COVER FEATURE

Overcoming the Internet Impasse through Virtualization

Most current Internet research involves either empirical measurement studies or incremental modifications that can be deployed without major architectural changes. Easy access to virtual testbeds could foster a renaissance in applied architectural research that extends beyond these incrementally deployable designs.

Larry Peterson
Princeton University

Thomas Anderson
University of Washington

Scott Shenker
UC Berkeley and ICSI

Jonathan Turner
Washington University

The Internet's stunning success has changed the way we work, play, and learn. The Internet architecture, developed over 30 years ago, has proven its worth by the vast array of applications it now supports and the wide variety of network technologies over which it currently runs. Nonetheless, the Internet's increasing ubiquity and centrality has brought with it a number of challenges for which the current architecture is ill-suited. Although developers and researchers have shown increasing interest in new architectures that could address these challenges,[1-8] the prospects for significant change in its existing architecture appear slim. In addition to requiring changes in routers and host software, the Internet's multiprovider nature also requires that ISPs jointly agree on any architectural change.

The need for consensus is doubly damning: Not only is reaching agreement among the many providers difficult to achieve, attempting to do so also removes any competitive advantage from architectural innovation.

Short of the Internet's imminent collapse, there seems little hope for major architectural changes—those innovations that would alter its basic architecture. Worse, the situation continues to deteriorate.

The inability to adapt to new pressures and requirements has led to an increasing number of ad hoc workarounds, many of which violate the Internet's canonical architecture. While derided by architectural purists, these modifications have usually arisen to meet legitimate needs that the architecture itself could not. These architectural barnacles—unsightly outcroppings that have affixed themselves to an unmoving architecture—can serve a valuable short-term purpose, but they significantly impair the Internet's long-term flexibility, reliability, and manageability.

The daunting barriers to deployment of new architectures, while discouraging, do not directly hinder further research. Architectural invention continues without limitations, even if without hope of adoption. However, live experimentation with new architectures has proven more problematic. The main avenue for live experimentation, as opposed to simulation or emulation, is to use testbeds.

However, traditional testbeds have severe limitations that constrain our ability to evaluate new architectures.[9] Instead of being satisfied with paper designs that have no future, the design community should return to its roots of applied architectural research with the intention of once again changing the world.

THREE REQUIREMENTS

Overcoming the current impasse will not be easy and will require addressing three separate requirements:

- Researchers must be able to experiment easily with new architectures on live traffic.

- There must be a plausible deployment path for putting validated architectural ideas into practice.
- Instead of focusing on a single narrow problem, the proposed solutions should be comprehensive so that they can address the broad range of current architectural problems facing the Internet.

We propose to meet these three requirements by constructing a virtual testbed that will support multiple simultaneous architectures, serving all the communication needs of standard clients and servers. This virtual testbed approach provides a clean path for unilaterally and globally deploying new architectures. Because it does not require universal architectural agreement, this approach offers a more plausible deployment scenario for radical new designs that systematically tackle the complete set of problems facing the Internet today.

Central to our proposal is the concept that *virtualization*—as used in virtual memory, virtual machines, and elsewhere—is nothing more than a high-level abstraction that hides the underlying implementation details. With virtualization, nodes can treat an overlay as if it is the native network, and multiple overlays can simultaneously use the same underlying overlay infrastructure. Both aspects of virtualization are crucial to our virtual testbed proposal.

PHYSICAL TESTBEDS AND OVERLAYS

Before they can even consider deployment of a proposed architecture, researchers must adequately evaluate it. Although simulation and emulation are valuable tools for understanding new designs, they cannot substitute for experimentation with live traffic.

Preparing an implementation to deal with the real world forces designers to confront the many unpleasant realities that paper designs frequently avoid, such as multiple providers, legacy networks, anomalous failures and traffic conditions, and unexpected and diverse application requirements. Moreover, live traffic provides a fuller picture of how an architecture will perform, strengthening the case that the architecture will actually provide the claimed benefit.

Currently, researchers use physical testbeds and overlays to experiment with new architectures. Overlays have also found favor as a valid deployment path. Both of these approaches, however, have limitations.

Physical testbeds

The traditional platform for live experimentation, physical testbeds consist of leased lines connecting a limited set of locations. Testbeds can be roughly categorized as production- or research-oriented.

Production testbeds, such as Internet2, support real traffic from real users, often in large volume and across many sites. As such, they provide valuable information about an architecture's operational behavior. However, a production testbed's users have no choice about participating in the testbed and usually don't even realize their traffic has become part of an experiment. They thus expect the performance and reliability to be no worse than the standard Internet. Production testbeds must therefore be extremely conservative in their experimentation, using well-honed implementations of incremental changes.

Traditional testbeds offer too little bang for the buck and clearly cannot lead us into the future.

Research testbeds such as DETER (Defense Technology Experimental Research) do not carry traffic from a wide variety of real users. Instead, they are typically driven by synthetically generated traffic, a small collection of intrepid users, or both. Thus, they are more adventurous and capable of running first-cut implementations of radically new designs.

Unfortunately, this lack of real traffic also means that the results are less likely to be indicative of real operational viability. As a result, neither a production nor a research testbed can produce the data needed to adequately evaluate new architectures.

Further, because they utilize dedicated transmission links, both testbed categories involve substantial cost, which makes operating them on a large scale prohibitively expensive. This typically limits their use to a small geographic area and even then requires substantial funding support.

These factors make it difficult to build a compelling case for new architectural designs based on a testbed evaluation. Given their limitations, traditional testbeds offer too little bang for the buck and clearly cannot lead us into the future.

Overlays

Becoming more widespread recently, overlays are being used both as an experimental platform and a deployment path.[10-12] They are not limited geographically and their usage is voluntary. Moreover, overlays typically do not involve significant expenditures, thus avoiding many of the problems that plague traditional testbeds. With the advent of PlanetLab[13]—an open platform for developing, The virtual testbed approach provides a focal point for a new attitude toward overlays.

deploying, and accessing planetary scale services—creating and maintaining an overlay has become a straightforward task. However, overlays still suffer from limitations of their own.

First, overlays have largely been seen as a way to deploy narrow fixes to specific problems in the Internet architecture, whether for performance,[10] availability,[11] denial of service,[11,14] content distribution, or multicast.[15] Researchers have viewed the solution to each of these problems as an isolated function, and they have done little to determine how any of the solutions might work together. More importantly, they have devoted little thought to identifying how a set of overlays might ultimately replace the underlying Internet architecture.

Second, to date, overlays have been architecturally tame. Because the emphasis has been on deployment in today's Internet rather than on architectural innovation leading to tomorrow's Internet, most current overlays typically assume IP or a close cousin as the architecture inside the overlay itself: the interoverlay node protocol. As such, overlays have not been the source of dramatic architectural advancement.

Thus, on their current trajectory, overlays will likely become just a better way of attaching yet another barnacle, rather than an agent of fundamental change. The field needs a philosophical revolution in how developers use overlays, not a technical alteration in how they build them. Therefore, the virtual testbed approach that we propose provides a focal point for a new attitude toward overlays rather than a technical advancement.

VIRTUAL TESTBED

To address these problems and provide an attractive platform for experimentation and possible deployment, we propose a *virtual testbed* approach. Virtual testbeds have two basic components: an overlay substrate and a client-proxy mechanism.

Key features

An *overlay substrate* provides a set of dedicated but multiplexed overlay nodes. By multiplexing these nodes, as first advocated in PlanetLab, multiple experiments can run simultaneously on the same infrastructure. The effort of instantiating and maintaining the overlay is amortized across the many concurrently running experiments, drastically lowering the barrier to entry that an individual researcher faces.

A host can use the *client-proxy mechanism* to opt in to a particular experiment running on a specific substrate overlay. This mechanism treats a nearby overlay node as the host's first-hop router without imposing any limitations on the experimental architecture. It also supports opt in at a fine granularity by, for example, routing local traffic directly or determining participation on a per-application basis. These two features resolve the barrier-to-entry and architectural limitations that overlays faced.

To encourage the use of overlays for more radical architectures, we have deployed a prototype of this approach on PlanetLab. It is relatively primitive in its original incarnation. PlanetLab currently includes more than 529 nodes that span 252 sites and 28 countries on five continents.

Technology overview

We estimate that a PlanetLab node is within a LAN hop of more than one million users. As the "PlanetLab Computing Platform" sidebar describes, PlanetLab software architecture multiplexes multiple slices, each running a different network service, application, or architecture. Users can view each slice as a set of virtual routers connected by tunnels to whatever topology the architecture selects.

Mostly, PlanetLab leverages straightforward technologies, but we still have some issues to explore. For example, achieving sufficiently high throughput rates on PlanetLab nodes is challenging: Stock PlanetLab nodes can forward packets at 60 Mbps. While we expect to achieve gigabit rates with modest optimizations, PlanetLab nodes clearly cannot compete with custom hardware.

Similarly, an overlay's virtual links cannot compete with dedicated links. In cases where timeliness is crucial, an overlay could use techniques such as those incorporated in OverQoS[16] MPLS paths to provide better service than a naïve tunnel over IP.

Moderately developed, the proxy technology still needs work. Our prototype proxy can catch and forward packets into the virtual testbed from interpose proxies on any IP address or port that the legacy client software identifies. Given that most client applications use name translation as the first step in communication, the proxy interposes on DNS requests and either returns the server's true IP address if the packets are for the normal Internet or a fake IP address if the packets are for the virtual testbed.

By interposing on the fake IP addresses, the proxy can then forward the packets to the nearest virtual testbed node, the ingress node. The proxy is

PlanetLab Computing Platform

PlanetLab is a geographically distributed computing platform for deploying, evaluating, and accessing planetary-scale network services. PlanetLab is a shared community effort by researchers at 252 sites in 28 countries, each of whom gets access to one or more isolated "slices" of PlanetLab's global resources via a *distributed virtualization* concept.

To encourage infrastructure innovation, PlanetLab's *unbundled management* principle decouples the operating system running on each node from a set of multiple, possibly third-party, network-wide services that define PlanetLab.[1] PlanetLab services and applications run in a slice of the platform: a set of nodes on which the service receives a fraction of each node's resources in the form of a virtual machine.

What's new in PlanetLab is distributed virtualization: the acquisition of a distributed set of VMs that the system treats as a single, compound entity. PlanetLab isolates services and applications from each other, thereby maintaining the illusion that each service runs on a distributed set of private machines. The platform must deliver isolation of slivers—one constituent VM of a slice running on a single node—by allocating and scheduling node resources, partitioning or contextualizing system namespaces, and enforcing stability and security between slivers sharing a node. The actual contents of a sliver within the VM are of little concern to the platform; for example, it should not matter to the platform whether the code in the sliver is running in a Java VM or written in assembly language.1

Figure A illustrates the PlanetLab node architecture. At the lowest level, each PlanetLab node runs a virtual machine monitor that implements and isolates virtual machines. The VMM also defines the API that implements the services.

PlanetLab version 3.0 currently implements the VMM as a combination of the Linux 2.6 kernel and a set of kernel extensions—in particular, vservers 1.9, a Linux patch that provides multiple, independently managed virtual servers running on a single machine and the SILK (Scout in Linux Kernel) module that provides CPU scheduling, network accounting, and safe raw sockets.[2,3]

The node manager, a privileged root VM running on top of the VMM, monitors and manages all the VMs on the node. Generally speaking, the node manager enforces policies on creating VMs and allocating resources to them, with services interacting with the node manager to create new VMs rather than directly calling the VMM. Moreover, all interactions with the node manager are local: Only services running in another VM on the node are allowed to call the node manager, meaning that remote access to a specific node manager is always indirect through one of the services running on the node.

Currently, most policy is hard-coded into the node manager, but we expect that local administrators will eventually be able to configure the policies on their own nodes. This is the purpose of the local administrator VM shown in Figure A.[2]

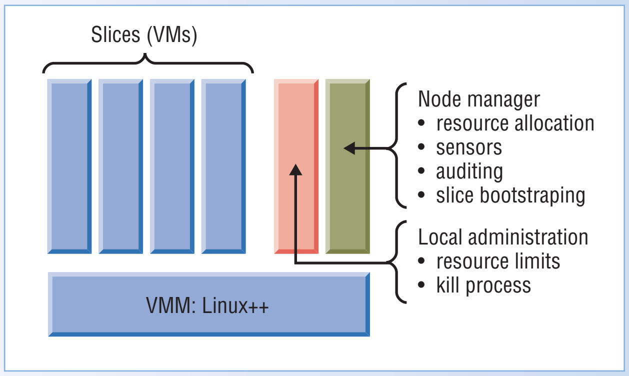

Figure A. PlanetLab node architecture. Each node runs a virtual machine monitor that implements and isolates virtual machines that the system treats as a single entity. Isolating services and applications from each other maintains the illusion that each service runs on a distributed set of private machines.

Example applications and services running on PlanetLab include network measurement, application-level multicast, distributed hash tables, storage **<<schemas?>>,** resource allocation services, distributed query processing, content distribution networks, management and monitoring services, overlay networks, router design experiments, and federated testbeds, among others.[4]

References

1. L. Peterson and Timothy Roscoe, "The Design Principles of Planet Lab," PDN-04-021; http://www.planet-lab.org/PDN/PDN-04-021/.
2. A. Bavier et al., "Operating System Support for Planetary-Scale Network Services," *Proc. 1st Symp. Networked Systems Design and Implementation* (NSDI), Usenix, 2004, pp. 253-266.
3. PlanetLab v. 3.0, PDN-04-023; www.planet-lab.org/pdn.
4. Timothy Roscoe, "What Are People Doing in/on/with/around PlanetLab?"; http://www.planet-lab.org/Talks/2003-04-15-HP-PlanetLab-Users.pdf.

designed to do this in as architecturally neutral a way as possible. The virtual testbed can then do whatever it wants with the packets, using the IP or non-IP protocols it deems appropriate to service the packet, then tunneling over protocols it hopes to replace. Because gaining real users requires providing access to legacy servers, the node on the far end of the virtual testbed—the egress node—reconverts the packet into Internet format for delivery to the server. The egress node behaves as a network address translator, manipulating the source address to ensure that reply packets also enter the virtual testbed.

Service hosting

Planet also can easily host a service within the virtual testbed that remains visible to nonparticipating clients. In this case, the virtual testbed provides DNS resolution to point the client to a nearby virtual testbed representative, in much the same

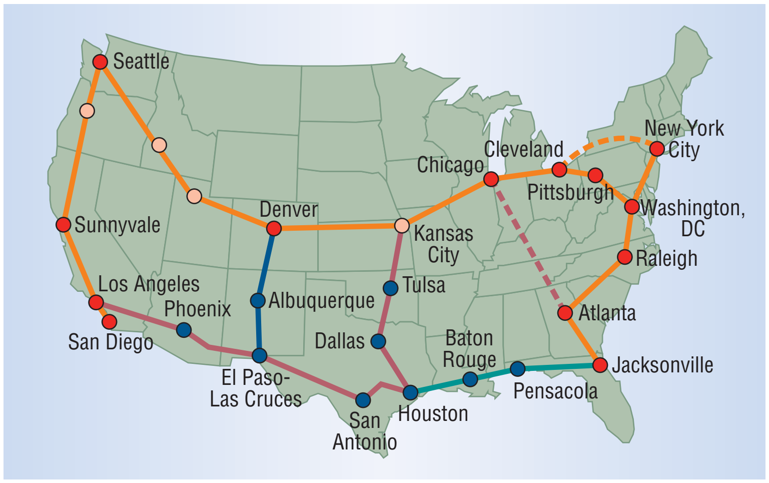

Figure 1. National LambdaRail. Using this backbone, the virtual testbed can support larger traffic volumes, with PlanetLab nodes aggregating traffic from local sites and feeding it to the backbone nodes.

way that content delivery networks operate. The local representative can then translate the packets into an internal format for delivery to the server and translate the packets back to Internet format for the reply. In addition, developers can use this approach to point to multiple virtual testbeds.

Some security issues must still be resolved, particularly about how to respect server address-based policy restrictions when the overlay shields the source's IP address.

Quality of service

One drawback of the virtual overlay approach is that it cannot control the quality of service for packets traversing the virtual testbed. This limits the extent to which virtual testbeds or any overlay can test architectures for QoS. We do not consider this a fatal flaw, however, because an architecture deployed on a virtual testbed would still deliver relative QoS, as good a service as possible given the underlying link characteristics, even if it could not maintain the absolute QoS of a dedicated link in all cases.

Moreover, simulation and emulation can effectively evaluate QoS. Further, the enormous amount of literature on QoS in the past decade has made it the least-mysterious aspect of new architectures. Many other issues that involve routing and addressing warrant more urgent attention and better suit the virtual testbed approach.

Inspiration

The virtual testbed borrows heavily from the ideas of the X-Bone[12] and the virtual Internet,[17] but we have a different emphasis. Because the X-Bone supports automated establishment and management of overlays, individual experiments running on the virtual testbed could use this suite of tools. The virtual testbed focus centers on virtualizing the overlay nodes themselves to support multiple simultaneous and potentially radically different architectures running on the same hardware. Although the X-Bone architecture supports this, it is not the major focus. The virtual Internet architecture,[17] based in part on the X-Bone work, allows multiple levels of virtualization. However, it remains closely tied to the current Internet architecture, which makes it unsuitable for experimenting with radical deviations from it.

Beyond this initial prototype, our future plans include a high-performance backbone, built using dedicated MPLS tunnels on Internet2, and then around a set of scalable substrate routers and links provided through the National LambdaRail (NLR), shown in Figure 1. With this backbone, the testbed will support larger traffic volumes, with PlanetLab nodes aggregating traffic from local sites and feeding it to the backbone nodes, while also enabling higher-bandwidth applications at sites close to backbone nodes. This hybrid approach captures the benefits of traditional testbeds without inheriting their flaws.

Fully utilizing the NLR backbone likely requires routers that also support virtualization. This can be accomplished at sufficient speeds using a pool of processing engines interconnected through a high-speed switch. We envision that most processing elements will include a network processor system capable of high-performance packet processing. A general-purpose processor will provide control functions, offer storage services, and facilitate migration from lower-performance sequential software designs to the parallelized designs needed to fully exploit network processor architectures.

Current-generation network processors provide enough processing resources to deliver approximately 3 to 5 Gbps of throughput for moderately complex applications. Thus, a backbone node capable of supporting 50 Gbps of throughput—3 backbone links at 10 Gbps each, plus 20 Gbps of access bandwidth—will require 10 to 16 such processing engines. These engines could provide even higher performance by incorporating advanced field-programmable gate arrays that combine reconfigurable hardware and multiple processor cores in a single device.[18]

Our plan to integrate a high-speed backbone with PlanetLab has two major advantages over other purely physical testbeds. First, PlanetLab-based overlays serve as an access network for the backbone, bringing traffic from a large user community onto the backbone. Second, developing and deploying the hardware does not gate the architectural work. Researchers can first experiment with their architecture as an overlay and then later expand it to include the high-speed backbone as the platform supports it.

DEPLOYMENT

The traditional but now discredited deployment story predicted that, after having been validated on a traditional testbed, a next-generation architecture would, through some magical process of consensus and daring, be simultaneously adopted by ISPs and router vendors alike.

With this story no longer even remotely possible, can we find a *plausible* deployment alternative? We use the term plausible because adopting new technologies is an unpredictable process that confounds the expectations of even the most informed observers. Thus, we don't need to know precisely how, and certainly not which, new architectures developers might adopt. We require only that deployment be at least remotely possible.

Our deployment strategy leverages the strength of overlays, unconstrained by their previously limited ambitions. In this scenario, a *new-generation service provider* chooses a particular new architecture, then constructs an overlay supporting that architecture. The NGSP then distributes proxy software that lets anyone, anywhere, access its overlay. Those NGSP users not directly connected would still be purchasing Internet service from their ISP, but if the overlay is successful, either the NGSP begins offering direct access to customers or current ISPs, seeing a viable competitive threat, begin to support this new architecture.

Although we call this an overlay, the NGSP could easily support the new architecture natively on most of its network, so only the first-hop access for users not directly connected would use the architecture in overlay mode. Thus, developers could still deploy architectures that promised enhanced QoS this way.

This approach differs little from the normal overlay deployment story, except with regard to the proxy mechanism's non-IP-centric nature. Overlays offer an opportunity to radically change the architecture, not merely provide limited enhancements. A single daring NGSP could accomplish this. It might also arise more naturally, especially when we consider that a long-running experiment on a large, well-maintained virtual testbed constitutes nothing more than an NGSP.

If the architecture in question offers substantial advantages, it will attract an increasing number of users over time. The architecture could gradually and seamlessly migrate from the virtual testbed infrastructure to a more dedicated one, or even remain on a commercial version of a virtual testbed, just as many commercial Web sites reside on Web hosting services. This way, natural market forces could take us gradually into a new architectural world.

However, instead of resulting in a single, radical architectural winner, easing the creation of new overlays could result in a large, ever-changing collection of more narrowly targeted overlays. To avoid architectural chaos and achieve some form of synergy, overlay designers must consider how to bring this union of overlays together to form a coherent framework, thereby becoming more than the sum of their individual functions.

Overlays offer an opportunity to radically change the architecture, not merely provide limited enhancements.

Such joint deliberations on how to achieve synergy among overlays could require a sociological change in research community interaction. When designing a single Internet architecture, we could not afford to ignore each other, since there would be only one place where research advancements could take effect. Overlay deployments can occur independently, without any coordination between or even cognizance of other efforts, yet coordination is required if overlays are to lead to a substantially different future.

VIRTUALIZATION: MEANS OR ENDS

The virtual testbed approach uses virtualization in two crucial ways. First, within its confines, the client proxy coupled with the virtual links between overlay nodes is qualitatively equivalent to a native network. This frees users from the tyranny of their local ISP and network providers no longer need to deploy new functionality at every node. Second, multiplexing overlay nodes creates many virtual testbeds that operate simultaneously, which greatly reduces the barrier to entry for any particular experiment.

Facilitating revolution

Researchers use virtualization techniques for experimentation and perhaps deployment, but these techniques remain independent of the architectures being tested. If architectural changes are rare, with long periods of quiescence or incremental evolution between times of architectural revolution, virtualization simply provides a means to accomplish these architectural shifts.

Given this situation, developers would want every architecture to include the seeds of its own destruction, seamlessly supporting proxy-like functionality and other hooks to make overlay establishment easier, but it isn't necessary for virtualization to be more deeply embedded.

If the Internet is, instead, in a constant state of flux, with new architectures always competing against the old and with many narrowly targeted

The purist/pluralist split is apparent not only when defining an architecture, but also when evaluating it.

architectures existing simultaneously, virtualization can play a more central role. The functionality to support overlays—virtual link establishment and proxy-like reachability—could conceivably become the architecture's core functionality, its narrow waist. In this scenario, PlanetLab would become the new model for the Internet.

Redefining Internet architecture

A change this profound makes us question what we mean by the term architecture. The two extreme points in the spectrum frame this debate. Our diverse experience spans the entire range of this spectrum, so our extreme characterizations are meant not to belittle any opinion but to clarify, if somewhat overstate, the differences.

Internet purists have a monolithic view of architecture centered around a single universal protocol, currently IP, required in each network element and around which all else revolves. They consider overlays blights on the architectural landscape, at best necessary evils reluctantly tolerated. In this view, virtualization provides only a means to install new architectures, not a fundamental aspect of the architecture itself.

Others take a more pluralist approach to architecture, with IP being only one component of an overall system we call the Internet. Overlays offer just one more way to deliver the service users want and are no less appropriate than any other approach to providing functionality. In this view, the dynamic and evolving architecture can, at any point, be defined as the union of the various existing overlays and protocols. The ability to support these multiple coexisting overlays then becomes the architecture's crucial universal piece.

The purist/pluralist split is apparent not only when defining an architecture, but also when evaluating it. Purists aim for architectural flexibility because the architecture will remain in place a long time. Often, however, this flexibility does not result in immediate user benefits. Pluralists, on the other hand, put more emphasis on short-term performance improvements, arguing that the desired flexibility derives from adding or augmenting overlays rather than from the nature of each individual overlay. Since a key challenge for pluralists is providing flexibility at the high speeds enabled by advances in optical networks, a hybrid approach is also possible—a pure architecture for the high-speed core and a more pluralist architecture closer to the edge.

We do not pretend to know which position is right. We anticipate, however, that the virtual testbed will serve as a fertile Petri dish, allowing the development of many different overlays, each with its different characteristics. Perhaps this process will itself to being an experiment from which we can observe either a drive toward uniformity or instead a synergy out of dynamic diversity.

The canonical story about architectural research's potential impact has long maintained that if testbed experiments show an architecture to be promising, ISPs and router vendors might adopt it. This story might have been realistic in the early days of the Internet—certainly DARTnet and other testbeds played an important role in the development of IntServ and Multicast—but it no longer applies. We as a community have long known that any nonincremental architectural change has little chance of adoption.

Further, we are rapidly reaching consensus that traditional testbeds have ceased being an effective way of experimenting with new architectures. Consequently, the research community has greatly narrowed its focus. Most current Internet research involves either empirical measurement studies or incremental modifications that can be deployed without major changes to the architecture.

Although empirical, incremental research plays a valuable role, it cannot meet the broader and more fundamental challenges the Internet faces. By providing easy access to virtual testbeds, we hope to foster a renaissance in applied architectural research that extends beyond incrementally deployable designs. Moreover, by replacing a discredited deployment story with a plausible story closely linked to the experimental methodology, we hope to raise the research community's sights.

We dare not simply complain about our current impasse—we must directly confront and overcome it. ■

Acknowledgments

We thank Guru Parulkar for many helpful comments and conversations regarding this paper. NSF planning grant CNS-0439842 funded this work in part.

REFERENCES

1. D.J. Wetherall, "Active Network Vision and Reality: Lessons from a Capsule-Based System," *Proc. 17th ACM SOSP*, ACM Press, 1999, pp. 64-79.

2. C. Tschudin and R. Gold, "Network Pointers," *Proc. ACM SIGCOMM,* ACM Press, vol. 33. no. 1, 2003, pp. 23-28.
3. T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet Denial-of-Service with Capabilities," *CCR*, vol. 34, no. 1, 2004, pp. 39-44.
4. Ion Stoica et al., "Internet Indirection Infrastructure," *Proc. ACM SIGCOMM*, ACM Press, 2002, pp. 73-86.
5. M. Walfish et al., "Middleboxes No Longer Considered Harmful;" www.pdos.lcs.mit.edu/papers/doa:osdi04/.
6. H. Balakrishnan, et al. "A Layered Naming Architecture for the Internet," *Proc. ACM SIGCOMM*, ACM Press, 2004, pp. 497-506.
7. D.R. Cheriton and M. Gritter, "TRIAD: A New Next-Generation Internet Architecture;" www-dsg/stanford.edu/papers/triad/triad.html.
8. D. Zhu, M. Gritter, and D.R. Cheriton, "Feedback-Based Routing," *CCR*, vol. 33, no. 1, 2003, pp. 71-76.
9. National Science Foundation, Report on NSF Workshop on Network Research Testbeds; http://gaia.cs.umass.edu/testbed_workshop.
10. S. Savage et al., "Detour: Informed Internet Routing and Transport," *IEEE Micro*, Jan./Feb. 1999, pp. 50-59.
11. D.G. Andersen et al., "Resilient Overlay Networks," *Proc. 18th ACM SOSP*, ACM Press, 2001, pp. 131-145.
12. J. Touch and S. Hotz, "The X-Bone," *Proc. 3rd Global Internet Mini-Conference at Globecom*, IEEE Press, 1998, pp. 59-68.
13. L. Peterson et al., "A Blueprint for Introducing Disruptive Technology into the Internet," *Proc. HotNets–I*, ACM Press, 2002, **pp. xx-yy?**.
14. A. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services, *Proc. ACM SIGCOMM,* ACM Press, 2002; citeseer.ist.psu.edu/article/keromytis02sos.html.
15. K. Svetz, N. Randall, and Y. Lepage, *MBone: Multicasting Tomorrow's Internet*, IDG Books, 1996.
16. L. Subramanian et al., "OverQoS: An Overlay-Based Architecture for Enhancing Internet QoS," *CCR*, vol. 33, no. 1, 2003, pp. 11-16.
17. J.D. Touch et al., "A Virtual Internet Architecture," ISI tech. report ISI-TR-2003-570, Mar. 2003.
18. Xilinx, "Virtex-II Pro Platform FPGAs: Introduction and Overview;" www.mangocom.com/xilinx.asp.

Larry Peterson *is professor and chair of computer science at Princeton University. His research interests focus on end-to-end issues in computer networks and systems built around networks. Peterson received a PhD in computer science from Purdue University. Contact him at llp@cs.princeton.edu.*

Thomas Anderson *is a professor in the Department of Computer Science and Engineering at the University of Washington. His research interest is in the practical issues in constructing robust, secure, and efficient computer systems. Anderson received a PhD in Computer Science from the University of Washington. Contact him at tom@cs.washington.edu.*

Scott Shenker *is a professor in the Department of Electrical Engineering and Computer Science at the University of California, Berkeley. His research interests include Internet architecture, distributed systems, and game theory. Shenker received a PhD in physics from the University of Chicago. Contact him at shenker@icsi.berkeley.edu.*

Jonathan Turner *is a professor in the Department of Computer Science and Engineering at Washington University. His research interests include the design and analysis of high-performance routers and switching systems, extensible communication networks, and algorithm analysis. Turner received a* ***<most advanced degree>*** *in* ***<discipline>*** *from* ***<institution>****. Contact him at jon.turner@wustl.edu.*