Abstract

This paper presents an investigation and analysis of the performance of services in publish-subscribe middleware working in the networks. The publish/subscribe paradigm of Message Oriented Middleware provides a loosely coupled communication model between distributed applications. Traditional publish/subscribe middleware uses keywords to match advertisements and subscriptions and does not support deep semantic matching. In this paper the message latency is measured as a relation with message interval for both methods of messaging: Message Oriented Middleware (MOM), and Java Message Service (JMS). It is found that the message latency have an inverse non–linear relationship with message interval.

Keywords: publish-subscribe, middleware, semantic matching

I Introduction

Many large-scale distributed systems today need to connect thousands of systems that are widely distributed and change frequently throughout their lifetime. This challenge motivates the demand for middleware that can provide loosely coupled communication models for distributed systems, allowing each component to evolve independently. While traditional point-to-point and synchronous communication models are popular in rigid and static applications, Message-Oriented Middleware (MOM), provides a versatile middleware system to loosely integrate distributed systems [3]. As the name suggests, a Message-Oriented Middleware system enables distributed applications to communicate by routing their messages through the middleware system. In this system, the client application sends messages to the MOM and the MOM is responsible for
delivering the message to remote receivers. There are two broad categories of message-oriented middleware: message queuing and publish/subscribe. The message queuing paradigm provides a point-to-point messaging model, in which typically messages are addressed to their recipients. By contrast, the publish/subscribe paradigm provides a many-to-many communication model so that messages can be efficiently disseminated across a large scale distributed system. In a publish/subscribe system, participant applications have two different roles – publishers that publish messages to the MOM as information sources, and subscribers that subscribe the information of their interests and receive messages from MOM as information consumers. The publish/subscribe paradigm enables communicating parties to be decoupled in communication time and channel and it has gained much attention in last several years.

A fundamental problem in a publish/subscribe system is how to match the interests of subscribers with the available messages from publishers. The publish/subscribe systems can be classified in two categories: topic-based or content-based [10]. In a topic-based system, each message is classified as belonging to one of a fixed set of topics, also referred as groups, channels, or subjects. A publisher labels each message it produces with a particular topic. There are many researches discussed and studied the issue of web semantic and publisher scriber. Amirhossein (et al. 2011), the authors proposed a method to improve the delivery guarantees of the basic unreliable service occurred by a best-effort publish/subscribe system. This method does not require any modification to the system's protocols or broker software, and instead simply uses the system's publish/subscribe API. The method is based on a technique, similar to reliable multicast that enables subscribers to cooperatively recover lost messages. We experimentally demonstrate the effectiveness and performance of our recovery scheme in the presence of frequent message losses, and show that it enables subscribers to recover more than 70% of lost messages with minimum negative effects on the overall network performance. Ioanna (et. al.2002), processed the current data infrastructure of the electric power grid was based on the existence of vertically integrated utilities that controlled the entire power operation within a geographical region. Further, it was designed decades ago by engineers who knew little about the state of the art (back then) in computer networking or distributed systems. Recent deregulation allows a wider set of participants to get involved in the power industry by having independent parties generating power or by permitting various traders negotiating power prices Delivering real-time and historical data to the interested parties needs to be done reliably and fast without jeopardizing the stability of the grid itself. A new data infrastructure is needed for distributing information to legitimate entities in a timely, secure and accurate manner. They were designing and implementing a new status service for the power grid; at the middleware layer to best support heterogeneity in devices and Applications. Eiko (et al. 2003), this paper presented the design, implementation, and evaluation of Pronto, a middleware system for mobile applications with messaging as a basis. It provides a solution for mobile application specific problems such as resource constraints, network characteristics, and data
Measuring the latency of semantic message optimization. Pronto consists of three main functions: 1- Mobile JMS Client, a lightweight client of Message Oriented Middleware (MOM) based on Java Message Service (JMS), 2-Gateway for reliable and efficient transmission between mobile devices and a server with pluggable components, and 3- Serverless JMS based on IP multicast. The publish-subscribe paradigm is ideal for mobile applications, as mobile devices are commonly used for data collection under conditions of frequent disconnection and changing numbers of recipients. This paradigm provides greater flexibility due to the decoupling of publisher and subscriber. Adding a gateway as a message hub to transmit information in real-time or with store-and-forward messaging provides powerful optimization and data transformation. Caching is an essential function of the gateway, and Smart Caching is designed for generic caching in an N-tier architecture. Server less JMS aims at a decentralized messaging model, which supports an ad-hoc network, as well as creating a high-speed messaging BUS. Pronto is an intelligent Mobile Gateway, providing a useful MOM intermediary between a server and mobile devices over a wireless network. Han Li (et al. 2002), the publish/subscribe paradigm of Message Oriented Middleware provides a loosely coupled communication model between distributed applications. Traditional publish/subscribe middleware uses keywords to match advertisements and subscriptions and does not support deep semantic matching. To this end, they designed and implemented a Semantic Message Oriented Middleware system to provide such capabilities for semantic description and matching. We adopted the DARPA Agent Markup Language and Ontology Inference Layer, a formal knowledge representation language for expressing sophisticated classifications and enabling automated inference, as the topic description language in our middleware system. A simple description logic inference system was implemented to handle the matching process between the subscriptions of subscribers and the advertisements of publishers. Moreover our middleware system also has a security architecture to support secure communication and user privilege control. Angelo C, 2006, During the last decade the publish/subscribe communication paradigm gained a central role in the design and development of a large class of applications ranging from stock exchange systems to news tickers, from air traffic control to defense systems. This success is mainly due to the capacity of publish/subscribe to completely decouple communication participants, thus allowing the development of applications that are more tolerant to communications asynchrony. This paper introduces the publish/subscribe communication paradigm, stressing those characteristics that have a stronger impact on the quality of service provided to participants. The article also introduced the reader to two widely recognized industrial standards for publish/subscribe systems: the Java Message Service (JMS) and the Data Distribution Service (DDS).

The system topology of the Semantic Message Oriented Middle is centralized client-server model. Each client acts as a publisher or subscriber, and a centralized messaging server is responsible for matching and brokering the messages between publishers and subscribers.
II SYSTEM STRUCTURE

The central middleware server contains two layers: the semantic broker layer and the JMS provider (J2EE server) layer. The semantic broker runs on the same machine with the JMS provider, and acts as an interface between the JMS provider and the application clients. The semantic broker handles the publish/subscribe requests of the client applications, organizes and maintains the topics of the JMS provider, matches and maps the clients’ topic descriptions with the JMS topics, and generates and receives the JMS messages for the JMS provider. The JMS provider takes care of the delivery of the messages that are grouped by JMS topics. The details of the JMS provider, such as the JMS topic names and wrapping and receiving JMS messages, are transparent to the publisher and subscriber client applications. A set of APIs are provided for publisher/subscriber client applications which encapsulate the procedures to make a connection with the semantic broker server, submit advertisement or subscription, and pass and receive messages to the server and from the server. Figure 1 shows the architecture of the whole system[4,5,6].

III. RESULTS AND DISCUSSION

Some experiments have been done to evaluate our Semantic Message Oriented Middleware system. The simulation is constructed on a single computer that did not connect with other machines by network. It would be ideal to simulate in a large scale real distributed environment, but it is difficult to deploy the system on a significant number of distributed nodes just for simulation, and in an open network environment, and even if we were able to do so, there are other
unpredictable factors such as network traffic congestion caused by other network dataflow which can influence the system performance. In our simulation environment, both the messaging server and client applications were running on the computer with different terminal windows, and all the network dataflow was just through this computer itself. We took average latency of a message from transmission to reception as one measurement of the message delivery capability of the messaging system. Figure 2 shows the simulation data of our Semantic Message Oriented Middleware system and the standard JMS provider (J2EE server) messaging system. The simulations show that there was a message delivery latency overhead of the Semantic Message Oriented Middleware system compared with JMS messaging system. The overhead comes from three main reasons, the first is the extra computation workload for running the semantic broker server beside the JMS provider; second is the workload for maintaining SSL encrypted communication with the clients; The last is the system penalty of multiple tasks running simultaneously within the operation system. The simulation data show that our Semantic Message Oriented Middleware system works stably and provides reasonable performance. From last figure there is an inverse non-linear relationship between the latency of the message and the message interval which can be represented as:

\[ \text{Latency} \propto \frac{1}{\text{MI}} \]  

(1)

Where: MI: denotes the message interval (ms).

Figure 2 Message transmitting latency of Semantic MOM and JMS

IV CONCLUSIONS

This paper has described our work on the Semantic Message Oriented Middleware system, a subject-based messaging system based on the Java Message
Service technology, that provides the capability of semantic topic description and matching, and also provides a specification of message encryption security. In the semantic broker layer of our system, a simple description logic inference system has been implemented to provide the semantic topic matching service between the publishers and subscribers. By adopting the DARPA Agent Markup Language and Ontology Inference Layer (DAML+OIL) as the topic description language, the event publishers and subscribers can flexibly describe and match their requests based on a pre-agreed ontology that defines the event information spaces. The semantic broker layer also makes the details of the JMS systems transparent to the client applications. With the Secure Sockets Layer architecture and user information management, the Semantic Message Oriented Middleware system provides secure data communication between the client applications and messaging system.

REFERENCES


[5] L. Han a and J. Guofei, 2000, Semantic Message Oriented MiddlewareFor Publish/Subscribe Networks, This work was partially supported by: ARDA Grant F30602-03-C-0248, DARPA projects F30602-00-2-0585 and F30602-98-2-0585.

[6] M. Amirhossein, C. Antonio, P. Fernando, T. Giovanni End-to-End Reliability for Best-Effort Content-Based Publish/Subscribe Networks, This work was supported in part by the Swiss National Science Foundation under grant number 200020-120188/1

Measuring the latency of semantic message

0107; National Institute of Justice, Department of Justice award number 2000-DT-CX-K001.


Received: March, 2012