The End of Internet Architecture

Timothy Roscoe
National ICT Australia, Sydney
Intel Research, Berkeley
ETH, Zürich

1 Introduction

Architecture. There's a lot of it about. Do we need it?

Recent years have seen considerable publishing activity in the area of "internet architecture". This paper steps back and asks a more radical question: is an internet architecture a good thing at all? Are we at a point in the development of distributed communications systems where the concept should be replaced by a different way of dividing up the design space?

This is not a paper about what the Right internetwork architecture should look like, but rather whether the very *idea* of a network architecture at this point in history is a help or hindrance in moving communication technology (and research in particular) forward.

We first examine critically what the role of the internet architecture is today. We argue that instead of acting as useful guide for network practitioners of all kinds (as it has in the past), the principle function the architecture performs these days is to keep the fields of "networking" and "distributed systems" separate, to the detriment of both. Put simply, the internet architecture, and more broadly the concept of a network architecture, is now *in the way*.

At the same time, trends in networking and computational hardware, and in particular in the kinds of testbeds available to researchers to validate their ideas, have made it both feasible and compelling to do research that finesses network architecture as an issue completely, and concentrates on the broader problem of building, deploying, and operating large-scale distributed applications.

Fortunately, this does not render research into "internet architecture" irrelevant, but it does call for a respinning of many of the ideas in a different context. This paper concludes by examining the new research opportunities in the area, and how they relate to tradition challenges in "architecture".

2 What has the architecture of the internet done for us lately?

It is hard to define precisely what the internet architecture is – it is a lot easier to formulate a definition of "the internet" itself, for instance. Some parts of the internet have been more or less specified (for example, protocols like TCP, and SNMP MIBs for standard components). However, unlike other networking technologies, the internet has never had a clear specification of its architecture – indeed, this may have been a prime factor in its success.

That said, some key elements today seem to be in general agreement: datagram-based connectionless service, layering of protocols, a single internet-wide protocol at the network level (the "thin waist"), placement of certain functionality (such as reliable transmission and congestion control) in the end-systems, and locating other functionality (routing, adaptation to heterogeneous networks) in the center of the network [4, 6, 24].

This architecture (if not rigid adherence to it) has had a profound effect on the internet's ability in the past to evolve into possibly the dominant networking technology today. Recently, however, the architectural view has come under increasing strain, as evidenced by deployed network technology, common practices of carriers, and debates in the research community. There is not enough space here to survey the field in detail, but we can divide the pressures on the architecture into three categories.

Pressures from within

The internet architecture is under pressure from within in two forms. The first is from required functionality which the architecture in its current form makes hard to provide, most notably security, resistance to denial-of-service attacks, and end-to-end quality-of-service, though one might also include a sound basis for charging (or, at the very least, cost recovery by ISPs).

The second is from functionality introduced into the network which doesn't fit with the principles of the architecture, such as MPLS, firewalls, network address translators, and other varieties of middlebox [25]. In many cases these developments have been a pragmatic response to requirements for extra functionality, but they invariably have consequences for the structure of the network beyond these basic requirements – for example, firewalls were introduced to provide a scoping function for network accessibility, but have resulted in a network without a systematic way to determine end-to-end connectivity for two hosts.

Pressure from above

A second, equally pragmatic response to network requirements not met by the architecture is to leave the

underlying architecture unchanged and deploy new networks above as overlays – indeed, this mimics the early design of the internet itself, and it is also in these terms that the GENI proposal is sometimes cast [3].

Ironically, by bypassing the architecture, overlays exert pressure on it in turn by making it harder for the underlying networks to perform traffic engineering without employing knowledge about what overlay a packet is part of [21]. This cross-layer peeking is, of course, somewhat contrary to the "thin waist" abstraction of the internet.

It also cuts both ways: there are well-justified calls for information to pass across the waist in the opposite direction to help overlays and other distributed applications (e.g. [17]).

Pressure from without

In addition to looking at overlays and middleboxes, it is perhaps most interesting to ask this question: architecturally, what is the internet's position with regard to other networks? Historically, the position is clear: the internet interacts with other networks by using them to carry IP packets [5]. The "thin waist" of IP makes this assimilation process easy to implement, and users of the other network gain the immediate ability to communicate any other node in the collective internet.

In practice, however, there are plenty of other relationships at work today. The various phone networks (landlines, mobile phones, SMS signalling, etc.) are actually gatewayed to the internet rather than running IP themselves, and this model is increasingly assumed for wireless sensor networks [12]. Even within the IP realm, large enterprise networks constitute significant users of bandwidth, yet do not adhere to the typical architectural principles of the internet (they typically have private address spaces, for instance).

Increasingly, the internet is viewed as one network among several, or many. Discussions of internet architecture rarely mention this shift.

Discussion

Irrespective of its past merits, it is a truism that the current internet does not conform to the traditional architecture, and there is no clear candidate architecture that captures the internet's current form. Furthermore, it is increasingly impossible to ignore the fact that the internet is just one network among many, and its current form will not allow it to encompass them as an overlay (for example, it is infeasible to run IP over sensor networks). Finally, it is unclear that the current facilities offered by the internet are where the action is: innovative communication applications like Akamai and Skype have resorted to overlays to achieve results.

None of this is news to the research community. There have been numerous proposals (indeed, entire workshops such as FDNA) for new internet architectures which address some of the problems of the internet. Given the oft-cited difficulty of evolving the internet in its current state into one with a cleaner architecture, some researchers have taken the sensible move of casting even this evolvability problem as a research challenge, and tackled it [22].

Of course, there are serious methodological problems in doing research in new network architecture. In particular, it is hard to claim success in this area without building a successful followup to the internet.

Recently in the U.S., the GENI project [1] has proposed constructing a testbed whose aims include the deployment and validation of new internet architectures. The philosophy behind GENI is articulated in Anderson et. al. [3], which also lays out two alternatives for a successful outcome of the project. The first, "purist" approach leads to a new network architecture for the next few decades. The second, "pluralist" approach results in several alternative network architectures co-existing.

However, this discussion is based on the unstated assumption that *there should be* a Network Architecture – that there is value in defining (however informally) such an architecture or architectures. The object of the present paper is to examine the opposite view: it is timely and valuable to abandon not simply the current internet architecture, but *the very idea of having one*.

3 ARCHITECTURE: WHY BOTHER?

Why have an architecture? Rather than getting bogged down in definitions ("I can't say what an architecture is, but I know it when I see it"), let's ask: What does an internet architecture hope to achieve? The traditional answers to this question [6] include: interoperability across diverse networks, easier for applications to code to, (recently) a framework for providers to compete, and finally to facilitate innovation. We should ask ourselves: does any internet architecture really address these issues?

These days, the answer seems to be "no". The internet doesn't fully handle interoperability, for example – it does not cover the space of disruption-prone networks [15] and sensor networks [12]. The uniform API of the internet has come to be a handicap to distributed application writers, who cannot exploit useful features of the underlying network, and must perform their own (often expensive) measurements to adapt to changing network conditions [10, 23]. As for competing providers, it has already been recognized that the current internet fails in this regard [7].

Since descriptions of internet architecture either refer to a non-existent present, an idealized past, or a (possibly unrealizable) future, one should ask what the role of internet architecture today is. Put another way, what does the *idea* of a network architecture do? What is the effect

of the concept being in common usage, part of "common sense"?

An alternative (and not incompatible) view of the role of network architecture is that it forms a boundary between, on the one hand "distributed systems" and "applications" research, design, and implementation, and on the other, "networking" (see figure 1). This boundary is real: it is reflected in institutions and practices as varied as large corporations (Microsoft, Google, etc. vs. Cisco, Sprint, etc.) and publishing venues (SOSP, PODC, etc. vs. SIGCOMM, IMC, etc.). Applications run in end-systems. The network carries packets.

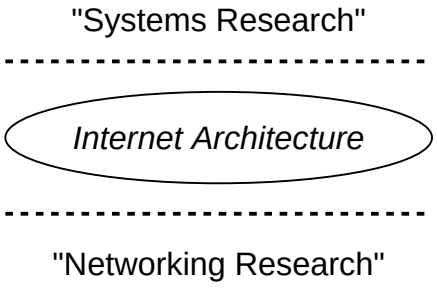

Figure 1: The Internet Architecture as a boundary between disciplines.

Of course this boundary is also rather permeable: a significant minority of researchers publish in both areas, and some venues (such as HotNets) aim at bringing together the two communities. Stated in such stark terms, the idea of the boundary looks odd, but in practice it is remarkably persistent.

For example, notice how this boundary still tends to frame the discussion: the purist vs. pluralist debate above is expressed in terms of "one or several network architectures": the purist approach is to work out what the next architecture should be by trying several out, and then build it. The pluralist approach is that there will be several network architectures in operation, and virtualization provides a way for them to share infrastructure.

To take another related example, compare the original PlanetLab paper [20], published in HotNets-I, with the "Impasse" paper [3], published two years later, in the same workshop (HotNets-III). The PlanetLab paper presents a strict superset of the vision of the Impasse paper, but from a distributed systems context[1].

The Impasse paper presents a more focussed, clearly-defined vision, but one framed entirely in networking terms – "below the line" in figure 1.

In the light of this, it is time to reassess whether the existence of a network architecture is a help or a hindrance to the general field of distributed communications, and whether it fits with the future of actual networking hardware. What would the future look like without a network architecture? What would be in its place?

4 REDEFINING NETWORKING

Modern networking hardware is very different to that available 15 years ago. It is not simply faster: there is an increasing trend toward programmability in network elements, from high-speed forwarding engines to wireless access points and radios. Programmability inevitably leads to the need to support more than one program at the same time, and so network elements increasingly support some form of *virtualization*. This trend has come at the same time as the resurgence of hardware virtualization as a building block in mainstream computing.

Virtualization has been recognized in the networking community as an enabling technology for performing basic research in network architecture. The emerging design of the GENI platform [2] can be viewed as collection of hardware "components" (computational nodes, forwarding engines, programmable radios, optical links, etc.), each of which can be sliced, or shared between different users. It is expected that most of GENI can be constructed with commodity hardware components, but using very different software.

GENI aims to be a testbed for experimenting as widely as possible with different networking technologies. Consequently, it aims (1) to mandate as little as possible about how experimenters will use a particular networking element (e.g. framing, addressing, etc.), and (2) to expose the capabilities of the hardware as much as possible to experimenters (a principle analogous to the argument for Exokernels [11]).

Experimenters are expected to acquire resources (in the form of slices of components) and build ensembles which can execute their systems. At first sight this appears to be a task of daunting complexity given the primitive building blocks available, but GENI is held together by a set of libraries and software management services which collectively enable users to compose these ensembles of components and make this a relatively straightforward process.

An important GENI deliverable is a reference implementation of a network architecture, running purely in a slice, which resembles the current internet in structure and peers with it, as an AS or collection of ASes. It is also recognized that the management services that run GENI require their own control network – initially this will be bootstrapped with an overlay[2] above the current

[1] PlanetLab has not delivered that vision, in part because it is based on deploying overlays. Overlays by themselves are incapable of providing some kinds of functionality not supported by the underlying network, for example QoS [9].

[2] An overlay is required because, ironically, the internet does not provide end-to-end connectivity between any pair of GENI nodes. For example, GENI nodes connected to Internet-2 cannot directly contact those connected to the commercial internet since Internet-2 does not peer with commercial providers.

internet, but this too is expected to move into a slice over time (see figure 2).

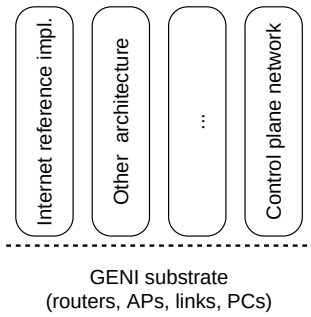

Figure 2: GENI's inversion of architecture and application.

It is a motivating goal of GENI to support research into network architectures, but note that from a broader perspective GENI inverts the traditional layering of applications and networks: the GENI substrate views experiments embodying new network architectures as applications sharing the platform, instead of defining a network architecture as a substrate for applications.

So, suppose one is a researcher who wants to deploy a new network architecture. Presumably this is because it provides some useful functions or supports some useful applications that the current internet cannot. One follows the following procedure:

1. Assemble a slice, that is, a set of virtual servers, routers, links, radios, sensors, etc.

2. Write and deploy the software implementing the network architecture to be used.

3. Write and deploy the software to peer the network with the existing internet in some way.

4. Write and deploy the newly-enabled services and applications.

From an engineering perspective, the last three steps here are all about constructing a software artifact. If the goal is to deploy distributed services that can be accessed remotely, there is no intrinsic reason to divide them up the way shown here. Calling the software in step (2) a network architecture is a rather grandiose name for what is, ultimately, just a few libraries. Carving it off into a separate unit called a network architecture is something only a network architecture researcher would care about.

Of course, we'd like code reuse, and so users deploying distributed systems atop GENI are likely to use libraries written by other parties if they are appropriate to the task at hand. Furthermore, it may make sense for some functionality to be shared between slices in the form of services accessed remotely. Over time, the use of certain libraries and services may come to be common across a wide variety of distributed systems running on the platform.

But these are purely pragmatic considerations. They do not imply anything like a "network architecture", and are unlikely to apply in all cases. Arguably, to impose a common "network architecture" on top of this substrate would be a clear violation of the end-to-end principle: ultimately, it is the application itself that can best decide how to discover, bind to, and use the resources available to it.

In this world, there is no "thin waste" – conceptually applications deal directly with physical resources sliced at as low a level of abstraction as possible, using libraries and services to make the task easier.

This not the same as saying that the GENI substrate and its management services define the "new" internet architecture (in other words, the thing that's common to all users of the hardware), for two reasons. Firstly, inasmuch as there is an architecture here at all, it is dealing with running users' code as much as shipping packets. It is not about creating a fictional boundary between two disciplines, or two types of equipment. Secondly, the structure of GENI (so far) leaves open the question of talking to other networks without mandating any common protocol, leaving the question of end-to-end connectivity an entirely application-defined issue, along the same lines as the Plutarch argument [8].

We note that this does also *not* mean that writing applications becomes much harder. It already requires tens of millions of lines of code to forward a packet from one side of the internet to the other. What changes with the dissolution of the architecture is not the complexity of this functionality, but the context in which it operates. The code now runs in application libraries and services rather than in routers - what has happened is that the total engineering space can now be carved up differently. Consequently, writing internet-like applications is no more complex than before, but writing other applications becomes possible. The substrate is no longer the barrier to innovation it is in the currently internet.

Indeed, some things may become simpler. For example, billing: each service is now using explicit resources rather than the implicit resources used in the Internet. Complex cross-provider bartering based on packet measurement isn't needed at all – if an application is sending traffic on a link, then it presumably has some code running at each end of the link, and hence it is already contracting with whoever operates each end. Carriers are now only selling low-level virtualized resources, and so have a somewhat easier operations task. At the same time, they have more opportunity to differentiate their services by innovating in the hardware they expose to users, where it is placed, and how it can be accessed.

5 REDIRECTING RESEARCH

Is this paper claiming, then, that research into architectures for the future internet and other networks is basically useless? Certainly not. The argument is that recent Internet Architecture research is not without merit, but it is currently misdirected towards the creation of one or more new "network" architectures which retain the outdated distinction between routers and end-systems.

This distinction will become increasingly at odds with reality as the PlanetLab, GENI, and Grid visions of remotely acquirable computation and forwarding resources are realized. A more productive path for the networking research community to pursue is to acknowledge that the boundary between networking and distributed systems (never more than tenuous) is dissolving, and that it is time to rethink where to draw the boundaries.

One approach is to step back and take a fresh, application-centric look. If one has the ability to create virtual machines, virtual routers, and virtual links, remotely, across diverse networks, then how does one write an application to run in this environment? What services, libraries, or other reusable components might such an application find useful?

Here is where most of the good ideas in internet architecture research can find new relevance, but they are likely to be undergo modification in the process. What those modifications are is an exciting direction for future networking and distributed systems research. We list a small selection of areas here; the reader can without doubt identify many more.

Some challenges

Routing as a library: Since applications control their own routing, a potentially rich space of application-specific routing protocols may be opened up. At the same time, many applications can of course benefit from sharing routing information and route computations. Each application is effectively setting up its own network (almost an overlay, though directly using sliced hardware rather than an existing network). There has been relatively little work in the internet arena on simultaneous routing on many overlapping graphs.

Discovery: how do applications discover and bind to a set of resources (links, routers, servers, devices)? This set is necessarily dynamic: resource availability will change due to failures, recovery, and upgrades and we can safely assume that applications will be written to adapt to changing load as well.

Unlike in traditional networking, discovery is clearly intimately tied to routing. Indeed, routing for an application might be cast as a continuous problem of discovering and acquiring the optimal set of network resources (where "optimal" is defined as some application-specific function of utility and cost).

This author has a fondness for a declarative query language approach to addressing this problem, since it neatly fuses routing and discovery [18, 19] and multi-query optimization techniques can be applied to sharing computation, but there are undoubtedly other approaches to the problem worthy of investigation.

Composition and federation: In the limit, as applications build their own networks for internal communication, how will they talk to each other? How will traffic be routed from an end system to a collection of different applications? This problem is somewhat analogous to the current problem of peering of ASes in the internet, except with a higher degree of heterogeneity to deal with, and correspondingly more freedom in implementation.

An interesting open question is whether the principal challenges in peering become harder or easier when carried out at the application layer, but note that access solutions at least can more or less directly apply techniques in systems like OCALA [16] and OASIS [13].

Operations: A final set of challenges that spring to mind with the vision of communications infrastructure in section 4 is how operators will manage the substrate. This is a worthy research challenge and is being actively pursued, but the issues are less central to the focus of this paper because they are more about individual pieces of hardware, and the distributed systems technology to remotely manage them, than about concerns which map more closely to traditional networking concerns.

Reconciling networking and distributed systems

Many of the new challenges are familiar from the field of distributed systems, but recast in such a way that they reach further down into the traditional networking stack.

In fact, the central argument in this paper has a parallel in the field of distributed systems. Traditional distributed systems research (DHTs being one good example) has tended to view the network as a black box – in particular, the network is assumed to provide connectivity between any pair of end-points [14], and quantitative differences in connectivity (bandwidth, latency, etc.) are to be recovered by the application through measurement.

6 CONCLUSION

The idea of having an architecture for a network – of carving up the space into a network and end-systems which use it – has been tremendously useful in advancing the state of the art in communications technology. However, the success of the internet has eventually resulted in this being an obstacle to radical innovation in the networking space. It is not that the architecture itself must

be fixed, but the idea itself of having a network architecture is now in the way.

Dissolving the category of network architecture allows us to move forward with the more basic problem of how to build and peer distributed applications, particularly in a future of mobile devices, sensors, smart objects, and the like.

In time, a new and useful consensus about how to build distributed communication systems may emerge, and it might then be termed an "architecture", though not necessarily of a network. Until then, we can make more progress by removing the blinkers imposed by the outdated idea of a network architecture.

7 ACKNOWLEDGEMENTS

The ideas in this paper have developed from numerous discussions on network architecture, in particular with Dave Clark, Jon Crowcroft, Kevin Fall, Steve Hand, Richard Mortier, Larry Peterson, Sylvia Ratnasamy, Andy Warfield, and John Wroclawski. I am particularly indebted to Tom Anderson for sparking the initial idea, and Scott Shenker for the invitation to present some of these concepts as a guest speaker in his Internet Architecture course at U.C. Berkeley.

REFERENCES

[1] GENI: Global Environment for Network Innovations. `http://www.geni.net/`, August 2006.

[2] T. Anderson, D. Blumenthal, D. Casey, D. Clark, D. Estrin, L. Peterson, D. Raychaudhuri, J. Rexford, S. Shenker, and J. Wroclawski. GENI: Conceptual Design Project Execution Plan. GENI Design Document GDD-06-07, January 2006. `http://www.geni.net/GDD/GDD-06-07.pdf`.

[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet Impasse through Virtualization. *Computer*, 38(4):34–41, 2005.

[4] B. Carpenter. Architectural Principles of the Internet. Internet RFC 1958, `http://www.ietf.org/rfc/rfc1958.txt`, June 1996.

[5] V. Cerf. The Catenet Model for Internetworking. Internet Experiment Note 48, `http://www.isi.edu/in-notes/ien/ien48.txt`, July 1978.

[6] D. Clark. The design philosophy of the DARPA internet protocols. In *Proc. ACM SIGCOMM 1988*, pages 106–114, New York, NY, USA, 1988. ACM Press.

[7] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's internet. In *Proc. ACM SIGCOMM 2002*, pages 347–356, New York, NY, USA, 2002. ACM Press.

[8] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Plutarch: An argument for network pluralism. In *Proceedings of SIGCOMM Workshop on Future Directions in Network Architecture (FDNA'03)*, pages 258–266, August 2003.

[9] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Qos's downfall: At the bottom or not at all! In *Proceedings of SIGCOMM Workshop on Revisiting IP QoS (RIPQOS'03)*, August 2003.

[10] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, pages 50–58, September/October 2002.

[11] D. R. Engler and M. F. Kaashoek. Exterminate all operating system abstractions. In *Proceedings of the 5th Workshop on Hot Topics in Operating Systems (HotOS-V)*, pages 78–83, Orcas Island, Washington, May 1995. IEEE Computer Society.

[12] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99)*, Seattle, Washington, August 1999.

[13] M. J. Freedman, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for Any Service. In *Proc. 3rd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '06)*, San Jose, CA, May 2006.

[14] M. J. Freedman, K. Lakshminarayanan, S. Rhea, and I. Stoica. Non-Transitive Connectivity and DHTs. In *Proceedings of USENIX WORLDS*, December 2005.

[15] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. ACM SIGCOMM 2004*, pages 145–158, New York, NY, USA, 2004. ACM Press.

[16] D. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, and K. Wehrle. OCALA: An Architecture for Supporting Legacy Applications over Overlays. In *Proc. 3rd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '06)*, San Jose, CA, May 2006.

[17] R. R. Kompella, A. Greenberg, J. Rexford, A. C. Snoeren, , and J. Yates. Cross-layer visibility as a service. In *Proceedings of HotNets-IV*, November 2005.

[18] B. T. Loo, T. Condie, J. M. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica. Implementing Declarative Overlays. In *20th ACM Symposium on Operating Systems Principles (SOSP)*, 2005.

[19] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *Proc. ACM SIGCOMM 2005*, 2005.

[20] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. HotNets-I*, 2002.

[21] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *Proc. ACM SIGCOMM 2003*, pages 151–162, New York, NY, USA, 2003. ACM Press.

[22] S. Ratnasamy, S. Shenker, and S. McCanne. Towards an evolvable internet architecture. In *Proc. ACM SIGCOMM 2005*, pages 313–324, New York, NY, USA, 2005. ACM Press.

[23] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. Opendht: A public dht service and its uses. In *Proc. ACM SIGCOMM 2005*, August 2005.

[24] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[25] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes No Longer Considered Harmful. In *Proc. 6th Usenix OSDI*, San Francisco, CA, December 2004.