End-to-End Architectures:
Experience to Date and Hard Problems Going Forward

Dr. Rick Schantz
and a host of talented colleagues, past and present

ITI Workshop
December 5, 2006
Background: Underlying Forces at Work

• Everything is a computer
• Everything is a networked computer
• Everything is potentially interdependent
• Things connect to the real physical world
• Increasing heterogeneity, distance and mobility

Leading to Current Trends and Directions

• Need for Integrated/Managed End-to-End Behavior
  – Evolving (String & Aggregate)
  – Multi-dimensional
• Multi-Layered Architectures, Network-centric Services & Systems of Systems
• Adaptive Designs Over Widely Varying and Changing Configurations
  – Static → Dynamic
• (More) Advanced Software Engineering (trying to keep pace)
Example String: Avionics Dynamic Mission Planning

A Net-meeting like mission replanning collaboration between C2 and fighter aircraft

QoS Techniques
- Tiling
- Compression
- Processor Resource Management
- Network Resource Management

QuO Components
- TAO components
- RT-ARM components
- TAO ORB
- Network Monitor
Example Aggregate: Multi-UAV Reconnaissance and Situation Monitoring

**End-to-End Objective-Driven QoS Management**

**Reconnaissance Mode**
- Maximize area monitored
- Sufficient resolution in delivered imagery to determine items of interest

**Situation Assessment**
- UAV observing item of interest provides high resolution imagery so that unfolding situation can be monitored, assessed, and acted upon

**After Action Assessment**
- UAV provides high resolution imagery until a human operator has determined that it is sufficient
- UAV over item of interest must continue to provide situation assessment imagery

The challenge is to program the dynamic control and adaptation to manage and enforce end-to-end QoS.

**Example DRE Application 2**

- Radio links are statically allocated
- Vehicle platforms are closed with statically scheduled tasks
- Ground based CPUs have variable dynamic load
- LAN/WAN links are shared with (a priori) unknown load

Heterogeneous, shared, and constrained resources

Multi-layer points of view: System-view, goal-view, application-string view, local resource view

Goal-defined requirements and tradeoffs (e.g., rate, image size, fidelity)

Changing modes, participants, and environmental conditions
Multi-Layer QoS Management Architecture

- **System Resource Manager (SRM)**
  - Knows mission goals and tradeoffs
  - Knows number and types of participants, roles and relative importance, and available shared resources
  - Produces policy defined for each participant

- **Local Resource Manager (LRM)**
  - Determines how to utilize allocated resources to meet mission goals
  - Configures and monitors QoS behaviors to enforce QoS policy

- **QoS behaviors**
  - Control and monitor individual resources or mechanisms, or adapt application behavior

---

Copyright 2006 BBN Technologies
End-to-end QoS management must

- Manage all the resources that can affect QoS, i.e., anything that could be a bottleneck at any time during the operation of the system (e.g., CPU, bandwidth, memory, power, sensors, …).
- Shape the data and processing to fit the available resources and the mission needs
  - What can be delivered/processed
  - What is important to deliver/process
- Includes capturing mission requirements, monitoring resource usage, controlling resource knobs, and runtime reallocation/adaptation

Control and Monitor Network Bandwidth
- Set DiffServ CodePoints (per ORB, component server, thread, stream, or message)
- Work with DSCP directly or with higher level bandwidth brokers
- Priority-based (DiffServ) or reservation-based (RSVP)

Control and Monitor CPU Processing
- CPU Reservation or CPU priority and scheduling
- Have versions that work with CPU broker, RT CORBA, RTARM

Shape and Monitor Data and Application Behavior
- Shape the data to fit the resources and the requirements
- Insert using components, objects, wrappers, aspect weaving, or intercepters
- Library that includes scaling, compression, fragmentation, tiling, pacing, cropping, format change

Coordinated QoS Management

System resource managers allocate available resources based on mission requirements, participants, roles, and priorities

Local resource managers decide how best to utilize the resource allocation to meet mission requirements

Dynamic QoS realized by
- Assembly of QoS components
- Paths through QoS components
- Parameterization of QoS components
- Adaptive algorithms in QoS components
Patterns of Component Composition

Hierarchical

Parallel

Sequential

Data Shaping QCs

Image Sender

Crop QC  Scale QC  Compress QC  Pacing QC  Network

DePacing QC  Decompress QC

Image Receiver/Display

Copyright 2006 BBN Technologies
Example of Multiple Properties: Architecting Survivability into Large Realtime Systems

Build an information management system that can survive sustained attacks from nation-state adversary and complete its mission.

Operate through attacks by using a layered defense-in-depth concept

- Accept some degradation
- Protect most valuable assets
- Move faster than the intruder

“Defense Enabling” Distributed Applications
The DPASA Project

Reliability requires architecting in multiple dimensions
Even more so, when the goal is to be resilient not only against errors, but also against attacks….

General principles for survivability
- Protect as best as possible
- Improve chances of detection
- Adapt to manage gaps
Looking Forward Again: Two Areas of Continued R&D

1. Predictable, Interoperable, Multi-Property Embedded Systems
   Lot’s of interacting pieces and continuous embedding, across platforms with realtime requirements over shared environments. We do not yet have the capability to routinely build predictable interoperable systems
   - Interoperability → sharing
   - Predictability → isolation and dedicated resources
   - Mandates new higher level abstractions for development and tools, tools, tools
   - Needs common, shared trust model across heterogeneous components

2. Certifiable Dynamic Architectures and Designs
   Many of the distributed, realtime, embedded environments we engage have certifiability requirements
   - Current approach is for static architectures and based on exhaustive testing
   - Interconnection drives dynamic behavior which breaks current approaches

   Elements of an Alternative Approach:
   - Utility measures can capture attributes of system performance and quality
     - Measure user-perceived value derived from control
     - Provide a quantitative measure for certification
   - Feedback control uses utility measurements/estimates to drive toward higher (increasing) utility
     - Allows system to dynamically respond to unforeseen situations