

Internet-Scale Web Services-based Event Notification Systems

Yi Huang

Department of Computer Science, Indiana University
150 S. Woodlawn Ave, Bloomington, IN, 47405, USA
yihuan@cs.indiana.edu

Abstract. Internet-scale Web services-based event notification services face several new challenges that have not been addressed in traditional event notification systems. In this paper, an Open Publish/Subscribe (OPS) model is proposed to address these new challenges. The openness of OPS model enables collaboration among different event notification systems from different organizations. The layered architecture of this model separate different concerns in event notification systems and can be used to address several challenges in Web services-based event notification systems. Two issues we will address in the thesis are supporting the expressive XPath filtering on XML messages without losing scalability and reconciling the competing Web services specifications on such systems.

1. Introduction

Web services-based event notification is an emerging technology that combines the asynchronous publish/subscribe communication feature of the event notification mechanism and the interoperability feature of Web services technologies. Web services-based event notification systems are getting more and more attention in the emerging Service-Oriented Architecture (SOA) and Event-Driven Architecture (EDA) for integrating applications both within an organization and across organization boundaries. WS-Eventing [1] and WS-Notification [2] are two major specifications for such systems. These specifications aim at defining standard Web service interfaces for subscription management and delivering XML-formatted event notification messages.

There are some major changes from traditional event notification systems to Web services-based event notification systems [3]. *First*, the format of notification messages is XML-based SOAP message format. XML has been widely adopted as a standard data exchange format. Comparing with name-value pair messages in traditional systems, XML messages provides both semantic and structural information on the data and it is supported by major software vendors. However, XML messages are larger and more complicated than name-value pair messages. Parsing these messages is much more computation-intensive. *Second*, interoperability among different administration domains on the Internet becomes a major concern since event sources and event consumers are less likely to be under the control of the same

administrator or implemented by the same vendor. *WS-Messenger* [4] is a research project on unique challenges in Web services-based publish/subscribe systems.

2. Problem statement

Notification brokers are key components for scalable, loosely coupled publish/subscribe systems. They provide an abstraction layer between event sources and event consumers. Broker network consists of two or more notification brokers to take advantage of locality, improve scalability and reduce network bandwidth usage. Internet-scale XML publish/subscribe services are getting more and more attentions since they are key components for large-scale distributed applications [5, 6]. Handling notifications on the Internet-scale faces many new challenges that are not addressed on the Intranet-scale, such as message routing, higher message rate and subscription populations, variable network conditions, unreliable resources and firewalls.

Existing researches on Internet-scale brokering system assume fully cooperative broker networks. Each broker in a broker network needs to trust every other broker and perform all required responsibilities, such as message routing, query processing, message transformation and message aggregation. Communications among them are proprietary protocols. However, to achieve a practical Internet-scale brokering system, the brokers usually need to be deployed in different administrative domains. Due to different economical and political concerns, cooperation among different organizations across the world is very hard and hoping everyone using brokers from the same vendor is impractical. Current Web services-based event notification specifications only define publish/subscribe interfaces for clients. They do not address interoperability among notification brokers from different organizations. An *open interoperable* brokering network model that can link different organizations is desired for Internet-scale brokering systems. How to achieve this is a research challenge.

Expressiveness and scalability have been treated as trade-offs in publish/subscribe systems [7]. Topic-based subscriptions are simple and easy to scale, but they are not expressive. If publishers cannot provide fine-grained topics, subscribers have to subscribe to all messages on that topic. Content-based subscriptions are more expressive. They allow subscribers to specify content characteristics. But they usually require more processing time in the message filtering process and their scalability has been a research problem [7]. Content-based XML message filtering aggravates this problem. XPath expression [8] is the desirable subscription language since it allows fine-grained selection by both structure and content. However, parsing XML messages and evaluating XPath query are much more expensive than processing traditional content-based name-value pair messages. How to achieve both the rich expressiveness in XPath subscriptions and scalability has been a challenge [5]. When brokers are distributed across Internet in different organizations, how to achieve them in an open and interoperable way without losing security is an even harder challenge.

Two major competing specifications on Web services-based event notification systems are WS-Eventing [1] and WS-Notification [2]. They are incompatible with each other and both have implementations from different projects. In many application integration scenarios, it is not feasible to change the specification

implementations used in existing systems, especially if the change involves two or more organizations [9]. In order to keep existing Web services-based event notification systems unchanged and achieve interoperability among various implementations of competing specifications, notification brokers should be able to transparently reconcile the differences among event notification specifications. No previous work can achieve this. Also, due to different specifications and different versions, there are potentially a large amount of different operation messages or notification messages in different formats, how to reconcile them in a scalable way is another challenge.

This thesis will address the above mentioned three problems by proposing an open model and by separating different concerns to different layers in this model.

3. Related work

XML message filtering and transformation in publish/subscribe systems. Many content-based notification broker projects are available. Traditional content-based publish/subscribe systems with broker networks include SIENA [7], Gryphon [10], HERMES [11], JEDI [12]. They usually filter messages with a simple name-value pair structure. Research efforts on efficient XPath-based filtering for more complicated XML message contents include XFilter [13], YFilter [14], Xaos [15], XSQ [16]. These XPath filters are designed for processing XPath evaluations in a single machine, mostly for XML database, and are not designed to support notification broker networks for scalability.

PsEPR project [6] deploys an Internet-scale XML message brokering system, but it uses topic-based message filtering. Currently, there are three broker networks that support XPath-based filtering, ONYX [5], XNet [17] and Naradabrokering [18]. Only NaradaBrokering has Web services implementation and only ONYX provides message format transformation based on XQuery. Unlike *WS-Messenger*, every broker in these projects needs to inspect XML message contents to make routing decisions which is ineffective and lack of security. Neither mediation on message formats from competing Web services specifications nor interoperability among event notification brokers from different vendors is addressed.

Open interoperability model for messaging systems Several open interoperability approaches are available. Some specifications, such as JMS, define *API-level* interoperability. Some communication protocols, such as TCP protocol and IP protocol, define *binary-level* interoperability. Recently, *message-level* interoperability is gaining momentum. Web services achieve interoperability by defining common XML formats. WS-Eventing [1] and WS-Notification [2] define interfaces between clients and brokers. However, they do not define an interoperable model for message filtering and delivery among brokers. Advanced Message Queuing Protocol (AMQP) specification [19] is a recent draft specification that aims to create a “totally open, platform agnostic, interoperable” messaging infrastructure for collaboration among queuing services in different organizations. The message-level open interoperability approach of this specification is similar to the OPS model in this paper, but its domain is not publish/subscribe messaging system.

3. OPS model

In this thesis, I propose Open Publish/Subscribe (OPS) model as an open and generic model for Internet-scale event notification broker networks for XML messages. This model is similar to the open TCP/IP model that Internet is built upon but it is at a higher level. This model separates different concerns in publish/subscribe systems to different layers. Five layers are included in this model: event transportation layer, event distribution layer, query processing layer, event transformation layer and event application layer. Each layer can achieve its responsibility independently without much effect on the other layers. Table 1 summarizes the responsibilities of each layer in the OPS model. In the process of delivering a notification message from a publisher broker to a consumer broker, intermediary brokers in a broker network only need the bottom two layers. This means much simpler architecture and much less processing load for intermediary brokers.

Table 1. Responsibilities of each layer in OPS model

Layers	Responsibilities
Event Application (layer 5)	Generates/consumes notification messages, including Complex Event Processing
Event Transformation (layer 4)	Transforms notification messages for common processing or for individual end consumers
Query Processing (layer 3)	Filter/check message according to subscriptions
Event Distribution (layer 2)	Determines list of next routing brokers and/or list of consumers for messages
Event Transport (layer 1)	Delivers messages between brokers

4.1 Features of OPS model

OPS model has two major features. **First, it is an open model for interoperability.** OPS model defines open and simple message formats at the event distribution layer for communications among notification brokers. Interoperability is based on messages on the wire. We believe that in order to have a wide deployment of collaborative brokers from different organizations on the Internet, a *least responsibility* approach is needed. Collaborative brokers only need to perform its core task which is message routing. Other responsibilities should be handled by higher level applications as services, possibly provided by different vendors. In this way, different organizations can achieve easier collaboration on creating an Internet-scale event delivery network without too much burden on resources. An analogy of this approach is the success of the Internet. We only need to trust the routers in other organizations to forward our messages based on IP address headers, and DNS servers can provide us with correct naming information. We do not require them to perform security checks on our IP packets or remove duplicates.

Second, OPS model uses a layered model to achieve separation of concerns for scalability and security. Each layer in OPS model can be scaled out independent of other layers. Different layers may have different scalability concerns and need

different scaling approaches. For example, scaling out event distribution layer is different from scaling out query processing layer as described in next section.

The security benefit of the separation of concerns is that it enables routing of encrypted messages in content-based filtering. This is not feasible in previous content-based message filtering systems. In OPS model, only publisher brokers and consumer brokers need to check message content for content-based message filtering, intermediary brokers just need to check FRS (filtering result summary) created by the query processing layer in the message headers to determine the message routing. Messages can be encrypted by publisher brokers for traversal on the Internet through different administrative domains. In this way, secure communication channels can be established between publishing organizations and consuming organizations.

4.2 Applications of OPS model

As sample applications of OPS model, I will address the other two research problems describe in section 2 using the open interoperability model. They can be solved at different layers in OPS model.

Expressiveness vs. Scalability problem for XML messages This problem will be addressed by separating the scalability of event distribution with the scalability of event filtering. At the event distribution layer, scalability concern is how to route messages efficiently from publisher brokers to consumer brokers. Scaling out approaches include adding more delivery agents for a broker and adding more brokers in different locations Routing decisions can be made by checking only the Filtering Result Summary (FRS) generated by the query processing layer and attached as the header of an XML notification message. Neither XML parsing nor XML content inspection is needed. The scalability concern in the query processing layer is how to efficiently matching subscriptions with XML messages. Since the query processing happens at the publishing brokers, scalability can be achieved by using more query processing servers in a cluster locally for a high-rate publisher. In our implementation, XPath evaluation will be based on YFilter [14] and FRS format will be based on a simple XPath canonicalization protocol we designed.

Scalable mediation among competing Web services specifications To solve this problem, the transformation layer in OPS model is used to transform message formats automatically based on the specifications used in subscription messages. Specifically, a Normalization-Processing-Customization (NPC) approach is used in our mediation solution [9]. The contribution of this solution is that it uses normalized information set to simplify mediation among Web services and makes mediation scalable. Mediation among N different formats only needs $2N$ mediation rules, instead of N^2 .

5. Conclusions

In conclusion, researches in the *WS-Messenger* project concentrates on unique challenges in Web services-based event notification systems. This thesis proposes an Open Publish/Subscribe model for Internet-scale XML message brokering. It studies how to achieve scalable XPath-based message filtering and how to reconcile

competing Web services specifications under this open interoperability model. We will deploy our system in wide-area networks, e.g. PlanetLab [20], and conduct various performance tests in real-world network environments.

References

1. Box, D., et al. Web Services Eventing. Available from: <http://ftpna2.bea.com/pub/downloads/WS-Eventing.pdf>.
2. OASIS. WS-Notification (v1.2). Available from: <http://docs.oasis-open.org/wsn/2004/06/>.
3. Huang, Y. and D. Gannon, A Comparative Study of Web Services-based Event Notification Specifications. ICPP Workshop on Web Services-based Grid Applications 2006.
4. Huang, Y., et al., WS-Messenger: A Web Services based Messaging System for Service-Oriented Grid Computing. International Symposium on Cluster Computing and the Grid (CCGrid06).
5. Diao, Y., S. Rizvi, and M.J. Franklin, Towards an Internet-Scale XML Dissemination Service. Proceedings of VLDB 2004, 2004.
6. Brett, P., et al., A Shared Global Event Propagation System to Enable Next Generation Distributed Services. Workshop on Real, Large Distributed Systems (WORLDS), 2004.
7. Carzaniga, A., D.S. Rosenblum, and A.L. Wolf, Achieving scalability and expressiveness in an internet-scale event notification service. Proceeding of Nineteenth ACM Symposium on Principles of Distributed Computing (PODC 2000), 2000.
8. Clark, J. and S. DeRose. XML Path Language (XPath) Version 1.0. Available from: <http://www.w3.org/TR/xpath>.
9. Huang, Y. and D. Gannon, A Flexible and Efficient Approach to Reconcile Different Web Services-based Event Notification Specifications. Proceedings of International Conference on Web Services (ICWS), 2006.
10. Banavar, G., et al., An efficient multicast protocol for content-based publish-subscribe systems. Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS'99), 1999.
11. Pietzuch, P. and J. Bacon. Hermes: A Distributed Event-Based Middleware Architecture. in Workshop on Distributed Event-Based Systems (DEBS). 2002.
12. Cugola, G., E.D. Nitto, and A. Fugetta, The jedi event-based infrastructure and its application to the development of the opss wfms. IEEE Transactions on Software Engineering, 2001.
13. Altinel, M. and M.J. Franklin., Efficient Filtering of XML Documents for Selective Dissemination of Information. Proc. of VLDB 2000, 2000.
14. Diao, Y. and M.J. Franklin, High-Performance XML Filtering: An Overview of YFilter. IEEE Data Engineering Bulletin, 2003(March, 2003).
15. Barton, C.M., et al., Streaming XPath Processing with Forward and Backward Axes. Proc. of ICDE, 2003.
16. Feng Peng, S.S.C., XPath Queries on Streaming Data In Proc. of SIGMOD, 2003.
17. Chand, R. and P.A. Felber, A Scalable Protocol for Content-Based Routing in Overlay Networks. In Proc. of Symposium on Network Computing and Applications, 2003.
18. Fox, G. and S. Pallickara, NaradaBrokering: An Event-based Infrastructure for Building Scalable Durable Peer-to-Peer Grids, in Grid Computing: Making the Global Infrastructure a Reality. 2003.
19. Advanced Message Queuing Protocol Specification. Available from: <http://www.iona.com/opensource/amqp/>.
20. Peterson, L., et al., Experiences Building PlanetLab. Proceedings of USENIX OSDI'06, 2006.