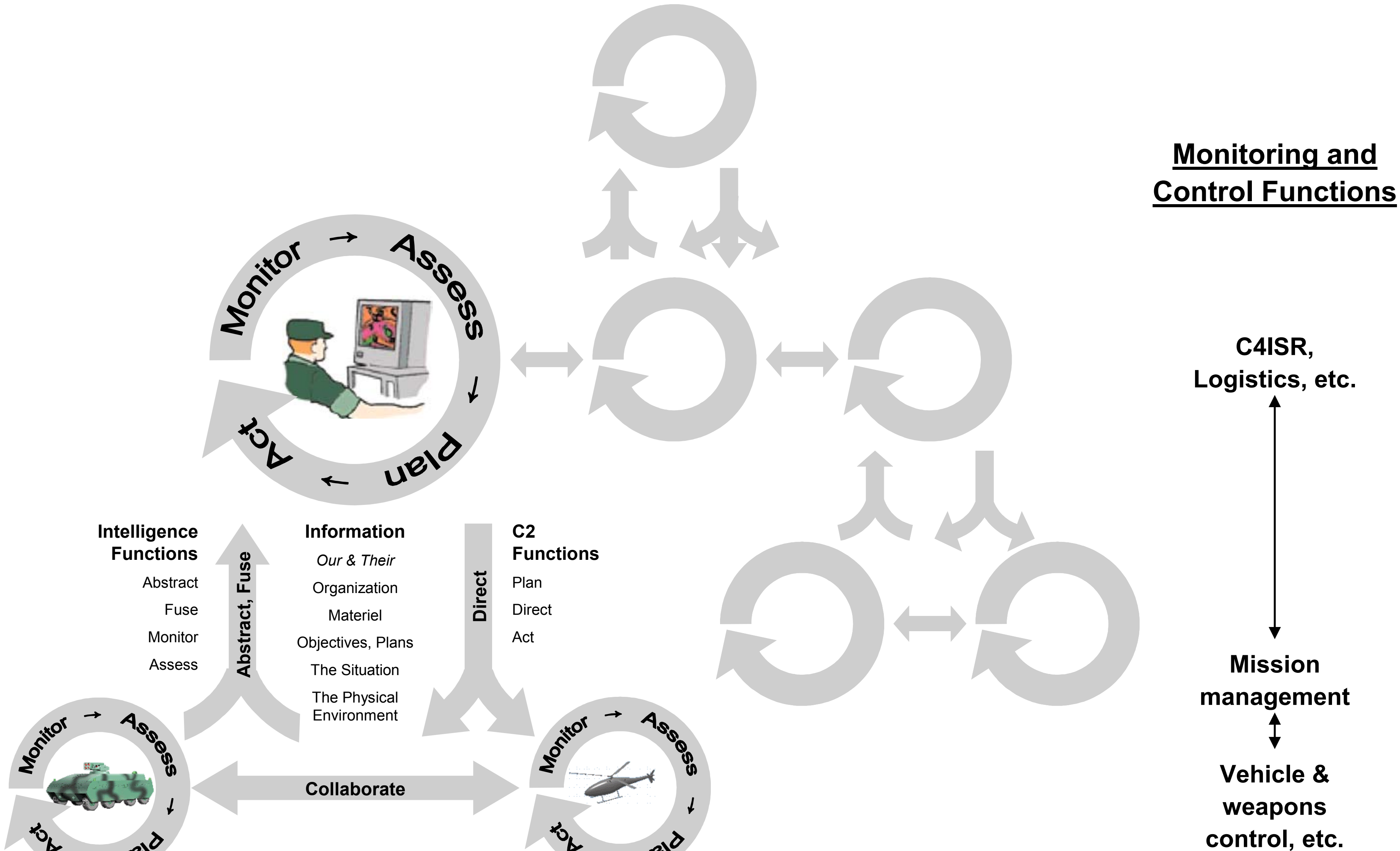
Honeywell

Computing and Communications Infrastructure for Network-Centric Warfare: Exploiting COTS, Assuring Performance

Dr. James P. Richardson
Mr. Lee Graba
Mr. Mukul Agrawal
Honeywell International, Inc.
{james.p.richardson,lee.graba,mukul.agrawal}@honeywell.com

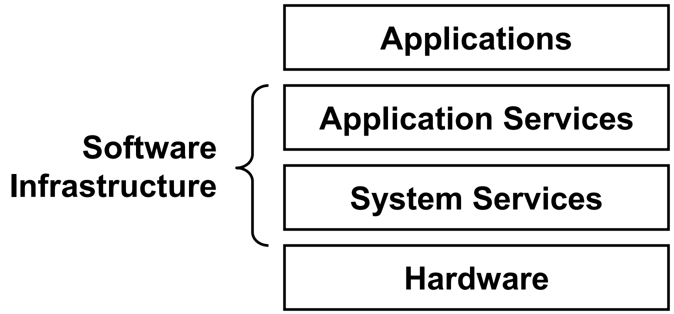
Command and Control Research and Technology Symposium
June 15-17, 2004
San Diego, CA

NCW as a Hierarchical Control System



Honeywell

Our Focus: Software Infrastructure at the Tactical Level



- Software infrastructure: application and system services that provide an execution environment for software applications
- Tactical level
  - Where the information world meets the physical world
  - Where information-rich applications meet vehicle and weapons control
  - Where communications bandwidth, power, and other resources are at a premium and under enemy attack

Challenges

- Supporting applications with diverse requirements
- Integrating these applications into a coherent system-of-systems
- Incorporating COTS IT components to reduce time to fielding and lifecycle costs

Diverse Application Requirements

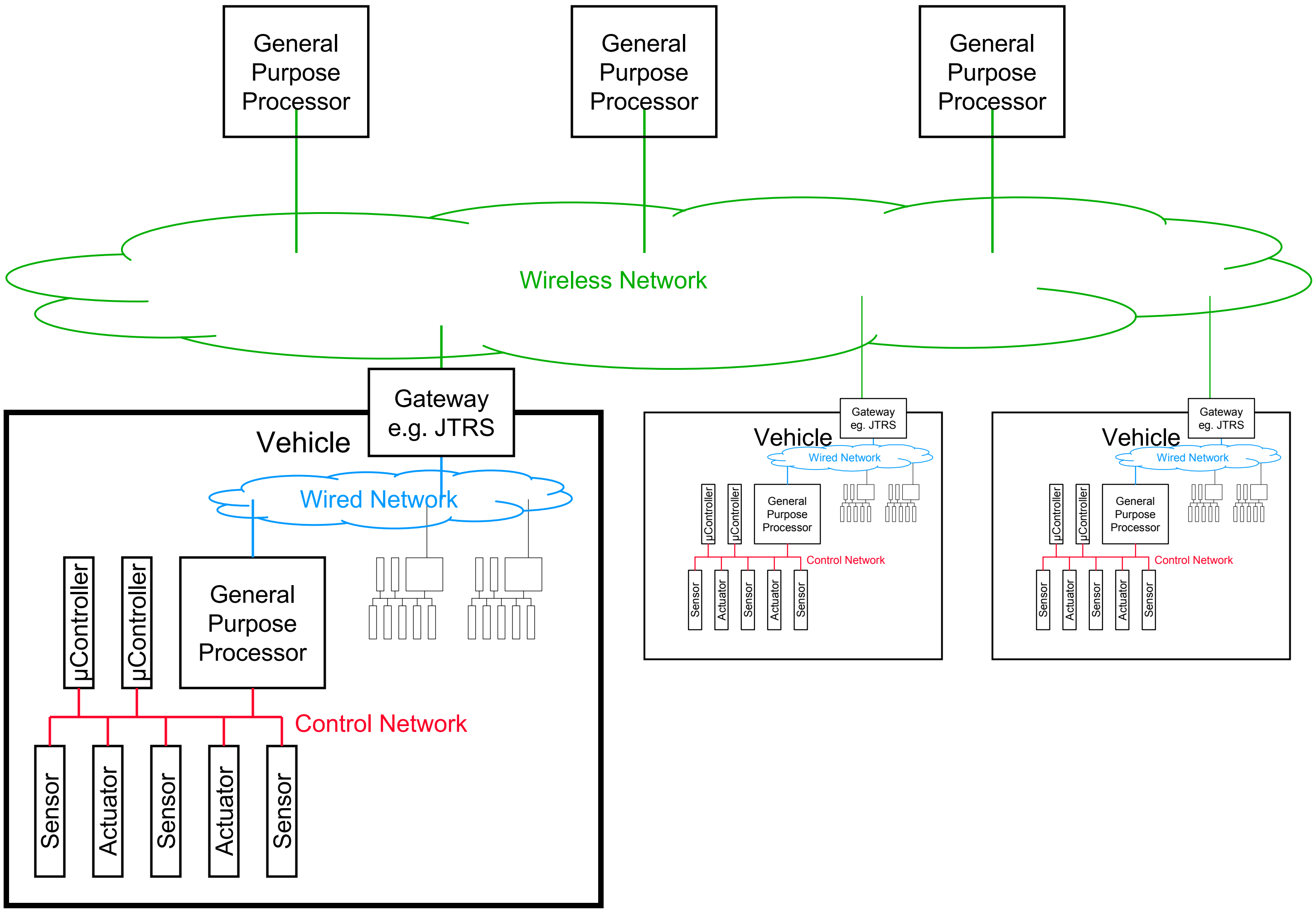
Applications requirements vary widely along several dimensions, yet must integrate or co-exist in the system-of-systems

Dimension	Definition	Range
Span	Scope of monitoring and control function	Vehicle subsystem Vehicle Multi-vehicle
Dependability	Level of assurance that functional and performance requirements are met	Safety critical Mission critical Other
Timing	Cycle/response time requirements	Millisec/sec/minute Hard-/soft-/non-RT
Workload	CPU cycles memory size IO & communications bandwidth	…
Workload variability	Variation in resource requirements over time	None/small/large
Security	Authentication, integrity, confidentiality, etc.	…

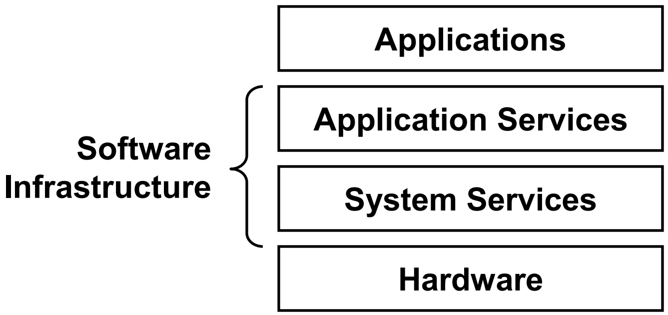
Outline

- Architectural assumptions
- System integration and resource management
- Information management
- Summary

Assumed Hardware and Network Architecture

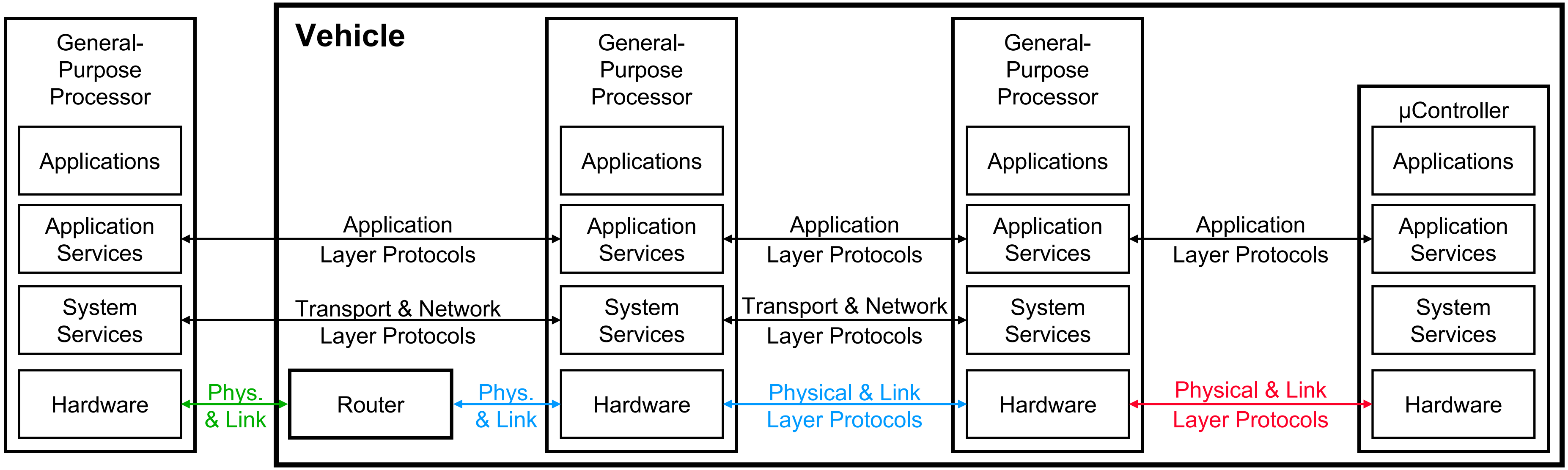


Top-Level Software Decomposition



- Applications
  - Implement user-visible components of system functionality
- Application services
  - Used by multiple applications
  - Are aware of application-specific algorithms and/or data structures
  - Examples
    - Naming services
    - ORB services
    - *Information management*
    - Workflow management
- System services
  - Manage computing and communications resources: CPU, memory, communications bandwidth, storage, I/O devices, etc.
  - Are insensitive to application-specific algorithms and/or data structures
  - Examples
    - OS services (POSIX)
    - Transport and lower-level communications services (TCP, UDP, IP, IEEE 802.x, etc.)
    - Time service

Inter- and Intra-Vehicle Protocol Stacks



Outline

- Architectural assumptions
- System integration and resource management
- Information management
- Summary

Key Requirements

- Assure system performance when applications share computing and communications resources
  - Essential for safety- and mission-critical applications
- Support diverse application requirements in the same system-of-systems
  - Span, dependability, timing, workload, workload variability, etc.
- Execute COTS applications and application services unmodified (*out of the box*)
  - Windows, POSIX, Java, etc.
- Adapt to changing mission goals, system workload, and resource availability

Alternatives for Assuring Performance

- "Tweak and test" system each time an application is added
  - Adjust execution priorities and cycle times until system works
  - Time consuming and brittle
- Explicitly manage computing and communications resources
  - Prevent an application's faulty, malicious, or gluttonous behavior from usurping other applications' resources
  - Resource management includes:
    - Allocating resources to applications
    - Enforcing allocations
      - Space- and time-partitioned operating system
      - Time-partitioned network
  - Static (design time) or dynamic (run time)

Static vs. Dynamic Resource Management

Static Resource Management

- Resources are allocated before the system is fielded
- Needed for safety- and mission-critical applications
- Appropriate for lower-level loops: vehicle/weapon control

Dynamic Resource Management

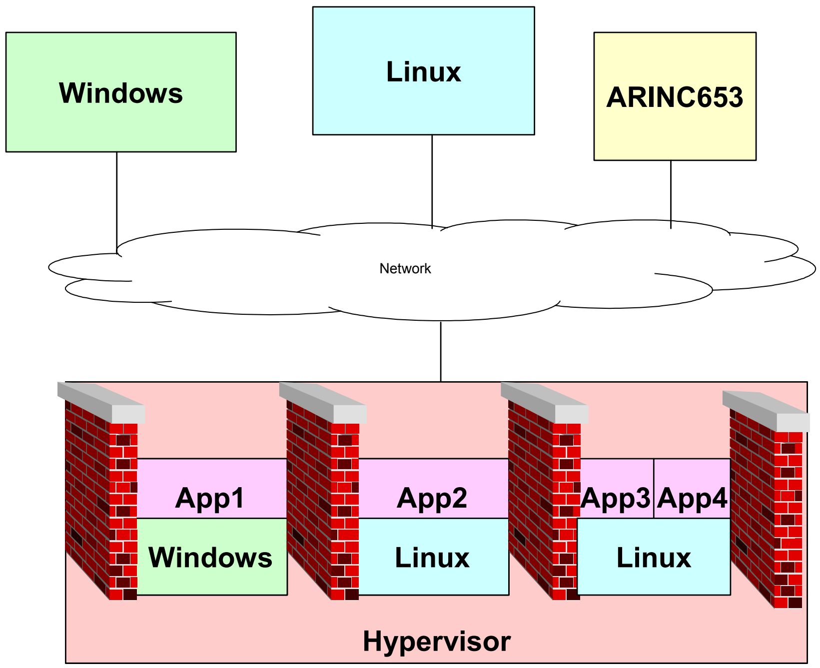
- Resources are reallocated when application workload, resources, or priorities change
- Needed when application workload and/or resources vary widely and efficient resource utilization is needed
- Applicable for more dynamic applications, e.g. mission planning

Operating System Alternatives

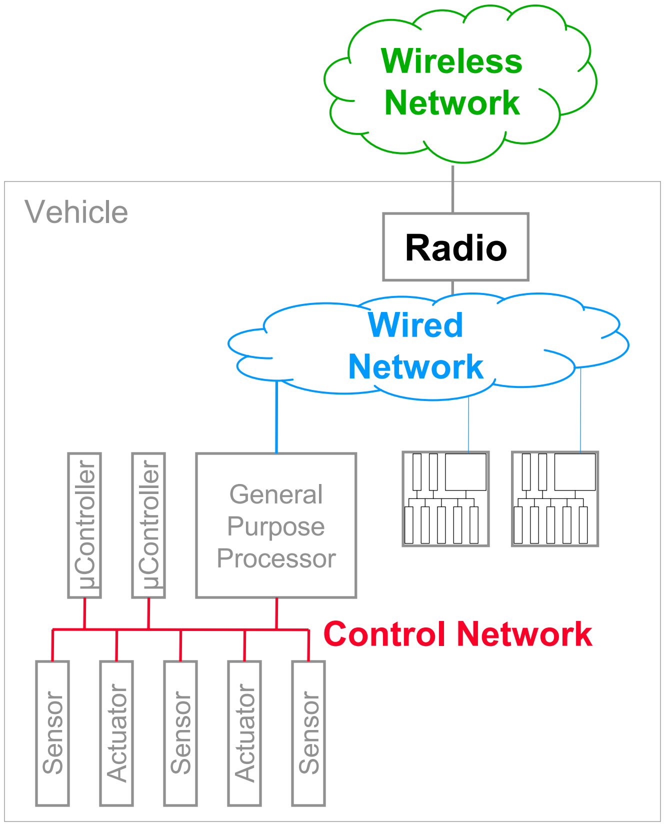
- A single commercial OS
  - *One size doesn’t fit all*: No single operating environment can support this range of requirements
    - Hard real-time vehicle control vs. mission planning
    - POSIX vs. Win32 API vs. ARINC 653
- A single common operating environment specific to military systems
  - *One size doesn’t fit all*
  - Can’t execute COTS applications & application services out of the box
- Conclusions
  - Multiple operating systems required
  - Broadly-accepted APIs are essential
  - Interoperability is essential

Time-Space Partitioned Operating Systems

- Static partitioning—for lower-level, safety/mission-critical functions
  - ARINC 653 standard
    - Used for commercial avionics
    - Several COTS implementations
      - LynxOS-178
      - Green Hills Integrity-178B
      - VxWorks AE653
    - Limited support for traditional OS services
  - 653+POSIX hybrids
- Dynamic partitioning—for higher-level monitoring and control functions
  - Linux variants e.g. TimeSys
- Hypervisor approach—multiple operating systems sharing hardware



Communications Services



- Again, one size doesn't fit all
- Control network for hard RT functions
  - Time-partitioned MIL-STD-1553B, ARINC659, Time-Triggered Protocol (TTP)
  - Assured performance through static allocation
- Wired intra-vehicle network for soft- and non-RT functions
  - IP over (redundant) Ethernet
  - High, constant available bandwidth
  - Bandwidth allocation possible—e.g. AFDX/ARINC 664
- Wireless inter-vehicle network
  - Trend is to IPv6 over JTRS
  - Lower, variable end-to-end bandwidth and availability
  - Performance can't be assured in battlefield situations
  - Time partitioning is a major research challenge

Dynamic Distributed Multi-Resource Management

- Especially important for soft-RT, distributed applications
- A continuing research challenge
- Requirements
  - Allocate multiple computing and communications resources to applications
  - Adapt to changing mission goals, system workload, and resource availability
- Recommended approach
  - Applications specify resource needs in terms of QoS ranges
  - Policy specification (tied to mission plan) defines relative importance of competing applications
  - Resource management component
    - Allocates resources within QoS ranges
    - Adjusts allocations as workload, resource availability, policy change

Outline

- Architectural assumptions
- System integration and resource management
- Information management
- Summary

Information Management Defined

- Run-time information management functions
  - Storage
  - Retrieval
  - Search
  - Dissemination
- Related design-time information modeling functions

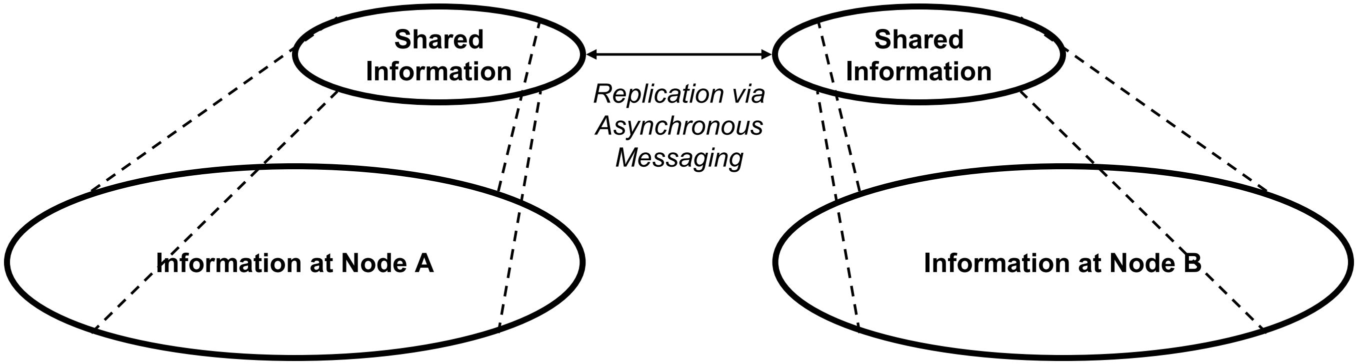
An essential application service for network-centric monitoring and control functions

Key Requirements

Interoperability	Common format and interpretation of shared information
Multiple implementations	Can't require common software base in a huge system-of-systems
Multiple types of information	Persistent structured Persistent documents, images, etc. Low-volume, time-critical, volatile
Search and access	Find relevant databases and documents Push and pull
Profiles	Define information needs
Effective use of available bandwidth	Send only what's needed Avoid duplicate transmission Send incremental updates
Integration with system resource management	Adapt to changing information needs, resource availability, mission goals
Information flow awareness	Monitor information delivery effectiveness

Information Management Approach

- Make maximum use of existing standards
  - SQL, X.500/LDAP, FTP, etc.
- View information dissemination as selective replication
  - *State-oriented* vs. *message-oriented* approach
  - Measure Information Quality of Service in terms of replica *freshness* and *accuracy*
- Integrate with distributed resource management
  - Freshness and accuracy requirements translate to bandwidth and delay
  - Adapt replication to available communications resources



Summary

- Extremely diverse application requirements force a mixture of operating systems and network protocols
  - One size does not fit all
- Broadly-accepted APIs and protocols are essential
- Manage system resources explicitly
  - Distributed, multi-resource management is a research challenge