Maintaining QoS for Publish/Subscribe Middleware in Dynamic Environments *

Joe Hoffert and Douglas C. Schmidt
Institute for Software Integrated Systems, Dept of EECS
Vanderbilt University, Nashville, TN 37203
{jhoffert, schmidt}@dre.vanderbilt.edu

1. INTRODUCTION

Emerging trends and challenges. The number and type of distributed systems that utilize publish/subscribe (pub/sub) technologies are growing due to the advantages of performance, cost, and scale compared with single computers [1, 2]. Examples of pub/sub middleware include Web Services Brokered Notification (`www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn`), the Java Message Service (JMS) (`java.sun.com/products/jms`), the CORBA Event Service (`www.omg.org/technology/documents/formal/event_service.htm`), and the Data Distribution Service (DDS) (`www.omg.org/spec/DDS`). These technologies support data propagation throughout a system using an anonymous subscription model that decouples event suppliers and consumers.

Pub/sub middleware is used in many application domains, ranging from shipboard computing environments to fractionated spacecraft constellations. The middleware supports policies that affect the end-to-end QoS of the system. Common policies across different middleware include *persistence* (*i.e.*, saving data for current subscribers), *durability* (*i.e.*, saving data for subsequent subscribers), and *grouped data transfer* (*i.e.*, transmitting a group of data as an atomic unit).

While tunable policies provide fine-grained control of system QoS, several challenges emerge when developing pub/-sub systems deployed in dynamic environments. Mechanisms used by the middleware to ensure certain QoS properties for a given environment configuration may not be applicable for a different environment configuration. For example, a simple unicast protocol (such as UDP) may provide adequate latency QoS when a publisher sends to a small number of subscribers. UDP can incur too much latency, however, when used for a large number of subscribers due to publishers sending UDP messages to each individual subscriber.

Challenges also arise when managing multiple QoS policies that interact with each other. For example, a system might specify low latency QoS and reliability QoS, which can affect latency due to data loss discovery and recovery. Certain transport protocols (again such as UDP) provide low overhead but no end-to-end reliability. Other protocols (such as TCP) provide reliability, but incur unbounded latencies due to acknowledgment-based retransmissions. Still other protocols balance reliability and low latency, but provide benefit over other protocols only for specific environment configurations. Determining when to modify parameters of a particular transport protocol or switch from one transport protocol to another can be complex. Moreover, human intervention is often not responsive enough to meet system timeliness requirements.

Solution approach → ADAptive Middleware And Network Transports (ADAMANT). The remainder of this document describes the research we are conducting to address the challenges of maintaining QoS in dynamic environments by integrating and enhancing (1) QoS-enabled pub/sub middleware, (2) adaptive transport protocols, (3) environment monitoring, (4) supervised machine learning, and (5) autonomic adaptation of transport protocols to manage specified QoS within dynamic environments.

2. MOTIVATING EXAMPLE - SEARCH AND RESCUE (SAR) OPERATIONS FOR DISASTER RECOVERY

To motivate the need for autonomic adaptation of QoS-enabled pub/sub middleware, we briefly describe the research challenges associated with search and rescue (SAR) operations. These operations help locate and extract survivors in a large metropolitan area after a regional catastrophe, such as a hurricane, earthquake, or tornado. SAR operations use unmanned aerial vehicles (UAVs), existing operational monitoring infrastructure (*e.g.*, building or traffic light mounted cameras intended for security or traffic monitoring), and (temporary) datacenters to receive, process, and transmit event stream data from sensors and monitors to emergency vehicles that can be dispatched to areas where survivors are identified.

In a SAR scenario infrared scans along with GPS coordinates are provided by UAVs and video feeds are provided by existing infrastructure cameras. These infrared scans and video feeds are then sent to a datacenter, where they are processed by fusion applications to detect survivors. Once a survivor is detected the application can develop a three dimensional view and highly accurate position information

*This work is supported in part by the AFRL/IF Pollux project and NSF TRUST.

DEBS 2009, July 6-9, 2009, Nashville, TN, USA

so that rescue operations can commence.

Several challenges that arise with SAR operations in dynamic environments are summarized below.

2.1 Challenge 1: Timely Adaptation to Dynamic Environments

Due to the dynamic environment inherent in the aftermath of a disaster SAR operations must adjust in a timely manner as the environment changes. If SAR operations cannot adjust quickly enough they will fail to perform adequately given a shift in resources. If resources are lost or withdrawn—or demand for information increases—SAR operations must be configured to accommodate these changes with appropriate responsiveness to maintain a minimum level of service. If resources increase or demand decreases, SAR operations should take advantage of these as quickly as possible to provide higher fidelity or more expansive coverage. Manual modification is often too slow and error-prone to maintain QoS.

2.2 Challenge 2: Managing Interacting QoS Requirements

SAR operations must manage multiple QoS requirements that interact with each other, *e.g.*, data reliability so that enough data is received to be useful and low latency for soft realtime data so that infrared scans from UAVs or video from cameras mounted atop traffic lights do not arrive after they are needed. The streamed data must be received soon enough so that successive dependent data can be used as well. For example, MPEG I frame data must be received in a timely manner so that successive dependent B and P frame data can be used before the next I frame makes them obsolete. Otherwise, not only is the data unnecessary, but sending and processing the data has consumed limited resources.

2.3 Challenge 3: Scaling to Large Numbers of Receivers

For a regional or national disaster, a multitude of organizations would register interest not only in the individual video and infrared scans for various applications, but also in the fused data for the SAR operations. For example, fire detection applications and power grid assessment applications can use infrared scans to detect fires and working HVAC systems respectively. Likewise, security monitoring and structural damage applications can use video stream data to detect looting and unsafe buildings respectively. Moreover, federal, state, and local authorities would want to register interest in the fused SAR data to monitor the status of current SAR operations.

2.4 Challenge 4: Specifying Standardized and Robust QoS

SAR applications should be developed with the focus on application logic rather than on complex or custom formats for specifying QoS. Time spent learning a customized or complex format for QoS is time taken from developing the SAR application itself. Moreover, learning a custom format will not be applicable for other applications that use a different QoS format. Application developers also need support for a wide range of QoS to handle dynamic environments.

3. SOLUTION APPROACH

Our solution approach combines and enhances the following technologies to resolve the challenges presented in Section 2.

- Standard QoS-enabled pub/sub middleware addresses the scalability of Challenge 3 in Section 2.3 by decoupling data senders from data receivers. Applications interested in published data can receive it any time without knowledge of the data sender. Moreover, standard QoS-enabled middleware addresses the QoS standardization of Challenge 4 in Section 2.4.
- Supervised machine learning helps address Challenge 1 in Section 2.1 and Challenge 2 in Section 2.2 by selecting an appropriate transport protocol and protocol parameters in a timely manner given a specified QoS and a particular environment configuration. The machine learning component includes features for several different environment configurations and supervised training techniques, such as decision trees, multilayer perceptrons, and support vector machines, to learn the correct protocol and parameters. The machine learning interpolates and extrapolates its learning based on the current environment configuration, which may not have been included in the supervised training.
- Adaptive network transports helps address Challenge 1 in Section 2.1 and Challenge 2 in Section 2.2 by providing the infrastructure flexibility to maintain interrelated QoS even within dynamic environments. For some environment configurations one particular transport protocol provides the required QoS. For other environment configurations another transport protocol provides the specified QoS. Adaptive network transports not only support fine tuning a protocol's parameters, but also switch from one protocol to another to provide functionality needed in dynamic environments.
- Environment monitoring helps address Challenge 1 in Section 2.1 by providing environment configuration information. Relevant environment configuration values are monitored as needed such as the number of subscribers, the percentage of network packet loss, and the sending rate of the data. These monitored values are input to the machine learning component to determine an appropriate network transport and accompanying parameters.
- Autonomic adaptation helps address Challenge 1 in Section 2.1 and Challenge 2 in Section 2.2 by (1) querying relevant values from the environment monitoring, (2) activating the machine learning component which will determine an appropriate transport protocol and parameters, (3) retrieving the recommended protocol settings, and (4) transitioning the adaptive network transports to use the recommended settings.

SDS

4. REFERENCES

[1] Yi Huang and Dennis Gannon. A comparative study of web services-based event notification specifications. *Proceedings of the International Conference on Parallel Processing Workshops*, 0:7–14, 2006.

[2] Sasu Tarkoma and Kimmo Raatikainen. State of the Art Review of Distributed Event Systems. Technical Report C0-04, University of Helsinki, 2006.