Smart Grid Communications: QoS Stovepipes or QoS Interoperability?

Dr. David E. Bakken, Washington State University
Dr. Richard E. Schantz, BBN Technologies
Mr. Richard D. Tucker, Tucker Engineering Associates
Big Picture for this Presentation

• Been re-thinking wide-area grid comms since 1999
  – Previously at BBN (fielded QuO middleware)
  – With NAE Fellow Anjan Bose (power)
  – Specialized pub-sub middleware “GridStat” since 2001
  – Joined by Carl Hauser in 2001
  – Very influential in emerging NASPInet

• Background (computer science)
  – I initiate, design, develop, measure new kinds of middleware
    with multiple Quality of Service (incl. security/privacy) properties
  – Lots of industry (BBN, Boeing, consult to Amazon.com etc)
  – But in 11th year as an academic

• Schantz a middleware pioneer, Tucker “Tucker Tables”
Outline of Presentation

• Context
• Functional Interoperability Issues
• Middleware and Functional Interoperability
• Non-Functional Interoperability Issues
• Middleware and Non-Functional Interoperability
Context of Interoperability [GWAC]

- Organizational
  - 8: Economic/Regulatory Policy
  - 7: Business Objectives
  - 6: Business Procedures
  - 5: Business Context
  - 4: Semantic Understanding
  - 3: Syntactic Interoperability
  - 2: Network Interoperability
  - 1: Basic Connectivity

- Informational

- Technical

- Cross-cutting Issues
  - Shared Meaning of Content
  - Resource Identification
  - Time Synch & Sequencing
  - Security & Privacy
  - Logging & Auditing
  - Transaction & State Mgt
  - System Preservation
  - Quality of Service
  - Discovery & Configuration
  - System Evolution & Scalability
Principle I09: An interoperability framework must be practical and achievable:

- Meets performance requirements
- Is reliable
- Is scalable
- Has sufficient breadth to meet the range of business needs

Principle I10: An interoperability strategy must accommodate the coexistence of and evolvement through several generations of IT standards and technologies that will reside at any point in time on the Grid.
Distance to Integrate [GWAC]

Party A

No standard exists, requires completely custom integration

Interfaces can be transformed and/or mapped

Interfaces use a common model

“Plug and Play” standard defined

QoS: Electricity industry is here (arguably here)

I am talking about here and moving towards here.
Take-Home Points

• In support of **Principle I09**, it is essential that syntactic interoperability involves middleware perspectives.

• In order to support interoperability across organizations and in support of the "future proofing" articulated in **Principle I10**, it is essential that APIs for Quality of Service (including security) should be expressed at a middleware layer, which maps down onto the lower-level mechanisms for providing a given property.

• In order to support multiple QoS properties (delay, rate, confidentiality, criticality/availability, ...), it is essential that APIs be expressed in middleware so that they can be co-managed.
Outline of Presentation

• Context
• Functional Interoperability Issues
• Middleware and Functional Interoperability
• Non-Functional Interoperability Issues
• Middleware and Non-Functional Interoperability
• **Stovepipe System**: a legacy system that is an assemblage of inter-related elements that are so tightly bound together that the individual elements cannot be differentiated, upgraded or refactored. The stovepipe system must be maintained until it can be entirely replaced by a new system

• Stovepipe system is the opposite of interoperability
  – Far too common in long-lived systems
  – Very expensive to maintain
  – “Smart Grid” cannot afford this (we can do better!)
Examples of Network Interoperability

- **OSI layers**
  - 2: LAN (ethernet)
  - 3: network (IP)
  - 4: transport (TCP, UDP, SCTP)
  - [GWAC08 mentions Layers 5 and 7 but we believe that this is at the higher “Syntactic Interoperability”.

- **Utility example**: ANSI C12.22 (distribution metering)

- **Functional Interoperability**: traditional interoperability via APIs, contracts, etc (“business logic”)

- **Non-Functional Interoperability**: end-to-end interoperability across behavioral issues (delay, security) that may span organizations, different underlying “QoS” and security technologies, etc.
Outline of Presentation

• Context
• Functional Interoperability Issues
  • Middleware and Functional Interoperability
• Non-Functional Interoperability Issues
  • Middleware and Non-Functional Interoperability
Middleware Defined

• **Middleware**: A layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system

• Middleware exists to help manage the complexity and heterogeneity inherent in distributed systems
  – 1980s: figuring out how to understand passed data structs

• Middleware provides higher-level building blocks (“abstractions”) for programmers than the OS provides
  – Can make code much more portable
  – Can make them much more productive
  – Can make the resulting code have fewer errors
  – Analogy — MW:sockets ≈ HOL:assembler
Middleware in Context

Host 1

Distributed Application

Middleware API

Middleware

Operating System API

OS Comm. Processing Storage

Host 2

Distributed Application

Middleware API

Middleware

Operating System API

OS Comm. Processing Storage

Network
Middleware typically encompasses OSI Layers above transport:
- 5: Session
- 6: Presentation
- 7: Application (OSI was created by network researchers!)

In our opinion, this is much of the “Syntactic Interoperability”: understanding passed data structs!

Middleware has been “best practices” in many industries for a long time (1980s)
- Alternative: hand-code Layers 5-6 with “app. protocols”
- Very hard to recreate “best practices” in house
- Note: middleware can be misused (beyond design point)
Middleware in Other Industries

- Middleware in wide use for decades in other industries (see paper for links); NASPI recent “aha”
- DoD has required in many areas for some time
  - US DOD DISA DISR (current DISR baseline version is 09-2; requires DoD PKI Cert to access)
  - US Navy NESI (see NESI-X Part 5 Developer Guidance Mid Tier)
  - US Navy Open Architecture Computing Environment (OACE)
  - US Navy PEO IWS Objective Architecture Software Design Document (Draft)
- ... there is good middleware & bad middleware ...

Outline of Presentation

• Context
• Functional Interoperability Issues
• Middleware and Functional Interoperability
• Non-Functional Interoperability Issues
• Middleware and Non-Functional Interoperability
Non-Functional Interoperability

- Functional interoperability deals with **what**
- Non-functional properties deal with **how**
  - How fast
  - How robust
  - How secure
  - How complete (searches etc)
  - ...

- Non-functional interoperability deals with composition of this how across organizations, underlying lower-level mechanisms, etc.
Non-Functional Properties

• Non-functional properties required for grid include:
  – End-to-end **latency** (as low as a few milliseconds)
  – **Rate** (from once a minute to 250 Hz)
  – Widely varying requirements for **Availability of Data**
  – **Confidentiality & Integrity**

• Implementation Reality: *you usually can’t have it all!*
  – Different properties must be traded off against others
  – Different mechanisms for a given property are appropriate for only some of the operating conditions an application may encounter (esp. a long-lived one)
  – Mechanisms can’t be combined in arbitrary ways

• A lot for an app. programmer to directly integrate in his/her program and expect it to work/evolve/interop.
APIs for Non-Functional Properties

• Make them as high-level as possible (i.e., middleware)
  – It is less error-prone
  – Very few application programmers are experts in low-level non-functional property mechanisms.
  – Different lower-level mechanisms are available in different configurations in different deployments.
  – The APIs of the lower-level mechanisms will change over time and perhaps with situation.
  – New lower-level mechanisms providing the same property or properties will become available over the lifetime of an application (which often can span many decades). Such new mechanisms will often be better than existing ones in one or more ways, including offering a higher level of a non-funct. property or usability across a wider range of IT conditions.
QoS Stovepipe System Defined

• **QoS Stovepipe System (QSS)**: a system of systems whose subsystems are locked into low-level mechanisms for QoS and security such that
  a) it cannot be deployed across many reasonable configurations, or
  b) combine some programs because they use different lower-level QoS mechanisms for the same property (e.g., latency) that cannot be composed, or
  c) be upgraded to “ride the technology” curve as better low-level QoS and security mechanisms.

• Smart grid would be “dumb” to allow this (IMO; YMMV)

• Middleware can help.....
Outline of Presentation

• Context
• Functional Interoperability Issues
• Middleware and Functional Interoperability
• Non-Functional Interoperability Issues
• Middleware and Non-Functional Interoperability
Mapping Towards Mechanisms

• Higher-level required QoS properties mapped onto lower-level properties and then onto available mechanisms
• Appl-level-1: freshness = max_period + max_latency
• Appl-level-2: rate to delivery a given update over given path of links (each with given link-level latencies)
• Network-level-1: bits/second over a given link
• Network-level-2: parameters of given network-level QoS mechanism

Must keep the app-level requirements as high as possible: (a) Will change
(b) Different mechanisms available in different configurations
Example from “delay” or BW mech’s

- All have some notion of delay, loss, thr’put, security
- “Guarantees” and semantics are very different
- Mechanisms themselves are very different
- NONE give per-message guarantees!

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Guarantees</th>
<th>Control Mechanism</th>
<th>Control Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>Strong</td>
<td>Reserve explicit circuit across network for entire connection</td>
<td>Runtime: connection Setup</td>
</tr>
<tr>
<td>INTSERV/RSVP</td>
<td>Soft (will inform program if violated)</td>
<td>Out-of-band channel following data path, setting up time slots or buffers etc.</td>
<td>Runtime: out of band, can be repeated</td>
</tr>
<tr>
<td>IPv6 Flow Label</td>
<td>Hints (not even soft), will likely happen in the aggregate over a long time</td>
<td>Byte in each IP packet tagged with a class; can be interpreted differently in different implementations</td>
<td>SLA purchase time</td>
</tr>
<tr>
<td>DIFFSERV</td>
<td>Hints (not even soft), will likely happen in the aggregate over a long time</td>
<td>Byte in each IP packet tagged with a class; can be interpreted differently in different implementations</td>
<td>SLA purchase time</td>
</tr>
<tr>
<td>MPLS (specialized form of DIFFSERV)</td>
<td>Aggregate economic guarantees over {user, location, protocol}</td>
<td>Internal to ISP: in ingress ISP adds header tag and used internally to queue by class and user, not packet</td>
<td>Traffic shaping time: out of band, periodically, aggregated across multiple customers.</td>
</tr>
</tbody>
</table>

Notes:
(a) only one with “strong” guarantees does not provide multicast
(b) how to compose difficult
(c) None will “take over the world” (or even half) -- interoperate
QoS-Enabled Middleware Needed!

• QoS-Enabled Middleware needed for above
  – Keep non-functional APIs high-level
  – Bridge end-to-end non-functional properties across different organizations and/or different underlying lower-level properties
Buy it 100% COTS?

- COTS middleware is a big part of any reasonable solution. 60%? 90%? But not 100%
- Appropriate for large parts of such “systems of systems”
- Not IMO designed for very low latencies, very high availabilities required for closed-loop protection and control in the grid (e.g., emerging NASPlinet)
  - No matter what any “sales sheep” says!
  - More R&D needed!
Conclusions

- Our bottom line:
  - GWAC “Syntactic Interoperability” should be largely considered middleware
  - APIs for non-functional properties should be as high-level as possible, not at the detailed mechanism API
  - QoS-enabled middleware can greatly help interoperability of non-functional properties.

- See our paper for LOTS more (active) links to related stuff...

- Questions?
Backup Slides Follow

• BACKUPS FOLLOW
Use this for brain surgery?

- 85 tools (Guinness World Record)
- $1000 (thinkgeek)
- Intended for very different environment than a humble scalpel (or even a simple Swiss Army Knife)
- How to verify right functionality?
Middleware and Heterogeneity

• Middleware’s programming building blocks mask heterogeneity
  – Makes programmer’s life much easier!!

• Kinds of heterogeneity masked by middleware (MW) frameworks
  – All MW masks heterogeneity in network technology
  – All MW masks heterogeneity in host CPU
  – Almost all MW masks heterogeneity in operating system (or family thereof)
    • Notable exception: Microsoft middleware (*de facto*; not *de jure* or *de fiat*)
  – Almost all MW masks heterogeneity in programming language
    • Notable exception: Java RMI
  – Some MW masks heterogeneity in vendor implementations
Middleware Benefit: Transparencies

• Middleware can provide useful transparencies:
  – Access Transparency
  – Location transparency
  – Concurrency transparency
  – Replication transparency
  – Failure transparency
  – Mobility transparency

• Masking heterogeneity and providing transparency makes programming distributed systems much easier to do!
Contact Info

• Bakken: bakken@wsu.edu; www.eecs.wsu.edu/~bakken; www.bakken-middleware.com

• Schantz: schantz@bbn.com; http://www.dist-systems.bbn.com/people/schantz/

• Tucker: richardaet@aol.com