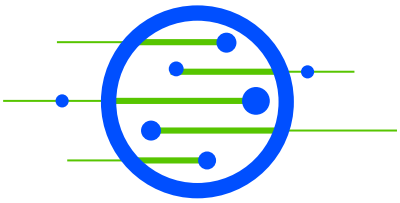

An End-to-End Approach to Globally Scalable Network Storage

Micah Beck
University of Tennesee

Terry Moore
University of Tennessee

James S. Plank
University of Tennessee

PDN-02-007
November 2002

Appears in: *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, August 2002.

An End-to-End Approach to Globally Scalable Network Storage

Micah Beck Terry Moore James S. Plank

Logistical Computing and Internetworking Laboratory
Computer Science Department
University of Tennessee
1 865 974 3548

{mbeck, tmoore, plank}@cs.utk.edu

ABSTRACT

This paper discusses the application of end-to-end design principles, which are characteristic of the architecture of the Internet, to network storage. While putting storage into the network fabric may seem to contradict end-to-end arguments, we try to show not only that there is no contradiction, but also that adherence to such an approach is the key to achieving true scalability of shared network storage. After discussing end-to-end arguments with respect to several properties of network storage, we describe the Internet Backplane Protocol and the exNode, which are tools that have been designed to create a network storage substrate that adheres to these principles. The name for this approach is Logistical Networking, and we believe its use is fundamental to the future of truly scalable communication.

Categories and Subject Descriptors

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

General Terms

Design

Keywords

Logistical Networking, store and forward network, asynchronous communications, network storage, end-to-end design, scalability, wide area storage, Internet Backplane Protocol, IBP, exNode.

1. INTRODUCTION

Logistical Networking seeks to model communication in *both* its synchronous and asynchronous aspects. One of its essential goals, therefore, is to create a resource fabric that unifies the co-management and co-scheduling of data transport and data storage, much as military or industrial logistics treat transportation lines and storage depots as coordinate elements of one infrastructure. To address the needs of a the Internet community, however, it must also achieve this goal in a way that can scale up in terms of the number of users and nodes it supports, the range of geographic, network and administrative boundaries it spans, and the level of provisioning it accommodates. Our view is that if end-to-end design principles are applied to network storage, then a **scalably sharable** communicative infrastructure with persistence can be created that will increase the efficiency, performance, and functionality of distributed applications of all types.

Our position is that introducing such a standard, flexible, exposed buffer service, which is based on the application of end-to-end principles to storage resources, will change the way that networking is done.

SIGCOMM'02, August 19-23, 2002, Pittsburgh, Pennsylvania, USA

2. NETWORKING, STORAGE, AND END-TO-END ARGUMENTS

Consider a generalized scenario in which a quantum of data originates from a node N_s at time t_s and either does or does not arrive at a destination at a node N_r at time t_r, and if it does arrive it may be corrupted.

If N_s and N_r can be members of a globally scalable network, and t_r-t_s is a delay that the delivery mechanism seeks in general to minimize, then this scenario fits the characteristics of layers 1 through 3 of the network stack. Under these conditions, minimizing both the delay and the probability of corruption, while at the same time maximizing the probability of delivery, is understood to be problematic. Sometimes one of these properties has to be compromised for the others.

If N_s and N_r are either identical or members of a small, closed network and there is no a priori bound on t_r-t_s, then this scenario fits the conventional characterization of a storage device connected directly to a node or a storage area network. It is traditionally understood that for closely coupled storage, delay and probability of corruption can be very low while availability is very high.

The characterization of data delivery in the network has led to the adoption of the *end-to-end* approach (also known as "end-to-end arguments" or "principles" [16, 17]) to network services. One simple formulation of the end-to-end principle is that any guarantees in a communication over and above a bare minimum of functionality ought to be applied at the endpoints. Although greater complexity and functionality in the middle of the network may enhance performance or reliability, the ultimate responsibility for ensuring that communications have the required properties, which may vary with the application, rests on the endpoints. The charm of the end-to-end approach stems from the fact that it does not rely on the network to be timely or accurate in the delivery of any particular packet, only that high delay and

corruption be of sufficiently low probability and be inflicted fairly on competing network participants. This allows for a high degree of autonomy and faulty behavior in the operation of the network, and leads to the ultimate goal of global network architecture: *scalability*!

But application programmers have a difficult time using services that are unpredictable in every dimension, which means that the end-to-end approach requires that higher software layers be developed in order to implement more predictable services on top of the unpredictable lower layers. TCP achieves this by requiring the sender N_s to maintain a copy of all data until its receipt is corroborated by the receiver N_r. Of course the construction of new services that are predictable in one dimension tends to burden the resources of the endpoints or to worsen unpredictability in other dimensions. TCP, for instance, achieves a high probability of accuracy at the cost of increased variation in delay and lowered availability.

The conventional characterization of closely coupled storage given above makes it clear that the end-to-end approach is not relevant in that case. If a storage device can be relied upon to operate with predictable delay, high accuracy and high availability, then it can be used without the burden of implementing layered end-to-end services. As long as the storage device can be relied upon, the writer N_s and reader N_r can be kept simple.

However, assuming that storage is reliable can impose a cost if the storage fails. Until the advent of cheap, unreliable disks for personal computers, the only approach to mitigating this cost was redundant storage of data on highly reliable storage systems. Of course, reliability comes at a price, and the price of reliable storage is quite high – redundant storage on reliable disks was reserved for mission-critical systems.

The advent of cheap, disks led to the realization that cheap and highly reliable storage can be implemented by using storage devices that are, in aggregate, less reliable. By weakening their assumptions about the high availability of storage, the designers of RAID storage can indeed make use of cheaper and less reliable disks [6]. However, high reliability has to be regained by layering algorithms for redundantly encoded storage of data on multiple disks and, in the face of failure, active reconstruction of the missing data. While these algorithms are often implemented in hardware, they unmistakably form a layer that is above the physical storage and provides greater predictability.

It is interesting to note that the developers of RAID decided to weaken their assumptions about storage in only one dimension: availability. Classical RAID algorithms rely strongly on the assumption that if data is retrieved from a storage device it is retrieved accurately, and these algorithms will fail if this assumption is violated. Weakening the assumption of accuracy leads to imposing the burden of checksum calculation on higher layers, which the developers of RAID sought to avoid. This decision has in fact caused problems due to the use of very inexpensive disks in RAID systems that not only display low availability but also can be inaccurate [Network Appliance, personal communication, 2001]

Globally scalable network storage, meaning storage systems attached to the global data transmission network and accessed from arbitrary endpoints, presents a problem because it does not fit the conventional characterization of closely coupled storage. As in data transmission, N_s and N_r can be different members of the network, but as in storage there is no a priori bound on t_r-t_s. Because it relies on the network, delay, accuracy and availability cannot be controlled. In other words, the shortcuts that are available to closely coupled storage do not apply to network storage. The solution is to embrace the end-to-end approach.

The end-to-end approach to network storage is an extension of the design approach of RAID, but taken to the extreme, abandoning strong assumptions of predictable delay and high accuracy along with high availability. In the world of scalably sharable storage, data written/sent to storage may or may not be accurately retrievable by the reader/receiver. While some basic guarantees may be provided by network storage, we accept that the stronger they are the more likely they are to be violated, resulting in failure of the network storage service as perceived by the endpoint.

The rest of this paper considers a number of formulations of the possible properties of network storage, the implications of those properties in the implementation and robustness of network storage services, and the impact on the service provided to endpoints. We will discuss the Internet Backplane Protocol, an experimental end-to-end storage service that is being deployed internationally and compare it to other network storage approaches. Finally we will discuss a unified approach to networked information resources that is based in the principles of the end-to-end approach.

3. NETWORK STORAGE ASSUMPTIONS

3.1 Availability

In any network setting, availability of stored data is contingent on both the availability of the storage system(s) upon which it is stored and the connectivity to those systems. On simple attached disk systems, both of these factors are of high enough reliability that users typically deal with issue of availability only in an ad hoc fashion – they make periodic backups. RAID systems lower the probability of disk failures to another degree, but do nothing to address the issue of communication failures. This is because catastrophic communication failures typically do not happen between a disk and the processor to which it is attached.

Storage Area Networks (SANs) have embraced the model of the directly-attached disk, inheriting the assumption that connectivity will not fail unless the entire system fails catastrophically. This assumption clearly invites questions about the viability of SANs in the wide area.

In a scalable Wide Area Network (WAN), storage resources can be intermittently unavailable (or available only with inadequate quality of service) due to a number of conditions in the network, including traffic congestion, routing problems, topology changes and malicious interference. These conditions are resolved in time frames ranging from less than a second to longer than days. As such, a variety of end-to-end strategies should exist for ensuring availability. These range from simple retry, to redundant data accesses spread across the network (augmented with RAID-like error correction to reduce data redundancy), to maintaining high-latency archival backups. Obviously, these algorithms must be implemented at the end-points, both in order to ensure delivery to the end-point and to achieve the necessary level of sensitivity to the requirements of the end-point operating systems, applications or users. We believe the design of Venti [13], where storage blocks are indexed by 160-bit hashes of their data, may be the right approach to unifying the various end-to-end strategies for ensuring availability.

3.2 Correctness

Assumptions about the accuracy of storage systems have changed over time. At one time, tape was considered unreliable due to environmental factors, while fixed disk was considered reliable. Therefore, data stored on tape was verified using checksums while data on disk was not. This assumption underlies the simplifying assumption in classical RAID systems, that disk storage fails only through corruption of entire sectors, which is easily detected by the hardware controller. However, the advent of cheap, mass-produced disks for personal computers has given rise to disk subsystems that have a wider variety of complex failure modes, including the undetected delivery of incorrect data [Quantum, personal communication 2001].

In a SAN, the assumption is that RAID storage systems have sufficient internal checking to provide essentially perfect accuracy and the storage network itself uses highly reliable protocols. The fact that SAN is deployed in highly controlled environments leads to the assumption that the composition of reliable components will be reliable. Whether or not this is a reasonable assumption in the SAN, an end-to-end approach to the WAN requires that data accuracy be checked by the end-systems, meaning the ultimate writers and readers of the data. This yields protection not only against errors in the composition of reliable disk and network components, but also against unreliable or malicious components that might be introduced undetectably in the Wide Area Network. Once again, the design of Venti may prove beneficial here, since the handle to data is a checksum of the data itself.

3.3 Security

Directly attached storage and physically localized SAN solutions have an assumed level of physical security that can be breached only by highly intrusive means (although in the case of SANs, eavesdropping is easier to imagine). However, once SAN is extended to campus or metropolitan area networks, and certainly if it is tunneled across the WAN, this assumption of security is lost. In IP networking, there is no assumption that intermediate nodes are trusted and so security must be implemented through end-to-end application of cryptographic techniques.

The element of security that cannot be adequately handled by use of end-to-end techniques alone is Denial of Service (DoS), and for this reason the security mechanisms implemented in the network must be used to control the right to allocate storage. However, it must be understood that these security mechanisms are best effort, and cannot be relied upon; the endpoint must be prepared to be affected by Denial of Service (DoS) attacks that make allocation at particular locations with the WAN impossible or to have stored data overwritten or corrupted due to breaches in security at intermediate nodes. Of these concerns, techniques for preventing or detecting DoS attacks are the only ones that cannot be addressed in a strictly end-to-end manner; techniques for handling DoS in IP networks hold promise in the analogous storage scenario.

3.4 Unbounded Size

While every data management system has capacity limits, they are often assumed to be large enough to be ignored by applications and to be manageable, through administrative mechanisms such as rearrangement of file systems on multiple disk volumes, in the time-frame of months or even years. The assumption that an application can make unbounded allocations thus rests on assumptions that do not hold in the WAN: that the use of the storage system is limited to a single administrative domain, and that it is within the power of the administrators of that domain to control provisioning as required.

In the WAN, the application must assume that any particular storage resource may be used by other administrative domains and so may not be able to fulfill a given request for storage resources. What this means is that the end-system must take responsibility for anticipating or arranging for the availability of resources, potentially distributed across the network in order to accommodate a request of a particular size.

3.5 Unbounded Duration

Persistence, taken to mean unbounded duration, is often taken to be the defining characteristic of storage. However, the implication of assuming unbounded duration for every allocation of storage resources is that it is impossible for a storage resource manager to make an allocation for any user other than those for whom that manager is willing to commit those storage resources indefinitely. Thus, for applications where allocations can be useful even if they only last a limited time, and for which it is valuable to be able to make allocations for unknown users, the assumption of unbounded allocation makes the sharing of storage impossible. The result is that in the context of public networking, end-to-end delivery of datagrams is the only allowable instance of resource sharing.

Thus, bounds on the duration of allocation are necessary in order to allow scalable sharing of storage resources, but it only enables applications in which allocations of limited duration can be used. Since unbounded duration is the normal assumption in applications that make use of storage, it is worth asking whether limited-duration storage allocations are even useful.

We have two answers to this question:

1. There are many natural uses of storage of limited duration. Two simple examples are checkpointing and caching. In checkpointing, a snapshot of a computation state is stored, so that the computation may be restored from the snapshot in the event of a failure. Typically, once a new checkpoint has been taken, the old one may be discarded. Since checkpoints are often taken at fixed intervals, checkpoint files are inherently of limited duration. In content delivery, data is often cached near anticipated receivers for performance enhancement. However, if the cached data is not there, the data may be retrieved from a faraway source. Thus, limited-duration storage fits caching applications naturally.
2. All physical storage is inherently limited in duration, albeit on the scale of years. The assumption of permanence requires a correlative assumption of active management by administrative means. The problem with such active management is that it interferes with the predictable and correct functioning of the storage system, often requiring periods of unavailability. The disruptions of service caused by changes in generation of media can be severe enough to require that end-users be notified and take account of them.

When data must be managed over periods of time that are longer than can be arranged in a single allocation (either because the resource manager is unwilling to grant a long enough allocation to the user in question, or because the physical medium cannot support an allocation for a sufficiently long period of time), end-to-end principles must be used.

A common objection to the applying the end-to-end approach to the duration of storage allocations is that it requires active management of storage by the end-point. It is commonly viewed as a positive aspect of unbounded storage allocations that a client can fail, go off-line, or for other reasons fail to interact with a storage allocation, but it will continue to hold their data until they return. Storage with bounded allocation cannot be used for the maintenance of state in case of failure of the end-point for unbounded periods of time. However, if there is a bound on the longest failure, then the duration of allocation can be adjusted to allow for it. In the case where the end-point can fail indefinitely, it will be necessary to employ storage that can be permanently allocated, and this will generally require that the end-user is known to the storage resource manager. We do not rule out unbounded allocations in these cases, but point out that the solution does not scale, which is fine: typically one uses directly connected or closely coupled storage for such core state.

4. IBP AND THE EXNODE

4.1 The Network Layer of the Storage Stack

In the context of Storage Networking, "IP/Storage integration" means putting IP networking into the interconnection fabric (i.e. into the data transmission substrate) that underlies the storage pool. For Logistical Networking, on the other hand, IP/Storage integration means *putting storage into the network infrastructure itself*, creating a *shared resource fabric* that exposes storage resources for general use in the same way that the Internet now exposes transmission bandwidth for shared use. To create a resource fabric of this kind that can also scale, we set out to define a new *storage stack* using a bottom-up and layered design approach that adheres to the same end-to-end principles that have guided Internet engineering for two decades [17]. According to this philosophy, the key to achieving flexibility and scalability lies in defining the right basic abstraction of the physical resource to be shared at the lowest levels of the stack. For Logistical Networking the *Internet Backplane Protocol (IBP)* plays this role.

IBP is the lowest layer of the storage stack that is globally accessible from the network (Figure 1). To provide an ideal resource fabric for Logistical Networking, it must supply an abstraction of *access layer resources* (i.e. file or block storage services at the local level) that has "network transparency"[16]. This means it must satisfy the following two requirements:

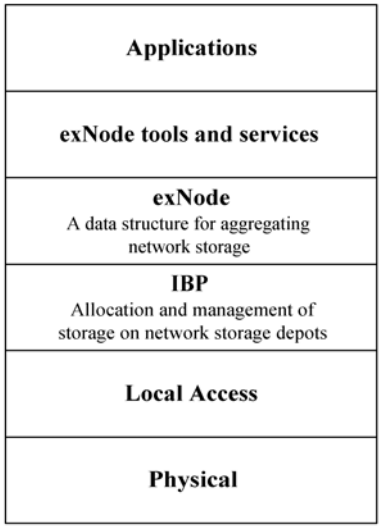

Figure 1: The network storage stack for Logistical Networking.

Expose underlying storage resources in order to maximize freedom at higher levels — The abstraction should create a mechanism that implements only the most indispensable and common functions necessary to make the storage usable *per se*, leaving it otherwise as primitive as it can be; all stronger functions must be built on top of this primitive layer. The goal of providing essential functionality while keeping the semantics of this layer as weak as possible is to expose the underlying resources to the broadest range of purposes at higher layers, and thereby foster ubiquitous deployment and free developers to innovate.

Enable scalable Internet-style resource sharing — The abstraction must mask enough of the peculiarities of the access layer resource (e.g. fixed block size, differing failure modes, and local addressing schemes) to enable lightweight allocations of those resources to be made by any participant in the network for their limited use and regardless of who owns them.

To implement this strategy we followed the IP paradigm and modeled the design of IBP on the design of IP datagram delivery. IP datagram service is based on packet delivery at the link level, but with more powerful and abstract features that allow it to scale globally. Its leading feature is the independence of IP datagrams from the attributes of the particular link layer, which is established as follows:

- Aggregation of link layer packets masks its limits on packet size;
- Fault detection with a single, simple failure model (faulty datagrams are dropped) masks the variety of different failure modes;
- Global addressing masks the difference between local area network addressing schemes and masks the local network's reconfiguration.

This higher level of abstraction allows a uniform IP model to be applied to network resources globally, which is crucial to creating the most important difference between link layer packet delivery and IP datagram service: *any participant in a routed IP network can make use of any link layer connection in the network regardless of who owns it.* Routers aggregate individual link layer connections to create a global communication service. This IP-based aggregation of locally provisioned, link layer resources for the common purpose of universal connectivity constitutes the form of sharing that has made the Internet the foundation for a global information infrastructure.

IBP is designed to enable the scalable, relatively unbrokered sharing of storage resources within a community in much the same manner. Just as IP is a more abstract service based on link-layer datagram delivery, IBP is a more abstract service based on blocks of data (on disk, tape or other media) that are managed as "byte arrays." The independence of IBP byte arrays from the attributes of the particular access layer (which is our term for storage service at the local level) is established as follows:

- Aggregation of access layer blocks masks the fixed block size;
- Fault detection with a very simple failure model (faulty byte arrays are discarded) masks the variety of different failure modes;

- Global addressing based on global IP addresses masks the difference between access layer addressing schemes.

This higher level byte array abstraction allows a uniform IBP model to be applied to storage resources globally, which is essential to creating the most important difference between access layer block storage and IBP byte array service: *Any participant in an IBP network can make use of any access layer storage resource in the network regardless of who owns it.* The use of IP networking to access IBP storage resources creates a global storage service.

Whatever the strengths of this application of the IP paradigm, however, it leads directly to two problems. First, in the case of storage, the chronic vulnerability of IP networks to Denial of Service (DoS) attacks is greatly amplified. The free sharing of communication within a routed IP network leaves every local network open to being overwhelmed by traffic from the wide area network, and consequently open to the unfortunate possibility of DoS from the network. While DoS attacks in the Internet can be detected and corrected, they cannot be effectively avoided. Yet this problem is not debilitating for two reasons: on the one hand, each datagram sent over a link uses only a tiny portion of the capacity of that link, so that DoS attacks require constant sending from multiple sources; on the other hand, monopolizing remote communication resources cannot profit the attacker in any way, it can only harm the victim. Unfortunately neither of these factors hold true for access layer storage resources. Once a data block is written to a storage medium, it occupies that portion of the medium until it is deallocated, so no constant sending is required. Moreover it is clear that monopolizing remote storage resources can be very profitable for an attacker and his applications.

The second problem with sharing storage network-style is that the classic definition of a storage service is based on processor-attached storage, so it includes strong semantics (near-perfect reliability and availability) that are difficult to implement in the wide area network. Even with Storage Networking technologies, which are used in "storage area" or local area networks, these strong semantics can be difficult to implement and are a common cause of error conditions. When extended to the wide area, it has so far proved impossible to support such strong guarantees for storage access, but then problems with strong service semantics in the wide area are not unique to storage systems [20]. Whether or not integrated IP and Storage Networking technologies can make progress on this front remains to be seen. Logistical Networking takes a different approach.

We address both of these issues through special characteristics of the way IBP allocates storage:

- Allocations of storage in IBP can be time limited. When the lease on an allocation expires, the storage resource can be reused and all data structures associated with it can be deleted. An IBP allocation can be refused by a storage resource in response to over-allocation, much as routers can drop packets, and such "admission decisions" can be based on both size and duration. Forcing time limits puts transience into storage allocation, giving it some of the fluidity of datagram delivery.
- The semantics of IBP storage allocation are weaker than the typical storage service. Chosen to model storage accessed over the network, it is assumed that an IBP storage resource can be transiently unavailable. Since the user of remote storage resources is depending on so many uncontrolled remote variables, it may be necessary to assume that storage can be permanently lost. Thus, IBP is a "best effort" storage service. To encourage the sharing of idle resources, IBP even supports "soft" storage allocation semantics, where allocated storage can be revoked at any time. In all cases such weak semantics mean that the level of service must be characterized statistically.

IBP storage resources are managed by "depots," which are servers on which clients perform remote storage operations. As shown in the Table 1 below, the IBP client calls fall into three different groups:

Table 1: IBP API calls

Storage Management	Data Transfer	Depot Management
`IBP_allocate`, `IBP_manage`	`IBP_store`, `IBP_load` `IBP_copy`, `IBP_mcopy`	`IBP_status`

The **`IBP_allocate`** function is the most important element. **`IBP_allocate`** is used to allocate a byte array at an IBP depot, specifying the size, duration (permanent or time limited) and other attributes. A chief design feature is the use of capabilities (cryptographically secure passwords) [7]. A successful **`IBP_allocate`** call returns a set of three capabilities: one for reading, one for writing, and one for management of the allocated byte array. A more detailed account of the API and its other functions is available [12] online at (http://loci.cs.utk.edu/ibp/documents/). A description of the status of the current software that implements the IBP client, servers, and protocol is available at (http://loci.cs.utk.edu/ibp/software).

4.2 A Data Structure for the Flexible Aggregation of Network Storage

From the point of view of the Storage Networking community, it is likely that one of the most striking (not to say shocking) features of the Logistical Networking storage stack is the way it appears to simply jettison the well known methods of usage for local area storage, viz. files systems, databases, and VM mapping. These familiar abstractions can be supported in the logistical paradigm, but that support must conform to its "exposed-resource" design principles. According to these principles, implementing abstractions with strong properties — reliability, fast access, unbounded allocation, unbounded duration, etc.— involves creating a construct at a higher layer that *aggregates* more primitive IBP byte-arrays below it, where these byte arrays are often distributed at multiple locations. For example, caching requires that data be held in a home site, but temporary copies be made at various remote sites. Similarly, replication requires that multiple copies of data exist in various locations for purposes of performance and fault-tolerance. More advanced logistical applications require that data be explicitly routed through the network, and thus may have many "homes" throughout their lifetime.

To apply the principle of aggregation to exposed storage services, however, it is necessary *to maintain state that represents such an aggregation of storage allocations*, just as sequence numbers and timers are maintained to keep track of the state of a TCP session. Fortunately there is a traditional, well-understood model to follow in representing the state of aggregate storage allocations. In the Unix file system, the data structure used to implement aggregation of underlying disk blocks is the *inode* (*intermediate node*). Under Unix, a file is implemented as a tree of disk blocks with data blocks at the leaves. The intermediate nodes of this tree are the inodes, which are themselves stored on disk. The Unix inode implements only the aggregation of disk blocks within a single disk volume to create large files; other strong properties are sometimes implemented through aggregation at a *lower level* [6] or through modifications to the file system or additional software layers that make redundant allocations and maintain additional state [10, 21].

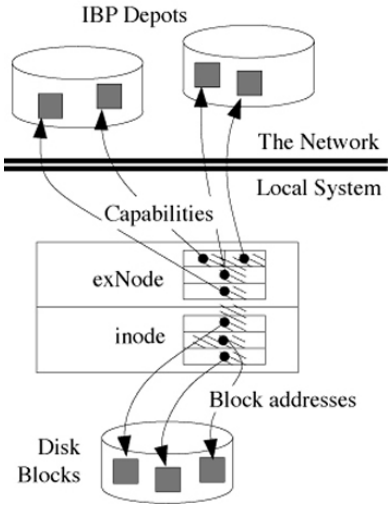

Figure 2: The exNode compared to a Unix inode.

Following the example of the inode, we have chosen to implement a single generalized data structure, which we call an *external node*, or *exNode*, to manage aggregate allocations that can be used in implementing network storage with many different strong semantic properties [3]. Rather than aggregating blocks on a single disk volume, the exNode aggregates byte arrays in IBP depots to form something like a file, with the byte arrays acting as disk blocks. We say "something like a file" because when an exNode uses IBP storage allocations, the time-limited or soft nature of those allocations gives it a transient quality that files normally should not have. Two major differences between exNodes and inodes are that the IBP buffers may be of any size, and the extents may overlap and be replicated. But the key point about the design of the exNode is that it has allowed us to create storage abstractions with stronger properties, such as a network file, which can be layered over IBP-based storage in a way that is completely consistent with the exposed resource approach.

The exNode can used to implement *replication for fault-tolerance*, storing files in multiple locations so that the act of downloading may succeed even if many of the copies are unavailable; by breaking the file up into blocks and storing error correcting blocks calculated from the original blocks (based on parity as in RAID systems [6]or on Reed-Solomon coding [11], downloads can be robust to even more complex failure scenarios.

4.3 End-to-End Services for Storage

While the exNode defines a framework for aggregation of IBP capabilities, it also provides a framework for the addition of metadata that describes the encoding of data in the file or byte array, enabling a variety of end-to-end services. These services range from the low level, such as the implementation of redundancy through error correcting codes, to higher level services such as the framing of data into TCP-like segments and the insertion of checksums on a per-segment basis, or application of strong end-to-end authentication and encryption of data. These higher-level services allow these end-to-end services to inform the process of, for instance, obtaining a correct copy of the datain the face of temporary unavailability or permanent loss of data, whatever the cause. In this way, the exNode is analogous to the state of a TCP connection, and the data stored on disk is an analogous to a TCP stream.

In fact, the alternations in the resemblance of the exNode, between a file descriptor on the one hand, and the state of a network connection on the other, depends on how much control information is encoded in the exNode rather than in the data stream. If control information is mostly in the exNode itself, then the use of the exNode is most like a file descriptor; but if the exNode merely specifies how control information is encoded in the data stream, then it is like the state of a network connection. Our implementation of the exNode allows the flexibility for either style of use..

5. OTHER APPROACHES TO SCALALBLE NETWORK STORAGE

5.1 FTP mirroring, Web Caching and Content Distribution

From the earliest days of the Internet, the traffic and load caused by downloading of popular content from FTP (and later HTTP) servers led to the widespread use of manual mirroring, which uses redundant storage at multiple locations in the network topology to localize traffic, spread server load and accelerate downloads. Because storage was a relatively scarce resource, policy was required to determine which content would be mirrored at each site, how often updates would be made to ensure consistency and when a mirror would be removed. Choice of mirror sites was a manual process, requiring an evaluation of the competence and reliability of the server by the end user. Web caching automated this process and made policy a simple function of content size and popularity [14]. However, when caching is implemented without the cooperation of the server there is still a question of consistency that sometimes requires user intervention to force a download from the origin server. Content Distribution takes the process a step further, deploying caches that are extensions of the origin server [19]. However, all of this elaborate, special purpose storage infrastructure is available only to specific application protocols (FTP, HTTP, streaming media) and sometimes (e.g. Content Distribution) only to paying customers.

5.2 Peer-to-Peer Storage

Peer-to-peer systems such as Napster [2] and Gnutella [1], and experimental approaches, such as Chord [18] and CANs [15] use shared storage resources provisioned at end-points to implement distributed content delivery. In such systems, allocations are made by the owner of the end-point and are maintained and made available at their discretion. A broad community of peers who have no control over availability or duration uses these allocations. The result is a system in which access to any one allocation is unreliable but adequate reliability is gained through the aggregation of a large number of replicas of

popular content. The application, free recreational use of rich media content, is also one that can withstand low reliability.

The fact that all storage allocations are made at end-points means that a large amount of data transfer between end-points is required in these peer-to-peer architectures, causing a flood of traffic. The use of servers or caches owned by the content distribution system is ruled out due by the need to decentralize control in peer-to-peer systems (either in an attempt to evade responsibility for redistribution of copyrighted material or because of a desire to build scalable ad-hoc communities without central administration). However, if storage can be shared within a community in a scalable, ad-hoc manner, and without assigning responsibility for the stored data to the operator of the resource, then peer-to-peer architectures can use such storage to optimize their performance and efficiency without compromising their goals.

5.3 Multipath Connectivity

The model of storage connectivity defined by Storage Area Networking (SAN) makes assumptions about the reliability of the network that simply do not scale to the global Internet. In order to leverage the enormous investment made in SAN, major storage vendors are seeking to deploy SAN tunneled over IP in networks that are over provisioned with capacity and implement multiple paths between endpoints to the degree that their strong assumptions can be maintained [D. Black (EMC), personal communication, 2002]. The result is equivalent to building a private IP network for the purpose of connecting storage systems. While this approach may work in restricted cases, it sacrifices the scalability of the global Internet.

5.4 Interplanetary Networking

A fundamental assumption of networking based on datagram delivery is that data links are sufficiently available to allow end-to-end delivery paths to be present. In cases where connectivity is intermittent, as is often the case when satellite links are used, an end-to-end path must include storage of data at intermediate nodes. Terrestrial examples of intermittent connectivity are often thought to be uninteresting, as they tend to occur in situations where telecommunications infrastructure is inadequate, and the dogma of the day says that the natural state of man is to be connected and that all of humanity will eventually be. However, when man takes to space, and communication requires the use of satellite relays, then even Buck Rodgers will have a problem with end-to-end delivery of datagrams. The answer to this problem, as formulated by Vint Cerf and the IRTF Interplanetary Networking Working Group [5] is to use storage-and-forward techniques to relay data asynchronously. End-to-end paths then consist of file delivery in the style of e-mail. However, this group has not considered making the storage resource available for sharing by other interplanetary applications that might require management of state [V. Cerf (WorldCom), personal communication, 2002].

6. CONCLUSION

The end-to-end argument was formulated in the context of IP network architecture, which has as its defining service the end-to-end delivery of datagrams from sender to receiver. However, as its progenitors knew, the argument applies more generally to scalable information systems and so, as we have demonstrated in this paper, a globally scalable service with the sharing of storage can be designed according to end-to-end principles. The resulting architecture provides a model for extending storage networking to the wide area and may have a profound impact on the development of that niche of storage technology. It is our belief that it also provides a model of scalable data networking that offers many advantages over end-to-end delivery of datagrams while still adhering to end-to-end principles.

Data cannot be communicated within an asynchronous system (one in which sender and receiver do not share a clock) without using buffers. Thus, the delivery of datagrams, which models the networks as if they were a single wires with no intermediate storage, inherently hides from users the fact that every physical layer connection involves send and receive buffers. This virtualization is achieved by implementing these buffers in fast memory, managing each one using a FIFO discipline and adhering to fair queuing. While this simplification of the network has obviously been sufficient to provide services adequate to drive global dominance of IP networking, it is interesting to note that Quality of Service, the most widely proposed modification of IP datagram routing, is a modification of fair queuing that takes greater advantage of the power of the fact that router receive buffers can be used to delay datagrams differentially and thus implement policy.

Logistical Networking is a model of networking that exposes the fact that data is buffered and allowed that fact to be used to implement novel communication strategies. While data flows between IBP depots using standard IP communication that hides fast FIFO buffers, the depots themselves implement buffers that can range from small and fast to huge and slow depending on the medium and architecture of the depot. Far from being determined by the depot itself, the regimen for using IBP buffers is specified by the end-user or an active management element outside of the depot. In this sense, the buffers implemented by the IBP depot are dumber and more passive than the send and receive buffers implemented in IP routers and end-systems.

Today, the IP network is beset by the introduction of active middleboxes that complicate and in some cases violate the semantics of datagram delivery. Overlay networks and systems of proxies introduce storage and buffer management policy that are balkanized with protocols specific to each application domain, and so miss out on the commonality of services and sharing of resources that IP has made possible in datagram delivery. In most of these cases, the designers are oblivious to end-to-end considerations or consciously abandon them in the mistaken belief that this orthodox route to scalability does not apply to systems that explicitly manage state.

It is our belief that the widespread deployment of storage Internetworking on the model of IBP will fundamentally increase the power and flexibility of the network, and will affect the way that middleware and application developers think about distributed system architecture and scalable communication. While we are working most directly with researchers in the areas of Grid computing, distributed visualization and multimedia content delivery, we forsee even greater potential in areas which are currently too reliant on management of state to be spread across the network. The direction is towards a blurring of the artificial divide between the processor bus, system local and wide area networks, between flexible resource scheduling and scalable services. In this vision, network architecture evolves into a radically decentralized form of computer architecture, and the world's information systems co-manage shared data transport, storage and computation as a unified fabric of shared resources.

7. ACKNOWLEDGEMENTS

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. The infrastructure used in this work was supported by the NSF Computer and Information Science and Engineering Research Infrastructure program, EIA9972889.

The authors would like to acknowledge the early contributions of several supporters and collaborators. Work on Logistical Networking was an outgrowth of the I2-DSI project, undertaken through the Internet2 project and with the particular early support of Guy Almes and Ann O'Beay. Jack Dongarra supported this work within his Innovative Computing Laboratory even before independent funding was obtained. Discussions regarding the scalability in the RCDS [9] and Snipe [8] projects of Keith Moore and Graham Fagg influenced the development of I2-DSI [4] and the formulation of IBP. Although he was a student, Martin Swany was also a teacher of network architecture and and contributed to important early design work. Finally, Rich Wolski made significant contributions to the formulation of the IBP API and to the authors' understanding of the subtleties of network topology sensing.

8. REFERENCES

[1] Gnutella, 2001. http://gnutella.wego.com/

[2] Napster, 2001. http://www.napster.com

[3] Bassi, A., Beck, M. and Moore, T., Mobile Management of Network Files. in *Third Annual International Workshop on Active Middleware Services*, (San Franscisco, 2001).

[4] Beck, M. and Moore, T. The Internet2 Distributed Storage Infrastructure Project: An Architecture for Internet Content Channels. *Computer Networking and ISDN Systems*, *30* (22-23). 2141-2148

[5] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Travis, E. and Weiss, H., Interplanetary Internet (IPN): Architectural Definition, IETF Internet Draft, 2001. http://www.ietf.org/internet-drafts/draft-irtf-ipnrg-arch-00.txt

[6] Chen, P.M., Lee, E.K., Gibson, G.A., Katz, R.H. and Patterson, D.A. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, *26*. 145-185

[7] Dennis, J. and Horn, E.V. Programming semantics for multiprogrammed computations. *Communications of the ACM*, *9* (3). 143-155

[8] Fagg, G.E., Moore, K., Dongarra, J.J. and Geist, A., Scalable Network Information Processing Environment (SNIPE). in *Proceedings of SuperComputing '97*, (San Jose, CA, USA, 1997).

[9] Moore, K., Browne, S., Cox, J. and Gettler, J., Resource Cataloging and Distribution Systemb, University of Tennessee, UT-CS-97-346. http://www.netlib.org/utk/projects/rcds/rcds-tr/main.html.

[10] Morris, J.H., Satyanarayan, M., Conner, M.H., Howard, J.H., Rosenthal, D.S.H. and Smith, F.D. Andrew: A Distributed Personal Computing Environment. *Communications of the ACM*, *29* (3). 184-201.March

[11] Plank, J.S. A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems. *Software -- Practice and Experience*, *27* (9). 995-1012.September

[12] Plank, J.S., Bassi, A., Beck, M., Moore, T., Swany, M. and Wolski, R. Managing Data Storage in the Network. *IEEE Internet Computing*, *5* (5). 50-58.September/October

[13] Quinlan, S. and Dorward, S., Venti: a new approach to archival storage. in *USENIX File and Storage Technologies (FAST) 2002*, (Monterey, CA, 2002).

[14] Rabinovich, M. and Spatscheck, O., *Web Caching and Replication*. Addison-Wesley, 2001.

[15] Ratnasamy, S., Francis, P., Handley, M. and Karp, R., A Scalable Content-Addressable Network. in *ACM SIGCOMM*, (2000).

[16] Reed, D.P., Saltzer, J.H. and Clark, D.D. Comment on Active Networking and End-to-End Arguments. *IEEE Network*, *12* (3). 69-71.May/June

[17] 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

[18] Stoica, I., Morris, R., Karger, D., Kaashoek, F. and (MIT), H.B., Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. in *ACM SIGCOMM*, (2001).

[19] Verma, D.C., *Content Distribution Networks: An Engineering Approach*. Wiley, 2001.

[20] Waldo, J., Wyant, G., Wollrath, A. and Kendall, S., A note on distributed computing, Sun Microsystems, TR-94-29. November.

[21] Watson, R.W. and Coyne, R.A., The Parallel I/O Architecture of the High-Performance Storage System (HPSS). in *IEEE Mass Storage Systems Symposium*, (1995), IEEE Computer Society Press.