Using High Performance Network to Enable Distributed Environments for Large Data-Objects[1]

William Johnston, Jin Guojun, Jason Lee, Brian Tierney, Mary Thompson
Imaging and Distributed Computing Group
Lawrence Berkeley National Laboratory
University of California
Berkeley, CA, 94720

Abstract

We are exploring the feasibility of using highly distributed architectures for all aspects of collecting, storing, analyzing, and otherwise manipulating and making generally available, large data-objects. These objects - typically the result of a single operational cycle of an instrument, and of sizes from tens of MBytes to tens of Gbytes - are the staple of modern analytical systems. It is further the case that many of the instrumentation systems that generate such data-objects are used by a diverse and geographically distributed community: Examples from the scientific community include physics and nuclear science high energy particle accelerators and detector systems, large electron microscopes, ultra-high brilliance X-ray sources, etc. There are correspondingly complex instrumentation systems in the health care community that generate large data-objects.

In this paper, we describe some of the general aspects of such systems, and specifically describe the use of a shared (public) IP over ATM network to facilitate collection, storage, analysis, distribution, and delivery of several kinds of data.

1.0 Introduction

The advent of shared, widely available high speed networks is providing the potential for new approaches to the collection, storage, and analysis of large data-objects. Health care information, especially high volume image data used for diagnostic purposes - e.g. X-ray CT, MRI, and cardio-angiography - that are increasingly collected at tertiary (centralized) facilities, may now be routinely stored and used at locations other than the point of collection. The importance of distributed storage is that a hospital (or any other instrumentation environment) may not be the best environment in which to maintain a large-scale digital storage system, and an affordable, easily available, high bandwidth network can provide location independence for such storage. The importance of remote, end-user access is that the health care professionals at the referring facility (frequently remote from the tertiary imaging facility) will have ready access to not only the image analyst's reports, but the original image data itself.

This general scenario extends to many fields other than health care. In particular, the same basic infrastructure is required for remote access to large-scale scientific and analytical instruments, both for data handling and for direct, remote-user operation. See [Johnston95b].

1. This work is supported by the U. S. Dept. of Energy, Energy Research Division, Mathematical, Information, and Computational Sciences office (http://www.er.doe.gov/production/octr/mics), under contract DE-AC03-76SF00098 with the University of California, and by ARPA ITO (http://ftp.arpa.mil/ResearchAreas.html). Author's address: 50B-2239, Lawrence Berkeley National Laboratory, Berkeley, CA 94720. Tel: +1-510-486-5014, fax: +1-510-486-6363, wejohnston@lbl.gov, http://www-itg.lbl.gov/~johnston. This document is report LBNL-nnnn.

In this paper we describe and illustrate a set of concepts that are contributing to a generalized, distributed information infrastructure, especially as it concerns the types of large data-objects generated in the scientific and medical environments. We will describe the general issues, architecture, and some system components that are currently in use to support distributed large data-objects, and will use a specific health care information system as an example.

The basic elements of the overall architecture include:

- data collection and the instrument-network interface
- on-line storage that is distributed throughout the network (for both performance and reliability)
- processing elements - also distributed throughout the network - for various sorts of data analysis
- data management that provides for the automatic cataloguing (metadata generation) of the data being stored
- data access interfaces, including application-data interface
- tertiary storage (“mass storage”) management
- user access to all relevant aspects (application, data, metadata, data management)
- transparent security that provides access control for all of the systems components based on the resource-owner’s policy

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

2.0 System Architecture

2.1 Data Collection

Instrumentation systems the front-end of many distributed large data-object environments. Examples include particle accelerator detectors, Earth environment monitoring satellites, and medical imaging systems. These sources generate essentially continuous data streams, but ones that have “natural” boundaries that define “objects”: Video sources (like the cardio-angiography studies where data is collected in bursts; large single images; detector “event” records etc. We make this “natural boundary” characterization to distinguish these data types from continuous-media data (such as video and audio multimedia streams): Even though the average data rates may be comparable, or even higher in the large data-object environment, the temporal synchronization requirements and steady-state nature of multimedia streams give them a somewhat different character.

For many of the instrumentation systems that we are interested in, one of the primary issues is getting the data out of the instruments and on to the network. One of the circumstances that has lead to both the interest in, and practicality of, the architecture being described here is that general purpose workstations now have memory and I/O bus structures that are fast enough to acquire, structure, and send to the network, significant bandwidth data streams. This is an important capability because it means that the only special hardware that is required to bring instruments on-line is the interface between the workstation and the data source, and even this interface may be able to be done in “software” using off-the-shelf DSP-based I/O boards.

The frontend workstation acquires the raw data, formats it as “objects” by adding or using metadata from the experiment environment, and then sends the objects into the distributed environment. The data collection workstation frequently also serves as a buffer so that brief interpretations or

slow-downs in the network do not result in loosing data. This frontend architecture is illustrated in

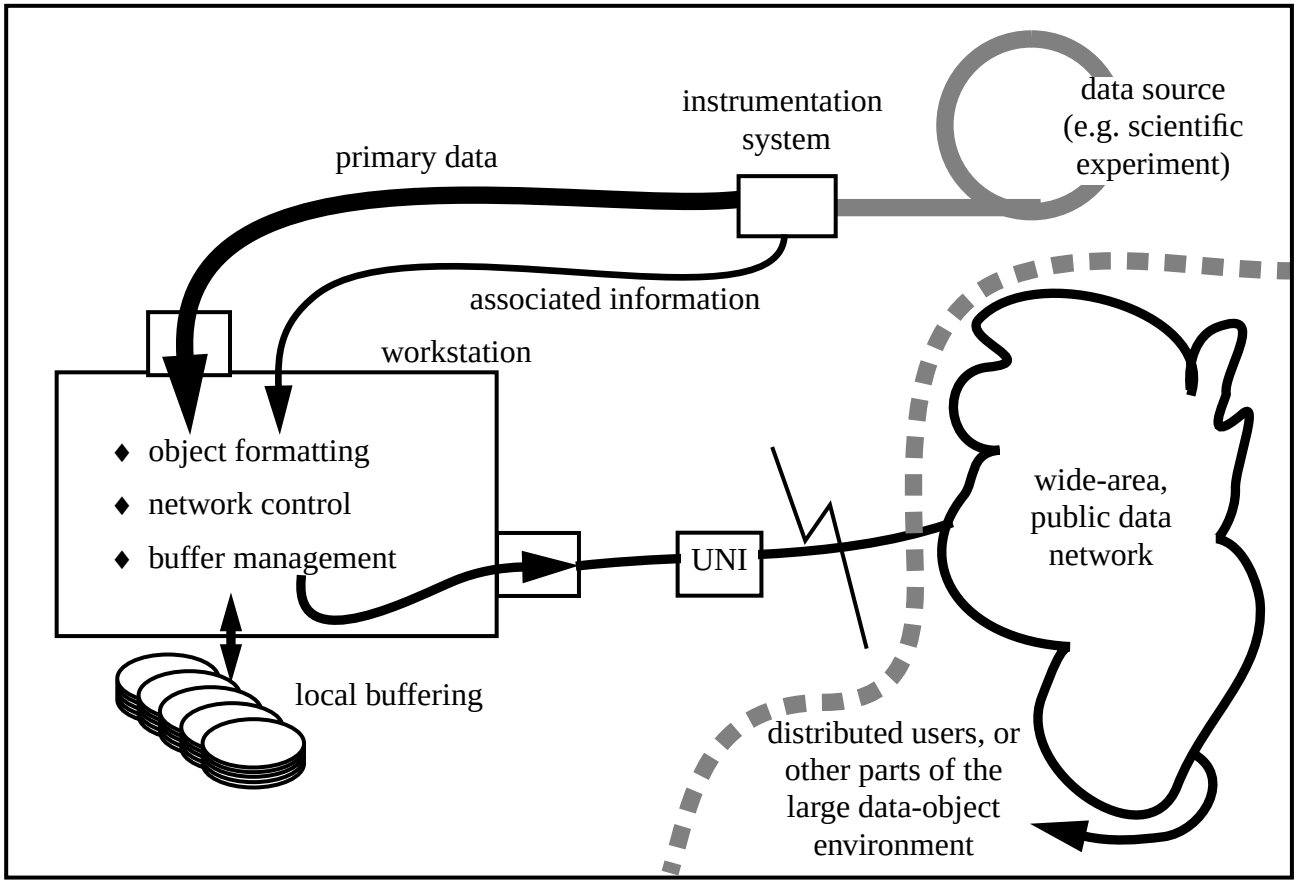

Figure 1 Architecture of the Frontend of a Distributed, Large Data-Object Environment

Figure 1.

2.2 Distributed Storage: The Distributed-Parallel Storage System

Widely distributed network storage systems are an essential component of a network-based large data-object environment.

The idea of such systems is that by distributing the components of a storage system throughout the network, that capacity, reliability, performance, and security are all increased. Capacity increases in conjunction with a widely deployed, generalized security infrastructure that can support dynamic brokering for resources. (See [Johnston96a].) Reliability increases because storage systems that can be dynamically configured from components that have as little as possible in common (e.g. location) provide the resilience that comes from independence. Performance is increased by the combined characteristics of parallel operation of many sub components, and the independent data paths provided by a large network infrastructure. Security is also potentially increased by having many independent components each of which has local and independent enforcement mechanisms that can limit the scope of a security breach.

The Distributed-Parallel Storage System (DPSS, aka ISS) is an experimental system in which we are developing, implementing, and testing these ideas. In most configurations, it is used as a network-striped disk array designed to supply and consume high speed data streams to and from other processes in the network. (See [DPSS].)

2.2.1 DPSS Architecture

The DPSS is essentially a “logical block” server whose functional components are distributed across a wide area network. (See Figure 2 illustrating the DPSS architecture.) The DPSS uses parallel operation of distributed servers to supply image streams fast enough to enable various multi-user, “real-time”, virtual reality-like applications in an Internet / ATM environment. There is no inherent organization to the blocks, and in particular, they would never be organized sequentially on a server. The data organization is determined by the application as a function of data type and access patterns, and is implemented during the data load process. The usual goal of the data organization is that data is declustered (dispersed in such a way that as many system elements as possible can operate simultaneously to satisfy a given request) across both disks and servers. This strategy allows a large collection of disks to seek in parallel, and all servers to send the resulting data to the application in parallel, enabling the DPSS to perform as a high-speed image server.

The implementation is based on the use of multiple low-cost, medium-speed disk servers which use the network to aggregate multiple server outputs for high performance applications. To achieve high performance all types of parallelism are exploited, including those available at the level of the disks, controllers, processors / memory banks, servers, and the network (See Figure 3 illustrating the DPSS implementation).

At the application level, the DPSS is a persistent cache of named objects, and at the storage level it is a logical block server. Although not strictly part of the DPSS architecture, the system is usually used with an application agent called a “data set structure server”. This component provides an object-like encapsulation of complex user-level data structures so that the application does not have to retain this information for each different data set. The function and interface of this structure server are left to the application domain, but one simple example is for video data. In this case the structure server allows applications to request data by “frame” number and “type” (e.g. in the video-angiography application, there are two images - 90 degree offset views - associated with every frame). The structure server converts the application requests into logical block requests. These logical block requests are then sent to the DPSS “data set manager”. The DSM maintains data set definitions, and is responsible for mapping the logical block requests to physical block requests. The DPSS “master” deals with interactions with the servers to determine available storage (a server is an independent entity and may deal with several DPSS masters) and to establish the “security context”.

The security model for the DPSS involves accommodating several different resource owners: The context established between the Master and the disk/storage server reflects an agreement between the owner of the physical resources (disks) and an agent that is providing storage to a user community. This context enforces the disk usage agreement. The context established between the DSM and the storage servers reflects the use-conditions imposed by the data “owner”, and provides for ensuring access control that enforces those use-conditions.

The overall data flow involves “third-party” transfers from the storage servers directly to the data-consuming application. (A model used by most high performance storage systems.) Thus, the application requests data, these requests are eventually translated to physical block addresses (server name, disk number, and disk block), and the servers deliver data directly to the application. (The servers and the application host are made known to each other during the data set “open” process, and sufficient information from the original request is carried along so that the application can identify the data provided by the servers.)

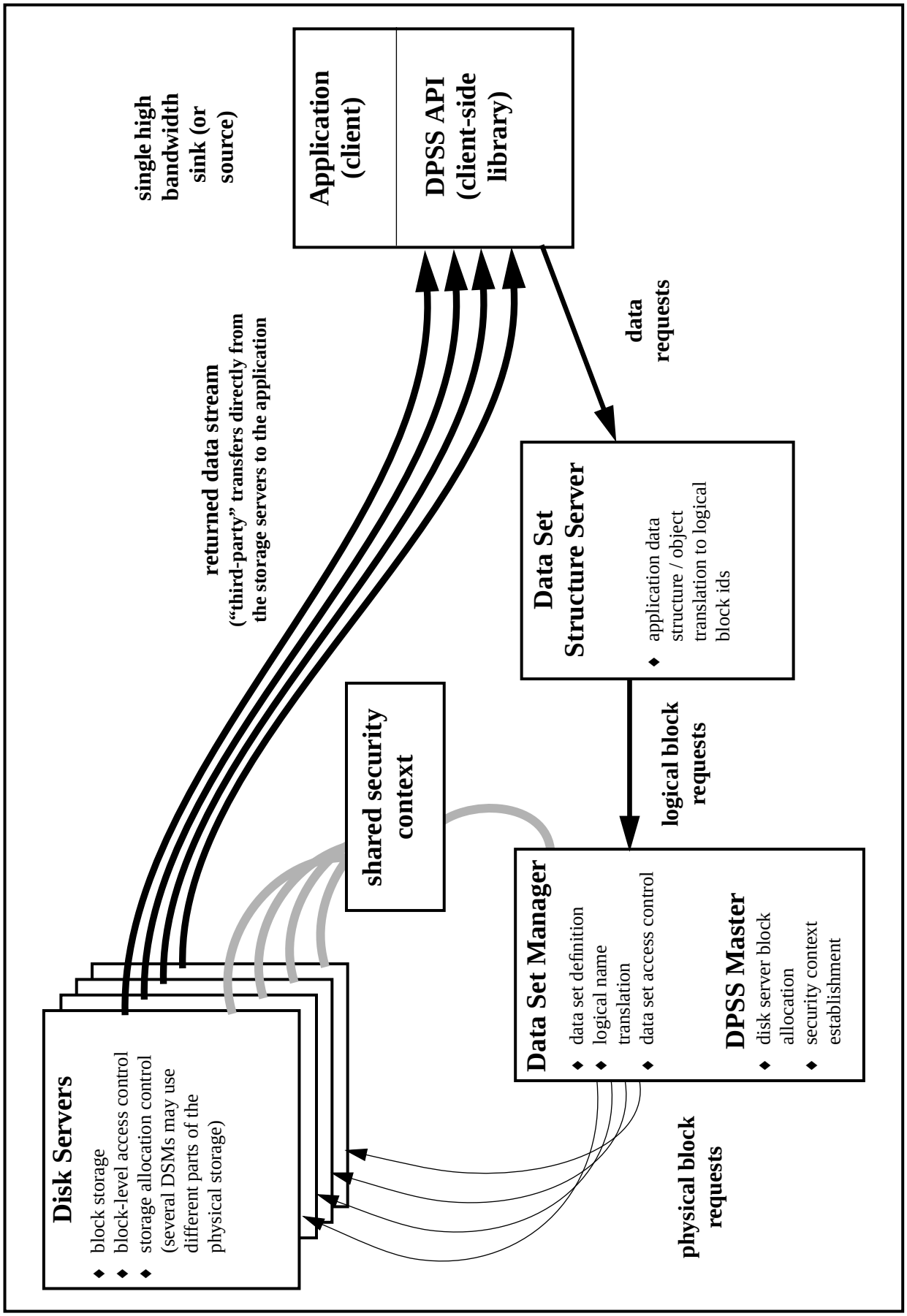

Figure 2 **Distributed-Parallel Storage System Architecture**

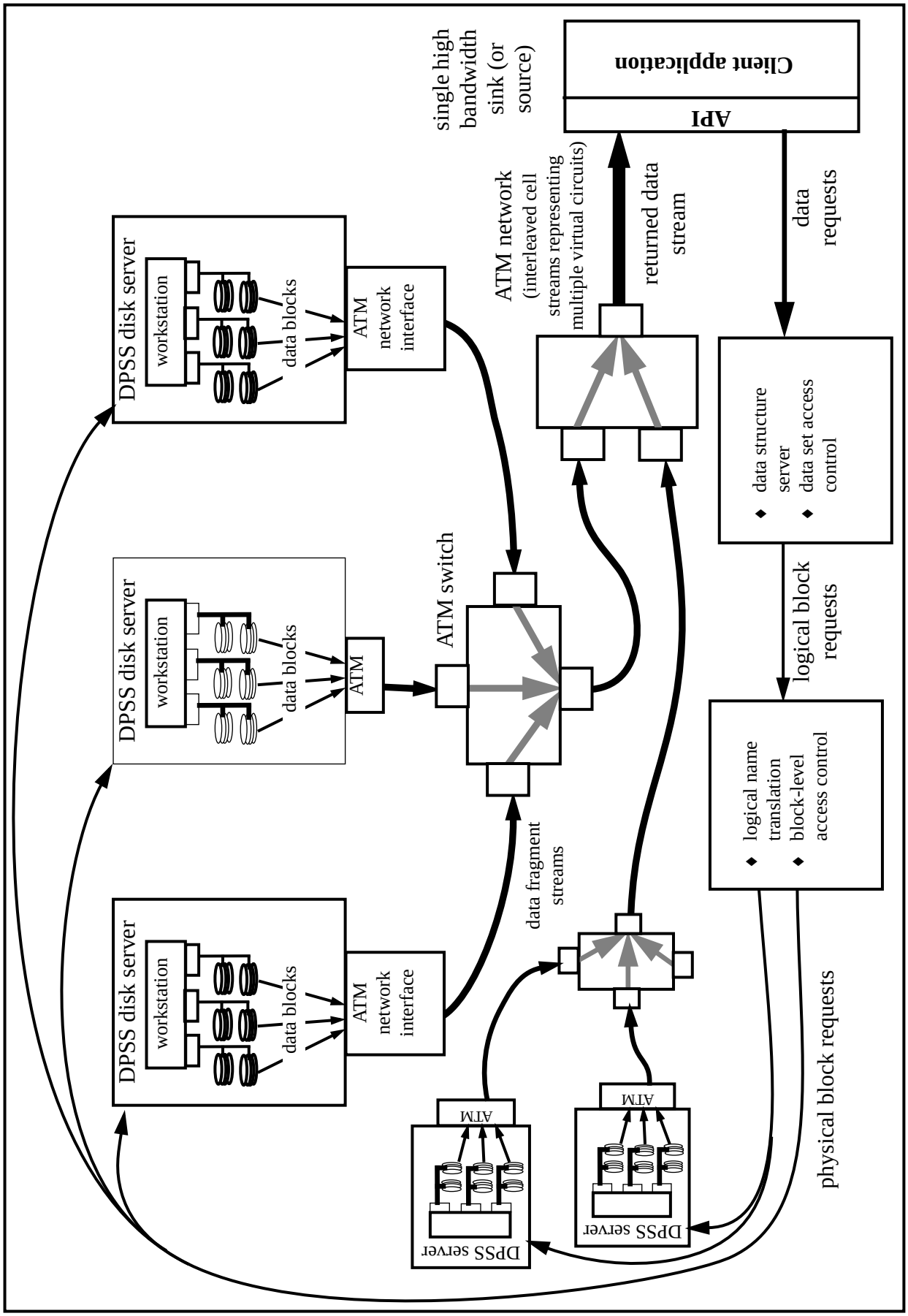

Figure 3 **Distributed-Parallel Storage System Implementation**

The DPSS is not an inherently reliable tertiary storage system, though the potential exists for this capability by adding tertiary backing storage on the storage servers.

2.2.2 DPSS and the TerraVision Application

Figure 4 illustrates the configuration and operation of the DPSS in the ARPA-funded MAGIC high speed ATM testbed [MAGIC] where the DPSS is being developed. The primary characteristics of this environment are the various networks (MAGIC backbone, Spartan [Spartan], and NTON [NTON]), the TerraVision terrain visualization application [TerraVision], and the model of a data curator (EDC). All components of the DPSS are geographically dispersed via the wide area, high speed, network infrastructure.

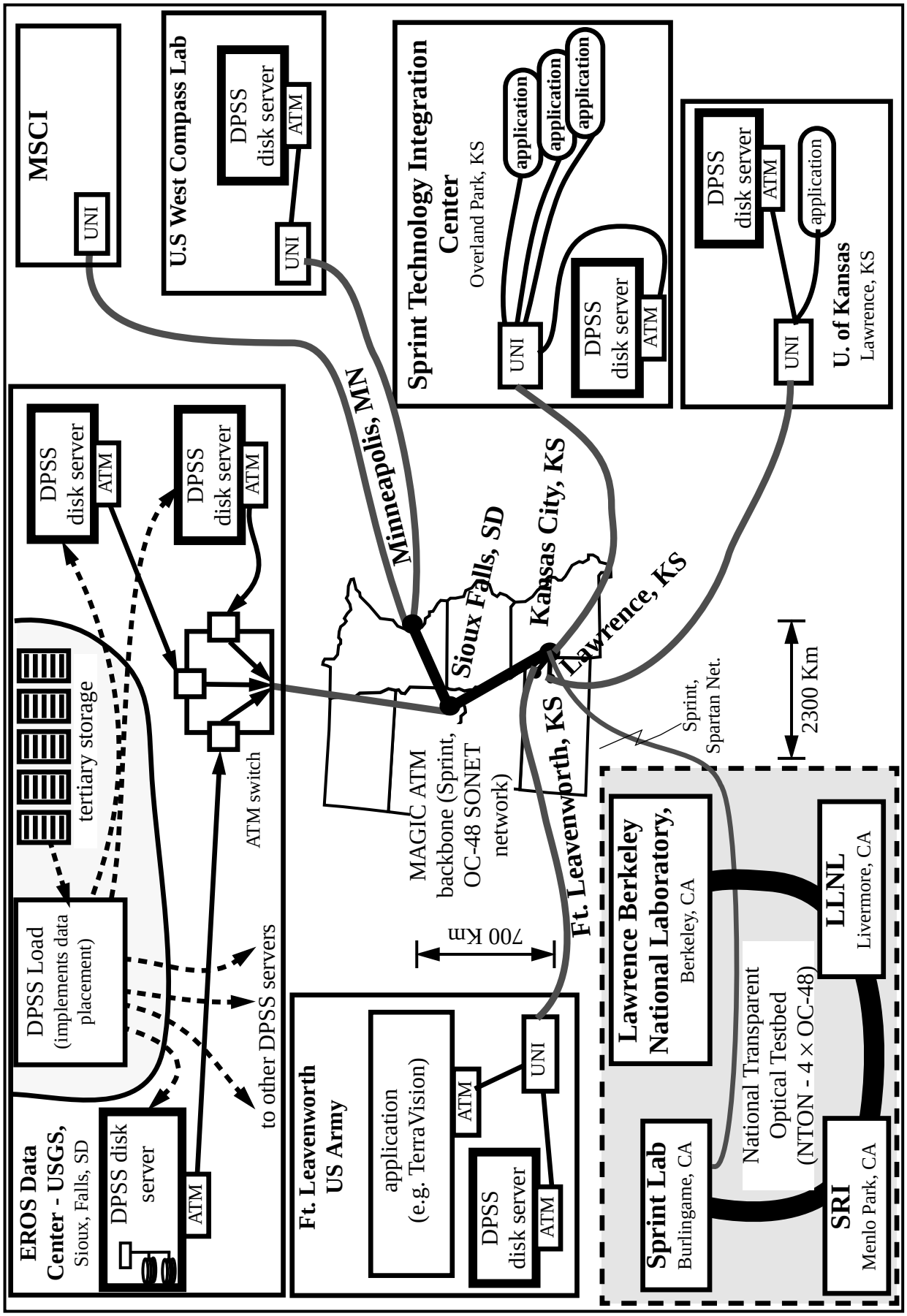

Figure 4 **The MAGIC Testbed Distributed Application Environment**

2.3 Data Management

In any scenario where data is generated in large volumes and with high throughput, and especially in a distributed environment where the people generating the data and the people cataloguing and using the data are geographically separated, a key issue is the automatic generation of at least minimal metadata, and the cataloguing of the data and the metadata as the data is received (or as close to real time as possible).

Such a capability is needed to help researchers organize, browse and search through data collections, especially collections where the original data are stored off-line in a mass storage system. The system should also facilitate co-operative research by allowing access to the data by specified users at local and remote sites, both by enabling immediate examination of the data, and thought incorporation of the data into other databases or documents.

Our approach to this capability (called “ImgLib” [ImgLib]) is to use the capabilities of the World Wide Web tools to provide a semi-automatic cataloguing of incoming data. This is done by extracting associated metadata and converting it into text records, together with generating auxiliary metadata and derived data, and combining these into Web documents.

In the case of image-like data, an ImgLib catalogue “entry” consists of (derived) thumbnail images and textual indexing material, together with pointers to the original images, video, or other data (whose permanent storage is typically off-line). The indexing material is kept on-line on a system that is running the ImgLib Web Server. The ImgLib server also directly manages a DPSS-based on-line cache, and manages migration of data from tertiary storage to cache.

Figure 5 illustrates some of these points in the context of a health care video imaging application. It shows the results of a search on the textual metadata. The information about the data-objects that result from the search is shown as an associated collection of thumbnails, associated pointers to several other types of derived data (including a “movie” representation), and a pointer to the original data-object. The data-objects are kept on tertiary storage (a tape-robot based mass storage system in this case), and there is an option for forcing migration of data back to the on-line cache if the data of interest is not already there. In this example, a JPEG “movie” is the derived representation (the original video data cannot be compressed) requires a special application to view.

2.4 Performance

2.4.1 Performance Issues Related to ATM

There are virtually no behavioral aspects of an ATM “network” that can be taken for granted, even in an end-to-end ATM network. By “network” we mean the end-to-end data path from the transport API through the host network protocol (TCP/IP) software, the host network adaptors and their device drivers, the many different kinds of ATM switches and physical link bandwidths, and then up through the corresponding software stack on the receiver. Further, the behavior of different elements at similar places in the network architecture can be quite different because they are implemented in different ways. The combination of these aspects can lead to complex and unpredictable network behavior.

In the next two sections a provide some information and observations on this issue. We only address issues in the flow of data after the host-to-host connection is established. (That is, we do not address routing and signaling.)

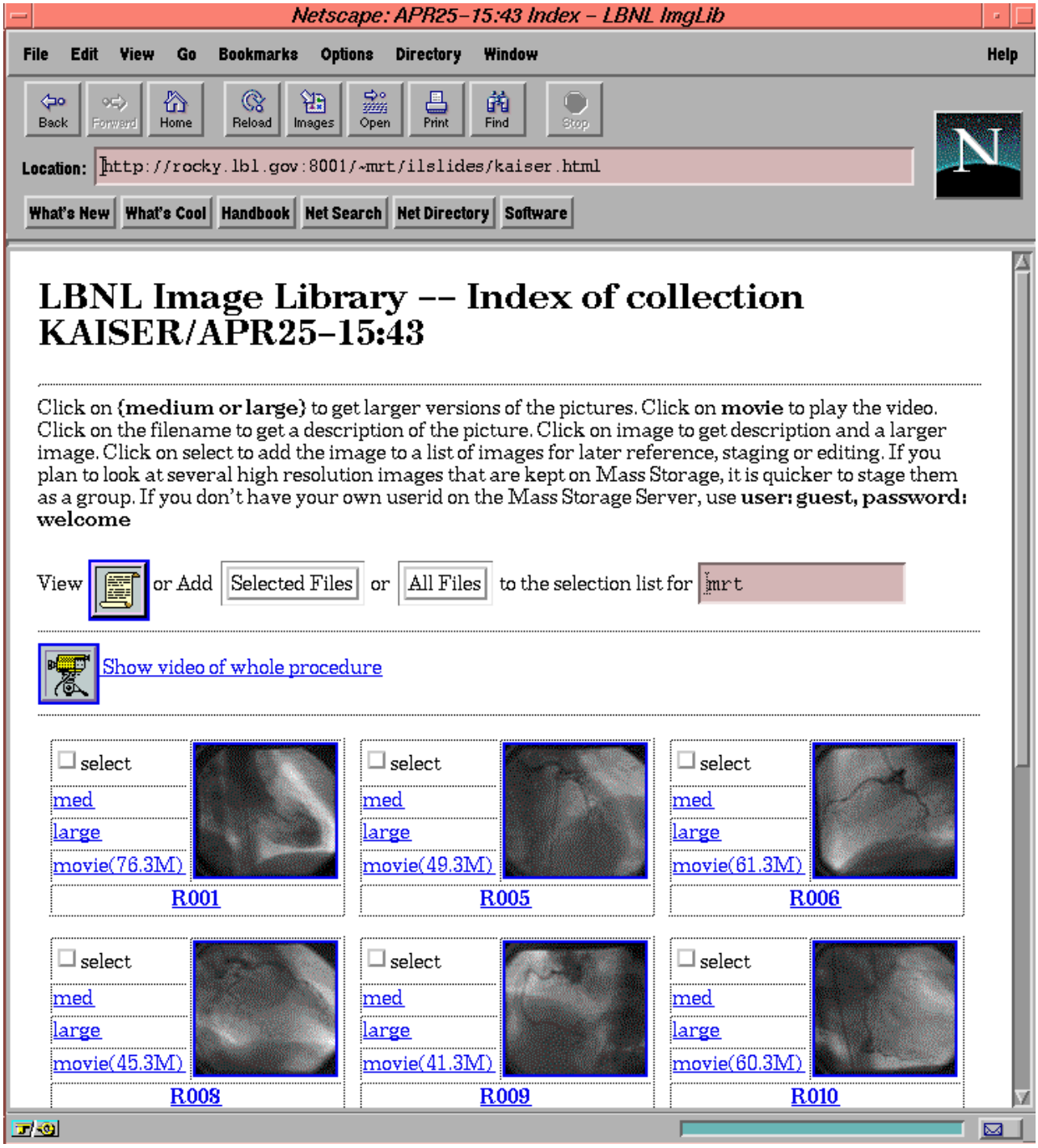

Figure 5 **Large Data-Object Management for Video-like Objects**

2.4.2 ATM Network to Workstation Performance

The architecture and resulting environment that we are describing is a "data movement intensive" environment. That is, in the normal course of dealing with large data-objects in a fully distributed environment, large amounts of data are constantly on the move. This circumstance generates many

the R&D issues from workstation network performance to security, and we discuss a few of them here.

In a scenario where you are counting on parallelism to provide high performance, then one of the important attributes is the independence of the parallel operating elements. One of the reasons that the DPSS has been so successful as a high performance storage system is that using wide area networks increases the independence of the servers (until you get to the final point of delivery).

One of the problems that we have encountered is that independence within the workstation platforms is much more difficult to achieve. Fortunately, platform hardware and software designers have put considerable effort into making key sub-systems operate independently. So, for example, we have found that the parallel operation of disks on controllers, controllers on busses, and memory sub-systems all do a reasonably good job of increasing performance through parallel operation (though they may still be slow in absolute terms).

On the other hand, this is not true for network interfaces. That is, through progress is being made, multiple network interfaces have, so far, generally done a poor job of providing increased throughput by parallel operation. For ATM interfaces, this is due in part to the newness of the technology (and corresponding lack of experience in designing the supporting system hardware and software), and in part due to the fact that prior to the commercial interest in video file servers, striping across network interfaces was not a common practice.

The following synopsis describes the progress in this area (which is very important in high speed distributed applications). Some of the progress is due to the maturing of ATM network code stacks, and some is due to several years of interaction with the workstation vendors pointing out the problems.

The evolution of the IP over ATM infrastructure:

- 1994
  - single interfaces were slow
  - multiple interfaces were no faster (poor independence of system data paths - nothing was multi-threaded)
  - switches dropped cells silently
- 1995
  - single interfaces are faster
  - multiple interface data path independence is improving
  - switches still drop cells silently
- 1996
  - single interfaces and systems are approaching wire speed
  - multiple interfaces better still, but not ideal
  - switches now report cell loss

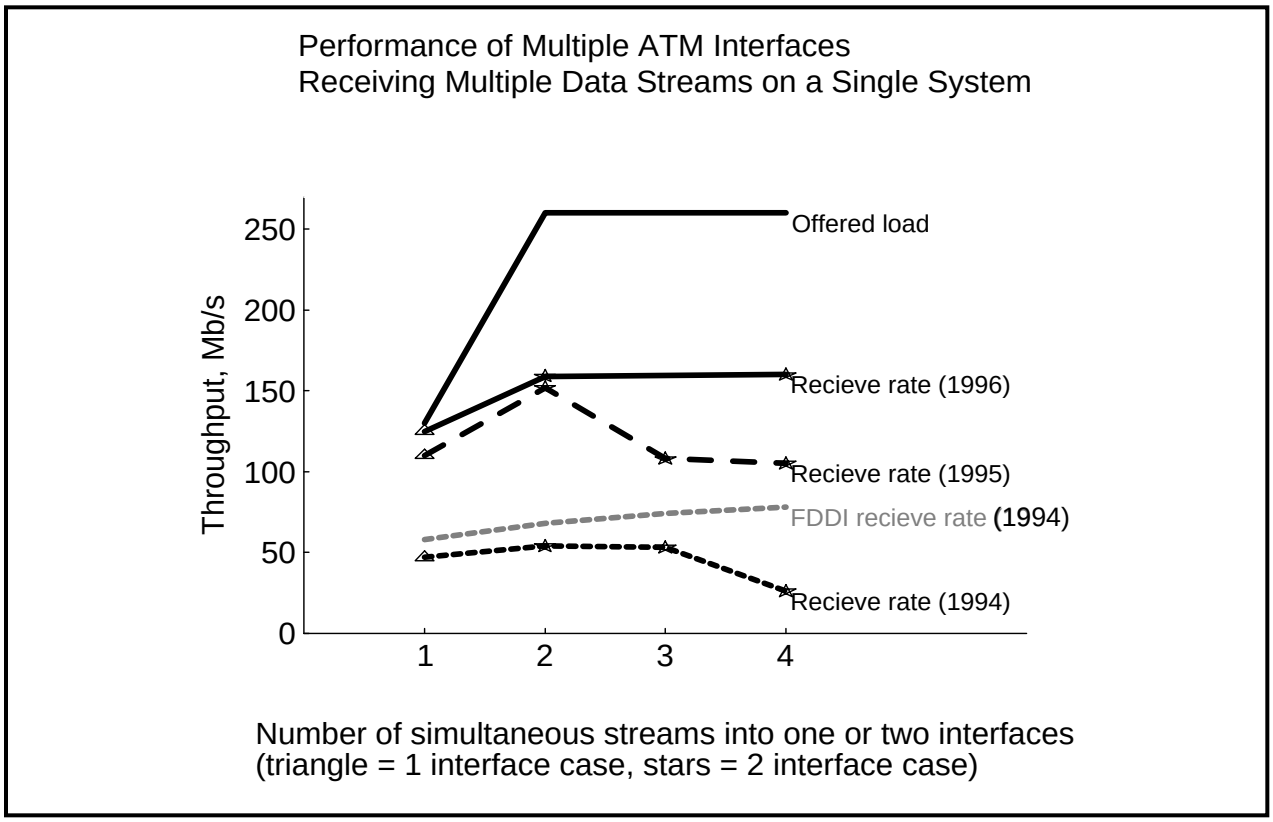

Figure 6 Evolution of the Performance of Workstation ATM Network Interfaces

2.4.3 ATM Network Performance

2.4.3.1 Un-congested Behavior

Application Requirements:

To provide a specific example, we consider the Kaiser CalREN project and environment [Kaiser-CalREN].

Medical imaging applications are a “systems” problem - no one requirement sets all of the necessary parameters). Distributed handling of video-angiography large data-objects is an example of an application constrained by the use of a public data network. Data is collected and stored locally during a short recording period, after which it is transmitted to a network-based storage system for subsequent processing, retrieval, and review at other locations. The quantitative characteristics are:

- data is collected at 30 fr/sec $\times$ (512 $\times$ 512 $\times$ 8 bits $\times$ 2 views) (monochrome video, no compression) = 126 Mbits/sec (one of several relevant parameters)

Four GBytes of data is collected in a local buffer in about four minutes in the form of data-objects that range in size from 50-300 MBytes. Following the data collection, there are about twenty minutes in which to off-load local storage before the next collection period.

Four GBytes in 20 minutes is a rate of about 25Mbits/sec of user-level data on the network, or, accounting for the various protocol overheads, approximately $25 \times 1.4 \approx 34$ Mbits/sec at the ATM level.

34 Mbits/sec is about 23% of an OC-3 network, and it was our initial estimate of the maximum reasonable use of a shared network, in this case Pacific Bell's SF Bay Area ATM network - an ATM OC-3 network which has 30-40 attached sites.

Observations:

We ran a series of throughput tests to determine suitability of the Pacific Bell Metropolitan Area ATM for video-angiography transmission.

Although this network was basically un-congested, several of the experiments were timed to correspond to some high resolution video and audio multicast sessions that generated 5-10 Mbits/sec of "continuous media" (steady state) traffic in the network, in addition to whatever else might have been going on in the network.

The experiments were done by putting the source host ATM adapter into "continuous bit rate" mode, but sending data via IP over AAL-5. This ensured that we transmitted a "true" 25 Mbits/sec, and did not inject high bandwidth "bursts" of data into the network (as would have been the case if we had rate-limited the TCP writes to an "average" of 25 Mbits/sec).

As can be seen in Figure 7, to the level of about 1%, there was no variation in our ability to send 25 Mbits/sec of data for any of the several week long periods during which the throughput experiments were run. Figure 7 shows typical results that include some times with high rate multicast traffic was present (in particular, Friday afternoon).

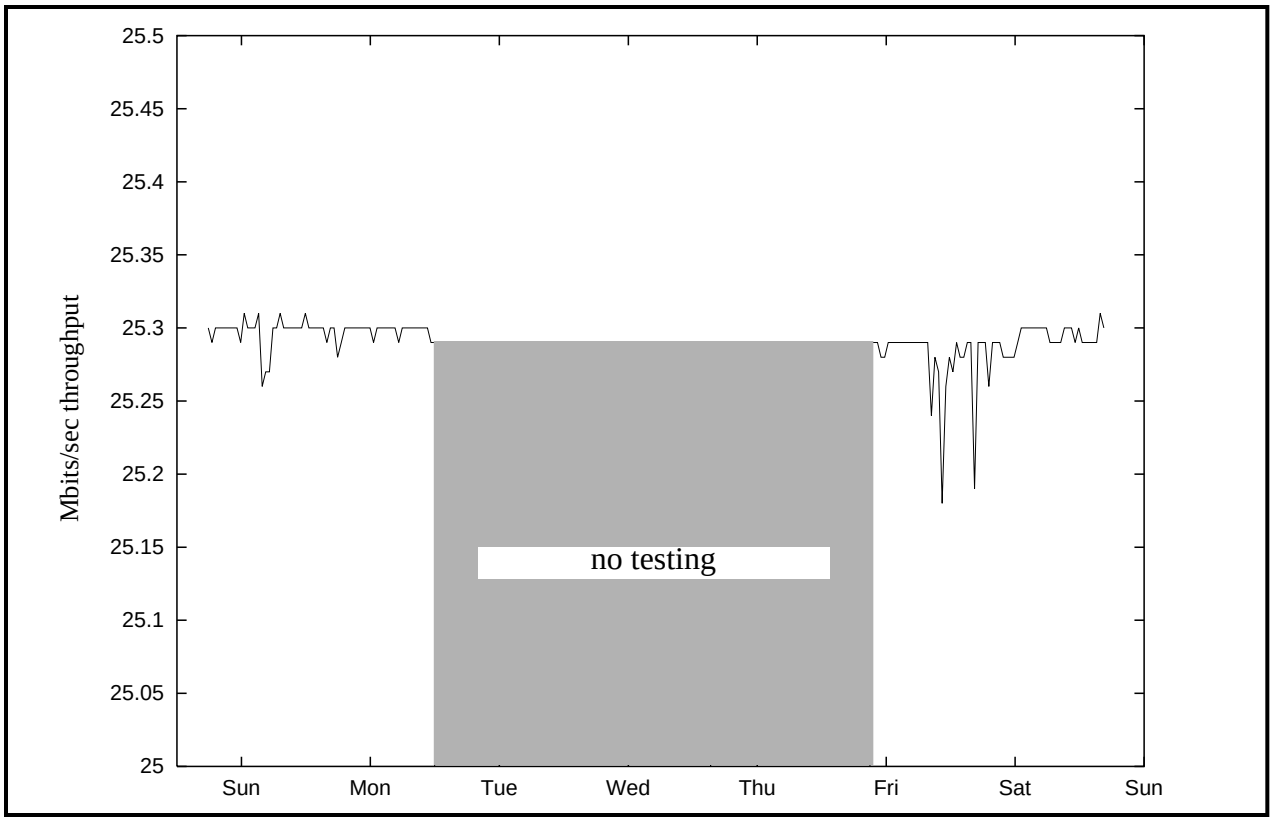

Figure 7 Throughput Testing to Determine the Suitability of the Pacific Bell, SF Metropolitan Area ATM for Video Angiography Transmission

2.4.3.2 Congested Behavior

The impact of congested ATM networks on TCP transport (which is used for all of the data transfers described here) can be very complex. Today's ATM networks and components are still relatively immature and with every new generation of switch software and hardware the congested behavior can change. The most severe impacts occur in circumstances when the characteristics of the various ATM links and network components conspire to defeat TCP's (by now otherwise well tuned) congestion control mechanisms. TCP has a suite of congestion detection and response algorithms [Stevens] that have evolved over the past 10-15 years in the Internet environment, and in that environment they work very well. (The Internet operates under conditions of very high load and high potential congestion, but congestion is generally avoided and the links usually operate at high utilization efficiency.) The problem in ATM networks is that the TCP control algorithms are based on certain assumptions about the relationships among parameters like the packet switch buffer sizes, link minimum transmission unit sizes, link bandwidth, etc. It is very easy for ATM networks to violate these assumptions, and when this happens both the throughput and the link utilization efficiency can be very low in the presence of congestion. (In current implementations of ATM networks, this congestion is frequently caused by small output port buffers in ATM switches. The good news is that most new switches are being designed with large output port buffers.) For a detailed analysis and case study of this situation, the reader is referred to "Performance Analysis in High-Speed Wide Area ATM Networks: Top-to-Bottom End-to-End Monitoring" ([Tierney]).

2.5 Security Architectures for Large-Scale Remote Environments

The access to remote resources, whether for monitoring, control of experiments, or access to data, requires that security and access control mechanisms be in place in order to provide safeguards against unauthorized access, and privacy of proprietary or otherwise restricted data.

To provide security in large-scale, open access environments, we have evolved a decentralized security architecture that uses the concept of use-conditions imposed by the entity primarily responsible for a resource. That is, while the access restrictions are enforced at the resource, the conditions of access are specified by the principal responsible for specifying an access policy for the resource. The responsible principal is the only one who imposes and records use conditions, and those conditions are made available in a way that has nothing to do with the operation of the resource itself: The security mechanism at the resource is only responsible for checking that the use-conditions are satisfied, and enforcing access controls appropriately.

We believe that the combination of this model, and the generalized public-key certificate infrastructure that supports it, can provide a highly scalable and transparent infrastructure for both authentication and authorization of access to the full range of network based resources. The goal is a security architecture that can encode, distribute, and protect the information needed to enable routine secure availability of remote resources in a widely distributed environment.

As with many scalable distributed systems, the decentralized aspect of the architecture is the key to scalability: the administrative task of maintaining an appropriate expression of access policy is only done by those entities (principals, or their agents) that are directly responsible for the policy governing the use of a resource. The operational administration of that resource has only to deal with providing an access control mechanisms, not with access control information. Central administration or administration, by the resource operator of access control lists, or multiple roles that have to kept straight by resource users, are not a scalable approach in a global distributed environment, and are implementation of security that this model is intended to replace.

Paired with the resource owner's use conditions are the entities that can attest to the relevant attributes of a user or agent that is seeking access to a resource: A school registrar may certify a level of training; a state examining board may certify a level of professional attainment; an organization can certify employment (and related group membership); a bank, a level of credit; and a certification authority may attest to individual identity.

In addition to conditions and certifications of those conditions, a collection of trust relationships must exist. So, for example, that a resource controller will accept a certificate that purports to represent satisfaction of a use-condition requires that the resource owner trusts the entity that is attesting to (certifying) an attribute of the user is qualified and/or certified to make such a claim. These trust relationships may be direct: the resource owner and the verifier of conditions have an explicit, pair-wise agreement; or they may be indirect: a resource owner accepts a statement of policy, perhaps policed by a third party, as an implicit guarantee of the satisfaction of a use-condition.

In the general case in our architecture, a broker / agent acting on behalf of the potential user, collects the certified use-conditions for a resource, and then collects the certificates / credentials that verify the user's satisfaction of the use conditions. The combination of resource identity, use-conditions, and matching credentials, all of which are cryptographically secured, are used to form a "capability": A capability to access, monitor, manipulate, use, etc., a remote resource.

The combination of direct reference to those responsible for the generation and assurance of information (the resource owners or policy makers), and standardization of representations of the information that permit automated verification, should form the basis of an easily used and general purpose authentication and authorization system that provides such services as qualifying remote experimenters to use instrumentation systems in distributed collaboratories, as well as forming the basis for dynamically configured, large-scale, distributed computing resources.

Automated acquisition and verification of attribute credentials will increase efficiency and reliability, and will play a critical role in scaling remote environments by removing the need for human intervention in repetitive and trivial tasks such as credential checking, especially when the human participants are never in the same place at the same time, and may never physically visit the site were service is being requested.

However, even with all of these tools and infrastructure in place, they are still just that: tools and infrastructure. In each given situation where a resource is being protected, those responsible must evolve a security model that provides the level of assurances, protection, auditing, etc., that meets their requirements. Flexible tools and infrastructure make it possible to implement the specific security model in a usable and cost effective way.

2.5.1 Implementation Issues

Implementation of the model described above requires a number of operational components with capabilities beyond what we see today:

- Distributed brokers that understand the encoding of the information in corporate database objects represented in signed certificates will be needed to automatically build capabilities.

  In the general model (see [Johnston96a]),when capabilities require a number of attribute certificates from different sources, a "broker" will locate the certificates required by the specified use-conditions and build the composite certificate (capability) on behalf of the user.

- Automated coupling between corporate databases and certificate servers will minimize the administrative overhead.

This will (eventually) provide the automated maintenance of attributes and automatic certificate generation based on information obtained from the primary corporate database (e.g. employee records, school records, etc.).

- Rules and mechanisms for encoding requirements (the “use-conditions”) for services will be required, as will mechanisms for determining the level of trust to place in other CAs (i.e. how are CA operating policies published and evaluated), etc. (For example, see [Mendez and Huitema].)

Our prototype implementation addresses:

- credential and key management via SSH and LDAP/X.500
- security context establishment via GSS/SPKM
- secure, authenticated, and integrity protected exchange via GSS-API

The SSH design [SSH] provides a convenient and transparent management of the credentials and public keys of servers and the private keys of the user. (Roles are possible through different “identities” of the user.) SSH also provides protection of unmodified applications by establishing a secure shell, and then redirecting communication with the outside world through secured ports.

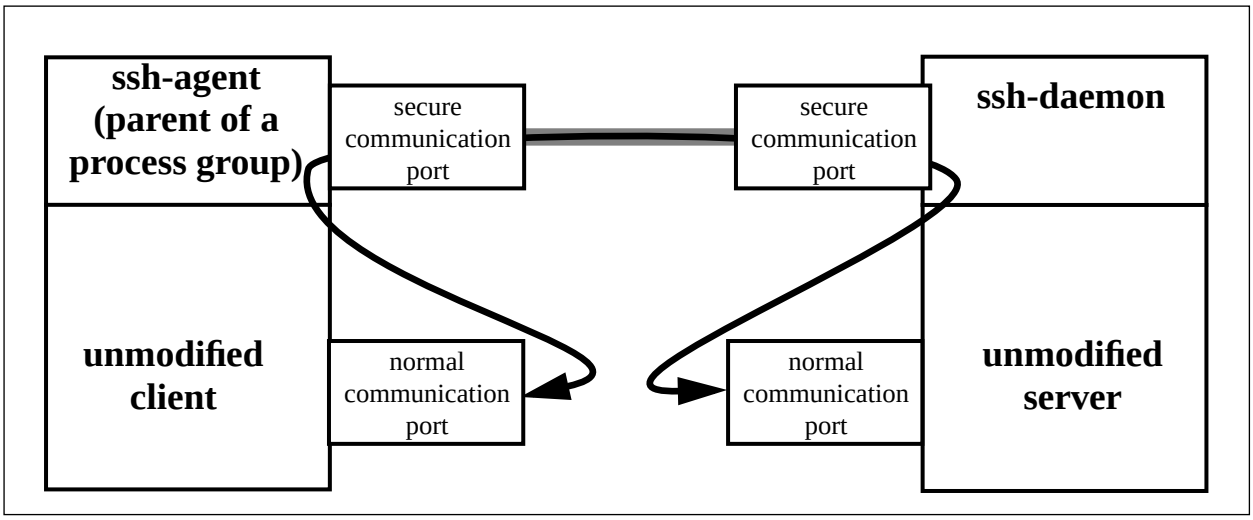

Figure 8 Protection of a Unmodified Client using a “Secure” Shell Approach

We have modified the SSH credential and key management facility to be able to set up the SPKM [GSS] security context for GSS. (See [Adams] for a good discussion of the rationale for SPKM and its relationship to GSS/Kerberos.)

SSH makes the user’s private key available to SPKM to do the session key exchange needed for GSS security context establishment. Once this is done, the GSS-API provides a simple application interface to send validated and/or confidential communication.

We currently use an ad-hoc use-condition encoding, but are examining other approaches. The automatic brokering depends on being able to collect, parse, and automatically match use-conditions and credentials. This is future work.

2.5.2 Relationship to Other Work

Our general approach is an application of public key certificates, and as such it draws ideas from an active community of researchers.

The basic concepts of distributed systems security are laid out in various ISO (International Organization for Standardization) and ECMA (a Europe based international association for the standardization of information and communication systems) documents (see [SIRENE]). SESAME is an implementation that incorporates many of the [ECMA-219] and [ECMA-138] concepts. (See [SESAME].)

We have, however, taken a somewhat different direction than the ECMA and SESAME approaches with respect to attribute certificates, which are central to our approach. In the ECMA approach, most of the emphasis on the use of attribute certificates seems to be in the form of access control lists and roles (party A, who controls a resource, grants to party B the right to use that resource, perhaps under restricted conditions (roles)). Our approach treats use-conditions, rather than identity, as the fundamental concept. That is, we see resources primarily being protected by requirements on the user (e.g. level of training) rather than identity. (ACLs are, of course, easily implemented in this model.)

2.5.3 Application of the Security Architecture

We are using the prototype implementation to provide security to the health care information application mentioned above (see [Johnston95a]) that uses the Distributed-Parallel Storage System ([DPSS]). The application uses a WWW-based system for managing certain kinds of medical imaging records. The data-object viewing application ("Vplayer") is invoked via a WWW image database frontend that uses WWW security to protect the metadata. The application (a specialized, combined video and image browser) then accesses data stored on the DPSS. The Vplayer access to DPSS data is protected by the security architecture described here. (That is, credentialing is handled by the secure shell that runs Vplayer, and the DPSS access is controlled by checking authorization certificates presented by Vplayer.)

3.0 Conclusions

We have described several elements of an architecture the using high speed, wide area networks as the basis of a highly distributed environment for handling all aspects of large data-objects. Variations of the overall approach have been applied in several different projects, and we have described some of the results.

We believe this approach to be both practical and powerful given the increasing availability of high speed networks. Further work will be done on both the various functional elements and specific applications. Please see http://www-itg.lbl.gov for more information.

4.0 Acknowledgments

In the Kaiser CalREN project, E-J. Pol of Philips Research, Palo Alto, CA, designed and implemented the workstation interface for the cardio-angiography system. Dr. J. Terdiman, of the Kaiser Permanente Division of Research provided the leadership for the health care aspects of the project. Dr. Bob Lundstrum, Kaiser San Francisco Hospital, Cardiac Catheterization Laboratory, also provided guidance and information specific to the health care application. Pacific Bell, through its CalREN program, provided funding for connecting the LBNL, Kaiser, and Philips sites to the SF Bay Area ATM network. Sun Microsystems assisted in various ways, including loaning equipment to various Kaiser sites.

5.0 Notes and References

Adams Adams, C. “IDUP and SPKM: Developing Public-Key-Based APIs and Mechanisms for Communication Security Services”. See http://bilbo.isu.edu/sndss/sndss96.html

DPSS The Distributed-Parallel Storage System (DPSS) Home Page (http://www-itg.lbl.gov/DPSS)

ECMA-138 “Security in Open Systems - Data Elements and Service Definitions”, 1st Edition (December, 1989). European Computer Manufacturers Association, http://www.ecma.ch/ecma-138.HTM

ECMA-219 “Authentication and Privilege Attribute Security Application with related key distribution functions”, 1st Edition (December 1994). European Computer Manufacturers Association, http://www.ecma.ch/ecma-219.HTM

GSS Linn, J., “Generic Security Service Application Program Interface” (http://ds.internic.net/rfc/rfc1508.txt) Also see more recent and related drafts at the IETF

ImgLib See http://www-itg.lbl.gov/ImgLib/ImgLib_intro.html

Johnston96a Johnston, W. and C. Larsen, “Authenticated Global Capabilities for Distributed Systems” (draft)

Johnston95a Johnston, W., J. Terdiman, E. J. Pol, “The Use of Public ATM Networks for Health Care Imaging Information” (see http://www-itg.lbl.gov/Kaiser/LKP)

Johnston95b W. Johnston, and D. Agarwal, “The Virtual Laboratory: Using Networks to Enable Widely Distributed Collaboratory Science” A NSF Workshop Virtual Laboratory whitepaper. (See http://www-itg.lbl.gov/~johnston/Virtual.Labs.html)

Kaiser-CalREN Kaiser - LBNL - Philips CalREN project. See http://www-itg.lbl.gov/Kaiser/LKP/homepage.html .

MAGIC MAGIC gigabit testbed. See http://www.magic.net

Mendez and Huitema “A New Approach to the X.509 Framework: Allowing a Global Authentication Infrastructure without a Global Trust Model”, S. Mendez and C. Huitema, in “Proceedings: Symposium on Network and Distributed System Security”, IEEE, San Diego, CA, February 16-17, 1995.

NTON The National Transparent Optical Network (http://www.cais.net/nton/) connects a number of organizations in the San Francisco Bay Area)

SESAME “Advanced Distributed Access Control for the Age of the Open System” http://www.esat.kuleuven.ac.be/cosic/sesame3.html

SIRENE “Security in Computer Networks” (http://www.zurich.ibm.com/pub/sti/www/g-kk/sirene/ also see http://www.zurich.ibm.com/pub/sti/www/g-kk/sirene/pointers.html)

Spartan Sprint test network. T-3 / OC-3 ATM from Kansas City, Kansas to the Sprint Test Laboratory in Burlingame, CA.

SSH Ylonen, T., “The SSH (Secure Shell) Remote Login Protocol” (ftp://ietf.cnri.reston.va.us/internet-drafts/draft-ylonen-ssh-protocol-00.txt). Also see “Ssh (Secure Shell) Remote Login Program” (http://www.cs.hut.fi/ssh/).

Stevens Stevens, W. “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”. IETF Draft. See ftp://ietf.cnri.reston.va.us/internet-drafts/draft-stevens-tcpca-spec-01.txt .

TerraVision A terrain visualization application that uses the DPSS to supply high speed data streams for realtime, “virtual reality” type navigation of real landscape. (See http://www.ai.sri.com/~magic/terravision.html)

Tierney Tierney, B., W. Johnston, G. Hoo, J. Lee, “Performance Analysis in High-Speed Wide Area ATM Networks: Top-to-Bottom End-to-End Monitoring”, To be published in IEEE Networking Journal, LBL Report 38246, 1996. (Also see http://www-itg.lbl.gov/DPSS/papers.html .)