Design and Run-Time Quality of Service Management Techniques for Publish/Subscribe Distributed Real-Time and Embedded Systems


Joe Hoffert
jhoffert@dre.vanderbilt.edu

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee
Client-server technology may not suffice for all DRE systems => move towards publish/subscribe middleware

- *i.e.*, client-server & pub/sub are complementary technologies

**Characteristics of Pub/Sub**
- Decouples location via anonymous pub/sub
- Decouples time via asynchronous, time-independent data distribution
- Decouples redundancy via unbounded # of senders/receivers

**Manifesto for QoS-enabled Pub/Sub**
The right data...to the right place...at the right time
Example: QoS-enabled Pub/Sub DRE Systems

- Net-centric & large-scale “systems of systems”
  - e.g., satellite systems, shipboard computing environments, emergency response systems
- Satisfying tradeoffs between multiple (often conflicting) QoS demands
  - e.g., security, timeliness, reliability
- Regulating & adapting to (dis)continuous changes in runtime environments
  - e.g., online prognostics, dependable upgrades, availability of critical tasks, dynamic resource management

DRE systems increasingly realized by composing loosely-coupled services (e.g., pub/sub)
Challenges in Realizing DRE Pub/Sub Systems

Variability in the **solution space** (both design- and run-time)
- Diversity in platforms, languages, protocols & tool environments
- Enormous accidental & inherent complexities
- Continuous evolution & change
- Management of diverse QoS requirements

Focus on QoS Management

**Data priority**
- Inter-arrival data spacing

**Data reliability**
- Ordered data

**Data redundancy**
- Data for late arriving readers

**Determining liveness**
- Provisioning of data resources

**Data with time deadlines**
- Support Systems
Overview of QoS Management Focus Areas

My PhD dissertation addresses 4 aspects of QoS management complexity.

1. Developed model-based techniques to reduce manual effort & ameliorate accidental complexities in deploying pub/sub DRE systems.

2. Designed new composite metrics & a flexible middleware framework to evaluate & benchmark QoS mechanisms.

3. Designed machine learning-based adaptation logic to provide accurate configurations & predictable response times in flexible envs.

4. Designed monitoring mechanisms & improved machine learning-based logic to improve adaptation accuracy in dynamic envs.
1. QoS Configuration Development Support:

QoS configurations can have numerous entities & QoS policies; how can we help DRE developers manage the complexity of developing configurations?
QoS Management Focus Areas Overview (cont.)

2. Evaluation of QoS mechanisms:

Several QoS mechanisms are available; how can we help developers evaluate QoS mechanisms for pub/sub middleware?
3. QoS Configuration for cloud computing environments:

Cloud computing resources which affect QoS aren’t known until runtime; how can we configure the middleware based on resources provided?
4. QoS adaptation in dynamic environments:

As environments or operating conditions change, QoS can diminish; how can we adapt the middleware to support predictable QoS?
QoS Management Focus Areas Overview (cont.)

Presented solutions to these in qualifying exam; briefly review here

Focus Area 1
Manually QoS Configuration Techniques
QoS Configuration Validation

Focus Area 2
Protocol 1 Protocol 2
Operating environment

System Lifecycle Timeline

QoS Mechanism 1
(adequate)
QoS Mechanism 1
(inadequate)
QoS Mechanism 2
(adequate)

System Execution Timeline
(initial environment)
(environment modification)
DQML addresses the QoS configuration management challenges of
(1) Correctly specified QoS properties,
(2) Correctly managed related & interacting QoS, and
(3) Implementation artifacts that accurately represent design.
## DQML Related Publications & Presentations

### Book Chapter

### Conference Publications


### Poster Publications


FLEXible Middleware & Transports (FLEXMAT)

Evaluate transport protocols with multiple operating environments using FLEXMAT testbed:

- OpenSplice, OpenDDS
- Various # of senders, % loss, sending rate
- Standard & custom protocols
- Leverage FLEXMAT testbed integrated with DDS implementations
- Leverage composite QoS metrics

FLEXMAT addresses the challenges of
(1) Supporting multiple “antagonistic” QoS via new, custom protocols &
(2) Understanding how environments affect multiple QoS concerns
FLEXMAT Related Publications & Presentations

Conference Publications

Workshop Publications
This was the focus area in my qualifying exam that I proposed to complete my dissertation.
This is a new focus area that I included while completing my dissertation.
Focus Area 3: Configuring DRE Systems in Flexible Envs.

Cloud Computing for DRE systems?

• Resources provided as service
  – Resources on demand
  – “Pay-as-you-go” usage fee
  – Computing resources
    • CPUs, RAM
  – Networking resources
    • Bandwidth, network latency
• Popular implementations
  – Amazon Elastic Compute Cloud (EC2), Google App Engine, GoGrid, AppNexus, Emulab
  – OS, Database, RAM, CPU, Disk space, cores, load balancing, applications (e.g., Apache, Facebook servers), bandwidth

Not straightforward to use Cloud in DRE systems
Motivating Example: Search & Rescue Missions (1/2)

DRE Cloud Scenario

– Regional disasters (e.g., hurricane, flooding)
– Survivors trapped
– Search & rescue mission initiated
– Search application fuses multiple sensor streams
  • Thermal scans from unmanned aerial vehicles (UAVs)
  • Video from existing camera infrastructure
  • Data streams sent to ad-hoc datacenter for fusion & dissemination
Datacenter Requirements

– Operate in flexible environments, e.g.,
  • Support multiple missions & applications
  • Varying # of senders, receivers
  • Local resources unavailable – adapt to leverage available resources
  • Cloud resources, e.g., network bandwidth, CPU speed, RAM

– Support Multiple QoS
  • Reliability & latency
    • e.g., video & streamed thermal scans
    • Multimedia data
Challenges for Datacenter in Cloud Environments (1/3)

Challenge 1: Reduction of Development Complexity

Developing adaptive behavior is challenging:

- Inherent complexity – designing appropriate responses for environment
- Accidental complexity – transforming & managing appropriate responses from design to implementation

Increased development complexity reduces availability, assurance, and portability.
Challenge 2: Accurate Configuration in Cloud Environments

Environment resources unknown \textit{a priori} make static configuration inadequate.

Cloud computing infrastructure

Ad-hoc datacenter

Available resources

TCP/IP

UDP/IP

Multicast

Custom protocol

QoS mechanisms

Inaccurate configuration can result in loss of life & property.
Challenge 3: Timely Configuration in Cloud Environments

DRE systems require timely configuration

Cloud computing infrastructure

- Ad-hoc datacenter
- Available resources
- TCP/IP
- UDP/IP
- Multicast
- Custom protocol
- QoS mechanisms

Untimely configuration can result in loss of life & property
### Related Research

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Journal/Publication Details</th>
</tr>
</thead>
</table>
# QoS in Cloud Environments: Related Research

<table>
<thead>
<tr>
<th>Related Research</th>
</tr>
</thead>
</table>

Good for **developing** autoconfiguration applications.
QoS in Cloud Environments: Related Research

<table>
<thead>
<tr>
<th>Related Research</th>
</tr>
</thead>
</table>

Good for autoconfiguration when **timeliness is not a driving concern**
Properties for QoS Support in Cloud Environments

What properties help us assess research to configure QoS in cloud environments?

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed Configuration</td>
<td>Does the technique help autoconfiguration of QoS across machine boundaries?</td>
</tr>
<tr>
<td>Online Configuration</td>
<td>Does the technique perform configuration adjustments while system is running?</td>
</tr>
<tr>
<td>Timely Configuration</td>
<td>Does the technique provide bounded-time – ideally, constant-time – response?</td>
</tr>
</tbody>
</table>
QoS in Cloud Environments: Related Work

Does the technique autoconfigure across machines?

Distributed configuration

Local configuration

Online configuration

Static configuration

Bounded time complexity

Unbounded time complexity

Bu Imtiaz

Grace David

Zoiti Eustache

Valetto
QoS in Cloud Environments: Related Work

Does the technique provide online configuration?
QoS in Cloud Environments: Related Work

Does the technique provide bounded time complexity?
Current gap makes it *hard* for DRE systems in cloud environments to configure QoS in a timely manner.
Solution Approach: Bounded time, accurate configuration of transport protocols for QoS-enabled pub/sub middleware in ADAMANT.
Some Configuration Approaches Considered

```java
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);
} else if (network_loss_percent == 1
```

Policy-based configuration

J48 pruned tree

---

network_bytes <= 25612604
network_bytes <= 11024041
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)
```
Initial Evaluation of Configuration Approaches

Evaluated approaches based on:
• Boundedness/time complexity
• Accuracy for environments known at training time
• Accuracy for environments unknown \textit{a priori}
• Complexity of managing environments with responses

<table>
<thead>
<tr>
<th>Approach</th>
<th>Boundedness</th>
<th>Accuracy (known)</th>
<th>Accuracy (unknown)</th>
<th>Development complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy based</td>
<td>Yes</td>
<td>Perfect (100%)</td>
<td>Low (default)</td>
<td>Medium -High</td>
</tr>
</tbody>
</table>
Initial Evaluation of Configuration Approaches

Evaluated approaches based on:

- Boundedness/time complexity
- Accuracy for environments known at training time
- Accuracy for environments unknown \textit{a priori}
- Complexity of managing environments with responses

<table>
<thead>
<tr>
<th>Approach</th>
<th>Boundedness</th>
<th>Accuracy (known)</th>
<th>Accuracy (unknown)</th>
<th>Development complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy based</td>
<td>Yes</td>
<td>Perfect (100%)</td>
<td>Low (default)</td>
<td>Medium -High</td>
</tr>
<tr>
<td>Reinforcement Learning</td>
<td>No</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>
## Initial Evaluation of Configuration Approaches

Evaluated approaches based on:
- Boundedness/time complexity
- Accuracy for environments known at training time
- Accuracy for environments unknown *a priori*
- Complexity of managing environments with responses

<table>
<thead>
<tr>
<th>Approach</th>
<th>Boundedness</th>
<th>Accuracy (known)</th>
<th>Accuracy (unknown)</th>
<th>Development complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy based</td>
<td>Yes</td>
<td>Perfect (100%)</td>
<td>Low (default)</td>
<td>Medium -High</td>
</tr>
<tr>
<td>Reinforcement Learning</td>
<td>No</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>Yes (data dependent)</td>
<td>High (99%)</td>
<td>High (87%)</td>
<td>Low</td>
</tr>
</tbody>
</table>
## Initial Evaluation of Configuration Approaches

Evaluated approaches based on:
- Boundedness/time complexity
- Accuracy for environments known at training time
- Accuracy for environments unknown *a priori*
- Complexity of managing environments with responses

<table>
<thead>
<tr>
<th>Approach</th>
<th>Boundedness</th>
<th>Accuracy (known)</th>
<th>Accuracy (unknown)</th>
<th>Development complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy based</td>
<td>Yes</td>
<td>Perfect (100%)</td>
<td>Low (default)</td>
<td>Medium - High</td>
</tr>
<tr>
<td>Reinforcement Learning</td>
<td>No</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>Yes (data dependent)</td>
<td>High (99%)</td>
<td>High (87%)</td>
<td>Low</td>
</tr>
<tr>
<td>Neural Network</td>
<td>Yes (constant)</td>
<td>Perfect (100%)</td>
<td>High (85%)</td>
<td>Low</td>
</tr>
</tbody>
</table>
### Initial Evaluation of Configuration Approaches

Evaluated approaches based on:
- Boundedness/time complexity
- Accuracy for environments known at training time
- Accuracy for environments unknown \textit{a priori}
- Complexity of managing environments with responses

<table>
<thead>
<tr>
<th>Approach</th>
<th>Boundedness</th>
<th>Accuracy (known)</th>
<th>Accuracy (unknown)</th>
<th>Development complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy based</td>
<td>Yes</td>
<td>Perfect (100%)</td>
<td>Low (default)</td>
<td>Medium -High</td>
</tr>
<tr>
<td>Reinforcement Learning</td>
<td>No</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>Yes (data dependent)</td>
<td>High (99%)</td>
<td>High (87%)</td>
<td>Low</td>
</tr>
<tr>
<td>Neural Network</td>
<td>Yes (constant)</td>
<td>Perfect (100%)</td>
<td>High (85%)</td>
<td>Low</td>
</tr>
<tr>
<td>SVM</td>
<td>Yes (constant)</td>
<td>Perfect (100%)</td>
<td>High (79%)</td>
<td>Low</td>
</tr>
</tbody>
</table>
ADAMANT incorporates:

**Artificial Neural Network (ANN)**
- Trained on protocol properties
- Interpolates/Extrapolates for new environments
- Determines optimal protocol/parameters
- Constant time performance

**Data Distribution Service (DDS)**
- OMG pub/sub standard, rich QoS support
- OpenDDS, OpenSplice implementations
  - Pluggable transport protocol frameworks
  - Open source

**Adaptive Network Transports (ANT) framework**
- Transport protocol framework
- Composable modules
- Fine-grained protocol control

---

**Protocol Optimization**

Interpolation between training data
1. ADAMANT queries environment for resources.

2. Resource information passed to ADAMANT.

3. ANN selects appropriate protocol in a timely manner & notifies ANT.

4. ANT configures the protocol for the middleware.

Key:
- Control interaction between subsystems
- Assoc. between reader/writer & topic
ADAMANT addresses challenge 1 (development complexity) via ANNs to manage protocol selection & implementation transformation

ANNs manage the development complexity of protocol management:

- Automatically manage inherent complexity of relationships between environment and protocols
- Used directly in implementations (i.e., avoids accidental complexity of developing implementation)

3 GHz CPUs, 1 Gb/s LAN
4 GB RAM

network loss
data sending rate

ANN

Adaptive Network Transport (ANT) Framework
ADAMANT addresses challenge 2 (accurate configuration) by overfitting ANN to data.

Overfitting data increases ANN’s accuracy for selecting appropriate protocol.

**ADAMANT accurately selects correct protocol**
ADAMANT addresses challenge 3 (timely configuration) via ANN w/ bounded constant-time response

ANNs are equation based:
• Equations based on nodes and connections
• Fixed number of inputs, hidden nodes, outputs (determined at off-line training time)
• Constant # of connections (determined at training time)
Empirical Results – Different Hardware ➔ Different Protocols

Experimental environment:
- Using protocols that balance reliability and low latency
  - IP Multicast w/ NAKs (NAKcast)
  - Modified FEC (Ricochet)
- Varied CPU speed, network bandwidth
- Conducted several training runs

**Difference in hardware triggers a difference in appropriate transport protocol**
Empirical Results - Accuracy

Experimental environment:
- 394 operating environments
- Varied CPU speed, network bandwidth, # of data receivers, sending rate
- Conducted several training runs
- ANN outputs tested against known correct responses

ANNs w/ 24 nodes provide most instances of 100% accuracy,
highest average accuracy with 2-fold cross-validation (78%)
Qualifying Exam Hypothesis for ADAMANT

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H1) Adjust for known environment</td>
<td>Hypothesize that ADAMANT will provide adjustment improvement for known environments at least 85% of the time</td>
</tr>
</tbody>
</table>

Using ANNs, ADAMANT provides 100% accurate adjustment for known environments.
Empirical Results – ANN Timeliness

Experimental environment:
- 394 operating environments
- Emulab: 3 GHz CPU, 2GB of RAM, Fedora Core 6 w/ real-time patches
- Sub 10 $\mu$s average response times for all ANN configurations
- Sub $\mu$s jitter for all ANN configurations

ADAMANT addresses the challenges of
(1) Development complexity via machine learning to determine protocols,
(2) Configuration accuracy via overfitted supervised machine learning &
(3) Configuration timeliness via equation-based machine learning
ADAMANT Related Publications & Presentations

Conference Publications


Workshop Publications
Focus Area 4: Adapting DRE Systems in Dynamic Envs.

Motivating Example: Smart City Ambient Assisted Living (SCAAL)

• Aging population increasing, # of health care workers decreasing
Motivating Example: SCAAL Application (cont.)

Scenario

• Aging population increasing, # of health care workers decreasing
• Increase elderly autonomy in urban areas via coordination of personal equipment & sensing/aware “smart cities”
Motivating Example: SCAAL Application (cont.)

Scenario

- Aging population increasing, # of health care workers decreasing
- Increase elderly autonomy in urban areas via coordination of personal equipment & sensing/aware “smart cities”
- Utilize personal data center to manage personal & environment data
Motivating Example: SCAAL Application (cont.)

Requirements

• Operate in dynamic environment, e.g.,
  • Varying # of senders, receivers
  • Varying network bandwidth, loss
  • Varying data sending rates (e.g., more updates for critical data)

• Support QoS as environment changes
  • Reliability & latency
    • e.g., high resolution health monitoring
  • Multimedia data
Challenges for QoS in Dynamic Environments (1/2)

**Challenge 1: Environment monitoring & update dissemination**

As environment changes, updates need to be propagated throughout the application.

- **Normal health information, Low update rate required**
  - Low sending rate

- **Doctor detects health anomaly, update rate increased**
  - High sending rate, More detailed information

(initial environment) ... (environment modification)

Elderly person traveling through smart city
Challenge 2: Optimal accuracy for unknown environments while maintaining timeliness

- Inaccurate adjustment could lead to reduced health or death

- Ideally want accuracy for unknown environments to rival accuracy for known environments

- Timeliness concerns need to be addressed while selecting an adequate QoS mechanism
## QoS in Dynamic Environments: Related Research

<table>
<thead>
<tr>
<th>Related Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrismTech’s Tuner application, <a href="http://www.opensplice.com">http://www.opensplice.com</a></td>
</tr>
<tr>
<td>Real-Time Innovations’ RTI Analyzer, RTI Scope, &amp; RTI Protocol Analyzer,</td>
</tr>
<tr>
<td><a href="http://rti.com/products/developer_platform">http://rti.com/products/developer_platform</a></td>
</tr>
</tbody>
</table>

Good for **manually checking** the run-time QoS status
**QoS in Dynamic Environments: Related Research**

<table>
<thead>
<tr>
<th>Related Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrismTech’s Tuner application, <a href="http://www.opensplice.com">http://www.opensplice.com</a></td>
</tr>
</tbody>
</table>

Good for **developing** adaptation applications
## QoS in Dynamic Environments: Related Research

### Related Research

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrismTech’s Tuner application, <a href="http://www.opensplice.com">http://www.opensplice.com</a></td>
<td></td>
</tr>
</tbody>
</table>
What properties help us assess research to support QoS in dynamic environments?

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Monitor environment</strong></td>
<td>Does the technique know when the environment has changed?</td>
</tr>
<tr>
<td><strong>Dynamic adaptation</strong></td>
<td>Does the technique perform adaptation adjustments while system is running?</td>
</tr>
<tr>
<td><strong>Timely adaptation</strong></td>
<td>Can the technique change to a more appropriate protocol in a timely manner?</td>
</tr>
</tbody>
</table>
QoS in Dynamic Environments: Related Work

- Does the technique monitor the system for changes?

Monitor environment

- No monitoring

Timely adaptation

Best-effort/no adaptation

Static configuration

Dynamic adaptation

- RTI
- PrismTech
- Bu Vienne Herssens
- Grace Caporuscio
QoS in Dynamic Environments: Related Work

- Monitor environment
- No monitoring
- Timely adaptation
- Best-effort/no adaptation
- Static configuration
- Dynamic adaptation

Does the technique adapt while the system is running?

- RTI
- PrismTech
- Bu Vienne Herssens
- Grace Caporuscio
QoS in Dynamic Environments: Related Work

Does the technique provide timely transition to support QoS?
Current gap makes it hard for DRE systems in dynamic environments to adapt QoS in a timely manner.
Solution Approach: Timely autonomic transport protocol adaptation for QoS-enabled pub/sub middleware in ADAMANT++
ADAMANT++ Architecture & Control Flow

1. Middleware disseminates environment feedback

2. Controller monitors feedback, sends to optimizer

3. Optimizer determines optimal protocol & settings (leveraging multiple machine learning techniques), returns to controller

4. Controller compares current & optimal settings, notifies ANT as needed

5. ANT dynamically updates the protocol and/or settings to maintain QoS

**Key:**
- N: Order of interaction between subsystems
- ——: Assoc. between reader/writer & topic

**Boldface:** Update from previous ADAMANT architecture for configuration
ADAMANT++ addresses Challenge 1 (disseminating updates) via environment monitoring topic.

ADAMANT++ leverages DDS to disseminate updates; QoS policies apply to monitoring topic.
ADAMANT++ addresses Challenge 2 (increasing accuracy) via Timely Integrated Machine Learning (TIML)

Timely Integration of Machine Learning (TIML)

TIML yields 8.6% accuracy increase for unknown environments (compared to just ANN), maintains timeliness
**Proposed Experiment: ADAMANT & Dynamic Environments**

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H2) Adjust for unknown environment</td>
<td>Hypothesize that ADAMANT will provide adjustment improvement for <em>unknown</em> environments more than 50% of the time</td>
</tr>
</tbody>
</table>

Leveraging TIML, ADAMANT++ provides 86% accuracy for unknown environments.
Empirical Results – TIML Timeliness

Experimental environment:

• 394 operating environments
• Emulab: 3 GHz CPU, 2GB of RAM, Fedora Core 6 w/ real-time patches
• 12 µs response times for determining to use ANN or SVM
• Jitter within timestamp resolution for ANN and SVM paths (i.e., +/- 1 µs)

TIML provides the predictable adaptation timeliness needed for DRE systems
Empirical Results – ANT Timeliness

Experimental environment:
- 394 operating environments
- Emulab: 3 GHz CPU, 2GB of RAM, Fedora Core 6 w/ real-time patches
- Sub 10 µs response times for switching between NAKcast and Ricochet
- No jitter for all ANT reconfigurations

![ANT Reconfiguration Times](chart.png)

ADAMANT++ addresses the challenges of
(1) Disseminating environment updates,
(2) Maximizing accuracy while maintaining timeliness
ADAMANT++ validates the three hypotheses from my qualifying exam:

1. > 85% accuracy for known environments (achieved 100%),
2. > 50% accuracy for unknown environments (achieved 86%),
3. Constant-time response

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H3) Provide bounded, constant time adaptation</td>
<td>Hypothesize that ADAMANT will adjust to new operating environment in bounded constant time (i.e., $O(1)$)</td>
</tr>
</tbody>
</table>

Leveraging equation-based machine learning and ANT, ADAMANT++ responds to new operating environments in constant time.

ADAMANT++ validates the three hypotheses from my qualifying exam:

1. > 85% accuracy for known environments (achieved 100%),
2. > 50% accuracy for unknown environments (achieved 86%),
3. Constant-time response
ADAMANT++ Sensitivity Analysis

Should adaptation *always* occur to get better QoS?
Are there times when the adjustment doesn’t warrant the adaptation?
How can we analyze the value of adapting?

Using threshold of 10% increase, we would reject adaptation for only 3% increase.

These decisions could be made at an application level & obeyed by ADAMANT++
## Enhancing Productivity & Flexibility for QoS-enabled Pub/Sub DRE Systems

<table>
<thead>
<tr>
<th>Focus Area</th>
<th>Challenge</th>
<th>Approach</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Valid QoS Design</strong></td>
<td>• Design-time QoS Validation</td>
<td>• DSML that validates QoS configuration &amp; generates implementation artifacts</td>
<td>• DQML</td>
</tr>
</tbody>
</table>
| **Evaluation of QoS Mechanisms** | • Run-time QoS Guidance & Flexibility                                  | • Pub/sub middleware with flexible protocol framework  
  • Composite metrics & empirical analysis                                                                 | • FLEXMAT  
  • ReLate2 metrics                                                             |
| **Autonomic Configuration for QoS** | • Manage QoS in Flexible Environments                                  | • Autonomic protocol config. in flexible resource envs  
  • Timely adaptation based on supervised learning                                                                                          | • ADAMANT                     |
| **Autonomic Adapation for QoS**   | • Manage QoS in Dynamic Environments                                   | • Autonomic adaptation of protocols in dynamic envs  
  • Increased accuracy via integration of supervised learning                                                                               | • ADAMANT++  
  • TIML                                                                        |

[www.dre.vanderbilt.edu/~jhoffert/research](http://www.dre.vanderbilt.edu/~jhoffert/research)
Summary of Publications & Presentations

Journal Publications


Conference Publications


Conference Publications (cont.)


Book Chapters

Summary of Publications & Presentations (cont.)

Workshop Publications


Summary of Publications & Presentations (cont.)

Poster Publications


Tutorials
Thank you for your time & attention.

Questions?

Soli Deo Gloria!