Chapter 10
Timely Autonomic Adaptation of Publish/Subscribe Middleware in Dynamic Environments

Joe Hoffert
Vanderbilt University, USA

Aniruddha Gokhale
Vanderbilt University, USA

Douglas C. Schmidt
Vanderbilt University, USA

ABSTRACT

Quality-of-service enabled publish/subscribe (pub/sub) middleware provides powerful support for scalable data dissemination. It is difficult to maintain key quality of service properties (such as reliability and latency) in dynamic environments for distributed real-time and embedded systems (such as disaster relief operations or power grids). Managing quality of service manually is often not feasible in dynamic environments due to slow response times, the complexity of managing multiple interrelated quality of service settings, and the scale of the systems being managed. For certain domains, distributed real-time and embedded systems must be able to reflect on the conditions of their environment and adapt accordingly in a bounded amount of time. This paper describes an architecture of quality of service-enabled middleware and corresponding algorithms to support specified quality of service in dynamic environments.

DOI: 10.4018/978-1-4666-2056-8.ch010

INTRODUCTION

Emerging Trends and Challenges

The use of publish/subscribe (pub/sub) technologies for distributed real-time and embedded (DRE) systems has grown in recent years due to the advantages of performance, cost, and scale as compared to single computers (Huang, 2006; Tarkoma, 2006). In particular, pub/sub middleware has been leveraged to ease the complexities of data dissemination for DRE systems. Examples of pub/sub middleware include the CORBA Notification Service (Ramani, 2001), the Java Message Service (JMS) (Monson-Haefel, 2000), Web Services Brokered Notification (Niblett, 2005), and the Data Distribution Service (DDS) (Pardo-Castellote, 2003). These technologies support the propagation of data and events throughout a system using an anonymous publication and subscription model that decouples event suppliers and consumers.

Pub/sub middleware is used across a wide variety of application domains, ranging from shipboard computing environments to cloud computing to stock trading. Moreover, the middleware provides policies that affect the end-to-end quality of service (QoS) of applications running in DRE systems. Policies that are common across various middleware technologies include grouped data transfer (*i.e.*, transmitting a group of data atomically), durability (*i.e.*, saving data for subsequent subscribers), and persistence (*i.e.*, saving data for current subscribers).

Even though tunable policies provide fine-grained control of system QoS, several challenges emerge when developing pub/sub systems deployed in dynamic environments. Middleware mechanisms used to ensure certain QoS properties for one environment configuration may be ineffective for different configurations. For example, a simple unicast protocol, such as the User Datagram Protocol (UDP), may address the specified latency QoS when a publisher sends to a small number of subscribers. UDP could incur too much latency, however, when used for a large number of subscribers due to its point-to-point property, leaving the publisher to manage the sending of data to each subscriber.

Challenges also arise when considering multiple QoS policies that interact with each other. For example, a system might need low latency QoS and high reliability QoS, which can affect latency due to data loss discovery and recovery. Certain transport protocols, such as UDP, provide low overhead but no end-to-end reliability. Other protocols, such as the Transmission Control Protocol (TCP), provide reliability but unbounded latencies due to acknowledgment-based retransmissions. Still other protocols, such as lateral error correction protocols (Balakrishnan, 2005), manage the potentially conflicting QoS properties of reliability and low latency, but only provide benefits over other protocols in specific environment configurations.

It is hard to determine when to switch from one transport protocol to another or modify parameters of a particular transport protocol so that desired QoS is maintained. Moreover, manual intervention is often not responsive enough for the timeliness requirements of the system. DRE systems operate within strict timing requirements that must be met for the systems to function appropriately. The problem of timely response is exacerbated as the scale of the system grows, *e.g.*, as the number of publishers or subscribers increases.

This article describes how our work (1) monitors environment changes that affect QoS, (2) determines in a timely manner which appropriate transport protocol changes are needed in response to environment changes, (3) integrates the use of multiple supervised machine learning techniques to increase accuracy, and (4) autonomically adapts the network protocols used to support the desired QoS. We have prototyped this approach in the ADAptive Middleware And Network Transports (ADAMANT) platform (as briefly outlined previously (Hoffert & Schmidt, 2009) that supports environment monitoring and provides timely