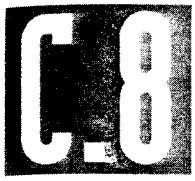

Interoperability

Principal Authors:
Frank K. Bamberger, Peter Ford and Jeannette Wing

Additional Contributors:
Forest Baskett, Edward G. Britton, James M. Burger, K. Mani Chandy, Frank DeMartin, Gary Demos, Seth G. Fearey, John Gannon, Susan Gerhart, Branko Gerovac, James Gosling, Allen Gough, Erik K. Grimmelmann, John V. Guttag, John Hemphill, Grace Hinchman, Suzanne Johnson, Kenneth R. Kay, Cordell Ratner, Jeff Rulifson, William L. Scherlis, Richard D. Schlichting, William T. Smith, Steve Stewart, Mary K. Vernon and Charles York

1. Introduction

The National Information Infrastructure will require an unparalleled interaction among technologies, operational efforts and multi-supplier development of infrastructure. These interactions cannot be wholly prescribed at the outset. Instead, the NII requires an open architecture in which many independent providers of services, information, applications and equipment can participate and interact. These actors will bring new visions, new technologies, and ultimately new hardware and software systems that will bring fundamental changes in the ways in which people, organizations and governments interact with one another.

It is implicit within the vision of the NII that users will want the NII to allow them to connect to any system that they are permitted to access, and to connect combinations of different systems. For example:

- Various participants of the health care industry, including doctors, insurance companies, laboratories and hospital administrators, will need to access a patient's records to locate and add information when appropriate.
- A K-12 student may wish to access different libraries across the nation to obtain relevant information for his or her term paper.
- Two users of a legacy proprietary protocol may wish to connect via the NII.

These activities entail the interoperation of many systems that were not originally designed to work together.

For the purposes of this report, "systems" include people, applications, services, appliances such as computers, and networks. Interoperability in the NII is the ability to combine two or more systems into a single acceptably seamless and acceptably efficient system, while also allowing systems to differentiate themselves and provide customer choice. We mention "seamless" because if too many seams between the constituent systems show, the effort required to use the system outweighs the gain. We mention "efficient" because significant attempts at composing systems have failed because, while the systems were able to work in concert, they resulted in unacceptably high consumption of resources.

2. Challenges for NII Interoperability

A challenge to the research and development community is to facilitate the interoperability of appliances, applications and services, on an unprecedented heterogeneity and scale, for users of the NII. Interoperability is not achieved among incompatible components. It must be designed in or subsequently added on via retrofits that allow components to "plug and play."

2.1 Interface and Protocol Mismatches

Even in the simple case where the objective is to integrate new systems that are designed from the beginning to interoperate, the challenge is substantial. In particular, large software and/or hardware systems that are intended to interoperate are built from specifications, and these specifications are inherently incomplete. This leads to syntactic and semantic mismatches that are usually discovered dur-

ing system integration and testing (but may not be discovered until long after the system is in production use).

Interoperability problems are of two kinds: interface mismatches and protocol mismatches. An "interface" describes a component's characteristics, e.g., its functionality, structure and performance. A "protocol" describes the connections the components use for communication, e.g., reliability (no lost messages, no duplicates), directionality, rules that govern the temporal ordering of messages and performance. If system A uses a different protocol than system B, then we have a protocol mismatch. If system A makes incorrect assumptions about the meaning or the format (grammar) of data that it receives from system B, then we have an interface mismatch.

Interfaces and protocols are contracts between interoperating systems. Their specifications define the requirements for communications between heterogeneous systems. They are thus the keys to interoperability. There is a problem when the contracts are violated, incomplete, inappropriate or not in place. More specifically, an interoperability problem arises between two components if their interfaces do not match, if their protocols do not match or if their interfaces and protocols do not match. If the protocols or interfaces are not clearly specified, it is difficult to determine whether an interoperability failure is truly a mismatch or whether one of the systems hasn't fully or correctly implemented the specified protocol or interface.

Interfaces and protocols may be open or closed, formal or informal. For the purposes of this report section, open roughly corresponds to "published" or otherwise openly available to anyone who wishes to use the protocol and interface. If open, the protocol and interface may be standard (i.e., developed and ratified by a technical committee that includes representation from many different suppliers and users) or not.

2.2 The Special Challenges of Legacy Systems

Interface and protocol mismatches are typically more severe when dealing with legacy systems. Furthermore, the options for resolving the mismatches are significantly more limited when attempts are made to integrate legacy and other systems that were originally designed and operated independently. In the past decades, thousands of systems have been built independently of each other with little thought to interoperation.

In the future, thousands more will be built, some explicitly to work together, others not. Systems that work in concert in one context will be combined with others in other contexts. The extensive reuse of these systems significantly increases their value, and thus adds to the nation's wealth. The NII presents us with the key challenge to make available to users of the NII the huge amount of data administered under the legacy systems that are made available on the NII. In the past we have learned, often to our considerable profit, how to get systems that were designed and implemented independently to work together, but generally only in very narrow contexts. The challenge is to develop more general methods for legacy systems to interoperate with each other and with new systems, in new ways and for new purposes, in the broader context of the NII.

2.3 Mechanisms for Resolving Mismatches

Three possible approaches to resolving or mitigating interface and protocol mismatch problems for legacy (and other) systems are:

- *Pair-wise approach:* For each pair of non-interoperable components, A and B, we need a pair of translation functions, one from A to B and one from B to A. Full connectivity among N components can require N(N-1) translation functions. The Information Access section calls this approach "point-to-point connectivity."

- *Common language approach:* For each component, A, we need a pair of translation functions, one from A to S and one from S to A, where S is a common language (i.e., protocol and interface) with which every component can communicate. S can be viewed as a single standard, e.g., RS232 or Ethernet, to which all players subscribe. Full connectivity among N components requires at most 2N translation functions. The Information Access section calls this approach "federated architectures" where the "facilitator" is the common language. This approach is also taken in the "hourglass" model of the Network Components and Protocols section; that is, the Internet Protocol (IP) is the common language used in the Internet.

- *Broker approach:* This is a hybrid approach. There are third parties ("brokers"), each of which understands some subset of the many protocols used by the components. If A wants to communicate with B, a broker acts as an intermediary. The broker figures out how to translate A's language so B can understand it and vice versa. Implementers of A and B do not have to provide translation functions. Instead, brokers supply them. One can view the union of the brokers as a set of standards. This approach is related to the Information Access section's "mediator network." If this approach can be realized effi-

ciently and without restricting functions, one advantage it offers for the NII is that it allows a set of acceptable standards rather than requiring that a single standard be agreed upon and adopted by government, industry, etc.

2.4 The Need for Common Interfaces

The NII will involve integrating communication and information resources from diverse suppliers to an extent not considered in the design of today's heterogeneous computing systems. A key challenge is to use the emergence of open systems and of reusable software technology to develop and implement future systems and services that are interoperable to an increasingly greater degree than in the past. Definition of common interfaces is as important to the NII as agreeing on electrical power standards and shapes of electrical sockets is to the production of domestic appliances. Common interfaces decouple design choices, enabling systems and service developers to respond more rapidly to new opportunities to enhance their products. The value of common system interfaces is clear at the level of telecommunications pathways. NII research should accelerate development of appropriate common system interfaces at higher levels of services.

3. Interoperability Research Goals

In practice, given a collection of system components to be composed, achieving interoperability consists of the following general steps:

1) Defining or selecting appropriate protocols and interfaces.

2) Implementing or adapting systems to employ the protocols and interfaces. This may require designing and implementing new systems, modifying or extending an existing system, or adding a new component (e.g., gateways).

3) Integrating and testing the interoperating components, which may be done incrementally and requires modifying or refining steps 1 or 2 if mismatches between the systems are discovered.

Goal #1: To enhance our ability to compose systems into a single acceptably seamless and acceptably efficient system.

Here we are concerned not only with systems that exchange information but also with systems that update information. Research needs to be conducted on techniques for each phase of the life cycle, including design, integration and testing. An important subgoal is to reduce risk of catastrophic failure caused by interoperating NII systems. Although two systems may operate innocently independent of each other, features in one system may interact in disastrous ways with features in another. On the scale of the NII, the consequences of unanticipated, undesirable interactions could be staggering. Furthermore, the particular features that interact poorly may obscure features, or the consequences may not be immediately detectable. Thus, standard test suites in artificial environments prior to full integration in the NII may not detect the problem.

Goal #2: To develop an intellectual framework to discuss, quantify and demonstrate interoperability.

The framework should enable us to reason about the interoperability of systems, to predict the resulting composite system's behavior, to measure its performance and to identify unexpected interactions between the components that make up the composite system. It should also enable us to determine to what degree a system is interoperable, e.g., under normal operation, under certain restricted assumptions, or using just one particular protocol for communication. An example is the ISO/OSI layered protocol architecture, which provides a framework for arguing that the Internet achieves interoperability at the IP layer.

There are a number of different contexts in which significant interoperability problems arise and the above goals should be addressed. Example contexts, defined by the types of systems that are supposed to interoperate, include (cf, Computer Systems Policy Project, "Perspectives on the National Information Infrastructure: Ensuring Interoperability," February 1994.):

- Appliance to network (e.g., connecting a terminal device or a computer system to a network).
- Appliance to service (e.g., playing back a multimedia presentation created on a particular computer on another computer with a different operating system).
- Service to service (e.g., a financial service and an authentication service).
- Network to network (e.g., sending e-mail from a user on one network to a user on another network).

Different examples in each of the contexts impact the nature of the interoperability problem. For example, particular health care, manufacturing, and/or financial services impose different dependability and/or real-time requirements on the interoperability of component systems. Note that a fifth context, person-to-application service, deals with human interfaces, which should be consistent and tailorable to each user. This type of interoperability is dealt with in the Ease of Use section.

4. Research and Development Recommendations

The Information Access section and the Network Components and Protocols section each contain recommendations for research in interoperability as it relates to that particular domain. In this section we take a broader view, including (at least) the four contexts defined above, protocol mismatches and interface mismatches, and various parts of the interoperability life cycle.

The first goal of interoperability research is to make it easier to build systems that interoperate seamlessly. The second goal of interoperability research is to provide a framework for quantifying and demonstrating interoperability. Below we recommend specific research projects for achieving each of these goals. We reference the Information Access and Network Protocols sections where appropriate.

4.1 Enhancing the Ability to Connect Systems Together

We suggest two broad strategies to pursue in parallel. The vertical approach is application domain-specific and aims to exploit the structure and semantics of the domain. The horizontal approach is domain-independent and aims to find commonalty across multiple domains. For each of the NII application areas, we suggest the development of 1) domain-specific paradigms and tools and 2) common interfaces and protocols (commonalty here is within the same domain). Some services cut across domains, and it would be more cost-effective to provide them to all domains using a common interface or protocol. This has the advantage of supporting interoperability across different domains. Examples of these services are auditing, billing, authentication, naming and transactions.

Research Projects

- Identify, compare and contrast interoperability paradigms on realistic problems in all interoperability contexts. Both realistic problems in integrating legacy systems (e.g., name servers) as well as realistic distributed services that will be designed to interoperate should be considered. Support for dynamic as well as static composition of components should be investigated. Example paradigms include the common language and third-party broker approaches outlined in Section 2 (e.g., mediator and facilitator architectures for database interoperability, also recommended for study in the Information Access section), multiple protocols (e.g., X/OPEN's Multiprotocol Transport Networking, multiprotocol data link controls), integrators such as workflow managers (e.g., National Software Works' Works Manager), reusable software technology and informal guidelines for designing systems that will interoperate in many different environments. Related issues include 1) trade-offs between using "wrapper" technology versus building in the interoperation capability and 2) comparison of various communication paradigms (e.g., client-server, peer-to-peer, pipeline).

- Identify paradigms that can exploit the vocabulary, architectures and semantics of particular application domains. We know from experience that the more knowledge we take into consideration, the more we can fine-tune a system we are building (usually by making it more efficient or by making it "smarter"). Also, each NII application area will have a set of problems unique to it; it may be more cost-effective to solve the specific problem rather than to build a general-purpose solution for just one user. For example, the Information Access section gives examples of different mediator and facilitator architectures for information systems. One might be appropriate for CAD/CAM design databases but not for K-12 distributed libraries.

- Identify and define common services that can be used by different applications in order to enhance their ability to interoperate. Use of common underlying services can avoid mismatches between components. As an example of a notable success in this arena, the use of the Kerberos authentication service by a wide variety of applications (file services, mail services, remote login services, etc.) has had remarkable leverage in the local area network context. Examples for the NII might include authentication services or support for transaction integrity for industries like banking and health care. Services specific to particular application domains and services that can be used by many different kinds of applications should be identified. Reference or prototype implementations of these NII interfaces and services should be developed for proof of concept.

- Do economic analysis to determine appropriate incentives and avenues for implementing common services.

- Develop mechanisms for making common services widely available in the "NII operating system." These services should be available to any component that needs to interoperate with other distributed components. The information access issue of locating these services is extremely important. The common services could be made available either from special NII "servers" or in the NII substrate of end systems. Common interfaces and protocols that are specific to a particular application domain should be made available as a library or repository for sharing among application builders. Providing common services in the infrastructure can also simplify the development and increase the reliability of applications.

- Develop approaches for migrating the common services from applications to the pool of common services. This research problem is also discussed in the Applications Development Infrastructure recommendations.

- For each common service, define a range of interfaces, e.g., minimal to maximal, and a range of protocols that support the service's functionality. For example, minimal transaction support might provide only the means to abort a computation at a single site; maximal might provide user-specifiable locking protocols for ensuring global consistency constraints. Some applications will want to interact with a narrow interface; others will need more functionality and will be willing to cope with the additional complexity.

- For each of the interoperability contexts, identify key protocols and interfaces that might be adopted by the relevant communities. For example, IP has been widely adopted as the protocol for network-to-network interoperability in the Internet. In the service-to-service context, it might be useful to develop protocols that satisfy weaker definitions of transaction integrity for distributed computing than is provided by the classical atomic commit protocol.

- Investigate the development of protocols that negotiate with each other and/or tolerate mismatches and errors. Negotiation could range in complexity from automatic to user-directed. For example, one way to resolve a mismatch is to inform the user and have the user perform some action. Another is to have the mismatched parties negotiate a way to resolve the conflict.

- Develop new or enhance existing techniques for error avoidance, error detection, error correction, and error containment (firewalls) for services and applications. In many existing systems, the effort expended on error containment is greater than the effort expended in designing and implementing the basic functionality of the service. Simpler approaches would thus significantly reduce design time and cost.

- Identify interoperability problems that could bring down the NII (at each level, from application to appliance, and between levels, e.g., person/application or appliance/network).

- Do sociological research into methods for more efficient standards development. Standards development requires careful consideration through a deliberate and democratic process. However, current standards development processes make very inefficient use of people's time, and some standards are never actually put into practice. On the other hand, some interfaces, e.g., MS-DOS, become standards without an explicit standards development effort. Can we learn to distinguish better than randomly? Can we learn by studying cases? Research to address these questions might be undertaken jointly by a computer technologist, a historian of science and a sociologist. Pilot research projects to determine the viability of this research should be undertaken.

4.2 Framework for Quantifying and Demonstrating Interoperability

The thrust of this goal is to better understand the systems we build in the context of interoperability.

- Identify and test (on realistic problems) methods that can be employed during the design phase to determine, given the specified behavior of constituent systems, as much of the behavior that will result from the composition as possible. Of particular importance are reliable techniques for identifying and resolving interface and protocol mismatches. Interface and protocol design tools such as finite-state machine compilers and logic checkers should be investigated. An example research project would be to experiment with specifications of a real system such as the Q.93B switching protocol to determine how well one can predict and reason about its actual behavior.

- Identify realistic systems and appropriate specification methods for which composition of interface specifications greatly facilitates the composition of the actual subsystems.

- Investigate tools and testbeds that can be employed during the integration phase to test whether each system to be composed conforms to the protocols and interfaces it uses and whether its resource consumption is acceptable. (e.g., applications that flood the network with messages are bad.) Can generic "interoperability testbeds" be developed?

- Develop techniques for identifying and resolving semantic mismatches, during system design or during integration and testing. Existing specification techniques need to be extended to describe timing requirements, performance, resource usage and fault modes, in addition to I/O and data manipulation functionality.

- Examine case studies of the development histories of existing interoperating systems to identify sources of unanticipated interoperability failures.

- For applications and application domains that have safety-critical properties or high data integrity constraints, identify key components that can benefit from formal specification and verification.

- Develop a model of interoperability such that the meaning of components, interfaces and protocols is clear. Use this model to define more precisely the interface mismatch and protocol mismatch problems. Use these definitions to identify when mismatches may arise in practice. Use it to find ways to resolve mismatches.