Instruments and Sensors as Network Services:

Making Instruments First Class Members of the Grid

Randall Bramley[1], Kenneth Chiu[1], John C. Huffman[2], Kia Huffman[2], and Donald F. McMullen[3]

{bramley,chiuk}@cs.indiana.edu,
{huffman,kihuffman,mcmullen}@indiana.edu

[1]Computer Science Department, Indiana University
[2]Indiana University School of Informatics
[3]Pervasive Technology Laboratories at Indiana University

1.0 Introduction

An aspect of Grid computing that has not yet been well developed is the integration of scientific instruments into a Grid computing environment. Instruments are still largely "off-line" as far as downstream software analytical components are concerned and instruments are not at all first class members of the Grid with respect to location, allocation, scheduling, and access control. This is a serious problem as three issues continue to grow in importance in research: 1) investments in geographically extended (e.g., international) collaborations organized around large shared instrument resources; 2) increasingly "real-time" use of instruments by remote researchers both for "first look" activities and pipelined data acquisition and reduction; and 3) sensor networks with hundreds to thousands of nodes being deployed.

Grid computing [1] has proven to be a useful paradigm for organizing and harnessing distributed resources, and for providing common services such as scheduling and authentication. Considerable work [2, 3] has been done in the areas of computational and storage Grids, and on combining computing and storage resources. Another important development in scientific computing is the application of component software technologies [4, 5] that promise to enhance code modularity, encapsulation and reuse. At the confluence of component software engineering and Grid computing is Web Services [6] and the Open Grid Services Architecture (OGSA) [7]. Web Services provides a platform independent, location independent means to execute code through a remote procedure call or document-oriented interface. Progress has been made recently [8] in developing and evaluating OGSA and in using Web Services as a model for software component interfaces [9].

The program described here seeks to research a Common Instrument Middleware Architecture (CIMA) to improve accessibility of instruments and to facilitate their integration into the Grid. CIMA middleware is based on current Grid implementation standards and accessible through platform independent standards such as the Open Grid Services Architecture (OGSA) [7] and the Common Component Architecture (CCA) [10]. Emphasis will be placed on supporting a variety of instrument and controller types including creating a small implementation that can be used with tiny wireless controllers such as the Berkeley Mote sensor package [11, 12, 13] as well as embedded PC-104- and VME-based controller systems.

The CIMA implementation will be evaluated in three settings representing a spectrum of shared instrument applications: X-ray crystallography at a synchrotron source, real-time acquisition of network performance data with embedded monitors, and small sensor network nodes based on Berkeley Mote wireless sensors. The end product will be a consistent and reusable framework for including shared instrument resources in geographically distributed Grids.

A primary challenge addressed by this research program is the lack of a generalized approach to instrument middleware that allows existing and new instruments to be integrated into Grid computing environments. Other issues to be explored include extending the accessibility of instruments to new classes of users, use of instruments by software agents, and increasing the longevity, flexibility and durability of software systems for instruments.

2.0 Scientific Instruments and Sensors

Scientific instruments and sensors[1] are crucial to scientific advancement. They provide the raw observations used to develop, verify, and falsify theories. Data from instruments typically has an extensive lifecycle, which includes corrections and calibrations, annotations with metadata to indicate its semantic meaning, and then storage in a database or filesystem on some computer system. Further stages in the lifecycle can include combining the data with other data, creating new data artifacts by performing analyses and simulations, and ultimately curation as it is transferred to new storage media. These processing transformations may be physical, numerical, and lexical; and may occur on a variety of processing units which we refer to as *nodes*.

Standards bodies, such as the Global Grid Forum (GGF), are working on every transitional data phase listed above, except for the earliest stage where the instrument provides its data as output. This is understandable; while database systems, for example, have had large scale convergence on a few API standards (such as SQL and Xquery), instruments vary widely in their architecture, construction, external interfaces, and usage modes. The Instrument Middleware Project proposes a single virtualization layer to hide this complexity, and present a relatively simple but flexible Web service interface to the rest of the data pipeline.

1. We will henceforth refer to scientific instruments and sensors collectively as instruments.

The following proposed classification schema should be considered vis-à-vis at least four exemplars: a unique large detector such as the ATLAS instrument on CERN's Large Hadron Collider, a set of detectors such as the beamline access points at a national synchrotron source such as diffractometers at Argonne's Advanced Photon Source, a few score telescopes or electron microsopes accessed and controlled remotely through computer mediation, and a large number of embedded sensors used to gather environmental data from a wide area.

2.1 Characteristics

Instruments form a diverse category of machines, with widely varying characteristics. To clarify the issues, we identify ones we consider important for building instrument systems.

2.1.1 Intrinsic Characteristics

The intrinsic characteristics of an instrument are determined by the scientific requirements. Intrinsic characteristics can not be changed by improved hardware.

Number of sources. Some instrument consist of a single data source. Others consist of myriad sources connected together. The number of sources impacts the design of the information system supporting the instrument. A large number taxes not only the scalability of the data transport, but also the management and administration interfaces. For example, manually adding metadata to specify provenance may be acceptable for single-source instruments, but not for one consisting of hundreds of individual sensors. Furthermore, systems comprised of many individual sensors must provide practical means for tasks such as identifying broken sensors and upgrading firmware.

Data rate. Some instruments monitor events that occur at a very high frequency, or perhaps generate images or even video. Such instruments output data at a very high rate.

Timeliness. Some instruments output data that can be analyzed later. Others require real-time processing of data. This might be because researchers are using the instrument interactively, or because it is being used to trigger emergency responses. Instruments requiring real-time processing can impose stringent requirements on their information systems.

Value. In some instruments, the data of each individual source is extremely valuable. Others, like a massive sensor network, the loss of some readings is not catastrophic. Systems for handling valuable data may need to incorporate fault-tolerance or redundancy into their design.

2.1.2 Extrinsic Characteristics

Besides intrinsic characteristics, other factors can affect the design of information systems for instruments. These factors are determined by the hardware design, and are termed extrinsic characteristics. Extrinsic characteristics may change as the hardware changes.

Processing power. Instrument systems must perform a variety of computational tasks, ranging from participating in network protocols to numerical analysis. These tasks may

be divided among a variety of nodes. Some nodes may consist of little more than an analog-to-digital converter. Others may be supercomputers.

Memory. Each node may have a varying amounts of memory.

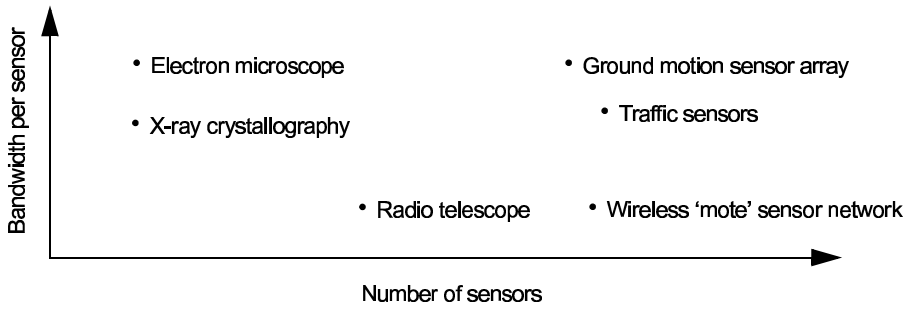
Bandwidth Availability. Some instruments have very little bandwidth available *in situ*. Others have ample. Bandwidth to the instrument interacts with the intrinsic data rate required by the instrument to affect protocol design. In situations where there is little bandwidth headroom, communication protocols must accommodate the required data rate, perhaps by techniques such as compression, which will in turn require more processing power.

Uniqueness of Design. To complicate the picture even further, many specialized instrument resources are of unique construction either by virtue of intrinsic design, or variability in product characteristics between product generations or among vendors if composed of off-the-shelf components. Some instruments are in effect never finished. Commonality between instruments simplifies system design, but may be difficult to achieve.

2.1.3 Instrument Space

We take two of the intrinsic characteristics, number of sensors and data rate, and combine them into a scalability space as shown in Figure 1. This illustrates a space of remote instrumentation based on number of sensors per instrument and bandwidth required per sensor. Three groups are shown: those with a few sensors, those with tens of sensors, and those with hundreds or more. This space serves to illustrate the range. These three groups span several orders of magnitude in numbers of sensors and duty cycle.

FIGURE 1. Space of typical remotely accessed instruments.



2.2 Operational Modes

Instruments may be operated in a number of different modes.

Collection. In this mode, the information flow is strictly from the instrument.

Collection with maintenance control. In this mode, information flow is primarily from the instrument. However, some control can be asserted on the instrument for tasks such as calibration and installing new firmware.

Person-in-the-loop. In this mode, a scientist evaluates the data during collection. Any adjustments or control operations are communicated to a technician controlling the instrument.

Remote Control. In this mode, the scientist is in direct control of the instrument. This requires fail-safe systems to prevent damage to the instrument or injury to bystanders.

3.0 Developments in Information Technology

Though technology has been applied to information well before the digital revolution, it is only recently that we have considered information and its embodied knowledge as artifacts worthy of independent study. In this section we look at some recent developments in various fields that help manage the consumption and production of information.

3.1 Grid Computing

The Grid is a grand plan for the future of scientific computation. The term Grid comes from an analogy to the electrical power grid, and Grid computation likens computer resources to electricity. Computational resources should be dependable, consistent, and ubiquitous. The user of the computational power should be unaware of its exact source. Like the electrical power grid, the Grid requires significant infrastructure, both software and hardware. On the Grid, resource management and scheduling cannot be accomplished by logging-on to the desired machine and checking to see if it is idle. Reliable and pervasive services must be in place to autonomously locate computational resources; retrieve data from disparate locations; perform the computation; and return the results to the user. All of this should take place without the user being aware of the details of geographical locations.

Grid computing has brought coherence to much of the development effort for large-scale computing projects, and has enhanced the ability to focus distributed computing and storage resources on a single large-scale application.

3.2 Web Services

The Web services community has not agreed upon a precise definition of a Web service. However, for the purposes of this paper we will use that given in the WS Arch[51]

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

The Web services community tends to emphasize openness, interoperability, and simplicity over performance and conciseness. Also Web service standards focus on network interoperability as opposed to code portability.[1]

The ideas behind Web services are arguably no different from those of many other distributed computing efforts. However, previous efforts have generally been aimed at smaller scales, or have been research projects. These efforts have tended to neglect the distateful ugliness of real-world, large-scale distributed systems, and so have found limited adoption. Web services is an attempt to deal with the real world, blemishes and all, and have a very active community of researchers and developers who are committed to applications.

Experience with the Web has suggested that interaction based on Representational State Transfer (REST) [22] is more interoperable, extensible, and scalable. The REST style uses few actions (such as the HTTP actions PUT and GET), but many resources on which those actions can operate (the URLs). The result is systems that are highly-connected, yet loosely-coupled.

3.2.1 Web Service Standards

The interfaces are usually described in Web Services Definition Language [16] (WSDL), which is fundamentally message-based but can be interpreted in terms of remote procedure calls or document exchanges. WSDL is a World Wide Web Consortium [17] (W3C) standards effort supported primarily by IBM and Microsoft.

The service interface is platform and language neutral and messages can be delivered by any protocol for which a WSDL binding has been defined, such as SOAP over HTTP or SMTP. Services are defined as collections of *ports* or addresses implementing the service, each with an abstract definition, the *port type,* and a concrete definition, the *binding*. Each port is further defined in terms of *operations*, which form the method signatures for the exchange of messages at the ports. Bindings for each port define which protocols can be used to deliver messages to that port.

Any number of protocols can be used to deliver messages to the Web Service but SOAP has become the de facto least common denominator for bootstrapping other protocols which may be necessary for performance, reliability, or security reasons. The application can be written in almost any language and SOAP bindings currently exist for Java, Perl, Python, C++, C#, Fortran and Visual Basic. SOAP can use a variety of transports including HTTP, FTP, SMTP, POP3, raw TCP, and instant messaging protocols such as Jabber [18].

In addition to the description, packaging, and transport elements described above, the Web Services effort also includes other W3C projects[19] that provide service discovery (UDDI and WS-Inspection); identity, security and encryption (XKMS, SAML, XML-DSig, XML-Enc); and workflow (BPEL4WS [20]).

1. Code portability is the ability to run or compile code written for one platform, such as Microsoft Windows, on a different platform, such as Linux.

3.2.2 Open Grid Services Architecture

The Web Services model is attractive for scientific computing where issues of contract complexity (requirements for using a service) are minimal. Recently, a proposal was made to integrate the Web Services model into Globus [7]. This model suggests the deployment of Grid services as Web Services and that applications adopt a Web Services approach to locating and interacting with the Grid. As a preliminary step in realizing a Web Services-based Grid the Globus group has rewritten the underlying Globus Toolkit to conform to an interface specification more useful for casting applications as Web Services. The Open Grid Services Infrastructure Reference Implementation that implements the Open Grid Services Infrastructure (OGSI) [21] specification provides secure SOAP messaging based on Globus security. Service lookup at the application level is available by Universal Description, Discovery and Integration (UDDI), with lower level Globus functions still relying on Globus MDS-2 LDAP calls.

The OGSI has also adopted some REST tenets in the form of Service Data Elements (SDEs). The elements provide a comprehensive way to query services for information without requiring a large interface.

3.3 Components

A fundamental approach to any difficult problem is to find abstractions that can be decomposed into less difficult subproblems. In mathematics, for example, a complex theorem is built up from a series of lemmas. In computer science, divide-and-conquer is a fundamental paradigm for algorithm development. The assumption here is that complexity can be localized. This technique has been applied to computer software in a number of forms, including structured programming and object-oriented programming (OOP).

Programming language objects are not exposed to the end-user, however, and so software components were introduced to apply some of principles of OOP to entities that can be manipulated by the end-user. Components do not replace objects, but are an application of the ideas behind OOP, such as abstraction and encapsulation, to the broader issues mentioned above. No agreed upon definition of software components exists, so rather than attempt a definition, we will list some of the characteristics we include in the notion of a software component:

Independently deployable. A component can depend on another component for correct operation, but it still must be possible to install and upgrade it individually. Furthermore, this deployment should not require programming skills.

Reusable by third-parties. Components can be reused in different applications by the end user. Such interchangability is crucial to making them effective at coping with rapid change.

Connectable by third-parties. Components can be connected to one another in a fashion similar to stereo components, or hardware modules. This allows the end user to solve completely new problems from existing components.

Large granularity. A component generally encapsulates more functionality than an object. Most components will be implemented internally with many objects.

Inherently Distributed. Remote operation is an inherent characteristic of components. This means that local operation is a special case of normal operation, rather than vice versa. This characteristic forces a degree of encapsulation that often makes components easier to use even in the local case.

3.4 Knowledge Engineering

As pace of interactions increases, interoperability becomes ever more difficult to achieve. Computer systems are still brittle. Seemingly minor changes can bring large systems to a grinding halt. Frustratingly, often the changed information is semantically equivalent, but merely with minor syntactic variations.

The area of knowledge management and engineering seeks to address these problems by providing techniques for managing "knowledge" about the world. By providing languages for specifying ontologies, and systems for manipulating them, semantic information can be inferred.

Ontologies provide static descriptions of a domain. Problem-solving methods (PSMs) are a knowledge engineering artifact for describing algorithms generically. A PSM can be used reused with different ontologies.

4.0 The Cyberinfrastructure Revolution

Information is the driver of scientic inquiry and insight. From observations and measurements, we validate, refine, and formulate models and theories. From the literature we extend previous work, and cross-fertilize between disciplines to discover new ways of applying existing ideas. From personal communications we discuss, argue, and brainstorm, fomenting new paradigms to challenge existing modes of thought.

Scientific progress results from the gradual accumulation of knowledge, enabled by the exchange of information. In prehistoric times, information technology consisted of our human oral, aural, and mnemonic abilities. Our sum knowledge was limited by what we could collectively remember. The invention of writing bypassed our limited memories, and increased our I/O bandwidth through our large visual cortex. For the first time, we could accumulate knowledge in a permanent form, greatly amplifying what we could accomplish otherwise. The invention of the printing press caused another exponential leap in knowledge. If we view the dissemination of information as data traveling through a graph, the printing press allowed the branching factor at each nexus to be much greater than what could be accomplished through manual transcription.

Each of these developments profoundly altered the frequency, depth, and pervasiveness of human discourse, which eventually led to disruptive changes to society and the nature of scientific inquiry. We believe that we are now undergoing a similar revolution, brought on by the enormous increase in processing, communication, and storage capabilities of the last few decades. By speeding the acquisition, processing, and analysis of data, information technology also speeds the intellectual processes crucial to scientific progress. This acceleration occurs at all levels, ranging from the cognitive process of the

individual scientist to the social interactions between disciplines. The result is a qualitative change in the way scientists work.

In the information network model mentioned above, information technology not only increases the branching factor, but also the frequency of mass exchanges. With electronic publishing in the form of such things as digital libraries and even weblogs, no longer does a two-way mass exchange take a significant amount of time and effort. Furthermore, search engines such as Google effectively increase the fan-in factor as well. Search engines allow us to quickly process a large amount of incoming data. In the future software agents may allow us take this to another level.

The advances in electro-optical technologies compose the foundation for these developments. Effective utilization of these advances, however, requires a layer of hardware, software, institutions, and personnel, which has been termed *cyberinfrastructure*. This layer bridges the gap from the switching capabilities of the base technologies to the problem-specific applications of each discipline. It transforms the raw functionality of integrated electro-optics into services and abstractions directly usable by domain specialists.

Cyberinfrastructure comprises a diverse set of elements. Among them are integrated services for knowledge management, observation and measurement, visualization, interaction, and collaboration.

4.1 Bringing Instruments Online

Instruments and sensors form a crucial part of cyberinfrastucture. They are the source of verification, and are the primary of the experimental and observational. Realizing the full benefits of cyberinfrastructure depends critically on complex, dynamic interactions between producers and consumers of scientific data.

4.1.1 A New Approach

Currently researchers treat observed data as primarily a computational commodity or raw material, to be turned into knowledge through reduction, fusion, and analysis. As such, the application of information technology has been focused on the latter part of this value-add chain, in the reduction and analysis phase, which can be performed off-line with respect to the instrument that collected the data and outside any possible interaction between the analyzer of the data and the instrument itself. Big data projects organized around instruments such as detectors at the Large Hadron Collider have yielded useful application testbeds such as GriPhyN [14] and iVDGL [15], that focus on data somewhat after it has been acquired from the instrument. These projects by design largely ignore the instrument from whence the data comes.

This single-minded concept of data as distinct from its source leaves a critical hole in the Grid that we believe must be filled to realize the full benefits of the cyberinfrastructure revolution. For some contexts, such as real-time analysis, the data and the instrument are so closely coupled that any separation simply interposes additional latencies and barriers to effective utilization. For other contexts, a closer association of the data with the control of the instrument will foster a tighter feedback loop between the instrument and the domain experts. Among other benefits, this will allow experts to analyze

and adjust the data gathering process itself while it is still in the critical early stages, which will avoid collecting poor quality data, saving time and resources.

4.1.2 A Common Instrument Middleware

Unfortunately, bringing instruments online is not as simple as connecting them to a network. For instruments to participate, they must be represented via portions of the cyberinfrastructure layer. These portions, known as instrument middleware, allow instruments to be active partiticipants in the Grid by allowing to interact with Grid services.

As part of the cyberinfrastructure, NSF envisions thousands of online instruments along with thousands of collaboratories [55]. Distinct middleware could be implemented ad hoc for each instrument. However, this would be wastefully redundant and, more importantly, risks balkanization, which would lead to numerous compatibility and interoperability problems. The benefits offered by cyberinfrastructure would be severely limited.

These concerns suggest the development of a common middleware architecture, utilizing modern approaches to distributed computing. Differences between disciplines would be hidden where possible by encapsulation within adaptation layers, or ameliorated through the use of knowledge engineering, REST, etc. To maximize reuse, we believe that this layer should be placed as close as possible to the instrument, which would shield the bulk of the cyberinfrastructure from changes to the instrument.

The recent developments in information technology described in Section 3.0 are directly applicable to a common instrument middleware.

4.2 Trends and Benefits

A confluence of trends are energizing the cyberinfrastructure revolution. These trends are both opportunities and pitfalls. In each of these trends, online instruments are a significant contributor to realizing the benefits.

Rapid growth in hardware performance. Researchers today have much greater access to powerful hardware. Computations that used to be performed at large centers can now be performed on desktop workstations. A recent NSF survey reported that 74% of scientists perform analysis on machine different from that which generated the data [57].

This decentralization of computer power allows scientists to directly access instrumnets from their desktop, but fully utilizing it requires standards to insure interoperability.

The increasing power of desktops also means that small groups can realistically benefit from direct access to instruments. They must be able to use the generated data, however.

Rapid growth in digital information and data. As more information comes online, the benefits of bringing instruments online are magnified synergistically. Having the information and the instrument online means that the analysis and comparison process can be performed more rapidly, which allows the scientist to be more selective of the quality of data that she collects.

Furthermore, a common instrument middleware will help in the automatic assimilation and cataloging of the huge amounts of collected data. A recent NSF survey found that 50% of scientists reported that ineffective data cataloging and/or difficulty locating required data was the primary impediment to the use of digital or federated data repositories. An additional 26% reported inadequate or missing information on quality control as the primary impediment [58].

To resolve these issues and prevent data archives from becoming data mortuaries requires accurate, reliable metadata and data format standards. The most logical place to bind the data to its metadata is at its source, the instrument itself, and doing so requires instrument middleware.

Another important use of good metadata is data mining.

Large shared resources. As many of our scientific disciplines mature, they place ever greater demands on our ability to measure the natural world. Further breakthroughs require measurements at the extremes of scale, resolution, and energy. This requires ever more expensive instruments. These high-value instruments impact instrument middleware in two aspects.

The first is that many of these instruments are unique, or one of only a few. They may in fact be essentially never "finished," undergoing constant revision to meet the particular dictates of each experiment. This causes continual re-invention of the controlling software and frequently results in redesign of data output formats from the instrument. As a consequence, software at later stages in the reduction and analysis pipeline is potentially subject to rewriting when any changes are made to the instrument, even if changes are remedial with no new capabilities being added.

The other aspect is an increased importance on collaboration, which is explored separately. High-value resources inherently are collaborative, since only large groups can afford them

Collaboration. As our ability to investigate ever more complex systems increases, we are finding that the behavior of these systems often involve complex interactions that span multiple areas of expertise. This requires interdisciplinary, often international, collaboration.

This interdisciplinary collaboration often involves using data from other fields. In fact, at a recent meeting of the environmental research and education community it was reported that some scientists are spending 75% of their time finding and converting data from other fields [56]. This is obviously an enormous waste of precious human capital.

Format differences will always exist, for historical reasons if nothing else. But by developing the proper knowledge management, formats can converted automatically when needed.

As an example of how a common instrument middleware can greatly increase the effectiveness of collaboration, consider the National Virtual Observatory. Currently astronomers can query the database for images of a specific wavelength. The quality of astronomical images can be affected by atmospheric conditions and seismic conditions,

among others. By correlating the image metadata with readings from weather, pollution, and seismic monitors, additional useful information can be gleaned from the imagery.

Real-time usage. Current scientific research requires increasingly detailed knowledge about our natural world. Often, acquiring this knowledge requires real-time interaction. For example, numerical weather models perform better when they can control meteorological sensors in real-time feedback loops to optimize data collection. Developing the cyberinfrastructure for such control benefits significantly from a common instrument middleware.

***In silico* simulation.** As computational power increases, computer simulations are being used increasingly in tandem with observed data. The comparative analysis of the simulated results with instrument measurements is greatly facilitated if middleware is available to bridge, automate, and verify. For example, changing a parameter in the simulation may require that observed data be renormalized. If the sensor data is well-described with an ontology, this renormalization can be automated.

Education. As our economy matures, we are seeing an accelerated shift to an information-based society. Education becomes increasingly paramount. Instrument middleware can make direct sensor data much more available to primary and secondary schools, which ordinarily might not have the budget to access specialized instruments nor the specialists to interpret complicated, legacy data formats.

Accessibility. Computer technology has been shown to be of great benefit in allowing the physically challenged to contribute to science. Some tasks, however, such as converting speech-to-text still require processing power beyond that of the average desktop. For these tasks, one can envision a Grid service running on a shared cluster. In this context, the microphone then becomes an instrument. Some desktops might have sufficient power to perform some of the computations locally, thus improving response-time and reducing the load on the shared cluster. The end result is that different classes of similar instruments need to interact with a Grid service. To facilitate interoperability and sharing, an instrument middleware would be beneficial.

Diversity. Minority serving institutions have historically lagged in the adoption of computer technology. Network connectivity has emerged as one of the most significant impediments. High bandwidth instrument data can always be filtered and aggregated to reduce bandwidth requirements, but such processing would have to be separately implemented for each data format. A common instrument middleware can facilitate reuse and automation by providing a common base for describing data in a manner than generic problem solving methods (from knowledge engineering) can then act upon.

5.0 Meeting the Challenge

The preceding section has described how a common instrument middleware infrastructure can benefit the conduct scientific research and spread its impact to segments of society that have had limited access to scientific instruments. In this section we outline how we might actually create such a common middleware.

5.1 Requirements

To meet the challenge, the instrument middleware will need to meet the following requirements:

- Support processing of data in real-time across a variety of timescales.
- Support different rates of data production under varying available bandwidth. This will probably require the ability to integrate multiple protocols within one system.
- Provide end-to-end reliability and fault tolerance.
- Support large numbers of sensors.
- Support integrated representations of instruments comprising disparate sensor types.
- Represent and support the different operational modes described in Section 2.2.
- Support real-time fusion of heterogeneous information from many sources to create one data stream.
- Support for pipelined data reduction and analysis.
- Provide functional transparency. Each function of the instrument must be completely and accurately accessible.

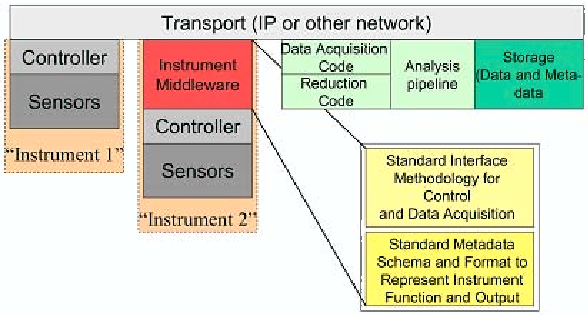
5.2 The Common Instrument Middleware Architecture

To realize these requirements, we propose a Common Instrument Middleware Architecture (CIMA) based on concepts from Grid computing and Web services. CIMA has two main aspects.

1. An instrument description for each instrument that is sufficient to fully use the instrument.
2. A component framework for building instrument systems that can be used to build reusable, interoperable systems.

Figure 2 illustrates how these aspects fit together in a typical instrument system. On the left side of Figure 2 are the sensors, controller, and data acquisition hardware that form a typical instrument. Data flows from this instrument over a transport layer, such as an IP network, to reduction and analysis codes hosted elsewhere on the network. Eventually the processed and analyzed data are archived on some medium, along with any relevant metadata. Clearly parts of this system will require complete knowledge of the instrument and the data it produces built-in. The salient feature of this process is that the data acquisition and reduction codes, and perhaps all components of the analysis pipeline must have complete knowledge of the instrument and the data it produces explicitly built into them.

FIGURE 2. Instrument, middleware, and the data reduction/analysis pipeline. The middleware component consists of both the interface and metadata about the instrument.



Additional functionality that could be built into instruments at the middleware layer include authorization and authentication, transport layer security, automatically generated metadata, access to diagnostic software, and functional simulation for backward compatibility with older hardware.

Our approach to developing instrument middleware for the Grid leverages existing and emerging standards in Web Services, in particular the Open Grid Services Architecture (OGSA), the Globus adaptation of Web Services. In addition we will leverage current work in component systems for scientific programming and existing efforts to develop a component architecture for data acquisition and instrument control in X-ray crystallography.

Web services provide a diverse set of standard protocols with which to build distributed systems. Though our requirements could arguably be met with CORBA or some other technology, much of the current work on cyberinfrastructure is based on Web services. Therefore, as much as possible, we base our work on Web service standards.

5.2.1 Instrument Description

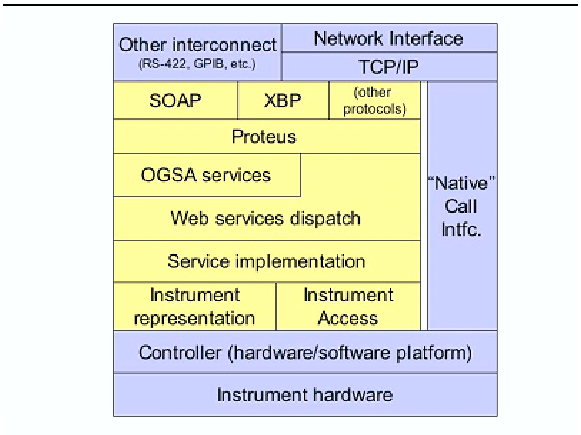
The benefits of a common instrument middleware rely critically on encapsulating each instrument within a instrument abstraction layer, which presents an device-independent interface to the rest of the system. This commonality between instruments facilitates code reuse between different instruments, and also eliminates the need to rewrite such code when the underlying instrument and controller are redesigned.

Fundamentally, abstraction is a mapping from a set of interfaces to a single common interface. The mapping must hide unimportant distinctions between interfaces, while preserving the important ones. Thus, if the common interface is too small, important functionality will be lost. On the other hand, if the single interface is simply the union of all the interfaces, then little will have been gained.

When specified appropriately, the abstraction layer will mitigate the dependence of acquisition and analysis software on the details of the hardware design, and thus facilitate the integration of instruments into the Grid infrastructure. Instrument users will transparently reuse existing software when instruments are modified or upgraded. By opening up their products, instrument manufacturers can take advantage of a broader range of control application and data sinks (data fusion and analysis software). Providing a Grid-aware middleware will facilitate interaction between research groups working at different places in the value chain of scientific inquiry.

Figure 3 illustrates the components of the Common Instrument Middleware Architecture (CIMA) (yellow/light background) and how they articulate with existing instrument controller and interface components (blue/darker background). In the initial implementation the instrument's existing functionality is preserved as completely as possible to allow performance comparisons between using the instrument through the CIMA "stack" and through "native" calls.

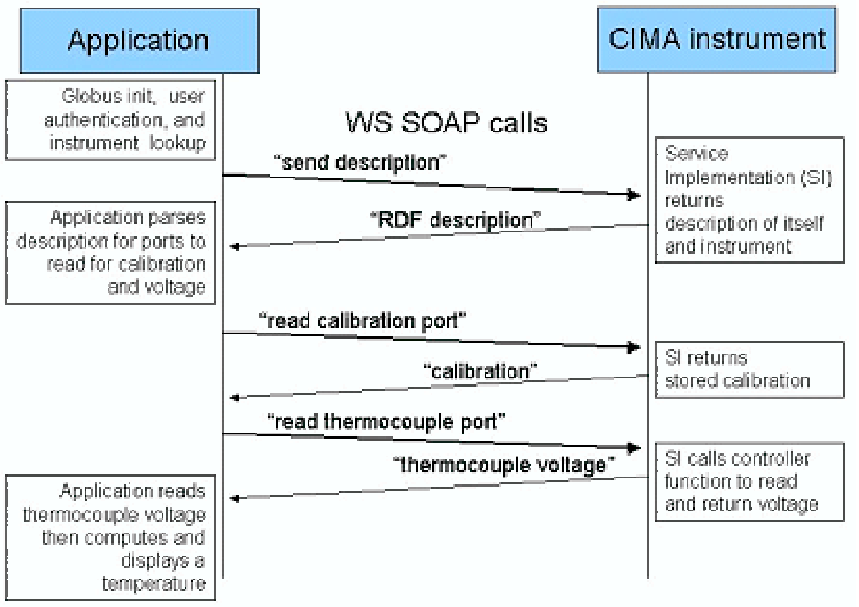
FIGURE 3. Instrument model implementation.



A central design requirement is that CIMA applications must be able to develop an operational model of the instrument from a minimum of external knowledge. This will reduce the burden of managing and administering a large variety of instruments. Each function of the instrument must be completely and accurately represented by the Grid interfaces provided by the middleware. The implementation of this requirement is straightforward: as a Grid service conforming to the OGSI specification, the instrument will implement the GridService port type (along the lines of the IUnknown interface in Microsoft's Distributed Component Object Model [31]) which provides a findServiceData operation. Given the appropriate query expression, this method will return a description of the instrument as RDF [32], which can be parsed and used by the application to build WSDL messages to interact with control and data ports and to obtain calibration information. This is illustrated in Figure 4 below.

FIGURE 4.

Example bootstrap and data acquisition exchanges between an application and a CIMA-enabled instrument. Initial calls can be made with SOAP, later ones with other, more efficient protocols supported by the Proteus layer. This example shows RPC-style interactions, with applications requesting data, but in other scenarios the instrument may send data asynchronously as an event..



For example, a thermocouple is used to measure temperature, but through a voltage measurement and a calibration curve. This calibration, and hence the "measured" temperature, may be derived from standard tables for the type of thermocouple used, or through a carefully performed manual calibration against a primary standard. The description of such a thermocouple in RDF might look something like this:

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<rdf:RDF xmlns:CIMA="&CIMA;" xmlns:rdf="&rdf;"
xmlns:rdfs="&rdfs;">
<CIMA:Description
        CIMA:my_key="CIMA:A7DB-2287-E7DB-6745"
        CIMA:my_name="TC7443"/>
<!-- either a URI and namespace pointing to the service
definition (CIMA:uri and uri_namespace) or the text of
the service definition as a CDATA block
(CIMA:binding_template) can be used -->
<CIMA:WSDL
```

```
         CIMA:uri="http://cima.kaplab.iu.edu/wsdl/
TC_service.wsdl"
         CIMA:uri_namespace=http://cima.kaplab.iu.edu/
ns/TC_service>
         <CIMA:binding_template xml:space='preserve'><
CDATA BLOCK OF WSDL >
       </CIMA:binding_template>
</CIMA:WSDL>
<CIMA:Characteristics CIMA:idesc="K-type thermocouple
on 16 bit A/D PC104 module"/>
<CIMA:Calibration
         CIMA:Port_name="termocouple_reading"
         CIMA:type="vector">
       <CIMA:value>30, 1.179, 35, 1.397, 40, 1.606, 45,
1.813, 50, 2.020, 55, 2.225
        </CIMA:value>
</CIMA:Calibration>
<CIMA:port
         CIMA:data_format="u16"
         CIMA:name="thermocouple_reading"
         CIMA:signal=voltage
         CIMA:port_direction="OUT"
         CIMA:port_type=":INTEGER" />
</rdf:RDF>
```

Listing 1. RDF description of an instrument in a prototype CIMA schema.

The schema consists of four main parts: a description that identifies the specific instrument platform, information about how to use the service (a WSDL document for the service or a URI pointer to it), calibration information for this instrument, and the characteristics of data produced by each channel. The latter may seem redundant to the WSDL document but it provides a place to put semantic information about the instrument's control and data channels and additional type information which may be necessary if WSDL types are too opaque for application code to parse.

After the application has queried the instrument for a description, it can parse the description to extract an operational model of the underlying instrument and information about how to interact with the service implementation's ports. In this example the application finds that there is one data (OUT) port that provides a voltage signal as a 16 bit unsigned integer. Furthermore the application can consult the CIMA RDF Schema [33] to determine what voltage means and how best to handle this data in a computational or user interface context.

5.2.2 Component Framework

The Grid will be a complex system composed of interoperable, distributed parts. To allow maintenance and changes to the Grid without interrupting operation, the parts will need to be independently deployable. These requirements suggest components as the paradigm for building the Grid.

Our component framework will require capabilities in a number of areas.

Communication. WSDL messages are received from an external application for a particular port, the parameters are unmarshaled and code associated with the requested operation is called with the given parameters. Though we will use SOAP as the common protocol, it is not appropriate for every situation. Performance is modest [23] and may be limiting, so middleware must be able to use other protocols. Also, it may be the case that our middleware is connected to the instrument via a wire protocol of some kind. In which case, we need some standard way of incorporating this vendor-specific protocol into our acquisition software without making the acquisition software vendor-specific. To handle these two needs, SOAP will be used for the initial handshaking, but Proteus [25] will be used to incorporate other protocols that may be more appropriate for the particular instrument scenario.

Security. Since a primary design goal of CIMA is to integrate instruments into a Grid of computing and storage resources, authentication and authorization is of some importance. We will track OGSI working group activities closely for emerging authentication strategies. Initially we will provide an external validation service for each instrument that the instrument can use to authenticate users based on Globus proxy certificates. In addition to authenticating potential users the validation service will provide authorization information and a limited life key that applications can use to make future calls to the instrument. All communications between the application, validation service and instrument could be encrypted via SSL or IPSec for transport level security.

Ultimately the security model for an instrument depends heavily on how the instrument is used, and a significant aspect of this project is to develop use models. Both authorization and authentication are problematic and in some cases multiple independent schemes may be required at the same time. Allocation of an instrument's resources may imply time dependent changes in the authorization scheme, as, for example, when a specific group of researchers and no others are using a telescope, and within this group specific individuals may be allowed to control the telecope while others may only observe.

6.0 Related Work

A Grid-based Collaboratory for Macromolecular X-Ray Crystallography Using Synchrotron Light Sources. DOE NGI program. The XPort [27] project targeted revolutionary improvements in telepresence for major scientific instrumentation systems. The goal was to exploit a combination of advanced networking technology, Grid services, and remote instrumentation technologies to achieve interactive “better-than-being-there” capabilities for remote experiment planning, instrument operation, data reconstruction, and data analysis. Some of these capabilities were deployed and demonstrated at major DOE facilities, including the Advanced Photon Source and the Advanced Light Source. Building on work from the Globus group and the DOE 2000 Common Component Architecture Forum, the project addressed several issues related to NGI network-based instrumentation including high-speed data collection, reduction, storage and visualization, and real-time instrument monitoring for the acquisition of X-ray crystallographic data from the DND-CAT beamline sector at the Advanced Photon Source and for a similar sealed-beam X-ray diffraction system at the Indiana University Molecular Structure Center. Each component in the system was implemented as a CCAT [5] component written in C, Python or Java. CCD frame data flows from the detector to a cache at the beamline, then to a network archive (currently HPSS servers at IU or ANL)

where data reduction and visualization applications that may be distributed across a computational Grid can retrieve individual frames and intermediate data sets. Each component is a CCAT instance running on a machine appropriate for its function. Currently these components include data acquisition, local cache management and instrument monitoring for Bruker goniostats and CCD detectors. We also experimented with the then-new SOAP protocol as a run-time system to replace the relatively heavy-weight Globus mechanisms for services such as instantiating components, connecting them, sending control signals among them, and more generally for data transfers which do not require high-speed, low latency communications

Distributed real-time systems using Tao CORBA [35]. Considerable work has been done to make CORBA suitable for real-time applications. The leading effort is the TAO object broker, a CORBA implementation designed to support avionics applications. As such TAO emphasizes bounded (low) latency communications and fault tolerance.

EPICS [36]. The Experimental Physics and Industrial Control System (EPICS), developed at the Accelerator Technology (AT-8) group at Los Alamos National Lab) and the Advanced Photon Source (APS) at Argonne National Lab, consists of an architecture for building scalable control systems and a collection of code and documentation comprising a software toolkit. EPICS is based on the idea of virtual channels between acquisition code and the underlying hardware. Although well designed for high data rate applications its complexity has limited acceptance and broader use.

Astronomical Instrument Markup Language (AIML) [37]. AIML is a NASA project to create an XML DTD for the HAWC airborne camera and related systems [38]. The aim was to create a representation of the control and data systems primarily as a specifications document to coordinate work between hardware and software engineering groups. AIML was also used to develop simulations of the hardware and to generate user interfaces. The vocabulary used in the AIML DTD was drawn primarily from the hardware engineering effort and included a large percentage of project-specific terminology, but the project has laid some useful foundations for developing general ontologies for instruments.

Universal Plug and Play (UPnP) [39, 40] is a Microsoft standard to allow devices to interact over an IP network using a zero configuration approach. Basics include using DHCP to acquire an address, a network registry scheme for service discovery based on pre-assigned device codes, and an on-line documentation system to allow users to map device codes to capabilities.

The Salutation Consortium [41] is a competing pervasive computing effort by Japanese gadget makers and NIST in the U.S. offering zero configuration networked device management and service discovery. Salutation code is proprietary although there is a Lite version available for experimentation and evaluation.

7.0 Case Studies

To illustrate how CIMA can be used in different domains, we present a number of case studies in this section.

7.1 X-Ray Crystallography

The class of sensor represented here is a custom design, single, large shared instrument that produces gigabytes of data per day.

Building on work already done at the Argonne Advanced Photon Source (APS) DND-CAT beamline [26] we will provide a CIMA-enabled version of the XPort [27] platform. This work is expected to initially support the efforts of two independent research teams, one located at IU Molecular Structure Center[28] and the other at the University of Sydney in Australia [52].

Researchers from both of these groups travel frequently to APS, to perform experiments and collect data at ChemMatCARS beamline [29] and DND-CAT beamline[26]. Further more, some reasearchers from IU will be utilizing the recently built MB-CAT beamline at Advanced Light Source (ALS) at Berkeley Lab [53].

Presently, the major disadvantages for researchers collecting data at remote synchrotron sites are, the travel, extra expenses and restricted accessibility for certain international users.

To address these issues we propose that synchrotron sites by adapting to CIMA and providing a small amount of assistance by beamline personnel, will be able to through Web based technologies, allow researchers to remotely perform a X-ray diffraction experiment from their "home laboratories". The researcher can have the samples to be studied, shipped to the beamline by e.g. a commerical shipping company. At the beam line the samples can be unpacked by the beamline personnel and mounted on the instrument using crystal mounting robots (or mounted manually). The entire process from unpacking, crystal mounting and data collection can be supervised by the researcher in the "home laboratory", using Web based tools and designated video/audio conferencing tools specially developed for this purpose.

A large body of data reduction and analysis codes is associated with diffraction experiments and these software and can be modified to use CIMA to interact directly with e.g. the DND-CAT and ChemMatCARS detectors. Since a significant effort has already been made in developing the web-based tools for the X-ray crystallography[27], we will also make efforts to extend the existing data evaluation and analysis tools. The analysis tools will allow rapid evaluation of the initial data from the APS by the remote collaboration team, optimizing the use of these over-subscribed resources.

Other aspects, in which adaptation to CIMA will greatly benefit the crystallographic community is that CIMA will provide a unified data access format for the raw unprocessed data .Currently there are very few options for storing raw data from X-ray diffraction experiments in a "standard" format that can later be read by a wide range of data processing software.

The variety of CCD detectors used at the various beamlines and their different output format is creating several issues for crystallographers. Usually the raw data collected at a certain beamline is processed using the CCD vendor's supported software. Once the crystallographer is back in the "home laboratory" a reprocessing of the raw data can be tedious unless a copy of the data processing software is purchased from the CCD ven-

dor. The problem is compounded by the fact that many researchers utilize multiple beamlines with different CCD detectors.

To resolve this issue, CIMA will seamlessly interface to a wide variety of makes and models of CCD detectors used at each beamline. Additionally to the CCD vendors native data output format , CIMA will be able to support a standardized "meta data" format/output format (img-CIF/CBF)[54] for each CCD detector, regardless of its make and model.

The recently operational NSF funded Analyzing and Visualizing Instrument-Drive Data (AVIDD) facility at Indiana University will be used to store and evaluate the data being generated by the CCD detectors located at the APS. Researchers at other universities will be able to monitor the experiment at APS while processing and evaluating the data being transferred to the IU's AVIDD system.

The availability of an CCD equipped instrument in the Indiana University Molecular Structure Center identical to that at ChemMatCARS will facilitate the development and testing of the specialized crystallographic analysis code that will be accessing the CIMA API from the application side. A major emphasis will be put on the development of software that will allow the remote collaboration team to convert the CCD frame images into a representation of reciprocal space that can be viewed using volume rendering techniques. This will allow the researchers to rapidly examine the initial data to insure that the mosaic character, twinning, and splitting can be handled by the data processing software.

7.2 Embedded Network Monitoring

End-to-end performance analysis for distributed applications and for facility-level monitoring of Grids (*e.g.* the iVDGL Grid Operations Center [30]) is increasingly dependent on measuring devices embedded in the network*e.g.* Network Weather Service nodes (modest data flows, real-time acquisition, moderate number of data sources). These monitoring nodes provide real-time network performance data needed to detect and correct network problems and represent a class of sensors used in groups of 10 to 100 that produce moderate data flows. Network embedded nodes are being developed for application performance monitoring and analysis under the NSF grant STI-0129592 Network Management Tools for End-to-end Performance Measurement. As a part of this project the interface for these nodes will be modified to include a CIMA interface and monitoring software will be modified to evaluate CIMA in this context.

7.3 Wireless Networked Sensors

The extreme test for CIMA as a unifying technology for ubiquitous computing is embedding it in wireless networked sensors such as the Berkeley Mote. (e.g. Motes, small data flows, real-time acquisition, large number of data sources). The current mote controller and radio transmitter implementation is approximately 1 cm^2, with a stated aim of reaching sub-mm^2 sizes co-fabricated with MEMS-based sensors and actuators. This class of sensor explores the use of CIMA in ubiquitous computing applications.

8.0 References

[1] C. Kesselman and I. Foster, *The GRID: Blueprint for a New Computing Infrastructure*: Morgan Kaufmann, 1999.

[2] C. Kesselman and I. Foster, "Globus Project," 1998, http://www.globus.org.

[3] A. Grimshaw, "Legion," 2000, http://www.cs.virginia.edu/~legion.

[4] J. Villacis, M. Govindaraju, D. Stern, A. Whitaker, F. Breg, P. Deuskar, B. Temko, D. Gannon, and R. Bramley, "CAT: A high performance distributed component architecture toolkit for the grid," presented at High Performance Distributed Computing Conference, 1999.

[5] R. Bramley, K. Chiu, S. Diwan, D. Gannon, M. Govindaraju, N. Mukhi, B. Temko, and M. Yechuri, "A component based services architecture for building distributed applications," presented at High Performance Distributed Computing Conference, 2000.

[6] M. Champion, C. Ferris, E. Newcomer, and D. Orchard, "Web Services Architecture," W3C Consortium, 2002, http://www.w3.org/TR/ws-arch/.

[7] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration (Draft)," 2002, http://www.globus.org/research/papers/ogsa.pdf.

[8] I. Foster, D. Gannon, J. Nick, and W. Johnston, "Open Grid Services Architecture: A Roadmap," 2003, http://www.ggf.org/ogsa-wg/ogsa_roadmap.0.4.pdf.

[9] H. Haas and D. Orchard, "Web Services Architecture Usage Scenarios," W3C Consortium, 2002, http://www.w3c.org/TR/ws-arch-scenarios/.

[10] "Common Component Architecture Forum," 2000, http://www.cca-forum.org/.

[11] A. Woo, "Mote Documentation and Development Information," 2000, http://www.cs.berkeley.edu/~awoo/smartdust/.

[12] K. S. J. Pister, "SMART DUST: Autonomous sensing and communication in a cubic millimeter," 2003, http://robotics.eecs.berkeley.edu/~pister/SmartDust/.

[13] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Mobile Networking for Smart Dust," presented at Proc. of ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA, 1999.

[14] "GriPhyN - Grid Physics Network," 2003, http://www.griphyn.org/index.php.

[15] P. Avery and I. Foster, "The International Virtual Data Grid Laboratory (iVDGL)," 2003, http://www.ivdgl.org.

[16] R. Chinnici, M. Gudgin, J.-J. Moreau, and S. Weerawarana, "Web Services Description Language (WSDL) Version 1.2," World Wide Web Consortium, 2002, http://www.w3.org/TR/wsdl12.

[17] "World Wide Web Consortium," http://www.w3.org/.

[18] "Jabber Software Foundation," Jabber, Inc., 2002, http://www.jabber.org/.

[19] "World Wide Web Consortium," 2003, http://www.w3c.org.

[20] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, "Business Process Execution Language for Web Services, Version 1.0," IBM, 2003, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.

[21] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, D. Snelling, and P. Vanderbilt, "Open Grid Services Infrastructure (OGSI) (draft)," Global Grid Forum, 2003, http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-23_2003-02-17.pdf.

[22] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California at Irvine, 2000,

[23] K. Chiu, M. Govindaraju, and R. Bramley, "Investigating the Limits of SOAP Performance for Scientific Computing," presented at Proceedings of the Eleventh IEEE International Symposium on High Performance Distributed Computing (HPDC'02), 2002.

[24] A. D. Birrell and B. J. Nelson, "Implementing Remote Procedure Calls," *ACM Transactions on Computer Systems*, vol. 2, pp. 39-59, 1984.

[25] K. Chiu, M. Govindaraju, and D. Gannon, "The Proteus Multiprotocol Library," presented at Proceedings of the 2002 Conference on Supercomputing, 2002.

[26] "DuPont - Northwestern - Dow Collaborative Access Team," 2003, http://tomato.dnd.aps.anl.gov/index.shtml.

[27] D. F. McMullen and R. Bramley, "The XPort Project," 2000, http://www.cs.indiana.edu/ngi.

[28] "Service Crystallography at Advanced Photon Sources," Indiana University Molecular Structure Center, 2003, http://www.iumsc.indiana.edu/synchrotron/scraps.html.

[29] A. S. R. Program, "Specialist Crystallography at the Advanced Photon Source," 2001, http://www.ansto.gov.au/natfac/scraps.html.

[30] "iGOC - iVDGL Grid Operations Center," International Virtual Data Grid Laboratory, 2003, http://igoc.iu.edu/.

[31] "DCOM," Microsoft, 1998, http://www.microsoft.com/com/tech/DCOM.asp.

[32] "Resource Description Framework (RDF)," World Wide Web Consortium, 2003, http://www.w3.org/RDF/.

[33] D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," World Wide Web Consortium, 2003, http://www.w3.org/TR/rdf-schema/.

[34] J. Cowan and R. Tobin, "XML Information Set," World Wide Web Consortium, 2001, http://www.w3.org/TR/xml-infoset/.

[35] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design of the TAO Real-Time Object Request Broker," *Computer Communications, Elsivier Science*, vol. 21, 1998.

[36] S. A. Lewis, "Overview of the Experimental Physics and Industrial Control System: EPICS," 2000: Lawrence Berkeley National Laboratory, 2000, http://csg.lbl.gov/EPICS/OverView.pdf.

[37] J. Breed, "Astronomical Instrument Markup Language," NASA GSFC, 2000, http://pioneer.gsfc.nasa.gov/public/aiml/.

[38] J. Breed, "High Resolution Airborne Wideband Camera," NASA GSFC, 2000, http://pioneer.gsfc.nasa.gov/public/hawc/.

[39] "Universal Plug and Play Device Architecture," 2000: Microsoft Corporation, 2000, http://www.upnp.org/download/UPnPDA10_20000613.htm.

[40] "Understanding Universal Plug and Play: A White Paper," 2000: Microsoft Corporation, 2000, http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc.

[41] "The Salutation Consortium," 2003, http://www.salutation.org/.

[42] "Indiana University Molecular Structure Center," 2003, http://www.iumsc.indiana.edu/.

[43] "Extreme! Computing," IU Extreme Computing Group, 2003, http://www.extreme.indiana.edu/.

[44] "PRAGMA, the Pacific Rim Applications and Grid Middleware Assembly," San Diego Supercomputer Center, 2003, http://pragma.sdsc.edu/.

[45] "Global Grid Forum," 2003, http://www.ggf.org.

[46] U. S. Navy, "Distributed Engineering Plant," 2000: US Navy, Naval Surface Warfare Center, 2000, http://www.nswc.navy.mil/dep/.

[47] D. Pearson, K. Adams, D. Gannon, D. McMullen, M. McRobbie, A. Robel, S. Wallace, and J. Williams, "TransPAC: A High Performance Network Connection for Research and Education between the vBNS and the Asia-Pacific Advanced Network (APAN)," presented at Proceedings of The International Workshop on Asia-Pacific Area Advanced Research Information Sharing Technology, Internet Workshop '98 (IWS'98), Tsukuba, Ibaraki, Japan, 1998.

[48] "TransPAC," 2003.

[49] "iGrid 2000: Empowering Global Research Community Networking," 2000, http://www.startap.net/igrid/.

[50] D. F. McMullen, "The iGrid Project: Enabling International Research and Education Collaborations through High Performance Networking," presented at Proceedings of the IEEE Internet Workshop, Osaka, Japan, 1999.

[51] http://www.w3.org/TR/ws-arch/

[52] Specialist Crystallography at the Advanced Photon Source (SCrAPS), http://www.ansto.gov.au/natfac/scraps.html.

[53] MB-CAT at the Advanced Light Source, http://www.mbc-als.org.

[54] img-CIF/CBF, http://www.iucr.org/iucr-top/cif/imgcif/index.html.

[55] Atkins, Daniel E., et. al. *Revolutionizing Science and Engineering Through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure*. National Science Foundation, 2003. Page 1.4.

[56] Ibid, Page 3-10.

[57] Ibid, Page B-4.

[58] Ibid, Page B-11.