Data-Centric Architecture for Space Systems

3rd Annual Workshop on Flight Software, Nov 5, 2009

The Real-Time Middleware Experts

Rajive Joshi, Ph.D.
Real-Time Innovations

Our goals are the same…
…but not the methods…

Agenda

- Flight software challenges
  - Issues & Trends
- Need for modular and adaptable systems architecture
  - Data Centric Architecture
- Open framework for integration
  - Standards based
- Examples
  - Demo

Flight Software Challenges

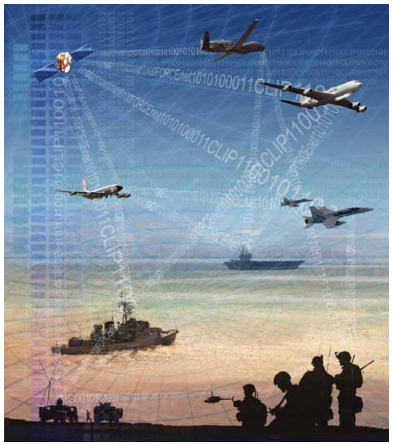
- Flight software complexity is exploding!
  - NASA Study, Dan Dvorak et al, 2nd Annual Workshop on Flight Software
  - Exponential growth in code size ~ 10x every 10 years
    - Size x Speed keeping up with Moore's law

- Flight software requirements make it harder!
  - predictable
  - repeatable
  - deterministic
  - reliable
  - reactive to anticipated events
  - robust to unforeseen events
  - maintainable remotely

Flight Software Challenges

- Trends
  - Advanced Hardware
    - multi-core cpus, network fabrics
  - Smaller physical footprint
  - Increasing demands on the software capabilities
    - "…widening gap between ambitions and achievements in software engineering" (NATO Report, 1968)
    - Same angst about software complexity in 2008 as in 1968

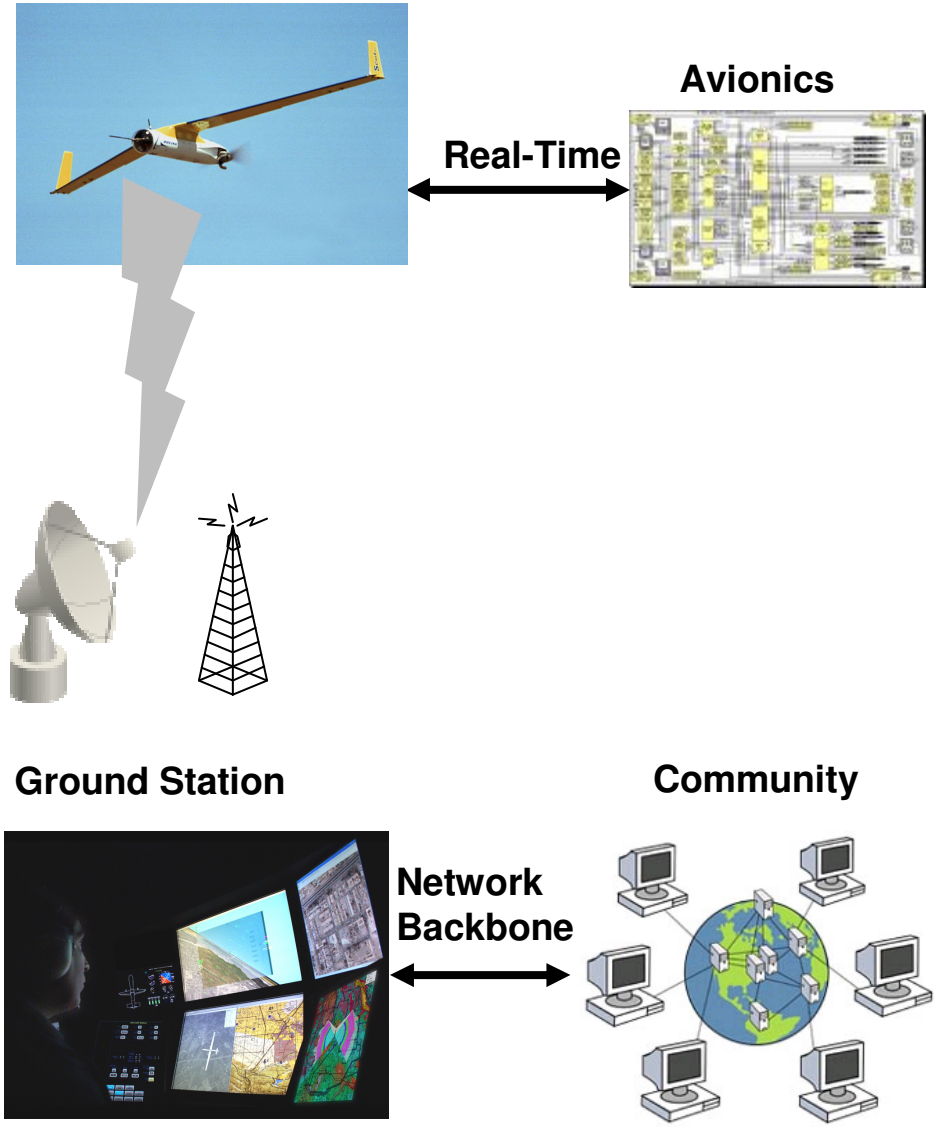
- New Requirements
  - Reconfigurable
    - variety of missions, payloads, platforms, ground systems
  - Major sub-systems designed independently in parallel
    - evolution of interfaces
    - late adaptation

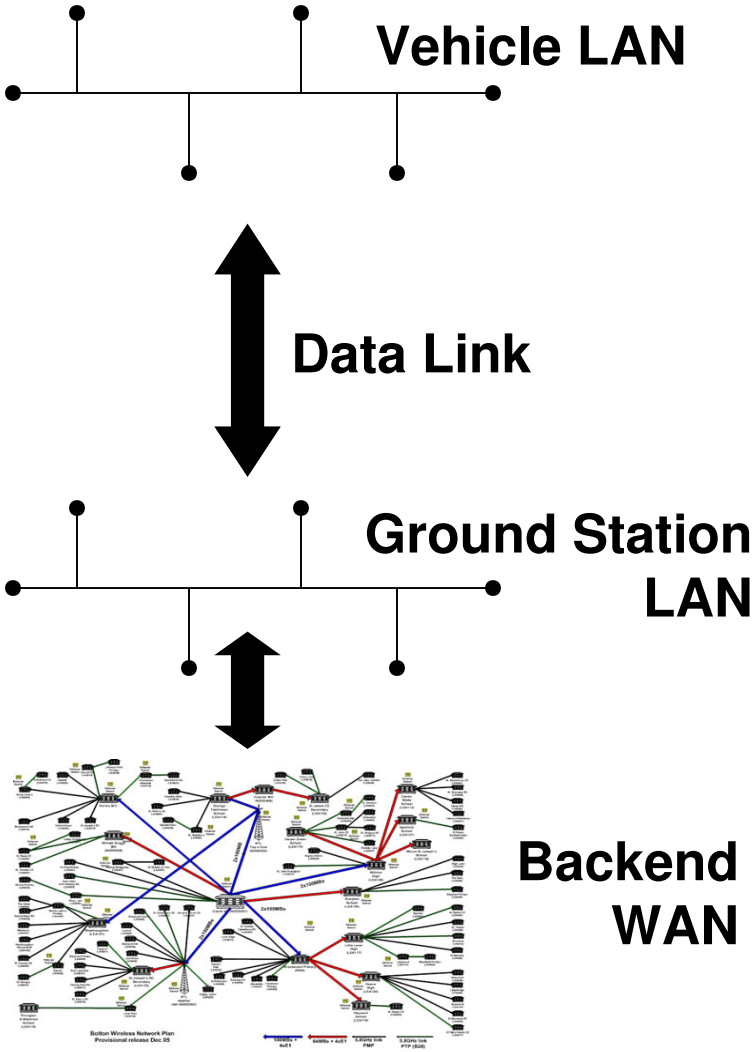


Goals

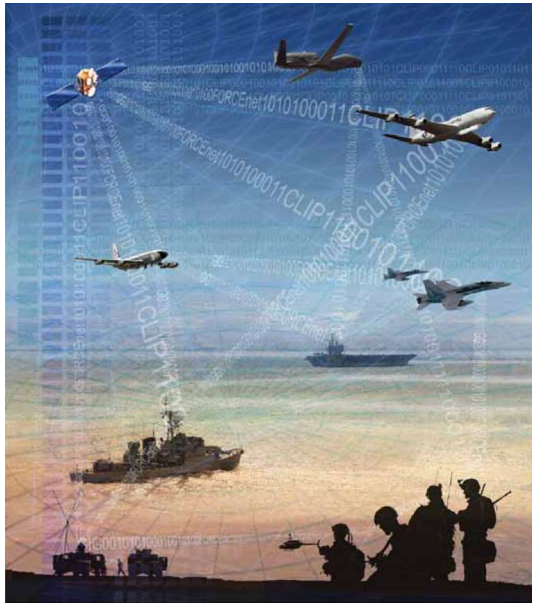
- Reduce software complexity (incidental)
  - simpler infrastructure
- Manage software development cost
  - portable “platform independent” application code
- Reduce integration cost & risk
  - loosely coupled components

More Efficient Communication Infrastructure Utilization





Baseline Capabilities for Flight Software Communication Platform



- Open standards based
  - Commonality and interoperability
- True peer-to-peer
  - No single point of failure
- Portable to any communication media
  - RF, optical links, high-speed interconnects, enterprise networks
- Available for heterogeneous environments
  - Embedded, low-power, small foot-print
  - RTOS (VxWorks, Integrity), MILS, ARINC653
  - Mainstream OS's (Windows, Linux) and CPU's (intel x86)

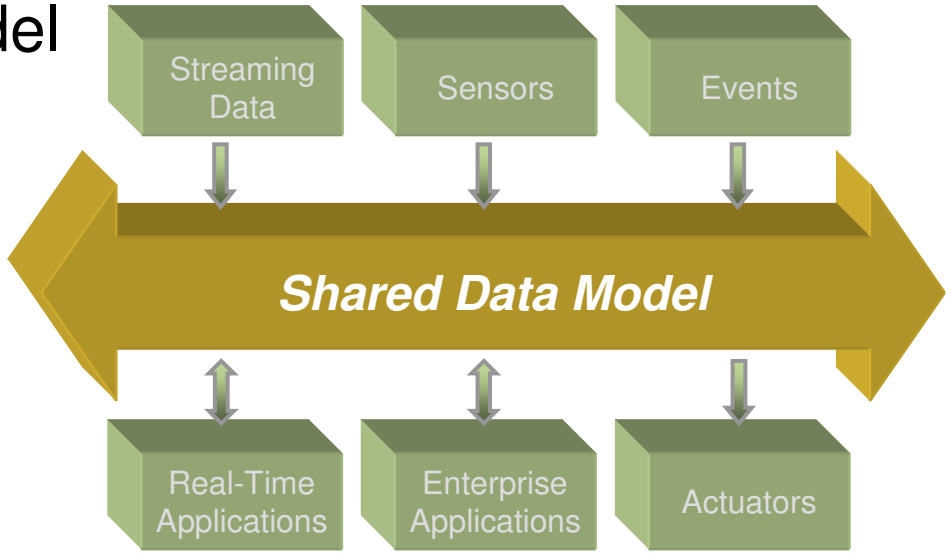
Communications “Matrix of Pain”

- Multiple traffic types:
  - Sensor data streams
  - Command and control data
  - Status, intelligence, mission, supervisory data
- Different traffic requirements for each type:
  - Response time, priority, reliability, volume
  - Stealth operations
- Challenging communications channel:
  - Large latency, low throughput
  - Lossy
  - Varying availability (Disconnected, Intermittent)
  - Asymmetric bandwidth (downlink vs uplink)

Modular And Adaptable Systems Architecture

"Data Centric"

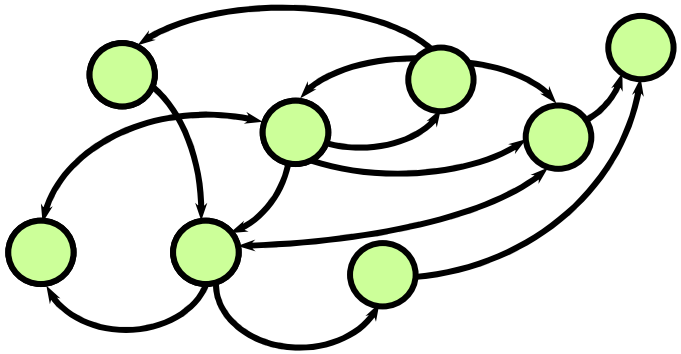
- Build down from data model
  - "Data-Centric"



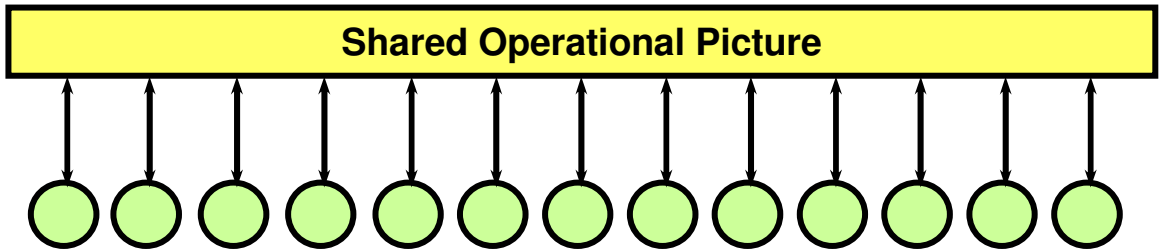
- Maximize decoupling between subsystems
  - Reconfiguration
  - Upgrades

Data Model is exposed as a "first-class" citizen

NOT THIS:
(connection-oriented)



BUT THIS:



= System Components

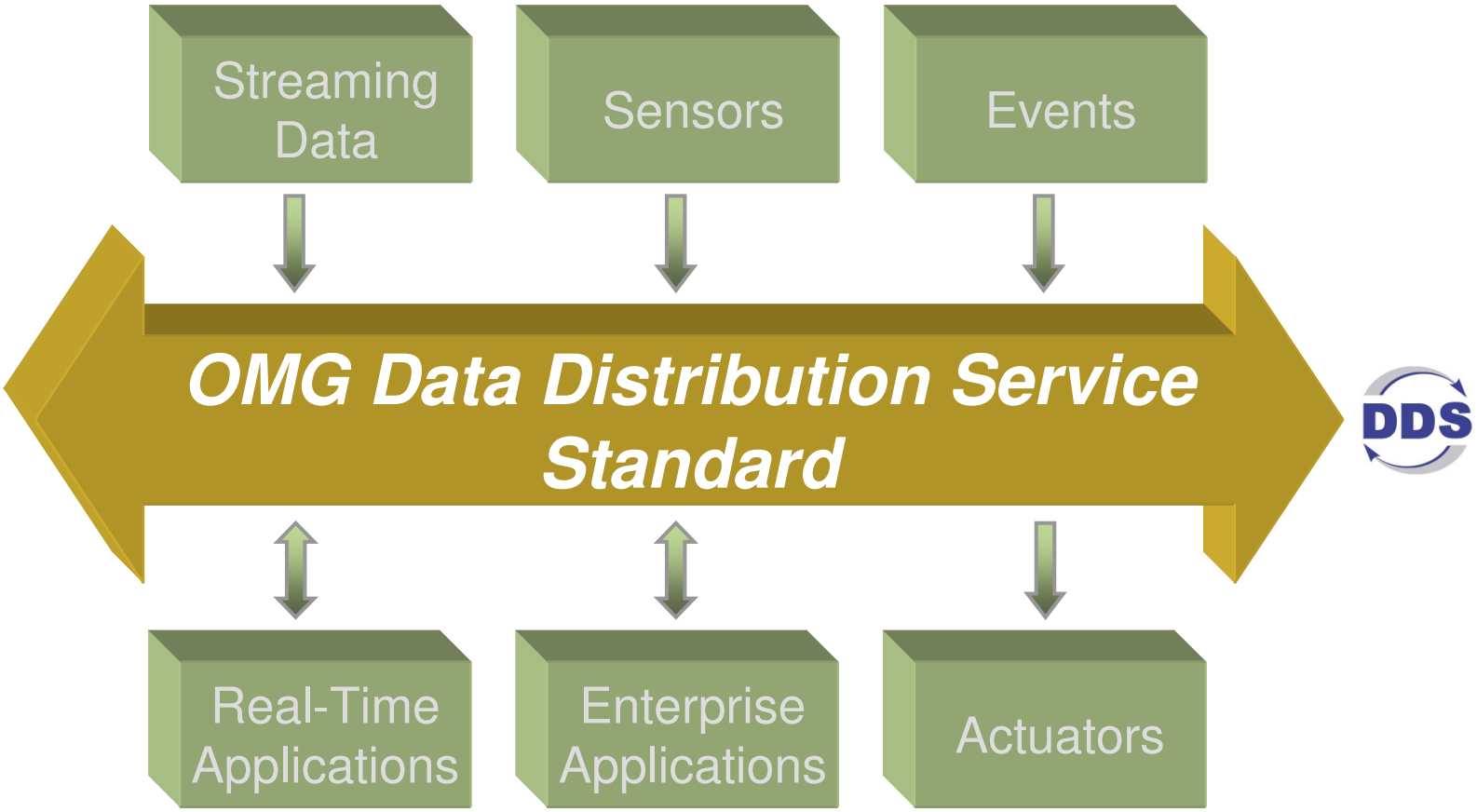
Use of an Information –Oriented design for Pub-Sub applications avoids creating stovepipe systems

Source: Raytheon Keynote Presentation September 2006 at DDS Information Day, Anaheim , CA

Communications Infrastructure Utilization

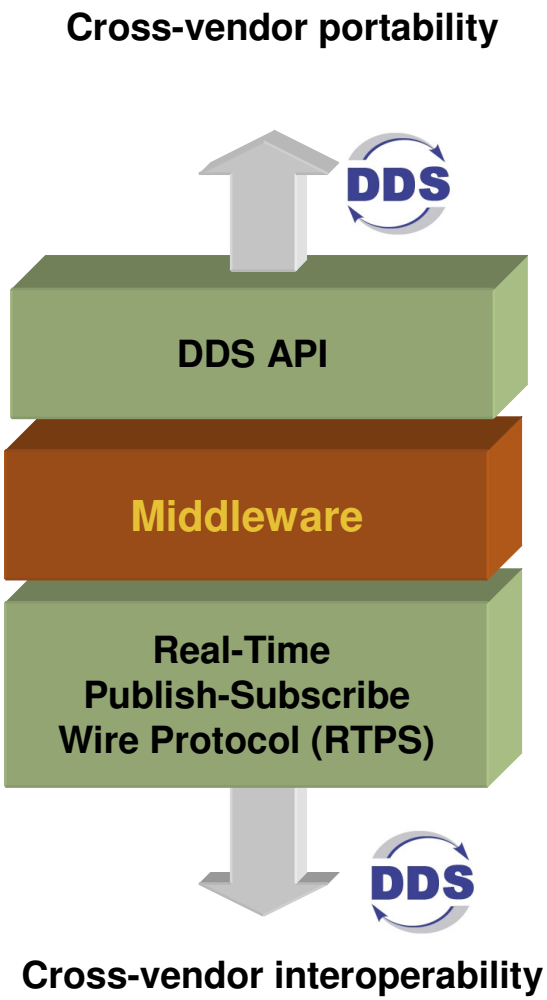
- Interfaces around the data-model
  - data-oriented
  - stateless
- Quality-of-Service between producer & consumer interfaces
  - define how/when data gets delivered
- Decouple end-points from communications infrastructure
  - location transparency
- Dynamically bind
  - N-to-N connections
- Example:
  - data reliability provided versus requested
  - caching

Open Framework for Integration



Open Architecture

- Vendor independent
  - API for portability
  - Wire protocol (RTPS) for interoperability
- Multiple vendors
  - 7 of API
  - 4 support RTPS
- Heterogeneous
  - C, C++, Java, .NET (C#, C++/CLI)
  - Windows, Linux, Unix, embedded, real-time, MILS
- Loosely coupled



Program Adoption

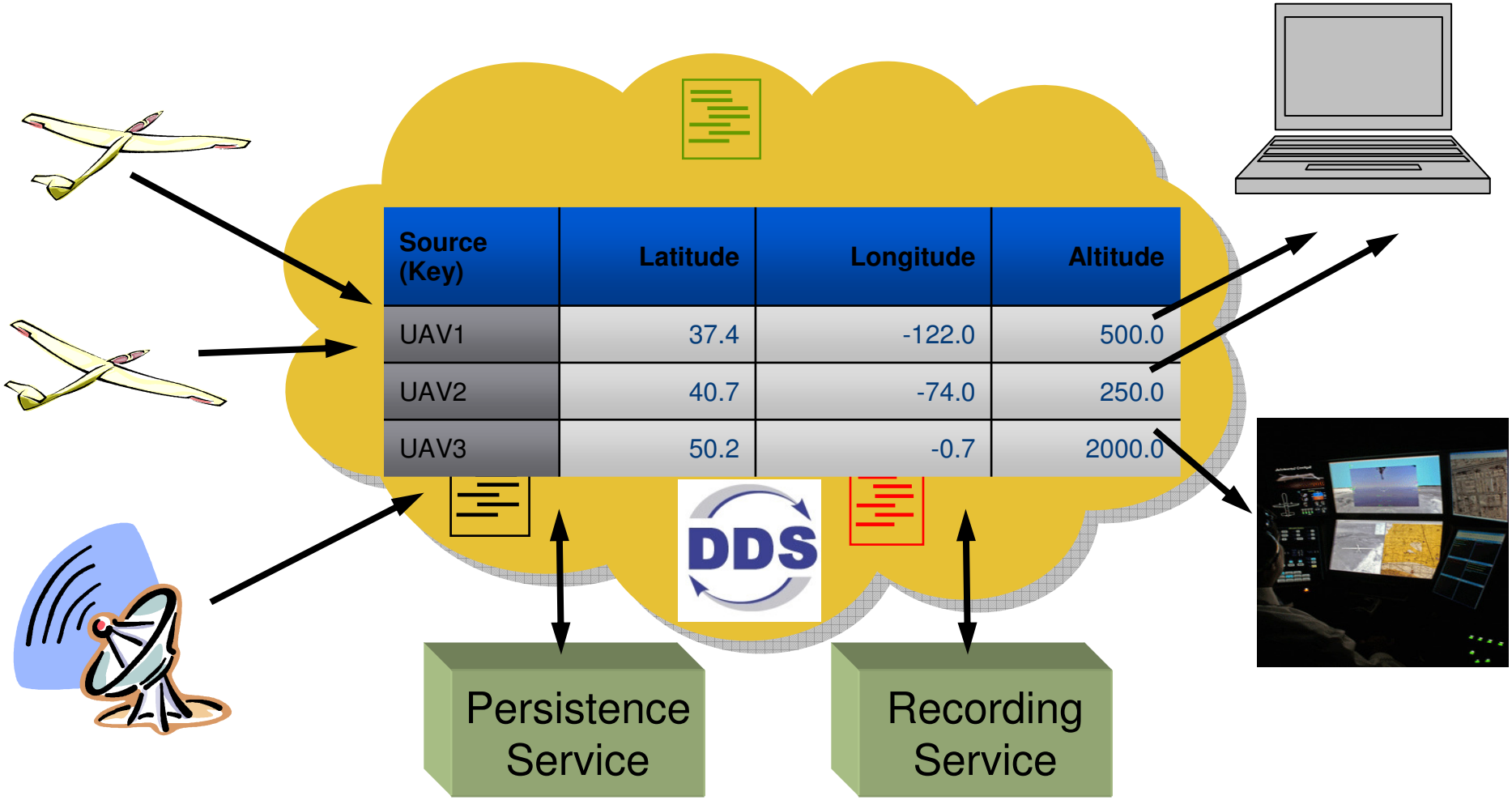
- DISA: DISR mandated
- Navy: Open Architecture, FORCEnet
- Air Force, Navy and DISA: NESI
- Army: FCS / SoSCOE
- Air traffic control for southern Europe

…plus over 300 individual projects

Data-Centric Publish-Subscribe Model

example

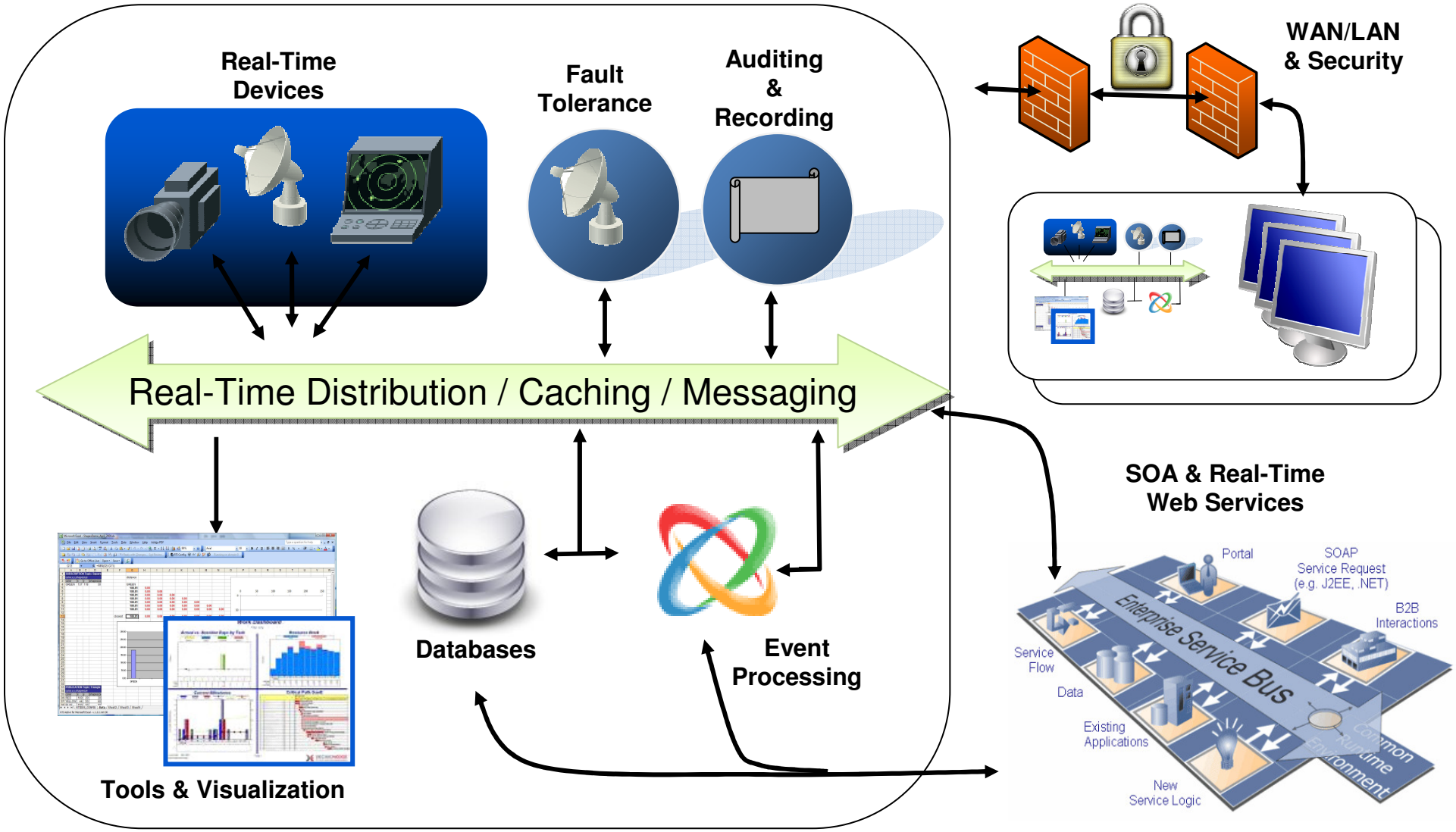
Essentially a virtual, decentralized global data space



Source (Key)	Latitude	Longitude	Altitude
UAV1	37.4	-122.0	500.0
UAV2	40.7	-74.0	250.0
UAV3	50.2	-0.7	2000.0

DDS: The Next Generation Integration Platform

System of Systems (SoS)



DDS Application Examples

Aegis Weapon System

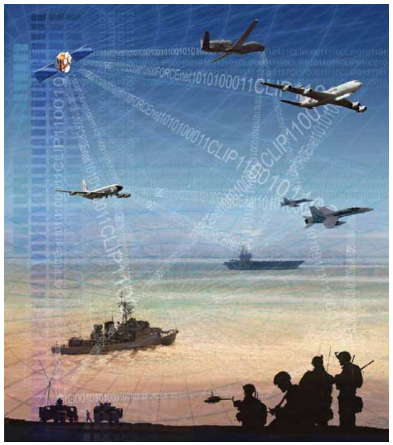
Lockheed Martin

Radar, weapons, displays, C2

B-1B Bomber

Boeing

C2, communications, weapons



Common Link Integration Processing (CLIP)

Northrop Grumman

Standards-compliant interface to legacy and new tactical data links

Air Force, Navy, B-1B and B-52

ScanEagle UAV

Boeing

Sensors, ground station

Advanced Cockpit Ground Control Station

Predator and SkyWarrior UAS

General Atomics

Telemetry data, multiple workstations

RoboScout

Base10

Internal data bus and link to communications center

Outcome

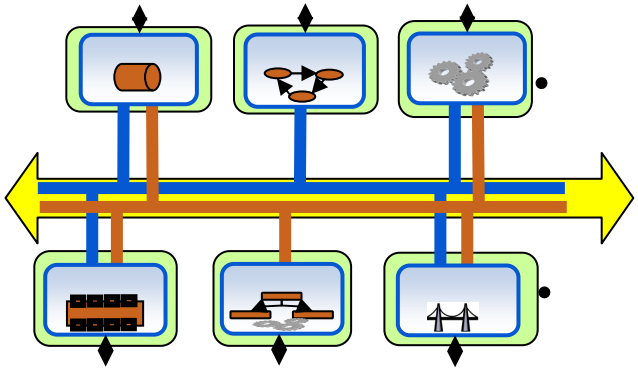
- Reduce software complexity (incidental)
  - leverage intelligence in the middleware
  - ✓ simplified infrastructure
- Manage software development cost
  - push the platform specifics in the middleware
  - ✓ portable “platform independent” application code
- Reduce integration cost & risk
  - use stateless data interfaces
  - ✓ loosely coupled components

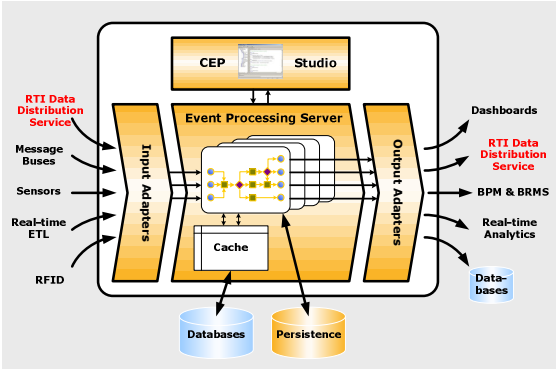
Next Steps

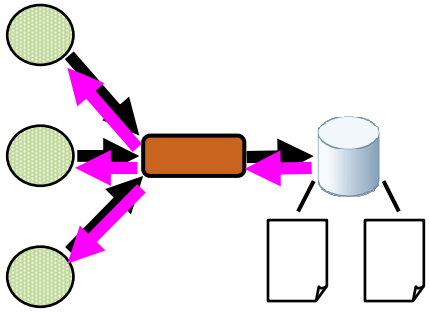
Experiment with Data-Centric Architecture

1. Pick a data-flow
2. Identify producers and consumers
   - Who? How many?
3. Identify data model
4. Identify quality of service (QoS)
   - Is an acknowledgement required?
   - Can there be late joiners?
   - Does the information expire?
   - Does it need to be cached? How much?
   - *…dozens more to consider…*
5. Formalize the **data-oriented interfaces**

Component-based System Architecture







Thank You!

- Contact
  - Rajive Joshi
  - rajive@rti.com
  - 408-250-7309
- Downloads
  - www.rti.com
- Demos
  - Visit the RTI booth (in the Cahill Center Library)