

Toward a Scalable Architecture for Logistical Management of Active Content

Micah Beck and Terry Moore
 Logistical Computing and Internetworking Laboratory
 Computer Science Department
 University of Tennessee
 {mbeck, tmoore}@cs.utk.edu

Abstract— In this paper we analyze the problem of creating Digital Library services for “active content” (e.g. adaptive video, interactive visualization) that can scale. We describe how this challenge can be addressed by technologies based on Logistical Networking, an innovative approach to communication infrastructure that combines bandwidth and storage in a way that supports the flexible co-scheduling of data transport and data storage. Two technologies are central: The Portable Channel Representation (PCR), which embodies an abstract model of the server in order to support the automated mirroring of DL services on a heterogeneous infrastructure; and the Internet Backplane Protocol (IBP), which provides a primitive network storage service that achieves both scalability and strong management of data locality while conforming to the Internet’s “end-to-end” design principles. We argue that this approach can lead to a DL service architecture for active content that will scale up just as the guiding vision of the DL community requires — to millions of users, distributed around the globe in diverse organizational environments.

Keywords -- digital library, active content, replicated services, distributed storage, logistical networking

I. FACING UP TO SCALABILITY

The vision guiding the development of Digital Libraries (DLs) has always included the important idea of universal access to information stores via the services provided over a ubiquitous network. Indeed, the explosive growth of the Internet and the Web over the past decade has seemed to set the stage for the realization of just this vision. But while the DL community has made substantial progress in building up collections of digital content, in developing the user services necessary to access and utilize it, and in establishing mechanisms for cross-DL interoperability among

This work is supported by the National Science Foundation Next Generation Software Program under grant # EIA-9975015, the Department of Energy Scientific Discovery through Advanced Computing Program under grant # DE-FC02-01ER25465, and by the National Science Foundation Internet Technologies Program under grant # ANI-9980203.

Micah Beck is Research Associate Professor and Director of the Logistical Computing and Internetworking Laboratory, Computer Science Department, University of Tennessee, Knoxville, TN 37996-3450. telephone: 865-974-3548, e-mail: mbeck@cs.utk.edu).

Terry Moore is Associate Director Logistical Computing and Internetworking Laboratory, Computer Science Department, University of Tennessee, Knoxville, TN 37996-3450. telephone: 865-974-5886, e-mail: tmoore@cs.utk.edu).

those services, attempts to address the basic problem of *scalability*, which has long been the *sine qua non* of a well-designed network service, have met with far less success. Generally speaking all the problems that continue to limit the scalability of standard client/server applications, and the therefore of Web applications as well, also infect the services that DLs have tried to offer.

In this paper we present an analysis of the problem of providing DL services for “active content” and describe some key infrastructure and middleware technologies that we have developed to attack this problem. These technologies are part of a general approach to communication infrastructure called *Logistical Networking* [1]. Logistical Networking aims to synthesize transmission bandwidth and storage in a way that supports the flexible co-scheduling of data transport and data storage, thereby increasing the performance, scalability and functionality of distributed applications of all types. We argue that this approach can lead to a DL service architecture that will scale up just as the guiding vision of the DL community requires — to millions of users, distributed around the globe, and located in a multiplicity of complex and varying organizational environments.

II. CONTENT-BASED SERVICES

The effort to understand the problem of scaling DL services should begin with an understanding of what a DL service is. The distinguishing features of DL services can be seen in relief against the still broader background of library information services generally.

A library is an instance of a class of institutions fitting this broad description: a location for obtaining information services, whether that is a physical location, a distributed and mobile infrastructure (such as a fleet of bookmobiles) or a virtual location accessible through the Web or other digital communication fabric. Information services vary in their regularity and degree of structure: in traditional libraries, if we think of access to each publication as an individual service, access to the stacks represents a large, highly regular and structured set of services. On the other hand, services such as access to occasional performances and traveling exhibits may be irregular and unstructured, but they are still part of the function of the library. Metaservices, such as learning the

location of a particular publication, range from the highly structured (e.g. the card catalog) to the highly irregular (e.g. asking the reference librarian).

The most structured category of library services are those that fall into the category of access to content. For that reason, content-based services are most scalable and most amenable to commonality between distinct physical locations and therefore implementation as distributed services (e.g. interlibrary loan).

The notion of “content” is fairly well defined in the context of traditional libraries. An item of content is a physical object held in the library and given to users on loan, usually for a limited time and sometimes with limitations on where it can be taken. This straightforward notion has a direct analog when the content in question is a file or digital object, if we replace “*loaning* the content to a user” with “*delivering a copy* of the content to a user.”

While the difference may seem small, it is crucial: a digital library provides information services in which all that changes is the state of the user and of the library itself. The physical objects that make up these elements in the transactions do not change. This is in contrast to a conventional library, which actually loses a physical constituent when a user borrows a book.

The reason for making this distinction, which may seem pedantic to some readers, is that when we consider active content, there may no longer be any unit of content in standard format that passes from the library to the user. For instance, when a video is displayed as a stream that cannot be captured, the service of playing the video is not in fact an object in any conventional sense. If a map or satellite photograph is viewed over the Web through an application-specific viewer, or a data analysis tool is used to cull information from an on-line collection, we are even further from the notion that “content” is an object being delivered to the user. Finally, if the library is a collection of programs that the user can run and interact with, the delivery of objects is gone completely from the interaction between library and user.

In our view the correct way to generalize digital library services to encompass the delivery of “active objects” is to treat the digital library services themselves as active, where the actions they perform are parameterized by digital objects in standard formats [2]. Thus, the content is what fills the library, and the service is what the library does with its content at the request of the user. Each type of content has one or more corresponding services that can be invoked using it as a parameter.

- Any object x of “static” content type (HTTP, GIF, JPEG, PDF, etc) parameterizes the class of active services *download-HTTP(x)*.
- A video object x of type MPEG parameterizes two different services: *download-HTTP(x)* and *stream-RTP(x)*, but library policy might prohibit download of certain files due to copyright restrictions.
- A Playstation 2 game object x parameterizes the service *play-PS2(x)*.

Note that in the case of services other than download, the object remains encapsulated within the library, but its name is used to invoke the parameterized service. Thus, it is important that the active content consists of clearly defined objects with unique names and that the service being invoked is either specified by the user or is inferred implicitly from the name.

The importance of this analysis of digital libraries of active content is that it makes clear that the basis of content management must be the *source object* and the *service* that is applied to it. Any scheme for management of active content must be able

- to access objects from the point of service, and
- to arrange for the proper services to be applied.

III. LOGISTICAL CONTENT MANAGEMENT

The Internet provides an elegant mechanism to provide many services to globally distributed users from a single point of service: users connect to a centralized server using service protocols such as FTP, HTTP, RTP, etc. However, there are many well-known cases where providing global service from a central server is not feasible. The main problem is that the network conditions may not always allow certain services to be delivered between arbitrary points in the network. Another is that some services place a high demand on the server, and so scaling up to address the needs of a large community from a single server becomes difficult. Copyright restrictions sometimes require that services be delivered only to specific communities, such as a campus, and this is sometimes best accomplished by locating the server within that community.

If we accept that it is sometimes desirable to deliver a particular service from multiple points within the network, the next question is how to create such “replica services.” One straightforward way is to capture the output of the service as provided by the central server and then replay it in response to subsequent equivalent requests, i.e. to use a cache. This approach is used by commercial CDNs to distribute the work of providing Web service. The problem with this approach is that it works only with services that can be captured and replayed, and for which it is possible to determine when two requests are “equivalent.” The great advantage of the cache-based approach is that it requires little or nothing of the origin server, but the limitations of this approach are well documented [3]. (The use of any replication approach creates a new problem, however, that is still not solved in a general and scalable way: how the client can transparently choose from among replica servers.)

A more general approach is to create a copy of the server itself and access the source object from the server to provide a replica service. Many services parameterized by a source object are implemented by software servers that can be installed and configured in a standard way to provide equivalent service from the same source object.

Thus, the key functionality required in order to invoke a replicated content-based service is to identify both the

parameterized service and the specific object it is to be applied to, and each of these must be specified in a location-independent manner. Unfortunately, service requests such as HTTP GET do not specify these two items in an unambiguous way, and so it is necessary to provide a mapping from the service specified (i.e. the URL) to them.

A. The PCR Data Model

The Portable Channel Representation (PCR) was originally defined in order to facilitate the creation of mirrors on the heterogeneous servers of the Internet2 Distributed Storage Infrastructure [2]. It addresses each of the problematic areas of mirror creation:

- Server-Independent Specification of Behavior — A PCR description is an encoding of metadata (using the eXtensible Markup Language (XML) and the Resource Definition Framework (RDF)) that specifies the behavior of the server in response to a set of requests, collectively known as a “channel.” In order to avoid dependence on configuration files specific to particular server software, PCR specifies in a platform-independent manner the source object and the method of interpretation that should be invoked on any particular request.
- File-system-Independent Specification of Objects — References to objects from within a PCR channel description do not name them through their installed location in a file system directory structure, but instead use local names defined by a “file store”.

The PCR description and the file store together define a complete channel description that can be correctly implemented on any platform that correctly interprets both elements.

PCR’s descriptive approach associates with each user request both a source file and a type. The type specifies the language and API of that file and is used to determine the interpreter that will be used to generate a response. Since these types are not reflected in the contents of the source file, but are specified by the PCR metadata external to the file, a single source file may be treated as having different types when accessed through different requests (e.g. using different protocols and servers). Possible types include standard HTML, GIF, 3Mb/s MPEG-1 video, Perl with a standard minimal API (no file or network access), or Java bytecode with an SQL database access API.

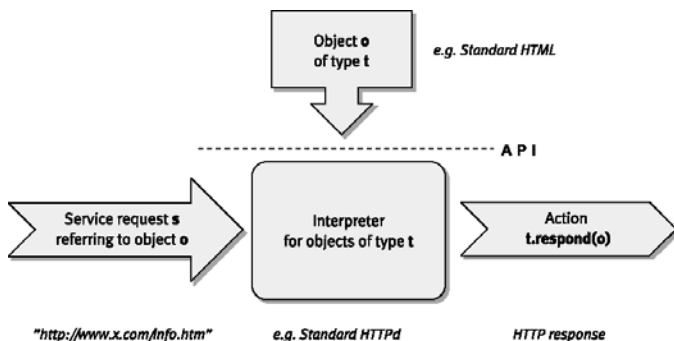


Figure 1: The PCR Behavior Model

The PCR data model is a language intended for the specification of the behavior of a server. This language does not support arbitrary behaviors (i.e. it is not a general model such as Java), but instead works within a highly structured server behavior model, shown in Figure 1. The central notion in the PCR server behavior model is a request fulfillment rule, which can be thought of as a pair consisting of a pattern and an action. When a request is presented to a server and matches a pattern, the server responds by performing the associated action. In concrete terms,

- a pattern consists of several fields which correspond to a standard URL: domain, port, and object name, and
- an action is specified by a source file and a type.

Every type is associated with a method or interpreter, and the action specified by the rule is to invoke that method on the specified data file. For example, a request associated with a stored file with type “Standard HTML” would be interpreted by a standard HTTP server allowing no non-standard extensions.

Every source object type defines syntactic rules for data objects of this type. We refer to these syntactic rules as the language in which the object is expressed. If there is a single language with multiple variants, we refer to these variants as APIs. Thus, HTML is a source object language, but a specific set of allowable server includes defines an API. Perl and Java byte code are also source object languages, and each of them may have multiple APIs. The combination of language and API together define the object type. The server must perform a combination of install- and run-time checks to ensure that a particular object conforms to its declared type.

The sum of all APIs constituting the channel is referred to as the Channel API. Extending the functionality of PCR requires introducing a new type, defining the interpreter and extending the Channel API to include the new object type. New service types can easily be added as they emerge, thus the PCR approach is highly extensible. Given that a service request can be translated to the names of a service and of an object to apply it to, and an appropriate location can be found from which to provide the service, the remaining problem is for the server to access the object by name. The delivery of active content services has thus been reduced to the problem of delivering static objects and metadata about them to replica servers.

Caching of course literally uses the origin server to deliver its objects to the replica, which are simply responses that will be replayed. However, because delivery of objects to a replica is “server to server” it need not adopt the protocols or the strategies used by clients in making requests for individual services. Alternative approaches that can be taken to distribution of source objects include:

- Ad hoc manual mirroring of entire collections.
- Using `rsync` or a similar utility can be used to automatically mirror specific subdirectories (using multicast `rsync+` for scalability).

- Proprietary content/application distribution protocols and algorithms can be used.
- A file replica manager such as those designed for Grid computation could be used [4].
- A distributed file system can be used with internal algorithms for caching and replication.
- A high performance wide area storage area network can be used to completely virtualize the location of data at the block level.

There are however problems with all of these approaches stemming from the fact that scalable wide area services do not implement powerful algorithms for management of data locality and yet solutions designed for local networks do not scale globally and across administrative domains.

B. The Internet Backplane Protocol

Current standard networking protocols such as TCP/IP support end-to-end resource control. An application can specify the end points associated with a communication stream, and possibly the buffering strategy to use at each end point, but has little control over how the communicated data is managed while it traverses the network. In particular, the application cannot influence where and when data can be stored (other than at the end points) so that it can be accessed efficiently.

A technology we have created called the *Internet Backplane Protocol (IBP)* manages storage within the network fabric, optimizing data movement at the domain and application levels [5]. The protocol allows applications to implement interprocess communication in terms of intermediate data-staging operations to exploit locality and manage buffer resources more effectively.

IBP can be characterized as a mechanism to manage either communication buffers or remote files; both characterizations are valid and useful in different situations. To keep our terminology as neutral as possible, we refer to the units of data that IBP manages as *byte arrays*.

- IBP as buffer management — IBP byte arrays can be viewed as application-managed communication buffers in the network. IBP allows the use of time-limited allocation and FIFO disciplines to constrain the use of storage. These buffers can improve communication by way of application-driven staging of data, or they may support better network utilization by allowing applications to perform their own coarse-grained routing of data.
- IBP as file management — IBP byte arrays can be viewed as files that live in the network. IBP allows an application to read and write data stored at remote sites, and direct the movement of data among storage sites and to receivers. In this way IBP creates a *shareable network resource for storage in the same way that standard networks provide shareable bandwidth for file transfer*.

A key point to note is that the Logistical Networking approach endeavors to achieve both scalability and strong

management of data locality by building up functionality according to the same “end-to-end” principles that have guided the design of the IP stack [6, 7]. Indeed, the features of IBP are modeled directly on the features of IP [8], as we discuss in more detail below.

IV. APPLICATION SCENARIOS

A. Adaptive Video-on-Demand

Advanced video playback services can require substantial infrastructure on both the client and the server end to provide features such as thumbnailing, feature finding, pause/zoom and hyperlinking [9]. Such services may use standard content formats (e.g. MPEG-4 or 7), can require both custom server and client software, and may use non-standard protocols for signaling.

A digital library that provides such extended services would need to not only distribute the video file, but to specify the model of video play back that was to be used to interpret it. It would be incorrect to reply to the URL that expected advanced video service with a simple video stream. Thus, the channel that implements the advanced portion of the library can only be installed on a server that supports advanced video delivery.

However, the same file delivered using a standard streaming protocol could be associated with another URL, and the channel representing the conventional portion of the library could be installed much more widely, perhaps serving a wider community of conventional clients. Thus, a single file can enable more than one service, resulting in channels with identical data content but different metadata, resulting in different management and providing different services.

B. Interactive Visualization / Virtual Reality

A range of applications from global Geographical Information Systems [10] to haptic investigation on a molecular level [11] allow users to explore stored data sets using highly interactive protocols that in some cases create an entire virtual world. Conventional files or databases underlie libraries of such experiential services, but it is the services themselves that are collected, and in many cases the user has no direct access to the raw data.

In such cases, it is important to install the channel data sets only on servers where the appropriate application servers are available. This means that those specific servers must be given identifiers within the namespace of services and the channel metadata must correctly identify those services by name. Standard CDN metadata intended for commercial content has little hope of capturing the generality, extensibility and diversity of services required by the academic community.

C. A Library of Games

One of the most socially important vehicles for publication of active content is the commercial video game. Running only on game consoles that are connected directly to peripherals such as a television and a joystick, there is in fact no

possibility of network delivery of the service they define. In effect, the server must be collocated with the user.

In order for a digital library to distribute games, the content repository must be directly connected to the game platform and the user must have direct access to it. Thus, there must be one server for each point of service to the user. That fact that the service requires that the user be physically collocated with the server does not change the fact that a library of games is a digital library, and that the distribution of games must be controlled in order to ensure that games and gaming platforms are correctly matched. For instance, attempting to run a Sony PS2 game on server with PS1 hardware will generate only errors. In this case there is no network interaction between server and user for a cache or other proxy to intervene on, but creating a replica server is quite feasible.

V. INFRASTRUCTURE

The research program of the Logistical Computing and Internetworking Laboratory at the University of Tennessee started with the Internet2 Distributed Storage Infrastructure (I2-DSI) project, an experimental project to provide advanced Content Delivery services to the academic community [12]. As a deployed infrastructure, I2-DSI started with servers donated by IBM (six AIX servers, 70GB disk each) and StorageTek (one Linux server, 700GB) and located at a number of partner institutions throughout the United States. Today, we are in the process of deploying a new generation of I2-DSI servers, mostly Linux servers with 1TB of IDE disk running on a PC platform.

The I2-DSI project has developed a set of tools that automate the process of creating mirrors of content-based sites using `rsync`, a standard file mirroring utility [13]. (Multicast features developed by Bert Dempsey of the University of North Carolina, Chapel Hill as part of the I2-DSI project are being incorporated in the standard `rsync` release.) The processes of channel publication and subscription are controlled by these tools, and the end user need not be involved in any of the details of setting up `rsync` sessions, modifying server configuration or DNS resolver configuration files. These tools and accompanying documentation are available on the I2-DSI project Web site [14].

These I2-DSI channel tools do not provide any abstractness in channel representation. Each channel is a subtree of the file system of the “master” node which it is published, and it is simply copied into the file system of each subscribing “slave” node. The only control on the implementation of a channel is a set of service guidelines that describe specific services, such as CGI execution of Perl programs, and detail the language and API restrictions for each. It is then up to the publisher and subscriber to respectively advertise and consult a description of the set of services used by each channel and to ensure that no slave server subscribes to a channel requiring a service that the slave does not implement. Thus, the current implementation of I2-DSI is extremely flexible, but not at all safe from user error.

PCR was developed as an abstract channel description that expresses a channel as a set of stored objects of known type that are invoked with specific methods [2]. The process of publication requires this type metadata to be provided, and the process of subscription enables the slave node to check the metadata for compatibility with the services it provides. Finally, the PCR metadata allows the files to use a localized installer at each slave node, allowing for installation in heterogeneous systems for which simple copying of a file hierarchy would not work correctly.

A suite of Content Distribution Networking software based on PCR has been developed by Lokomo Systems, a software company based in Stockholm, Sweden. Implemented entirely in Java, but interfacing with standard non-Java utilities such as Apache, the Lokomo CDN suite can deliver channels including active objects across heterogeneous operating systems and architectures. A basic release of the Lokomo tools will be made available to academic projects on a cost-recovery basis.

The Internet Backplane Protocol (IBP) is a scalable mechanism developed to facilitate system state management in the wide area using shared storage resources. IBP is intended to provide a common interoperable layer of network storage resources, enabling portability in applications and allowing common infrastructure to be deployed that is neutral as to the nature of applications layered on top of it.

The key to scalability and interoperation in IBP is the primitive nature of the service and the weakness of the semantics of allocation. IBP service is somewhat more abstract than a block storage service such as iSCSI [15] in order to facilitate sharing, but the main difference is that in IBP, every allocation of storage can be time limited, providing the client with a lease on storage resources that may be renewed, but for which there is no guarantee of renewal.

The primitive and weak nature of IBP allocations, modeled after the weak properties of IP datagram delivery, means that it is an inconvenient programming interface for most applications. Just as IP service is not accessed directly by users, but only through layer 4 services such as TCP and UDP that are implemented at the endpoints, most clients will access IBP through similar end-to-end services.

The `exNode` is a standard data structure that is intended to facilitate the aggregation of IBP allocations into storage objects having more of the strong properties associated with files, including unbounded size and duration, reliability and accessibility [16, 17]. As with TCP/IP, these properties are implemented on a probabilistic basis, tools based on the `exNode` achieve these goals by aggregating weaker IBP allocations. The Logistical Runtime System (LoRS) is a suite of `exNode`-based tools developed under the direction of Dr. James S. Plank of the LoCI Laboratory [18]. This suite of tools is freely available for download [19].

Initial deployment of IBP logistical storage resources has been through a cooperative project between U.S. and European research universities. Individual institutions deploy IBP depots and then enter them in the Logistical Backbone, or

L-Bone, which serves as a registry of available depots and a resource discovery infrastructure. Based on an underlying LDAP server, the L-Bone server software is also available for download, and several independent resource pools have been created by researchers who do not want to add their depots to the common L-Bone pool. However, keeping a separate resource discovery mechanism does not affect interoperability at the level of the IBP protocol.

Total storage resource registered with the L- Bone were about 1 TB as of June 1, 2002, with several additional individual 1 TB depots expected to come online by 1 September. LoCI Lab is seeking to increase the shared storage resources available to the research community by at least two orders of magnitude through the creation of an NSF-funded National Logistical Networking Testbed and similar parallel projects to be funded by the Department of Energy and other agencies. Through these efforts, and other cooperative provisioning of resources by institutions and foreign networks, our goal is to achieve an overall provisioning of 1 petabyte of storage managed by IBP three years from now.

For the purposes of Distributed Digital Library projects, shared storage is necessary for distribution but highly reliable and predictable archival storage is necessary to serve the role of a library. In these cases, an IBP interface to a conventional archival storage system is a mechanism for enabling interoperability with the wide area storage network, but need not imply weak semantics of storage or lack of predictability in duration of allocation.

VI. CONCLUSIONS

While taking a layered approach to management of active content may seem like it requires extensive infrastructure development, in fact all of the components are developed and in current use. I2-DSI currently serves active content for academic projects from multiple servers, with automated management of data movement functions. The Lokomo implementation of PCR-based content distribution can provide an additional layer of checking, safety and portability. IBP is current and widening deployment as an underlying network facility to support data movement.

The greatest advantage to the layered approach is that it provides a path to a common infrastructure, and to global scalability. IBP, as an infrastructure for moving and storing raw data, is an adjunct to IP datagram service that extends the services provide by the network to include a primitive storage service. IBP can be used by a great variety of applications, and can be supported by a much larger advanced networking community that includes but does not define the digital library community. In fact, the community that can support IBP is much broader than the "advanced networking community" as usually defined: those that can support high bandwidth and advanced IP services such as multicast , IPv6 and QoS.

However, working up the stack to services such as PCR-based active content management, sharing would be more

limited, mainly to commercial services such as CDNs that increasingly seek to provide active services along with their more traditional cache-based replay services [20].

It has traditionally been the case that the Distributed Computing and Networking communities have been able to implement systems without broad user communities, either designed from first principles of their own devising or based on very specific requirements geared to the needs of particular agency. In the past, the Internet itself was such as system, and today there are many possible vehicles for active services, ranging from Active Networks [21, 22] and Content Distribution Networks [23, 24] to "the Grid." [25].

One traditional role for the Information Science community has been to provide a structured framework for the use of advanced information services that makes them accessible to large and scalable communities of both information service providers and users. This has been accomplished by drawing lines between the seemingly limitless potential of the unstructured system and the greater utility of an information system circumscribed by rules of use. The outstanding example is the Web, which is a very limited solution for use of the Internet. The management of active content is today paralyzed by the very openendedness of its potential and the desperation of commercial players. It is in need of a solution that is adequate to the needs of a large enough community, but adapted and restricted for usability. Engineering such boundaries one of the important roles of the Information Science and Digital Library communities.

REFERENCES

1. M. Beck, T. Moore, J. Plank, and M. Swany, "Logistical Networking: Sharing More Than the Wires," in *Active Middleware Services*, vol. 583, *The Kluwer International Series in Engineering and Computer Science*, S. Hariri, C. Lee, and C. Raghavendra, Eds. Boston: Kluwer Academic Publishers, 2000.
2. M. Beck, T. Moore, L. Abrahamson, C. Achouiantz, and P. Johansson, "Enabling Full Service Surrogates Using the Portable Channel Representation," presented at Tenth International World Wide Web Conference, Hong Kong, May 1-5, 2001.
3. M. Beck, R. Chawla, B. Dempsey, and T. Moore, "Portable Representation for Internet Content Channels in I2-DSI," presented at 4th International World Wide Web Caching Workshop, San Diego, CA, 1999.
4. C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC Storage Resource Broker," presented at CASCON'98, Toronto, Canada, 1998.
5. J. S. Plank, A. Bassi, M. Beck, T. Moore, M. Swany, and R. Wolski, "Managing Data Storage in the Network," *IEEE Internet Computing*, vol. 5, no. 5, pp. 50-58, September/October, 2001.
6. J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp.

- 277-288, November, 1984.
7. D. P. Reed, J. H. Saltzer, and D. D. Clark, "Comment on Active Networking and End-to-End Arguments," *IEEE Network*, vol. 12, no. 3, pp. 69-71, May/June, 1998.
 8. M. Beck, T. Moore, and J. S. Plank, "An End-to-end Approach to Globally Scalable Network Storage," presented at ACM Sigcomm 2002, Pittsburgh, PA, August 19-23, 2002 (to appear).
 9. S.-J. Lee, W.-Y. Ma, and B. Shen, "An Interactive Video Delivery and Caching System Using Video Summarization," presented at The Sixth International Workshop on Web Caching and Content Distribution, Boston, MA, USA, June 20-22, 2001.
 10. M. Reddy, Y. Leclerc, L. Iverson, and N. Bletter, "TerraVision II: Visualizing Massive Terrain Databases in VRML," *IEEE Computer Graphics and Applications*, vol. 19, no. 2, pp. 30-38, March/April, 1999.
 11. K. Jeffay, T. Hudson, and M. Parris, "Beyond Audio and Video: Multimedia Networking Support for Distributed, Immersive Virtual Environments," in *Proceedings of the 27th EUROMICRO Conference*. Warsaw, Poland, 2001, pp. 300-307.
 12. M. Beck and T. Moore, "The Internet2 Distributed Storage Infrastructure Project: An Architecture for Internet Content Channels," *Computer Networking and ISDN Systems*, vol. 30, no. 22-23, pp. 2141-2148, 1998.
 13. B. J. Dempsey and D. Weiss, "Towards an Efficient, Scalable Replication Mechanism for the I2-DSI Project," University of North Carolina at Chapel Hill School of Information and Library Science, TR-1999-01, April 30, 1999.
<http://www.ils.unc.edu/ils/research/reports/TR-1999-01.pdf>.
 14. *Internet2 Distributed Storage Infrastructure (I2-DSI)*, <http://dsi.internet2.edu>, 2002.
 15. J. Satran, D. Smith, K. Meth, O. Biran, J. Hafner, C. Sapuntzakis, M. Bakke, M. Wakeley, L. Ore, P. V. Stamwitz, R. Haagens, M. Chadalapaka, E. Zeidner, and Y. Klein, "iSCSI," IETF, Internet Draft, draft-ietf-ips-iSCSI-12.txt, April 17, 2002.
<http://www.haifa.il.ibm.com/satran/ips/draft-ietf-ips-iSCSI-12.txt>.
 16. A. Bassi, M. Beck, and T. Moore, "Mobile Management of Network Files," presented at Third Annual International Workshop on Active Middleware Services, San Francisco, August 6, 2001.
 17. S. Atchley, S. Soltesz, J. S. Plank, M. Beck, and T. Moore, "Fault-Tolerance in the Network Storage Stack," presented at Annual Workshop on Fault-Tolerant Parallel and Distributed Systems (in conjunction with International Parallel & Distributed Processing Symposium), Ft. Lauderdale, FL, USA, April 15-19, 2002.
 18. *Logistical Computing and Internetworking Laboratory*, <http://loci.cs.utk.edu>, 2002.
 19. *exNode Tools*, <http://loci.cs.utk.edu/exnode/tools.html>, 2002.
 20. A. Barbir, R. Chen, M. Hofmann, H. Orman, R. Penno, and G. Tomlinson, "An Architecture for Open Pluggable Edge Services (OPES)," IETF, Internet Draft, draft-ietf-opes-architecture-00, May 3, 2002.
<http://www.ietf.org/internet-drafts/draft-ietf-opes-architecture-00.txt>.
 21. D. L. Tennenhouse and D. J. Wetherall, "Towards an Active Network Architecture," *Computer Communications Review*, vol. 26, no. 2, pp. 5-18, April 1996, 1996.
 22. D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A Survey of Active Network Research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80-86, January 1997, 1997.
 23. D. C. Verma, *Content Distribution Networks: An Engineering Approach*: Wiley, 2001.
 24. K. L. Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek, "The Measured Performance of Content Distribution Networks," in *Proceedings of the 5th International Web Caching and Content Delivery Workshop*, 2000.
 25. I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufman Publishers, 1999, pp. 677.