Reflective Channel Hierarchies

Edward Curry, Desmond Chambers and Gerard Lyons
{edward.curry, des.chambers, gerard.lyons}@nuigalway.ie
Department of Information Technology,
National University of Ireland, Galway, Ireland.

Abstract

Hierarchical channel structures are used to create granular sub-channels from a parent channel. Utilized in routing situations that are more or less static, they require that the channel namespace schema be both well defined and universally understood. The publish/subscribe messaging model currently requires a message publisher to place messages into a specific channel within the hierarchy. A relocation of responsibility for channel selection logic from the publishing client to the middleware service would open up static channel hierarchies to the application of reflective techniques. This shift in responsibilities enables the service more control over the definition, creation and maintenance of the channel hierarchy. The service is now able to apply reflective and adaptive techniques to dynamically adapt, grow and improve the hierarchy to better meet the needs of its changing environment and operating conditions. This paper describes work-in-progress on the definition of reflective channel hierarchies.

1. Introduction

The field of messaging middleware is a broad one, ranging from basic message delivery issues to questions regarding the semantic richness of client subscription interests. Our research focuses on the hierarchical channels or topics within publish/subscribe messaging systems.

This type of channel structure allows for channels to be defined in a hierarchical fashion, so that channels may be nested under other channels. Each sub-channel offers a more granular selection of the messages contained in its parent channel. Clients of hierarchical channels subscribe to the most appropriate level of channel in order to receive the most relevant messages. In large-scale systems, the grouping of messages into related types (i.e. into channels) helps to manage large volumes of different messages.

Channel hierarchies are analogues to event type hierarchies used in Hermes [1]. Similar relationships exist between a channel and sub-channels, as does between parent and child event types in Hermes. This relationship allows for super-type subscriptions, where subscriptions that operate on a parent channel/type will also match all subscriptions of descendant channels/types.

A channel hierarchy for an automotive trading service could be structured by categorizing messaging into buys or sells, then further sub-categories breaking down for vehicle types. A sample hierarchy illustrating this categorizing structure is presented in table 1.

Sell.vehicles
Sell.vehicles.cars
Sell.vehicles.trucks
Sell.vehicles.motorcycles
Buy.vehicles
Buy.vehicles.cars
Buy.vehicles.trucks
Buy.vehicles.motorcycles

Table 1: An Automotive Hierarchical Channel Structure.

Hierarchical channels are used in routing situations that are more or less static. They require that the channel namespace schema be both well defined and universally understood by the participating parties. Responsibility for choosing a channel in which to publish messages is left to the publishing client; this *channel selection logic* is integrated in the publisher's application code. The location of this logic results in a tight coupling between the channel hierarchy and publisher. Any alterations to this logic will need to be propagated throughout the codebase of the publisher's, making changes in the hierarchy difficult, restricting the ability of the hierarchy to evolve and grow.

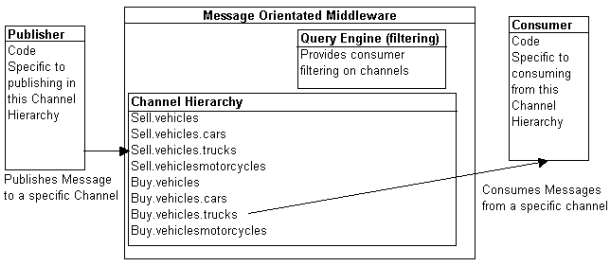

Figure 1: Current Messaging Architecture.

As illustrated in figure 1 the tight coupling between publishers and consumers to the channel hierarchy make it important to define accurate and correct channel hierarchies early in the development process. Alterations made to the hierarchy downstream will result in a high cost of change; once a system is deployed it becomes problematic to modify the hierarchy. Consumer clients are responsible for selecting the channel from which they wish to receive messages. Coupling between consumers and the hierarchy also hampers large-scale changes in the hierarchy's structure.

2. Research Overview

Research carried out at the IBM T.J. Watson center in New York has taken a unique approach with application message conditions. This work has identified that message receipt and processing by final recipients are often important criteria that represent a condition on further processing by the sender [2]. For example, a message representing the notification of a group meeting is sent to a set of participants, some of whom may be required to acknowledge the receipt and accept before the meeting can be scheduled and databases (for room reservation and other purposes) can be updated. At present no defined middleware support exists to aid in the development of applications that require the management of conditions on messages. The IBM solution uniquely shifts the responsibility for implementing the management of conditions on messages from the application to the middleware.

The static nature of channel hierarchies has limited research opportunities in this domain. Currently, it is necessary for a publisher to place messages into a specific channel within the hierarchy. We propose to move the responsibility of the *channel selection* to the middleware. This relocation of responsibility unlocks channel hierarchies to the application of new techniques allowing the service more control over the definition, creation and maintenance of the channel hierarchy. The enhanced service is now able to apply reflective and adaptive techniques to dynamically grow and improve the hierarchy to meet the needs of its changing environment and operating conditions. The resulting service is analogous to a post office: messages are placed into a black box, how the messages are sorted (published) is of little concern to the sender.

This work-in-progress research is conducted within the Virtual Logistics multi-Agent Brokerage (V-LAB) research project at the Department of Information Technology, National University of Ireland, Galway. The objective of V-LAB is to develop a system for the virtual brokerage, optimisation and management of road freight carriers. V-LAB combines diverse technologies to create a multi-agent software system that will facilitate companies competing for extra work convenient to their individual schedule. The remainder of this paper contains a brief overview of our current progress on defining a service that provides reflective channel hierarchies.

3. Chameleon Messaging Service

In order to realize our hypothesis we are developing a new reflective messaging service called Chameleon. This section presents an illustration of the Chameleon architecture in figure 2 and gives a brief description of the key areas of the architecture.

3.1. Channel Hierarchy Administrator

Responsible for the management of the channel hierarchy at run-time, the channel hierarchy administrator exposes a meta-model to express the structure of the channel hierarchy. This meta-model is a causally-connected representation of the channel hierarchy; any alternations to the model will be replicated in the underlying hierarchy structure enabling its run-time inspection and adoption.

3.2. Reflection Engine

The instigator of adaptive behavior in the service is the reflection engine. The engine inspects and adapts the behavior of the hierarchy by modifying the hierarchy model exposed by the channel hierarchy administrator. The engine exposes a meta-interface that allows for reflection policies to be plugable. These intelligence policies contain the rules and strategy used in the adoption of the service.

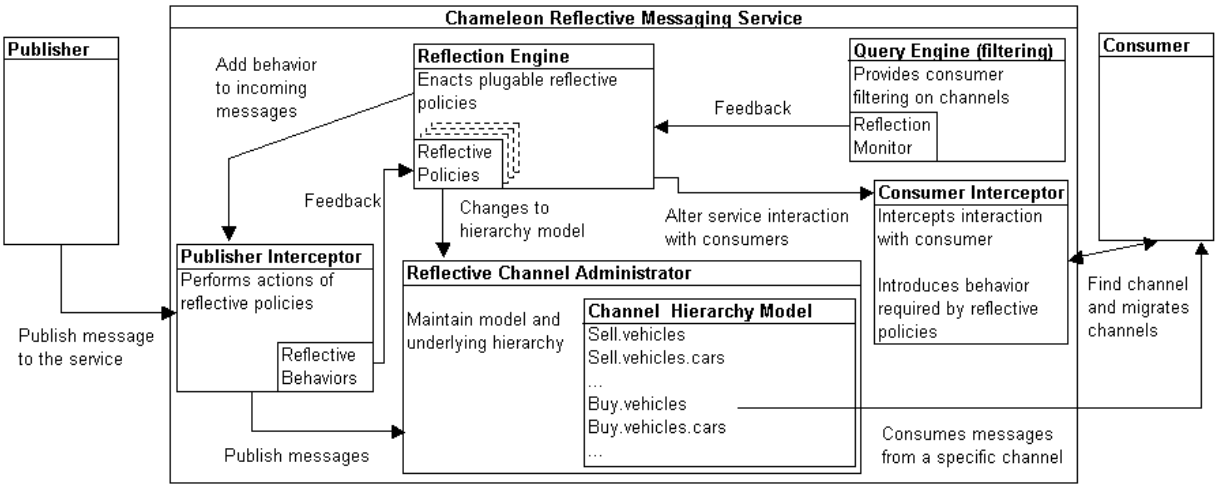

Figure 2: The Chameleon Reflective Architecture implementing Reflective Channel Hierarchies.

3.3. Reflection Policies

Reflective policies control the reflective behavior of the service. These policies can be designed with different motivations for the adaptation of the hierarchy, using different techniques to achieve them.

Potential policies could be designed to optimize the distribution mechanism for group messaging using IP multicasts or to provide support for federated messaging services using techniques from Hermes [1] or Herald [3]. Policies could also be designed to work with different levels of filtering (subject, content, content + patterns) or to support different formats of message payloads (XML, JPEG, PDF, etc).

3.4. Publisher and Consumer Interceptors

Interceptors are a mechanism used by the reflective engine to carry out the work of its policies. Reflective policies can use these interceptors to dynamically introduce a behavior into the service. The role of the publisher interceptor is to capture messages submitted to the service and perform any desired processing on them. Consumer interceptors are used to allow policies to alter the interaction of the service with its consumers.

4. Reflective Channel Hierarchies

The Chameleon messaging architecture will be used to implement reflective channel hierarchies. In order to achieve this objective the following policies and related interceptors have been defined.

4.1. Message Publisher Interceptor

The eradication of responsibility for channel selection from the publisher has placed the burden of channel selection on the middleware. In order to handle this a publisher interceptor is used to publish messages in the hierarchy. Once a message is intercepted based on the rules of the active reflective policies the interceptor selects the most relevant channel for the message to be published in.

4.2. Consumer Migration Interceptor

Consumer migration is an important consideration when new channels are being created. The consumers for whom the channel has been created must migrate to the new channel seamlessly. To aid in this transition a consumer migration interceptor has been introduced to the service. This interceptor will liaise with consumers to gracefully migrate them to their new channel.

Currently a consumer subscribes to a specific "hard coded" channel within a hierarchy, receiving messages from it. However with reflective hierarchies there is no guarantee that a channel will exist.

Consumers know the type of message they wish to receive. In order for true runtime consumer adoption of the hierarchy to take place, consumers are now responsible for informing the service of their requirements. Once the service receives these requirements from a consumer it needs to recommend the most relevant channel in the hierarchy to the

consumer. The service is responsible for directing the consumer to the channel most relevant to its needs.

4.3. Reflective Channel Hierarchy Policies

The objective of these policies is to adapt the channel hierarchy to better represent its contents, and to meet the requirements of its users. Currently two policies have been identified to create reflective channel hierarchies.

4.3.1. Consumer Filter Monitoring

This policy monitors the filtering carried out by consumers on channels within the hierarchy. Consumer filters are examined to expose the common fields of a message being filtered and reveal the most frequent values applied in these filters. If large numbers of similar filters for a specific field exist, the policy reacts by instructing the reflective channel administrator to create a new sub-channel to publish these messages in.

The message publisher interceptor is instructed to place any messages with this common filter into the newly created sub-channel. With the new sub-channel in place, the consumer migration interceptor transfers the relevant consumers to the new channel. The consumer interceptor is updated with the relevant information for the new channel; the interceptor has the role of directing consumers to the channel that best suits their needs. There is now no need for these consumers to apply this filter on the messages in this new channel. This policy will be developed to utilize a variety of filtering mechanisms with support for multiple message payload formats.

4.3.2. Message Traffic Pattern Analysis

Designed to complement the consumer filter monitoring policy as a cause of adoption in the service, the message traffic pattern analysis policy adapts the hierarchy through the pattern analysis of messages submitted to the service. New sub-channels are created based on this analysis. This policy will create new sub-channels, or locate appropriate existing channels within the hierarchy to publish these messages in.

As with the consumer filter monitoring policy, support for multiple filter types and payloads is envisioned. Depending on the message format, subject, attribute or content based filtering may to used to perform an analysis of messages. The choice of filtering would be dependant on the type of message as some payloads are more amenable to inspection, for example an XML payload would be easier to examine than a JPEG.

4.4. Benefits of Reflective Hierarchies

The benefits that reflective channel hierarchies have over conventional hierarchies will now be explored further.

4.4.1. Hierarchy Decoupling

Potentially, each of the major international stock exchanges could have their own stock notification services with a uniquely structured channel hierarchy. Unless the exchanges have reached a consensus on the structure and namespace schema of a single common hierarchy, a consuming client wanting to connect to these exchanges will need to be hard coded (coupled) to interact with each of the individual exchange hierarchies. With a reflective hierarchy, a consuming client is decoupled from the hierarchy. It only needs to specify its interests to the exchanges notification service and be dynamically directed to the relevant channel.

4.4.2. Hierarchy Growth

Depending on the deployment environment of the service, different criteria become important to the participating parties. We provide the motivating scenario of a system that deals with job requests for the courier/logistics sector. A job's location and completion time are an important factor for companies competing in this marketplace. However, depending on the scenario of the deployment, the scales used to measure the location and times differ.

In a continental deployment the country of origin and destination are of high priority to shippers, as not every company will service all the potential countries and regions. Completion times are measured in days and weeks. A potential channel hierarchy structure for a continental scale deployment is outlined in table 2. In this example structure, channels are sorted first by country, then regions of the country, then by the required time-scale of the delivery.

Ireland
Ireland.Dublin
Ireland.Dublin.within_2_Days
Ireland.Dublin.within_1_Week
…
Ireland.Galway
…
Germany
Germany.Munich

Table 2: Continental Deployment.

With a metropolitan scale deployment the source and destination become less important as it is more likely that any city courier will service the entire city, criteria such as package size and speed of delivery become more important, with completion times in hours. A potential channel hierarchy for a metropolitan scale deployment is shown in table 3. Channels are sorted by package weight and then by time scale of delivery.

Under_10_KG.within_1_Hour
Over_10_KG.within_2_Hours

Table 3: Metropolitan Deployment.

Other deployment scenarios with different domain specific criteria include fruit, furniture, money, dangerous materials, liquid, oil, gas, cars, etc. A static channel hierarchy requires predefined criteria for every potential deployment scenario in advance. An approach utilizing a reflective channel hierarchy lets the consumers choose the criteria, allowing a channel hierarchy to evolve (or grow) and constantly adapt into one customized specifically for the deployment environment. Starting from a single channel (seed) a customized channel hierarchy (tree) can grow based on the expressed requirements of its consumers.

5. Future Plans

Future plans focus on completing the development of the Chameleon architecture and the implementation of policies for the definition of reflective channel hierarchies. Policies to improve integration with agent-based systems will also be explored.

A benefit of Agent Based Systems has been for messages to be constructed from a common universally understood ontology. Through the use of this ontology each of the agents in a system has agreed on the semantic meaning of certain words and their usage. Reflective channel hierarchy will be dynamically created at run-time; consumers that understand the semantics of the channel names would have an advantage when interacting with the service. If the naming scheme used in the hierarchy was linked with an ontology, agents (consumers) familiar with the ontology could browser the hierarchy and reason as to a channel's contents and purpose.

6. Conclusions

This paper describes work-in-progress on a messaging service for the V-LAB project at the Department of Information Technology, National University of Ireland, Galway. A central goal of this research is to introduce reflective techniques into pub/sub systems, specifically channel hierarchies. This is an area where its application has not been investigated.

The introduction of reflective capabilities has been achieved by relocating channel selection responsibilities to the middleware in publish/subscribe systems. This shift has opened the static domain of hierarchies to the application of new techniques. As a result the producer is no longer tightly coupled to the channel hierarchy, and is now able to treat message services like a Post Office. They simply need to place their messages into a box and let the service worry about sorting (publishing) it correctly. The middleware is now able to adapt its channel hierarchies to suit its deployment scenario. Based on consumer usage patterns (filter monitoring and traffic analysis) channel hierarchies can adapt to meet the current needs of the user base. Hierarchies can also grow from a single channel into a customized channel hierarchy based on the expressed requirements of its consumers.

7. Acknowledgement

The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged.

8. References

[1] P. R. Pietzuch and J. M. Bacon, "Hermes: A Distributed Event-Based Middleware Architecture," 2002.

[2] S. Tai, T. Mikalsen, I. Rouvellou, and S. M. S. Jr, "Conditional Messaging: Extending Reliable Messaging with Application Conditions," presented at 22 nd International Conference on Distributed Computing Systems, Vienna, Austria, 2002.

[3] L. F. Cabrera, M. B. Jones, and M. Theimer, "Herald: Achieving a Global Event Notification Service," presented at 8th Workshop on Hot Topics in OS, 2001.