues a request to the TDN to request the creation of a topic collection. This request includes the following:

1. Information regarding the creator and possibly the institution that the creator belongs to
2. Information regarding the lifetime of this collection
3. Information regarding the collection –descriptors outlining the collection.
4. Authorization information related to the discovery of this collection – this controls the entities that authorized to discover the collection and those that are not.

5. Information regarding restrictions on the breadth and depth of the collection tree hosted by this node. The owner may also specify the limit on the total number of topics that can be part of this collection.

Some collections may be created with a fixed number of topics, where the structure of the collection cannot be changed or modified at run-time. In some cases, the mutations to the collection's structure may be allowed but would be based on the credentials associated with the entities that are initiating this action. When a collection is being created the owner can specify if the collection can be

1. Part of other collections, i.e. it could be a node of a larger collection tree
2. Can have child collection nodes in the sub-tree of which it is the root node.
3. Restrictions on who is authorized to perform the actions outlined above.
4. Restrictions on the Topics, i.e. credentials of the topic owners, that will be the leaf nodes of the collection or those that would be part of the sub-tree of which the collection node in question is the root.

After discovering the best available TDN, the entity issues a collection request to the TDN. This TDN then generates a new UUID. This UUID will now be unique identifier associated with the collection. Next, the TDN takes all the information that has been specified in the topic creation request, along with the modified template synopsis structure and proceeds to generate a hash (such as using SHA-256 algorithm) and sign it. This constitutes the collection advertisement. The collection advertisement is then encapsulated in a collection creation response and issued back to the entity through the broker network by publishing the creation response on Entity/EID topic. Upon receipt of the collection creation response, the entity verifies the signature and the integrity of the message, and then proceeds to glean the collection advertisement. The entity then proceeds to advertise this information to the available TDNs within the system.

8.2.2 Addition of Topics and Collections to an existing Collection

Additions of topics and collections to an existing collection can be performed at any time. When an entity tries to register a topic as part of collection, it needs to satisfy three requirements. First, the entity should be the owner of the topic and demonstrate possession of the appropriate credentials. Second, the entity must indeed be authorized to do so by the creator of the collection. Third, the addition of the topic to the collection does not violate restrictions pertaining to the breadth of the collection tree. In order to deregister a topic from a collection, the topic owner simply needs to demonstrate possession of the appropriate credentials.

When a collection node is added as a sub-tree to an existing collection structure, the addition should be consistent with the constraints specified by the immediate parent nodes right up till the root node of the collection tree. Besides the authorization constraints, the addition of the collection should not violate restriction pertaining to the depth of the collection tree.

8.2.3 Keeping track of the anatomy of the Collection

The owner of a collection, and others as authorized by the collection owner, can keep track of the anatomy of the collection by registering to receive change notifications corresponding to changes to the collection's structure. Associated with every collection is a UUID-based topic over which these change notifications are issued. Authorized entities can subscribe to this topic and keep track of the topics that they need to subscribe to. There can sometimes be several collections that are part of the collection's hierarchy. Each of these collections may have their own change-notification topics, which entities may or may not be authorized to receive. Authorizations that are valid for the root collection are valid for the sub-collections. Thus, changes to sub-trees within the collection tree will be routed to those authorized to receive notifications about the collection tree.

8.2.4 Discovery operations related to a Collection

Depending on their credentials different entities will be authorized to discover information pertaining to different parts of the collection tree. The discovery operation will support the discovery of the following.

1. The complete list of topics within the collection hierarchy
2. The list of collections that constitute the collection tree or sub-trees within the structure.
3. The list of topics that are part of a collection sub-tree within the structure.
4. The list of collections that a given topic is a part of.

It should be noted that the number of topics within a collection is in a constant state of flux, the discovery operations will return the list of topics/collections based on the information available at the TDN at any given time.

9 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,	Pentium 4, 1.6 GHz, 1 GB RAM, Win XP Professional	Java HotSpot(TM) Client VM (build 1.4.2_06-b03, mixed mode)

IN	Edition	

Table 1 Machine Configuration

In the first topology, the Topic Discovery Node (TDN) and the Clients were placed on separate machines (Refer Figure 2).

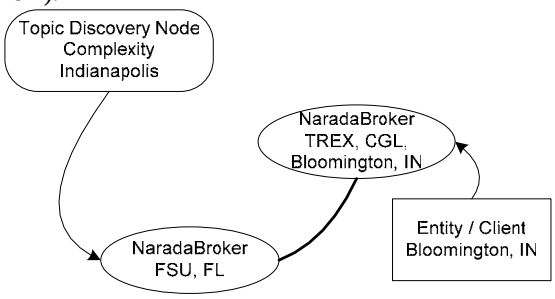

Figure 2 Wide Area Topology

Figure 3 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 Discovery response,
3. Entity sends an encrypted Topic Creation request,
4. TDN creates a Topic Advertisement,
5. TDN 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. This response is encrypted and sent to the Entity,
6. Entity validates the signed Topic Advertisement and broadcasts the signed advertisement over a pre-defined topic to all TDNs.

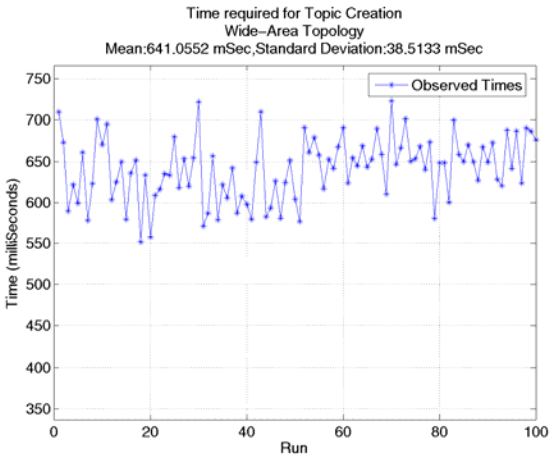

Figure 3 Topic Creation (Wide Area Topology)

Figure 4 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 5 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 (Pallickara and Fox, 2004) contains a detailed discussion on other types of topic synopsis matching such as tag-value pairs, integers and regular expressions among others.

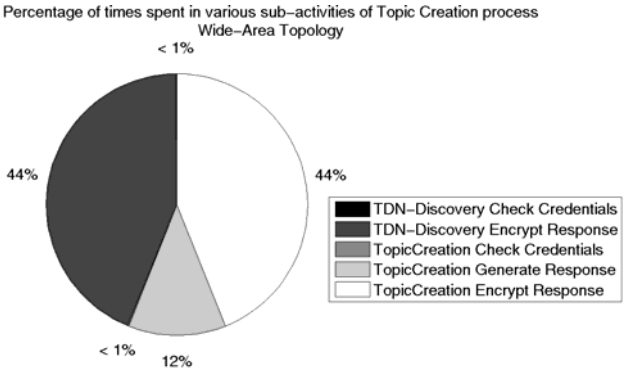

Figure 4 Time spent in Topic Creation sub-activities (Wide Area Topology)

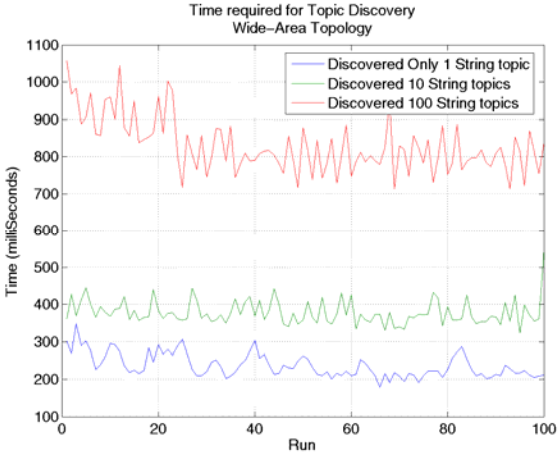

Figure 5 Topic Discovery (Wide Area Topology), String topics

	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					
Discover 1 Topic	236.86	32.01	348.88	178.95	3.20
Discover 10 Topics	378.33	33.42	548.11	32..68	3.34
Discover 100 Topics	829.69	73.61	1057.5	712.84	7.36

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

Note that in Figure 5 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 primarily because of the increase in time for encrypting

bigger response packet. Table 2 lists the metrics associated with wide area topology.

In our second topology (Figure 6), we ran the TDN and the clients on the same machine. Figure 7 - Figure 9 show the corresponding results.

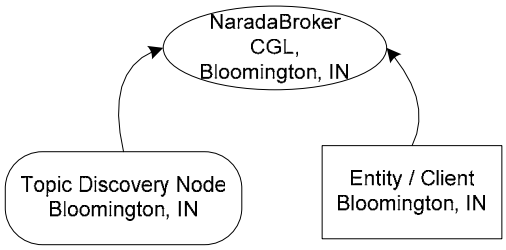

Figure 6 Local Machine Topology

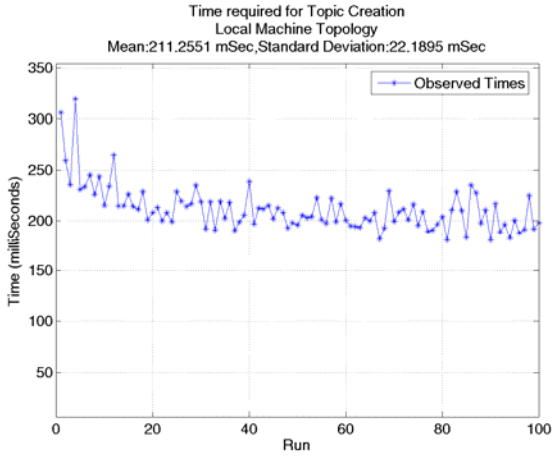

Figure 7 Topic Creation (Local Machine Topology)

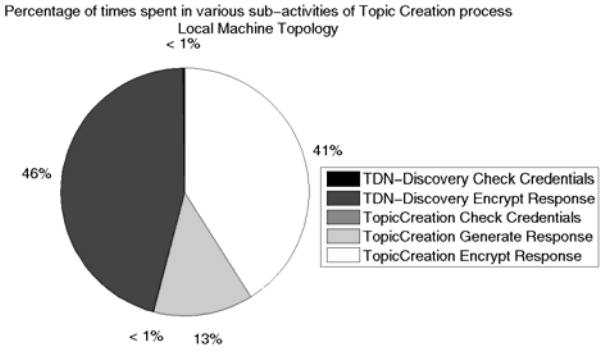

Figure 8 Time spent in Topic Creation sub-activities (Local Machine Topology)

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.

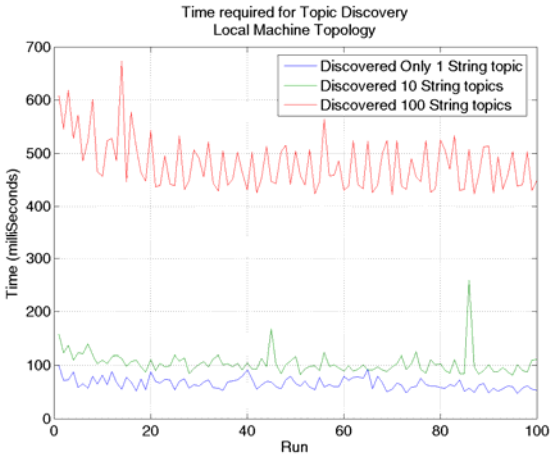

Figure 9 Topic Discovery (Local Machine Topology) , String Topics

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	319.5	180.35	2.22
Topic Discovery					
Discover 1 Topic	65.86	10.74	99.38	47.26	1.07
Discover 10 Topics	104.25	21.49	258.50	81.29	2.15
Discover 100 Topics	480.72	49.89	672.75	422.67	4.99

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

10 RELATED WORK

The WS-Topics (IBM et al., 2004) specification which is part of the WS-Notification suite of specifications incorporates the concept of topic spaces which facilitates the organization and categorization of topics. There a typically one or more hierarchically organized topic trees within a topic-space. Here we note that the specification does indeed support cases where there no hierarchical topics. Our scheme can leverage the WS-Topics specification by using it to organize topic spaces within the TDN. We were a little surprised with the lack of activity in the area of creation and discovery of topics in publish/subscribe systems. Systems such as Gryphon (Banavar, 1999), Siena (Carzaniga et al.,

2000), Elvin (Segall and Arnold, 1997) do not seem to have addressed this issue. As far as we know JMS based systems such as SonicMQ (S SonicMQ JMS Server http://www.sonicsoftware.com/) and openJMS (The OpenJMS Project http://openjms.exolab.org/) do not address this discovery issue either; though both these maintain something similar to topic spaces. It is our conjecture that applications developers include ad hoc solutions to ensure the propagation of topic information to relevant clients.

In the area of P2P systems a related issue is that of resource discovery. In one of the variants of P2P systems based on distributed hash tables (DHT) every peer has a unique identifier. This identifier determines the location of the peer within the overlay network. To discover a resource, such as a file, the peer first needs to know the hash value associated with this resource. DHT based systems store resources at the peer who identifier is *closest* to that of the hash associated with the aforementioned resource. Systems based on DHTs include Pastry (Rowstron and Druschel, 2001), JXTA (The JXTA Project and Peer-to-Peer Technology http://www.jxta.org) and Squid (Schmitd and Parashar, 2004). FLAPPS (Michel and Reiher, 2001; Michel and Reiher, 2001) provides a generalized infrastructure for peer network design. It should be noted that though FLAPPS is not DHT based it can indeed be used to support DHT-style systems. FLAPPS-based services define hierarchical, flexible and decomposable namespaces. Here, service representation of names can vary from delimited text strings to an XML parse tree. Furthermore, this namespace also supports prefix names while locating specific resources.

11 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 section 7.
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

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

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

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

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

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.

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

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/.

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.

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.

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.

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.

Web Services Reliable Messaging Protocol (WS-ReliableMessaging) ftp://www6.software.ibm.com/software/developer/library/ws-reliablemessaging200403.pdf, BEA et al., 2004

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

Web Services Topics (WS-Topics). IBM, Globus, Akamai et al. ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-Topics.pdf

G. 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.

Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In Proceedings of the 19th ACM Symposium on Principles of Distributed Computing, pages 219–227, Portland OR, USA, 2000.

Bill Segall and David Arnold. Elvin has left the building: A publish/subscribe noti.cation service with quenching. In Proceedings AUUG97, pages 243–255, Canberra, Australia, September 1997.

Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. Proceedings of Middleware 2001.

C. Schmidt and M. Parashar. Enabling Flexible Queries with Guarantees in P2P Systems, IEEE Network Computing, Special Issue on Information Dissemination on the Web, IEEE Computer Society Press, Vol. 8, No. 3, pp. 19- 26, May/June 2004.

B. Scott Michel, Peter L. Reiher: Peer-through-Peer Communication for Information Logistics. GI Jahrestagung (1) 2001: 248-256

B. Michel and P. Reiher. Peer-to-Peer Internetworking. In OPENSIG, September 2001