Transnet Architecture and Logistical Networking for Distributed Storage

Micah Beck Ying Ding Terry Moore James S. Plank

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

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

Abstract

This paper addresses the possibility that IP, in the role of the common service, is not as general as is needed in order to directly address application requirements, including scalable storage services, that go beyond simple datagram delivery. We propose to generalize the view of layer 2 to include local storage and processing services, and to situate a new common protocol between the local and network layers in the current network stack. Since a protocol at this layer would provide an abstraction of services for data that is *in transit* at the intermediate node, we propose to call it the *transit* layer Our view is that a transit layer protocol that abstracts the particular characteristics of different technologies at the local layer, while being more general and sitting below the network layer in the stack, would exhibit greater deployment scalability and provide a broader foundation for network interoperability than IP. We present an account of scalable storage middleware developed under the auspices of the Logistical Networking project as an overlay implementation of transit layer functionality over the current Internet

1. Introduction

Our work addresses the possibility that IP, in the role of the common service, is not as general as is needed in order to directly address application requirements beyond simple datagram delivery. Consider, then, the primitive services — transfer of data between neighboring nodes, storage of data locally, node, and processing of data locally — that network intermediate nodes, as key elements of the shared infrastructure, could make available for the creation of other network services. Clearly these three fundamental services are at an architectural level that is below the network layer of the current stack, since they are required in order to implement end-to-end IP datagram delivery. IP service is specialized in the sense it exposes only data transfer, encapsulating the other two fundamental services in order to implement optimized packet forwarding. However, if we look at services applications want, but which IP does not model, chief among them are storage and processing of data in transit, i.e. at intermediate nodes.

Unlike IP datagram delivery, storage and processing resources are inherently local to the intermediate node, rather than being defined across a homogeneous network layer. In that respect, they are more analogous to the link layer, which connects adjacent nodes. Thus, as an initial step, we propose to generalize the view of layer 2 to include local storage and processing services. We call this more general layer, which includes link, storage and processing as coequal elements, the local layer (Figure 1).

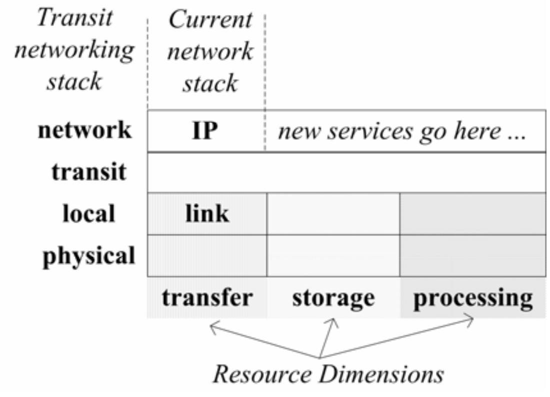

Figure 1: The transit networking stack compared to the traditional stack

The local layer exhibits the same extreme heterogeneity as the link layer in each of its three

This work is supported by the National Science Foundation (NSF) 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 NSF Internet Technologies Program under grant # ANI-9980203. The infrastructure used in this work was supported by the NSF CISE Research Infrastructure program, EIA-9972889 and Research Resources program EIA-0224441.

elements or dimensions. Storage media can be as different as fast access RAM and large disk arrays. Processing elements can be implemented on commodity microprocessors or large field programmable gate arrays. The key point to notice, however, is that all of the operations of the local layer can be modeled as providing services of various kinds to arrays of bytes of data stored in transit at the intermediate node.

For this reason, a network protocol capable of expressing a broad class of operations on byte arrays located at the intermediate node can abstract a more general class of lower layer services than IP does. By choosing to model operations that are either local to the intermediate node or restricted to operating on nodes adjacent at the link layer, the protocol can avoid implementing routing or wide area algorithms of any kind. Because network layer protocols can in fact be implemented on top of it, we would situate the new protocol between the local and network layers in the current network stack. Since a protocol at this layer would provide an abstraction of services for data that is in transit at the intermediate node, we propose to call it the transit layer (see Figure 1).

Our view is that a transit layer protocol that abstracts the particular characteristics of different technologies at the local layer, while being more general and sitting below the network layer in the stack, would exhibit greater deployment scalability and provide a broader foundation for network interoperability than IP. We call the architectural approach based on such a protocol, and on the interpretation of the end-to-end paradigm that should inform its progressive development [6, 10], the *Transnet architecture* [4].

Of course it is one thing to argue, in a somewhat a priori way, for Transnet architecture as a new architectural paradigm, but something quite different to show that that paradigm can actually deliver on its promises, especially with regard to the difficult challenge of creating a network compute service that can both scale and perform. Fortunately the ideas for the transit networking and a transit layer protocol were not developed in an empirical vacuum. They were developed in the context of real world design and experimentation on a network storage service intended to be scale globally — logistical networking. In fact, logistical networking technology, and the Internet Backplane Protocol at its core, amount to an implementation of transit networking functionality overlaid on the current IP model. Since logistical networking already has a substantial software base and a globally deployed testbed, we believe it provides an ideal vehicle for exploring the potential for programmable networking within transit networking paradigm. Below we describe logistical networking in more detail and show how it can be extended to create a scalable computing service that adheres to the end-to-end principle, but can none the less achieve satisfactory performance.

2. Building on Logistical Networking

As described above, transit networking is an architectural approach whose purpose is to provide a foundation for interoperable resource sharing that can scale beyond the limitations of the current IP paradigm, especially in terms of offering applications access to storage and processing resources at network intermediate nodes. But the question of whether or not transit networking architecture can deliver on this goal, and in particular, whether or not it can support programmable network services of value whose deployment can scale, cannot be answered with out a software implementation to support experimentation and testing. Fortunately, our previous work in logistical networking (LN) yielded a design and implementation of transit layer functionality for network storage that can be extended to fill this role today. But in order to evaluate the significance for transit networking of work done on our logistical networking platform, it is important to understand the way in which former builds directly on our previous work in the latter.

In a general sense, the relationship between these two concepts: transit networking is an architectural approach that aims to maximize scalable interoperability; logistical networking represents one way of implementing this architecture. Historically, however, the order is reversed: the effort to develop LN provided the concrete research context for uncovering and refining the ideas that are now embodied in the transit networking stack. LN is an effort to integrate shared storage infrastructure into the communication fabric in order to address the mounting “logistical” problems associated with application areas like content distribution and distributed data intensive scientific applications. To achieve the kind of global deployment scalability this challenge required, we created a highly generic, best effort storage service, called the Internet Backplane Protocol (IBP), shaping its design by analogy with the design of IP in an effort to produce a common storage service with similar characteristics. Though IBP, and the higher software layers necessary to use it, have been implemented as an overlay on TCP/IP, the analysis of the design of that “storage stack,” and the experience acquired working with it, has led directly to the development of the transit networking stack as a more pure expression of the ideas involved. The extremely close fit between requirements of transit networking architecture and the characteristics of logistical networking middleware is therefore no accident.

The relation between the transit networking stack and the overlay implementation in LN software is displayed in Figure 2. IBP, which implements the transit layer functionality of a common storage service, is the foundational layer of the logistical networking stack. As noted, its design is modeled on the design of IP datagram delivery. Just as IP is a more abstract service based on link-layer datagram delivery, so IBP is a more abstract service based on blocks of data (on disk, memory, tape or other media) that are managed as "byte arrays." By masking the details of the local disk storage — fixed block size, different failure modes, local addressing schemes — this byte array abstraction allows a uniform IBP model to be applied to storage resources generally. The use of IP networking to access IBP storage resources creates a globally accessible storage service.

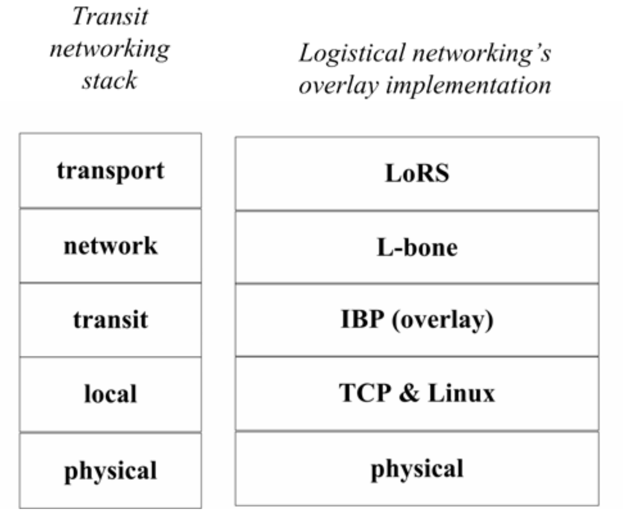

Figure 2: Comparison of the transit networking architecture and it's overlay implementation

Since computation is inherently more complex than storage in various ways, it is natural to assume that including processing in the LN model will prove to be inherently more challenging than LN with storage alone. While we do not discuss the computational aspect of the Transnet architecture in detail here, there are two good reasons to think that this discussion of distributed storage in LN provides a useful preview of how such a computational extension of LN might work.

The first reason is that it is easier to see the outlines of the basic idea Transnet architecture in the context of network storage than it is in the context of network computation. Although the same end-to-end paradigm is being applied in both cases, the relative simplicity of storage, and its obvious parallels with the case of bandwidth, show more clearly how the model can be extended to a new resource.

Second, and more importantly, the work on LN processor services builds directly on the storage service that LN supplies, and this provides the logistical approach to computation in the network with a tremendous advantage that previous efforts in this general area did not possess: a scalable, general purpose solution to the problem of managing the state of distributed applications. The absence of such an infrastructure for interoperable state management is well known to represent a major impediment to the creation of advanced distributed applications that can actually be deployed. Since LN technology provides a generic storage service, which is exposed for application scheduling and is scalable to the wide area, it lays the foundation for solving this problem and thereby opens up opportunities for new lines of attack on the problem of scalable network computing.

3. The Internet Backplane Protocol

As the name suggests, the goal of logistical networking is to bring data transmission and storage within one framework, much as military or industrial logistics treat transportation lines and storage depots as coordinate elements of one infrastructure. Achieving this goal requires a form of network storage that is globally scalable, meaning that the storage systems attached to the global data transmission network can be accessed and utilized from arbitrary endpoints. To create a shared infrastructure that exposes network storage for general use in this way, we set out to define a new "storage stack", analogous to the Internet stack, using a bottom-up and layered design approach that adheres to the end-to-end principles [5]. We discuss this design model in more detail below, but the important thing to note here is that the key to attaining scalability using this model lies in defining the right basic abstraction of the physical resource to be shared at a low level of the stack. In the case of storage the Internet Backplane Protocol (IBP) plays this role.

IBP is the lowest layer of the storage stack that is globally accessible from the network. Its design is modeled on the design of IP datagram delivery. Just as IP is a more abstract service based on link-layer datagram delivery, so IBP is a more abstract service based on blocks of data (on disk, memory, tape or other media) that are managed as "byte arrays." By masking the details of the storage at the local level — fixed block size, differing failure modes, local addressing schemes — this byte array abstraction allows a uniform IBP model to be applied to storage resources generally. The use of IP networking to access IBP storage resources creates a globally accessible storage service.

As the case of IP shows, however, in order to scale globally the service guarantees that IBP offers must be weakened, i.e. it must present a "best effort" storage service. First and foremost, this means that, by default, IBP storage allocations are time limited. When the lease on an IBP allocation expires, the storage resource can be reused and all data structures associated with it can be deleted. Additionally an IBP allocation can be refused by a storage resource in response to over-allocation, much as routers can drop packets; 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; more importantly, it makes network storage far more sharable, and easier to scale.

The semantics of IBP storage allocation also assume that an IBP storage resource can be transiently unavailable. Since the user of remote storage resources depends 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. IBP client calls fall into three different groups: IBP_allocate and IBP_manage for storage management; IBP_store, IBP_load, IBP_copy, and IBP_mcopy for data transfer; and IBP_status, for depot management. The IBP_allocate function is the most important operation. It 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). A successful IBP_allocate call returns a set of three capabilities for the allocated byte array — one for reading, one for writing, and one for management — that may be passed from client to client, requiring no registration from or notification to the depot

The other component of the storage stack that is critical to the logistical approach to network computation is the exNode. Building on end-to-end principles means that storage services with strong properties — reliability, fast access, unbounded allocation, unbounded duration, etc.—must be created in higher layers that aggregate more primitive IBP byte-arrays beneath them. 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 (e.g. distribution or replication across multiple depots), just as sequence numbers and timers are maintained to keep track of the state of a TCP session.

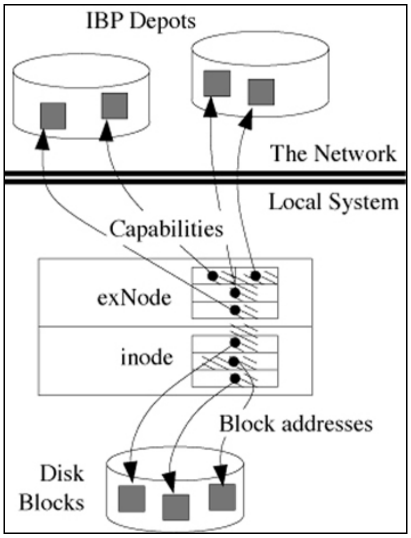

Figure 3: The exNode compared to a Unix inode.

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.

Following the example of the Unix inode, a single, generalized data structure called an external node, or exNode, has been implemented to aggregate byte arrays in IBP depots to form a kind of pseudo-file (Figure 3). We call it a pseudo-file because it lacks features (e.g. a name, ownership metadata, etc.) that a file is typically expected to have. To maximize application independence and interoperability, exNodes encode their resource information (e.g. IBP capabilities, URLs, etc.) and associated metadata in XML.

4. Logistical Networking Middleware

The IBP depot can leverage a variety of data transport mechanisms between nodes, and this is a key part of the common buffer service that provides the interoperability that Transnetworking is designed to achieve. Data transfer between IBP depots is parameterized by an operation code which specifies one of a number of alternative modules to implement the operations. Data transfer modules, called Data Movers [3], implement different protocols for moving data between depots, including standard TCP, UDP multicast, and SABUL, a specialized high throughput protocol based on UDP [8].

The LN middleware that has so far been built to use the transit layer functionality of IBP depots focuses on storage and implements limited network layer functionality at network endpoints in overlay style. The *Logistical Runtime System* (*LoRS*) consists of a set of tools and associated APIs that, by allowing users to draw on a pool of depots, enable the implementation of files and other storage abstractions with a wide range of characteristics, such as large size (through fragmentation), fast access (through caching), and reliability (through replication). Operations, such as replication and deletion, can be combined to provide more complex functionality, such as simple routing. LoRS tools also implement some transport layer services such as checksums, encryption, and compression and erasure codes, all of which is implemented at the end-points. Along with command line and GUI interfaces, LoRS provides an extensive, low-level C API that provides developers with even greater flexibility. Because LoRS was developed before computation was added to IBP, it makes no use network layer functionality implemented at depots, and so inherits from IP the vulnerability to Denial of Service attacks that all open networks suffer from; this is not an inherent limitation of the full transit layer architecture, because access to the transit layer can be limited by physically limiting access to link layer connectivity. The *Logistical Backbone (L-Bone)* is the LN resource discovery service [1, 2]; it maintains information about IBP depots such as hostname, port, storage availability, proximity between depots, etc. Users can also query the L-Bone to determine proximity between depots and the user to improve upload or download performance. Finally, the *Logistical Distribution Network* (*LoDN*) is a directory service that supports a global file service and many-to-many content distribution. Although IBP storage is time limited, the LoDN directory that holds the exNodes to the stored data will periodically refresh the appropriate allocations, moving the data to another depot in case of refresh or depot failure. In this later regard, LoDN can be viewed as a kind of asynchronous router of data in time and space (Figure 4).

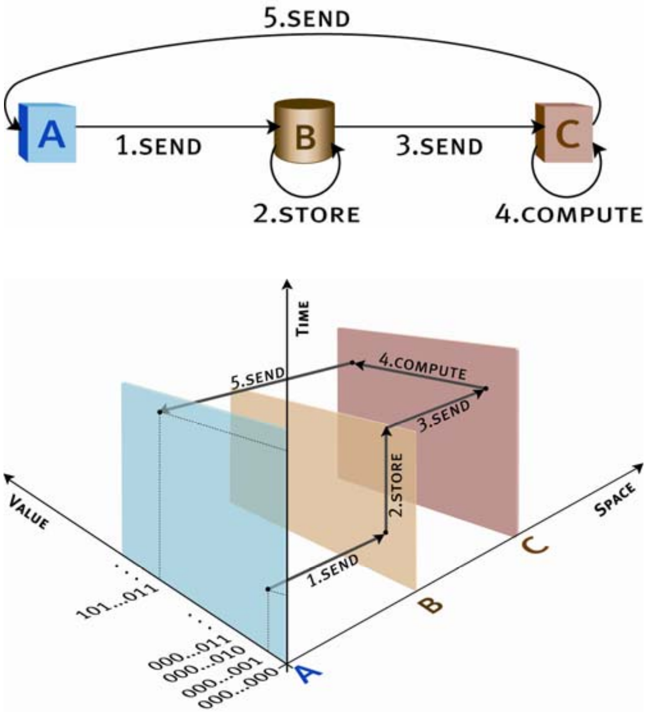

Figure 4: Transit layer connectivity in the transfer, storage, and processing dimensions.

All the components of LN software are not only freely available, but in widespread use within various applications communities. In particular, IBP is already deployed on hundreds of nodes of a global testbed (see Facilities Description). Multi-terabyte private depot deployments in the DOE science community, Brazil and the Czech Republic are also in use. The availability of these resources and the tools to exploit them has generated considerable research and application interest. The technology is being used for a growing list of project, including the movement of mass data sets in Astrophysics, Fusion Energy, and Bioinformatics; content distribution for the Linux community [11]; on-demand, adaptive caching service in distributed visualization applications [7]; for research into video on demand [9].

5. References

[1] *The Logistical backbone*, http://loci.cs.utk.edu/lbone, 2004.

[2] A. Bassi, M. Beck, T. Moore, and J. S. Plank, "The Logistical Backbone: Scalable Infrastructure for Global Data Grids," in *Asian Computing Science Conference 2002*. Hanoi, Vietnam: Springer Verlag, 2002.

[3] M. Beck, Y. Ding, E. Fuentes, and S. Kancherla, "An Exposed Approach to Reliable Multicast in Heterogeneous Logistical Networks." In Workshop on Grids and Advanced Networks (GAN03), Tokyo, Japan, May 12-15, 2003.

[4] M. Beck and T. Moore, "Locating Interoperability in the Network Stack," Computer Science Dept., University of Tennessee, Knoxville, TN, Technical Report, ut-cs-04-520, February 15, 2004.

[5] M. Beck, T. Moore, and J. S. Plank, "An End-to-end Approach to Globally Scalable Network Storage," in *Proceedings of SIGCOMM 2002*. Pittsburgh, PA, 2002, pp. 339-346.

[6] Computer Science and Telecommunications Board (CSTB) of the National Research Council, "The Internet's Coming of Age." Washington, DC: National Academy Press, 2001, pp. 256.

[7] J. Ding, et al., "Remote Visualization by Browsing Image Based Databases with Logistical Networking," in *Proceedings of SC2003*. Phoenix, AZ, 2003.

[8] R. L. Grossman, et al., "Experimental Studies Using Photonic Data Services at IGrid 2002," *Journal of Future Computer Systems*, vol. 19, no. 6, pp. 945-955, 2003.

[9] K. Mayer-Patel and M. Beck, "A Logistical Networking Model for Video-On-Demand." In IEEE International Conference on Multimedia and Expo (ICME'02), Lausanne, Switzerland, August 26-29, 2002.

[10] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp. 277-288, November, 1984.

[11] G. Wood, "Linux users on Internet2 networks enjoy the benefits of Logistical Networking," in *I2-News*, 2003.