The Logistical Networking Testbed

Scott Atchley Micah Beck Jeremy Millar
Terry Moore Jim Plank Stephen Soltesz

Logistical Computing and Internetworking Laboratory
Tech Report UT-CS-02-496
Computer Science Department
University of Tennessee

{atchley, mbeck, millar, tmoore, plank, soltesz}@cs.utk.edu
http://www.cs.utk.edu/~plank/plank/papers/CS-02-496.html

ABSTRACT

We describe the Logistical Networking Testbed, built using the Network Storage Stack, and sample applications that use the 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 sample applications include content delivery, multimedia streaming, overlay routing, checkpointing, and an integrated example. This networking testbed provides over 10 TB of shared storage that is available to all for research.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *distributed networks, network communications, store and forward networks.*

General Terms

Performance, Design, Reliability, Experimentation, Security.

Keywords

Logistical Networking, store and forward, network storage, end-to-end guarantees, scalability, wide area storage, Internet Backplane Protocol, IBP, exNode, LoRS, Logistical Runtime System, Logistical Backbone, L-Bone, overlay routing, content delivery, caching, asynchronous storage, peer to peer.

1. INTRODUCTION

Logistical Networking is the term we use to describe the combination of the movement and storage of data. The Logistical Computing and Internetworking Lab at the University of Tennessee has developed the Network Storage Stack, which is modeled after the TCP/IP stack. The purpose of this stack is to provide a **scalably sharable** storage service that will increase efficiency, performance, and functionality of distributed applications [4].

The paper is organized as follows: In section 2 we provide an overview of the components of the Network Storage Stack including the Logistical Runtime System, which is our testbed. In section 3 we describe applications that have been or are being built using this infrastructure. These applications include content delivery (higher performance downloads of widely distributed Linux ISO images), multimedia streaming (viewing DVD-quality video stored around the world), overlay routing (point-to-point with single and multiple intermediate nodes as well as point-to-multipoint), checkpointing, and, finally, an integrated example using all the above.

2. THE NETWORK STORAGE STACK

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 [10] 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 outline each layer.

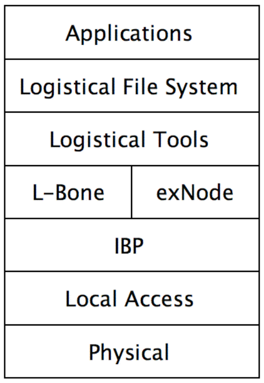

Figure 1: The Network Storage Stack.

2.1 IBP

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 (no root access required). The user sets the amount of space to share and the maximum time allowed for any single allocation. The user could set that time to be infinite, but if the client ever loses the pointer to that space, the space will be lost forever. Typically, LoCI suggests that depot owners set the duration between one day and one month for disk depots and less than a day for memory depots.

IBP is a simple service with the following operations:

Allocate

A user can request an allocation (a specific amount of space on the machine for a specific amount of time). If the depot (IBP server) has the space and the time request is less than or equal to the depot's maximum duration policy, then the request is granted and the depot sends a capability set (keys) to the user. The keys grant write (store), read (load), and manage access to the space. These keys are long, variable length, random strings. Note, that the key does not infer anything about the depot's file system structure.

Store

Once granted a set of capabilities, the user may write data to an allocation. The writes are append-only.

Load

After data has been written to an allocation, the user may read from that allocation, beginning from any offset within the allocation.

Copy

Copy allows the user to perform a third-party transfer that is to send data from one allocation to another allocation (point-to-point) without bringing the data back to the client. The Copy call uses TCP and is available on every IBP depot.

MCopy

MCopy allows the user to 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

Manage allows the user to change the properties of the allocation. The user may request a renewal of the lease (extend the expiration time), a shortening of the lease, enlarge the space, shorten the space or free the allocation.

Given the above, a client may request 1 GB for 5 days. If the depot has the space and 5 days is within its duration policy, then the depot will return a set of capabilities to the client. Once the client has the capabilities, she may write to the space, read from the space, copy it to another allocation, etc. If the client gives the read capability to someone else, they can read the stored data as well. This is a very basic type of file sharing and is similar to a soft link.

The IBP depot supports two policies for allocation revocation, Hard and Soft. The depot treats Hard allocations like any other file on the hard drive and will not revoke it until it expires or until a user frees it with a Manage call. Since other users and other processes on that machine will also use disk resources, it is possible to use all the disk space on the drive (or array). IBP also supports the policy of Soft allocations, which the depot may prematurely revoke if the available hard drive space falls beneath a preset threshold. This allows a depot owner to initially allow the depot to allocate most of a new, empty drive and then as other applications consume disk space, the depot revokes Soft allocations to maintain enough disk space to prevent write-to-disk failures.

Because IBP has a few, simple operations, we believe that the service can scale as easily as the Internet. See [4] for a more detailed look at IBP including types of allocations and other policies.

2.2 L-Bone

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 in-

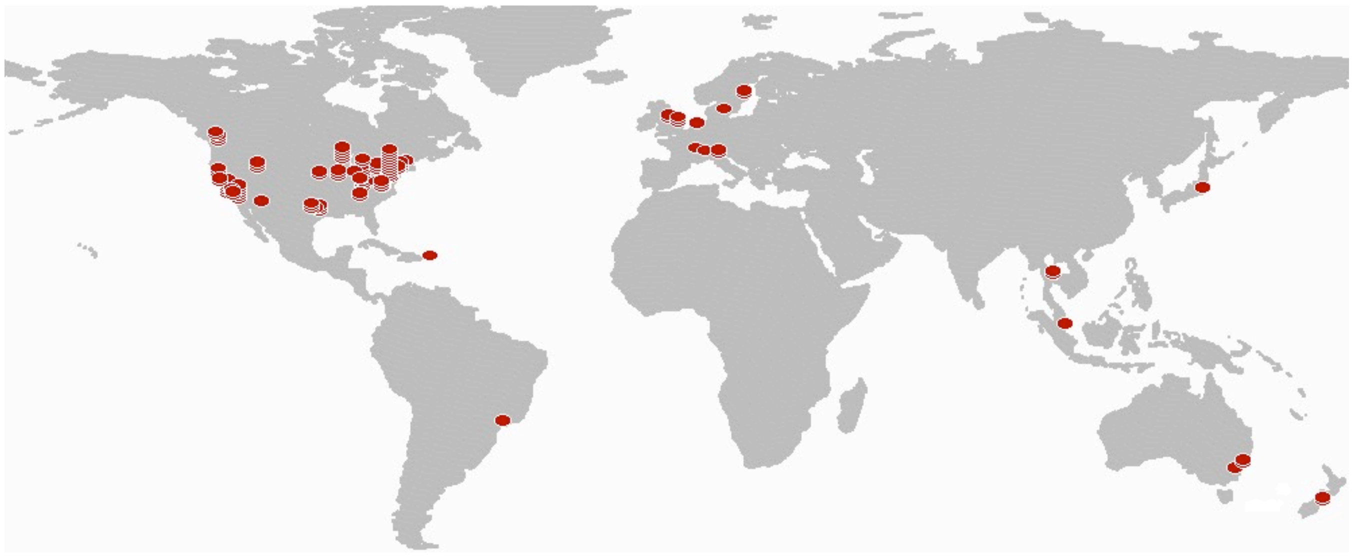

Figure 2: The L-Bone has over 140 depots worldwide share over 10 TB of storage.

formation 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) [11] 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.

2.3 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).

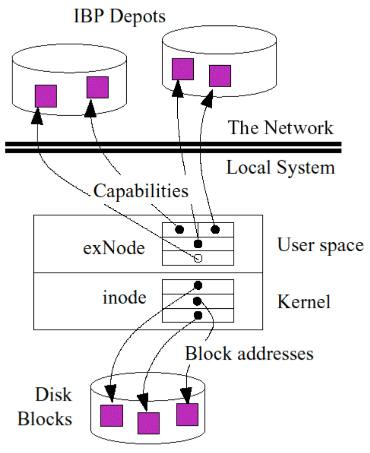

Figure 3: exNode compared to inode.

Even though current IBP allocations have a 2 GB limit, the exNode will allow the user to chain two billion IBP allocations together, representing a 4 Exabyte (2^{62}) file.

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.

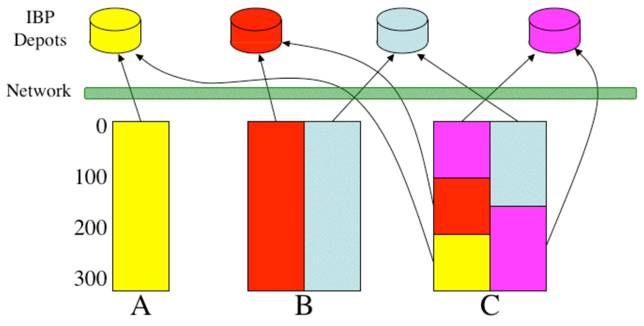

Figure 4: Sample exNodes.

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.

An individual mapping may have an allocation, which can be a set of URLs. Typically, these are IBP capabilities, but they may also be other URLs, e.g. HTTP, FTP, or GridFTP URLs. Typically, the mapping will also have specialized metadata, such as logical offset, length and physical offset (where in the URL to begin reading) as well as user-defined, arbitrary 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 a mapping to it, store IBP capabilities into the mapping and add metadata to the mapping. 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.

2.4 LoRS

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. APPLICATIONS

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 Content Distribution

The dissemination of Linux ISO images ranks as one of the thorniest content distribution problems. The large size of each image (approx. 650 MB) and large demand conspire to make ISO delivery a problem for the major distribution maintainers. The traditional solution has been to increase the aggregate bandwidth available to users via mirroring. Additionally, geographic and network distribution of mirrors provides users with potentially lower latencies.

However, mirroring is not without disadvantages. In particular, users tend to consistently use the same mirrors. Generally speaking, users make use of mirrors at the top of the list, or those with a reputation for speed. In the large, this behavior tends to cluster load on a small number of mirrors, often resulting in overload while other resources are under-utilized.

The common solution to the problems of mirroring is the application of content distribution networks, e.g., Akamai [6] or I2-DSI [5]. These networks operate by caching or replicating content at strategic locations throughout the network. Unlike mirrors, however, content distribution networks also manage URL redirection such that users do not need to choose a cache or replica. Instead, the content distribution network chooses the copy best able to satisfy the user and redirects the request. The best copy is determined by a number of metrics, including hop count from user to cache, load on the replica server, available throughput from user to cache, etc.

Content distribution networks are not without their problems. Perhaps non-intuitively, they remain susceptible to flash crowds, although to a lesser extent than traditional content delivery systems. This phenomenon has been reported in the literature [7] and empirically verified by the I2-DSI project. I2-DSI is a replica-based content distribution network that makes Linux ISO images available to the Internet2 community. During the four days following the RedHat 7.3 release in April 2002, I2-DSI moved approximately 1.1 TB of data. Additionally, the I2-DSI network experienced an extremely high load during this period, resulting in refused connections and an inability to access non-Linux content.

While content distribution networks address the problems of simple mirroring, a solution to the flash crowd problem is clearly needed in order to efficiently distribute Linux content. We propose that logistical networking technologies can provide this solution by enabling the creation of extensible, adhoc content distribution networks. The exNode Distribution Network (exDN) utilizes the existing logistical and I2-DSI infrastructures to distribute Linux ISO images as exNodes.

Each ISO image is uploaded to the logistical network via LoRS. A number of geographically dispersed replicas (typically 12) are created in order to provide fault tolerance and effective load balancing. The resulting exNodes (one per ISO image) are collected and placed onto the I2-DSI network. Logistical clients retrieve the Linux exNodes from I2-DSI. They then download the ISO images from the exNodes via the LoRS tools.

exDN overcomes the limitations of traditional content distribution networks by leveraging logistical technologies to quickly add replicas as needed to meet demand. In the event of a flash crowd, replicas are added via the LoRS tools and the Linux exNodes are replaced. Load balancing across the exDN network is achieved via the LoRS download algorithms. The LoRS download tools use an adaptive algorithm (described in detail in section 3.2) that attempts to retrieve data from multiple sources in parallel. If an IBP depot is overloaded, the download tools automatically use the other depots to retrieve the data. An additional benefit of exDN is the improved performance garnered by utilizing parallel TCP streams.

3.2 Multimedia Streaming

Servers in any kind of content distribution network will suffer from finite availability, limited connections or gross capacity over a single connection. While the general goal of "faster download" helps motivate solutions such as exDN or other content distribution strategies, no hard throughput requirements exist for delivering Linux ISOs or other cached content.

However, in the case of streaming multimedia, a predictable quality of service is necessary to sustain a minimum throughput to provide uninterrupted playback.

Commercial multimedia delivery solutions typically require special purpose server software, proprietary protocols and dedicated machinery, which is quite costly. If the provider wishes to prevent a single point of failure, then mirrors or a content delivery network is required, greatly increasing the cost. An ideal solution is to find some way of combining the benefits of exDN with predictable quality of service.

Video IBPster [2] demonstrated at iGrid2002 that Logistical Networking could reliably deliver high bit-rate video 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 [8].

Figures 5 and 6 illustrate how the download algorithm behaves. Initially, a pool of available threads is assigned equally across available replicas. Past a certain threshold the algorithm replicates work to improve overall throughput. In this way, higher priority is placed on pending blocks before those needed later. If one block lags, an available thread is assigned to the lagging block. Whichever thread finishes first "wins" and both threads are freed for work on new blocks or other lagging blocks.

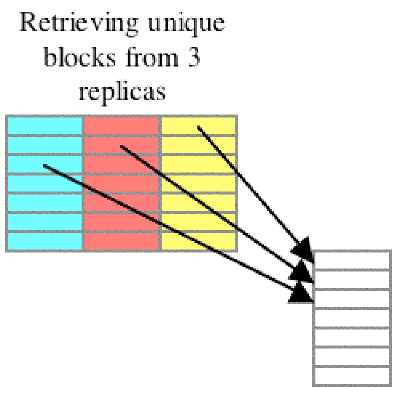

Figure 5: Progressive-Redundancy Algorithm.

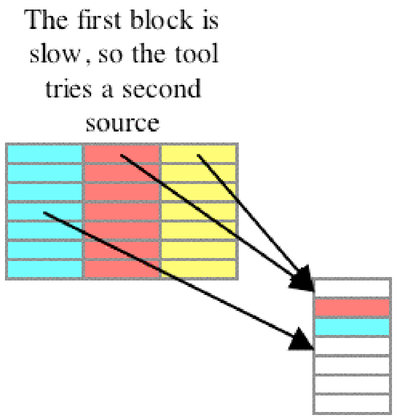

Figure 6: Redundant work when needed.

At the iGrid2002 and SC2002 conferences, we were able to stream 10 and 15 Mbps video from Australia to the conference sites in Amsterdam and Baltimore.

3.3 Overlay Routing

In addition to 3rd party point-to-point transfers, Logistical Networking supports a variety of overlay routing techniques.

3.3.1 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.

Logistical networking admits a variety of variations on this theme. Using multiple intermediaries is a trivial extension. A more interesting variation is "interleaved store-and-forward". In this variation, the data to be transferred is fragmented into a number of blocks, and each block is transferred using the previously described store-and-forward mechanism. Since block transfers are logically separate, there is no need to wait for all of the blocks to arrive at the intermediary before forwarding them. Interleaved performance is no worse than the straightforward store-and-forward scheme, and it remains to be seen if it provides any advantage.

A third variation involves treating the intermediary as a FIFO buffer. In this case, there is no need to wait for all of a block to arrive at the intermediary before forwarding can begin. Although we've conducted only the most preliminary of experiments, we expect this method to perform best.

3.3.2 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 [9], and we expect to see similar performance gains.

3.3.3 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.

3.4 Data Staging

Another application of Logistical Networking is data staging for distributed computation. Pre-staging inputs and caching of intermediate values improves the performance due to data locality. 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 (Figure 7).

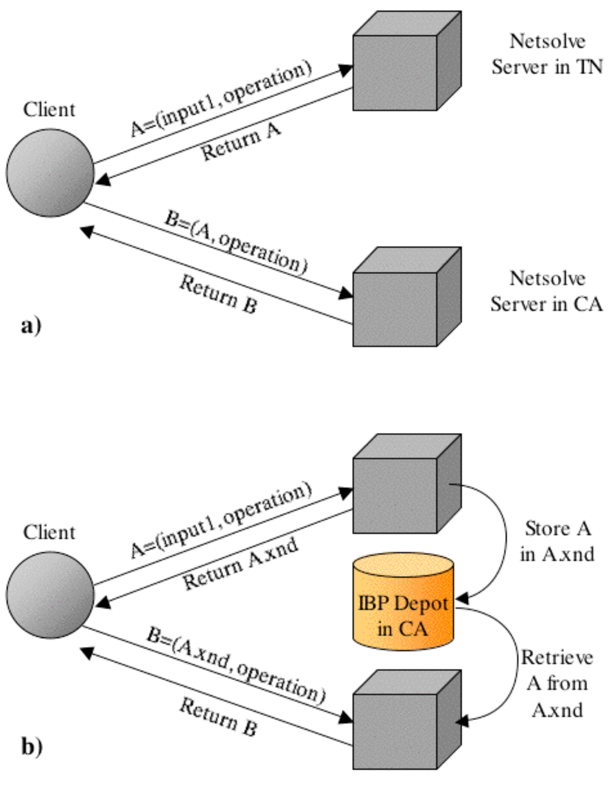

Figure 7: In a), NetSolve returns all intermediate results to the client which must be re-sent before the next computation. In b), NetSolve uses LoRS for data staging, which minimizes traffic to and from the client and also reduces latency for the next computation.

3.5 Checkpointing

As computations increase in complexity and time to completion, and as parallel computing on clusters becomes more prevalent, the potential for compute node failure also increases. 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 [ns]).

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 a nearby compute platform. Logistical networking is poised to enable large-scale data and process migration for the next generation of geographically distributed computing environments (i.e. grids).

3.6 Integrated Example

The Terascale Supernova Initiative (TSI) [2] 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 [1]. 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.

4. INVITATION TO USE THE TESTBED

The Logistical Networking Testbed is open to all researchers. No account is needed to use the publicly available storage.

IBP, the L-Bone, the exNode and the Logistical Runtime System are supported by the Logistical Computing and Internetworking (LoCI) Laboratory and are available at http://loci.cs.utk.edu. All of the packages compile on Linux, Solaris and MacOSX. We also provide a set of binaries for Windows2000 that require Cygwin.

The power of this suite of software has been demonstrated with several applications:

IBP-Mail is an application that allows users to mail large files to other users by uploading them into the network, and then mailing the exNode to the recipient.

IBPster is a media player that plays audio and video files stored in exNodes on the network. The files may be arbitrarily striped and replicated, and the player performs a streaming download to play them.

IBPvo is an application that allows users may schedule recording of television programs into IBP allocations. The user is sent an exNode, which he or she may use to re-play the program from the network storage buffers.

LoCI supports a main L-Bone (http://loci.cs.utk.edu/lbone/cgi-bin/lbone_list_view.cgi) that currently is composed of 140 depots in North and South America, Europe, Asia and Oceania (Australia and New Zealand), serving over ten terabytes of network storage. This storage is available for all to use without need of an account. The LoRS software has been designed to run without an L-Bone, or for users to configure their own, private L-Bone.

People interested in adding more public storage resources to the L-Bone can do so easily. They can simply download and install the IBP depot software. Then they would register the depot with the public L-Bone at http://loci.cs.utk.edu/lbone/cgi-bin/lbone_depot_modify.cgi?action=add.

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, the NetSolve team, Nagi Rao for discussing his work on multiple path overlay routing, Rich Wolski and Martin Swany for their work on the Network Weather Service and the Logistical Session Layer, and Hunter Hagewood for his systems support.

6. REFERENCES

[1] Arnold, D., et. al. User's Guide to NetSolve V1.4.1. Technical 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., and Plank, J.S. An end-to-end approach to globally scalable network storage. *Proc. of ACM SIGCOMM '02*, Pittsburgh, August 2002.

[5] http://dsi.internet2.edu.

[6] http://www.akamai.com.

[7] Jung, J., Krishnamurthy, B., and Rabinovich, M. Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites. *Proc. 11th Intl. World Wide Web Conference*. Honolulu, HI, 2002.

[8] Plank, J.S., 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.

[9] 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.

[10] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[11] R. Wolski, N. Spring, and J. Hayes. The Network Weather Service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5-6):757–768, 1999.