Useful and robust operation of real-time distributed systems requires many capabilities, including automated management of data streams and distributed system components; semiautonomous operation of remote instrument systems; generalized access control; dynamic scheduling and resource reservation; application designs that can adapt to congestion; and brokering mechanisms. These capabilities will be built on supporting architecture, middleware, and low-level services, such as high-speed network-based caches, realtime cataloging systems, and agent-based systems that provide, for example, dynamic performance analysis.

In this chapter we first provide some rationale for remote realtime applications and then characterize the associated problems, discussing the nature of remote operation in terms of an example: a remotely controlled beamline at the Advanced Light Source. Then, we provide more detailed descriptions of three additional applications that we have constructed and that illustrate the practical issues that arise in systems of this type: a cardioangiography system that depends on realtime data cataloging, a high-energy physics data-processing architecture that supports very high data rates and volume, and a remote-control system for an electron microscope.

Finally, we discuss issues and approaches to providing the infrastructure required to support the cited applications. We describe a model architecture, network-based caches, agent-based management and monitoring, and policy-based access control systems.
4.1 DISTRIBUTED REALTIME APPLICATIONS

High-speed data streams result from the operation of many types of online instruments and imaging systems and are a staple of modern scientific, health care, and intelligence environments. The advent of shared, widely available high-speed networks is providing the potential for new approaches to the collection, organization, storage, analysis, and distribution of the large data objects that result from such data streams. These new approaches will make both the data and its analysis much more readily available. To illustrate this emerging paradigm, we examine several examples that come from quite different application domains but that have a number of similar architectural elements.

Health care imaging systems illustrate both high data rates and the need for realtime cataloging. High-volume health care video and image data used for diagnostic purposes (e.g., X-ray CT, MRI, and cardioangiography) are collected at centralized facilities and, through widely distributed systems, may be stored, managed, accessed, and referenced at locations other than the point of collection (e.g., the hospitals of the referring physicians). In health care imaging systems, it is important that the health care professionals at the referring facility (hospitals or clinics frequently remote from the tertiary imaging facility) have ready access not only to the image analyst's reports, but also to the original image data. Additionally, it is important to provide and manage distributed access to tertiary storage because laboratory instrumentation environments, hospitals, and so on are frequently not the best place to maintain a large-scale digital storage system. Such systems can have considerable economy of scale in operational aspects, and an affordable, easily accessible, high-bandwidth network can provide location independence for such systems.

High-energy physics experiments illustrate both very high data rates and volumes that have to be processed and archived in real time and must be accessible to large scientific collaborations—typically hundreds of investigators at dozens of institutions around the world. High-bandwidth (20–40 MB/s) data handling for analysis of high-energy and nuclear physics data is increasingly likely to have a source of data that is remote from the computational and storage facilities. The output from particle detectors (the instrument) is subjected to several stages of data reduction and analysis. After the initial processing, the analysis functions are carried out by dispersed collaborators and facilities. Their analysis is then organized in information systems that may reside on a single storage system or be distributed among several physical systems.

Remote microscopy control illustrates the problem of the human always being remote from the controlled system or object of interest. Data is typically
collected as images (in the spatial or Fourier domains) that are then analyzed to provide information both for experiment control and for analysis. Experiment and instrument control includes object tracking, both in order to keep the object visible (e.g., drift and depth-of-focus compensation) and to observe changes in the object. Some of this information may be fed back to the apparatus that is acting on the object, as in application of electromagnetic fields and thermal gradients. In all of these cases, the precision, repetition, or time scale means that humans cannot directly perform the required tasks effectively. The human operators provide the high-level control, such as initially identifying objects of interest, establishing operating set points, and defining protocols for the in situ experiments. Automated operation of the low-latency, low-level control enables the human functions to be carried out over wide area as well as local area networks.

4.2 PROBLEM CHARACTERIZATION AND PROTOTYPES

Realtime management of distributed instrumentation systems involves remote operation of instrument control functions, distributed data collection and management, and/or distributed data analysis and cataloging. Each of these regimes requires a supporting infrastructure of middleware and of systems and communications services.

The required middleware services include automated cataloging and tertiary storage system interfaces (i.e., a digital library system between the instrument and the user; see Chapter 5); automated monitoring and management systems for all aspects of the distributed components (see Chapters 14 and 15); policy-based access control systems to support scheduling and resource allocation (e.g., quality of service, or QoS; see Chapter 19), security, distributed system integrity, and (potentially) automated brokering and system construction; and rich media capabilities to support telepresence and collaboration (see Chapter 6).

Supporting systems and communications services include flexible transport mechanisms; reliable and unreliable wide area multicast; resource reservation and QoS for computing, storage, and communications; and security to protect the network-level infrastructure (see Chapter 16).

These capabilities are not sufficient, but are a representative collection of necessary services for remotely operated, high-performance data systems. In the next few sections we will illustrate some of the issues that give rise to the need for these services.
4.2.1 The Nature of Remote Operation

Distributed instruments can be remote in space, scale, or time. Remote in space is the typical circumstance for network-distributed scientific collaboration, where instruments are located at one facility, users are located at others, and data processing and storage at yet others. Another common circumstance is that the controlled function is sufficiently remote in scale that direct control is not possible. Many microscopic experiment environments fall into this category. The operation of the Mars Pathfinder mission Rover vehicle provides an example of functional control that is remote in time. (Rover operation was specified a day in advance, and then the actions were uploaded for the following day’s mission, which was carried out autonomously.) Each of these scenarios provides circumstances that have to be addressed for remote operation.

When the operator is remote from the instrument, as is the case when the instrument is located at a national facility like the Lawrence Berkeley National Laboratory (LBNL) Advanced Light Source, and the investigators are located at universities and laboratories scattered across the country, several issues arise. Multiple media streams are typically required in order to support human interaction (audio and video conferencing and worksurface sharing) and to provide a sense of presence (remote environment monitoring) so that the general environment, including the equipment area and local personnel, can be observed in order to verify general operational status. The experiment itself (e.g., a sample chamber) must typically be visually monitored as a “sanity” check to ensure that the data stream is actually the result of the intended experiment. Finally, since data is shared in real time among several experimenters, additional data streams are required for online analysis and control (see Plate 7) [10, 9, 485].

Multiple collaborators, each of whom need to see the instrument output in real time and potentially control the instrument, require synchronized and reliable access to the data and control. The shared control panels shown in Plate 7 illustrate such a capability, which is based on reliable multicast protocols.

When the scale of the operations is very different from human scale, remote operation must typically involve some machine intelligence. Automated operations analyze the sensor data in real time and adapt the progress of the experiment depending on the results of analysis. The human function is to set up the experiment, identify the object of interest in the experiment environment, and so on. The actual operation, however, cannot be in human hands.
Automation can also be critical to the remote operation of experiments when operating over a wide area IP network of unpredictable or high latency. Such a network cannot be used to provide fine-grained, real-time control, as required, for example, in a closed-loop servo system where the operating functions are at one end of the network and the data analysis that provides the feedback is at the other end. Incorporating machine intelligence into the experiment control system and remotely performing monitoring and data analysis address this problem.

In the rest of this section, we use three examples to illustrate some approaches to addressing these issues.

4.2.2 Cardioangiography: Realtime Data Cataloging

In many environments the key aspect of realtime data is the immediate and automated processing necessary to organize and catalog the data and make it available to remote sites. The online cardioangiography system that we describe here is typical of such an environment. Data is generated in large volumes and with high throughput, and the people generating the data are geographically separated from the people cataloging or using the data.

There are several important considerations for managing this type of instrument-generated data:

1. Automatic generation of at least minimal metadata
2. Automatic cataloging of the data and the metadata as the data is received (or as close to real time as possible)
3. Transparent management of tertiary storage systems where the original data is archived
4. Facilitation of cooperative research by providing specified users at local and remote sites immediate as well as long-term access to the data
5. Incorporation of the data into other databases or documents

For the online cardioangiography system (a remote medical imaging system), a realtime digital library system collects data from the instrument and automatically processes, catalogs, and archives each data unit together with the derived data and metadata, with the result being a Web-based object representing each data set. This automatic system operates 10 hours/day, 5–6 days/week, with data rates of about 30 Mb/s during the data collection phase (about 20 minutes per hour); see Figure 4.1.
WALDO (Wide Area Large Data Objects) is a realtime digital library system that uses federated textual and URL-linked metadata to represent the characteristics of large data sets (see Figure 4.2 and [296]). Incoming data is automatically cataloged by extracting associated metadata and converting it into text records, by generating auxiliary metadata and derived data, and by combining this data into Web-based objects that include persistent references to the original data components. Tertiary storage management for the original data sets is achieved by using the remote program execution capability of Web servers to manage the data on a mass storage system. For subsequent use, the data components may be staged to a local disk and then returned as usual via the Web browser or, as is the case for high-performance applications, moved to a high-speed cache for direct access by the specialized applications. The location of the data components on tertiary storage, information on how to access them, and other descriptive material are all part of the object definition. The creation of object definitions, the inclusion of “standardized” derived-data objects as part of the metadata, and the use of typed links in the object definition are intended to provide a general framework for dealing with many different types of data, including abstract instrument data and multicomponent, multimedia programs.
4.2 Problem Characterization and Prototypes

WALDO uses an object-oriented approach for capture, storage, catalog, retrieval, and management of large data objects (LDO) and their associated metadata. The architecture includes a collection of widely distributed services to provide flexibility in managing storage resources, reliability and integrity of access, and high-performance access, all in an open environment where the use conditions for resources and stored information are guaranteed through the use of a strong, but decentralized, security architecture.

**Elements of the WALDO Model**

The WALDO model offers realtime cataloging of extensible, linked, multicomponent data objects that can be asynchronously generated by remote, online data sources. Class-based methods are used to manage large data objects. Collections of data objects are handled by flexible curator/collection owner management, including “anytime” management of the collection organization...
and object metadata. There is also globally unique and persistent naming of the objects and their various components via URLs and URNs. There is strong access control at the level of individual object components based on use condition certificates managed by the data owner. Additionally, there is high-performance application access to the data components.

**WALDO Software Architecture**

Figure 4.2 illustrates the data flow and overall organization of the WALDO architecture. The basic elements of the architecture include the following:

1. Data collection systems and the instrument network interfaces
2. High-speed, network-based cache storage for receiving data, for providing intermediate storage for processing, and for high-speed application access
3. Processing mechanisms for various sorts of data analysis and derived data generation
4. Data management that provides for the automatic cataloging and metadata generation that produces the large data object definitions
5. Data access interfaces, including application-oriented interfaces
6. Flexible mechanisms for providing various searching strategies
7. Transparent security that provides strong access control for the data components based on data owner policies
8. Transparent tertiary storage ("mass storage") management for the data components
9. Curator interfaces for managing both the metadata and the large data object collection
10. User access interfaces for all relevant aspects of the data (applications, data, and metadata)

These elements are all provided with flexible, location-independent interfaces so that they can be freely (transparently) moved around the network as required for operational or other logistical convenience.

The model just described has been used in several data-intensive computing applications; however, it raises a number of issues. The distributed cache is an important component, but one that requires distributed management and distributed security. The incorporation of a digital-library-like function is an important consideration, but such automatic cataloging in the
face of human error in the operation of the instrument (and the resulting errors in the metadata and cataloging) requires human curation of the library. Access control is a critical aspect when sensitive or confidential data is involved, and the management of the access control must also be distributed to the various principals. Approaches to several of these issues are discussed below.

4.2.3 Particle Accelerators: High-Data-Rate Systems

Our next example concerns a detector system at a high-energy physics particle accelerator. Modern detectors like STAR (Solenoidal Tracker at the Relativistic Heavy Ion Collider at Brookhaven National Laboratory) will generate 20-40 MB/s of data that must be processed in two phases: data collection and event reconstruction (Phase 1) and physics analysis (Phase 2) [247, 295].

In Phase 1, a detector puts out a steady-state high-data-rate stream. Traditionally, the data is archived, and a first level of processing is performed at the experiment site. The resulting second-level data is also archived and then used for the subsequent physics analysis. The data is thus archived at the experiment site in “medium-sized” tertiary storage systems. This approach has disadvantages: large mass storage systems are one of the few computing technologies that continue to exhibit significant economies of scale, and therefore central sites remain an important architectural component in high-data-volume systems. However, the potential problems of network access to large-scale storage systems can be overcome with network-based caching.

In a grid environment, medium-sized tertiary systems at experiment sites can be replaced by a distributed cache consisting of a high-speed, high-capacity network-based cache and very large tertiary systems at dedicated storage sites.

The Distributed Parallel Storage System (DPSS), described below, can serve as the cache for all stages of data manipulation. DPSS provides a scalable, dynamically configurable, high-performance, and highly distributed storage system that is usually used as a (relatively long-term) cache of data. It is typically used to collect data from online instruments and then supply that data to analysis applications or to high-data-rate visualization applications (as in the case of the MAGIC, the wide area gigabit testbed where DPSS was originally developed; see Chapter 21 [330, 539, 476]). The system is also being used in satellite image-processing systems and for the distributed online, high-data-rate health care imaging systems described above.
The architecture illustrated in Figure 4.3 supports distributed computational systems doing the Phase 1 data processing in real time. Realtime data processing potentially also supports two capabilities. First, it can provide auxiliary information to assist in the organization of data as it is transferred to tertiary storage (the STAR experiment will generate about 1.7 TB/day). Second, it can provide feedback to the instrument operators about the functioning of the accelerator detector system and the progress of the experiment, so that changes and corrections may be made.

In the Phase 2 processing (interactive analysis), the architecture enables an efficient implementation of the second-level analysis of the data. This involves using a high-speed cache like DPSS as a large "window" on the tape-based data in the tertiary storage system in order to support the use of both local and remote computational resources (Figure 4.3). Prototype versions of this architecture have been successfully tested [295].

The issues raised in this environment include the use of distributed caches, the organization of the cache, the various interfaces to the cache, the management of the movement of data to and from the tertiary storage systems, and the management of the cache components in a wide area network.
4.2.4 Electron Microscopy: Control-Centered Systems

Our final example concerns the remote control of an electron microscope. An evolutionary step in multimedia systems is for them to provide a computational framework for the extraction of information from images and video sequences in real time. This information can then be used to manipulate experiments or to perform other operations, based on the information content of the images. This realtime analysis enables semiautonomous remote control based on the image content. One such application of this approach is a system for remote operation of in situ microscopy. A testbed for this approach is a 1.5 MeV transmission electron microscope, shown in Figure 4.4, that is operated by the National Center for Electron Microscopy (NCEM).
In situ microscopy refers to a class of scientific experiments in which a specimen is excited by external stimuli and the response must be either observed or controlled. The stimuli could, for example, be in the form of temperature variation or stress in the sample environment. The interaction of the external stimuli and specimen can result in sample drift, shape deformation, changes in object localization, changes in focus, or simply anomalous specimen responses to normal operating conditions. Currently, during the in situ experiments the operator must constantly adjust the instrument to maintain depth of focus and compensate for various drifts. These activities are labor intensive and error-prone, require a high-bandwidth video link to the operator, and are nearly impossible to do in wide area networks because of limited network bandwidth.

For example, a class of in situ electron microscopy experiments requires dynamic interaction with the specimen under observation as it is excited with external stimuli (e.g., temperature variation, EM field variation). The dynamic operations include control of the sample's position and orientation under the electron beam and the illumination conditions and focus. Remote control via wide area networks like the Internet that do not offer realtime data and command delivery guarantees are not practical for the finely tuned adjustments that dynamic studies require.

Enabling remote control of dynamic experiments involves such tasks as separating the basic human interaction of establishing control system parameters like gross positioning and identifying objects of interest (that do not require low-latency interaction) from the control servoing that performs operations like autofocus, object detection, and continuous fine positioning to compensate for thermal drift.

The human interaction operations, together with the supporting human communication involving video and audio teleconferencing, can easily be performed in a wide area network environment [194, 367]. The dynamic control operations, on the other hand, must occur in a much more controlled environment where the control operation and the monitored response to the control or stimuli have to be coupled by low-latency communication that is not possible in wide area networks. Therefore, dynamic remote-control applications usually involve automated control operations performed near the instrument, in order to eliminate the wide area network realtime delivery requirement.

This approach requires determining the type of servo loops needed to enable remote operation and collaboration, and the implementation of a control architecture. The basic aspect of the architecture is a partitioning that separates the low-frequency servo loop functions that enable direct human interaction performed over the wide area network from those functions that require
4.5 Remote, semiautonomous, dynamic experiment operation architecture.

low-latency control and are performed locally by using automated techniques (see Figure 4.5). The approach hides the latencies in the wide area network and permits effective remote operation. The result is telepresence that provides the illusion of close geographical proximity for in situ studies. With this approach, the testbed 1.5 MeV transmission electron microscope can now be used online via the global Internet.

In the case of image-based instrumentation, control may be automated by using computer vision algorithms that permit instrumentation adjustments to be made automatically in response to information extracted from the video signal generated by the microscope imaging system. Thus, by relieving the operator of having to do the dynamic adjustment of the experimental setup, remote collaboration and remote operation of the in situ studies over a wide area network are made possible. The computational vision techniques that support remote in situ microscopy applications include image compression, autofocusing, self-calibration, object detection, tracking by using either high-level or low-level features, and servo loop control mechanisms [441, 443].
The image content analysis that provides the information that is fed back to the control system is automated and performed in the environment local to the instrument. That is, the computers that acquire and analyze the video images and then communicate with the control system are all connected by fast local area networks. The set points that initialize the servo loops—the selection of objects of interest and the parameters of external forcing functions, as well as the monitoring of the experiment—may be carried out in a wide area network environment.

The microscope and experiment control interface, a typical image, and the results of video content analysis for shape and drift velocity are illustrated in Figure 4.6.

The main issues that are raised by this sort of remote operation are the servoing architecture and the algorithms used for information extraction and control [442].

4.2.5 Summary

The four examples presented in this section (a media-rich instrument control environment, a health care imaging system doing autonomous data collection...
and cataloging, a high-data-volume physics experiment environment, and a semiautonomous control system) illustrate several aspects of remote operation and expose some of the capabilities that will be needed to support routine construction and use of these types of systems in the future. In summary, the online angiography system requires automated management of data streams, the use of a network cache, automatic cataloging, and distributed access control. These, in turn, require semiautonomous monitoring and QoS guarantee mechanisms in the network and in the processing and storage systems. The STAR detector scenario uses a widely distributed configuration of the network cache and distributed management of computational resources and data. The shared interface example of the Advanced Light Source Beamline 7 requires reliable multicast in wide area networks and rich-media management mechanisms. All of the examples require distributed management of system resources and of distributed access control, both for security and for the “distributed enterprise” management of users and resources. In the next section we examine some approaches to providing these capabilities.

In addition, most of the scenarios would potentially benefit from bandwidth adaptive interface features; the Beamline 7 and microscopy scenarios are candidates for dynamic system construction with brokered resources to support their transient needs for significant computational resources. These desired capabilities are, however, not addressed in our current systems.

4.3 ISSUES, CAPABILITIES, AND FUTURE DIRECTIONS

In this section we describe some of the architectural and middleware approaches that are proving useful, and sometimes critical, in implementing high-performance distributed instrumentation and data systems.

4.3.1 A Model Data-Intensive Architecture

In our research, we have demonstrated the utility, in the automated cataloging and high-data-rate application domains, of using a high-speed distributed cache as a common element for all of the sources and sinks of data involved in high-performance data systems. This cache-based approach provides standard interfaces to a large, application-oriented, distributed, online, transient storage system. Each data source deposits its data in the cache, and each data consumer takes data from the cache, usually writing the processed data back to the cache. In almost every case there is also a tertiary storage system manager that migrates data to and from the cache at various stages of processing (see Figure 4.7).
For the various data sources and sinks, the cache, which is itself a complex and widely distributed system, provides a standardized approach for high-data-rate interfaces; an “impedance”-matching function (e.g., between the coarse-grained nature of parallel tape drives in the tertiary storage system and the fine-grained access of hundreds of applications); and flexible management of online storage resources to support initial caching of data, processing, and interfacing to tertiary storage.

Depending on the size of the cache relative to the objects of interest, the tertiary storage system management (object manager + archive data mover of Figure 4.7) may involve only moving partial objects to the cache; that is, the cache is a moving window for the offline object/data set. The application interface to the cache can support a variety of I/O semantics, including UNIX disk I/O semantics (i.e., upon posting a read, the available data is returned; requests for data in the data set but not yet migrated to cache cause the application-level read to block until the data is migrated from tape to cache).

Generally, the cache storage configuration is large compared with the available disks of a typical computing environment, and very large compared with any single disk (e.g., hundreds of gigabytes).
4.3.2 Network-Based Caches

DPSS serves several roles in high-performance, data-intensive computing environments. This application-oriented cache provides a standard interface for high-speed data access and provides the functionality of a single, very large, random-access, block-oriented I/O device (i.e., a “virtual disk”). It provides high capacity (we anticipate a terabyte-sized system for physics data) and isolates the application from the tertiary storage system. Many large datasets can be logically present in the cache by virtue of the block index maps being loaded even if the data is not yet available. In this way processing can begin as soon as the first data has been migrated from tertiary storage.

Generally speaking, DPSS can serve as an application cache for any number of high-speed data sources (instruments, multiple mass storage systems, etc.). The naming issue (e.g., resolving independent name space conflicts) is handled elsewhere. For example, in the online health care imaging system mentioned above, the name space issue is addressed by having all of the data represented by Web-based objects that are managed by WALDO [296]. At the minimum, WALDO provides globally unique naming and serves as a mechanism for collecting different sources of information about the data. The Web object system can also provide a uniform user (or application) front end for managing the data components (e.g., migration to and from different mass storage systems), and it manages object use conditions (PKI access control [297]).

DPSS provides several important and unique capabilities for the distributed architecture. The system provides application-specific interfaces to an extremely large space of logical blocks (16-byte indices). It may be dynamically configured by aggregating workstations and disks from all over the network (this is routinely done in the MAGIC testbed [476] and will in the future be mediated by the agent-based management system). It offers the ability to build large, high-performance storage systems from inexpensive commodity components. It also offers the ability to increase performance by increasing the number of parallel DPSS servers. A cache management policy module operates on a per-data-set basis to provide block aging and replacement when the cache is serving as a front end for tertiary storage.

The high performance of DPSS—about 10 MB/s of data delivered to the user application per disk server—is obtained through parallel operation of independent network-based components. Flexible resource management—dynamically adding and deleting storage elements, partitioning the available storage, and so on—is provided by design, as are high availability and strongly bound security contexts. Scalability is provided by many of the same design features that provide the flexible resource management (that provides the
Agent-based management (storage server and network state vis-à-vis applications)

Disk servers (block storage and block-level access control)

Physical block requests

Agent-based management (redundant masters)

DPSS master
Resource manager (disk resources)

Request manager (logical-to-physical name translation, cache management)

Data set manager (user security, data set access control, etc.)

Security context 1 (system integrity and physical resources)

Returned data stream ("third-party" transfers directly from the storage servers to the application)

Security context 2 (data use conditions)

Application (client)

Memory buffer

Data requests

Application data access methods (data structure to logical block-id mapping)

DPSS API (client-side library)

Agent-based management (data sets, metadata locations, etc.)

Logical block requests

Data stream manager

DPSS architecture.

4.8

FIGURE

capability to aggregate dispersed and independently owned storage resources into a single cache).

When datasets are identified by the object manager (e.g., as in Figure 4.7) and are requested from tertiary storage, the logical-to-physical block maps become immediately available. The data mover operates asynchronously, and if an application "read" requests a block that has not yet been loaded, the application is notified (e.g., the read operation blocks). At this point the application can wait or request information on available blocks in order to continue processing.

While the basic interface provides for requesting lists of named logical blocks, many applications use file I/O semantics, as provided in the DPSS client-side interface library.

The internal architecture of DPSS is illustrated in Figure 4.8. Typical DPSS implementations consist of several low-cost workstations, each with several disk controllers, and several disks on each controller. A three-server DPSS
can thus provide transparent parallel access to 20–30 disks. The data layout on the disks is completely up to the application, and the usual strategy for sequential reading applications is to write the data “round-robin” (stripe across servers). Otherwise, the most common strategy is to determine the physical block locations randomly when they are written. Our experience has shown that, with the high degree of parallelism provided at the block level when a DPSS is configured from, say, 30 disks spread across three servers, random placement of blocks provides nearly optimal access time for a wide range of read patterns.

DPSS provides several features to ensure that distributed caches provide significant value to the remote operation and computational grid environments. These features include agent-managed dynamic reconfiguration (i.e., adding and deleting servers and storage resources during operation), agent-managed replication (of data, name translation, and disk servers) for reliability and performance, data block request semantics to support application data prediction, and application access semantics (e.g., a large block index space allows encoding of some application information, such as longitude, latitude, and elevation for tiled geographical image data).

4.3.3 Agent-Based Management and Monitoring

The combination of generalized autonomous management of distributed components and accurate monitoring of all aspects of the environment in which data moves has turned out to be a critical aspect of the debugging, evaluation, adaptation, and management of widely distributed high-data-rate applications.

In widely distributed systems, when we observe that something has gone wrong, it is generally too late to react. In fact, we frequently cannot even tell what is wrong, because the problem depends on a history of events or because the needed information is no longer accessible or because it will take too long to ask and answer all of the required questions.

An agent-based approach for analysis of the operation of distributed applications in high-speed wide area networks can be used to monitor and identify all of the factors that affect performance and to isolate the problems arising from individual hardware and software components. Agents not only can provide standardized access to comprehensive monitoring, but they can also perform tasks such as keeping a state history in order to answer the question, How did we get here? Active analysis of operational patterns (e.g., pattern analysis of event-based lifeline traces) will lead to adapting behavior/configuration to avoid or correct problems.
Monitoring

One successful monitoring methodology involves recording every event of potential significance together with precision timestamps, and then correlating events on the basis of the logged information. This allows constructing a comprehensive view of the overall operation, under realistic operating conditions, revealing the behavior of all the elements of the application-to-application communications path in order to determine exactly what is happening within complex distributed systems. This approach has been used in the DPSS distributed storage system and its client applications. As data requests flow through the system, timestamps and log records are generated at every critical point. Network and operating system monitoring tools are used to log additional events of interest using a common format. This monitoring is designed to facilitate performance tuning, distributed application performance research, the characterization of distributed algorithms, and the management of functioning systems (by providing the input that allows adaptation to changes in operating conditions). The approach allows measuring of network performance in a manner that is a much better real-world test than, for example, ttcp, and allows us to accurately measure the dynamic throughput and latency characteristics of our distributed application code—top to bottom and end to end [295].

This sort of detailed monitoring is also a practical tool for system-level problem analysis, as has been demonstrated in the analysis of a TCP over ATM problem that was uncovered while developing the monitoring methodology in the ARPA-funded MAGIC gigabit testbed (a large-scale, high-speed ATM network [540]).

The high-level motivation for this work is twofold. First, when developing high-speed, network-based distributed services, we often observe unexpectedly low network throughput and/or high latency. The reason for the poor performance is frequently not obvious. The bottlenecks can be (and have been) in any of the components: the applications, the operating systems, the device drivers, the network adapters on either the sending or receiving host (or both), the network switches and routers, and so on. It is difficult to track down a performance problem because of the complex interaction between the many distributed system components and the fact that problems in one place may be most apparent somewhere else. A precision and comprehensive monitoring and event analysis methodology is an invaluable tool for diagnosing such problems (see Chapters 14 and 15).

Second, such monitoring is one aspect of an approach to building predictable high-speed components that can be used as building blocks for high-
performance applications, rather than having to tune the applications top to bottom, as is all too common today. Continuous and comprehensive monitoring can provide the basis of adapting distributed system behavior to “congestion” in processing, storage, and communication elements.

**Agent-Based Management of Widely Distributed Systems**

If comprehensive monitoring is the key to diagnosis, agent-based management may be the key to keeping widely distributed systems running reliably.

In one prototype system [568], “agents” are autonomous adaptable monitors, managers, information aggregates, and Knowledge Query and Manipulation Language (KQML)-based information filters implemented in Java and constantly communicating with peers and resources. Initial experimentation with such agents in DPSS indicates several potential advantages (see Figure 4.9).

The first is structured access to current and historical information regarding the state of DPSS components.

The second is reliability. Not only does this system keep track of all components within the system, but it restarts any component that has crashed, including one of the other agents (addresses fault tolerance). “Associated” agents communicate with each other using IP multicast.

A third advantage is automatic reconfiguration. When new components (such as a new disk server) are added, the agents do not have to be reconfigured. Rather, an agent is started on the new host, and it will inform all other agents about itself and the new server. Brokers and agents may discover interesting new agents via a dynamic directory protocol like SDR or by the support of reliable multicast protocols that provide interagent communication. Following discovery, the new agent—and the resource that it represents—is added to the configuration.

Information management is a fourth potential advantage of agent-based management. Broker agents manage information from a collection of monitor agents—usually on behalf of a user client—and provide an integrated view for applications. For example, the Java-based graphical status interface illustrated in Figure 4.10 shows aggregated information from DPSS system-state and data-set-state monitoring agents—the two elements emphasized in Figure 4.9. Agents manage data-set metadata—dynamic state, alternate locations, tertiary location—at each storage system as well as the state of all network interfaces and data paths and the load of each DPSS disk server.

A fifth advantage is user representation. Brokers can perform actions on behalf of a user. For example, if a data set is not currently loaded onto a DPSS
An agent-based monitoring architecture that addresses adaptive operation, reliability/survivability, and dynamically updated metadata for data repositories. 1, distributed system management; 2, data state management.

(which is typically used as a cache), the broker can cause the data set to be loaded from tertiary storage.

The broker/agent architecture also allows for efficient system administration. In particular, rule-based operation of the agent can be used to determine what policies are to be enforced while remaining separate from the actual mechanism used to implement these policies.

Finally, agent-based management provides flexible functionality. New agent methods can be added at any time. For example, the brokers have an algorithm for determining which DPSS configuration to use based on a set of parameters that include network bandwidth, latency and disk server load. This algorithm can be modified on the fly by loading new methods into the agents. Related agents are part of the same security context, and new code/methods presented to the agents are cryptographically signed for origin verification and integrity.
4.3 Issues, Capabilities, and Future Directions

4.3.4 Policy-Based Access Control

Collaborative distributed environments that involve multiuser instruments at national facilities, widely distributed supercomputers and large-scale storage systems, data sharing in restricted collaborations, and network-based multimedia collaboration channels give rise to a range of requirement for distributed access control. For example, administration of such resources as network QoS
will need to be handled by an automated authorization infrastructure so that management of both resource availability and allocation, as well as subsequent enforcement of use conditions, can be done automatically and without recourse to a central or single authority.

In all of these scenarios, the resource (data, instrument, computational and storage capacity, communication channel) has multiple stakeholders (typically the intellectual principals and policy makers), and each stakeholder will impose use conditions on the resource. All of the use conditions must be met simultaneously in order to satisfy the requirements for access. This model is common in society and is illustrated in Figure 4.11.

Further, scientific collaborations often are diffuse, with the principals and stakeholders being geographically distributed and multiorganizational. Therefore, the access control mechanism must accommodate these circumstances by providing distributed management of policy-based access control for all
resources; authentication, integrity, confidentiality, and so on of resource-related information; and mechanisms supporting the internal integrity of distributed systems.

We also anticipate that the resulting infrastructure will support automated brokering and policy-based negotiation for resources.

Goals

The goal for access control in such distributed environments is to reflect, in a computing and communication-based working environment, the general principles that have been established in society for policy-based resource access control.

All responsible entities—principals and stakeholders—should be able to make their assertions (as they do now by signing, for example, a policy statement) without reference to a mediator, and especially without reference to a centralized mediator (e.g., a system administrator) who must act on their behalf. The mechanism must be dynamic and easily used, while maintaining strong assurances. Only in this way will computer-based security systems achieve the decentralization and utility needed for the scalability to support large distributed environments.

The computer systems-based resource access control mechanisms should be able to collect all of the relevant assertions (stakeholder use conditions and corresponding attributes) and make an unambiguous access decision without requiring entity-specific or resource-specific local, static configuration information that must be centrally administered. (This requirement does not imply that such specific configuration is precluded, only that it should not be required.) The mechanism (Figure 4.12) should also be based on, and evolve with, the emerging, commercially supplied public-key certificate infrastructure components.

Expected Benefits

For security to be successful in distributed environments, providing both protection and policy enforcement, each principal entity should have the same involvement as in the currently established procedure in the absence of computer security—no more, no less. That is, those who have the authority to set access conditions or use conditions, by, for example, holographically signing statements in a paper environment, will digitally sign functionally equivalent statements in a distributed computing-based environment. The use of these
Credentials should be automatic, and the functions of checking credentials, auditing, and so on should be performed by appropriate entities in either circumstance.

The expected advantages of computer-based systems are in maintaining access control policy, but with greatly increased independence from temporal and spatial factors (e.g., time zone differences and geographic separation), together with automation of redundant tasks such as credential checking and auditing. The intended outcome is that the scientific community will more easily share expensive resources, unique systems, and sensitive data. A further expected benefit is that this sort of security infrastructure should provide the basis of automated brokering of resources that precede the construction of dynamically and just-in-time configured systems to support, for example, scientific experiments with transient computing, communication, or storage requirements.
Authorization-Based Distributed Security

An approach that addresses the general goals noted above can be based on authorization and attribute certificates. These digitally signed documents have the characteristic that they assert document validity without physical presence of the signer or physical possession of holographically signed documents. The result is that the digitally signed documents that provide the assertions of the principals, stakeholders, attribute authorities, and so on may be generated, represented, used, and verified independent of time or location.

Other parts of the approach are implemented through the use of “authorities” that provide delegation mechanisms and assured information as digitally signed documents: identity authorities connect human entities and systems to digital signatures; stakeholder authorities provide use conditions; attribute authorities attest to user characteristics. Additional components include reliable mechanisms for generating, distributing, and verifying the digitally signed documents; mechanisms that match use conditions and attributes; and resource access control mechanisms that use the resulting credentials to enforce policy for the specific resource. (For a general introduction to public-key infrastructure, see [196, 493].)

Architecture for Distributed Management of Fine-Grained Access Control

A prototype implementation [297] that is addressing distributed management of access control to limited, valuable, or large-scale resources, data, and objects (e.g., large scientific instruments, distributed supercomputers, sensitive but unclassified databases) is providing some experience with decentralized security environments. The prototype includes fully distributed resource management and access. In our target environment, the resource users, resource owners, and other stakeholders are remote from the protected resource—the norm in large-scale scientific instrument environments, among others. In the prototype, all significant resources have multiple stakeholders, all of whom provide their own use conditions, which are specified in the environment of the stakeholder and then provided to the resource access control mechanism. At the heart of the prototype is an attribute-based access policy. Users are permitted access to resources based on their attributes that satisfy the stakeholder use conditions. These attributes are attested to by trusted third parties. Validation of the right of access is typically used to establish the security context for an underlying security system such as SSL (e.g., between Web browser and servers [268]) and GSS (secure messaging between components of distributed systems [347]).
The prototype provides for objects, data, resource owners, and other stakeholders to be able to remotely exercise control over access to the resource, for legitimate users (those that satisfy the use conditions of the resource stakeholders) to obtain easy access, and for unqualified or unauthorized users to be strongly denied access. The architecture is illustrated in Figure 4.12.

In addition to the technology issues of integrity and management of the access control system and associated computing platforms, useful security is as much (or more) a deployment and user ergonomics issue. That is, the problem is as much trying to find out how to integrate good security into the end user (e.g., scientific) environment so that it will be used, trusted to provide the protection that it claims, easily administered, and genuinely useful in the sense of “providing distributed enterprise capabilities” (that is, providing new functionality that supports distributed organizations and operation) as it is trying to address the more traditional security issues.

While the security architecture provides the basic technology, in order to accomplish a useful service the architecture must be applied in such a way that the resources are protected as intended by the principals. This involves understanding the information/resource use and structure model and developing a policy model that will support the intended access control. These must be supported by a security model that specifies how the elements of the security architecture and infrastructure will implement the policy model.

A prototype implementation of this architecture [297] provides a policy engine that implements both flat and hierarchical multiple-use-condition policy models, uses X.509 identity certificates and ad hoc attribute and use-condition certificates obtained from Web and LDAP servers, and provides a policy evaluation service to the Apache Web server and an implementation of SPKM/GSS.

ACKNOWLEDGMENTS

The material presented in this chapter represents the work of numerous people: in particular, Deborah Agarwal, Bahram Parvin, Mary Thompson, and Brian Tierney, as well as the author and others at Lawrence Berkeley National Laboratory. More information may be found at www.itg.lbl.gov.

Physicists Craig Tull and Doug Olson are our collaborators in the STAR project; Joe Terdiman and Bob Lundstrum of Kaiser Permanente and Evert-Jan Pol of Philips Research are our collaborators for the cardioangiography project; Brian Tonner of the University of Wisconsin–Milwaukee is our collaborator on the ALS Beamline 7, Spectro-Microscopy Collaboratory; and Ulrich Dahmen is our collaborator at the NCEM.
We also acknowledge Stewart C. Loken, division director of the Information and Computing Sciences Division, LBNL, for his long-term support and contributions to this work specifically, and the idea of collaboratories generally.

The work described here is supported by the U.S. Department of Energy, Energy Research Division, Mathematical, Information, and Computational Sciences and ER-LTT offices, under contract DE-AC03-76SF00098 with the University of California, and by DARPA, ISTO.

FURTHER READING

For more information on the topics covered in this chapter, see www.mkp.com/grids and also the following references:

- The National Research Council report [410] first described the concept of a collaboratory.
- Craig Partridge’s book [440] discusses gigabit networking and includes useful bibliographical references.