A Blueprint for Introducing Disruptive Technology into the Internet*

Larry Peterson
Princeton University

Tom Anderson
University of Washington

David Culler, Timothy Roscoe
Intel Research – Berkeley

ABSTRACT

This paper argues that a new class of geographically distributed network services is emerging, and that the most effective way to design, evaluate, and deploy these services is by using an overlay-based testbed. Unlike conventional network testbeds, however, we advocate an approach that supports both researchers that want to develop new services, and clients that want to use them. This dual use, in turn, suggests four design principles that are not widely supported in existing testbeds: services should be able to run continuously and access a slice of the overlay's resources, control over resources should be distributed, overlay management services should be unbundled and run in their own slices, and APIs should be designed to promote application development. We believe a testbed that supports these design principles will facilitate the emergence of a new *service-oriented network architecture*. Towards this end, the paper also briefly describes PlanetLab, an overlay network being designed with these four principles in mind.

1. INTRODUCTION

The Internet was founded on a simple model in which the routers inside the network are responsible for forwarding packets from source to destination, and application programs run on the hosts connected to the edges of the network. However, the last few years have seen a blurring of the distinction between packet forwarding and application processing, as new widely-distributed applications are making their own forwarding decisions. This emerging class of applications includes network-embedded storage [12], peer-to-peer file sharing [16, 4], content distribution networks [20], robust routing overlays [17, 2], scalable object location [3, 15, 18, 14], and scalable event propagation [5]. At the same time, network elements such as layer-7 switches and transparent caches are performing application-specific processing.

These emerging services are the result of a convergence of two historically separate research communities. One is the distributed systems community, which traditionally viewed the network as merely providing bit-pipes between edge machines, but is increasingly embedding functionality at important crossroads and access points throughout the network. The other is the network community, which traditionally worried about forwarding packets without regard to application semantics, but is increasingly aware of the synergy that comes from unifying the computational and storage resources within the network with the requirements of the application.

We believe that the full integration of these two perspectives will result in a new, *service-oriented network architecture*. Unfortunately, this work is being done in an environment in which the Internet is less and less influenced by research, and increasingly shaped by commercial interests. In fact, a recent report from the National Research Council points to the ossification of the Internet [13]:

> ...successful and widely adopted technologies are subject to ossification, which makes it hard to introduce new capabilities or, if the current technology has run its course, to replace it with something better. Existing industry players are not generally motivated to develop or deploy disruptive technologies...

This paper offers a blueprint for introducing disruptive technologies into the Internet through the use of overlay networks. Overlays provide the right opportunity for innovation because they can be rapidly programmed to provide an innovative capability or feature, without having to compete with the existing infrastructure in terms of performance, reliability, administrative tools, and so on. Over time, we expect the "weight" such overlays place on the underlying Internet will result in the emergence of a new architecture, much in the same way the Internet (itself an overlay network) influenced the telephony network on which it was built. This paper concludes by speculating about what this new architecture might look like.

2. FROM DESIGN TO DEPLOYMENT

The example applications mentioned above are currently being designed and studied using a combination of simulation, network emulation, and small-scale testbeds. All of these systems would greatly benefit from a testbed that supports large-scale, real-world experiments. Overlays would play two roles in such a testbed: applications using the testbed will be structured as overlays, but the testbed itself is also an overlay from the perspective of controlling, managing, and deploying applications over it. We define such a testbed along three main dimensions.

First, the physical dimensions of the overlay network should

*This work is supported by Intel Corporation, NSF grant ANI-9906704, and DARPA contract F30602–00–2–0561.

be large—on the order of 1000 sites—to enable wide deployment of services and measurement tools. We envision the majority of these sites running a single overlay node that connects a large client population to the overlay. We can think of these nodes as providing a thousand viewpoints on the network. They should be selected to provide a rich diversity of link behavior, and wide-spread geographic coverage. We also envision the overlay including roughly a hundred sites with more substantial computing resources (e.g., a cluster of machines) located at network crossroads (e.g., peering points and co-location centers).

Second, the overlay consists of two main software components: (1) a *virtual machine monitor* (VMM) running on each node; and (2) a *management service* used to control the overlay. The VMM specifies the interface to which the services distributed over the testbed are written. This is a controlled interface to abstract resources (e.g., network ports and logical disks), rather than an interface that provides direct access to hardware. The management service is used to control the testbed; for example, to discover the set of nodes in the overlay, monitor their health, and to keep the software running on these nodes up-to-date.

The final dimension, which is the most distinguishing characteristic of the approach we advocate, is the overlay's mode of operation. Rather than view the overlay strictly as a testbed, we take the long-term view in which the overlay is both a research testbed and a deployment platform. In other words, the overlay should support the seamless migration of an application from early prototype, through multiple design iterations, to a popular service that continues to evolve.

Using an overlay as both a research testbed and a deployment platform is synergistic. As a testbed, the overlay's value is to give researchers access to (1) a large set of geographically distributed machines; (2) a realistic network substrate that experiences congestion, failures, and diverse link behaviors; and (3) the potential for a realistic client workload. Its value as a deployment platform is to provide (1) researchers with a direct technology transfer path for popular new services, and (2) users with access to those new services. We believe that supporting both roles is critical to the success of the system.

An important consequence of dual use is that the testbed must support the continuous operation of network services, as opposed to providing mechanisms for starting and stopping experiments. This leads to an obvious tension between the needs of "test & measure" researchers for reproducible results, and those interested in the system as a deployment platform. As an overlay, however, the bandwidth/latency/loss that can be achieved through the network is variable, and hence, unpredictable. The overlay's real value as a research platform is in providing realistic network conditions.

The dual use paradigm also implies that many experimental services will have to share nodes, since it is unreasonable to dedicate on the order of 1000 distributed nodes to a single experimental service for months at a time. This leads to strong security and resource control requirements on the virtual machine monitor.

Another way to understand the essential aspects of the approach we are proposing is to compare it to several current testbed efforts.

- Internet2, which includes the Abilene backbone, is a physical network that includes high-speed optical links connecting major research universities [10]. The network nodes are closed commercial routers, making it impossible to introduce new functionality into the middle of the network. In contrast, the main virtue of an overlay is that the nodes are fully programmable.
- Emulab is a network experimentation facility supported by the University of Utah [22]. Researchers are able to schedule a set of nodes for experiments, and dictate how the nodes are configured to emulate different network topologies. An overlay also has an experimental aspect, but its dual role as a deployment platform means that experimental systems run continuously, rather than for a limited period of time.
- The Grid is a collection of middleware, called Globus [6], that allows researchers to distribute large scientific applications across a distributed set of computational resources [9]. The main difference between the Grid and the proposed overlay is one of emphasis: the Grid is primarily interested in gluing together a modest number of large computing assets with high-bandwidth pipes, while the overlay is primarily concerned with scaling less bandwidth-intensive applications across a wider collection of nodes.
- The ABONE is an overlay testbed that grew out of the Active Networks initiative [1]. It allows service developers to dynamically load their applications onto the overlay's nodes. Although the high-level goals are similar, one important difference is that Active Networks is primarily focused on supporting extensibility of the network forwarding function, whereas we take a more inclusive view of the types of applications that will be deployed throughout the network, including those that involve a significant storage component.
- The XBONE is an overlay network with support for IP-in-IP tunneling [19]. It also includes a GUI-based toolset for establishing and monitoring specific overlay configurations. The XBONE and the proposed overlay share the goal of supporting multiple independent overlays on the same set of machines, but the XBONE is limited to IP tunnels, whereas we also hope to support higher-level overlays, such as those implemented by peer-to-peer systems.

An alternative to developing a new service on a traditional testbed is to package it as an application that can run on any desktop machine. If it proves to be a popular service, users will install it. File sharing systems like Napster and KaZaA have successfully adopted this approach, but it is not clear that it extends to other styles of services. More importantly, deploying services in the wild by viral dissemination has several shortcomings.

First, it does not work unless the service is immediately and widely popular. This makes it impossible, for example, to do research studies into algorithms for managing overlay networks. Often, the technically superior solution will not be used if its applications or content happen to be less popular.

Second, it is difficult to modify such a system once it is deployed, making the process of learning from experience very cumbersome. The next version of the algorithms—or more generally, the next iteration of the design-evaluate-refine cycle—often requires a new set of compelling applications.

Third, such systems are not secure. The recent problem with KaZaA exposing all files on the local system is just one example of the potential dangers. We prefer a system that allows us to understand how to sandbox viral peer-to-peer applications so that users need not trust their entire systems to the coding standards of random peer-to-peer developers.

3. DESIGN PRINCIPLES

Our vision of an overlay that serves both service developers and service users has several implications for the architecture of the system. This section outlines the key design principles that shape such an overlay.

3.1 Slice-ability

Because services are expected to run continuously, rather than be globally scheduled to run one at a time, the overlay must support distributed virtualization. That is, each application acquires and runs in a *slice* of the overlay. Distributed virtualization, in turn, requires each node to multiplex multiple competing services. Thus, a key responsibility of the VMM running on each node is to allocate and schedule slices of the node's processor, link, and storage resources.

The node slicing mechanism must be *secure* in that it protects the node from faulty or malicious services. It must also use a *resource control mechanism* such as proportional share scheduling to enforce bounds on the resources consumed by any given service. Finally, it must be *scalable* in the sense that each node is able to efficiently multiplex resources among a large number of services.

Note that while each node is able to enforce slices of its local resources (including its outgoing link bandwidth), since the system is an overlay network, it is not possible to ensure that a given application receives predictable network performance, given that the Internet does not yet support bandwidth reservations.

Finally, in addition to viewing a slice as the collection of resources available on some set of nodes, a slice can also be characterized at the global level in terms of how those nodes (resources) are spread throughout the Internet. For example, one slice might contain resources that are uniformily distributed over as wide of area as possible, while another might wish to ensure that its resources are clusterd in autonomous systems with a high degree of connectivity.

3.2 Distributed Control of Resources

In its dual role as testbed and deployment platform, there will be two types of users: (1) researchers that want to install and evaluate new services, and (2) clients that want to access these services. Initially, the researchers are likely to be the only users (it is important that the researcher community develop applications that they themselves want to use), but in order to function as a deployment platform, the overlay must also provide explicit support for people that are willing to add nodes to the overlay for the sake of accessing its services.

These two user populations have different views of the nodes. Researchers want to dictate how their services are deployed. It may be as simple as "on as many nodes as possible" but they may also want to dictate certain node properties (e.g., at a crossroads site with sufficient storage capacity). Clients want to decide what services run on their nodes. They should be required to allocate slices of their machines to experimentation—thereby postponing future ossification—but they need to be able to set policy on how resources are allocated to different services.

This shared control of resources implies a highly-decentralized control structure. For example, a central authority may provide legitimate service developers with credentials that allow them to request a slice of a node, but each node will independently grant or deny such a request based on local policy. In essence, the node owner decides how many of the node's resources may be consumed by different services.

From a security perspective, applications have to trust both the central testbed authority and the physical security of the nodes at individual sites. Ultimately, this means service overlays need to be aware of where they cross administrative domain boundaries, and protect themselves from rogue elements.

3.3 Unbundled Management

Rather than view testbed management as a single, fixed service, overlay management should be unbundled into a set of largely independent sub-services, each running in their own slice of the overlay. For example, overlay management might be partitioned as follows:

- discover the set of nodes in the overlay and learn their capabilities;
- monitor the health and instrument the behavior of these nodes;
- establish a default topology among the nodes;
- manage user accounts and credentials;
- keep the software running on each node up-to-date; and
- extract tracing and debugging information from a running node.

Some of these sub-services are part of the core system (e.g., managing user accounts), and so there must exist a single, agreed-upon version. Others can be provided through a set of alternative services. The system will need to provide a default set of such services, more or less bundled, but we expect them to be replaced by better alternatives over time. In other words, the management structure should be engineered for innovation and evolution.

To better appreciate the power of being able to run new management services in a slice of the overlay, imagine if we were able to go back to the days when IP was defined; we could then put in the instrumentation hooks we need today to measure Internet traffic. An overlay that ensures that some fraction of each node can be programmed will give us that ability.

The strategy of unbundling the management service requires that appropriate interfaces be defined. First, the individual services are likely to depend on hooks in the VMM that, for example, make it possible to retrieve the status of each node's resources. Second, the various sub-services may depend on each other; for example, the node monitoring service might provide input to a realtime database that is later queried by the resource discovery service. Allowing these two services to evolve independently will require a common representation for node attributes.

3.4 Application-Centric Interfaces

Perhaps the single greatest failure of testbeds, in general, is that they do not promote application development. One reason is that they are short-lived: experience teaches us that no one builds applications for pure testbeds since their lifetime is, by definition, limited. Related to this point, there is usually little motivation to integrate the testbed with desktop machines, making it nearly impossible for clients to access applications that might be available. The hope is that by designing the overlay to serve as both a research testbed and a deployment platform we will lower the hurdle for application development.

A more tangible problem is that it is difficult to simultaneously do the research needed to create an effective testbed, and use the testbed as a platform for writing applications. Users require a stable platform, which is at odds with the need to do research on the platform. To make matters worse, such research often results in new APIs, requiring that applications be written from scratch.

Thus, our final design principle is that the overlay must support an existing and widely adopted programming interface, with platform-related research changing the underlying implementation over time, while the API remains largely unchanged. Should an alternative API emerge from this effort, new applications can be written to it, but the original API is likely to be maintained for legacy applications.

4. PLANETLAB

We are currently building an overlay testbed, called PlanetLab, that adheres to these design principles. We envision PlanetLab achieving the goals outlined in this paper in three phases. Our strategy is to incrementally enhance the capabilities of PlatnetLab in accordance with the next user community we hope to attract.

Seed Phase: We have seeded PlanetLab with 100 machines and provided just enough functionality to meet the needs of a small, known set of researchers. These researchers are implementing and experimenting with many of the services mentioned in the introduction. We do not expect to support a client community during this phase, but instead PlanetLab will function as a pure testbed.

Researchers as Clients: We are now opening PlanetLab to the larger research community, which we expect to drive the size towards 1000 sites. We recognize, however, that adding clusters at strategic Internet crossroads will require broader government and industrial support. During this phase, the user community will still be primarily researchers experimenting with their services. We also expect these researchers will themselves begin to use primitive services provided by these applications.

Attracting Real Clients: Our thesis is that the research community is poised to develop innovative services, and that a true client base will follow. Accessing a service in this world is equivalent to joining an overlay network, and so we expect a growing client community to connect to PlanetLab. To the extent successful services are developed, we also expect to spin-off physically distinct copies of PlanetLab.

The hardware is dedicated PlanetLab nodes, as opposed to client-owned desktop machines. To minimize heterogeneity headaches, we prescribe the permitted hardware configurations. To ensure conformance with the common interfaces and policies, only the central PlanetLab authority (as opposed to node owners) have root-access to the machines. Moreover, while node owners will be able to establish policy on how their nodes are sliced, PlanetLab will retain the right to allocate some fraction of each node to experimental services. Finally, to ensure stability, PlanetLab will maintain a "core" of highly available nodes and sites.

The initial default management service is provided by a combination of the Ganglia resource monitoring tool [8], a boot and software update process based on Xenoboot [23], and an account/project management interface patterned after Emulab. We are currently evolving Ganglia into two separate components: a *resource monitor* that reports the resources available on a given node, and a *resource broker* that aggregates this information from a set of nodes, and responds to requests for slices.

Moreover, we are evolving account managment in a way that moves PlanetLab away simply providing a set of Unix accounts, towards a service-oriented architecture in which services dynamically create the slices in which they run. This is a three stage process: (1) a service manager first contacts a resource broker to select (discover) a set of candidate nodes that constitute a slice, (2) it then contacts the nodes in that set to initialize a network of virtual machines (this involves a per-node *admission control* decision), and (3) it launches the service in the resulting slice.

In terms of the VMM, our strategy is to evolve the kernel component of PlanetLab toward a strong notion of an *isolation kernel*, while maintaining an operational system that is usable by researchers for both experimentation and long-term deployment. Towards this end, we are pursuing two complementary development efforts.

The first builds on a traditional Unix-like operating system that does not distinguish between the interface at which resource allocation and protection is applied (the *isolation* interface) and the system call interface used by program developers (the *application* interface). Speficically, we have settled on Linux since it is a popular platform for implementing network services. We then plan to augment Linux with functionality to provide better security and resource isolation between services running over it. The most attractive way of doing this includes virtualizing the kernel (rather than the hardware) in the style of Vservers [11], replacing privileged kernel functionality with safe alternatives (e.g., safe raw sockets), and adding support for slice-ability (e.g., resource containers plus proportional scheduling of the link and CPU).

The second effort revolves around isolation kernels like Denali [21] and Xenoservers [7], which provide a low-level isolation interface that closely resembles virtualization of the hardware. Operating systems such as Linux, BSD, and Windows XP can be ported to this virtual machine, an operation that is greatly simplified by the similarity between the virtual machine architecture and the "real" hardware the operating systems were originally developed for. The expectation is that it will be easier to assure the correctness of a minimal isolation kernel, thereby improving the overall security of the system.

5. CONCLUDING REMARKS

Just as the Internet was originally an experimental packet-switched network that evolved into a ubiquitous communication substrate over time, we believe it is possible to design a shared overlay infrastructure that can evolve from a modest research testbed into a planetary-scale service deployment platform. In fact, it is not an accident that our underlying design philosophy is similar to that of the Internet: we define the minimally required interfaces, and through the use of virtualization (slice-ability), the system is engineered to evolve.

There is a second interesting comparison point between the Internet and experimental testbeds like PlanetLab. The Internet was originally an overlay network that viewed the underlying telephony system as providing a collection of leased lines, but as it grew, the "weight" of the Internet eventually contributed to a complete re-design of the phone network. Likewise, overlay networks like PlanetLab initially view the Internet as providing a set of bit-pipes, but over time it is likely that the Internet architecture will evolve to better support service-level overlays. In other words, one of the most interesting questions emerging from this effort is how the interaction between the Internet and an overlay network like PlanetLab eventually results in a new service-oriented network architecture.

As an illustrative example of how this process might take place, it is already the case that multiple overlay services running on Planetlab independently probe the network as part of their topology-selection process. This is inefficient, and at the very least, there needs to be a shared "topology probing" mechanism. An additional step beyond such a mechanism would be to define an interface that allows overlays and the Internet to share topology information with each other. Eventually, one could imagine a few well-designed topology services evolving, with other services employing one of them rather than inventing a new one of their own. Whether a single "winner" emerges from this effort—and perhaps is subsumed into a new Internet architecture—or the very nature of a programmable overlay means that services will continue to define their own routing machinery, remains a subject of speculation.

6. REFERENCES

[1] ABONE. http://www.isi.edu/abone/.

[2] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 131–145, Chateau Lake Louise, Banff, Alberta, Canada, October 2001.

[3] M. Balazinska, H. Balakrishnan, and D. Karger. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In *Proceedings of Pervasive 2002 - International Conference on Pervasive Computing*, Zurich, Switzerland, August 2002.

[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, Chateau Lake Louise, Banff, Alberta, Canada, October 2001.

[5] P. Druschel, M. Castro, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20, 2002.

[6] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Internationall Journal of Supercomputer Applications*, 11(4):115–128, 1997.

[7] K. Fraser, S. Hand, T. Harris, I. Leslie, and I. Pratt. The Xenoserver Computing Infrastructure, 2002. http://www.cl.cam.ac.uk/Research/SRG/netos/xeno/xeno-general.pdf.

[8] Ganglia. http://ganglia.sourceforge.net.

[9] Grid. http://www.globus.org.

[10] Internet2. http://www.internet2.edu.

[11] J. Gelinas. Virtual private servers and security contexts. http://www.solucorp.qc.ca/miscprj/s_context.hc.

[12] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth international Conference on Architectural Support for*

Programming Languages and Operating Systems (ASPLOS 2000), Nov. 2000.

[13] National Research Council. *Looking Over the Fence at Networks*. National Academy Press, Washington D.C., 2001.

[14] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *Proceedings of the IEEE INFOCOM Conference*, New York, NY, June 2002.

[15] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany, November 2001.

[16] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, A Large-Scale Persistent Peer-to-Peer Storage Utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 188–201, Chateau Lake Louise, Banff, Alberta, Canada, October 2001.

[17] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-end Effects of Internet Path Selection. In *Proceedings of the ACM SIGCOMM Conference*, Cambridge, MA, September 1999.

[18] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM Conference*, San Diego, CA, September 2001.

[19] J. Touch and S. Hotz. The X-Bone. In *Proceedings of the Third Global Internet Mini-Conference at Globecom '98*, pages 75–83, Sydney, Australia, November 1998.

[20] L. Wang, V. Pai, and L. Peterson. The Effectiveness of Request Redirection on CDN Robustness. In *Proceedings of the 5th Symposium on Operating System Design and Implementatio (OSDI)*, Boston, MA, December 2002.

[21] A. Whitaker, M. Shaw, and S. Gribble. Scale and Performance in the Denali Isolation Kernel. In *Proceedings of the 5th Symposium on Operating System Design and Implementatio (OSDI)*, Boston, MA, December 2002.

[22] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the 5th Symposium on Operating System Design and Implementatio (OSDI)*, Boston, MA, December 2002.

[23] Xenoboot. http://www.cl.cam.ac.uk/Research/SRG/netos/xeno/boot/.