Using Machine Learning to Maintain QoS for Large-scale Publish/Subscribe Systems in Dynamic Environments

Joe Hoffert, Daniel Mack, and Douglas Schmidt
Department of Electrical Engineering & Computer Science
Vanderbilt University, Nashville, TN 37235
jhoffert@dre.vanderbilt.edu, dmack@isis.vanderbilt.edu, schmidt@dre.vanderbilt.edu

***Abstract*—Quality-of-service (QoS)-enabled publish/subscribe (pub/sub) middleware provides powerful support for large-scale data dissemination. It is hard, however, to maintain specified QoS properties (such as reliability and latency) in dynamic environments (such as disaster relief operations or power grids). For example, managing QoS manually is not feasible in large-scale dynamic systems due to (1) slow human response times, (2) the complexity of managing multiple interrelated QoS settings, and (3) the scale of the systems being managed.**

Machine learning techniques provide a promising approach to maintaining QoS properties of QoS-enabled pub/sub middleware in large-scale dynamic environments. These techniques include decision trees, neural networks, and linear logistic regression classifiers that can be trained on existing data to interpolate and extrapolate for new data. By training the machine learning techniques with system performance metrics in a wide variety of configurations, changes to middleware mechanisms (*e.g.*, associations of publishers and subscribers to transport protocols) can be driven by machine learning to maintain specified QoS.

This paper describes research we are conducting on highly configurable QoS-enabled middleware, adaptive transport protocols, and machine learning techniques. The goal of our approach is to maintain specified QoS as the environment and configuration of large-scale systems changes dynamically. Our preliminary research shows that decision trees and neural networks provide strong initial classification of the best protocols to use. The decisions trees' results answer questions about which measurements and variables are most important when considering the optimal network configurations from a reliability and latency standpoint.

I. Introduction

Emerging trends and challenges. The number and type of distributed systems that utilize publish/subscribe (pub/sub) technologies have grown due to the advantages of performance, cost, and scale as compared to single computers [7], [9]. 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 a wide spectrum of large-scale application domains, ranging from shipboard computing environments to grid computing. 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 atomically).

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

While tunable policies provide fine-grained control of system QoS, several challenges emerge when developing pub/sub systems deployed in large-scale dynamic environments. Middleware mechanisms used 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 could incur too much latency, however, when used for a large number of subscribers due to the publisher needing to send 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, such as UDP, provide low overhead but no end-to-end reliability. Other protocols, such as TCP, provide reliability but unbounded latencies due to acknowledgment-based retransmissions. Still other protocols, such as lateral error correction (LEC) protocols [1], balance reliability and low latency, but only provide benefits over other protocols for specific environment configurations. Determining when to modify parameters of a particular transport protocol or switch from one transport protocol to another is hard. Moreover, human intervention might not be responsive enough for the timeliness requirements of the system. The problem of timely response is exacerbated by increasing the scale of the system, *e.g.*, increasing the number of publishers or subscribers.

Solution approach → ADAptive Middleware And Network Transports (ADAMANT). The remainder of this document describes our ADAMANT approach to addressing the challenges of maintaining QoS for large-scale systems in dynamic environments. ADAMANT integrates and enhances the following technologies: (1) QoS-enabled pub/sub middleware, (2) adaptive transport protocols, and (3) machine learning to manage specified QoS within dynamic environments. By utilizing the machine learning techniques trained on data collected from multiple configurations, ADAMANT can adjust the middleware mechanisms to provide the transport protocol and parameter settings that maintains the specified QoS.

II. Motivating Example

To motivate the need for integrating machine learning techniques with QoS-enabled pub/sub middleware, this section describes 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 can utilize 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.

Figure 1 shows an example SAR scenario where 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 for rescue operations.

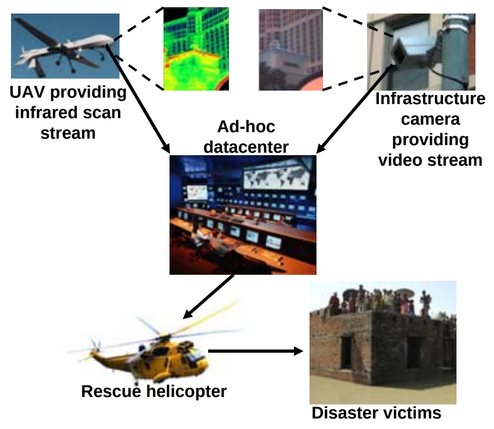

UAV providing infrared scan stream
Ad-hoc datacenter
Infrastructure camera providing video stream
Rescue helicopter
Disaster victims

Fig. 1. **Search and Rescue Motivating Example**

The following are key challenges that arise with SAR operations in dynamic environments.

A. 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.

B. 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.

C. 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.

D. 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.

III. Overview of Related Work

This section analyzes related research efforts in light of the challenges presented in Section II.

Support for adaptive middleware. The Mobility Support Service (MSS) [3] provides a software layer on top of pub/sub middleware to enable endhost mobility. The purpose of MSS is to support the movement of clients between access points of a system using pub/sub middleware. In this sense, MSS adapts the pub/sub middleware used in a mobile environment. MSS is solely focused on supporting mobility of pub/sub, however, and therefore does not address Challenge 2 in Section II-B. Moreover, MSS fails to address Challenge 4 in Section II-D since it does not present a standardized and robust interface for QoS.

Gridkit [5] is a middleware framework that supports reconfigurability of applications dependent upon the condition of the environment and the functionality of registered components. Gridkit focuses on grid applications which are highly heterogeneous in nature. For example, these applications will run on many types of computing devices and will operate across different types of networks.

Gridkit focuses on reconfiguration for installing an application and does not address Challenge 1 in Section II-A. Within Gridkit no consideration is given to making timely adaptations based on environment changing for a single application installation. Moreover, Gridkit fails to address Challenge 4 in Section II-D as it provides no standardized QoS specification.

David and Ledoux have developed SAFRAN [4] to enable applications to become context-aware themselves so that they can adapt to their contexts. SAFRAN provides reactive adaptation policy infrastructure for components using an aspect-oriented approach. However, SAFRAN is a component framework that only provides development support of maintaining specified QoS. The adaptive policies and component implementation are the responsibility of the application developer. Moreover, SAFRAN does not specifically address Challenge 3 in Section II-C since it does not focus on scalability. Moreover, SAFRAN does not address Challenge 4 in Section II-D since it provides no standard QoS specification.

Machine learning in support of autonomic adaptation. Vienne and Sourrouille [10] present the Dynamic Control of Behavior based on Learning (DCBL) middleware that incorporates reinforcement machine learning in support of autonomic control for QoS management. However, DCBL provides a customized QoS specification that does not address Challenge 4 in Section II-D and DCBL focuses on single computers rather than addressing scalable distributed systems, as outlined with Challenge 3 in Section II-C.

Autonomic adaption of networks. The Autonomic Real-time Multicast Distribution System (ARMDS) [2] is a framework that focuses on decreasing excessive variance in service quality for multicast data across the Internet. The framework supports the autonomic adaptation of the network nodes forming the multicast graph so that the consistency of service delivery is enhanced. ARMDS does not address Challenge 2 in Section II-B, however, nor does it address Challenge 4 in Section II-D.

IV. Overview of ADAMANT

The *ADAptive Middleware And Network Transports* (ADAMANT) is enhanced pub/sub middleware that adjusts underlying transport protocols and associated parameter settings to maintain specified QoS. ADAMANT addresses the challenges presented in Section II to resolve gaps in related work described in Section III by combining the following techniques.

• Standard QoS-enabled pub/sub middleware addresses the scalability of Challenge 3 in Section II-C 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 II-D.

• Adaptive network transport protocols work to address Challenge 1 in Section II-A and Challenge 2 in Section II-B by providing the infrastructure flexibility to maintain interrelated QoS even within dynamic environments. Adaptive network transport protocols not only support fine grained control of a protocol's parameters, but also switching from one protocol to another to provide the adaptation needed within dynamic environments.

• Machine learning helps address Challenge 1 in Section II-A and Challenge 2 in Section II-B by selecting in a timely manner an appropriate transport protocol and protocol parameters given specified QoS and a particular environment configuration. The machine learning will interpolate and extrapolate its learning based on the current environment configuration, which might not have been included in the training.

Figure 2 shows how ADAMANT captures key metrics into data files and classifies middleware behavior using the WEKA 3 data mining software (www.cs.waikato.ac.nz/ml/weka). Specifically, ADAMANT captures (1) data update latency times (*i.e.*, the time from when the data writer writes the data to the time the data reader receives the data), (2) the number of updates received compared to the number of updates sent, and (3) network bandwidth usage statistics (*e.g.*, total bytes on the network and min/max/avg bandwidth usage). Figure 2 also shows how we integrated and enhanced the OpenDDS implementation (www.opendds.org) of the OMG Data Distribution Service (DDS) standard (www.omg.org/spec/DDS) with the Adaptive Network Transports (ANT) framework, which supports various transport protocol properties, such as NAK-based and ACK-based reliability and flow control. ADAMANT leverages ANT to appropriately modify transport protocols and parameters settings as needed to maintain QoS.

OpenDDS is standards-based anonymous QoS-enabled pub/sub middleware for exchanging data in event-based distributed systems. It provides a global data store in which publishers and subscribers write and read data, respectively, so applications can communicate by publishing information they have and subscribing to information they need in a timely manner. OpenDDS provides support for various transport protocols including TCP, UDP, IP multicast, and reliable multicast. OpenDDS also provides a pluggable transport framework that allows integration of custom transport protocols within OpenDDS. We chose the OpenDDS implementation due to (1) its source code being freely available, which facilitates modification and experimentation and (2) its pluggable transport framework that allows us to integrate OpenDDS with the ANT framework.

We chose the ANT framework due to its infrastructure for composing transport protocols. ANT builds upon the properties provided by the scalable reliable multicast-based Ricochet transport protocol [1]. ANT also provides a modular framework whereby protocol modules can be tuned, enhanced, and replaced to maintain specified QoS.

We chose the Weka data mining software due to its user-friendly interface, ease of use, robust analysis tools, and support for a wide range of machine learning techniques. These techniques include decision trees, multilayer perceptrons, and support vector machines. We input collected metrics and configuration information for the environment and transport protocol used into Weka. We are classifying and analyzing the data using the various machine learning techniques to determine which techniques provide the best guidance in selecting a transport protocol for a given environment.

We are currently running experiments and collecting metrics using the Emulab network testbed (www.emulab.net). Emulab provides computing platforms and network resources that can

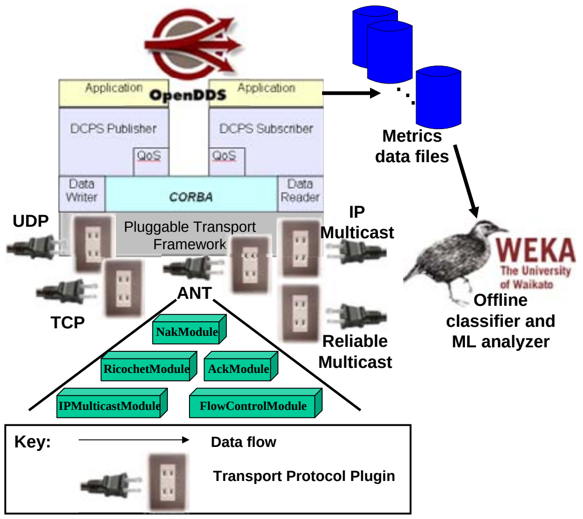


Fig. 2. **Current Research Prototype**

be easily configured with the desired computing platform, OS, network topology, and network traffic shaping. Moreover, Emulab provides facilities to capture network bandwidth usage.

Table I outlines the points of variability for the Emulab experiments. The NAKcast timeout period configures the amount of time that elapses before a receiver notifies the sender of lost packets. The Ricochet R value determines the number of packets received by an individual receiver before error correction data is sent to other receivers. The Ricochet C value determines the number of receivers to which an individual receiver sends error correction data. Table II outlines the data that is being collected to classify and evaluate middleware performance. The ReLate2 value as defined in our current research [6] is a metric that evaluates both packet latency and reliability as determined by the number of packets received by an application.

Point of Variability	Values
# of data receivers	3 - 25
Frequency of sending data	10Hz, 25 Hz, 50 Hz, 100Hz
% end-host network loss	0 to 5 %
Processor speed	850 MHz, 3 GHz
Network speed	100 Mb/s, 1 Gb/s
Protocols used	NAKcast, Ricochet
NAKcast timeout	0.5, 0.1, 0.05, 0.025 seconds
Ricochet R value	4, 8
Ricochet C value	3, 6

TABLE I
Emulab Experiment Variables

V. Empirical Evaluations of Machine Learning Techniques for ADAMANT

The section presents analysis of the experimental data for purposes of machine learning, an overview of the machine learning techniques evaluated, and analysis of results from the learning techniques.

A. Subsampling the Data

Since we are concerned with reliability and latency we focused on the ReLate2 values to provide relevant data reduction. For a given environment configuration we selected the transport protocol and parameter settings with the lowest ReLate2 value. Using the values for the ReLate2 metric as an indicator of quality, the reduced data set can represent the best transport and parameterizations. For a set of 5 runs for each experimental setup with varying transport protocols and parameter settings, 5 examples are left when including only the data from the protocol and parameters that produced the lowest ReLate2 value for that run. Across all experiment configurations, roughly 150 examples are preserved.

Metrics	Units
Number of data updates received	integer
Latency of data updates	microseconds
Std. deviation of latency	microseconds
Maximum network bandwidth usage	bytes/sec
Minimum network bandwidth usage	bytes/sec
Average network bandwidth usage	bytes/sec
Network bandwidth usage	bytes
ReLate2 value	integer

TABLE II
Metrics Captured from Experiments

With this transformation, machine learning can start to select the most appropriate transport protocol and parameters settings. The transformed data will accurately identify a protocol that currently produced the best ReLate2 values. Looking from the other direction, it will identify when a class and parameterization will offer the best ReLate2 values, given a set of measurements in the network.

B. Learning Techniques Used

To explore the topic of which protocols responds best under different circumstances, three different techniques were used on the transformed data set. All three algorithms were run using 10 fold cross validation, which partitions the data into separate training and testing sets. The selected learning technique is trained on the training set and evaluated with the testing set.

The number of folds (*i.e.*, 10 in our case) represents the number of times the data is partitioned into different training and testing sets where the testing sets are mutually exclusive. The accuracy results are then averaged out over all folds, which helps provide greater coverage and avoids over-fitting of the classifier. Since the remaining data points from the transformations are not necessarily evenly distributed among the different experiment variables, cross fold validation is the best approach to maximizing data coverage while maintaining an unbiased accuracy [8].

The first learning technique we investigated is a decision tree (DT). This algorithm attempts to create a tree where a set of decisions leads down to a leaf node that can accurately classify a new example. A DT will attempt to produce the shortest and smallest tree possible while maintaining accuracy by looking for features that best split the data as completely as possible and use them closer to the root. DTs are designed for data sets with more than a binary set of classes, such as ours, where there are more than two possible classifications of an appropriate transport protocol and parameter settings.

The second technique we investigated is an Artificial Neural Network (ANN). An algorithm originally developed in the mid-80s, ANNs work well on sets with a small number of features and can produce accurate classifiers with medium sized data sets. ANNs can be used to craft a very precise classifier that can be fast to utilize. The learning produced when using ANNs are not as accessible as a DT since, however, the weights for the different classes from the back-propagation are difficult to transform into higher level abstractions for human comprehension.

The third technique we investigated is a Linear Logistic Regression Classifier (LLRC). LLRC uses a weighting of the features to produce a value that is converted into the correct class. LLRC is the most straightforward of the three approaches. The results from LLRC has increased comprehensibility as compared to ANNs as the weights have more meaning in this classifier along with the ability to optimize the classification for speed.

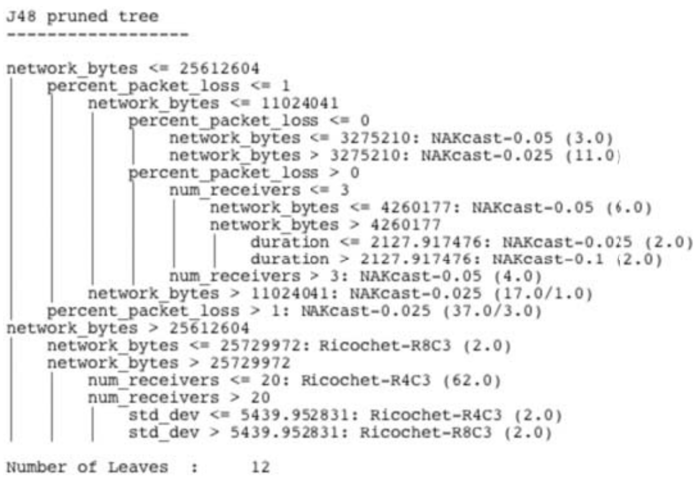
C. *Analysis of Results*

Applying the three techniques presented above on the reduced data set of 150 examples, we see clear differences in the results. The DT produced the best base accuracy of the tree at 87% and the worst area under the curve (AUC) score at .925, which is another measure of accuracy that removes the bias of false positives. The ANN produced an accuracy that was lower at 85.3%, but provided the highest AUC at .966. LLRC posted the lowest accuracy at 80% but also a higher AUC than DT at .935.

These results indicate that ANNs are the most robust classifier, but DTs post a higher base accuracy. In particular, DTs show their ability to over-fit the data too easily, whereas ANNs are more likely to stand up better in a real application. We are collecting more data to explore this assessment. LLRC appears to provide the worst results with the lowest base accuracy and only slightly better AUC than DTs.

While it appears one might choose ANNs for an application at the moment and LLRC is thus the least useful, the results from DTs answer and raise interesting questions. Due to the nature of DTs, one can look at the implications the tree finds about the features. As seen in Figure 3, the tree utilizes a relatively few features, *i.e.*, the amount of bandwidth used in bytes, the controlled variable of packet loss, and the controlled variable of number of receivers in the environment. While some of the controlled variables are utilized, the most important discriminator of the measured variables is the bandwidth usage.

These results motivate us to understand applications not just in terms of receivers and how likely applications will drop packets, but also how much the application utilizes the network. DT gives us a sense of how the experiments also may need to change. Classifying NAKCast using a DT produces a fairly complicated subtree. This complexity would seem to indicate that more coverage through additional experiments is needed to try and find a more simplified subtree. Moreover, more data coverage is also needed to gain greater confidence that the other subtree is as simple as it appears.



```
J48 pruned tree
------------------

network_bytes <= 25612604
|   percent_packet_loss <= 1
|   |   network_bytes <= 11024041
|   |   |   percent_packet_loss <= 0
|   |   |   |   network_bytes <= 3275210: NAKcast-0.05 (3.0)
|   |   |   |   network_bytes > 3275210: NAKcast-0.025 (11.0)
|   |   |   percent_packet_loss > 0
|   |   |   |   num_receivers <= 3
|   |   |   |   |   network_bytes <= 4260177: NAKcast-0.05 (6.0)
|   |   |   |   |   network_bytes > 4260177
|   |   |   |   |   |   duration <= 2127.917476: NAKcast-0.025 (2.0)
|   |   |   |   |   |   duration > 2127.917476: NAKcast-0.1 (2.0)
|   |   |   |   num_receivers > 3: NAKcast-0.05 (4.0)
|   |   network_bytes > 11024041: NAKcast-0.025 (17.0/1.0)
|   percent_packet_loss > 1: NAKcast-0.025 (37.0/3.0)
network_bytes > 25612604
|   network_bytes <= 25729972: Ricochet-R8C3 (2.0)
|   network_bytes > 25729972
|   |   num_receivers <= 20: Ricochet-R4C3 (62.0)
|   |   num_receivers > 20
|   |   |   std_dev <= 5439.952831: Ricochet-R4C3 (2.0)
|   |   |   std_dev > 5439.952831: Ricochet-R8C3 (2.0)

Number of Leaves  :     12
```

Fig. 3. **Initial Decision Tree**

VI. Concluding Remarks

Developers of large-scale QoS-enabled pub/sub middleware and applications face a number of challenges in dynamic environments. To address these challenges ADAMANT integrates and enhances QoS-enabled pub/sub middleware with adaptive transport protocols and machine learning. This combination of technologies provides a basis for maintaining specified QoS even in dynamic environments. Our initial research shows that artificial neural networks are the most robust machine learning technique evaluated. We are continuing our research to determine the optimal technique for ADAMANT.

References

[1] M. Balakrishnan, K. Birman, A. Phanishayee, and S. Pleisch. Ricochet: Lateral error correction for time-critical multicast. In *NSDI 2007: Fourth Usenix Symposium on Networked Systems Design and Implementation*, Boston, MA, 2007.

[2] B. Brynjulfsson, G. Hjalmtysson, K. Katrinis, and B. Plattner. Autonomic network-layer multicast service towards consistent service quality. In *AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, pages 494–498, Washington, DC, USA, April 2006. IEEE Computer Society.

[3] M. Caporuscio, A. Carzaniga, and A. Wolf. Design and evaluation of a support service for mobile, wireless publish/subscribe applications. *Software Engineering, IEEE Transactions on*, 29(12):1059–1071, Dec. 2003.

[4] P.-C. David and T. Ledoux. *Software Composition*, chapter An Aspect-Oriented Approach for Developing Self-Adaptive Fractal Components, pages 82–97. Springer LNCS, Berlin / Heidelberg, 2006.

[5] P. Grace, G. Coulson, G. S. Blair, and B. Porter. Deep middleware for the divergent grid. In *Middleware '05: Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware*, pages 334–353, New York, NY, USA, 2005. Springer-Verlag New York, Inc.

[6] J. Hoffert, A. Gokhale, and D. C. Schmidt. Evaluating Transport Protocols for Real-time Event Stream Processing Middleware and Applications. In *(submission to) the 11th International Symposium on Distributed Objects, Middleware, and Applications (DOA '09)*, Vilamoura, Algarve-Portugal, Nov. 2009.

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

[8] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36(1):111–147, 1974.

[9] S. Tarkoma and K. Raatikainen. State of the Art Review of Distributed Event Systems. Technical Report C0-04, University of Helsinki, 2006.

[10] P. Vienne and J.-L. Sourrouille. A middleware for autonomic qos management based on learning. In *SEM '05: Proceedings of the 5th international workshop on Software engineering and middleware*, pages 1–8, New York, NY, USA, 2005. ACM.