Next Generation Content Distribution Using the Logistical Networking Testbed

Scott Atchley, Micah Beck, Hunter Hagewood, Jeremy Millar,
Terry Moore, James S. Plank, and Stephen Soltesz

{atchley, mbeck, hagewood, millar, tmoore,
plank, soltesz}@cs.utk.edu
Technical Report UT-CS-02-498
Logistical Computing and Internetworking Laboratory
Department of Computer Science
University of Tennessee
1122 Volunteer Boulevard, Suite 203
Knoxville, TN 37996 USA

Abstract. We describe the difficulties of content distribution and building an ad hoc content distribution network using the Network Storage Stack and a publicly available testbed. The testbed uses the Network Storage Stack, developed at the University of Tennessee, which allows for flexible sharing and utilization of writable storage as a network resource. The ad hoc content distribution network improves resource utilization and user throughput without highly centralized control. The networking testbed provides over 10 TB of shared storage around the world that is available to all for research.

1 Introduction

Information dissemination for producers and consumers of large datasets over the wide-area network is a logistical challenge. Data that is generated by research simulations and multimedia enthusiasts can be difficult to obtain because it is often stored at network endpoints and served by slow transfer services such as HTTP and FTP. We will discuss current strategies (mirroring and content distribution networks) and their benefits and shortcomings including scalability issues. To overcome these problems, the Logistical Computing and Internetworking Laboratory at the University of Tennessee has developed the Logistical Networking Stack that allows users to store data in the network and access it quickly and easily. We discuss the layers of the stack and how we built a high-performance content distribution system that overcomes many of the deficiencies of modern content delivery techniques using the components of the networking stack.

2 Current Methods

Content replication strategies and techniques have continued to evolve. We examine in this section two commonly used methods for making popular content more easily accessible and some of the benefits and shortcomings of both.

2.1 Traditional Mirroring

Mirrors are non-authoritative replicas of resource-intensive Internet content. They are a simple content delivery solution designed to spread network traffic across machines that do not share a common Internet connection. This strategy is used when the content attracts a large number of visitors, saturating available HTTP or FTP connections, or when the content sought has the potential to consume a significant percentage of available bandwidth. It is very much akin to web-caching solutions [14], but with longer persistence.

The mirroring process begins with the master copy. This is usually an HTTP or FTP service with a recognized brand name (e.g. redhat, mandrake). Other locations with available resources and interest in the content (either direct or indirect) agree to replicate the service by downloading the contents of the master node and then keeping the contents current using scripts and file transfer tools (such as rsync [7]). Once that relationship is established, the master node advertises the additional servers that mirror its content. Mirror nodes with substantial resources may also be used as intermediaries for other nodes, creating a replication tree and relieving the load on the master node.

The mirroring solution does achieve the basic objective of distributing network traffic, but creates significant problems for content owners and does little to improve the user's access to information. Content owners must depend on the administrators of the mirrors to preserve the quality of their work. Much more coordination is required because administration is not centralized, therefore reducing system-wide control and responsiveness. Other possible administrative problems include rogue synchronization jobs and clandestine mirrors.

Little information about the official mirror sites is offered to the user, resulting in poor load-balancing and trial-and-error access. The domain names of the mirrors usually indicate what country they are in and other geographical pointers are sometimes included in the list. Occasionally the maximum number of connections allowed at each mirror will be provided, but no information is typically available about how many are already in use. This limited information can influence the user in unpredictable ways such as always picking the first mirror even if the list is ordered alphabetically. Asking the user to choose based on such limited information does not maximize the efficiency of the available resources and results in an uneven distribution of traffic and lower performance for the user. Information about each mirror's network connection status and number of connections used may provide a basis for better decision-making, but the bottom line is that the user is the resolution service.

2.2 Advanced Content Distribution Networks

The Internet2 Distributed Storage Infrastructure (I2-DSI) project is a replicated hosting platform for Internet content and services [6]. Its purpose is to investigate issues surrounding content distribution networks (CDN) by building a geographically distributed test bed that supports various types of Internet services. I2-DSI, like most CDNs, relies heavily on file replication to ensure the benefits advertised by CDN services, namely improved latency, reduced bandwidth consumption, and automated backup.

The I2-DSI project looks for high-demand content and services of interest to the academic and research communities. This content is then "channelized" for portability and replicated across the I2-DSI [4]. I2-DSI uses Cisco's Distributed Director to map a single hostname to multiple machines, which directs the request for content to the appropriate mirror. The appropriate mirror is determined by weighted network metrics such as border gateway protocol (BGP), which determines network proximity by hop count, and other metrics [8]. This method is superior to traditional mirroring since the resolution process is transparent to the user and ensures a better utilization of the replicated content and associated resources. This approach does not address the problem of servers becoming overloaded if more of the clients are closer to one of the mirrors.

Content being served by I2-DSI can manifest detectable patterns of utilization. The patterns are often temporal or geographical in nature. Once understood, this information allows the administrators to forecast CDN resource utilization and develop content replication strategies to further improve content delivery. Once setup and properly configured, a CDN is an effective and sophisticated means of information access and delivery.

There are two barriers to making CDNs a more common service: cost and replication integrity. Deploying a CDN for publicly available content is expensive. It requires administrative control over nodes with large storage capacity at geographically dispersed locations with adequate connectivity. Details regarding machine maintenance and co-location agreements are drawn out. Each node must be configured to participate in the resolution scheme and to automatically update its directory structure. The administrative system for a CDN must be concerned with both hardware and content availability. Failed hardware components must be reported as well as failed or corrupted data replication. CDNs can be scalable, but due to this administrative overhead, not rapidly expandable.

The difficulty in maintaining replication integrity over a CDN is not because of inadequate corruption detection of single files, but the delay caused by the replication of very large files. I2-DSI hosts mirrors of popular Linux distributions. These are CD images stored as files with a .iso extension whose typical size is 660MB. A single release of these distributions usually includes three CDs with an aggregate file size of 1.98 GB. When a new release is published, the I2-DSI system must make four uncorrupted copies of each of the three CDs from the publishing I2-DSI node. Before this occurs, the publishing I2-DSI node must retrieve its copy from an authoritative source. It would take approximately eleven hours to complete the replication on I2-DSI under optimal network conditions and perfect job scheduling. It is common for full replication to take twenty-four hours or more.

This delay creates serious problems. Since the replication is a two-step process, users directed to the publishing node will be able to access the Linux files before users directed to nodes that have not finished replicating. Therefore, users nearer to mirrors may receive the older version of the content and they cannot override the automatic mirror selection in order to get the new content. Also, the master site is not only sending replicas to the mirrors, but it is also serving client requests, which greatly adds to its load. Once other machines finish replicating, more users will have access to the files, but inconsistencies will be obvious until the process is finished. Strategies for reducing user load on nodes doing replication have been suggested, but tend to increase the duration of the process by reducing the level of parallelism. Even when the replication is finished, the traffic generated by the Linux user community reduces the delivery performance of other content hosted by I2-DSI resulting in an overall cancellation of CDN benefits.

3 exDN: Ad Hoc Content Distribution Networks

What is needed is a more flexible framework for moving content to distribution sites, decentralized load-balancing to ensure use of all available resources while maintaining scalability, the ability to quickly add more replicas as demand requires, and improve throughput to end users. After designing and implementing the Internet2-DSI infrastructure, the LoCI Lab developed the Network Storage Stack to provide storage as a scalable and sharable resource to the wide-area [5]. Based on this protocol stack, the LoCI Lab has deployed a Logistical Networking Testbed. Using this testbed, content owners can quickly set up an ad hoc content distribution network.

3.1 The Network Storage Stack and Logistical Networking

The Internet Protocol (IP) Stack serves as the basis for networking communication worldwide. The IP stack has at its foundation a simple service — datagram delivery. The failure mode is simple as well — the datagram either arrives or it does not. The IP layer makes no guarantees. Based on this simple service, more complex services can be built such as TCP, which does offer delivery guarantees. This notion of building complex services on top of unreliable services is based on the end-to-end arguments [13] and it is because of this design the Internet has been able to scale so well.

Using the IP stack as a guideline, the Logistical Computing and Internetworking (LoCI) Lab at the University of Tennessee has developed the Network Storage Stack. The goal of the Network Storage Stack is to add storage resources to the Internet in a sharable, scalable manner just as IP made possible scalable, sharable communication. The parts of the Network Storage Stack are shown in figure 1. Next, we will briefly outline each layer.

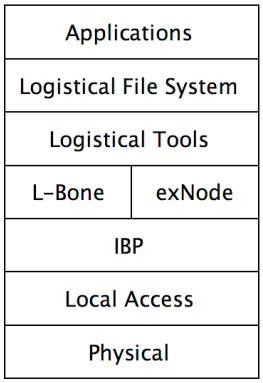

Fig. 1. The layers of the Network Storage Stack abstract necessary details such as hardware devices while exposing as much policy as possible.

3.1.1 Internet Backplane Protocol

The foundation of the Network Storage Stack is the Internet Backplane Protocol (IBP). IBP allows anyone to share disk space or memory space over the network. The server software can run in user space. The user sets the amount of space to share and the maximum time allowed for any single allocation.

IBP is a simple service with the following operations:

Allocate: Request space for a limited time

Store: Write data to an allocation. The writes are append-only.

Load: Read from an allocation.

Copy: Perform a third-party transfer from one allocation to another without bringing the data back to the client. The Copy call uses TCP and is available on every IBP depot.

MCopy: Perform third-party transfers in point-to-point or point-to-multipoint mode. The MCopy call can use many protocols including TCP, reliable and unreliable UDP, and non-IP protocols. The depots allow plug-in modules to provide additional protocols. All depots involved in the transfer must support the specified protocol.

Manage: Change the properties of the allocation.

Because IBP has a few, simple operations, we believe that the service can scale as easily as the Internet. See [5] for a more detailed look at IBP including types of allocations and other policies.

3.1.2 Logistical Backbone

Once a user knows the hostnames and ports for some IBP depots, the user can allocate space and then write and read from it. To help users find IBP depots, part of the next layer up of the stack is the Logistical Backbone (L-Bone). The L-Bone is a resource discovery service that maintains a list of public depots and metadata about those depots.

The metadata includes IBP information such as hostname, port, and allocation duration policy, as well as recent space availability values. The L-Bone periodically polls the depots to update the space availability values. In addition to the IBP metadata, the L-Bone can also store geographic location information as well as machine room characteristics such as data backup policy, power backup availability, and others. The L-Bone client library provides the ability to find depots matching specific criteria (available space, duration policy, IBP allocation type, etc.). It also provides the ability to check if a list of depots are available and still meet the needs of the user.

The L-Bone servers use the Network Weather Service (NWS) [15] to monitor throughput between depots. NWS takes periodic measurements between depots, which it stores and uses to produce forecasts when needed. The L-Bone client library allows the client to provide a source list of depots and a target list of depots to the server. The server will query NWS and then return a matrix of the throughput forecasts.

As of December 2002, the L-Bone lists over 140 depots (Figure 2) on five continents. These depots are serving over 10 TB of publicly available storage.

Fig. 2. As of December 2002, the L-Bone has over 140 IBP depots worldwide that are sharing over 10 TB of publicly available storage.

3.1.3 exNode

Handling a large number of IBP capabilities can be cumbersome. The exNode library automates the handling of IBP capabilities and allows the user to associate metadata with the capabilities.

In the Unix file system, the inode maintains an index of disk blocks that map the file's contents to the disk. In Unix, these blocks must be a uniform size throughout the disk and there must be only one block per offset within the file. The exNode is a similar data structure that allows a user to chain IBP allocations together into a larger logical entity that is much like a network file (Figure 3).

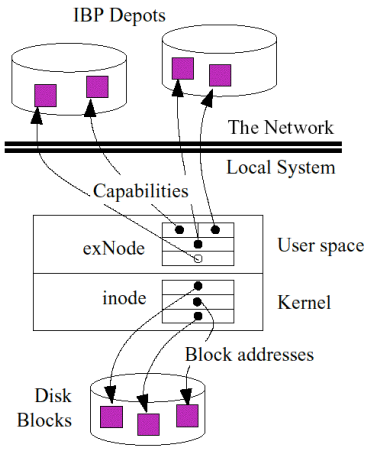

Fig. 3. exNode compared to the inode.

Unlike the inode, the exNode is not limited to fixed size allocations and the exNode may have varying-size IBP allocations. Also, unlike the inode, which is assumed to be on a reliable disk, the exNode holds IBP allocations on a network that is assumed to be unreliable. Because of this inherent unreliability, the exNode may have many replicas of the data over different IBP depots to improve fault-tolerance. See figure 4 for sample exNodes.

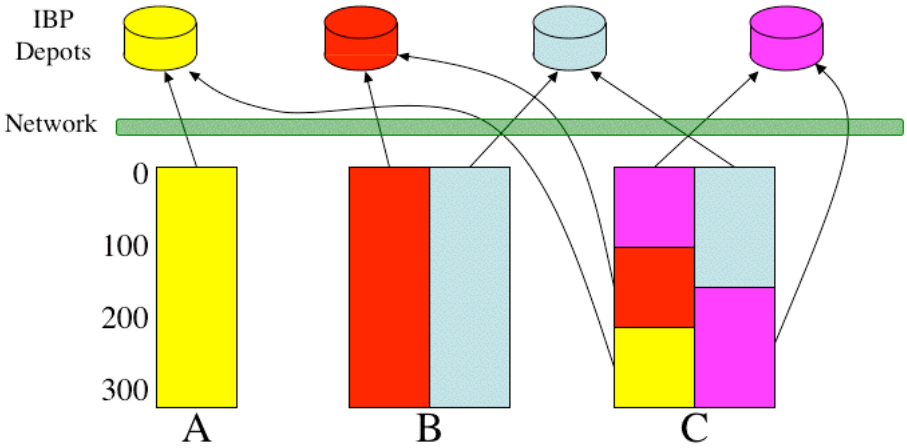

Fig. 4. Sample exNodes.

The exNode also allows for allocations that contain auxiliary data, which is not part of the file extent. For example, the allocations may contain Reed-Solomon code blocks or parity codings. These blocks can improve fault-tolerance with a lower space requirement than replication alone.

The exNode has two major components, arbitrary metadata and mappings. Mappings may have metadata as well. Metadata consists of <name, value, type> triplets where the types are 64-bit integers, 64-bit floating point numbers, character strings, and metadata lists. The metadata lists allow nesting of metadata.

Each mapping may also have function metadata that describes how the data was encoded. The function metadata is a nested list that describes the type of encodings and their relative order. Each function has arguments and optionally metadata. If the user has encrypted and check-summed the data, he can store the encryption algorithm name, the encryption key and the checksum algorithm name using the function metadata. The exNode library does not include any data conditioning functions itself.

The exNode library allows a user to create an exNode, attach mappings to it, store IBP capabilities into the mappings and add metadata to the mappings. When a user wants to write the exNode to disk or to pass it to another user, he can use the exNode library to serialize it to XML. Because of the XML format, exNodes created on one platform can be interchanged with exNodes from any other supported platform.

3.1.4 Logistical Runtime System

Although the L-Bone makes it easier for the user to find depots and the exNode handles IBP capabilities for the user, the user still has to manually request allocations, store the data, create the exNode, attach mappings to the exNode and insert the IBP allocations and metadata into the mappings. The next layer on the Network Storage Stack is the Logistical Runtime System (LoRS). The LoRS layer consists of a C API and a command line interface (CLI) tool set that automate the finding of IBP depots via the L-Bone, creating and using IBP capabilities and creating exNodes. Using LoRS is akin to creating and using network "files".

The LoRS tools provides six basic functions:

Upload: Store data to a network file.
Download: Retrieve data from a network file.
Augment: Add replicas to a network file.
Trim: Remove replicas from a network file.
Refresh: Modify the expiration time of a network file.
List: View the network file's metadata

The LoRS API provides much more fine-grain control. The API can store data from files or memory. Third parties may also use the API to implement new tools or capabilities such as multicast augments and overlay routing.

Both the LoRS tools and API provide end-to-end services. To ensure that the data stored on the IBP depots was not altered in transit or while on disk, LoRS can insert MD5 checksums. During a download, if a block's checksum does not match, the block is discarded and the same extent is downloaded from another source.

To protect data while in transit and while stored on a depot, which should be considered an un-trusted server, LoRS provides multiple types of encryption, including DES. With the API, the application may use additional encryption algorithms and then add the algorithm type and key as function metadata to the exNode.

In addition to replication for improving fault-tolerance, LoRS allows coding blocks to be stored as well. These coding blocks are like the parity blocks used in RAID storage systems. While simple replication may provide an adequate measure of fault-

tolerance, the addition of coding blocks can greatly improve fault-tolerance. Even if a certain extent of the file does not exist in any replica, it may be possible to regenerate that data from the remaining data and the coding blocks.

Lastly, to reduce the amount of data transmitted, stored and retrieved, LoRS supports compression.

3.1.5 Applications Using the Logistical Networking

We have shown that Logistical Networking provides a flexible framework. We now outline several applications where the strengths of Logistical Networking are shown more clearly.

3.1.5.1 Multimedia Streaming

In addition to content delivery (discussed in 3.2), multimedia-streaming servers have the additional requirement for a sustained minimum throughput to provide uninterrupted playback. Commercial multimedia delivery solutions typically require special purpose server software, proprietary protocols and dedicated machinery, which can be quite costly.

Video IBPster [2] demonstrated at iGrid2002 that Logistical Networking could reliably deliver high bit-rate video (10-15 Mbps) using freely available, generic infrastructure (typically desktop class PCs located around the world). A progress-driven redundancy download algorithm makes it possible to maintain the minimum necessary throughput for streaming multimedia on top of un-tuned TCP [11].

3.1.5.2 Overlay Routing

In addition to third party point-to-point transfers, Logistical Networking supports a variety of overlay routing techniques.

Store-and-Forward Routing

While it is relatively well known that store-and-forward routing is effective at the link level, the abstractions provided by the TCP/IP stack encourage programmers to avoid similar algorithms at the application level. Indeed, TCP provides a reliable, point-to-point view of the network. However, significant gains in performance can be made simply by applying low-level routing techniques to application-level transfers. Logistical networking enables source-directed overlay routing by providing explicit control of network buffers to applications.

In particular, we have conducted a number of preliminary experiments in the area of store-and-forward overlay routing. Routing is accomplished by replacing a single point-to-point data transfer with a number of smaller transfers through intermediate buffers. For instance, a single transfer from A to C can be replaced with two transfers: A to B, followed by B to C. Early experiments indicate performance gains of approximately 75% on average.

Multi-Path Routing

In addition to allowing applications to benefit from link-level type data routing, Logistical Networking also allows applications to make use of novel new approaches to data transfer. A promising example of such an approach is multi-path routing.

Multi-path routing is the simultaneous transfer of multiple data blocks via separate intermediaries. This transfer can be conceptualized as a scatter from the source to

some number of intermediaries, followed by a gather from those intermediaries to the ultimate destination. This approach is similar to that taken by the NetLets project at ORNL [12], and we expect to see similar performance gains.

Overlay Multicast

Logistical networking can also be used to implement overlay multicast, or point-to-multipoint routing. Here, multicast trees are built from a number of logistical storage depots and transfers within distinct sub-trees occur in parallel. Transfers between nodes of the multicast tree may take advantage of any of the previously described routing mechanisms in order to enhance performance. Additionally, IP multicast may be used in networks where it is supported.

3.1.5.3 Data Caching

By providing storage within the network, Logistical Computing allows applications to take advantage of data locality. Two uses of data caching are checkpointing and data staging.

Checkpointing

Checkpointing is a well-understood means for dealing with the possibility of failures. However, checkpointing a large distributed application can be problematic, particularly if the application is running on some sort of RPC platform without dedicated storage (e.g., NetSolve [1]).

Logistical networking is uniquely suited to the problem of managing checkpoints in distributed computing systems. The time-limited storage provided by IBP maps cleanly to the intrinsically short duration of checkpoints. Furthermore, the anonymous nature of logistical storage ensures that applications have access to storage as needed. Moreover, the geographic distribution of logistical storage depots makes it likely that the application will be able to access storage that is nearby.

A second application of checkpointing is to facilitate the migration of compute jobs from one node to another. Again, the capabilities of a logistical network are well suited to this sort of interaction. Process state can be checkpointed to logistical storage and moved quickly to another depot, perhaps across the world, for resumption on another compute platform.

Data Staging

We define data staging as pre-staging inputs and caching of intermediate values in distributed systems. One project that is exploring the potential of Logistical Networking for data management is NetSolve [1, 3].

Typically NetSolve clients pass the data and the operation to be performed to the NetSolve server. The server performs the computation and returns the result to the client. If the output of one computation is needed as input to another computation, it needs to be resent by the client.

In order to eliminate returning intermediate values to the client, NetSolve can store the first computation's results using the LoRS API and generate an exNode. The NetSolve server then returns the exNode to the client. Next, the client sends the exNode and the requested operation as input parameters to the next NetSolve server. This lightens the burden on the client and reduces the time to complete all the computations [5].

3.1.5.4 Distributed, Parallel IO

The Terascale Supernova Initiative (TSI) [9] is a collaborative, multi-disciplinary effort sponsored by the Department of Energy's SciDAC program. TSI is developing models for core collapse supernovae and enabling technologies in radiation transport, radiation hydrodynamics, nuclear structure, linear algebra, and collaborative visualization. The TSI team is composed of members from one national lab and eight universities.

The centerpiece of the TSI effort is a large-scale magneto-hydrodynamics simulation. To date, simulations have been run on 320^3 and 640^3 node meshes, and runs on 1024^3 node meshes are planned for the near future. The storage requirements of this simulation are massive: small runs produce output data on the order of 120 GB, while the largest runs are expected to produce data on the order of 20 TB.

The storage and movement of such large data sets between institutions is a major challenge for the TSI effort. In order to meet this challenge, TSI is a major integrator of Logistical Networking technologies. Logistical networking will be used to augment or replace legacy systems such as HPSS [10]. While such systems are able to meet the storage requirements of TSI, they are unable to address the distributed collaboration requirements. In particular, data managed by HPSS must reside in one mass storage system at a time, at one of the TSI sites. When users at another site need to access the data, it must be packaged and shipped over the network using standard data transfer mechanisms such as FTP.

Logistical networking will be used as the default output mechanism for TSI application codes. Rather than outputting to a parallel file system for staging to HPSS, applications will output data directly to IBP buffers. Once a portion of the output is generated and stored in IBP, that IBP allocation will be added to the exNode that contains all of the output. The exNode will then be replicated to logistical depots at collaborating sites and to mass storage for archival purposes. The primary advantage of this scheme is that data will reside at all sites simultaneously, allowing collaborators across many different administrative domains to access the data as soon as it is available. This will greatly simplify the complex data packaging and transport requirements of the project.

Additionally, the overlay routing techniques described previously can be applied to significantly speed up the necessary data transfer operations. Since the size of the output data will be quite large, the use of explicit overlay routing and the ability to use different protocols within IBP's mcopy() operation will greatly reduce transfer times.

3.2 Creating Ad Hoc Content Distribution Networks

The I2-DSI project has leveraged the speed, storage, and resolution service of the Logistical Networking Stack to create an exNode Distribution Network (exDN) that overcomes many of the problems discussed in 2.1 and 2.2.

Using a combination of HTTP and IBP services, websites that consume large amounts of bandwidth because of large file delivery can store the files in IBP depots and provide links to their respective exNodes. Others who also suffer from high number of simultaneous visitors can replicate the HTML portions of the site across a CDN such as I2-DSI. The replication integrity problem is nullified by separating web con-

tent from data content since traditional CDN synchronization techniques are better suited to the small size of HTML-type files. Additional copies at additional locations of popular data files can be generated quickly within IBP storage without disrupting user access or compromising data integrity.

Users who visit the I2-DSI Linux mirrors are able to retrieve the CD images via HTTP or using the LoRS tools. An exNode for each of the CD images is advertised as well as the .iso files. The user can download the exNode and use LoRS to retrieve the CD image from IBP storage. A browser plug-in for LoRS is available to reduce the number of steps and give IBP storage a web interface. The download of the file starts automatically when a link to an exNode is selected if the plug-in is installed.

Once the publishing node for the Linux I2-DSI channels receives a full copy of the latest distribution, each CD image is uploaded into IBP storage to generate its exNode. The upload procedure uses the lors_upload tool in conjunction with the -l (location) argument and creates twelve copies of the .iso file across 30-36 IBP nodes spanning the continental United States in approximately 20-25 minutes. In other words, 7.9 GBs of data are transferred and stored over the wide-area network with an average throughput of 42 Mbps compared to the 2 Mbps that I2-DSI gets using rsync. The resulting exNodes are published and replicated across I2-DSI in a matter of minutes giving visitors to all mirrors immediate access the new content even if the traditional I2-DSI rsync copies are still being copied.

Forty-five minutes is a realistic download time for retrieving a Linux CD image over HTTP from a non-saturated source. If the user has at least a 10 Mbps network connection, she can download the same file using LoRS in approximately eleven minutes. This is made possible by the advanced download algorithm and the multi-threaded capabilities of the LoRS download tool [11]. The LoRS download tool uses an adaptive algorithm to determine which file fragments to retrieve from each source node. The LoRS download tool handles a failure from one source (i.e. if the source is not available or is it not accepting connections) and automatically tries retrieving the block from an alternate source. The download tool adapts to the various sources performance. For example, if a closer node is overloaded and its performance drops, the download tool will adapt and retrieve from more remote copies that perform better. This is highly significant because the resolution service is exNode-specific and performed by the client instead of using the entire system and being centralized, as is the case of I2-DSI.

Given the time-sensitive nature of IBP storage, methods must be implemented to account for exNode decay. It is possible that allocations and IBP depots used during the initial upload procedure will become unreachable/unavailable after the exNode is generated and published via HTTP. Strategies for simulating persistence are developed using the lors_refresh tool and are assisted by increasing the number of copies and checking for problems with the lors_trim tool. The content owner is responsible for maintaining a satisfactory level of reliability. Another concern is the level of access intended by the content owner. Making the full exNode available to users grants them the ability to make additional copies or delete allocations. To prevent this, content owners can create read-only versions of exNodes by removing the encoded administrative capabilities.

4 Conclusions

We have described an architecture that allows data to be stored for a limited time in the wide-area without requiring a user account. We have also put in place a testbed open to researchers to explore the capabilities of sharable storage that has over 140 nodes serving over 10 TB of storage. We have specifically detailed how to create an ad hoc content distribution network using the testbed that has the benefits of improved performance and better resource usage than traditional mirroring and content distribution networks.

The IBP, L-Bone, exNode and LoRS software is open-source and available for download (http://loci.cs.utk.edu). People interested in adding more public storage resources to the L-Bone can do so easily. They can simply download and install the IBP server software. They would then register the depot with the public L-Bone.

5 Acknowledgements

This material is based upon work supported by the National Science Foundation under grants ANI-0222945, ANI-9980203, EIA-9972889, EIA-9975015 and EIA-0204007, the Department of Energy under grant DE-FC02-01ER25465, and the University of Tennessee Center for Information Technology Research. The authors would like to acknowledge Alex Bassi, Xiang Li and Yong Zheng for their work on IBP, Ying Ding for her contributions with the LoRS download algorithm and the NetSolve team.

References

1. Arnold, D. et al. User's Guide to NetSolve V1.4.1. *Techical Report, ICL-UT-02-05*. University of Tennessee Innovative Computing Laboratory. June, 2002.
2. Atchley, S., Soltesz, S., Plank, J. S., and Beck, M. Video IBPster. Accepted for publication in *Future Generation of Computer Systems*.
3. Beck, M., et. al. Middleware for the use of storage in communication. Accepted for publication in *Parallel Computing*.
4. Beck, M., Moore, T. The I-2 DSI Project: An Architecture for Internet Content Channels. *Computer Netowrking and ISDN Systems*, 1998 30(23-23): pp. 2141-2148.
5. Beck, M., Moore, T., and Plank, J.S. An end-to-end approach to globally scalable network storage. *Proc. of ACM SIGCOMM '02*, Pittsburgh, August 2002.
6. http://dsi.internet2.edu/
7. http://samba.anu.edu.au/rsync/
8. http://www.cisco.com/univercd/cc/td/doc/product/iaabu/distrdir/dd2501/index.htm
9. http://www.phy.ornl.gov/tsi/
10. http://www4.clearlake.ibm.com/hpss/index.jsp
11. Plank, J., Atchley, S., Ding, Y. and Beck, M., Algorithms for High Performance, Wide-Area, Distributed File Downloads. *Technical Report CS-02-485*, University of Tennessee Department of Computer Science, October 8, 2002.

12. Rao, N. S. V. Multiple paths for end-to-end delay minimization in distributed computing over the Internet. *Proc. of the ACM/IEEE Conference on Supercomputing*. Denver, Colorado, 2001.
13. Saltzer, J. H., Reed, D. P., and Clark, D. D.. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277-288, November 1984.
14. R. Tewari, M. Dahlin, H.M. Vin, and J. Kay. Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet. *Technical Report CS98-04*, Department of Computer Sciences, UT Austin, Austin, Texas, USA, May 1998.
15. Wolski, R., Spring, N., and Hayes, J. The Network Weather Service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5-6):757-768 , 1999.