Adapting and Evaluating Distributed Real-time and Embedded Systems in Dynamic Environments

Joe Hoffert, Douglas Schmidt, and Annirudha Gokhale
Vanderbilt University, Nashville, TN, USA
{joseph.w.hoffert, d.schmidt, a.gokhale}@vanderbilt.edu*

ABSTRACT

Quality of Service (QoS)-enabled publish/subscribe (pub/-sub) middleware provides needed infrastructure for data dissemination in distributed real-time and embedded (DRE) systems. It is hard, however, to quantify the performance of mechanisms that support multiple interrelated QoS concerns, e.g.*, reliability, latency, and jitter. Moreover, once an appropriate mechanism is selected, it is hard to maintain QoS properties as an operating environment fluctuates since the mechanism might no longer provide the needed QoS. For DRE systems operating in such environments, adjustments to mechanisms supporting QoS must be both timely and resilient to unforeseen environments.*

*This paper describes our work to (1) define composite metrics to evaluate multiple interrelated QoS concerns and (2) analyze various adjustment techniques (*i.e.*, policy-based approaches, machine learning techniques) used for the QoS mechanisms of a DRE system in a dynamic environment. Our results show that (1) composite metrics quantify the support that mechanisms provide for multiple QoS concerns to ease mechanism evaluation and creation of related composite metrics and (2) neural network machine learning techniques provide the constant-time complexity needed for DRE pub/-sub systems to determine adjustments and the robustness to handle unknown environments.*

Keywords

C.2.4.b Distributed applications, C.3.d Real-time and embedded systems, D.2.8 Metrics/Measurement,

1. INTRODUCTION

QoS-enabled pub/sub middleware, such as the OMG Data Distribution Service (DDS) or the JAIN Service Logic Execution Environment (SLEE), is increasingly used in mission-critical distributed real-time and embedded (DRE) application domains ranging from shipboard computing environments to air traffic management systems, telecom and fractionated spacecraft constellations. Pub/sub middleware supports the dissemination of data throughout a DRE system via an anonymous subscription model that decouples event suppliers from event consumers and supports policies that affect the end-to-end system QoS. Various mechanisms in the middleware, such as transport protocols, can be used to support QoS and provide specified behavior.

It can be hard, however, to quantitatively evaluate multiple QoS concerns that interrelate with each other. For example, data reliability and jitter can be at odds since reliability often implies retransmission of data which can add to jitter. Evaluating transport protocols to handle multiple QoS concerns increases accidental complexity as compared to a single QoS concern. Evaluators must keep track of the different QoS aspects (*e.g.*, reliability and jitter) that are being evaluated and merge the results together.

Moreover, challenges arise when developing QoS-enabled pub/sub systems deployed in dynamic operating environments. Transport protocols used by the middleware to support 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 but might incur unacceptable latency when having to manage a large number of connections due to an increase in subscribers.

Managing transport protocols manually is not feasible in dynamic environments due to slow human response times exacerbated by the complexity of determining appropriate adjustments. QoS-enabled pub/sub DRE systems must be robust enough to anticipate that changes will occur in the operating environment and handle those changes appropriately. Several approaches to managing changes exist each with advantages and disadvantages.

This paper describes our ongoing work on *composite metrics*, which quantitatively measure multiple QoS concerns to evaluate pub/sub mechanisms for DRE systems. We build on previous composite metrics of reliability and average latency [1] to include the concerns of jitter, average network bandwidth usage, and network *burstiness*, *i.e.*, standard deviation of network bandwidth usage. Moreover, we analyze the composability of various QoS concerns and analyze techniques to manage adaptations for DRE systems in dynamic environments. In particular, we analyze how policy-based approaches and machine learning techniques help determine appropriate QoS mechanisms in a changing operating envi-

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

ronment in terms of (1) bounded time complexity in searching for a solution, (2) accuracy of solution for known operating environments, (3) robustness to unknown operating environments, and (4) accidental development complexity.

2. MOTIVATING EXAMPLE - AMBIENT ASSISTED LIVING IN SMART CITIES

To motivate the need for managing QoS interactions and providing timely adjustments of transport protocols for QoS-enabled pub/sub middleware, this section describes research challenges associated with *Smart City Ambient Assisted Living* (SCAAL) applications, which combine *Ambient Assisted Living* (AAL) in the context of a *smart city* (SC). The goal of a smart city is to dissolve the computational infrastructure and establish ubiquitous, context-aware services in a metropolitan area [2].

An example SCAAL scenario involves elderly people navigating a large metropolitan area equipped with multiple technological devices that aid in various aspects of the person's mobility, sensory enhancement, communication, and monitoring. In particular, 3 dimensional high-resolution health monitoring equipment is periodically sending data. A *personal data center* (PDC) publishes and subscribes to the data that is being managed by the personal devices and interfaces with the smart city by publishing and subscribing to data from the ambient environment.

The environment in which the PDC operates is dynamic since (1) the elderly people move through space in the smart city and update personal information in time and (2) the smart city enhances and updates the amount and kind of data that it provides as it moves through time. Our research focuses on (1) metrics to evaluate transport protocols in support of multiple QoS concerns and (2) adjustment techniques for QoS that a PDC device must manage in a SCAAL application. To support multiple QoS aspects of the ambient data (such as 3-dimensional health monitoring information) the PDC presents the following challenges:

Challenge 1: Managing interacting QoS requirements with constrained resources. The PDC must manage multiple interacting QoS requirements, *e.g.*, data reliability so that enough data is received to be useful and low latency and jitter for soft realtime data so that detailed 3-dimensional health monitoring data do not arrive after they are needed. The streamed data must be received soon enough so that successive dependent data (such as dependent MPEG B and P frame data being received before the next I frame makes them obsolete) can also be used. Moreover, the PDC must balance the interacting QoS requirements with limited resources, *e.g.*, network bandwidth.

Challenge 2: Timely Adaptation Due to timeliness concerns of DRE systems such as SCAAL applications, the PDC must adjust in a timely manner as the environment changes. If the PDC cannot adjust quickly enough it will fail to perform adequately and critical data such as 3-D health information will not be received in time. As the amount of data relevant to the SCAAL application fluctuates and the demand for information varies, the PDC must be configured to accommodate these changes with appropriate responsiveness to maintain a minimum level of service. Configuration changes must not only be timely in general but they must also be bounded so that critical information (such as health monitoring) is not impeded.

3. SOLUTION APPROACH: COMPOSITE METRICS AND MACHINE LEARNING FOR AUTONOMIC ADAPTATION

This section describes the research we are conducting with (1) composite metrics to evaluate transport protocols as QoS mechanisms and (2) adaptation techniques to determine appropriate transport protocol adjustments in support of QoS as the operating environment changes. This research is done in the context of the *ADAptive Middleware And Network Transports* (ADAMANT) project described next.

3.1 Overview of ADAptive Middleware And Network Transports (ADAMANT)

ADAMANT leverages composite QoS metrics and machine learning along with several other technologies described below to resolve the challenges presented in Section 2.

• The Artificial Neural Network (ANN) machine learning technique helps ADAMANT address Challenges 1 and 2 in Section 2 by (1) using composite metrics to determine how a transport protocol will address multiple QoS concerns for a particular environment and (2) selecting an appropriate transport protocol and protocol parameters at runtime in a timely manner given a specified QoS and a particular environment configuration.

The ANN is trained using features for several different environment configurations along with the evaluations of several transport protocols using composite metrics. The ANN interpolates and extrapolates its learning based on the current environment configuration, which might not have been included in the supervised training. We discuss adaptation approaches and our selection of using ANNs in Section 3.2.

• Composite metrics allow several QoS properties to be evaluated simultaneously. Composite metrics are used to evaluate transport protocols in multiple operating environments to determine the best protocol and parameter settings for a particular environment. We present several new composite metrics and experimental results in Section 3.3.

The architecture of the ADAMANT solution approach is shown in Figure 1, which highlights the use of composite QoS metrics to evaluate transport protocols and machine learning to select an appropriate transport protocol when the current one is insufficient. The ADAMANT controller receives QoS monitoring data and feeds the data to the protocol optimizer which uses an ANN to determine the appropriate transport protocol and settings. The protocol optimizer then sends the protocol selection to the controller to make appropriate adjustments to the system.

3.2 Evaluating Adaptation Approaches for SCAAL Applications

Several approaches can be used to adapt transport protocols for QoS-enabled pub/sub systems operating in dynamic environments. Below we evaluate (1) policy-based approaches, (2) reinforcement learning, and (3) supervised machine learning with and without bounded search times. We also present evaluation criteria relevant to developing and deploying an adaptation approach for SCAAL applications, including (1) boundedness in searching for a solution, (2) accuracy of solution for known environments, (3) robustness to unknown operating environments, and (4) accidental development complexity.

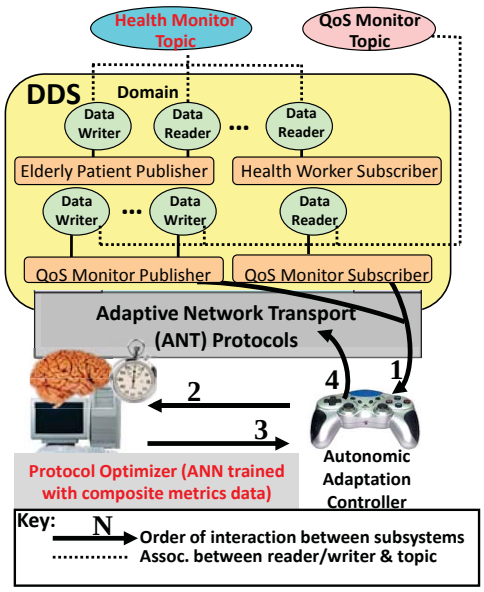

Figure 1: Architecture for ADAMANT Solution Approach

3.2.1 *Evaluating Policy-based Adaptation Approaches*

Policy-based approaches provide a straightforward way to determine optimal transport protocols for a given operating environment. After certain operating conditions are checked and met the system can be directed by the policies to alter its behavior. Figure 2 shows an example where the application checks for three environment aspects: (1) percentage loss in the network (*i.e., network_loss_percent*), (2) number of data receivers (*i.e., num_receivers*), and (3) the rate at which data is published (*i.e., sending_rate*).

```
if (network_loss_percent == 1
    && num_receivers < 5
    && sending_rate < 0.01) {
  transport_framework->use (transport1);
} else if (network_loss_percent == 5
           && num_receivers < 5
           && sending_rate < 0.01) {
  transport_framework->use (transport2);
} else if (network_loss_percent == 10
           && num_receivers < 5
           && sending_rate < 0.01) {
  transport framework->use (transport3);
}
```

Figure 2: Policy-based Example

Policy-based approaches can be optimized since the bounded number of (1) conditions that are checked and (2) the behaviors used to direct the system are explicitly identified. As shown in Figure 2, a switch statement or nested if statements in a programming language can be used to implement policy-based approaches. In general, policy-based approaches can provide boundedness in searching for an adaptation solution and therefore address the boundedness evaluation criterion for adaptation approaches (*e.g.*, switch statements can be optimized to constant time performance). Policy-based approaches also are highly accurate for known solutions since developers can codify the exact behavior needed for a known environment, thereby addressing the evaluation criterion for accuracy in known environments.

Policy-based approaches, however, do not provide robustness in the face of conditions not considered *a priori*. Policy-based approaches must have complete knowledge of all conditions that can affect the system so that this knowledge can be imperatively codified. If conditions exist that were not anticipated then unexpected system behavior can occur, which can be disastrous for mission-critical pub/sub DRE systems, so policy-based approaches do not address the robustness evaluation criterion for adaptation approaches.

Even when all relevant conditions are considered and all appropriate responses are codified, manually managing the conditions and responses for policy-based approaches increases accidental complexity. Figure 2 presents only three operating environment aspects that are checked. Since each aspect can take an infinite range of values there is an infinite number of combinations that can be checked. Even using ranges of values can lead to infinite number of combinations. Moreover, if the policies need to be modified the chance of introducing an error increases with the number of aspects considered along with the number of ranges of values for each aspect. Policy-based approaches therefore do not address the accidental development complexity criterion for adaptation approaches.

3.2.2 *Evaluating Reinforcement Learning*

Reinforcement learning provides robustness and flexibility when not all conditions and appropriate system responses are known *a priori*. Reinforcement learning approaches leverage high-level abstract guidance for a proposed solution, *e.g.*, determining the solution to be good or bad. For example, reinforcement learning sets certain system behaviors as goals and uses positive and negative reinforcements to guide the resolution of system behavior as change in an operating environment occurs [3].

Reinforcement learning explores the possible solution space to determine generalized solutions of the negative and positive reinforcements given. Reinforcement learning is thus unbounded in its determination of an appropriate response due to online exploration of the solution space and modification of decisions while the system is running. As indicated in [4], performance of reinforcement learning benefits from an additional run-time initialization period before system startup. Reinforcement learning therefore does not address the evaluation criterion for boundedness when searching for an adaptation solution.

Reinforcement learning generalizes knowledge gained from positive and negative reinforcements of multiple proposed solutions. With this generalization comes a loss of information for the specific solutions that have been tried. Reinforcement learning thus does not entirely address the criterion for accuracy in known environments.

In contrast, the generalization of knowledge for reinforcement learning does allow the approach to interpolate and extrapolate from solutions of known environments to unknown environments. Accordingly, reinforcement learning addresses the criterion for robustness to unknown operating environments.

Even when all conditions of the operating environment are known and all appropriate responses determined, reinforcement learning can manage the conditions and appropriate responses rather than forcing developers to address these

areas programmatically. Knowing how to respond to various operating environments is resolved by the reinforcement learning approach itself. Reinforcement learning thus addresses the criterion of accidental development complexity.

3.2.3 *Evaluating Supervised Machine Learning*

Supervised machine learning techniques classify new examples while incorporating generalized knowledge from previous examples. These techniques are supervised by being provided solutions to the problem which they use to expand the generalized knowledge. Supervised machine learning techniques generally have an offline training period to build up knowledge and then are used online when a system is running. Below we evaluate two common supervised machine learning approaches of decision trees (DTs) and artificial neural networks (ANNs).

DTs build a tree structure that branches on decisions which lead down to a leaf node that can accurately classify a new example [5]. A DT generates decision branches that split the data. Decisions that split the data more evenly are placed closer to the root of the tree. In general, a DT can be unbounded in the levels of the tree that is generated.

The attributes that are used for classification to generate the tree can be combined in an exponential number of ways. These combinations are then used to determine branches in the tree. As shown in Figure 3, the attribute *network_bytes* is used multiple places in the tree to branch the tree. DTs

```
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)
```

Figure 3: A Decision Tree For Determining Appropriate Protocol

thus do not address the boundedness criteria for adaptation.

Moreover, as with machine learning in general, the knowledge obtained is generalized to apply to a wide variety of operating environments. This generalization of knowledge implies that DTs do not perfectly address the accuracy criterion for known operating environments. They can, however, provide solutions to environments not seen previously and therefore address the robustness criterion. In addition, DTs automatically capture branching decisions to determine an appropriate transport protocol configuration, thereby addressing the criterion of accidental development complexity.

An ANN is a supervised machine learning technique that is modeled on the interaction of neurons in the human brain [6]. As shown in Figure 4, an ANN has an input layer for relevant aspects of the operating environment, *e.g.*, percent network loss, sending rate. An output layer represents the solution that is generated based on the input. Connecting the input

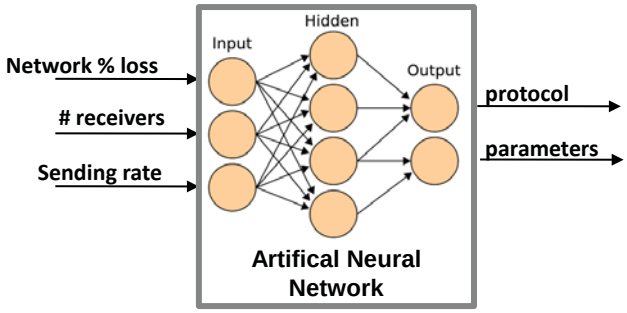

Figure 4: Artificial Neural Network For Determining Appropriate Protocol

and output layers is a hidden layer. As the ANN is trained on inputs and correspondingly correct outputs, it strengthens or weakens connections between the layers to generalize based on the inputs and outputs.

Figure 4 also shows how an ANN can be configured statically in the number of hidden layers and the number of nodes in each layer that directly affects the processing time complexity between the input of operating environment conditions and the output of an appropriate transport protocol and settings. This static configuration structure supports bounded response times. ANNs thus address the boundedness criterion for generating a solution.

As ANNs generalize the knowledge from the supervised training, they provide interpolation and extrapolation of known training sets to handle conditions for which the techniques have not been trained. Although ANNs do not entirely address the accuracy criterion for known environments they do address the robustness criterion for unknown operating environments. Moreover, since ANNs automatically encapsulate the generalization of knowledge across supervised training data, they address the criterion of accidental development complexity since developers need not determine appropriate transport protocol configurations given operating environment features.

Based on the evaluation of adaptation approaches in this section, we have determined ANNs are a promising adaptation approach for DRE systems like SCAAL. ANNs support (1) bounded time complexity for determining a solution, (2) robustness for unknown operating environments, and (3) reduction of accidental development complexity. We are researching ways to increase the accuracy of ANNs for known environments to be comparable with policy-based approaches.

3.3 Composite Metrics for SCAAL Applications

Challenge 1 in Section 2 outlines the interacting QoS concerns of reliability, latency, and jitter for SCAAL applications. Our prior work [1] showed how to quantitatively evaluate reliability and average latency via the *ReLate2* metric, which multiplies the average latency by the percent packet loss as follows:

$$ReLate2_p = \frac{\sum_{i=1}^{r} l_i}{r} \times (\frac{t-r}{t} \times 100 + 1)$$

where p is the protocol being evaluated,
r = number of packets received,

l_i = latency of packet i,
and t = total number of packets sent.

This metric produces a numeric value that increases with either an increase in the average latency of packets received or an increase in the percentage of packets lost. When there are no lost packets ReLate2 values become the average latency. Lower ReLate2 values are desirable.

This paper extends the ReLate2 metric to include other QoS properties relevant to DRE systems. In particular, *jitter* (*i.e.*, standard deviation of the latency of network packets) is also an important QoS consideration for applications using multimedia data, such as SCAAL's personal surveillance video or 3-dimensional health monitoring information. For example, as outlined in Section 2, late arriving MPEG data can be worse than not receiving the data at all. Jitter provides a way to measure the variance of data arrival times and is an important QoS consideration for data that depends on preceding or succeeding data.

Our new *ReLate2Jit* metric extends the original ReLate2 metric to include jitter. ReLate2Jit yields a numeric value that can quantifiably compare the performance of transport protocols wrt reliability, average latency, and jitter. ReLate2Jit values increase with an increase in jitter and low values are more desirable than high values. We calculate the standard deviation of the packet arrivals and multiply this value by the ReLate2 metric as follows:

$$ReLate2Jit_p = ReLate2_p \times \sigma_p$$

where p is the protocol being evaluated and
σ_p = standard deviation of packet latency times for protocol p.

We present experimental results using the ReLate2Jit metric. Our experimental environment is similar to the one used for our ReLate2 results [1], except that we use the OpenSplice DDS rather than OpenDDS. We send data from a publisher to subscribers while varying the sending rate, the percent loss in the network, and the number of subscribers.

We focus only on the transport protocols from previous work that balanced reliability and latency, *e.g.*, the NAKcast protocol with a retransmission timeout set to 0.05 and 0.025 seconds and the Ricochet protocol with the R parameter set to 4 and 8 and the C parameter set to 3. Ricochet's R value determines how many packets are received before an error correction packet is sent out to the other receivers. Ricochet's C value determines how many other receivers are sent the error correction packet.

Figure 5 shows results of using the ReLate2Jit metric for an operating environment with 3 receivers, 1% network loss, and a sending rate of 100Hz. The results show that the Ricochet protocol performs well when considering reliability, average latency, and jitter compared against NAKcast. The ReLate2Jit values between Ricochet and NAKcast are not as profound for lower sending rates although Ricochet generally outperforms NAKcast. As sending rates increase Ricochet's jitter decreases proportionately accounting for greater disparity between Ricochet and NAKcast at higher rates.

Being able to predict and provision for adequate resources is an important aspect of DRE systems. If allocated resources are inadequate then DRE systems running in resource constrained environments will not perform as intended, so that related QoS (such as reliability and latency) will not be met. Network bandwidth is an important resource

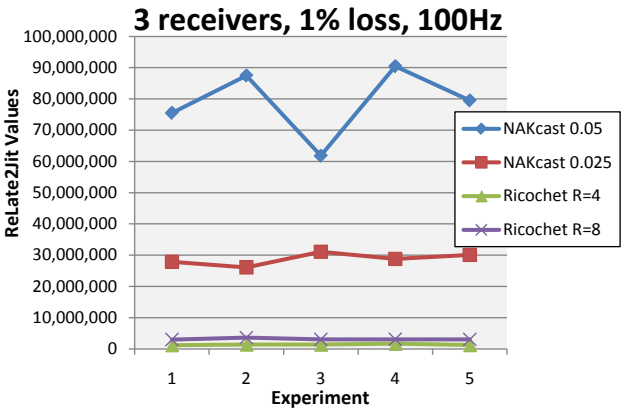

Figure 5: ReLate2Jit for 3 Receivers, 1% Network Loss, and 100Hz Sending Rate

consideration in DRE systems since it must be provisioned and managed appropriately. Moreover, as ultra-large-scale systems [7] become more prevalent, changes in network resources, *e.g.*, bandwidth, will become more dynamic and require more online adjustments.

To evaluate the concerns of reliability, latency, and network bandwidth usage we also have developed the *ReLate2Net* and *ReLate2Burst* composite metrics. ReLate2Net multiples the ReLate2 metric by the average network bandwidth usage per second to determine how a transport protocol balances reliability, latency, and network bandwidth. ReLate2Burst multiples the ReLate2 metric by the network bandwidth's *burstiness*, *i.e.*, the standard deviation of average bandwidth usage per second of time, to determine how a transport protocol balances reliability, latency, and packet burstiness.

The inclusion of network bandwidth information with the ReLate2 metric provides guidance for DRE systems in evaluating transport protocols and appropriately provisioning constrained DRE systems to function as needed in dynamic environments. Moreover, for ULS systems that must manage fluctuating network bandwidth capacity the ReLate2Net and ReLate2Burst metrics can be used to select an appropriate transport protocol.

Figures 6 and 7 show results of using the ReLate2Net and ReLate2Burst metrics respectively for an operating environment with 3 receivers, 1% network loss, and a sending rate of 100Hz. The ReLate2Net results highlight that the Ricochet protocol uses considerably more network bandwidth on average than NAKcast which is to be expected. Note that NAKcast with a shorter retransmission timeout, *i.e.*, 0.025 seconds, has a consistently lower ReLate2Net value than does NAKcast with a longer retransmission timeout, *i.e.*, 0.05 seconds. While NAKcast 0.025 does use more network bandwidth on average than NAKcast 0.05, it also has a lower average latency since lost messages are requested sooner. The ReLate2Burst results mimic the ReLate2Net results for this environment configuration.

4. RELATED WORK

This section compares our work on composite metrics and evaluating adaptation approaches within ADAMANT for dynamic applications like SCAAL with related work.

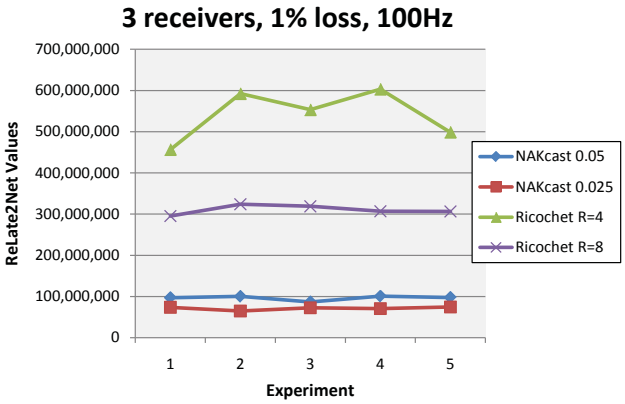

Figure 6: ReLate2Net for 3 receivers, 1% network loss, and 100Hz sending rate

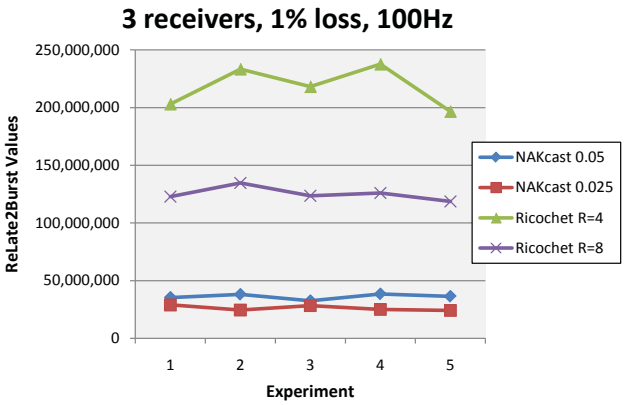

Figure 7: ReLate2Burst for 3 receivers, 1% network loss, and 100Hz sending rate

Grace *et al.* [8] describe an architecture metamodel for adapting components that implement coordination for reflective middleware distributed across peer devices. This work also investigates supporting reconfiguration types in various environmental conditions. The architecture metamodel, however, only provides infrastructure for autonomic adaptation and reconfiguration and does not directly address the challenges in Section 2.

Vienne and Sourrouille [9] present the Dynamic Control of Behavior based on Learning (DCBL) middleware that incorporates reinforcement machine learning in support of autonomic control for QoS management. System developers provide an XML description of the system, which DCBL then uses together with an internal representation of the managed system to select appropriate QoS dynamically. However, DCBL's focus on single computers does not address the challenge of distributed systems. Moreover, DCBL does not address the challenge of timely adjustments (*i.e.*, Challenge 2 in Section 2) as reinforcement learning can exhibit unbounded time complexities.

David and Ledoux have developed SAFRAN [10] 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. The SAFRAN component framework, however, only provides development support for maintaining specified QoS. The adaptive policies and component implementations are the responsibility of the application developer. Moreover, SAFRAN does not provide support for timely adaptation (*i.e.*, Challenge 2 in Section 2).

5. CONCLUDING REMARKS

Transport protocols satisfy needed multiple QoS properties for pub/sub DRE systems in dynamic environments. This paper presents (1) composite metrics to evaluate transport protocols in the context of multiple QoS concerns and (2) evaluations of adaptation approaches for pub/sub DRE systems in changing operating environments. Additional ADAMANT papers and all its source code are available at `www.dre.vanderbilt.edu/~jhoffert/ADAMANT`.

6. REFERENCES

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

[2] M. Chandy, O. Etzion, R. von Ammon, and P. Niblett, "07191 summary – event processing," in *Event Processing*, ser. Dagstuhl Seminar Proceedings, M. Chandy, O. Etzion, and R. von Ammon, Eds., no. 07191. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[4] X. Bu, J. Rao, and C.-Z. Xu, "A reinforcement learning approach to online web systems auto-configuration," in *ICDCS '09: Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 2–11.

[5] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[6] D. W. Patterson, *Artificial Neural Networks: Theory and Applications*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[7] S. E. Institute, "Ultra-Large-Scale Systems: Software Challenge of the Future," Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep., 2006.

[8] P. Grace, G. Coulson, G. S. Blair, and B. Porter, "A distributed architecture meta-model for self-managed middleware," in *ARM '06: Proceedings of the 5th workshop on Adaptive and reflective middleware (ARM '06)*. New York, NY, USA: ACM, 2006, p. 3.

[9] 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*. New York, NY, USA: ACM, 2005, pp. 1–8.

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