Semantic Distributed Messaging Middleware

Ananth Krishna, Simon Miles, Luc Moreau, Michael Luck

Abstract

Messaging middleware provides asynchronous communication between services in distributed environments. However, security, reliability and performance issues compel such middleware to be distributed, and distribution throws up its own problems such as identifying messaging channels which could then be subscribed to. In particular, interested parties need to identify channels defined in remote locations while not knowing details of how they are defined. A common vocabulary using semantic descriptions offers a solution to this problem. In this paper, we describe the design and implementation of federated messaging middleware using semantic description of channels.

1 Introduction

Messaging middleware is required to provide asynchronous communication between services and users in distributed environments. However, significant security, privacy, reliability and performance issues in messaging middleware are yet to be fully addressed. In relation to security and privacy, many organisations want to have access to their messages and subscription information under their control. In such cases, they would want to deploy messaging middleware that is only accessible from inside their organisation, but still receiving messages from external sources. Reliability is affected by having a single instance of messaging middleware, since it is a single point of failure. Finally, the performance of messaging middleware is adversely affected by increasing the number of subscribers to a message channel, because subscribers have different preferences for receiving messages on the same channel and their subscriptions must be managed concurrently. In this paper, we argue that distributing messaging middleware provides a clean solution for the above problems.

Although existing messaging middleware such as Websphere MQ (formerly IBM MQ Series) [3] and Sonic MQ [2] can work in a distributed context, our approach to distributing messaging middleware and addressing concerns involved with it is very different. In this paper, we think of distribution in terms of the messaging middleware being deployed at multiple locations, each deployment qualifying as an instance of the messaging middleware. We envisage multiple instances of the messaging middleware, each working independently as a peer. Each instance of the messaging middleware is secure within its own context and has messages and subscription information stored in a place under its control.

This addresses privacy and security issues, which have not been fully addressed in Websphere MQ's implementation, that provide access to messages and other related information through sharing repositories and/or memory. Similarly, Sonic MQ's implementation is focussed more on high scalability and message brokers deployed at multiple locations to balance load between them, which requires sharing subscription information with other message brokers in the distribution.

Messages in a messaging middleware are sent on message channels. Identifying such message channels is essential to allow interested parties to send messages to, or to subscribe to, these channels. This is difficult because interested parties do not have knowledge about message channels defined in remote locations, and would need to identify them to receive or send messages. When messaging middleware is distributed, the problem is further compounded by the fact that one instance of messaging middleware does not have knowledge of the message channels of other instances. A common vocabulary is therefore required to identify message channels, and we propose using semantic descriptions for this purpose in distributed environments.

In this paper, we first provide a brief background for our messaging middleware and motivation for our work, which can be found in Section 2. In the next section, we then proceed to describe the design and implementation of federated messaging middleware using semantic descriptions of message channels, which we call *topics*. Sections 4 and 5 specify the algorithms for creation of, subscription to and publishing on semantically-described distributed topics. Section 6 discusses the application of our work to the automatic management of user interests. In Section 7, we compare our approach to related work, identifying the impact of our approach, and we consider future research avenues in Section 8.

2 Background: $^{\text{my}}$Grid and its Application

2.1 $^{\text{my}}$Grid and the Williams-Beuren Syndrome

$^{\text{my}}$Grid is an e-Science project that aims to help biologists and bioinformaticians perform workflow-based *in silico* experiments and also help them in automating the management of such workflows through personalisation, notification of change and publication of experiments. The focus of $^{\text{my}}$Grid is on increasingly data-intensive bioinformatics, and the provision of a distributed environment that supports the *in silico* experimental process.

In $^{\text{my}}$Grid, we have considered several bioinformatics use cases, each involving a project to analyse the properties of a particular disease. One such project is to study the Williams-Beuren Syndrome [8]. Here, we have an organisation with large databases with access restricted to users and applications inside the organisation's private network. The data in these databases is changed frequently by applications and users working over this data. Services known as *filter services* are deployed in the organisation to analyse the current content of the data and determine whether there is a significant change related to a particular biological entity, e.g. a given gene. Now, there is a set of genes for which scientists analysing the Williams-Beuren Syndrome are interested in having the most up-to-date information at all times. These scientists include members of other organisations, and in particular universities who do not have direct access to this data.

2.2 $^{\text{my}}$Grid Notification Services and Topics

To ensure up-to-date information about the data requires that scientists are notified of the results of the analyses performed by the filter services. We have developed a messaging middleware called the $^{\text{my}}$Grid Notification Service (NS), which is a part of the $^{\text{my}}$Grid system, and essentially provides asynchronous communication between publishers and subscribers. In this sense, a *publisher* (or producer) is any application or component that publishes notifications, while a *subscriber* (or consumer) is any application or component that consumes notifications.

In the use case, the organisation holding the databases with restricted access and the analysis services has a notification service deployed in its network by which they can send out notifications about the results of the analysis. By contrast, the scientists on the Williams-Beuren Syndrome project use their own notification service, allowing them to receive notifications at the time most convenient to them, and store them securely within their organisation until then. As messages are published in a notification service different to the ones the scientists of Williams-Beuren Syndrome are subscribing to, this requires a mechanism to identify and share *topics* on both notification services. The mechanism required to achieve this is the semantic distributed messaging middleware that we have designed and shall discuss in detail in this paper.

A *topic* is a message channel in the messaging middleware and is used to identify the kind of messages being sent or received. Typically, topics are created by publishers with the intent that messages they publish under a given topic are all related in the same way. For instance, $^{\text{my}}$Grid has defined a topic *WorkFlowEvents*, for sending all events generated when running *in silico* experiments. When describing details of distributing the messaging middleware, topics residing on each instance of the messaging middleware are called *locally-defined topics.*

A subscriber interested in receiving messages related to a specific topic can register an interest in that topic with the NS hosting that topic. This registration of interest represents a *subscription*, which has an expiration time representing duration of interest. Publishers publish messages on a given topic; these messages are retained by the NS and later propagated to the appropriate subscribers who possess an unexpired subscription. The use of topics and subscriptions in this way constitutes the *publish-subscribe* model, and facilitates an asynchronous messaging model.

2.3 ^{my}Grid Information Model and User Profiles

The ^{my}Grid project also has a conceptual data model for capturing the entire e-Science process. This data model, referred to as the *information model* [6], defines users and information held about users of ^{my}Grid. This latter information is referred to as the *user profiles* of the scientists.

A user profile is the collection of computer-parsable information that the system holds about a particular user. This information could be used by deployed services to allow the user to access those services or enable the user to customise services for themselves. Customisation of a service for a particular user by using information provided by them is known as *personalisation*, and the ^{my}Grid project aims to allow scientists to personalise services for themselves. To enable such personalisation to a high degree, ^{my}Grid provides structured user profiles that have information, such as areas of interest, to particular users of ^{my}Grid.

Thus, user profiles of scientists contain interest expressed by scientists in certain areas like the Williams-Bueruen Syndrome. The interest expressed by a scientist is expressed semantically so that some reasoning of can be performed on this semantic description to automatically subscribe the scientist to receive results of analysis of the relevant genes. Subscribing to users' interests in this manner requires automatic user interest management, which we have designed and will discuss in detail in Section 6.

2.4 Ontologies

If the vocabulary for describing message channels is known in advance by all clients that may wish to subscribe to, or publish on, a topic, then they can do so without having to know about or contact the other clients. In an open distributed system, this is fundamentally important, as knowledge of other participants of that system at any given time cannot be defined at design or development time. We propose using an ontology to provide a common vocabulary of terms for semantically describing topics. From a notification service's perspective, each term is a single, arbitrary URI.

In judging whether two message channels are identified by the same semantic description (i.e. they share semantic terms), we do not perform any reasoning over those descriptions. The match is purely syntactic between semantic concepts and does not take account of equivalence of terms, subsumption of one concept in another, or any other ontology relation. This is a deliberate design decision to ensure that the middleware remains flexible and independent of the availabilty of any ontology language, and to ensure that performance is not unduly affected, as it would be if reasoning was performed every time a topic was created or a message was published. The advantage of using an ontology for identifying topics, as opposed to just a list of terms agreed by the community, is because it allows for more flexible behaviour in the context in which the messaging middleware is used. For example, users may browse for topic identifiers using the relations in a community-agreed ontology. As another example, we do use reasoning on the relations between ontology terms to automatically manage user interests. This is discussed in detail in Section 6. Where two computational processes with no prior knowledge of each other interact, there must be a mechanism to allow them to interpret each others' messages. This problem applies to messaging middleware but is far more general and outside the scope of this work. In $^{\text{my}}$Grid, we assume that clients of $^{\text{my}}$Grid architecture components, including the notification service, can interpret messages following the $^{\text{my}}$Grid information model expressed in XML, except where another form of interaction is explicitly defined.

Our work on federated notification services and automatic user interest management is particularly applicable to the problems described above. We have implemented federated notification services and have started applying the results to this application; testing is in progress at the time of writing this paper.

2.5 Motivation Summary

In summary, the following requirements motivate our work.

- Due to security requirements and the interaction of parties that have no prior knowledge of each other, such as in virtual organisations, messaging middleware is needed where each organisation can keep control over its own subscription information and messages received.
- Current technologies, such as IBM MQ Series and Sonic MQ, are based on sharing of either messages, subscription information or both and, therefore, are not suitable.
- We have particular use cases in bioinformatics, where database providers send notifications to scientists, with both parties controlling their own messaging context.
- Given that the communicating members of a community may not have direct contact or prior knowledge of each other, we need a mechanism to identify and connect *topics* on notification services.

3 Architecture and Design

As mentioned in Section 1, we envisage distributed messaging middleware as having multiple instances of notification services deployed in separate locations but still routing messages to each other. In our model, we have more than one instance of the NS, with each instance working independently of others in a federation. Distributing messaging middleware in our model effectively means distributing the channels on which messages are sent. In this section, we describe the architectural design of the federated Notification Services by distributing the message channels.

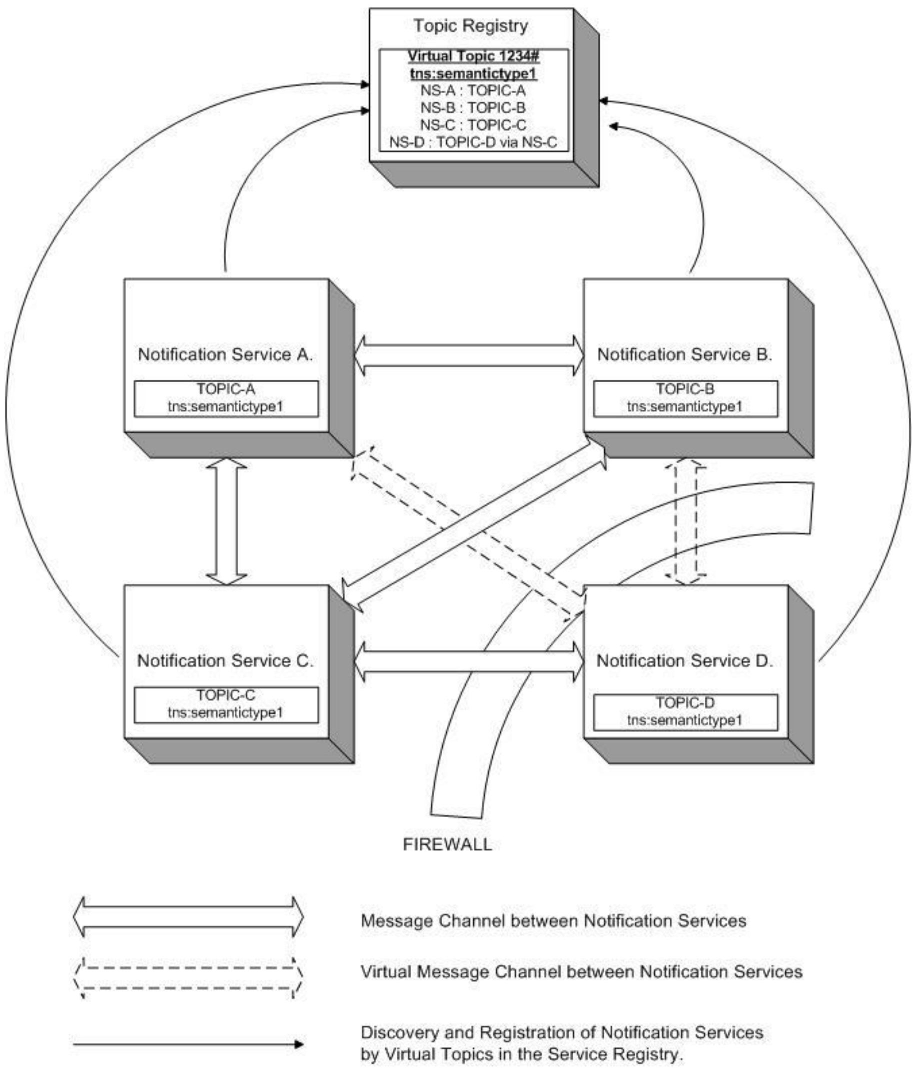


Figure 1: Federation of Notification Services

3.1 Federated Notification Services

In Figure 1, we show four distributed Notification Services (NS-A, NS-B, NS-C and NS-D). Users of each service, who have access to create topics, define topics (TOPIC-A, TOPIC-B, TOPIC-C and TOPIC-D) on notification services NS-A, NS-B, NS-C and NS-D respectively. At the time of creation, each locally-defined topic has semantic terms associated with it. It can be seen from figure 1 that all four semantic terms are syntactically the same (i.e.tns:semantictype1). As described in Section 2.4, there is no reasoning done on semantic terms associated with these topics and a syntactic match between the semantic terms suggests that all four topics described above share the same semantic terms. Hence, these topics are grouped under a single *virtual topic*, a virtual topic being a unique identifier created and managed by the Topic Registry. A Topic Registry is an extended service registry that is discussed in detail in Section 3.2. Virtual topics contain locally defined topics sharing the same semantic descriptions in the federation, thereby facilitating lookup of such topics, and services that host these topics.

Each service discovers other services publishing messages on the virtual topic by looking up the information in a Topic Registry. Here, a virtual topic advertisement (Virtual Topic 1234#) is identified by the semantic term, and contains the contact details of each notification service, along with the locally defined topic on which messages will be published. Using the information contained in the virtual topic, each NS sets up message channels to other notification services in the federation. Details of how these message channels are set up with other notification services is dicussed in detail in Section 4.2. Some services might choose not to allow message channels to be established to every other service in the federation, e.g. NS-D in the figure cannot communicate with NS-A or NS-B directly due to a firewall. In this case, messages to this NS are routed through an intermediary, NS-C in the figure. NS-A and NS-B have minimal information about NS-D, drawn from the Topic Registry, so that they know to send NS-D a copy of all messages via NS-C, thus creating a *virtual message channel*. Routing messages to notification services behind a firewall is discussed in more detail in Section 5.2. Below, we provide more detail about the discovery mechanism and the ontologies of semantic terms.

3.2 Topic Registry for Discovery of Virtual Topics

For one notification service to discover which other notification services are publishing messages on a virtual topic, there must be a central broker to which all the services advertise and discover these facts. In ^{my}Grid, we provide an extended service registry [7], known as the Topic Registry, which also holds centralised information on virtual topics for discovery by notification services. A notification service first publishes its contact details in the registry, so that others can know how and where to subscribe to it. Virtual topics are described in the registry by *virtual topic adverts*. Each virtual topic advert is identified by an associated piece of semantic metadata and contains a list of notification services

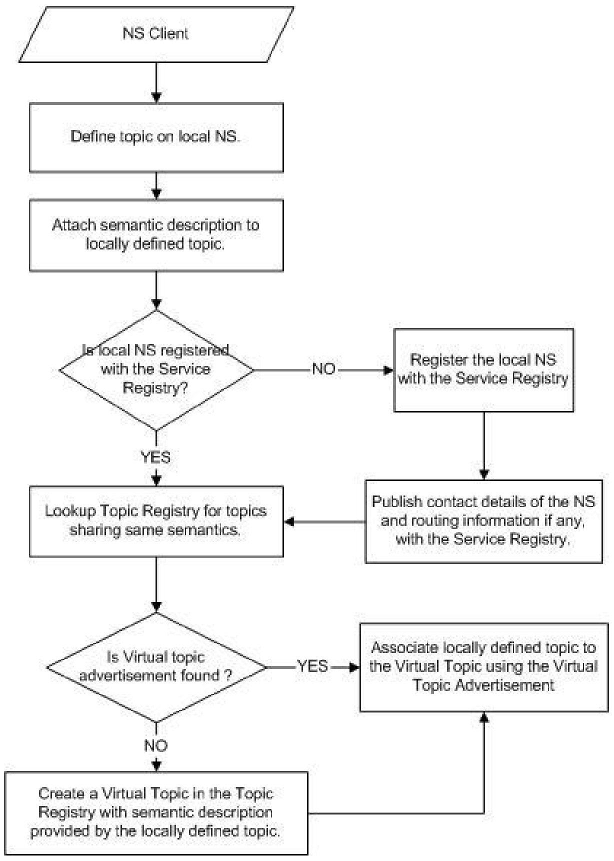

NS Client
Define topic on local NS.
Attach semantic description to locally defined topic.
Is local NS registered with the Service Registry?
NO
Register the local NS with the Service Registry
YES
Publish contact details of the NS and routing information if any, with the Service Registry.
Lookup Topic Registry for topics sharing same semantics.
Is Virtual topic advertisement found ?
YES
Associate locally defined topic to the Virtual Topic using the Virtual Topic Advertisement
NO
Create a Virtual Topic in the Topic Registry with semantic description provided by the locally defined topic.

Figure 2: Algorithm for Discovery and Creation of Virtual Topics

that are publishing on that virtual topic, with the associated contact details of the service, the locally-defined topic name that is used for the virtual topic on the service, and possibly routing information. The advert is first discovered, or created if it does not exist, by the notification service providing the identifying semantic metadata to the registry.

4 Creation of Semantic Distributed Topics

In this section, we describe the creation of virtual topics that are identifiable in the federation, the self-organisation of distributed topics, and subscription to virtual topics.

4.1 Discovery and Creation of Virtual Topics

The algorithm for discovery and creation of virtual topics is presented as a flow diagram in Figure 2. As shown in the figure, a client of a NS can locally define a topic with some semantic information associated with it. That semantic information will comprise one or more terms in an ontology, as described in the design and architecture section. The NS then uses the Topic Registry to discover any notification services with topics that share the same semantic description. Messages published on a locally-defined topic do not have to carry any semantic information, as there is no semantic reasoning performed anywhere in the messaging middleware. This is done to maintain flexibility and facilitate high performance.

If the lookup from the registry finds no existing virtual topic, the NS creates one in the registry. If a virtual topic is found and returned by the Topic Registry, the new locally-defined topic is associated with the virtual topic.

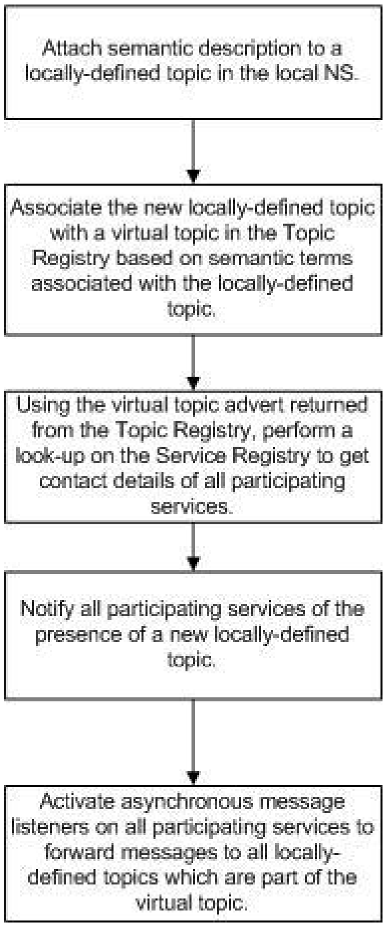
4.2 Self Organisation of Distributed Topics

When a NS discovers a virtual topic and associates a locally-defined topic with this virtual topic, it should be made known to other notification services participating in the federation. Services participating in the federation are notification services that host locally-defined topics, which are associated with the virtual topic. Notifying all pariticipating services in the federation about any changes associated with a virtual topic facilitates automatic and dynamic reorganisation of services in the federation. We have developed an algorithm for this purpose, which is presented as a flow diagram in Figure 3.

As shown in Figure 3, when some semantic term is associated with a new locally-defined topic in a NS, the NS discovers and associates the locally-defined topic with a virtual topic in the Topic Registry. The NS hosting the new locally-defined topic then performs a look-up on the Service Registry, which holds information about other participating notification services. The information returned is used by this NS to invoke a callback on all other participating services to notify them of the presence of a new locally-defined topic. Similarly, when a locally-defined topic is deleted from a NS, the NS performs a look-up on the Topic Registry and subsequently the Service Registry to get contact details of all participating services as described before and triggers these services to initiate suitable action to account for the deleted topic. In this way, notification services can either define a new locally-defined topic or delete an existing topic dynamically and other participating services will organise their locally-defined topics to work in this distributed context.

4.3 Subscription to Virtual Topics

Subscribers to a virtual topic, if they choose to, can be unaware of the fact that it is a *virtual topic* they are subscribed to. This is because the mechanism of subscribing to a virtual topic is similar to registering a subscription to a locally-


Attach semantic description to a locally-defined topic in the local NS.
Associate the new locally-defined topic with a virtual topic in the Topic Registry based on semantic terms associated with the locally-defined topic.
Using the virtual topic advert returned from the Topic Registry, perform a look-up on the Service Registry to get contact details of all participating services.
Notify all participating services of the presence of a new locally-defined topic.
Activate asynchronous message listeners on all participating services to forward messages to all locally-defined topics which are part of the virtual topic.
Figure 3: Algorithm for Self Organisation of Distributed Topics

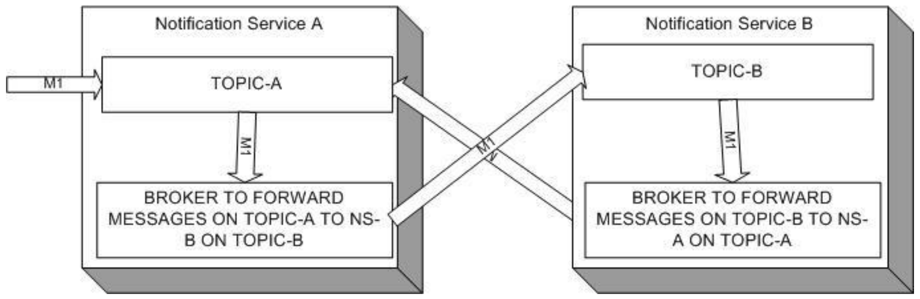

Notification Service A
M1
TOPIC-A
M1
BROKER TO FORWARD MESSAGES ON TOPIC-A TO NS-B ON TOPIC-B
M1
Notification Service B
TOPIC-B
M1
BROKER TO FORWARD MESSAGES ON TOPIC-B TO NS-A ON TOPIC-A

Figure 4: A Simple Example of the Cyclical Dependency Problem

defined topic as described in Section 2.2. In order to subscribe to a virtual topic, a client of a NS discovers a relevant topic by browsing the contents of the NS, and registers a subscription to receive messages from it. If the topic under consideration is associated with a virtual topic, the client receives messages published by trusted remote publishers who are part of the federation.

5 Publishing Messages on Distributed Topics

Messages sent by publishers on topics associated with a virtual topic should be broadcast around the federation to all other services participating in that virtual topic. However, we also need to ensure the efficiency of message distribution and to overcome security issues. We discuss two algorithms to do this below.

5.1 Cyclical Messaging Filter

Every notification service in a federation both publishes and subscribes to the locally-defined topics of all the other services. This means that a published message would, without prevention, return to the service that originally forwarded the message. This problem, the *cyclical dependency problem*, is illustrated in Figure 4, in which message M1 is published on TOPIC-A on NS-A. Because TOPIC-A and TOPIC-B are part of the same virtual topic, a *broker* exists in NS-A to forward all messages published on TOPIC-A to NS-B on TOPIC-B. When NS-B receives the message, however, it has its own broker which forwards the message back to NS-A on TOPIC-A. Therefore, message M1 would continue to cycle between the notification services in the federation forever.

The mechanism that we propose to solve the cyclical dependency problem is instantiated as a suitable filter in the brokers that forward messages to other services in the federation. The filter ignores messages forwarded by other brokers and so prevents a cycle occurring. So, for example, in Figure 4, message M1 would be stopped by the broker on NS-B because it recognises that the message has been forwarded by the broker on NS-A. Recognition that a message has been forwarded from another broker requires additional information to be attached to the message itself. This information takes the form of metadata in the header of the message and is attached by each broker before it forwards the message. The final concern is that the metadata is not accidentally or maliciously attached by publishers to the original message, which would prevent the mechanism from working correctly. We are still evaluating possible solutions to this and in particular are considering use of digital signatures to solve this problem.

5.2 Routing

Some notification services cannot be sent messages directly for many possible reasons, particularly if the security infrastructure of the owning organisation prevents incoming messages from untrusted sources. In such cases, messages must be routed through a trusted intermediary. This routing information can be registered, along with the locally-defined topic, under the virtual topic in the registry. When the brokers at each notification service in a federation wish to forward messages to others that cannot be directly contacted, they use the routing information to send it instead to the intermediary. As the intermediary is a notification service itself, it can forward messages to the hidden service. For the intermediary to recognise that a particular message should be routed to another service, this information must be attached to the message. Alternative approaches like WS-Routing exist to support this, and we discuss these in detail in Section 7 below.

6 User Interest Management

In some cases, users and the middleware that supports them must subscribe to topics without prior knowledge of how those topics have been defined. For instance, a scientist running a set of experiments needs to receive notifications about the progress and results of each. A new topic is created for each experiment by the middleware enacting that experiment. The scientist should be subscribed to this topic on the basis of their interest in the experiment. The semantic identification of topics, along with suitable *user profiles*, allows automatic user interest management to take place in $^{\text{my}}$Grid. In the following sections, we describe the use of semantics in defining user profiles, and the method by which subscription can be automated using this information.

6.1 Semantic User Profiles

To enable users to subscribe automatically to topics in the NS, there has to be a mechanism of matching user profiles to topics. In the mechanism described in the sections above, we discussed the use of semantic terms as suitable identifiers of topics. If semantic descriptions are also contained with the user profiles, a direct match can be made between them.

In ^{my}Grid, we provide structured user profiles with semantic information on user interests, organisational context, and so on. In addition, there exists an ontology of semantic terms (which is the same ontology as is used to identify topics), from which descriptions of user interests can be made. We can use the relations between terms in this ontology to develop a more sophisticated matching algorithm. For example, a user with a broad interest in all experiments related to a particular project can be automatically subscribed to topics related to each specific experiment in that project. As opposed to just syntactic matching in virtual topics, which we have described in Section 2.4, the matching algorithm in this case would reason on the semantics to automatically identify the topic concepts and subscribe to the user's profile. Details of the structure of the user profiles can be found in [6].

6.2 Automatic Subscription

Semantic information contained in user profiles can be used to discover topics with the same semantics in the Topic Registry. Using the federation mechanism, the user is subscribed to all the relevant virtual topics found. This can be done with no intervention from the user, so they do not have to be aware of the number or nature of the topics to which they are subscribed; it simply requires that the system ensures they are kept informed of all relevant notifications.

7 Related Work

In the upcoming GOLD [1] project, virtual organisations for chemical analysis are being addressed. As it is a high security domain, trust management between parties participating across virtual organisations is essential. The trust management proposed in this project involves a trusted third party service enforcing a given contract between the participants (a *contract service*), and a policy for every organisation that controls interactions with this contract service. With our federated notification services, an alternative solution could be envisaged. Notification services, private to their virtual organisations, can be made to work in a trusted federation, where the interactions are performed through a message channel identified by a semantic description, i.e. by a virtual topic. The semantic description defines the contract to which all participants must conform in order to be members of the virtual organisation. Given the security between the notification services and the registry that holds the virtual topic advertisement, the message channel can be set up securely only between these partipants, creating a trusted domain.

The WS-Notification [4] specification proposes standards for implementing both point-to-point and publish-subscribe messaging middleware. In publish-subscribe messaging, significant importance has been given to metadata attached to message channels, to enable them to be identified in a distributed context. There is also mention of federated message brokers to achieve improved performance in messaging middleware. While the specifications are still under development, we can see our work being compatible with WS-Notification.

WS-Routing [5] proposes specifying which intermediaries a message will be routed through, from source to destination, in the message header itself. Several alternative routes could be specified in a header to avoid performance problems or failures in particular intermediaries. We are investigating using this protocol to aid the routing of notifications through several services.

8 Conclusions

Due to the problems of controlling security, privacy, performance and reliability in the sending of messages between distributed parties, messaging middleware must be distributed. Although some existing messaging middleware can be made to work in a distributed context, we have proposed a mechanism different to the existing ones and have addressed concerns which we believe have not been fully addressed in other messaging middleware. As we have multiple deployments of our messaging middleware, each working independently of one another in a peer-to-peer manner, we have ensured privacy of messages in the federation. Message channels are distributed over different locations, which has resulted in simpler and faster concurrency management for subscribers having different preferences to receive messages, and increased performance for clients of the messaging middleware. Finally, distributing message channels has enabled us to eliminate a single point of failure in the system.

Distributing messaging middleware separates the clients defining the message channels from those that wish to use them. Thus, a mechanism by which all distributed parties can identify and discover message channels is required. For this purpose, we proposed using semantic descriptions to identify message channels.

We have developed an algorithm to achieve distribution of our messaging middleware using semantic information registered in a central location, and implemented this as part of our Web Service messaging middleware. As part of this work, we have addressed the problems of cycles of messages between notification services and the routing of messages to services behind firewalls, and developed techniques to avoid them. Further, we have extended our work on semantically-described message channels to the automatic management of user interests. Our work has been applied in the $^{\mathrm{my}}$Grid project to the analysis of Williams-Beuren Syndrome.

The next stage of our work will be to address security issues in the architecture of distributed messaging middleware. In particular, interactions between the messaging middleware and the registry holding information on distributed message channels must be made secure in order for trust to exist between the parties sending and receiving notifications. Furthermore, we are also evaluating potential routing algorithms for messages between services in different private domains. Finally, we are investigating techniques to support reasoning over ontologies to define message channels without compromising substantially on performance of the messaging middleware.

9 Acknowledgements

This research is funded in part by the ^{my}Grid e-Science pilot project grant (EPSRC GR/R677643), the PASOA project (EPSRC GR/S67623/01). Hannah Tipney, May Tassabehji, Andy Brass from St Marys Hospital, Manchester, UK performed the research on Williams-Beuren Syndrome and were supported by the Wellcome Trust.

References

[1] GOLD: Project Description. http://www.neresc.ac.uk/projects/GOLD/projectdescription.html, 2004.

[2] Sonic MQ. http://www.sonicsoftware.com/products/sonicmq/, 2004.

[3] WebSphere MQ. http://www.ibm.com/websphere, 2004.

[4] WS-Notification. http://www-106.ibm.com/developerworks/library/specification/ws-notification/, 2004.

[5] WS-Routing. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-routing.asp, 2004.

[6] M. Alpdemir, J. Ferris, R. M. Greenwood, P. Li, N. P. Sharman, and C. Wroe. The mygrid information model. In *UK e-Science Programme All Hands Meeting 2004 (AHM2004)*, September 2004. To appear.

[7] Simon Miles, Juri Papay, Vijay Dialani, Michael Luck, Keith Decker, Terry Payne, and Luc Moreau. Personalised grid service discovery. *IEE Proceedings Software: Special Issue on Performance Engineering*, 150(4):252–256, August 2003.

[8] R. Stevens, H.J. Tipney, C. Wroe, T. Oinn, M. Senger, P. Lord, C.A. Goble, A. Brass, and M. Tassabehji. Exploring Williams-Beuren Syndrome Using myGrid. In *Proceedings of 12th International Conference on Intelligent Systems in Molecular Biology*, Glasgow, UK, July 2004.