

# Aspects in Architectural Descriptions Thoughts about Quality of Service

David Emery  
DSCI  
Jan 2007

## Abstract

This paper frames an architectural problem in End-to-End Quality of Service (E2E QoS). I sketch an architectural description for the problem, identifying stakeholders, concerns and viewpoints for a description of an architecture to solve the E2E QoS problem. Finally, I provide an approach to applying Aspects to this problem, with the provisional assertion that Aspects map very nicely as implementation mechanisms for system-wide constraints identified in an architectural description. However, the application of Aspects to the Architectural Description itself is an open issue. Hopefully this example generates interesting discussion during the workshop.

## 1. Introduction

As one of the organizers of this workshop, I guess I'm not obligated to provide a paper detailing my current work and its relevance to gain admittance to the workshop ☺ Instead, the purpose of this paper is to explain my interest in aspects and software/systems architectural descriptions, through a particularly interesting problem I have been working on recently. I hope<sup>1</sup> this paper sparks some discussion during the workshop.

## 2. An End-to-End Quality of Service (E2E QoS) architectural problem

Consider a large scale distributed network, built over limited communications channels, with large numbers of various kinds of sensors, user workstations, etc. The communications load includes audio (VoIP), video teleconferencing, classic data and real-time sensor feeds (e.g. sensor video). The network's total value is based on the information it delivers to human users, and there exists an operations facility that wants to control the behavior of the full network to optimize network behavior based on a perception of what kind of information has the greatest value to which human recipients.

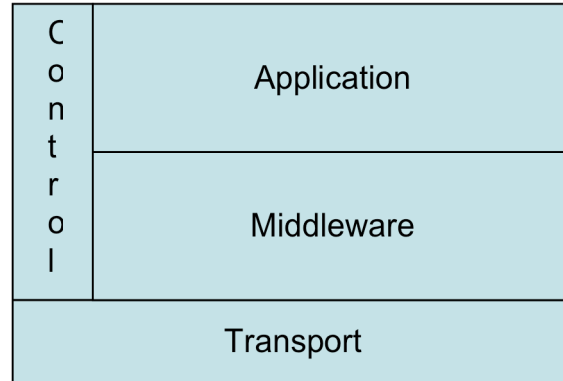
Traditionally, QoS problems have been phrased as transport-level networking problems. Mechanisms such as DiffServ ([http://en.wikipedia.org/wiki/Differentiated\\_services](http://en.wikipedia.org/wiki/Differentiated_services)) assume that some agent (application or human) can establish the priority of packets on the network, and the goal for the transport is to optimize network routing based on the

---

<sup>1</sup> Happy with the better English, Paul? ;-)

priority, etc., of packets through the network. In short, based on metadata and ‘static’ rules, the network makes appropriate routing/queuing decisions throughout the network.

Consider an architectural approach that can be represented as follows:



This model establishes 4 components, each with a specific responsibility for an End-to-End QoS solution:

Control provides the overall policy for the behavior of the system to meet the human objectives. Control talks to each of Applications, Middleware and Transport.

Applications have a (selfish) need to exchange data, based on an Application notion of importance. Applications are able to query Control Policy and Middleware-reported Status to adjust the Application’s behavior. (For example, if the application is told, “You’re not important right now and the network is saturated, throttle down your data requests,” the application can modify its own behavior.) The Application makes data requests to the Middleware, indicating appropriate QoS properties for each data stream.

Middleware accepts data streams from Applications. Based on the associated QoS properties the Application requests for the stream, the Control Policy in effect at the time, and the Middleware’s understanding of the state of the network, the Middleware either passes the request to Transport assigning the request to the appropriate DiffServ Code Point (providing Application with an ACK that the request will be honored), or notifies the Application that the Application’s request including QoS cannot be satisfied (a NAK). Middleware can also provide information/feedback from Transport back to Control. It’s the Middleware’s job to make sure that the set of Application requests meet Control’s Policy objectives.

Transport provides queueing and routing using protocols such as DiffServ. The various code points and routing protocols, etc, are established by the Control. The Transport accepts data packets with appropriate QoS properties from Middleware, and also provides status on the network back to Middleware. For purposes of this example I’m assuming the DiffServ approach is a given. I acknowledge that there are other approaches, but this is the one I understand so it’s the one I’ll use for this example.

### 3. Applying IEEE 1471/ISO 42010: Stakeholders, Concerns, Views & Viewpoints

Following the model of ISO/IEC 42010/IEEE 1471-2000, we can identify the following stakeholders for (this part of) the architecture:

- End Users (of applications)
- Operational Managers/Administrators
- Implementers
  - System Engineers/Architects
  - Control implementers
  - Application implementers
  - Middleware implementers
  - Transport implementers
- Verifiers
- Acquisition/Procurement

The following table shows some concerns for each stakeholder:

Stakeholder	Concerns
End users	<p>How do I tell the App my expectations for transmitting and receiving information?</p> <p>How does the App tell me if the system can't meet these expectations?</p> <p>How much control do I have over negotiating a QoS agreement with "the system"?</p>
Managers/ Administrators	<p>How do I specify global policy (i.e. what kinds of information are more important, what kinds of senders/receivers are more important?) What is the 'policy language'?</p> <p>How does the system tell me if it thinks it can implement my policy?</p> <p>How much detailed control do I have on various configuration parameters, and will the system tell me if tweaking X will break my more general policy statement?</p> <p>How do I monitor the dynamic execution of the system, both measured loads and policy fulfillment?</p> <p>What is the planning window for establishing/changing policy?</p> <p>What is the response time/lag time for feedback from the system?</p>
System Engineers/ Architects	<p>What are the system level components? <i>[My 4-component model is not necessarily binding, but is used here as a starting point for this example.]</i></p> <p>What are the key interfaces between components?</p> <p>What are the functional and non-functional requirements that drive the system (e.g. is QoS ever fully deterministic – can we guarantee that any QoS agreement is always met?)</p> <p>What are the hardware &amp; physics constraints (e.g. bandwidth, error rates, etc on the communications systems)?</p> <p>How do I model/ simulate the end-to-end system?</p>
Control	<p>What is the Language used by human managers/ administrators to</p>

Implementers	<p>express Policies?  How do I translate the human language to the interface I have with Applications, with Middleware and with Transports?  How do I get feedback from Transport on network loading? (What do I do with that feedback?)  What information (if any) do I receive from Middleware? From Applications?  How do I model/simulate a set of Policies to see if they are feasible to implement?  What is the rate of change for Policies? How long does it take to implement a Policy end-to-end? What is the latency/rate for feedback from each of Transport, Middleware, Applications?  How do I support QoS modeling/ simulation/ verification? What are my test points?</p>
Application Implementers	<p>How do I obtain QoS needs from my (human) user?  How do I associate QoS needs with data requests? What is the language that the Middleware provides to me?  How do I get a NAK from the Middleware? Is there more information than “No!” How do I negotiate QoS properties with Middleware (and with the human user) to reach a “ACK”?  When do I get Policies from Control?  What feedback do I need to give to Control (if any)?  How do I support QoS modeling/ simulation/ verification? What are my test points?</p>
Middleware Implementers	<p>What is the language I provide to Applications?  What is the language I exchange with Control?  What is the language I exchange with Transport?  How do I translate Application requests to Transport requests?  How do I verify that a given App-&gt;Transport request is consistent with Control’s Policy?  Can Transport provide me with any sense of guarantee/ likelihood for a given set of QoS values associated with a data request?  How does Transport inform me of changes to the current network state (e.g. “We’re pretty saturated right now, stuff at Priority 3 isn’t likely to get through like it normally does.”)  What is the rate-of-change for Policies from Control? How do I make sure that a new Control Policy is consistently understood before I start to follow it myself?  How do I support QoS modeling/ simulation/ verification? What are my test points?</p>
Transport Implementers	<p>What is the language I get from Control?  What is the language I exchange with Middleware?  How much information do I need to ‘guarantee’ a given data stream with a given set of QoS properties will succeed (i.e. how do I calculate an ACK)?  How do I calculate a NAK? What information do I provide back</p>

	<p>with a NAK?</p> <p>What is the timeliness of information I need to do my own job, and I need to provide to Middleware and Control?</p> <p>How do physics/hardware constrain what I can express? How many different permutations of QoS parameters can I support?</p> <p>How do I support QoS modeling/ simulation/ verification? What are my test points?</p>
Verifiers	<p>What are the typical scenarios for E2E QoS? What are the stressing scenarios?</p> <p>What is the traffic load?</p> <p>Who will provide the human inputs for test/ verification scenarios?</p> <p>What are the modeled/ simulated/ predicted results?</p> <p>What special verification regimes apply (e.g. safety-critical?)</p> <p>What constitutes a ‘realistic test environment’ in terms of numbers and types of communications channels, nodes, sensors, users, etc?</p>
Acquisition/ Procurement	<p>How will the various components be procured?</p> <p>What are the measures of effectiveness/ measures of performance that apply?</p> <p>What part of the development is ‘current practice’ and what is ‘risky state of the art’?</p> <p>How much will modeling/ simulation cost and what does it buy me (vs test &amp; verification at the end)?</p> <p>How will the system be tested/ verified?</p>

This leads to a candidate set of Viewpoints, addressing the following set of concerns:

### Use Cases

- What are examples of End-to-End QoS message exchanges?
- How does the human recipient ‘evaluate’ data s/he receives?
- How does the human sender understand the network, including establishing QoS needs?

### Data/Metadata

- What and how are data elements marked with QoS properties?
- What information is contained in a QoS request and a QoS response? (i.e. how does Middleware tell Application ‘no, you can’t get that level of QoS right now.’?)
- What information is provided by Control to each of the other components?
- What information is provided by Transport to Middleware?

### Behavior/Function

- What does Control do?
- What does Application do?
- What does Middleware do?

What does Transport do?

What are the operational roles for the system (e.g. applications user, system administrator) and where/how does each human role interact with the components?<sup>2</sup>

### **Performance Model**

What are the typical traffic patterns the system should anticipate?

What is a typical network architecture?

How will the QoS effects be measured through the network?

How will 'success' be measured?

## **4. Thoughts on the application of Aspects**

So what do "Aspects" have to do with all this? That's where I'm hoping for insight from this workshop. In thinking about this, whenever I've tried to apply an 'Aspects aspect' (pun intended ☺) to this problem and its architectural description, what I've come away with is a representation of QoS as ultimately a series of constraints on other parts of the problem. For example, all streams created by an Application should be QoS-aware, in that they (a) generate the QoS request metadata and (b) respond to a NAK from Middleware. Similarly Middleware must query Control for appropriate policy whenever it receives a request from Application, and must ensure that the answer (ACK or NAK) to Application and the request to Transport meet the constraints of Control's QoS Policies. Transport probably doesn't get to say "NAK" to Middleware. Its role is to queue and route packets as defined by Middleware's assignment of DiffServ Code Point

From an implementation perspective, some of these constraints seem to map quite well to Aspect-oriented implementation approaches. For example, Applications will design their code for a "QoS-managed aspect," that implements the queries to Control's Policy and that is prepared to negotiate with Middleware (including handling a NAK from Middleware.)

Thus, at least before the workshop, my going-in position is that Aspects are a very good implementation mechanism for items that are identified in Architectural Descriptions as constraints on the solution. However, I'm less clear about how Aspects themselves become 'first class' within the description itself.

---

<sup>2</sup> Where should the description of component interfaces go? In my factoring of concerns to viewpoints, I put these into the Data viewpoint. Rich Hilliard suggests these might fit better in the Behavior/Functional viewpoint. Either way, this raises the interesting question of inter-viewpoint consistency (items in the Data viewpoint passed between elements of the Behavior/Functional viewpoints need to be consistently represented in both viewpoints.) But it's not clear that's a topic relevant to the Aspects workshop.