Submitted to:
2004 Command and Control Research and Technology Symposium
The Power of Information Age Concepts and Technologies
Topic: C4ISR/C2 Architecture

Computing and communications infrastructure for network-centric warfare: Exploiting COTS, assuring performance

James P. Richardson, Lee Graba, Mukul Agrawal

Honeywell International
3660 Technology Drive
Minneapolis, MN 55418

(612) 951-7746 (voice), -7438 (fax)

james.p.richardson@honeywell.com

lee.graba@honeywell.com

mukul.agrawal@honeywell.com

Abstract

In network centric warfare (NCW), the effectiveness of warfighters and their platforms is enhanced by rapid and effective information flow. This requires a robust and flexible computing and communications software infrastructure, and a degree of system integration beyond what has ever been achieved. The design of this infrastructure is a tremendous challenge for a number of reasons. We argue that an appropriately architected software infrastructure can employ COTS software to realize much of the required functionality, while having the necessary degree of performance assurance required for military missions.

Perhaps the biggest challenge is system resource management—allocation of computing and communications resources to COTS and custom software alike so that performance requirements can be met. We describe an approach to system resource management appropriate to the NCW environment.

1 Introduction

Network centric warfare has been defined as:

> …an information superiority-enabled concept of operations that generates increased combat power by networking sensors, decision makers, and shooters to achieve shared awareness, increased speed of command, higher tempo of operations, greater lethality, increased survivability, and a degree of self-synchronization. In essence, NCW translates information superiority into combat power by effectively linking knowledgeable entities in the battlespace.[1]

We view an NCW-enabled military force as a hierarchical, enterprise-scale monitoring and control system utilizing personnel, equipment, and information technology to achieve the commander's objectives. (See Figure 1) At the bottom of the hierarchy are individual autonomous and manned platforms, the humans who operate them, and dismounted soldiers. The lowest-level loops perform safety-critical control functions such as vehicle control and weapons control. Slightly higher up are supervisory control functions such as mission management, which typically have a human in the loop but with unmanned platforms will be increasingly automated. Moving up the hierarchy, classic control functions give way to C^4ISR, logistics, and related information-rich monitoring and control functions that nonetheless are intended to achieve desired objectives within specific timelines in the face of adversarial action, a possibly hostile environment, and uncertainty about the current situation.

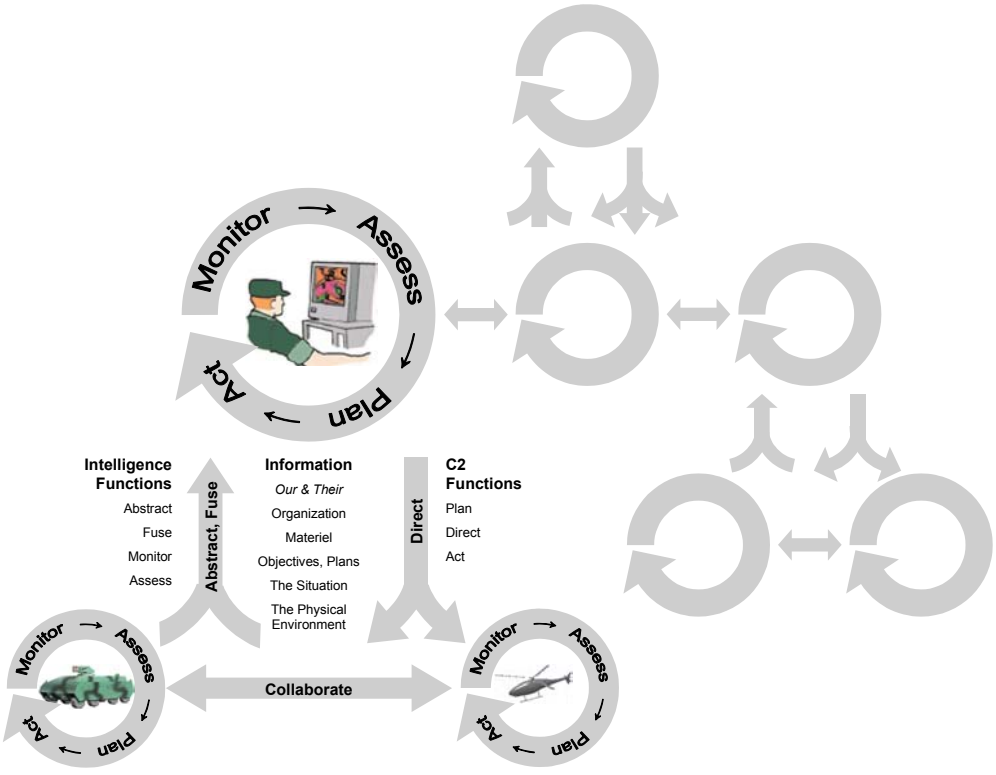

Figure 1. NCW-enabled military force as a hierarchical, enterprise-scale control system

Each loop involves monitoring the current situation, assessing it with respect to objectives, developing a plan to progress toward the objectives, and executing that plan. Executing a plan may involve providing objectives (or commands) for lower-level loops and monitoring their execution. These lower loops may collaborate ("self-synchronize") without direct intervention by the supervisory loop. For the lowest-level loops, planning (better known as control synthesis) is typically done at design-time, and the resulting control strategy is executed at run-time.

A rich variety of information flows up, down, and across the control hierarchy. In general,

- A warfighter's information needs depend on his functional role and place in the hierarchy.
- Information at higher levels of the hierarchy has broader scope but less detail. Information is abstracted and fused as it moves up the hierarchy.
- The information seen by different warfighters, though different, must be consistent.

The DoD calls this "information world" the *Global Information Grid* (GIG), and has laid out an excellent set of requirements for it [2]. The information world supports operations in the physical world, where (most of) the battle takes place. The information world meets the physical world at various sensors and weapon systems.

Designing a computing and communications infrastructure to support this hierarchical monitoring and control system is a tremendous challenge for a number of reasons, including:

- The sheer size of the system (or really, system of systems). The Army's Future Combat System, large as it is, is just one component that must interoperate effectively with other components in the system, not just within the Army but also the other services. Interoperability has been defined as the ability of two or more systems or components to exchange information and to use the information that has been exchanged.[6]
- The need to evolve incrementally from the current computing and communications infrastructure.
- The large number of US and allied military organizations whose procurements must be coordinated to achieve the necessary interoperability.
- The unique requirements that the military mission and environment places on the computing and communications infrastructure.

In this paper, we focus on the design of the computing and communications software infrastructure at the tactical level, where the information world meets the physical world of the battlefield; where C^4ISR, logistics, and similar information-rich functions meet vehicle and weapons control; and where communications bandwidth, power, and other resources are at a premium and under enemy attack. We address the following issues:

- Certain forms of system or component failure can be life-threatening. How can a system this large and complex be integrated with the required level of assurance that functionality and performance guarantees will be met? In particular, different parts of the system have different assurance requirements, but still need to be integrated.

- Mission goals, system workload, and system resource availability all vary over time. How can scarce system resources be put to their best use in support of mission goals? In particular, the system must adapt to changes in resource availability due to unintentional system failures, hostile actions, and compromised components.
- Information superiority is key to network-centric warfare. How can each warfighter get the information he or she needs in this highly dynamic environment?
- System architects must resist the urge to start with a clean sheet of paper. ("*Our requirements are different!*") Existing commercial off-the-shelf (COTS) components and information technology standards should be exploited wherever possible to reduce time to fielding and system lifecycle costs, and to enhance interoperability. COTS components must be integrated *without changing any source or even binary code*. Which NCW requirements can be met by existing COTS components and standards, and where will research and development be needed?

We identify key requirements related to system integration, system resource management, and information management, and propose specific design approaches that exploit COTS technology wherever possible. For example, COTS technology from the avionics industry (such as ARINC 653) can be applied for highly safety-critical components. At the network level, protocols such as RSVP provide mechanisms to define resource requirements.

Perhaps the biggest gap where COTS technology does not provide a solution is system resource management. Computing and communications resources must be allocated and controlled so that mission-critical information flows and application functions are performed reliably in the dynamic network-centric warfare environment. The problem is even more challenging when these mission-critical functions are performed in part by resource-intensive, resource-unaware COTS application software. We propose an approach to system resource management that uses:

- Commanders' policy to determine the criticality of each information flow and application function as a function of mission goals.
- Application-specific quality of service specifications such as information freshness and accuracy, mapped to system-level resource requirements such as network bandwidth and delay.
- A distributed resource management system that allocates to each information flow and application function the CPU time, network bandwidth, and other resources it needs to operate, consistent with its criticality, quality of service requirements, and resource availability. Reallocation occurs dynamically in response to changing mission goals, system workload, and resource availability. Isolation of different traffic streams ensures that a lower criticality channel does not adversely impact higher criticality channels.
- COTS time-space partitioning and hypervisor- technology to enforce CPU and memory resource allocations and to isolate applications from each other to enable integration of applications with differing assurance requirements without changes to any application software. The approach is more dynamic and adaptive for C^4ISR application functions but provide tighter validation for vehicle control functions. Enforcement of network resource

allocations and isolation of information flows across the network remain a challenge, though point solutions exist.

Section 2 outlines our hardware/software architectural assumptions as the context for subsequent discussions. Section 3 lists key requirements for system integration, system resource management, and information management. Section 4 outlines design approaches these functions, including opportunities for COTS exploitation and applicable standards. Section 5 summarizes our recommendations.

2 Architectural Assumptions

Figure 2 shows the hardware and network architecture we assume for an NCW environment. Vehicle and weapons control functions require hard real-time behavior. The controllers, sensors, and actuators that perform these functions are linked by a real-time network such as CAN (Controller Area Network) or MIL-STD-1553. Supervisory control applications implemented on general-purpose processors within the vehicle typically require only soft real-time performance, and can utilize an IP-based wired network. C^{4}ISR applications that span vehicles must use a wireless networks, and due to the nature of those networks must accept soft real-time behavior with delivery times and reliability commensurate with the quality of the network link.

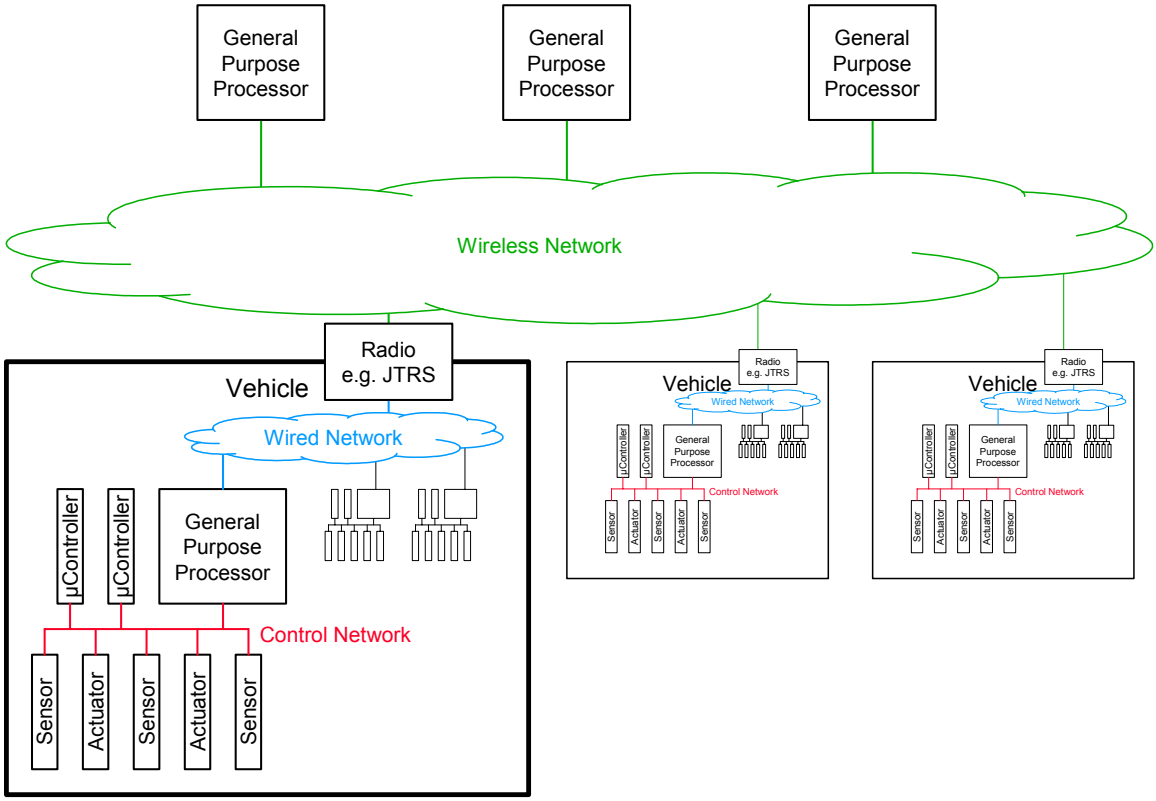

Figure 2. Assumed hardware and network architecture for an NCW environment

Figure 3 shows the protocol stacks involved. Real-time control networks typically have a three-layer stack (application, link, physical) because the routing provided by the network layer is not required. The intra-vehicle wired network and the wireless network are ideally part of the same IP-based network, linked by a router within the vehicle. In this way, intra- and inter-vehicle

communication is accomplished using the same IP-based communication services, the only difference between intra- and inter-vehicle communications being the quality of service available.

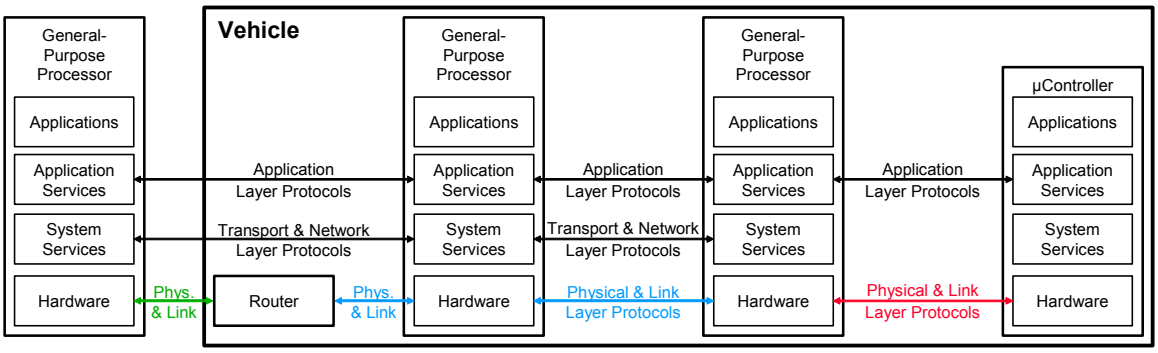

Figure 3. Intra- and inter-vehicle protocol stacks

Figure 3 also shows a coarse breakdown of software functions within each compute node into *applications*, *application services*, and *system services*.

Applications implement user-visible components of system functionality. Application services are those functions, used by multiple applications, that are aware of the structure of the processing and/or data they handle. Application services include naming services, ORB services, information management (addressed in this paper), workflow management, etc. Applications may vary widely in the application services they use.

System services are those functions that manage computing and communications resources: CPU, memory, communications bandwidth, storage, I/O devices, etc., but have little or no knowledge of the purpose to which those resources are put. System services include typical operating system services (e.g. POSIX), transport and lower-level communications services (TCP, UDP, IP, IEEE 802.x, etc.), time service, distributed transaction management, etc. Every application requires at least CPU and memory resources; many applications also require communications and storage resources. System resource management, addressed in this paper, is a system service whose purpose is to allocate system resources to applications so that system performance requirements are met.

3 Key Requirements

Detailing a complete set of requirements for a system of this scale is well beyond the scope of this paper. In this section, we list some key requirements related to system integration, system resource management, and information management.

3.1 System Integration and Resource Management

Interoperability—One purpose of a NCW system is to avoid a system of stovepipes that are unable to interoperate, and which are unable to take advantage of commercial software innovations. In order to make this change, a NCW system must be written with standard protocols and API's in mind, and must support legacy applications and COTS applications, as

well as applications written specifically for the NCW system. Support for COTS operating systems would provide much of this capability.

Resource Management—It can also be said that the purpose of a NCW system is to deliver to the warfighter the information he needs, when he needs it, and with the required quality. Due to the greater integration in a NCW system, information can be easily passed from sensors in the battlefield to computers that aggregate and process the information and to the warfighter who must take action. Delivering that information on time depends on the summed latency of all the networking and processing segments between the source and destination for the information. The latency of each segment is dependent on the capacity of the segment and the loading (network traffic, processor load) of the segment. In addition, many of the segments will be supporting multiple applications. Multiple information flows may be using each network link, and each computer may be supporting multiple applications. If the warfighter is to rely on timely information delivery, the latencies of these segments, and the resource allocations that determine them, must be explicitly managed, controlled, and protected.

Static Resource Management—The allocation of resources to meet computing and communication performance within a system is not new to military systems, or complex distributed systems. Typical weapons systems, such as tanks or airplanes, have fixed sets of applications that are assigned to fixed sets of computers and networks within the vehicle. These assignments are carefully done in an integration lab before system deployment to assure that all applications can meet their performance requirements. Typically the integration process requires multiple cycles of testing, tuning of application priorities, and possibly reassignment of applications to processors and networks in order to meet performance measures. In cases in which performance goals cannot be met through these means, additional processors and/or networks may need to be added. This process is manual and time-consuming, but it is necessary, especially with applications that are safety- and mission-critical.

Dynamic Resource Management—In the NCW environment, there will still be 'self-contained' vehicles that contain their own processors and networks, and for which many applications have been assigned to processors and networks ahead of time. However, some processors within those vehicles will host C^4ISR applications, some of which must be assigned dynamically, on an as-needed basis. Additionally, some NCW applications will span across vehicles, and the set of participants in an application (computing nodes, networks, sensors) that will be available to host these applications can be changing frequently, as mission needs change. The key issue here is that as these applications are deployed and started, the luxury of a manual integration process to assure that performance requirements can be met is not available. Essentially, what has traditionally been done in an integration lab must be done in the field, in a dramatically shorter time frame. If this is not done correctly, an application that is started may push the usage of a resource (such as a processor) to its capacity, resulting in some or all applications using that resource to get less performance that they need.

For this reason, a system for explicitly and automatically managing these resources and the need for these resources by the applications must be provided. Such a system should provide an admission protocol for dynamically starting an application. During admission, the resource requirements for the applications must be presented to the admission system, which will check for the availability of the resources, and then reserve the resources before allowing the

application to start. The resource management system would serve as a broker to the resources, making sure that resources are not over-committed.

Automation—Resource allocation must be done automatically; otherwise, the number of decisions that must be made could potentially overwhelm a system administrator, and those decisions would not be made quickly enough. When the resource broker must resolve conflicting needs for resources, it needs some way of making decisions. For instance, if two separate applications require 60% of a processors time, how will the resource management system react? One approach would be to only support one of the applications, and refuse to start the other. In order to make this decision, the resource management system must understand some notion of importance; the more important of the two applications would be admitted, while the less important would not. Here, importance is an abstract notion taking into account many factors such as the importance of an application to the mission and the rank of the person requesting the application. The resource management system must balance the needs of all applications in the context of what is needed for mission success, and allocate resources accordingly.

Graceful Degradation—The previous example points out the harshness of either/or admission decisions, and motivates the need for flexibility in resource allocations. Some applications may work best at a particular QoS (Quality-of-Service), but will work acceptably at a lower QoS. An example is an application that ideally would run at a periodic rate of 100 Hz, but can operate in a degraded mode at 60 Hz, which also results in a lower resource usage. In the example above, if the application with lower importance can get by with 40% processor usage, then both applications can be admitted, because the summed resource usage would be less than or equal to 100%. For this reason, it is believed that specifying a acceptable resource range, rather than a desired resource point value, is desired. During operation, some applications can be gracefully degraded in order to fit in more important applications.

Adaptability—Adaptability is also a key requirement of the system, due to the changeable nature of the NCW environment. The set of applications operating will be changing, the set of resource available (processors, networks) will be changing, and the performance of wireless networks will be changing (due to mobility and environmental factors). Without the ability to adapt, the NCW warfare infrastructure will be brittle, with each change possibly breaking something. The primary tool available to the resource management system is to adjust the resource usage of applications in response to changes. A good example is the admission of an application when other applications are already utilizing resources. If the application can fit within the available capacity, no adaptation is required. However, if it cannot and it is of higher importance than other applications, the other applications will be 'throttled back', i.e. have their resource allocations adjusted downward, within the limits of their requested resource range, so that the higher priority application can be fitted in. Specifying a resource range for an application, rather than a fixed value, is a necessity for this type of capability; otherwise, the only way to accommodate the new application is to terminate or move others.

Application QoS—Resources would be typically allocated at a low level, such as a processor time allocation or a network bandwidth level. However, the needs of an application would typically be expressed in application-specific terms. For instance, in a sensor-to-shooter chain, the application may need a bounded latency for sensor information to be processed, aggregated,

interpreted, and forwarded to the warfighter, and yet assuring that latency requires the reservation of several different resources. One approach to this is to require the application to specify each individual resource when it is admitted. Another approach is to specify the QoS requirement at an application level, and provide a means to translate that application-level QoS into resource level QoS' so that the individual resources can be reserved. Translations in the other direction are useful also; they allow constraints in a low-level resource to be translated into an application-level constraint.

Resource Isolation—Equal in importance to resource allocation is the enforcement of those allocations. Otherwise an application might state a resource need, but then use much more than its allocation, and take away resources from other applications, causing them to under-perform. This is an especially important point when you consider the presence of safety- and mission-critical applications, and COTS applications.

Safety-critical applications are those whose failure could cause death, dismemberment, or substantial equipment and/or property damage. Mission-critical applications are those whose failure could compromise the mission in which they are a participant. Typically, safety- or mission-critical applications receive more scrutiny of their software, both through testing and reviews of the design and implementation, than less critical applications. The review process primarily concentrates on the correctness of the application, but it must also be shown how the application might be affected by other applications. If a safety-critical application can have the resources available to it reduced by an adjacent rogue application, then either that application must be examined as well to assure that it won't misbehave, or some sort of barrier must be provided to assure that it cannot adversely affect the safety-critical application.

The first approach can be expensive, since all applications must receive a high level of scrutiny, and all possible combinations of applications and resources must be examined. In a NCW environment, the number of combinations is unbounded, since the combinations of applications, vehicles, and environmental conditions are unbounded. This approach generally only works when the number of combinations can be kept relatively small, and when new applications do not enter the overall system.

The second approach is well known, and usually takes the form of physical partitioning, which is separating safety- or mission-critical applications from other applications. For instance, in a nuclear power plant, computers and networks that control the fuel rods are physically separated from the computers and networks on which the accounting applications are used. However, in commercial avionics systems, the notion of logical partitioning, in the form of time-space partitioning, is well accepted. Time-space partitioning is a way of partitioning the time and space of a resource, such as the processor or network. Space partitioning of a processor is really just allocating and guaranteeing a certain amount of memory for an application, while time partitioning is allocating and guaranteeing a certain amount of processor time for an application. The key attribute of time-space partitioning is that it provides the type of 'resource isolation' necessary to guarantee allocated processor resources to an application.

This 'resource isolation' capability is attractive for COTS applications as well. COTS applications are attractive from a cost and functionality standpoint, but often the size of the application and the lack of source code make it more difficult to assure the resource needs of the

application, and its ability to adversely affect safety-critical applications. If source code is available, but it must be modified to assure that it will live within resource constraints, it will lose many of the attributes that make COTS attractive. Modification of a COTS product to target a non-standard infrastructure raises the cost of the application, and since it is a variation off the main branch of the product, it is likely to receive less attention in maintenance from the original vendor. Therefore, it is desirable to be able to execute COTS applications 'out of the box'. With time-space partitioning, a COTS application can be given an enforced allocation, and there would be no concern that the COTS application would consume more than its share of processor time.

On a network, time- and space-partitioning is allocating and guaranteeing a time slot for the delivery of a data item over the network, or allocating a portion of the network bandwidth for a specific data item. The viability of this approach is very dependent on the type of network. In general, we must recognize that an application's QoS can only be as good as what the underlying resources will provide. In particular, it is impossible to get absolute guarantees across inter-vehicle wireless networks that cannot provide absolute guarantees. In particular, applications that are intolerant of excessive latencies cannot rely on network transport in which the latency cannot be guaranteed to be bounded.

By design, control networks provide deterministic behavior. Some networking standards, such as those for automotive busses (TTP™, FlexRay™), and commercial avionics busses (SafeBus™ (ARINC 659), ASCB-2™) explicitly support the notion of time-space partitioning, typically through static allocations of time slots. Others, such as CAN, provide a level of determinism through a non-preemptive priority system. Wired IP networks typically do not provide deterministic behavior, although special adaptations, such as AFDX (ARINC 664), are able to provide a level of partitioning through an allocation of send and receive bit rates, and specialized switch designs. In general, the strength of the guarantees one can make for transportation over the network is highly dependant on the network. Applications that require determinism cannot be dependent on non-deterministic elements, or they must adapt to the non-determinism.

3.2 Information Management

The Information Management function implements the information flows shown in Figure 1 to give warfighters current, accurate, consistent views of the battlespace state. Information management includes functions for storage, retrieval, search, transmission, and dissemination of information in support of network-centric monitoring and control applications. It also includes functions for creating and maintaining models of information and information transformations.

In today's military force, the problem is not so much gathering the information as getting it to the warfighters who need it, when they need it, without overwhelming them with irrelevant information. Lack of interoperability among stovepipe systems currently prevents effective information dissemination.[5] If that problem were solved, the problem would then be to provide the warfighter some way to get the information he needs within the mass of available data.

Information Management must meet several key requirements:

Interoperability—Interoperability at the application level requires communicating applications to have a common interpretation of the information they share. This is difficult to achieve for a

"business process" as complex as warfighting, but is imperative, especially if the shared information is to be interpreted by application software as well as human beings.

Multiple implementations—It must be possible for different nodes to have different, independently-developed implementations of Information Management, possibly with different levels of functionality and performance, while still being interoperable. Given the scope of the Global Information Grid and the variety of systems that must interoperate, it is not reasonable to require all systems to use the same implementation or implementations from the same vendor.

Types of information—Information Management must support the following types of information:

- Structured, record-oriented information.
- Documents—text documents, imagery, and other large pieces of unstructured information.
- Low-volume, time critical information—both structured and unstructured—that is relayed across the network but may never be recorded in persistent storage.

Information search and access—Individual warfighters will generally receive information from a few sources appropriate to their place in the command hierarchy, with the option to search for other sources as needed. The dissemination occurs through a combination of push (publish/subscribe) and pull (query, browse, download).

Profiles—The warfighter (and/or his commander, acting on the warfighter's behalf) must be able to define his information needs precisely. This definition is called a *profile*. The profile can include the type of information required (e.g. high-resolution imagery), subject matter, geographic area, publication date and time, etc. The profile may be *dynamic*, meaning that it varies over time as a function of factors such as the warfighter's current location or task, the current situation, or environmental conditions.

Effective use of available bandwidth—Information dissemination requires bandwidth, which is often in short supply in the tactical environment. This bandwidth should be used efficiently. Profiles help with this, since they define what information warfighters actually need; other information need not be disseminated.

There are other obvious economies that should be realized. The commonality of information needs among warfighters should be recognized and duplicate transmissions avoided. When publishing a large body of information, incremental changes should be transmitted rather than retransmitting each complete updated version.

Further economies can be achieved using information quality of service and policy, as described next.

Information quality of service—A profile states what information a warfighter needs. It must also be possible to specify timeliness, accuracy, and possibly other dimensions of *information quality of service*. Information Management must meet these requirements whenever sufficient system resources are available, and degrade gracefully when they are not.

Timeliness requirements depend on the nature of the information and its intended use. In many cases, information that arrives late may be of little or no use. The GIG CRD[2] distinguishes between *survival information* and *planning information*. Survival information is information that a recipient requires to avoid danger, to defeat an enemy, or to prevent fratricide. It is typically small in volume (<12KB) and must be delivered within a few seconds after the source generates it.

Timeliness requirements affect bandwidth utilization. If information management has more lead time for delivering information, it can make more efficient use of available bandwidth by smoothing out communications workload peaks and valleys.

There are at least two kinds of information accuracy that affect bandwidth utilization. First, some kinds of information, e.g. imagery, are available at different resolutions. The lower the resolution, the lower the bandwidth requirement. The second kind arises in publish-subscribe. The information source, e.g. a track report, may undergo frequent small changes. Disseminating each change can consume significant bandwidth. A subscriber may be content to receive less frequent updates, each providing the net change since the previous update.

Policy—There will be times when the bandwidth available cannot support all information needs. This bandwidth should be used to disseminate the information that makes the greatest contribution to mission success. Less critical information should be disseminated with lower quality of service, or not at all. The information dissemination function must be able to adapt to available resources, disseminating the most critical information possible under the circumstances.

Information Management must allow commanders to establish a policy that defines the criticality of any information flow. The policy may determine criticality as a function of the information type, source, recipient, and other factors. Each commander in the hierarchy can define a policy at a level of scope and detail appropriate to his/her mission. Individual commanders' policies are then merged so that the relative criticality of any two information flows that compete for the same resources can be determined.

Integration with system resource management—System resource management allocates resources in the common currencies of bandwidth, delay, CPU time and the like. Application clients of Information Management state their resource requirements indirectly in terms of information volume and information quality of service. Information Management must map these application-level requirements to explicit resource requirements so they maybe requested and allocated. It must also mediate between applications and system services when quality of service requirements must be renegotiated due to changes in resource availability. Similarly, Information Management must compute the criticality of each system resource request (transport connection, CPU slot, etc.) based on commanders' policy.

Information flow awareness—Commanders must be able to monitor the flow of information across the network, determine how effectively information is being delivered, and understand the impact of changes in workload, resources, and policy.

4 Approach

The overall system is an *interoperable set of systems*. If we consider a single node in the systems, the overall NCW software at the particular node can be layered as shown in Figure 4. Together, application services and system services form the *software infrastructure*.

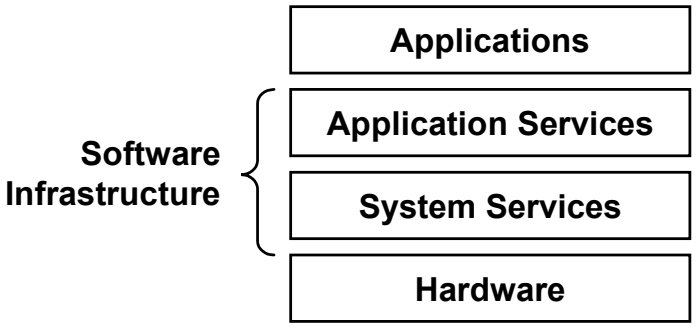

Figure 4. Top-Level Software Decomposition

- ***Applications*—**Applications implement user-visible components of system functionality.
- ***Application Services*—**Application services are those functions, used by multiple applications, that are aware of the structure of the processing and/or data they handle. Application services include naming services, ORB services, information management, workflow management, etc. Applications may vary widely in the application services they use.
- ***System Services*—**System services are those functions that manage computing and communications resources: CPU, memory, communications bandwidth, storage, I/O devices, etc., but have little or no knowledge of the purpose to which those resources are put. System services include typical operating system services (e.g. POSIX), transport and lower-level communications services (TCP, UDP, IP, IEEE 802.x, etc.), time service, distributed transaction management, etc. Every application requires at least CPU and memory resources; many applications also require communications and storage resources.

The key to our approach is the focus on integrating pre-existing and COTS software into the overall system. The main challenge is in supporting different APIs, and providing appropriate levels of assurance for different applications. The assurance must be provided, in not only information assurance (encryption, etc.) but also resource and timeliness guarantees. This is done by providing an environment that COTS applications expect but provides for and enforces resource constraints, and guarantees timing behavior. In particular, System Services provide mechanisms to define, and enforce Information and Resource QoS specifications of the applications.

First, we discuss a set of technologies that we believe are important to meeting our requirements. We then describe our approach to meeting the specific requirements mentioned earlier.

4.1 Some Applicable Technologies

4.1.1 Common Operating Environment & "Middleware" Technologies

One of the common approaches to solving the integration problem is to force all applications to use the same set of System and Application Services, and perhaps the same OS services. In this approach, all applications are developed to a common API, which is separately developed and maintained. This common API (supposedly) meets the performance needs of all the applications

and provides a common integration framework. The Future Combat System (FCS) program is using this approach for integration. The FCS System-of-Systems Common Operating Environment (SOSCOE) layer provides a common API and framework for integrating all FCS applications. There are likely to be multiple implementations (editions) of the FCS SOSCOE API to meet multiple sets of performance requirements.

While this approach is valid in principle, there are some caveats to keep in mind:

- As technologies evolve, the features provided by the COE are insufficient. In some cases, the COE may be adapted to use these new technologies, but in general, it is impractical.
- If existing software is modified to use the COE, porting and validation costs may be prohibitive and the resulting software will not interoperate with other existing systems it currently operates with.
- COTS and third-party packages are usually written to a different set of specifications, and they will never be updated to use the COE.
- In some cases, the performance and reliability characteristics and resource requirements of each application are so different that it is impractical to define a common API with these differing requirements.

Thus, the middleware may become a bottleneck, not only in performance but also in deployment schedule, and becomes a critical path item for almost all development.

Our approach is to say that the COE is common only to the extent that the performance and reliability requirements of the two applications are similar. Applications executing over these COEs are integrated together using partitioning approaches described later.

4.1.2 Use of Desktop Commercial Operating Systems

Traditional desktop commercial OS's such as Windows, and Linux (and its various derivatives) may directly be used for applications that need these OS's and no additional applications need to be executed on the same node. For some applications, it is possible to use variants of Linux (such as TimeSys, or RT-Linux) for soft-real-time applications.

9. DISCLAIMER OF WARRANTIES. To the maximum extent permitted by applicable law, Microsoft and its suppliers provide to you the SOFTWARE PRODUCT, and any (if any) support services relating to the SOFTWARE PRODUCT ("Support Services").

AS IS AND WITH ALL FAULTS, and Microsoft and its suppliers hereby disclaim with respect to the SOFTWARE PRODUCT and Support Services all warranties and conditions, whether express, implied or statutory, including, but not limited to, any (if any) warranties, duties or conditions of or related to: merchantability, fitness for a particular purpose, lack of viruses, accuracy or completeness of responses, results, workmanlike effort and lack of negligence. ALSO THERE IS NO WARRANTY, DUTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT. **THE ENTIRE RISK** ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE PRODUCT AND ANY SUPPORT SERVICES **REMAINS WITH YOU**.

10. EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN **NO EVENT SHALL MICROSOFT** OR ITS SUPPLIERS **BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT**, PUNITIVE OR CONSEQUENTIAL **DAMAGES** WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR: LOSS OF PROFITS, LOSS OF CONFIDENTIAL OR OTHER INFORMATION, BUSINESS INTERRUPTION, **PERSONAL INJURY**, FAITH OR OF REASONABLE CARE), NEGLIGENCE, AND ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SOFTWARE PRODUCT OR THE SUPPORT SERVICES, OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS EULA, EVEN IF MICROSOFT OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Figure 5. Typical End-User License Agreement

Another major reason why COTS OS's are hard to use is that they are designed with different criteria in mind. In particular, lower reliability requirements and large volumes drive their cost lower. Figure 5 shows a representative end-user

license-agreement. Their business models are based on quick upgrade cycles, whereas NCW systems' lifecycle durations are many orders of magnitude longer. We then architecturally need to decide: "how far down do we go with COTS." Many of these problems can be mitigated for C^4ISR systems if these OS's can be used with a Time-Space partitioned or Hypervisor-constrained environments.

4.1.3 Use of Time-Space Partitioned Operating Systems

The Avionics industry has always had a need to reduce certification costs. As a strategy for this cost reduction, the concept of "Time-Space Partitioning" is well established within the industry. Time-Space Partitioning enables isolation of applications in not only space, but also time. This ensures that an errant application partition will not affect the correct behavior of another application partition. In particular, it supports coexistence of applications with differing assurance and resource requirements by enforcing that applications execute within well-defined and constrained time windows, and interact with each other through well-defined interfaces that ensure that the interfaces are safe from interference from other applications. One of the main standards for time-space partitioning at the CPU is ARINC653 [8], and is now available through many commercial Real-Time OS products (such as LynxOS-178, Greenhills Integrity-178B, VxWorks AE653, etc.). There are other time-space partitioning strategies using the Rate-Monotonic scheduling policies that have been successfully employed in Avionics architectures (e.g. Honeywell's Primus Epic architecture for business jets).

ARINC653 based time-space partitioned operating systems have very limited support of traditional OS services. This limitation exists to simplify validation and verification of avionics software. Thus, these operating systems are not directly applicable to dynamic NCW requirements. However, time-space partitioned OS's excel at ensuring applications from each other, and in enforcing application resource constraints. The capability to *dynamically create additional partitions, and allocate resources to these new partitions* will be important for dynamic NCW applications.

The underlying OS for the business and regional commercial avionics market's Primus Epic™ architecture (called DEOS™) provides time-space partitioning based on Rate-Monotonic Analysis based CPU scheduling and complete space partitioning. This enables better response time for interrupt driven and aperiodic applications, while retaining the performance parameters for periodic applications.

4.1.4 Use of Multiple Operating Systems

Once again, because of the number and variety of NCW applications, they will be written for different Operating Systems. Any NCW system must allow these operating systems to coexist in the same network. Imposing a single OS for all applications is hard due to the varied requirements (e.g., if we use out-of-the-box Linux, what are we going to do about safety-critical applications?)

4.1.5 Hypervisor- or Virtualization- Based Approaches

Another interesting OS technology of interest is the "Hypervisor." Hypervisor essentially partitions a CPU and re-exports the same (or possibly emulating another) instruction set to its

applications. It can be used to execute multiple operating systems concurrently, each with its own set of resources independent of the other OS instances. For example, in IBM's blade architecture, the CPUs may be subdivided (or joined together) using the Hypervisor, with a "partition" executing OS/360, AIX or Linux.

Hypervisor based approaches have not been widely tried in the industry but hold great promise for multi-platform applications. In particular, for the NCW system, COTS applications would execute on their "native" OS with resources managed by the Hypervisor. This approach holds great promise in reducing overall system costs and reducing deployment delays.

4.1.6 Communications Services

Communications Services are obviously key to Network Centric Warfare. As shown in Figure 6, there are three very different classes of networks in a system.

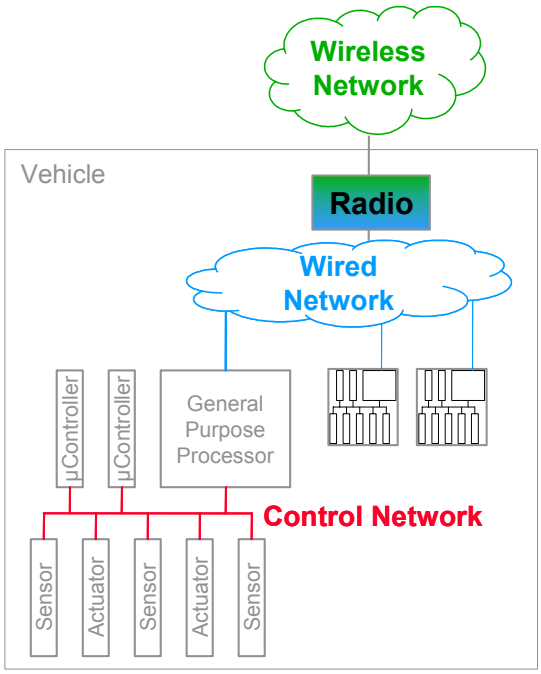

Figure 6. Communication Networks in a NCW System

At the highest level is a set of wireless networks connecting various platforms and soldiers. Examples of such network protocols and products are JTRS, WIN-T, and LINK16. In the long term, these networks will likely support the Internet Protocol-suite—particularly TCP and IP. The inter-platform networks are shared RF and are subject to hostile attacks, infiltration, jamming and intermittent service losses and overloads. The bandwidth available on these networks is constrained and is only a fraction of the bandwidth available within the vehicle.

The platform (a vehicle such as a C^2 vehicle, NLOS-C, UGV, or UAV) typically will have an in-vehicle network connecting various computing nodes on the platform. This network is typically wired. While it may be subject to temporary overloads, it is not as susceptible to outside interference and security violations. This network may have a combination of soft-real-time traffic with possibly some non-real-time traffic. This network is likely to be redundant for possible failures.

Finally, there are a number of Control Networks in the vehicle. These are special networks used in control systems. Most high-rate and real-time control loops are executed across these control networks. Examples of such networks are CAN, MIL-STD-1553B, ARINC659, etc. These networks (except CAN) are very deterministic, and usually provide built-in redundancy in the network. Most safety-critical control is done over these networks. We expect that there will be one or more such networks in the vehicle, with one or more subsystems over the same network.

Figure 7 shows some traffic characteristics of different types of data sent across these different networks.

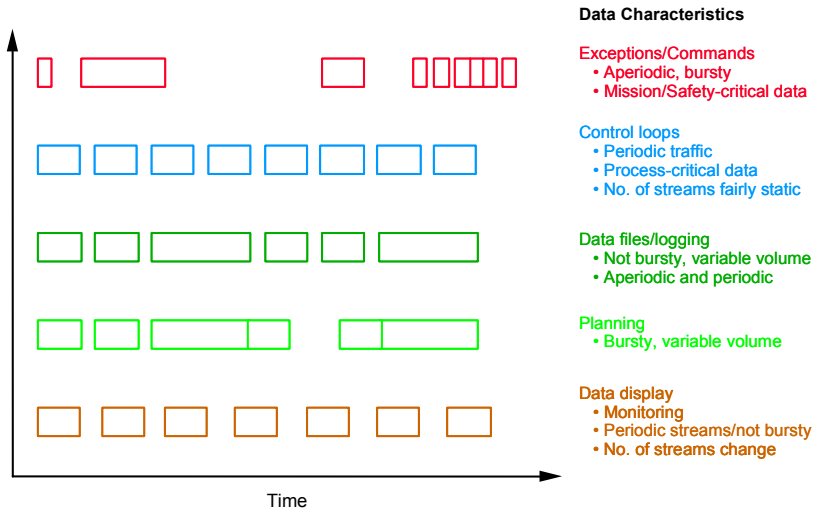

Figure 7. Traffic Characteristics for Different Classes of Data

- The highest criticality messages tend to be exceptions or alarms, and certain urgent messages (e.g. Check Fire). Delays or loss in transmission of such information may lead to loss life or mission failure. These kinds of messages exist over all three categories of the network.

- The second most critical category of messages (depending on the specific process being controlled) are control loop inputs and outputs. High-rate control loops such as propulsion and armament control are typically closed across the in-vehicle control network. For tele-operations, commands must be received from the operator to the control system in a timely fashion; otherwise, operator-induced oscillations may occur in the control system causing unstable behavior of the system. These messages may be periodic (e.g. giving periodic state of the joystick) or aperiodic (giving only movements of the stick as they occur).

- The third category is where most of the communication traffic will lie. It includes planning, simulation data, sensor reports, battle command execution, and most other activities. Most logging-, and sensor- data tends to be variable volume but not very bursty. On the other hand, battlefield planning and execution information is bursty and variable volume.

- Finally, there are a number of applications displaying information from the battlefield, as well displays of the control system inside a vehicle. These streams tend to be periodic and not bursty; however, the number of streams may change over a given time-period. Some streams may have higher criticality.

It is important to recognize the significant differences between the three different types of networks. In principle, it is a good idea to develop APIs and middleware layer that makes the applications agnostic to the actual network. In practice, it is extremely hard due to the significant differences in their characteristics.

4.1.7 Isolation through Hardware Separation

One common way to isolate hard- and soft- real-time applications is to use different hardware boxes for these applications. In some cases, the separation extends to individual networks as well. The same issues occur with security and information-assurance issues.

These issues are well-known and are not discussed in this paper.

4.2 Our Integration Approach

Our main integration approach is to use a combination of time-space partitioned OS's and specific instances of commercial OS's on a Hypervisor-based OS. While this approach has not yet been proven, it shows great potential in not only meeting the real-time and safety-critical needs of NCW applications, but also in enabling direct use of COTS and legacy applications. The virtualization and time-space partitioning approaches essentially define a resource "straightjacket" for applications that enforce and guarantee their resource requirements, thereby providing some level of determinism *even for applications never so designed*.

4.2.1 Support for Multiple Environments

Our goal is to provide the environment an application expects. The environment includes not just the operating system, but also a set of resources available for the execution.

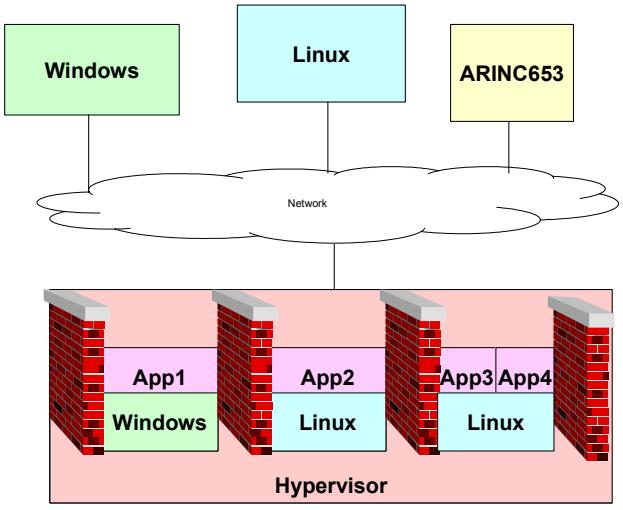

Figure 8. Providing Multiple OS APIs through use of Virtualization

Figure 8 shows a possible mechanism for using Virtualization (or Hypervisor) technology to achieve support for multiple OS environments. The Hypervisor creates hard resource containers in which applications (and OS's) can reside. These resource containers *guarantee and enforce* resources available to each of the partitions. In this example, App1 and App2 have a fixed amount of resources available, whereas App3 and App4 share resources given to the particular partition. The figure also shows physical partitioning of physical resources, each with its own supported OS. A key question here is what happens to dynamically created applications and resource allocations for them. For dynamically allocated partitions, we will need to dynamically allocate these Hypervisor partitions, or share one or more statically allocated partitions.

The biggest concern in our approach is the allocation of network resources. Hypervisor and ARINC653 OS technologies deal with only the CPU side of the problem. For Control Networks, we believe that a time-space partitioned bus such as TTP would be a good choice. For the intra-vehicle main network, resource reservation and policing through limiting outputs from each node can be used.

For the inter-vehicle network, there can be no absolute guarantees. Any guarantees that are provided must take into account the unreliability of the inter-vehicle radio network. Historically, most applications have not assumed any fixed bandwidth allocation from the radio network, so it

would not be a problem to not provide absolute guarantees for pre-existing applications. New applications must be adaptable to the reliability issues in the network.

4.2.2 Multi-Resource Management

The infrastructure would provide an API and framework to provide explicit management of resources. The important parts of it are:

- An admission system for automatically controlling and allocating access to resources
- A means of expressing QoS needs both at the application level and the resource level, with the ability to translate between the two
- Support for specifying a range of QoS' in order to allow flexibility to adjust resource allocations as the situation evolves.
- Tied to policy to enable automatic trade-offs, thus favoring some applications over others according to their contribution to mission goals.
- The ability to adapt to changing situations, including the admission of new applications, the termination of applications, the reduction or elimination of some resources, and changing military priorities.
- Reliance on the underlying OS and network system to provide resource isolation, thus protecting the allocations of applications.

All applications would either need to express their QoS needs or accept best effort performance. In addition, each application type would need to provide translation code to translate application QoS needs to resource QoS allocations, and vice versa. To start a QoS-aware application would first require an admission process, in which the application-level QoS is translated into resource-level QoS', and then the availability of these resources is checked. All the resources for the application would need to be reserved; if some resources were insufficient, the overall application QoS would need to be adjusted to be consistent, or the admission of the application would fail.

The ability to adapt is key. When an admission request is made, it may be that insufficient resources exist, with some being tied up by admitted applications. However, if the application, according to its current policy, is more important than the applications that are running, it will make sense that the other applications be 'throttled back', thus freeing up resources for the new application. On the other hand, if an application terminates, the resources that it was using must be freed up for new applications, or used to 'throttle up' existing applications. This is also true when the capacities of various resources change. A distributed application in which the capacity of one network link is reduced may need to have the other resource allocations for that applications reduced in order to be consistent; it makes no sense for upstream parts of a distributed application to produce data faster than downstream elements can process it.

Another crucial part of the admission process is the tie in to the underlying OS, especially with time-space capabilities. For the CPU resource, allocations of resources should lead directly to new partitions with time and space guarantees.

4.3 Information Management

This section outlines approaches for meeting the Information Management requirements listed in Section 3.2.

Much of Information Management can be implemented using COTS products that implement vendor-neutral APIs and protocols. The ability to run this software "out of the box" in the NCW environment, as described above, results in significant saving in cost and time to deployment compared to custom development.

4.3.1 Information Stores

The three types of information listed in Section 3.2 require different kinds of *information stores*:

- Structured, record-oriented information can be stored in a SQL-compliant relational database, where it can be queried, inserted, updated, and deleted.
- Documents (text, imagery, etc.) can be stored in the operating system's file system. A document's descriptive metadata (author, date created, etc.) can (for some operating system) be stored in the file system itself or in an associated relational database. Alternatively, both the document and metadata can be stored in a relational database that supports storage of large binary and character objects. The DoD is defining standard descriptive or "discovery" metadata.[7]
- Low-volume, time critical information can be stored in a main-memory database. There are several available commercially for the embedded system market. Many implement a subset of the SQL language for query and update operations. These are appropriate for storing both structured data and unstructured data (such as text messages) with associated descriptive metadata.

Information stores such as these will reside on many nodes across the system of systems. Applications need to be able to locate these information stores, read and update them, and disseminate their contents on a regular basis via publish/subscribe.

4.3.2 Locating Information Stores

Two vendor-neutral directory standards, X.500 and Lightweight Directory Access Protocol (LDAP) can be used to catalog the information stores so that applications can locate them. X.500 was defined as part of the OSI protocol suite, but can also be run on top of TCP. LDAP has been developed more recently, and is more widely implemented. Both use the same model of directory information, so that the same directory can be made available for access via either protocol. Both support white page (lookup by name) and yellow page (lookup by search predicates over attributes) operations.

LDAP and X.500 support replicated, distributed directories. That is, the directory need not be stored at a single node. (In the NCW environment, this would be an unacceptable single point of failure, and would have low availability due to intermittent network connectivity.) Rather, the directory tree can be broken into fragments, and each fragment stored on one or more nodes. For instance, each vehicle could store the directory entries for all its information stores. The

X.500/LDAP referral capability allows a client application to query one directory server, and be referred if necessary to another directory server where the information can be found.

X.500 and LDAP both support client authentication, so that directory read and update can be controlled. However, the LDAP access control model has not yet been standardized, so access control behavior may differ from server to server.

X.500 and LDAP are general-purpose directories that can record entries for many kinds of objects—persons, organizational units, network nodes, etc. The advantage of using these standards for locating information stores is that they build on the DoD X.500 and LDAP infrastructure already in place.

4.3.3 Accessing Information Stores

Once an application locates an information store, it must be able to locate, read, update, and delete specific documents or records, and create new ones.

For document stores, LDAP can be used to locate specific documents using yellow-page lookup against the associated descriptive metadata. Once the desired document(s) are located, they can be accessed using FTP. Similarly, a client can use FTP to create a new document in the store, and create an associated directory entry. All these operations are subject to access control.

Remote access of SQL databases is nothing new and can be implemented in the NCW environment. Unfortunately, the ISO/IEC standard Remote Database Access (RDA) protocol, the only vendor-neutral application protocol for executing SQL statements against remote databases, has seen little vendor acceptance. Each database (Oracle, DB2, MySQL, etc.) has its own client-server protocol. Therefore each client node that needs to access a particular remote database must have software specific to that database (e.g. an ODBC or JDBC driver) installed. The purchasing power of the U. S. Government might be used to encourage COTS database vendors to comply with the RDA standard.

Furthermore, the ISO/IEC SQL standard is very complex, and specific products vary in the completeness and correctness of their implementations. To avoid vendor lock-in, developers must avoid vendor-specific SQL features or standard features that are not widely implemented.

A reasonable alternative to sending explicit SQL to remote databases for execution is to define application-specific database query and update services (procedures) that can be invoked remotely via web services, CORBA, or other vendor-neutral protocol. This may be desirable in any case to enhance database security and integrity, but is definitely less flexible than ad-hoc SQL queries.

4.3.4 Publish-Subscribe and Information Replication

In the NCW environment, accessing remote information stores is problematic because of intermittent communications connectivity and/or limited bandwidth. A widely-used alternative is to selectively push information from node to node via asynchronous messaging so that applications can access it locally when they need it. This push model is often implemented using message-oriented middleware such as IBM's MQSeries. Tactical messaging protocols such as Link 16 also use this approach.

We propose a variation of the usual message-oriented publish-subscribe paradigm that better meets the requirements stated in Section 3.2: profiles, effective use of available bandwidth, information quality of service, and integration with system resource management. The concept of operations can be summarized as follows.

Each node has one or more information stores. Applications produce and consume data by writing to and reading from the local information stores; they do not concern themselves with inter-system communications.

To the extent that there are multiple copies of the same information on different systems, those copies must be kept *more or less synchronized*. Information Management uses inter-system communication, in the form of messaging (or file transfer, in the case of a document store), to maintain this synchronization. Each message encodes a change in state of the information store at the message source; the same change of state must be effected at the message destination to keep the information stores synchronized. More precisely, as shown in Figure 9, a *view* of the information store at Node A must be kept synchronized with a compatible view of the information store at Node B. If an update to the information store at Node A does not affect the shared view, no message to Node B is required. These views extend the usual notion of an information dissemination profile.

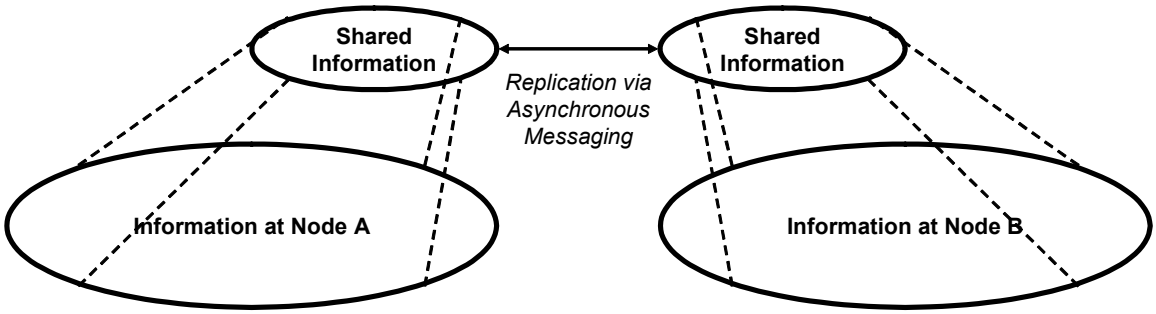

Figure 9. Selective replication of information across nodes.

"More or less synchronized" means that an application can tolerate some delay between the time the information it needs is produced on a different system and the time it is available locally for consumption. The tolerable delay could vary from a fraction of a second to hours or more. The application may also tolerate some inaccuracy in its local copy of the information—it may not care about small changes in data values. An application's information quality of service (IQoS) requirements include its accuracy and delay tolerances for specific information elements. These might be specified at design time as part of the system-of-system model, or at run-time.

XML-based message formats are a natural choice for the inter-system messages, but other formats (e.g. J series) can also be used. As stated earlier, each message represents a state change of the (a view of) the local information store. Consequently, the message contents should be compatible with DoD standard information models.

Inter-system communication is assumed to have limited bandwidth and/or time-varying connectivity, bandwidth, or cost. When Information Management performs inter-system communication to maintain data synchronization, it exploits applications' IQoS requirements to optimize use of available communications resources. In general, lower IQoS requirements reduce

the communication resources required to maintain adequate synchronization. Information Management must translate IQoS requirements to system-level QoS requirements (chiefly network bandwidth and delay requirements) prior to requesting those resources. This translation is relatively straightforward if rate at which applications update their local information stores is known.

This concept of operations differs from the usual approach to distributed information management in commercial and military vehicle systems. In that approach, applications explicitly invoke communications services to send and receive information between systems. This could be called the *message-oriented approach*. Our approach, in contrast, is *state-oriented* because applications merely store and retrieve information locally, with no notion of how this information might be shared across systems. Information Management is responsible for:

- Detecting the presence of new information in a local information store,
- Sending appropriate messages (or files) to other systems,
- Receiving these messages on the remote systems, and
- Using them to update the remote systems' information stores.

The decision to send a message, and the message content, are based on

- The content of messages previously sent,
- The current information state at the sending system, and
- The recipients' IQoS requirements

Information Management makes this decision, relieving applications of this responsibility.

5 Summary

In this paper, we have summarized the goals and intent of Network Centric Warfare, identified key requirements, and proposed an approach to the infrastructure necessary to meet those requirements.

There remain questions that need to be answered for our approach:

- No commercial software package or operating system exists that provides both time-space partitioning and hypervisor-like capabilities. What would be the required effort to build this capability? Would this capability be built on top of a particular OS, or would it sit between the hardware and the hosted OS'?
- The resource management capabilities described do not appear to exist in a COTS package, and have only been demonstrated in research environments. What is required to mature the technology such that it is suitable for deployment?
- What is the impact of resource reservation protocols in a complex wireless network? For a network supporting many distributed applications, might resource adjustments in one application lead to adjustments in others, causing changes to ripple through the network.

References

1. David S. Alberts, John J. Garstka, and Frederick P. Stein. Network Centric Warfare: Developing and Leveraging Information Superiority. 2nd Edition (Revised). DoD C4ISR Cooperative Research Program, Washington, D.C., 1999. Available at http://www.dodccrp.org/Publications/pdf/ncw_2nd.pdf

2. U. S. Joint Forces Command Capstone Requirements Document: Global Information Grid. August 30, 2001. Available at http://handle.dtic.mil/100.2/ADA408877

3. Department of Defense Joint Technical Architecture, version 6.0, October 3, 2003. Available at http://jta.disa.mil/

4. L. Peter Deutsch, The Eight Fallacies of Distributed Computing. Available at http://java.sun.com/people/jag/Fallacies.html.

5. Gen. Kevin Byrnes, remarks at Annual Meeting of the Association of the U. S. Army, October 6, 2003. Available at http://www.ausa.org/www/news.nsf/0/53E45CC12023D80E85256DB80050064C?opendocument

6. Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.

7. Deputy Assistant Secretary of Defense, Department of Defense Discovery Metadata Specification (DDMS) Version 1.0. 29 September 2003.

8. Avionics Application Software Standard Interface, ARINC Specification 653, Published: January 1, 1997. Aeronautical Radio, Inc. Also, Draft 3 of Supplement 1 to ARINC653, published July 15, 2003.