Software Infrastructure for the I-WAY High-Performance Distributed Computing Experiment

Ian Foster, Jonathan Geisler, Bill Nickless, Warren Smith, Steven Tuecke
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439, U.S.A.
{foster,geisler,nickless,wsmith,tuecke}@mcs.anl.gov
http://www.mcs.anl.gov/globus/

Abstract

High-speed wide area networks are expected to enable innovative applications that integrate geographically distributed, high-performance computing, database, graphics, and networking resources. However, there is as yet little understanding of the higher-level services required to support these applications, or of the techniques required to implement these services in a scalable, secure manner. We report on a large-scale prototyping effort that has yielded some insights into these issues. Building on the hardware base provided by the I-WAY, a national-scale Asynchronous Transfer Mode (ATM) network, we developed an integrated management and application programming system, called I-Soft. This system was deployed at most of the 17 I-WAY sites and used by many of the 60 applications demonstrated on the I-WAY network. In this article, we describe the I-Soft design and report on lessons learned from application experiments.

1 Introduction

Recent developments in high-performance networks, computers, information servers, and display technologies make it feasible to design *network-enabled tools* that incorporate remote compute and information resources into local computational environments, and *collaborative environments* that link people, computers, and databases into collaborative sessions. The development of such tools and environments raises numerous technical problems, including the naming and location of remote computational, communication, and data resources; the integration of these resources into computations; the location, characterization, and selection of available network connections; the provision of security and reliability; and uniform, efficient access to data.

Previous research and development efforts have produced a variety of candidate "point solutions" [19]. For example, DCE, CORBA, Condor [16], Nimrod [1], and Prospero [18] address problems of locating and/or accessing distributed resources; file systems such as AFS [17], DFS, and Truffles [4] address problems of sharing distributed data; tools such as Nexus [10], MPI [14], PVM [11], and Isis [2] address problems of coupling distributed computational resources; and low-level network technologies such as Asynchronous Transfer Mode (ATM) promise gigabit/sec communication. However, little work has been done to integrate these solutions in a way that satisfies the scalability, performance, functionality, reliability, and security requirements of realistic high-performance distributed applications in large-scale internetworks.

It is in this context that the I-WAY project [6] was conceived in early 1995, with the goal of providing a large-scale testbed in which innovative high-performance and geographically-distributed applications could be deployed. This application focus, argued the organizers, was essential if the research community was to discover the critical technical problems that must be addressed to ensure progress, and to gain insights into the suitability of different candidate solutions. In brief, the I-WAY was an ATM network connecting supercomputers, mass storage systems, and advanced visualization devices at 17 different sites within North America. It was deployed at the Supercomputing conference (SC'95) in San Diego in December 1995, and used by over 60 application groups for experiments in high-performance computing, collaborative design, and the coupling of remote supercomputers

and databases into local environments.

A central part of the I-WAY experiment was the development of a management and application programming environment, called I-Soft. The I-Soft system was designed to run on dedicated I-WAY point of presence (I-POP) machines deployed at each participating site, and provided uniform authentication, resource reservation, process creation, and communication functions across I-WAY resources. In this article, we describe the techniques employed in I-Soft development and we summarize the lessons learned during the deployment and evaluation process. The principal contributions are the design, prototyping, preliminary integration, and application-based evaluation of the following novel concepts and techniques:

1. *Point of presence* machines as a structuring and management technique for wide-area distributed computing.

2. A computational resource broker that uses *scheduler proxies* to provide a uniform scheduling environment that integrates diverse local schedulers.

3. The use of *authorization proxies* to construct a uniform authentication environment and define trust relationships across multiple administrative domains.

4. *Network-aware parallel programming tools* that use configuration information regarding topology, network interfaces, startup mechanisms, and node naming to provide a uniform view of heterogeneous systems and to optimize communication performance.

The rest of this article is as follows. In Section 2, we review the applications that motivated the development of the I-WAY and describe the I-WAY network. In Section 3, we introduce the I-WAY software architecture, and in Sections 4–8 we describe various components of this architecture and discuss lessons learned when these components were used in the I-WAY experiment. In Section 9, we discuss some related work. Finally, in Section 10, we present our conclusions and outline directions for future research.

2 The I-WAY Experiment

For clarity, in this article we refer consistently to the I-WAY experiment in the past tense. However, we emphasize that many I-WAY components have remained in place after SC'95 and that follow-on systems are being designed and constructed.

2.1 Applications

A unique aspect of the I-WAY experiment was its application focus. Previous gigabit testbed experiments focused on network technologies and low-level protocol issues, using either synthetic network loads or specialized applications for experiments (e.g., see [8]). The I-WAY, in contrast, was driven primarily by the requirements of a large application suite. As a result of a competitive proposal process in early 1995, around 70 application groups were selected to run on the I-WAY (over 60 were demonstrated at SC'95). These applications fell into three general classes [6]:

1. Many applications coupled immersive virtual environments with remote supercomputers, data systems, and/or scientific instruments. The goal of these projects was typically to combine state-of-the-art interactive environments and backend supercomputing to couple users more tightly with computers, while at the same time achieving distance independence between resources, developers, and users.

2. Other applications coupled multiple, geographically distributed supercomputers in order to tackle problems that were too large for a single supercomputer or that benefited from executing different problem components on different computer architectures.

3. A third set of applications coupled multiple virtual environments so that users at different locations could interact with each other and with supercomputer simulations.

Applications in the first and second classes are prototypes for future "network-enabled tools" that enhance local computational environments with remote compute and information resources; applications in the third class are prototypes of future collaborative environments.

2.2 The I-WAY network

The I-WAY network connected multiple high-end display devices (including immersive CAVE^{TM} and ImmersaDesk^{TM} virtual reality devices [5]); mass storage systems; specialized instruments (such as microscopes); and supercomputers of different architectures, including distributed memory multicomputers (IBM SP, Intel Paragon, Cray T3D, etc.), shared-memory multiprocessors (SGI Challenge, Convex Exemplar), and vector multiprocessors (Cray C90, Y-MP). These

devices were located at 17 different sites across North America.

This heterogeneous collection of resources was connected by a network that was itself heterogeneous. Various applications used components of multiple networks (e.g., vBNS, AAI, ESnet, ATDnet, CalREN, NREN, MREN, MAGIC, and CASA) as well as additional connections provided by carriers; these networks used different switching technologies and were interconnected in a variety of ways. Most networks used ATM to provide OC-3 (155 Mb/sec) or faster connections; one exception was CASA, which used HIPPI technology. For simplicity, the I-WAY standardized on the use of TCP/IP for application networking; in future experiments, alternative protocols will undoubtedly be explored. The need to configure both IP routing tables and ATM virtual circuits in this heterogeneous environment was a significant source of implementation complexity.

3 I-WAY Infrastructure

We now describe the software (and hardware) infrastructure developed for I-WAY management and application programming.

3.1 Requirements

We believe that the routine realization of high-performance, geographically distributed applications requires a number of capabilities not supported by existing systems. We list first *user-oriented* requirements; while none has been fully addressed in the I-WAY software environment, all have shaped the solutions adopted.

1. *Resource naming and location.* The ability to name computational and information resources in a uniform, location-independent fashion and to locate resources in large internets based on user or application-specified criteria.

2. *Uniform programming environment.* The ability to construct parallel computations that refer to and access diverse remote resources in a manner that hides, to a large extent, issues of location, resource type, network connectivity, and latency.

3. *Autoconfiguration and resource characterization.* The ability to make sensible configuration choices automatically and, when necessary, to obtain information about resource characteristics that can be used to optimize configurations.

4. *Distributed data services.* The ability to access conceptually "local" file systems in a uniform fashion, regardless of the physical location of a computation.

5. *Trust management.* Authentication, authorization, and accounting services that operate even when users do not have strong prior relationships with the sites controlling required resources.

6. *Confidentiality and integrity.* The ability for a computation to access, communicate, and process private data securely and reliably on remote sites.

Solutions to these problems must be scalable to large numbers of users and resources.

The fact that resources and users exist at different sites and in different administrative domains introduces another set of *site-oriented* requirements. Different sites not only provide different access mechanisms for their resources, but also have different policies governing their use. Because individual sites have ultimate responsibility for the secure and proper use of their resources, we cannot expect them to relinquish control to an external authority. Hence, the problem of developing management systems for I-WAY–like systems is above all one of defining protocols and interfaces that support a negotiation process between users (or brokers acting on their behalf) and the sites that control the resources that users want to access.

The I-WAY testbed provided a unique opportunity to deploy and study solutions to these problems in a controlled environment. Because the number of users (few hundred) and sites (around 20) were moderate, issues of scalability could, to a large extent, be ignored. However, the high profile of the project, its application focus, and the wide range of application requirements meant that issues of security, usability, and generality were of critical concern. Important secondary requirements were to minimize development and maintenance effort, both for the I-WAY development team and the participating sites and users.

3.2 Design overview

In principle, it would appear that the requirements just elucidated could be satisfied with purely software-based solutions. Indeed, other groups exploring the concept of a "metacomputer" have proposed software-only solutions [3, 12]. A novel aspect of our approach was the deployment of a dedicated I-WAY Point of Presence, or I-POP, machine at each participating site. As we explain in detail in the next section, these machines provided a uniform environment for deployment

of management software, and also simplified validation of security solutions by serving as a "neutral" zone under the joint control of I-WAY developers and local authorities.

Deployed on these I-POP machines was a software environment, I-Soft, providing a variety of services, including scheduling, security (authentication and auditing), parallel programming support (process creation and communication), and a distributed file system. These services allowed a user to log on to any I-POP and then schedule resources on heterogeneous collections of resources, initiate computations, and communicate between computers and with graphics devices—all without being aware of where these resources were located or how they were connected.

In the next four sections, we provide a detailed discussion of various aspects of the I-POP and I-Soft design, treating in turn the I-POPs, scheduler, security, parallel programming tools, and file systems. The discussion includes both descriptive material and a critical presentation of the lessons learned as a result of I-WAY deployment and demonstration at SC'95.

4 Point of Presence Machines

We have explained why management systems for I-WAY–like systems need to interface to local management systems, rather than manage resources directly. One critical issue that arises in this context is the physical location of the software used to implement these interfaces. For a variety of reasons, it is desirable that this software execute behind site firewalls. Yet this location raises two difficult problems: sites may, justifiably, be reluctant to allow outside software to run on their systems; and system developers will be required to develop interfaces for many different architectures.

The use of I-POP machines resolve these two problems by providing a uniform, jointly administered physical location for interface code. The name is chosen by analogy with a comparable device in telephony. Typically, the telephone company is responsible for, and manages, the telephone network, while the customer owns the phones and in-house wiring. The interface between the two domains lies in a switchbox which serves as the telephone company's "point of presence" at the user site.

4.1 I-POP design

Figure 1 shows the architecture of an I-POP machine. It is a dedicated workstation, accessible via the Internet and operating inside a site's firewalls. It runs

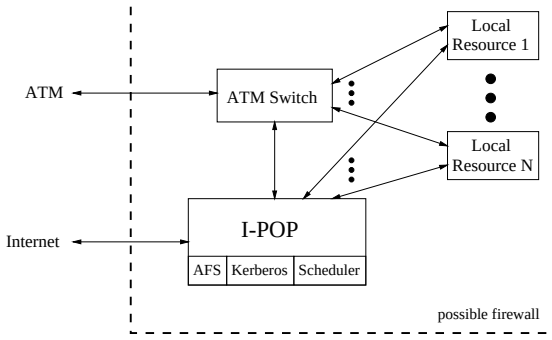

Figure 1. An I-WAY Point of Presence (I-POP) machine

a standard set of software supplied by the I-Soft developers. An ATM interface allows it to monitor and, in principle, manage the site's ATM switch; it also allows the I-POP to use the ATM network for management traffic. Site-specific implementations of a simple management interface allow I-WAY management systems to communicate with other machines at the site to allocate resources to users, start processes on resources, and so forth. The Andrew distributed file system (AFS) [17] is used as a repository for system software and status information.

Development, maintenance, and auditing costs are significantly reduced if all I-POP computers are of the same type. In the I-WAY experiment, we standardized on Sun SPARCStations. A standard software configuration included SunOS 4.1.4 with latest patches; a limited set of Unix utilities; the Cygnus release of Kerberos 4; AFS; the I-WAY scheduler; and various security tools such as Tripwire [15], TCP wrappers, and auditing software. This software was maintained at a central site (via AFS) and could be installed easily on each I-POP; furthermore, the use of Tripwire meant that it was straightforward to detect changes to the base configuration.

The I-POP represented a dedicated point of presence for the I-WAY at the user site. It was jointly managed: the local site could certify the I-POP's software configuration, and could disconnect the I-POP to cut access to the I-WAY in the event of a security problem; similarly, the I-WAY security team could log accesses, check for modifications to its configuration, and so forth. The dedicated nature of the I-POP meant that its software configuration could be kept simple, facilitating certification and increasing trust.

4.2 I-POP discussion

Seventeen sites deployed I-POP machines. For the most part the effort required to install software, integrate a site into the I-WAY network, and maintain the site was small (in our opinion, significantly less than if I-POPs had not been used). The fact that all I-POPs shared a single AFS cell proved extremely useful as a means of maintaining a single, shared copy of I-Soft code and as a mechanism for distributing I-WAY scheduling information. The deployment of I-POPs was also found to provide a conceptual framework that simplified the task of explaining the I-WAY infrastructure, both to users and to site administrators.

While most I-POPs were configured with ATM cards, we never exploited this capability to monitor or control the ATM network. The principal reason was that at many sites, the ATM switch to which the I-POP was connected managed traffic for both I-WAY and non–I-WAY resources. Hence, there was a natural reluctance to allow I-POP software to control the ATM switches. These authentication, authorization, and policy issues will need to be addressed in future I-WAY–like systems.

We note that the concept of a Point of Presence machine as a locus for management software in a heterogeneous I-WAY–like system is a unique contribution of this work. The most closely related development is that of the ACTS ATM Internetwork (AAI) network testbed group: they deployed fast workstations at each site in a Gigabit testbed, to support network throughput experiments [8].

5 Scheduler

I-WAY–like systems require the ability to locate computational resources matching various criteria in a heterogeneous, geographically distributed pool. As noted above, political and technical constraints make it infeasible for this requirement to be satisfied by a single "I-WAY scheduler" that replaces the schedulers that are already in place at various sites. Instead, we need to think in terms of a negotiation process by which requests (ideally, expressible in a fairly abstract form, e.g., "N Gigaflops," or "X nodes of type Y, with maximum latency Z") are handled by an independent entity, which then negotiates with the site schedulers that manage individual resources. We coin the term Computational Resource Broker (CRB) to denote this entity. In an Internet-scale distributed computing system, we can imagine a network of such brokers. In the I-WAY, one was sufficient.

5.1 Scheduler design

The practical realization of the CRB concept requires the development of fairly general user-to-CRB and CRB-to-resource scheduler protocols. Time constraints in the I-WAY project limited what we could achieve in each area. On the user-to-CRB side, we allowed users to request access only to predefined disjoint subsets of I-WAY computers called *virtual machines*; on the CRB-to-resource scheduler side, we required sites to turn over scheduling control of specified resources to the I-WAY scheduler, which would then use the resources to construct virtual machines. In effect, our simple CRB obtained access to a block of resources, which it then distributed to its users.

The scheduler that was defined to meet these requirements provided *management functions* that allowed administrators to configure dedicated resources into virtual machines, obtain status information, and so forth; and *user functions* that allowed users to list available virtual machines and to determine status, list queued requests, or request time on a particular virtual machine.

The scheduler implementation was structured in terms of a single central scheduler and multiple local scheduler daemons. The *central scheduler* daemon maintained the queues and tables representing the state of the different virtual machines, and was responsible for allocating time on these machines on a first-come, first-served basis. It also maintained state information on the AFS file system, so as to provide some fault tolerance in the case of daemon failures. The central scheduler communicated with *local scheduler* daemons, one per I-POP, to request that operations be performed on particular machines. Local schedulers performed site-dependent actions in response to three simple requests from the central scheduler.

- *Allocate resource.* This request enables a local scheduler to perform any site-specific initialization required to make a resource usable by a specified user, for example, by initializing switch configurations so that processors allocated to a user can communicate, and propagating configuration data.

- *Create process.* This request asks a local scheduler to create a process on a specified processor, as a specified user: it implements, in effect, a Unix remote shell, or `rsh`, command. This provides the basic functionality required to initiate remote computations; as we discuss below, it can be used directly by a user, and is also used to implement other user-level functions such as `ixterm` (start an

X-terminal process on a specified processor), `ircp` (start a copy process on a specified processor), and `impirun` (start an MPI program on a virtual machine).

- *Deallocate resource.* This request enables a local scheduler to perform any site-specific operations that may be required to terminate user access to a resource: for example, disabling access to a high-speed interconnect, killing processes, or deleting temporary files.

5.2 Scheduler discussion

The basic scheduler structure just described was deployed on a wide variety of systems (interfaces were developed for all I-WAY resources) and was used successfully at SC'95 to schedule a large number of users. Its major limitations related not to its basic structure but to the too-restrictive interfaces between user and scheduler and scheduler and local resources.

The concept of using fixed virtual machines as schedulable units was only moderately successful. Often, no existing virtual machine met user requirements, in which case new virtual machines had to be configured manually. This difficulty would have been avoided if even a very simple specification language that allowed requests of the form "give me M nodes of type X and N nodes of type Y" had been supported. This feature could easily be integrated into the existing framework. The development of a more sophisticated resource description language and scheduling framework is a more difficult problem and will require further research.

A more fundamental limitation related to the often limited functionality provided by the non–I-WAY resource schedulers with which local I-WAY schedulers had to negotiate. Many were unable to inquire about completion time of scheduled jobs (and hence expected availability of resources) or to reserve computational resources for specified timeslots; several sites provided timeshared rather than dedicated access. In addition, at some sites, networking and security concerns required that processors intended for I-WAY use be specially configured. We compensated either by dedicating partitions to I-WAY users or by timesharing rather than scheduling. Neither solution was ideal. In particular, the use of dedicated partitions meant that frequent negotiations were required to adapt partition size to user requirements, and that computational resources were often idle. The long-term solution probably is to develop more sophisticated schedulers for resources that are to be incorporated into I-WAY–like systems. However, applications also may need to become more flexible about what type and "quality" of resources they can accept.

We note that while many researchers have addressed problems relating to scheduling computational resources in parallel computers or local area networks, few have addressed the distinctive problems that arise when resources are distributed across many sites. Legion [13] and Prospero [18] are two exceptions. In particular, Prospero's "system manager" and "node manager" processes have some similarities to our central and local managers. However, neither system supports interfaces to other schedulers: they require full control of scheduled resources.

6 Security

Security is a major and multifaceted issue in I-WAY–like systems. Ease-of-use concerns demand a uniform authentication environment that allows a user to authenticate just once in order to obtain access to geographically distributed resources; performance concerns require that once a user is authenticated, the authorization overhead incurred when accessing a new resource should be small. Both uniform authentication and low-cost authorization are complicated in scalable systems, because users will inevitably need to access resources located at sites with which they have no prior trust relationship.

6.1 Security design

When developing security structures for the I-WAY software environment, we focused on providing a uniform authentication environment. We did not address in any detail issues relating to authorization, accounting, or the privacy and integrity of user data. Our goal was to provide security at least as good as that existing at the I-WAY sites. Since all sites used clear-text password authentication, this constraint was not especially stringent. Unfortunately, we could not assume the existence of a distributed authentication system such as Kerberos (or DCE, which uses Kerberos) because no such system was available at all sites.

Our basic approach was to separate the authentication problem into two parts: authentication to the I-POP environment and authentication to the local sites. Authentication to I-POPs was handled by using a `telnet` client modified to use Kerberos authentication and encryption. This approach ensured that users could authenticate to I-POPs without passing passwords in clear text over the network. The scheduler software kept track of which user id was to be used at each site for a particular I-WAY user, and served

as an "authentication proxy," performing subsequent authentication to other I-WAY resources on the user's behalf. This proxy service was invoked each time a user used the command language described above to allocate computational resources or to create processes.

The implementation of the authentication proxy mechanism was integrated with the site-dependent mechanisms used to implement the scheduler interface described above. In the I-WAY experiment, most sites implemented all three commands using a privileged (root) `rsh` from the local I-POP to an associated resource. This method was used because of time constraints and was acceptable only because the local site administered the local I-POP, and the `rsh` request was sent to a local resource over a secure local network.

6.2 Security discussion

The authentication mechanism just described worked well in the sense that it allowed users to authenticate once (to an I-POP) and then access any I-WAY resource to which access was authorized. The "authenticate-once" capability proved to be extremely useful and demonstrated the advantages of a common authentication and authorization environment.

One deficiency of the approach related to the degree of security provided. Root `rsh` is an unacceptable long-term solution even when the I-POP is totally trusted, because of the possibility of IP-spoofing attacks. We can protect against these attacks by using a remote shell function that uses authentication (for example, one based on Kerberos [20] or PGP, either directly or via DCE). For similar reasons, communications between the scheduling daemons should also be authenticated.

A more fundamental limitation of the I-WAY authentication scheme as implemented was that each user had to have an account at each site to which access was required. Clearly, this is not a scalable solution. One alternative is to extend the mechanisms that map I-WAY user ids to local user ids, so that they can be used to map I-WAY user ids to preallocated "I-WAY proxy" user ids at the different sites. The identity of the individual using different proxies at different times could be recorded for audit purposes. However, this approach will work only if alternative mechanisms can be developed for the various functions provided by an "account." The formal application process that is typically associated with the creation of an account serves not only to authenticate the user but also to establish user obligations to the site (e.g., "no commercial work" is a frequent requirement at academic sites) and to define the services provided by the site to the user (e.g., backup policies). Proxy accounts address only the authentication problem (if sites trust the I-WAY). Future approaches will probably require the development of formal representations of conditions of use, as well as mechanisms for representing transitive relationships. (For example, a site may agree to trust any user employed by an organization with which it has formalized a trust relationship. Similarly, an organization may agree on behalf of its employees to obligations associated with the use of certain resources.)

7 Parallel Programming Tools

A user who has authenticated to an I-POP and acquired a set of computational resources then requires mechanisms for creating computations on these resources. At a minimum, these mechanisms must support the creation of processes on different processors and the communication of data between these processes. Because of the complexity and heterogeneity of I-WAY–like environments, tools should ideally also relieve the programmer of the need to consider low-level details relating to network structure. For example, tools should handle conversions between different data representations automatically, and be able to use different protocols when communicating within rather than between parallel computers. At the same time, the user should be able to obtain access to low-level information (at an appropriate level of abstraction) when it is required for optimization purposes.

7.1 Parallel tools design

The `irsh` and `ixterm` commands described above allow authenticated and authorized users to access, and initiate computation on, any I-WAY resource. Several users relied on these commands alone to initiate distributed computations that then communicated by using TCP/IP sockets. However, this low-level approach did not hide (or exploit) any details of the underlying network.

To support the needs of users desiring a higher-level programming model, we adapted the Nexus multithreaded communication library [10] to execute in an I-WAY environment. Nexus supports automatic configuration mechanisms that allow it to use information contained in resource databases to determine which startup mechanisms, network interfaces, and protocols to use in different situations. For example, in a virtual machine connecting IBM SP and SGI Challenge computers with both ATM and Internet networks, Nexus uses three different protocols (IBM proprietary MPL on the SP, shared-memory on the Challenge, and

TCP/IP or AAL5 between computers), and selects either ATM or Internet network interfaces, depending on network status. We modified the I-WAY scheduler to produce appropriate resource database entries when a virtual machine was allocated to a user. Nexus could then use this information when creating a user computation. (Nexus support for multithreading should, in principle, also be useful—for latency hiding—although in practice it was not used for that purpose during the I-WAY experiment.)

Several other libraries, notably the CAVEcomm virtual reality library [7] and the MPICH implementation of MPI, were extended to use Nexus mechanisms [9]. Since MPICH is defined in terms of an "abstract point-to-point communication device," an implementation of this device in terms of Nexus mechanisms was not difficult. Other systems that use Nexus mechanisms include the parallel language CC++ and the parallel scripting language nPerl, used to write the I-WAY scheduler.

7.2 Parallel tools discussion

The I-WAY experiment demonstrated the advantages of the Nexus automatic configuration mechanisms. In many cases, user were able to develop applications with high-level tools such as MPI, CAVEcomm, and/or CC++, without any knowledge of low-level details relating to the compute and network resources included in a computation.

A significant difficulty revealed by the I-WAY experiment related to the mechanisms used to generate and maintain the configuration information used by Nexus. While resource database entries were generated automatically by the scheduler, the information contained in these entries (such as network interfaces) had to be provided manually by the I-Soft team. The discovery, entry, and maintenance of this information proved to be a significant source of overhead, particularly in an environment in which network status was changing rapidly. Clearly, this information should be discovered automatically whenever possible. Automatic discovery would make it possible, for example, for a parallel tool to use dedicated ATM links if these were available, but to fall back automatically to shared Internet if the ATM link was discovered to be unavailable. The development of such automatic discovery techniques remains a challenging research problem.

The Nexus communication library provides mechanisms for querying the resource database, which users could have used to discover some properties of the machines and networks on which they were executing. In practice, few I-WAY applications were configured to use this information; however, we believe that this situation simply reflects the immature state of practice in this area, and that users will soon learn to write programs that exploit properties of network topology, etc. Just what information users will find useful remains to be seen, but presumably enquiry functions that reveal the number of machines involved in a computation and the number of processors in each machine would definitely be required. One application that could certainly benefit from access to information about network topology is the I-WAY MPI implementation. Currently, this library implements collective operations using algorithms designed for multicomputer environments; presumably, communication costs can often be reduced by using communication structures that avoid intermachine communication.

8 File Systems

I-WAY–like systems introduce three related requirements with a file-system flavor. First, many users require access to various status data and utility programs at many different sites. Second, users running programs on remote computers must be able to access executables and configuration data at many different sites. Third, application programs must be able to read and write potentially large data sets. These three requirements have very different characteristics. The first requires support for multiple users, consistency across multiple sites, and reliability. The second requires somewhat higher performance (if executables are large), but does not require support for multiple users. The third requires, above all, high performance. We believe that these three requirements are best satisfied with different technologies.

The I-Soft system supported only the first of these requirements. An AFS cell (with three servers for reliability) was deployed and used as a shared repository for I-WAY software, and also to maintain scheduler status information. The AFS cell was accessible only from the I-POPs, since many I-WAY computers did not support AFS, and when they did, authentication problems made access difficult. The only assistance provided for the second and third requirements was a remote copy (`ircp`) command that supported the copying of data from one machine to another.

While the AFS system was extremely useful, the lack of distributed file system support on I-WAY nodes was a serious deficiency. Almost all users found that copying files and configuration data to remote sites was an annoyance, and some of the most ambitious I-WAY applications had severe problems postprocessing, transporting, and visualizing the large amounts of

data generated at remote sites. Future I-WAY–like systems should support something like AFS on all nodes, and if necessary provide specialized high-performance distributed data access mechanisms for performance-critical applications.

9 Related Work

In preceding sections, we have referred to a number of systems that provide point solutions to problems addressed in I-Soft development. Here, we review systems that seek to provide an integrated treatment of distributed system issues, similar or broader in scope than I-Soft.

The Distributed Computing Environment (DCE) and Common Object Request Broker Architecture (CORBA) are two major industry-led attempts to provide a unifying framework for distributed computing. Both define (or will define in the near future) a standard directory service, remote procedure call (RPC), security service, and so forth; DCE also defines a Distributed File Service (DFS) derived from AFS. Issues such as fault tolerance and interoperability between languages and systems are addressed. In general, CORBA is distinguished from DCE by its higher level, object-oriented architecture. Some DCE mechanisms (RPC, DFS) may well prove to be appropriate for implementing I-POP services; CORBA directory services may be useful for resource location. However, both DCE and CORBA appear to have significant deficiencies as a basis for application programming in I-WAY–like systems. In particular, the remote procedure call is not well-suited to applications in which performance requirements demand asynchronous communication, multiple outstanding requests, and/or efficient collective operations.

The Legion project [13] is another project developing software technology to support computing in wide-area environments. Issues addressed by this wide-reaching effort include scheduling, file systems, security, fault tolerance, and network protocols. The I-Soft effort is distinguished by its focus on high-performance systems and by its use of I-POP and proxy mechanisms to enhance interoperability with existing systems.

10 Conclusions

We have described the management and application programming environment developed for the I-WAY distributed computing experiment. This system incorporates a number of ideas that, we believe, may be useful in future research and development efforts. In particular, it uses point of presence machines as a means of simplifying system configuration and management, scheduler proxies for distributed scheduling, authentication proxies for distributed authentication, and network-aware tools that can exploit configuration information to optimize communication behavior. The I-Soft development also took preliminary steps towards integrating these diverse components, showing, for example, how a scheduler can provide network topology information to parallel programming tools.

The SC'95 event provided an opportunity for intense and comprehensive evaluation of the I-Soft and I-POP systems. I-Soft was a success in that most applications ran successfully at least some of the time; the network rather than the software proved to be the least reliable system component. Specific deficiencies and limitations revealed by this experience have been detailed in the text. More generally, we learned that system components that are typically developed in isolation must be more tightly integrated if performance, reliability, and usability goals are to be achieved. For example, resource location services in future I-WAY–like systems will need low-level information on network characteristics; schedulers will need to be able to schedule network bandwidth as well as computers; and parallel programming tools will need up-to-date information on network status.

We are now working to address some of the critical research issues identified in I-Soft development. The Globus project, involving Argonne, Caltech, the Aerospace Corporation, and Trusted Information Systems, is addressing issues of resource location (computational resource brokers), automatic configuration, scalable trust management, and high-performance distributed file systems. In addition, we and others are defining and constructing future I-WAY–like systems that will provide further opportunities to evaluate management and application programming systems such as I-Soft.

Acknowledgments

The I-WAY was a multi-institutional, multi-individual effort. Tom DeFanti, Rick Stevens, Tim Kuhfuss, Maxine Brown, Linda Winkler, Mary Spada, and Remy Evard played major roles. We acknowledge in particular Carl Kesselman and Steve Schwab (I-Soft design), Gene Rackow (I-POP software), Judy Warren (AFS), Doru Marcusiu (I-Soft deployment), Bill Gropp and Ewing Lusk (MPI), and Gary Minden, Mike St Johns, and Ken Rowe (security architecture). This work was supported in part by the Mathematical, Information, and Computational Sciences Division sub-

program of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

References

[1] D. Abramson, R. Sosic, J. Giddy, and B. Hall. Nimrod: A tool for performing parameterised simulations using distributed workstations. In *Proc. 4th IEEE Symp. on High Performance Distributed Computing*. IEEE Press, 1995.

[2] K. Birman. The process group approach to reliable distributed computing. *Communications of the ACM*, 36(12):37–53, 1993.

[3] C. Catlett and L. Smarr. Metacomputing. *Communications of the ACM*, 35(6):44–52, 1992.

[4] J. Cook, S.D. Crocker, Jr. T. Page, G. Popek, and P. Reiher. Truffles: Secure file sharing with minimal system administratore intervention. In *Proc. SANS-II, The World Conference On Tools and Techniques For System Administration, Networking, and Security*. 1993.

[5] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, R.V. Kenyon, and J.C. Hart. The CAVE: Audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):65–72, 1992.

[6] T. DeFanti, I. Foster, M. Papka, R. Stevens, and T. Kuhfuss. Overview of the I-WAY: Wide area visual supercomputing. *International Journal of Supercomputer Applications*, 1996. in press.

[7] T. L. Disz, M. E. Papka, M. Pellegrino, and R. Stevens. Sharing visualization experiences among remote virtual environments. In *International Workshop on High Performance Computing for Computer Graphics and Visualization*, pages 217–237. Springer-Verlag, 1995.

[8] B. Ewy, J. Evans, V. Frost, and G. Minden. TCP and ATM in wide area networks. In *Proc. 1994 IEEE Gigabit Networking Workshop*. IEEE, 1994.

[9] I. Foster, J. Geisler, and S. Tuecke. MPI on the I-WAY: A wide-area, multimethod implementation of the Message Passing Interface. In *Proceedings of the 1996 MPI Developers Conference*. IEEE Computer Society Press, 1996.

[10] I. Foster, C. Kesselman, and S. Tuecke. The Nexus approach to integrating multithreading and communication. *Journal of Parallel and Distributed Computing*, 1996. To appear.

[11] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine—A User's Guide and Tutorial for Network Parallel Computing*. MIT Press, 1994.

[12] A. Grimshaw, J. Weissman, E. West, and E. Lyot, Jr. Metasystems: An approach combining parallel processing and heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 21(3):257–270, 1994.

[13] A. Grimshaw, W. Wulf, J. French, A. Weaver, and P. Reynolds, Jr. Legion: The next logical step toward a nationwide virtual computer. Technical Report CS-94-21, Department of Computer Science, University of Virginia, 1994.

[14] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. MIT Press, 1995.

[15] G. H. Kim and E. H. Spafford. Writing, supporting, and evaluating Tripwire: A publically available security tool. In *Proc. USENIX Unix Applications Development Symp.*, pages 89–107. The USENIX Association, 1994.

[16] M. Litzkow, M. Livney, and M. Mutka. Condor - a hunter of idle workstations. In *Proc. 8th Intl Conf. on Distributed Computing Systems*, pages 104–111, 1988.

[17] J.H. Morris et al. Andrew: A distributed personal computing environment. *CACM*, 29(3), 1986.

[18] B. Clifford Neumann and Santosh Rao. The Prospero resource manager: A scalable framework for processor allocation in distributed systems. *Concurrency: Practice and Experience*, June 1994.

[19] S. Mullender (ed.). *Distributed Systems*. ACM Press, 1989.

[20] J. Steiner, B. Neumann, and J. Schiller. Kerberos: An authentication system for open network systems. In *Usenix Conference Proceedings*, pages 191–202. 1988.