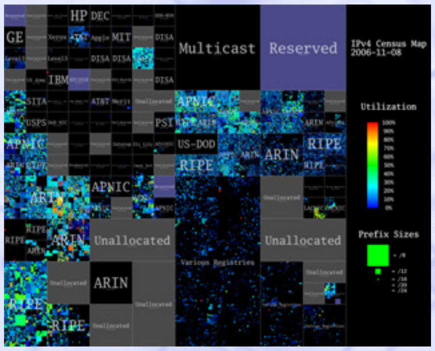
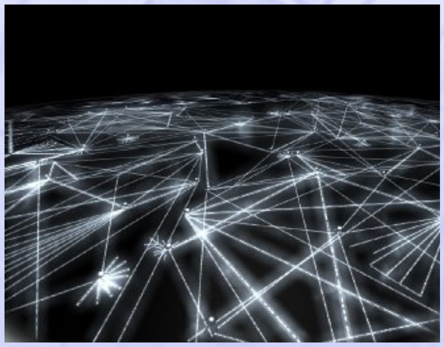
Designed for Change:

End-to-End Arguments, Internet Innovation, and the Net Neutrality Debate





Richard Bennett

Information Technology and Innovation Foundation

Designed for Change:

End-to-End Arguments,
Internet Innovation, and the Net Neutrality Debate

Richard Bennett

ITIF
The Information Technology
& Innovation Foundation

September 2009

Table of Contents

I. Executive Summary 1

II. Introduction 6

End-to-End in Network Engineering 7

A. ARPANET 7

B. CYCLADES 9

1. Flexibility 9
2. Simplicity 10
3. Multiple Gates 10
4. The Influence of CYCLADES on Subsequent Networks 11

C. The Connectionless vs. Connection-Oriented Debate 12

D. Moving From Practice to Theory 12

E. Single Points of Failure 14

F. Internet Congestion Collapse 15

The Rehabilitation of the End-to-End Arguments 18

A. End-to-End as Limited Fate-Sharing 18

B. Who Do You Trust? 19

C. Layering Considered Harmful 20

D. The Evolution of End-to-End 20

E. Clark's Tussle 21

F. Current Status of End-to-End 22

End-to-End in Regulation 23

A. From the Fringe to the Mainstream 26

B. Network Neutrality: The Birth of a Notion 28

C. Functional Layering as a Network Regulatory Model 30

Birth of the Public Internet 31
A. The End-to-End Tax: “World Wide Wait” 33
B. Internet Innovation Cycle 34
Challenges to the Internet Architecture 34
A. Denial of Service Attacks 34
B. Traffic Jams 34
C. Address Unknown 35
D. The Proliferation of Prefixes 35
E. LTE Needs Quality of Service for Voice 36
F. InterStream Needs Quality of Service for Video 36
G. RINA Rethinks Protocol Layering 36
Lessons from End-to-End’s Evolution 37
A. In Search of a Regulatory Framework 38
B. Striking the Balance 39
Conclusion 39
Endnotes 41

List of Figures

Figure 1: ARPANET Architecture 8
Figure 2: CYCLADES Architecture 9
Figure 3: Internet Architecture 11
Figure 4: Equal Pipes Congestion Scenario 17
Figure 5: Creating Congestion by Over-Provisioning. 25

Executive Summary

The principle of constant change is perhaps the only principle of the Internet that should survive indefinitely.
—RFC 1958

Advocates of net neutrality stress the fact that the Internet's free and open character stimulates innovation. This is undeniably true, of course: The fact that practically anyone with an idea can write an Internet program, test it, and deploy it on a standard, world-wide platform at minimal expense is an essential part of the Internet's appeal and the major generator of novel, useful and entertaining applications.

Some neutralists argue that the Internet's open character is the consequence of a single design decision made more than 35 years ago involving the end-to-end principle, which in their view forbids network operators from actively managing network traffic, even to improve the consumer experience. What this report shows, however, is that while end-to-end arguments reflect important aspects of the Internet's organization, the belief that the Internet is built on negative principles—prohibitions on taking action—is a serious misunderstanding of how the Internet was designed and how it works.

The Internet's capacity to stimulate innovative new applications flows from its ability to meet the needs of applications previously unimagined, not from deliberate impairment of the network core. No single-service network is neutral, as any given network service has greater utility for some types of applications than others. Tim Wu, the man who coined the term "network neutrality," concedes this point:[1]

> Proponents of open access have generally overlooked the fact that, to the extent an open access rule inhibits vertical relationships, it can help maintain the Internet's greatest deviation from network neutrality. That deviation is favoritism of data applications, as a class, over latency-sensitive applications involving voice or video.

As new generations of application designers imagine novel ways to employ networks, traditional paradigms of network usage can present obstacles—discrimination by design—that can only be overcome by improvements to the network in the form of service model modifications appropriate to new requirements. The Internet was designed for change: The real power of the end-to-end architecture lies in its ability to stimulate innovation by providing service model enhancements that in turn enable new types of applications at the network edge. Modern networks are no longer constrained to provide one and only one delivery service; they're now capable of providing multiple services, each tailored to a broad class of applications, and allowing application developers to choose the one that best meets their needs.

The Internet's flexibility comes from the decision to use a novel method of information transfer called the "datagram" instead of a traditional telecom control system. Datagram networks such as the Internet export a significant portion of the network's control logic to end-user systems located on the network edge, where it's easy to experiment with policies and protocols simply by writing new software; they are therefore termed "end-to-end" networks. End-to-end arguments describe the structure of datagram networks and provide a blueprint for their continual improvement.

Since the dawn of the datagram era, experimentation has identified communication functions that provide greatest utility as core network services. The Internet is an adaptable system because its experimental character provides it with a built-in learning function that enables engineers to make network and protocol design decisions empirically. It's fair to say that its design is dictated more by a commitment to continual improvement than by obedience to hard and fast rules: It was meant to be a system in which experiments would have the consequence of improving the network. This method of designing networks was pioneered by CYCLADES, the datagram network created by Louis Pouzin in France in the early 1970s, and was subsequently adopted by the Internet and several other networks. The experimental approach to network design and operation is challenged by the modern-day network neutrality framework, which relies more on a rigid system of rules to make decisions about the placement of network functions than on experience and empirical evidence.

In Pouzin's formulation, network engineering was a process consisting of design and deployment followed by either standardization or redesign. He structured the CYCLADES network the way he did—with datagrams at the network layer and virtual circuits in the endpoints—on the belief that this division of labor would allow the greatest degree of freedom to develop network protocols and services, not just applications.

The commercial Internet has developed as a loose federation of separately owned and operated networks that have agreed to interconnect at a few hundred peering points around the world on terms mutually agreed by each pair of networks. Internet applications running on end systems rely on this infrastructure to communicate with each other. As the underlying technology improves, interconnection agreements occasionally change as well; when such changes increase overall network capability, users and application providers stand to benefit. Steady improvement, punctuated by brief periods of radical disruption, is the normal state of affairs in evolving technical systems and should not be feared. It is likely that the only principle of the Internet that should survive indefinitely is the principle of constant change.[2]

To understand the current debate over Internet regulation, it's necessary to appreciate that the end-to-end arguments of network engineering differ significantly from network neutrality advocates' idiosyncratic end-to-end principle, a demand for a low-function, "stupid" network. A review of the Internet's history can help us understand the difference.

The design of the Internet, as is frequently the case in network engineering, was a critique of a predecessor network—in this case, ARPANET, a packet-switching network developed by BBN, a contractor to the Advanced Research Projects Agency (ARPA) of the U.S. Department of Defense in the late 1960s. ARPANET was a highly functional system that proved the viability of packet-switching, but it wasn't friendly to research on network protocols. The Internet, designed in the mid-70s, aimed to overcome ARPANET's barriers to research; it was meant to serve as a testbed for research on network protocols, network applications, and distributed systems generally.

Academics in the network engineering field developed end-to-end arguments in order to communicate the Internet's design principles to the larger world nearly 10 years after the basic design of the Internet protocols was developed.[3] Early end-to-end arguments pertained to the optimal placement of functions in distributed systems for scalability, simplicity, and performance, with an eye toward an innovation cycle. End-to-end arguments represent a kind of "rebuttable presumption" that new network services should be provided in end systems unless a compelling reason can be found for implementing them in the network infrastructure. The nature of the exceptions to end system placement, the grounds for rebuttal, has changed significantly over the years, as has the Internet itself.

The early Internet was a walled garden, accessible only by universities and other research establishments—an open network for a closed community. Early end-to-end arguments, therefore, did not emphasize the issues of trust and network stability that arise on general-use networks in which network access is not restricted to a close-knit community of skilled users and network operations are not constantly monitored by highly qualified administrators at every site. The current Internet has to police itself, so it's inevitable that it should operate differently than its coddled ancestor. Not only does today's Internet serve a more diverse user community, it has to support a much more diverse group of applications. The conflict between the old and the new is the basis of the net neutrality debate.

Over the years, the network engineering community has debated and refined its end-to-end arguments to keep them relevant to the changing context of the Internet's social and technical dynamics. Efforts to redefine and clarify the rationale for the end-to-end arguments have become a mainstay of the ongoing discussion of Internet architecture within the Internet Engineering Task Force (IETF)—the international community of network designers, operators, vendors, and researchers that is responsible for creating Internet standards. The rationale for end-to-end provides guidance about which exceptions are acceptable. As the discussion that follows suggests, these arguments are not only very difficult for the lay audience to follow, they're extremely vulnerable to misapplication.

Between the development of end-to-end arguments in network engineering and their discovery by the network policy community, a seismic shift took place around the Internet. The system that had been designed to support a well-mannered community of academics was opened to commercial use and all the attendant consequences. The public Internet unleashed an unparalleled wave of creativity and innovation, including commerce, the World Wide Web, Voice over Internet Protocol (VoIP) and video conferencing, audio and video streaming, Napster and BitTorrent, online news and news archives, blogs and personal web sites, centers for organizing social and political groups, and social networks. At the same time, the public Internet also gave rise to malicious and criminal uses, such as worms, viruses, botnets, theft, spam, and online scams. Internet service providers (ISPs), telecom and cable providers, and Internet Protocol (IP) network providers invested hundreds of billions of dollars to make the public Internet faster, cheaper, more reliable, and more pervasive, ranging all the way from undersea optical cables and residential broadband networks to sophisticated systems to identify and mitigate attacks by automated means.

Internet policy arose as an academic exercise in the late 1990s, long after the research Internet that formed the original context for the end-to-end arguments had given way to a Wild West show. Policy analysts studied the academic literature on Internet design and struggled to connect theory with current practice. Lawrence Lessig, the author of *Code and Other Laws of Cyberspace*,[4] for example, began his study of the Internet in the late 1990s—a milieu very different from the one that had produced the paper Lessig regarded as the seminal statement of Internet architecture, "End-to-End Arguments in System Design," in 1981.[5] The transition of the Internet from the highly restricted academic environment to the open, commercial setting was also unsettling to academic users who had come to regard the absence of commercial transactions as an essential feature of their 'Net.'[6]

As a consequence of the Internet's commercialization, major technical changes took place in the Internet ecosystem—the advent of the Border Gateway Protocol (BGP) and the rise of residential broadband. BGP made it possible for networks that didn't trust each other to exchange routing information but at the expense of Quality of Service (QoS) information. Residential broadband systems deployed technologies originally developed for Video-on-Demand to bring about data transfer speeds 30 to 100 times faster than dial-up modem speeds. The demand for broadband was caused by acceptance of the Web once graphical browsers were made available for standard operating systems.[7] A new generation of Internet routers, the Cisco 7500 series, had to deal with these new realities as well. The Internet population exploded faster in the late 1990s than it ever had before.

Lessig and those he has influenced are extremely pessimistic about the commercialization-related changes to the Internet, which they see as curtailing freedom and erecting barriers to innovation. One leading acolyte, Free Press, has regularly declared, for the last five years, that the end was near for the Internet's salubrious effects on democracy and technology:[8]

> We can't allow the information superhighway to become the phone and cable companies' private toll road. If they get their way, the Internet as we know it—as a democratic platform for free speech and innovation—will cease to exist.

From political blogs to online newspapers to social networks and online repositories of real-time legislative information, there's no doubt that Internet-enabled systems have insinuated themselves into every aspect of politics and government, for good and for ill.

Unfortunately, however, network neutrality advocates err when they suppose that the power of the Internet to extend political participation depends on the relatively narrow elements of system design bracketed within the scope of the end-to-end arguments. Network neutrality advocates have apparently reached deep inside the Internet's technical structure less to understand it than to find the authority to regulate it. Compounding the error, they did so at a time when the structure of the network was radically changing to meet new demands. They've also failed to stay up-to-date with the engineering community's ongoing discussions about Internet architecture, and have consistently asked regulators to require network operators to employ engineering solutions within the Internet that are more appropriate to the traditional, single-function telephone network, such as over-provisioning. The legitimate concerns of network neutrality take place at the network edge, where the network interacts with the user; what goes on inside the network is largely beside the point, and has typically been misstated by network neutrality advocates in any case. The Internet is not a "level playing field" in which each packet gets equal treatment; the design of the Internet, the facilities that users purchase, and the location of servers all cause varying degrees of inequality. The Internet also discriminates by design for and against various uses; structural discrimination can only be mitigated by active management within the network.

It's more productive to make a diligent effort to understand the Internet's dynamics, its structure, the challenges it faces, and the tradeoffs that circumscribe the work of network engineers *before* trying to constrain the Internet's ever-changing nature. If we do this, we can avoid creating a program of regulation that's more likely to retard genuine innovation than to nurture it.

Saying this, however, is not saying that engineers are in happy agreement with each other about how to design large-scale networks. They certainly are not, and they often disagree with each other quite passionately in a vast number of worldwide forums for academic and standards discourse. Saying this is also not saying that issues of Internet regulation all come from the failure to understand network engineering. Many regulatory debates concern issues that engineers are ill equipped to resolve, such as the balance of public and private participation in infrastructure development, conflicting rights of use, access, and ownership, and constraints on human behavior that are largely independent of the means by which they take place: you don't need an engineer to tell you that an electronically delivered murder threat is a bad thing.

Applied blindly, end-to-end can become a dogma that limits network efficiency, increases costs, and constrains opportunities to innovate. The more information a network component has about the overall state of the network, the sooner it can react to changing conditions and the more effectively it can heal network failures. In some instances, the insistence on endpoint-heavy infrastructure and limited commercial relationships between users and network providers limits the horizon for new applications. This is especially true to the extent that new applications use the Internet in non-traditional ways. Applications that require extremely low latency and applications that transfer extremely large amounts of data are both penalized by network neutrality advocates' insistence that their end-to-end principle mandates a single delivery service for all packets, for example.[9]

Interconnected networks are powerful systems, but they can only achieve their full potential when rules of interconnection don't force loss of capability as a matter of technical or regulatory fact. It's likely, though far from inevitable, that the specialized networks custom-built for television and telephony will merge with the

Internet on a common, converged IP infrastructure. Convergence can happen only to the extent that the common infrastructure is capable of meeting the specific requirements of telephony and television for service and cost.

The Internet of the Future will embed a great deal more traffic engineering capability into the infrastructure than the research Internet did. This development need not be cause for alarm as long as access to new capabilities inside the network is reasonable, pro-competitive, and conducive to innovation in the application space.

As the evolution continues, we will come to understand that the ultimate value of end-to-end arguments is the caution they provide against migrating functions from network endpoints to the network middle without empirical justification. We should understand end-to-end arguments as laying out a design and implementation process, not a wholesale ban on network-based acceleration. This understanding of the end-to-end arguments brings them in line with the current state of the Internet—and is actually consistent with the original formulations of the datagram architecture that's found its fullest expression in the Internet.

Designed to Change: End-to-End Arguments, Internet Innovation, and the Net Neutrality Debate

As the Internet has moved from the research setting into the general-use community and computer and communications technology has continued to improve, its shortcomings have become apparent.

End-to-end arguments play an enormous role in the engineering literature on Internet architecture, and references to an end-to-end principle were very common in the formative works in the emerging academic discipline around Internet policy known as Cyberlaw. In many instances, we find the assertion that *end-to-end* notions explain the Internet's essential nature, and in the case of Cyberlaw, a further claim that a regulatory framework is actually embedded in the Internet's structure.[10]

In reality, the end-to-end *arguments* of network engineers and the end-to-end principle of Internet policy advocates have significantly different implications. Advocates of network openness routinely confuse the end-to-end principle with the ability to support diverse network applications, while engineers see end-to-end arguments as advice about function placement in a modular system.[11] Whether mentioned explicitly or not, assumptions about the meaning of end-to-end shape the net neutrality argument. A close examination of the Internet's history and architecture shows that the end-to-end principle of the policy advocates is often reductive, incomplete, or even harmful.

In engineering, end-to-end arguments are part of an overall implementation strategy for distributed systems and network protocols as well as an abstract principle of network architecture. The designers of the early research networks—CYCLADES, the Internet, Digital Equipment Corporation's DECNet, and the Xerox Network System (XNS)—chose to implement virtual circuit transport protocols in end systems and datagrams in the network for reasons related to their network protocol research programs and the limitations of the computer systems of their era, the 1970s. These early research networks were used by people with extraordinary technical skills and administered by specialists, so they weren't troubled by the security and management problems inherent in purely end-to-end networks.

As the Internet has moved from the research setting into the general-use community and computer and communications technology has continued to improve, however, its shortcomings have become apparent. The contemporary Internet doesn't respond linearly to increased demand or deployment of bandwidth; is highly vulnerable to attack, abuse, and misuse; demands too much time and skill to administer; is not scalable; and is effectively impossible to upgrade without exceptional effort. For reasons very loosely related to the end-to-end strategy, the contemporary Internet also fails to support the increasingly diverse mix of applications very well, and therefore fails to serve the needs of innovators as well as alternative architectures and implementations might.

Network neutrality advocates have unwittingly attempted to shackle network engineers and operators to a network implementation strategy that is incomplete at best, essentially seeking to commit a mistaken analysis of the network's structure to law. Rather than going down this unproductive road, regulators should actively encourage network engineers and operators to experiment with new models, services, and technologies to better meet social needs and expectations in the present and the future. A proper regulatory framework for the Internet has to take at least one step back from the features bracketed by end-to-end: rather than specifying network implementation, it would emphasize business practices. It would speak to disclosure, services, and economic analysis rather than demand a blind obedience to a narrow technology model. The correct framework would also be substantially technology-independent, applying equally well to both wireless and wireline networks. End-to-end should be seen as a license to innovate, not as a straightjacket.

END-TO-END IN NETWORK ENGINEERING

In order to appreciate the role of end-to-end in the Internet, it is useful to see when and why this notion was introduced into packet data networking. Two significant packet networks preceded the TCP/IP Internet: ARPANET and CYCLADES. The designers of the Internet borrowed heavily from these systems, especially CYCLADES, so it's instructive to understand how they worked and what lessons the Internet learned from them.

ARPANET was the first open-access packet network built for the research community, but if didn't employ an end-to-end architecture. CYCLADES, designed by Louis Pouzin, pioneered the use of the datagram, functional layering, and the end-to-end strategy adopted by many subsequent packet networks, including the Internet. In Pouzin's formulation, network engineering was a process of ongoing experimentation consisting of design and deployment followed by either standardization or redesign. In designing CYCLADES, Pouzin created a sandbox for experimentation on network protocols.

ARPANET

ARPANET was developed by ARPA contractor BBN in the late 1960s. It was comprised of a series of variations on a host-to-host protocol called "Network Control Program" (NCP) and a packet-switched subnet. This system was not end-to-end as we understand the term today. ARPANET consisted of specialized packet-switching nodes, called Interface Message Processors (IMPs) that performed network functions for application programs running in attached computers. The IMP was a general purpose Computer Control Company[12] DDP-516 minicomputer with 64K bytes of memory,[13] optimized I/O capability, and a customized interface attaching to a host computer in much the same way that disk drives of the era attached to their hosts.

ARPANET—the first open-access packet network built for the research community—did not employ an end-to-end network control strategy.

The ARPANET platform was excellent for experimentation with applications, as it off-loaded the entire networking burden to a separate system. It was also an "open-access" system in the sense that applications of various types could be implemented and tested on the ARPANET without destabilizing the network, but it wasn't an end-to-end system as network protocols were confined to the IMP and outside the control of users, even skilled ones. The design of ARPANET was suitable for application development, but it was frustrating for researchers who wished to experiment with network protocols, and for good reason: ARPANET was built to allow time-shared access to expensive computers and to prove the viability of packet-switching, not to stimulate network research on the finer points of network design. ARPANET planner and system architect Larry Roberts explained his design rationale before the system was built:[14]

One way to make the implementation of a network between a set of time-shared computers more straightforward and unified is to build a message switching network with digital interfaces at each node. This would imply that a small computer, in interface message processor (IMP), would be located with each main computer to handle a communications interface. It would perform the functions of dial up, error checking, retransmission, routing, and verification. Thus the set of IMPs, plus the telephone lines and data sets would constitute a message switching network.

The major advantage of this plan is that a unified, straightforward design of the network can be made and implemented without undue consideration of the main computer's buffer space, interrupt speed and other machine requirements. The interface to each computer would be a much simpler digital connection with an additional flexibility provided by programming the IMP. The network section of the IMP's program would be completely standardized and provide guaranteed buffer space and uniform characteristics, thus the entire planning job is substantially simplified. The data sets and transmission lines utilized between the IMPs would most likely be standardized upon, but as changes occurred in the communication tariffs or data rates, it would be more straightforward just to modify the program for the IMPs rather than twenty different computers. As soon as the need became apparent, additional small computers could be located at strategic interconnection points within the network to concentrate messages over cross-country lines. Finally, the modifications required to currently operating systems would be substantially less utilizing these small computers since there would be no requirement to find buffer space, hold messages for retransmission, verify reception of messages and dial up phone lines.

While ARPANET's design was elegant, research users were stymied as well as assisted by the IMP. Like TCP, NCP was a virtual circuit protocol that made connections between processes and controlled the rate at which packets were passed from Host to Host, but it didn't really control the network. Unlike TCP, which employs a windowing technique inherited from CYCLADES, NCP employed a "stop-and-wait protocol" that permitted one "message" per virtual circuit at a time. This design eliminated congestion concerns, but made bulk file transfers very slow owing to the high delays associated with packets moving through several hops from source to destination. These delays were acceptable since ARPANET was mainly used for time-sharing.

There was a certain degree of friction between ARPANET users and the contract engineering firm that

Figure 1: ARPANET Architecture

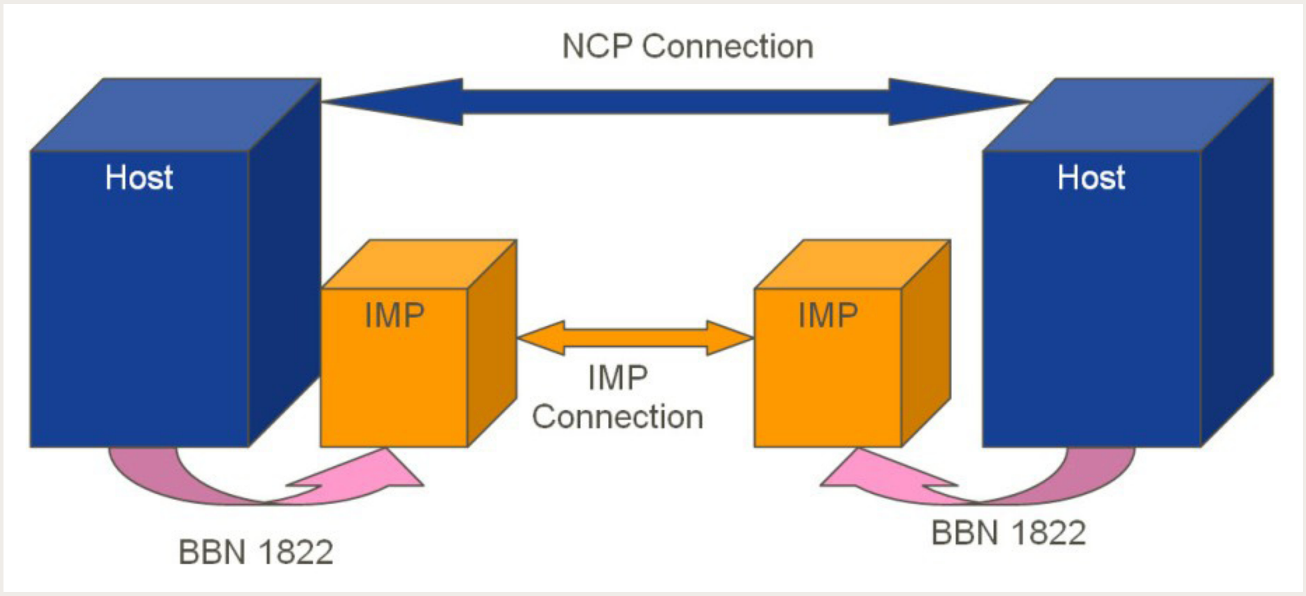
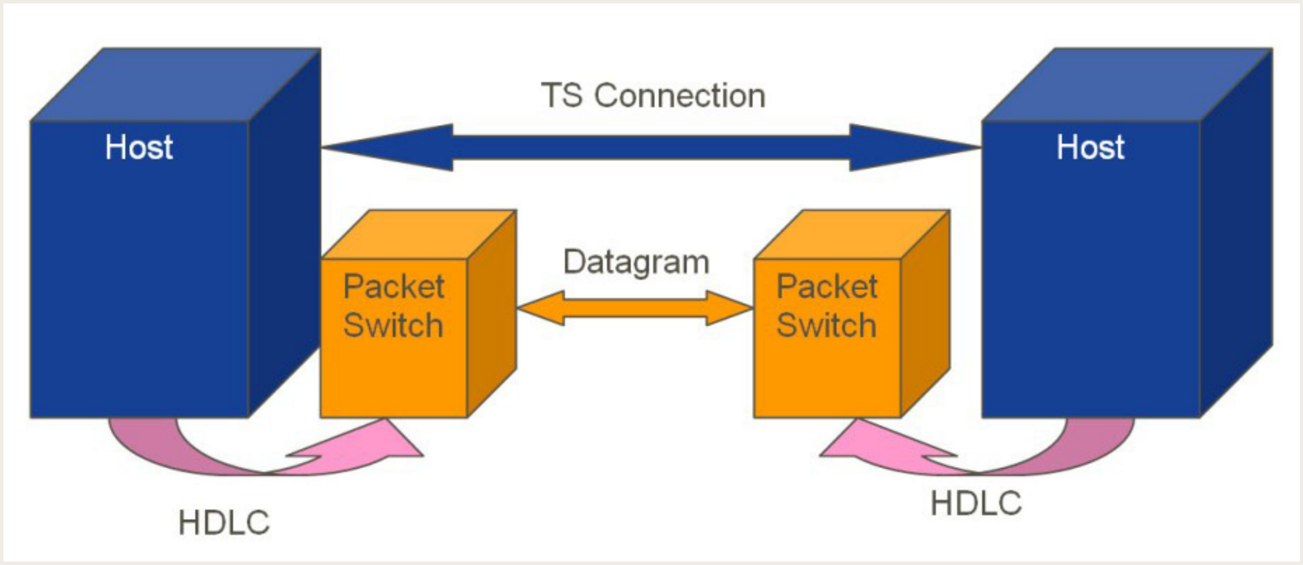


Figure 2: CYCLADES Architecture



built the IMP, BBN.[15] The IMP was essentially a black box from the standpoint of end users, whose host systems simply connected through a specialized I/O interface that had to be custom-built for each type of host. BBN even discouraged host programmers from checking data packets for integrity as they passed between IMP and host—a point that was disputed in the very first Request for Comments (RFC) published in April 1969.[16]

This black-box system had limited utility for network research. Users were free to develop application programs and to experiment with NCP, but they were constrained with respect to the networking functions performed by the IMP. They had no control over the methods used inside the network to route, fragment, de-fragment, serialize, prioritize, manage congestion, or control rates of data flow. ARPANET proved the feasibility of packet-switching for interactive applications, but the specifics of managing diverse traffic streams with varying duty cycles under different network load scenarios remained opaque; in fact, they remain significant network engineering research topics to this day.

Enclosing network protocols in the IMP also prevented university researchers from developing and measuring alternatives to the BBN implementation; ARPANET underscored the need for a networking research system.

CYCLADES

The first end-to-end research network was CYCLADES, designed by Louis Pouzin at IRIA in France with the support of BBN's Dave Walden and Alex McKenzie and deployed beginning in 1972.[17] CYCLADES employed a connectionless, unreliable datagram service at the network layer, adding reliability in the path between one end system and the other through a virtual circuit transport protocol implemented in host computers, as TCP would. CYCLADES thus made a clean separation of the datagram routing system from the virtual circuit mechanism, presaging the division of IP from TCP. CYCLADES pioneered the datagram, functional layering, and the end-to-end strategy adopted by subsequent packet networks, including the Internet, the Xerox Network System (XNS), Digital Equipment Corporation's DECNet,[18] and the Open Systems Interconnection (OSI) system defined by the International Organization for Standardization (ISO).

Flexibility

CYCLADES was a much more flexible system than ARPANET, as befits a research network. The CIGALE packet switch performed only the most elementary task, and left the rest to software running on the general purpose host systems. It gave host programmers a free hand to build network protocols as they saw fit. This was a very deliberate choice:[19]

> …the design effort bears on a carefully layered architecture, providing for an extensible structure of protocols and network services, tailored to various classes of traffic and applications.
>
> This concern for built-in evolutionism translates itself in putting as few features as possible at levels buried in the sensitive parts of the network.

The CYCLADES system was also much more economical than ARPANET, as the CIGALE packet switch was less expensive to build than the more heavily burdened IMP.

Simplicity

Another reason to employ the end-to-end strategy in CYCLADES was to achieve implementation simplicity. CYCLADES was a network built almost entirely out of general-purpose computers rather than the specialized switching equipment used by the telephone network. These computers were extremely limited with respect to processor speed and memory (the Blackberry 9000 has more than 2,000 times the processor speed and memory of the CII 10-070 computer initially used as a CYCLADES host computer),[20] and it was therefore essential for the system to be engineered in such a way as to avoid duplication of functions. Consequently, the primary design principle in early internetworking protocols was simplicity rather than separation of functions, smartness at the ends, or any other single thing. (The desire for simplicity in the Internet is reflected in the naming of protocols such as "Simple Mail Transfer Protocol" and "Simple Network Management Protocol.")

CYCLADES implemented its endpoint Transfer Station (TS) protocol in general-purpose host computers, but its packet switching protocol (CIGALE) was implemented in Mitra-15 minicomputers with only 32K bytes of memory,[21] half the memory of the ARPANET IMP. The IMP dealt with virtual circuit creation and maintenance, congestion management, routing, error checking, retransmission, and serialization, but CIGALE simply had to route packets, and it had just enough compute power to do so.

In this era, simplicity was achieved by avoiding duplication; it was believed that basic functions, such as error correction, could be done once and only once if they were located at the proper point in the network system. In the case of ARPANET, this function was performed by the IMP, although this choice was a controversial decision. In TCP as in CYCLADES, error correction was performed in host software. In the absence of a simplicity mandate, better performance can be obtained by doing it in both places; when TCP runs over Wi-Fi, this is exactly what happens, for example. But if the system has only enough power to do error correction once, it had better be done as close to the edge as possible: The larger the scope of the function, the greater the number of errors the system can detect and correct.

The use of end-to-end protocols in CYCLADES and TCP was dictated by the engineering problems designers set out to solve.

Multiple Gates

The structure of the CYCLADES implementation was also motivated by the desire to experiment with network protocols, especially a mode in which end systems were multiply-connected through a number of dissimilar networks:[22]

> CYCLADES uses a packet-switching sub-network, which is a transparent message carrier, completely independent of host-host conventions. While in many ways similar to ARPANET, it presents some distinctive differences in address and message handling, intended to facilitate interconnection with other networks. In particular, addresses can have variable formats, and messages are not delivered in sequence, so that they can flow out of the network through several gates toward an outside target.

Given these goals, it was essential to provide for the serialization (temporal ordering) of packet streams above the layer at which the networks interconnected. Of particular importance was the desire of CYCLADES' designers to split traffic between two endpoints among "several gates," foreshadowing the operation of today's peer-to-peer network overlays. The use of multiple, concurrent paths between networks by a single application more than anything dictated an end-to-end architecture.[23]

The Influence of CYCLADES on Subsequent Networks

CYCLADES was the first system designed for interconnecting networks—that is, internetworking. The architecture of CYCLADES was adopted by the designers of the Internet protocols:[24]

> In June 1973, Cerf, Kahn, and Metcalfe started to work on the design of a host-to-host protocol for internetworking. They called it the Transfer Control Protocol (TCP), and they were helped by a fourth person: Gérard Le Lann, who had worked with Louis Pouzin on Cyclades and was taking a sabbatical at Stanford. Le Lann's experience with Cyclades proved decisive. "Gérard came to Stanford for a sabbatical during the period of critical design of the TCP protocol," says Cerf, going on to recall that "some of Pouzin's datagram ideas and sliding window ideas ended up in TCP." "Datagrams" extended the notion of the packet. Instead of leaving the underlying network, IMPs in the case of the ARPANET, to arrange packets into a sequential stream like carriages on a train, each datagram could be delivered independently, with the host-to-host protocol being responsible for assembling them when they arrived.

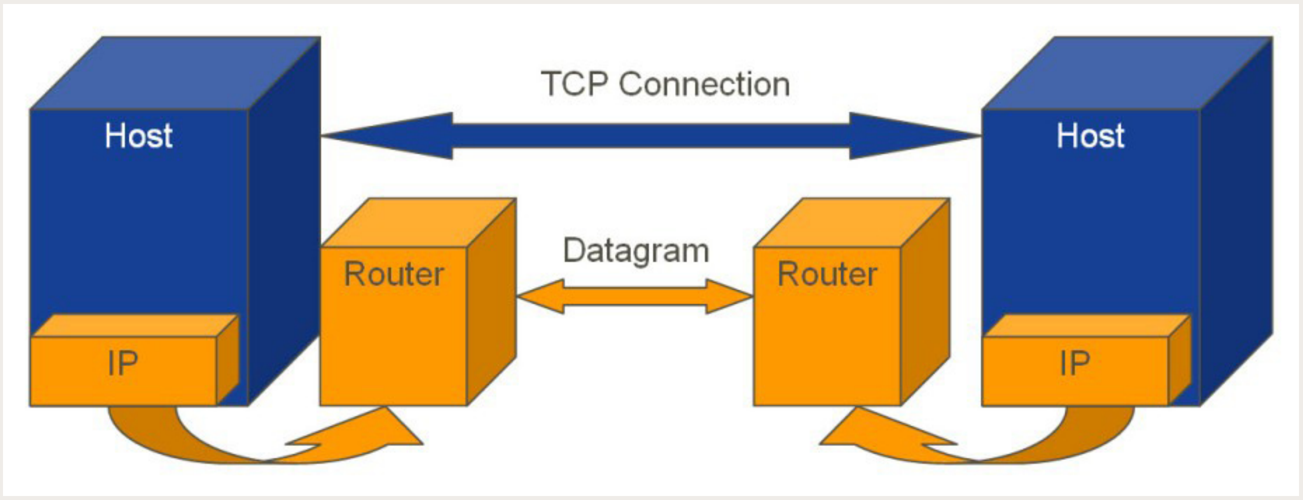
BBN's Walden also contributed design ideas to the Internet designers. Walden's, Pouzin's and Le Lann's contributions are credited, somewhat sparingly, in the first specification of TCP, RFC 675,[25] and in the initial paper on TCP by Cerf and Kahn.[26]

The use of end-to-end protocols in CYCLADES and TCP wasn't an arbitrary choice; it was dictated by the engineering problem the designers set out to solve. The earlier ARPANET didn't need to adopt this approach in part because it was a single, unified network rather than a rambling mesh of different networks interconnected at multiple "gates"[27] and also because it wasn't intended to be a platform for packet-switching research. Commenting on the architecture of CYCLADES (and by implication, that of the Internet protocols), ARPANET planner Larry Roberts declared the design interesting for experimenters but unsuitable for public networks:[28]

> Since a major part of the organization and control of the network is embedded in the CYCLADES computers, the subnetwork, CIGALE, is not sufficient by itself. In fact, Pouzin himself speaks of the network as "including" portions of the host computers. The packet assembly and disassembly, sequence numbering, flow control, and virtual connection processing are all done by the host. The CYCLADES structure provides a good testbed for trying out various protocols, as was its intent; but it requires a more cooperative and coordinated set of hosts than is likely to exist in a public environment.

The allocation of functions in TCP/IP and CYCLADES is more de-centralized than that of ARPANET, in part because the state of the art in processing power and memory was more advanced than it had been

Figure 3: Internet Architecture



when ARPANET was designed; network engineers were also better informed and more eager to try out their ideas on a suitable testbed.

The Connectionless vs. Connection-Oriented Debate

The CYCLADES style of network architecture is characterized by the "connectionless" nature of its network layer protocol: datagrams are offered to the network at any arbitrary time; the network can route them in any arbitrary fashion; they're assembled in proper sequence by the receiving host, not by the network. In other words, an end system doesn't have to do any network setup before sending a datagram to another end system; it simply sends it, relying on the network to deliver it (or not; it has to be prepared for either outcome.) This isn't the only way to design a network layer protocol, however.

In 1976, the International Telegraph and Telephone Consultative Committee (CCITT, now ITU-T) standardized a "connection-oriented" network interface called the X.25 protocol. In connection-oriented networks, the network itself provides the functions of serialization, error correction, and flow control that are performed by end systems in the connectionless model.

From the standpoint of production applications, there's not much difference between the connectionless and connection-oriented models. In both cases, virtual circuits are created and managed by some lower level system function using its own means. While these strategies don't matter a great deal to application developers, they're enormously important to network protocol designers, who are cut out of the loop by connection-oriented network layers.

The general public probably perceives engineers to be highly rational people, more likely to be motivated by facts and logic than by emotion. This view is typically true, but engineers have been known to engage in "holy wars" over apparently small matters of system design (as well as literal holy wars in some cases)[29] This tendency was apparent in the "protocol wars" of the 1980s, when engineers argued about how best to design local area networks (LANs), network architectures, and network protocols.

In many of these battles, the core issues were very similar. In the battle between Ethernet and the IBM Token Ring to become the dominant LAN standard, for example, network engineers hotly debated the value of packet management and prioritization at the Media Access Control sub-layer of the ISO data link layer, a parallel to the connectionless/connection-oriented debate at the network layer. The common understanding that Ethernet won this debate is not correct: the system called "Ethernet" that we use today has almost nothing in common with the Ethernet of the early 1980s. Traditional Ethernet was a fat, shared coaxial cable with no intelligence, but the Ethernet we use today is an intelligent switch to which computers connect by unshared copper or fiber optic cable. Modern Ethernet matches systems running at different speeds and performs Virtual LAN (VLAN) management functions and QoS management, even at the highest speeds (100 Gb/s). It is therefore more accurate to see it as a synthesis of the Token Ring and traditional Ethernet approaches. When the first standard for Ethernet over twisted pair was devised, synthesis indeed was a common goal.[30]

The battle over connectionless and connection-oriented network protocols was carried into the Open Systems Interconnection (OSI) design activity, and never successfully resolved. The impasse between these two approaches—and the lack of constructive American government support—proved fatal to OSI.

Moving From Practice to Theory

In 1981, a trio of operating systems researchers fresh off the Multics project at Massachusetts Institute of Technology (MIT)[31]—Jerome Saltzer, David Reed, and David Clark—fired a salvo in the connectionless/connection-oriented debate with a paper that sought to justify the connectionless design on a theoretical basis, "End-to-End Arguments in System Design" (*End-to-End Arguments*). [32] Notwithstanding the fact that their paper was delivered in Paris, the authors did not seem to have been aware of the French origins of end-to-end architecture: their references don't include Pouzin or his interpreters at Stanford, Xerox, or Digital.

Their argument justifies the end-to-end model on two different bases. The first argument offers end-to-end as the "Occam's razor" of network engineering, proposing that most network functions should be provided only once. The second argument, the philosophical basis of the subsequent debate, asserts that high-level implementation strategies enable solutions that are more "complete and correct" than low-level ones. These are fighting words to the engineer:[33]

> The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)
>
> We call this line of reasoning against low-level function implementation the "end-to-end argument."

This statement is meant to assert that connectionless networks intrinsically yield more "correct" distributed applications than connection-oriented networks such as NCP ARPANET and X.25. Error correction the TCP/IP way is done only once, in the end system (error detection is still done hop-to-hop, however). Given enough time, it corrects all correctable errors introduced by the network or the host system's network interface hardware. Error correction in connection-oriented networks, on the other hand, is done hop-by-hop, but not outside the host's network interface. Connection-oriented networks are capable of detecting and correcting errors faster than connectionless ones, but are less complete since they can't correct errors that creep into a stream after it has crossed the interface from the network to the host system. The TCP method was more complete, and therefore more "correct" than the connection-oriented method even though it was slower.[34]

The logic of *End-to-End Arguments* derives from Pouzin's design, but its attempt to create grand theory is tautological. By hedging the "placement of functions problem" toward a class of problems that can be solved *correctly* in only one way it's not so much a principle as a truism. In such cases, there is no real choice to be made and therefore no need for a guiding principle; defining classes of problems that can be solved only by applications as immune to network solutions merely states the obvious. But the implicit dismissal of speed considerations opens the door to network-based solutions for "performance enhancement." Pouzin made a similar concession in more coherent language:[35]

> With experience gradually building up, and depending on trends in international standards, more stable characteristics will eventually emerge. By putting them at some lower system level, it will be possible to obtain higher efficiency and reduce duplication, at the cost of freezing a few more parameters.

Pouzin was aware that he was building a research network to facilitate experimentation in network architectures, protocols, traffic management systems, and services. He understood that his network would have the greatest utility for its intended use if it "froze" as few functions as possible in the network infrastructure.

Pouzin anticipated a cycle of experimentation in which new functions would be tested in end-point devices and then migrated into network infrastructure to increase efficiency.

Pouzin anticipated a cycle of experimentation in which functions would be tested in devices attached to the network and migrated into network infrastructure only when experimental data showed that efficiency could be enhanced by "lower system level" implementation. This network innovation cycle is evident in the history of the Internet as a public network. When the "lower system level" implementation part of this process is denied, end-to-end degenerates into dogma.

The sensible application of the Occam's razor proposed by the authors of *End-to-End Arguments* to network engineering would simply hold that a given function should be implemented once and only once in a network; this formulation would still be fundamentally incorrect, however, as many functions are performed multiple times. Error correction, for example, is implemented at least four times in modern wireless systems: at the level of coding, at the data link layer, in the TCP layer, and in the application space for P2P protocols such as BitTorrent. This is simply because of the fact that errors creep into a system for many different reasons and have to be corrected on the time scale pertinent to the type of error. The disregard for time when judging the correctness of implementations actually foreshadows much of the development of the Internet, however, and exposes a fundamental weakness in the initial attempts at *post hoc* network architecture.

End-to-End Arguments glosses over the issue of network efficiency. While packet-switching is more bandwidth-efficient than circuit-switching for applications with low duty cycles (i.e., those that are frequently idle from the network's point of view,) implementing real-time network functions as close to the application as possible carries a number of significant harmful side effects. An application endpoint is by its nature not as well aware of network conditions affecting its data stream as a management sensor inside the network, and is therefore unable to predict whether a given disruption is short- or long-lived. Application endpoints are also slower to react to network conditions than embedded network sensors. And the former are driven by different economics: each application is driven by its desire to obtain as much bandwidth as it can get, while the network is more likely to seek the "greatest good for the greatest number." Network traffic events are often ambiguous to the application endpoint. If an endpoint detects packet loss, it doesn't know whether the cause was an error on a Wi-Fi link or congestion in a core router. The correct response to the one is entirely incorrect in the case of the other. Work continues in network engineering to address such problems with clever heuristics,[36] but the real answer is better data-sharing, such as the queue depth information exchanged between end system and network by IEEE 802.11e.

What are we to do about problems that can only be solved correctly inside the network or those that can only be solved correctly by the cooperation of the network with the end system? Congestion management is one such problem, and its correct resolution is essential to the correct operation of a large-scale Internet, as we shall see when we discuss congestion collapse.

End-to-End Arguments was more a polemic on behalf of one side of the debate than a serious analytical offering. While it paid lip service to the features that connections provide, it was clearly a heavy-handed attempt to justify a particular system architecture. It drew a line in the sand that exaggerated the conflict over network layer protocols, taking a step back from the cycle of "experiment in the sandbox, implement in the core" proposed by Pouzin. Nevertheless, it did succeed in starting a discussion in the network engineering community about why the Internet was designed the way it was, and erected a burden of proof that would have to be met by alternative architectures. The engineering community attacked these challenges enthusiastically, as we shall see.

Single Points of Failure

If Saltzer, Reed, and Clark had waited for the wide-scale deployment of TCP/IP before publishing *End-to-End Argument*, they might have found cause to revise their thinking. Their thinking was biased, by their own admission, by a single incident of hardware failure:[37]

> An interesting example of the pitfalls that one can encounter turned up recently at M.I.T.: One network system involving several local networks connected by gateways used a packet checksum on each hop from one gateway to the next, on the assumption that the primary threat to correct communication was corruption of bits during transmission. Application programmers, aware of this checksum, assumed that the network was providing reliable transmission, without realizing that the transmitted data was unprotected while stored in each gateway. One gateway computer developed a transient error in which while copying data from an input to an output buffer a byte pair was interchanged, with a frequency of about one such interchange in every million bytes passed. Over a period of time, many of the source files of an operating system were repeatedly transferred through the defective gateway. Some of these source files were corrupted by byte exchanges, and their owners were forced to the ultimate end-to-end error check: manual comparison with and correction from old listings.

The MIT programmers should have read the passage on error checking in RFC 1.[38] Few things are more painful to programmers than retyping source code files, so some blindness is understandable. But the larger problem with *End-to-End Arguments* is that it compares a fully mature production network that had been in operation for some 12 years (NCP ARPANET) with a new network system that had scarcely seen life outside the lab (TCP/IP). Moreover, it relied on a single hardware failure[39] coupled with some laziness on the part of an application programmer to condemn its architecture. The declaration of such a grand and sweeping principle needs a much stronger foundation than a single anecdote.

The end-to-end strategy moved flow control—the part of a network system that matches the capabilities of sending and receiving systems to prevent overflow at the receiver—to the endpoints. It intended to leave congestion control inside the network through a system of feedback messages taken from CYCLADES but adapted poorly. Testing under laboratory conditions failed to expose any weakness in the feedback system, so deployments began in 1981 at such places as Ford Aerospace. Ford discovered a problem with TCP hosts that was a near mirror image of MIT's problem with a malfunctioning IMP:[40]

> On one occasion in late 1983, a TCP bug in an ARPANET host caused the host to frantically generate retransmissions of the same datagram as fast as the ARPANET would accept them. The gateway that connected our net with the ARPANET was saturated and little useful traffic could get through, since the gateway had more bandwidth to the ARPANET than to our net. The gateway busily sent ICMP Source Quench messages but the malfunctioning host ignored them. This continued for several hours, until the malfunctioning host crashed. During this period, our network was effectively disconnected from the ARPANET…
>
> Ideally the gateway should detect malfunctioning hosts and squelch them; such detection is difficult in a pure datagram system. Failure to respond to an ICMP Source Quench message, though, should be regarded as grounds for action by a gateway to disconnect a host. Detecting such failure is non-trivial but is a worthwhile area for further research.

The bad TCP module exhibited a behavior much like that of the modern bandwidth hog who uses more than his fair share of the network; Nagle's suggestion for this latter problem was to implement a bandwidth mediator in between TCP and IP to fairly distribute transmission opportunities among the users of an IP network. Thirty years later, Nagle offered commentary on the contemporary Comcast system for managing peer-to-peer-induced overload that illustrates its use:[41]

> As the one who devised much of this congestion control strategy (see my RFC 896 and RFC 970, years before Van Jacobson), I suppose I should say something.
>
> The way this was supposed to work is that TCP needs to be well-behaved because it is to the advantage of the endpoint to be well-behaved. What makes this work is enforcement of fair queuing at the first router entering the network. Fair queuing balances load by IP address, not TCP connection, and "weighted fair queuing" allows quality of service controls to be imposed at the entry router…
>
> I'd argue for weighted fair queuing and QOS in the cable box. Try hard to push the congestion control out to the first router.

Balancing load by IP address is critically different in effect from balancing by TCP connection; a computer typically has only one IP address, but it can use dozens or more TCP ports at any give time. End-to-end manages by TCP port, and network-based systems manage by IP address.

Taken together, MIT's experience with a malfunctioning IMP interface card and Ford's experience with a malfunctioning TCP software module teach the lesson that robust networks need to guard against design dogmas. In the fullness of time, any single component in a network system that can fail will fail; the task of network design and operation is to mitigate the effects of these inevitabilities. But the problems at Ford and MIT were minor compared to what was to come.

Internet Congestion Collapse

January 1, 1983, was Flag Day for ARPANET, when the last NCP hosts were turned off and TCP/IP systems took over. The new Internet performed reasonably well unti 1986, when users began to notice periods of extremely low throughput:[42]

> In October of '86, the Internet had the first of what became a series of "congestion collapses".

> During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and two IMP hops) dropped from 32 Kbps to 40 bps. We were fascinated by this sudden factor-of-thousand drop in bandwidth and embarked on an investigation of why things had gotten so bad. In particular, we wondered if the 4.3 BSD (Berkeley UNIX) TCP was misbehaving or if it could be tuned to work better under abysmal network conditions. The answer to both of these questions was "yes."

The congestion collapse that hit the Internet in 1986 was caused by TCP's reaction to packet loss:

1. When routers were overloaded, they dropped packets.
2. When TCP detected packet loss, it retransmitted.
3. The process repeated until some sending hosts were turned off.

A debate ensued about how best to address the problem in which at least three alternatives were put forward. The best one was a proposal from Raj Jain, K. K. Ramakrishnan, and Dah-Ming Chiu of Digital Equipment Corporation for IP to signal the onset of congestion by setting a "congestion bit" in the IP datagram header:[43]

> We studied a number of alternative schemes for congestion avoidance. Based on a number of requirements described later in this report, we selected an alternative called the binary feedback scheme for detailed study. This scheme uses only a single bit in the network layer header to feed back the congestion information from the network to users, which then increase or decrease their load on the network to make efficient and fair use of the resources.

In one of network engineering's greatest fits of myopia, this method was rejected in favor of Jacobson's Algorithm—a simple, four-line patch. Jacobson claimed it was better than the alternatives:[44]

> …Packets get lost for two reasons: they are damaged in transit, or the network is congested and somewhere on the path there was insufficient buffer capacity. On most network paths, loss due to damage is rare (<<1%) so it is probable that a packet loss is due to congestion in the network.
>
> A 'congestion avoidance' strategy, such as the one proposed in [Jain, et. al.], will have two components: The network must be able to signal the transport endpoints that congestion is occurring (or about to occur). And the endpoints must have a policy that decreases utilization if this signal is received and increases utilization if the signal isn't received.
>
> If packet loss is (almost) always due to congestion and if a timeout is (almost) always due to a lost packet, we have a good candidate for the 'network is congested' signal. Particularly since this signal is delivered automatically by all existing networks, without special modification (as opposed to [Jain et. al.] which requires a new bit in the packet headers and a modification *to all* existing gateways to set this bit).

From today's vantage point, the parochialism of Jacobson's critique of the binary feedback system is painfully obvious: not only was he blind to the effects of wireless networks on error rates, he was worried about the difficulty of upgrading a few hundred routers to correct a design defect in a system that would grow to encompass *many million* routers and more than a billion users. The four line Jacobson patch was the preferred method to shore up a fundamental design defect in IP because it could be deployed without network cooperation. It was incorporated into Berkeley UNIX 4.3 and then to the rest of the Internet. It's now known as Jacobson's Algorithm.

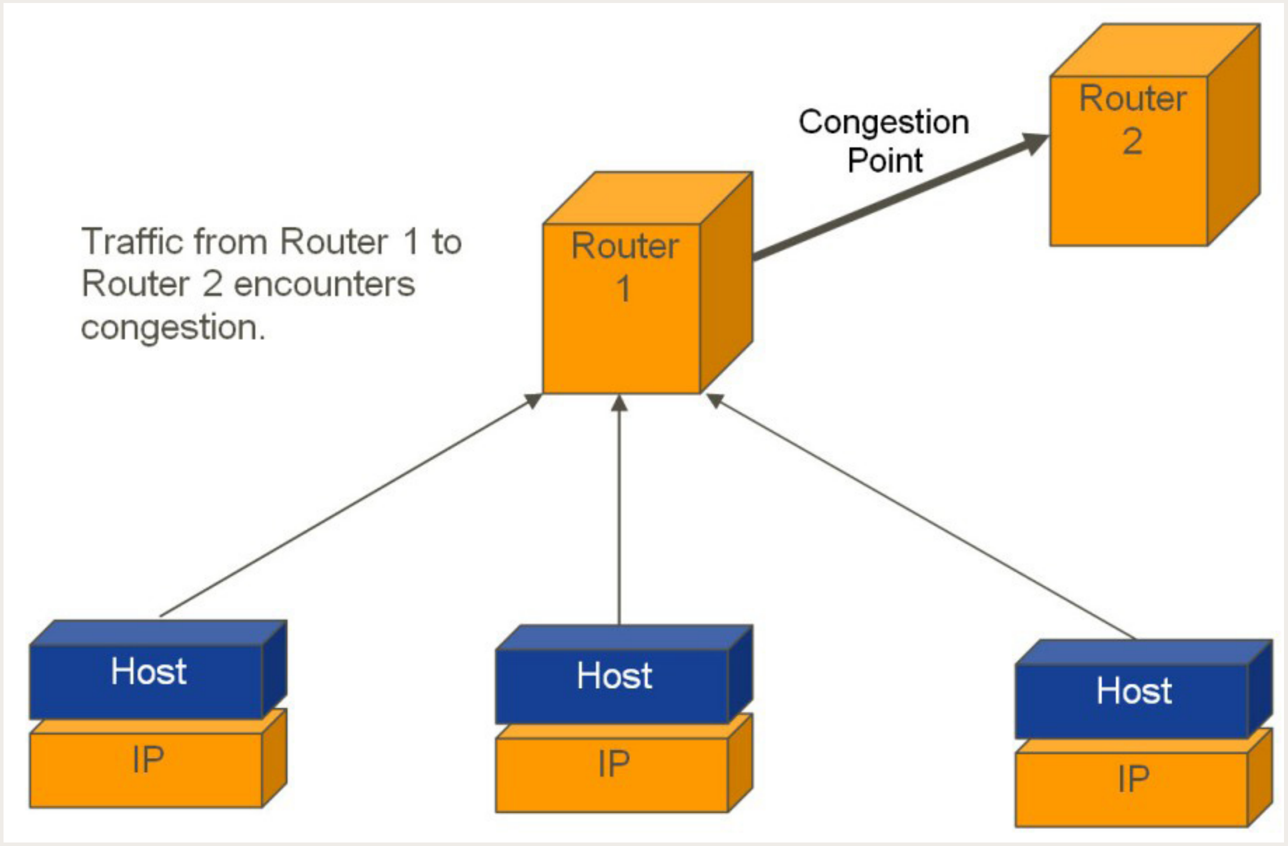
The Jain solution was eventually adopted as an Internet Standard in 2001,[45] but by then, barriers to adoption were real: many thrown-together home gateways could not process packets with the congestion bit set, so Microsoft refused to enable it as the default mode of operation for Vista;[46] ECN is disabled by default in Linux as well.

Jacobson's Algorithm imposes an enormous cost on the Internet's core links. Its behavior—cycling between 50 percent utilization and overload—dramatically reduces the efficiency of Internet links. In the judgment of some of the Internet Engineering Task Force's best minds, it's an insufficient solution to the congestion problem unless complemented by intelligent management of router queues *inside the Internet's core*:[47]

> The original fix for Internet meltdown was provided by Van Jacobson. Beginning in 1986, Jacobson developed the congestion avoidance mechanisms that are now required in TCP implementations [Jacobson88, HostReq89]. These mechanisms operate in the hosts to cause TCP connections to "back off" during congestion. We say that TCP flows are "responsive" to congestion signals (i.e., dropped packets) from the network. It is primarily these TCP congestion avoidance algorithms that prevent the congestion collapse of today's Internet.
>
> However, that is not the end of the story. Considerable research has been done on Internet dynamics since 1988, and the Internet has grown. It has become clear that the TCP congestion avoidance mechanisms [RFC2001], while necessary and powerful, are not sufficient to provide good service in all circumstances. *Basically, there is a limit to how much control can be accomplished from the edges of the network*. Some mechanisms are needed in the routers to complement the endpoint congestion avoidance mechanisms. [Emphasis added]

One of the mechanisms that can be used inside the network is Explicit Congestion Notification (ECN), which provides a performance boost of roughly 25 to 50 percent over Jacobson by reducing packet loss and global synchronization.[48] It's widely acknowledged that Jacobson's Algorithm is no longer appropriate for the Internet. The deployment of high-speed lines extending over long distances— "Long Fat Pipes" or LFPs— has completely broken its arithmetic: Web connections over LFPs don't graduate from their initial Slow Start state to full utilization before their transfers terminate.[49] To put it plainly, the seminal example of end-to-end network control proved ineffective.

Figure 4: Equal Pipes Congestion Scenario



One attempt to rectify the problems caused by congestion management in the modern Internet is an informal committee forming within the IETF to investigate alternative methods to the classical approach. This committee is examining extensions to ECN that increase the cooperation between network and end systems such as the Re-ECN proposal put forward by Bob Briscoe of University College, London, and British Telecom. Briscoe proposes that the congestion state signaled by network routers through ECN be echoed back to the sender of the packet which encountered congestion (ECN only communicates it to the receiver). Re-ECN allows the sender to predict which packets will be likely to encounter congestion and to make a policy choice about them, and for the network operator to cede control over such decisions to the sender, along with the responsibility for economic side effects. Such systems, if implemented correctly, eliminate the need for arbitrary limits on packet volume and for classification based on deep-packet inspection. These effects satisfy some of the demands of the net neutrality movement; facing-up to the economic consequences of congestion raises their ire, however.[50]

THE REHABILITATION OF THE END-TO-END ARGUMENTS

Following the 1981 publication of *End-to-End Arguments*, a number of voices in the engineering community argued its merits and demerits, including original author, David D. Clark.[51] Clark revised his thinking on end-to-end arguments in later articles, arguing that the reason to prefer high-level implementations was reliability: low-level implementations tied the success of the application to network components, which could fail. Clark coined the term "fate-sharing" to recast the rationale for end-to-end in these less incendiary technical terms.

Out of these critiques, a consensus emerged to the effect that the Internet needed an architecture, that this architecture should be couched, at least in part, in terms of *end-to-end arguments*, but that the 1981 paper didn't adequately capture the meaning of this central principle or its rationale. Ultimately, the rationale is the most important part of the end-to-end arguments, and there's no getting around the fact that the various rationales are in their own way driven by external conditions. As the Internet's patterns of use change, decisions about function allocation (and many other things) must be reviewed. Changes in patterns of use cause changes in operational methods, which raise questions about architecture. The process of review, revision, and improvement is continual.

End-to-End as Limited Fate-Sharing

In 1996, Brian Carpenter of the European Organization for Nuclear Research (CERN) made the Internet community's first attempt to provide a formal description of the Internet's architecture, RFC 1958. The effort was equivocal:[52]

> Over the 25 years since the ARPANET started, various measures of the size of the Internet have increased by factors between 1000 (backbone speed) and 1,000,000 (number of hosts). In this environment, some architectural principles inevitably change. Principles that seemed inviolable a few years ago are deprecated today. Principles that seem sacred today will be deprecated tomorrow. The principle of constant change is perhaps the only principle of the Internet that should survive indefinitely.

Carpenter gave deference to the end-to-end arguments, but offered a justification wholly different from the correctness standard set forth in *End-to-End Arguments;* Carpenter endorsed the notion of limiting fate-sharing, which had been part of the justification for datagrams in the Pouzin era:[53]

> An end-to-end protocol design should not rely on the maintenance of state (i.e. information about the state of the end-to-end communication) inside the network. Such state should be maintained only in the endpoints, in such a way that the state can only be destroyed when the endpoint itself breaks (known as fate-sharing). An immediate consequence of this is that datagrams are better than classical virtual circuits. The network's job is to transmit datagrams as efficiently and flexibly as possible. Everything else should be done at the fringes.

So we're no longer concerned about "correctness" of implementations in a domain where time isn't part of the definition, we're concerned about reliability and resiliency: end-to-end solutions are preferred over "smart network" solutions because they survive network crashes, packet corruption, and modem problems. It's rather apparent that Carpenter is following in the footsteps of *End-to-End Arguments* insofar as he attempts to extract architecture from a system that had actually been designed for practical ends rather than for grand architectural ones. Carpenter's effort is more successful, however, as it focuses on an aspect of the Internet that had become very well understood by 1996, the datagram.

Carpenter's argument is fundamentally incorrect, however. As John Day points out, datagram systems are capable of routing around failed links not because they have so little information about the network's state, but because they have so much.[54] A router in a datagram network must know all viable routes to all *possible* networks and must know them all the time; a connection-oriented router only needs to know viable routes between the endpoints that are *actively connected* to each other. The datagram network must be a great deal smarter than the connection-oriented one. Perversely, datagram networks are maximally fate-sharing, not minimally; they're reliable not because they're simple, but because they're highly redundant.

Who Do You Trust?

In 2002, Tim Moors showed that *End-to-End Arguments* confused the notion of a communication *endpoint* with that of an *end system*:[55]

> The Paper suggests that file transfer applications are the endpoints, and that after the destination has stored the file to disk, it should read the file off disk and calculate an integrity check, which it returns to the source application, which decides whether the transfer was "correct." This use of application-layer checks is also supported by a recent study of the sources of data transfer errors. However, it is a popular belief that the transport layer (e.g. TCP) is the endpoint for applications such as file transfers. The transport layer merely resides in the *end system*. The transport layer checks will not protect against errors introduced as the information is written to disk. (Note that while natural noise and potential design errors are omnipresent, physical security measures (e.g. electromagnetic shielding) can isolate a system from security threats. Thus, if the user trusts the end system then the transport layer of the end system may constitute a security endpoint.) *According to the end-to-end arguments, applications, not transport layers, should check integrity.*
>
> So why do most applications entrust transport layers such as TCP to provide reliable transfer? There are two reasons for this: the first reason is the high cost of the application performing the error checking, and the second reason is the low *benefit* from this activity.
>
> … It is relatively easy for the application to know about the reliability of its local system, but more difficult for it to know about the reliability of intermediaries that its data must pass through as it crosses a network. Thus, *the decision to implement reliable transfer in the transport layer is not justified on the basis of end-to-end arguments, but rather on the basis of trust.* [References omitted, emphasis in original]

File transfer applications do not routinely check the integrity of file input/output operations for the same reason that ARPA programmers didn't check packets after they were delivered to host memory by the IMP, because there's not enough payback in terms of additional data integrity to justify the extra programming and increased execution time. But why don't programmers simply turn off TCP error checking in favor of error checking one and only one time, after file data have been written to disk? This question goes to the essence of the fundamental flaw in the end-to-end bias.

File transfer applications want to complete as quickly as possible, and moving end-to-end integrity checks from TCP to the application would slow them down appreciably. Not only will the sender be constrained by the speed with which the receiver can write parts of the file to disk, read them back, check them, and signal readiness for the next part, it will additionally be constrained by the fact that these operations are performed in the end system's application space rather than by its operating system kernel, where TCP resides. One of the functions of TCP, post Jacobson, is to keep network traffic moving as fast as the network's congestion state allows. When this function is moved from the kernel, where code runs at predictable intervals because the operating system has high execution priority, to application space, variations in end system application load would directly affect network efficiency and throughput. Thus the sending system would not be able to tell the difference between network congestion and application congestion, and would constantly slow down in response to transient conditions on the receiving system such as disk head movement. Packet loss and delay aren't the unambiguous network congestion signals Jacobson took them for.

The tradeoff in current file transfer applications is to have one system's TCP send as fast as it can to the other system's TCP, and to simply assume that the file data will be transferred to the file system both correctly and faster than the network transfer. As long as the file operation is faster on average than network operation, its speed doesn't affect the end-to-end file transfer operation; and if it becomes slower than the network, the TCP sender will eventually slow down to match it.

The incompletely end-to-end error checking by file transfer applications provides performance benefits to both end systems and to the network. The Internet's file transfer protocol ("ftp") implements error checking and recovery at the lowest level of protocol in the end system at which it can be implemented, not at the highest level as *End-to-End Arguments* predicts. The design of ftp is dictated by performance and simplicity considerations.

Just as file transfer applications relegate error checking to an intermediate layer within the end system, real-time applications such as video streaming and conferencing benefit by relegating it to intermediate network nodes.

Layering Considered Harmful

In 2003, Randy Bush and Dave Meyer undertook an update of RFC 1958 that reinstated the simplicity principle to the status of primary component of end-to-end.[56] Bush and Meyer rejected the notion that the network should have no "intelligence":[57]

> [T]he End-to-End Argument does not imply that the core of the Internet will not contain and maintain state. In fact, a huge amount coarse-grained state is maintained in the Internet's core (e.g., routing state). However, the important point here is that this (coarse grained) state is almost orthogonal to the state maintained by the end-points (e.g., hosts). It is this minimization of interaction that contributes to simplicity.

It's particularly noteworthy that RFC 3439 finds fault with functional layering, the prerequisite principle to end-to-end. Single-function protocol layering adds unnecessary complexity to a system, causing problems for scalability and resiliency:[58]

As a result of inter-layer dependencies, increased layering can quickly lead to violation of the Simplicity Principle. Industry experience has taught us that increased layering frequently increases complexity and hence leads to increases in OPEX, as is predicted by the Simplicity Principle. A corollary is stated in RFC 1925 [RFC1925], section 2(5):

> "It is always possible to agglutinate multiple separate problems into a single complex interdependent solution. In most cases this is a bad idea."

The first order conclusion then, is that horizontal (as opposed to vertical) separation may be more cost-effective and reliable in the long term.

Horizontal separation is the principle upon which contemporary efforts to create a recursive architecture for Internet protocols, such as RINA, are based. Its fundamental insight is that networking problems such as route discovery, flow control, name-to-address mapping, and location discovery don't have to be solved once and only once for the whole Internet. It's much more beneficial, in fact, to solve them separately in each management domain through which a packet moves. We'll have more to say about this later.

The Evolution of End-to-End

In 2004, James Kempf and Rob Austein wrote RFC 3724 on challenges facing the end-to-end arguments. The principal pressures were actually new requirements that came to the Internet as the result of the broader user base, new multimedia applications, and the requirements of commerce and government. [59]

> Given the pressures on the end-to-end principle discussed in the previous section, a question arises about the future of the end-to-end principle. Does the end-to-end principle have a future in the Internet architecture or not? If it does have a future, how should it be applied? Clearly, an unproductive approach to answering this question is to insist upon the end-to-end principle as a

> fundamentalist principle that allows no compromise. The pressures described above are real and powerful, and if the current Internet technical community chooses to ignore these pressures, the likely result is that a market opportunity will be created for a new technical community that does not ignore these pressures but which may not understand the implications of their design choices. A more productive approach is to return to first principles and re-examine what the end-to-end principle is trying to accomplish, and then update our definition and exposition of the end-to-end principle given the complexities of the Internet today.

Kempf and Austein proposed recasting the rationale for end-to-end once again, this time in terms of protecting innovation:[60]

> One desirable consequence of the end-to-end principle is protection of innovation. Requiring modification in the network in order to deploy new services is still typically more difficult than modifying end nodes. The counterargument—that many end nodes are now essentially closed boxes which are not updatable and that most users don't want to update them anyway—does not apply to all nodes and all users. Many end nodes are still user configurable and a sizable percentage of users are "early adopters," who are willing to put up with a certain amount of technological grief in order to try out a new idea. And, even for the closed boxes and uninvolved users, downloadable code that abides by the end-to-end principle can provide fast service innovation. Requiring someone with a new idea for a service to convince a bunch of ISPs or corporate network administrators to modify their networks is much more difficult than simply putting up a Web page with some downloadable software implementing the service.

This effort goes in the right direction, even if it isn't entirely successful. The degree of innovation that can be performed in an end system is entirely dependent on the capabilities of the network to link end systems together. Hence, a reasonable variety of pre-existing delivery services embedded in the network facilitates end system innovation by providing close approximations of application requirements. And as we learned from the congestion issue, there are some network problems that can only be solved efficiently and correctly by the close coordination of end systems with the network. RFC 3724 was concerned with beating back a system called "Open Pluggable Edge Services"[61] that inserted application-level services inside the network core, as content delivery and cloud computing services do today.

Clark's Tussle

By 2005, the end-to-end arguments had been defined and redefined several times. What was originally a process became a rule based on an ever-shifting series of rationales. It had become clear that there would never be consensus on the reason to prefer end-to-end, even though its general value was an accepted part of Internet engineering culture.

In 2005, David Clark stepped back into the end-to-end fray with a new group of colleagues and a new central principle—design for "tussle", the ongoing contention among different parties in the Internet milieu in favor of particular interests. Their article, "Tussle in Cyberspace: Defining Tomorrow's Internet" modestly summarized the then-current state of the end-to-end arguments:[62]

> One of the most respected and cited of the Internet design principles is the end-to-end argument, which state that mechanism should not be placed in the network if it can be placed at the end node, and that the core of the network should provide a general service, not one that is tailored to a specific application. There are two general dimensions to the arguments: innovation and reliability.
>
> *Innovation:* If the core of the network has been tailored to one specific application, this may inhibit the deployment of other applications. If the core of the network must be modified to deploy a new application, this puts a very high hurdle in front of any unproven idea, and almost by definition, a new idea is unproven.
>
> *Reliability and robustness:* If bits of applications are "in the network," this increases the number of points of failure that can disable the application. The more simple the core of the network, the more reliable it is likely to be.

> The simplest application of the end-to-end arguments produces a network that is *transparent*; packets go in, and they come out, and that is all that happens in the network. [References omitted]

Networks are never innovation-neutral, of course: any base level of service is more useful to some applications than to others. Moreover, networks are never transparent: some paths are shorter and faster than others, machines crash, links congest, network administrators advertise incorrect BGP routes, and optimizations overlay network structure for both good and bad reasons. *Tussle in Cyberspace* exposes the end-to-end arguments for what they've always been—a convenient fiction.

Given the character of the end-to-end arguments, it's understandable that nearly 30 years of discussion has failed to converge on a consistent rationale. End-to-end arguments exist to fill a void in the design space. Paraphrasing Voltaire, "if Internet architecture did not exist, it would be necessary to invent it." Architecture provides guidance to future network engineers about how best to extend and enhance the existing Internet. Overly rigid end-to-end arguments simply don't do this very well.

In the place of end-to-end arguments, Clark and his colleagues proposed that "design for tussle" should become the central principle of Internet architecture, noting that this new principle enables competition:[63]

> Anyone who designs a new enhancement for the Internet should analyze the tussles that it will trigger, and the tussles in the surrounding context, and consider how they can be managed to ensure that the enhancement succeeds. As noted above, a powerful force is the tussle of competition. Protocol design, by creating opportunities for competition, can impose a direction on evolution.

Competition, of course, is squarely within the bailiwick of policy; had the point of the end-to-end arguments all along been more about influencing public policy than elucidating network architecture? It seems reasonable to suppose that it was.

Had the Internet's technical overseer, the Internet Engineering Task Force (IETF), been guided more by pragmatic engineering goals than by ideology, it might have reached a connectionless/connection-oriented synthesis and might not have failed to deliver workable end-to-end QoS and reliable multicast standards, to mention two transport services that the new generation of real-time video applications require that got stuck in the fog around end-to-end.[64]

A good way of understanding "tussle" is to see it as a replacement for "end-to-end" in protocol arguments. Although "tussle" is still a bit vague, it doesn't yet suffer from the ambiguity around end-to-end that's the legacy of numerous redefinitions, and it also restores the centrality of the "innovation cycle" to Internet engineering. But it's difficult to apply as an architectural principle, since the network protocol designer has no way of knowing where future tussles are going to break out. The ultimate value of tussle, however, is that it refers back to the exception to end-to-end absolutism described by Pouzin: There are instances in which network-based solutions are superior to endpoint solutions, especially those where performance is a primary concern. If we understand networking as a results-oriented process rather than the blind application of inherited wisdom, our systems will work better.

Current Status of End-to-End

It should be evident that *End-to-End Arguments* has come to be regarded by the Internet engineering and operations community with something less than abject reverence in the nearly 30 years since it was written. The paper was chock full of insight and laudable in its effort to uncover an architecture embedded in the Internet protocols, but the fact that it was written before the TCP/IP protocols had been widely deployed limited its usefulness. The 1981 paper's key insight turned out to be the emphasis on simplicity. Its key drawback was its lack of consideration for trust and performance, and indeed its failure to offer up an abstract view of the Internet protocols—in the end, the paper is little more than question-begging.

Pouzin's concept of protocol implementation as an iterative process seems to be coming back in style: rather than seeking the *a priori* correct implementation point, Internet engineers have come to appreciate that experimenting in the endpoint and implementing in the core was the way to achieve the best balance of flexibility and performance. The notion of "correct implementation" that end-to-end fundamentalists proposed as the central principle of placement is an indisputable tautology, serving primarily to open a debate about what we mean by "correct" in a world where applications have varied requirements in the domains of both time and money. The consensus in the Internet community is that, to the extent feasible, end-to-end implementations (in the sense of those at a high level in a vertical protocol stack) are preferred over network-based solutions. The discussion continues as to how we evaluate feasibility at Internet scale and what we do when end-point implementations aren't practical.

Pouzin's concept of protocol inplementation as an iteractive process seems to be coming back in style. Experimenting in the end-point and implementating in the core is best way to balance flexibility and performance.

One of the most important lessons in Internet operations (emphasized in RFC 3429) is that many problems only show up at scale, so research results obtained in small-scale systems have limited utility in terms of predicting behavior on the Internet.[65] This lesson has been learned by the operations community quite painfully through Congestion Collapse and a series of related incidents. Hence, engineering and operations continue to operate in the climate described by Carpenter: "The principle of constant change is perhaps the only principle of the Internet that should survive indefinitely." Clark's "tussle" is attracting considerable attention. At the 75th meeting of the Internet Engineering Task Force (IETF) in Stockholm in July 2009, Mark Handley touted the tussle principle to attendees of the technical plenary, in a session devoted to consideration of the effects of network neutrality regulations on Internet engineering.[66] "Designing for Tussle" has become part of the conversation around Internet protocols, and end-to-end fundamentalism is receding into the background.

END-TO-END IN REGULATION

Debates about the role and definition of the end-to-end arguments might have been confined to the network engineering community were it not for the efforts of a group of law professors, policy advocates, and activists to create an academic discipline around Internet policy called Cyberlaw[67] and a related popular movement.

One extreme of Cyberlaw is represented in a manifesto drafted by one of the Berkman Center for Internet and Society's early fellows, cyber-libertarian John Perry Barlow.[68] Barlow asserts that networks define a new reality immune to law:[69]

> Governments of the Industrial World, you weary giants of flesh and steel, I come from Cyberspace, the new home of Mind. On behalf of the future, I ask you of the past to leave us alone. You are not welcome among us. You have no sovereignty where we gather.
>
> We have no elected government, nor are we likely to have one, so I address you with no greater authority than that with which liberty itself always speaks. I declare the global social space we are building to be naturally independent of the tyrannies you seek to impose on us. You have no moral right to rule us nor do you possess any methods of enforcement we have true reason to fear.

Barlow's views echo similar musings by Timothy Leary "extolling the PC as the LSD of the 1990s."[70]

These views were subsequently echoed by a pair of influential papers by David Isenberg, a former Bell Labs engineer and subsequent Berkman Fellow. In the first paper, published in 1997, "Rise of the Stupid Network," Isenberg transforms end-to-end arguments into a system of network management:[71]

> A new network "philosophy and architecture," is replacing the vision of an Intelligent Network. The vision is one in which the public communications network would be engineered for "always-on" use, not intermittence and scarcity. It would be engineered for intelligence at the end-user's device, not in the network. And the network would be engineered simply to "Deliver the Bits, Stupid," not for fancy network routing or "smart" number translation.
>
> *Fundamentally, it would be a Stupid Network.*
>
> In the Stupid Network, the data would tell the network where it needs to go. (In contrast, in a circuit network, the network tells the data where to go.) [Sic] In a Stupid Network, the data on it would be the boss...

> [T]he Stupid Network would have a *small repertoire of idiot-savant behaviors to treat different data types appropriately*. If the data identified itself as financial data, the Stupid Network would deliver it accurately, no matter how many milliseconds of delay the error checking would take. If the data were two-way voice or video, the Stupid Network would provide low delay, even at the price of an occasional flipped bit. If the data were entertainment audio or video, the Stupid Network would provide wider bandwidth, but would not necessarily give low delay or absolute accuracy. And if there were a need for unique transmission characteristics, the data would tell the Stupid Network in more detail how to treat it, and the Stupid Network would do what it was told. [Emphasis added]

"Fancy network routing" is actually the central role of Internet routers and a vital service for mobile devices, but we need not discuss these details et.

Isenberg's "idiot-savant behaviors" preserved a small role for intelligence in the network, roughly consistent with the IETF work on Active Queue Management. In a second *stupid network* paper, "Dawn of the Stupid Network," Isenberg stripped these behaviors out:[72]

> Intelligent Network advocates point out that networks need to treat different data types differently. Right now, they're absolutely correct. There is a network for telephony, another network for TV, and proprietary leased-line networks for financial transactions—and none of these are ideal for public Internet traffic. You need to have low delay for voice telephony, the ability to handle megabit data streams with ease for TV, and low error rates and strong security for financial transactions.
>
> Quality of Service (QOS) is an intermediate step in the journey from separate networks to a single, simple Stupid Network. QOS, in standard telco thinking, means a repertoire of different ways of handling each type of data on a single network. If the Stupid Network is to become a bona fide integrated service network, it will need to carry all kinds of data with different needs.
>
> But suppose technology improves so much that the worst QOS is perfectly fine for all kinds of traffic, without a repertoire of different data handling techniques. Suppose, for example, that everyday normal latency becomes low enough to support voice telephony, while at the same time allowing enough capacity for video, plus data integrity strong enough for financial transactions. This would be a true Stupid Network—one treatment for all kinds of traffic.

Thus, Isenberg made a crucial step in end-to-end's journey from a research-oriented model of distributed system implementation towards a mandate that forbids active management, paving the way for modern net neutrality regulations.

Isenberg's second Stupid Network paper dropped "idiot-savant behaviors."

Isenberg's speculations about QoS betray a certain naïveté about the distribution of traffic and delay on packet networks, perhaps a consequence of his telephony background. On the public switched telephone network (PSTN) each call is allocated a constant and fixed quantum of bandwidth, and the network is designed to eliminate side effects between users that would otherwise lead to the variations in delay characteristic of degraded QoS. The Internet is a dynamic, shared bandwidth system that relies on statistics to provide QoS for most uses most of the time. Its design precludes a state where "normal latency" can ever be sufficient for all applications unless we severely limit the diversity of applications to the set that can be served by generic delivery; this is simply because every user affects every other user on a shared-resource network. Some of the virtual private networks (VPNs) in the Internet's optical core serve financial applications, which require latencies as low as 1 to 2 milliseconds (ms), while generic latencies tend to range from 50 ms to 200 ms end-to-end. Simply adding bandwidth doesn't solve the latency problem because Internet traffic tends to come in self-similar clumps, not in the Poisson Distributions imagined by early network researchers.

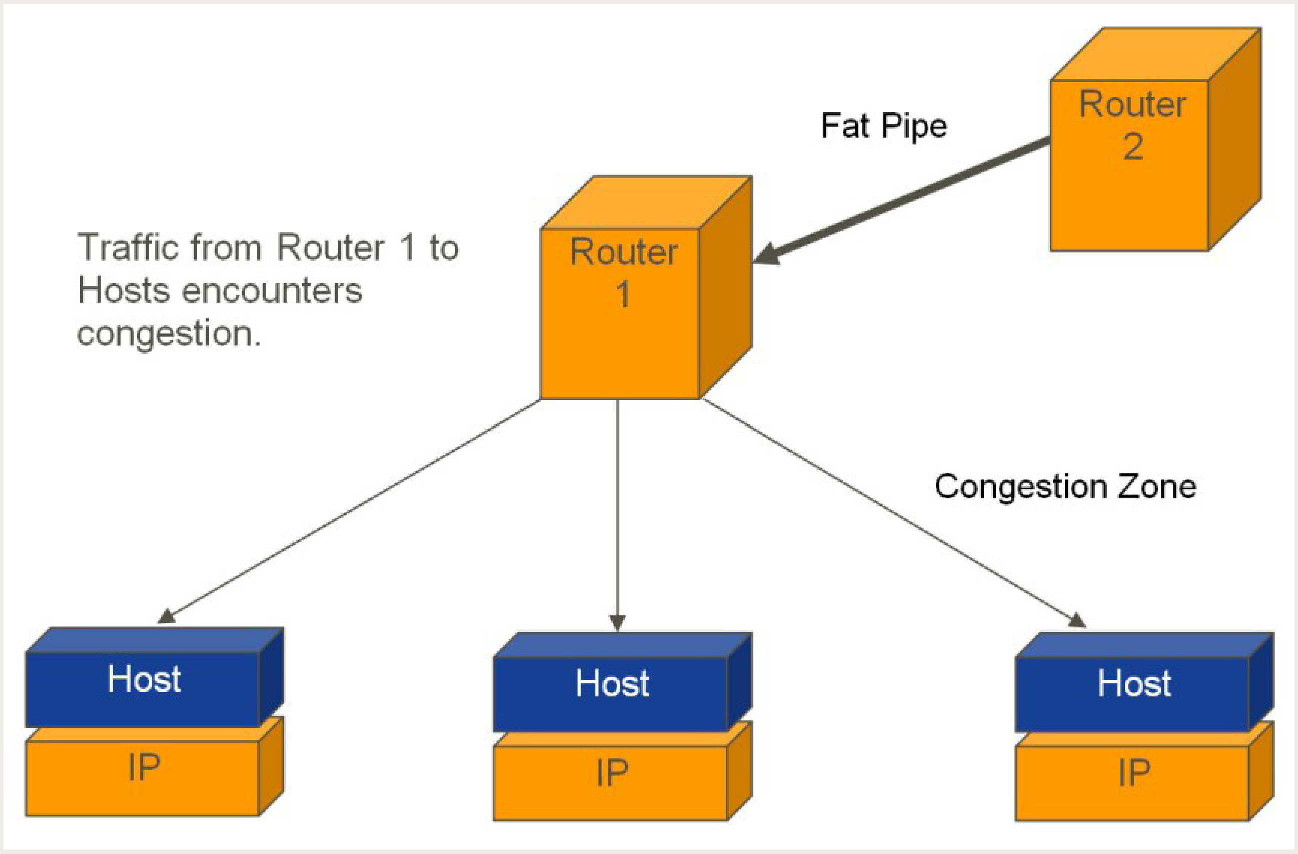
The clumping of Internet traffic is largely a consequence of Jacobson's Algorithm, which causes synchronization of flows through common links: traffic on flows synchronized by Jacobson surges and recedes in unison.

This phenomenon, known as "global synchronization" or "self-synchronization,"[73] is not altered by the addition of bandwidth. Jacobson's Algorithm essentially makes over-provisioning uneconomical, creating the requirement for network-based QoS. Over-provisioning as an alternative to prioritization is a trick right out of the PSTN designer's handbook; its relevance to networks that allocate bandwidth dynamically, like the Internet, is dubious.

Over-provisioning is the deployment of high capacity links at various points in a network. If we assume symmetrical capacity, there is no point in a packet network in which unequal capacity will not create congestion in the opposite direction from the one in which it relieves congestion. In Figure 4, congestion is created between Routers 1 and 2 by equal capacity links throughout; in Figure 5, over-provisioning the link between the two routers creates congestion between Router 1 and the Hosts. Over-provisioning is simply under-provisioning from a different point of view. There is no magic bullet for resource-sharing networks. On the PSTN, over-provisioning is effective because all streams use the same, fixed capacity, but the Internet is not a telephone network.

Isenberg's demand for "one treatment for all kinds of traffic" is infeasible both technically and economically. One treatment implies one charge, which raises the price of large file transfers that are less time-sensitive in the user's own estimation than other types of traffic. "One charge" imposes an idealism tax on mainstream users. In shared-resource networks like the Internet, delay is a consequence of load: the more instantaneous load, the greater the delay. Shared-resource networks attempt to smooth load by queuing. If more packets are offered to a network switch than it can transmit, it stores excess packets in queues until transmission facilities are available. When queue capacity is exceeded, packets are discarded and transmitters must reduce their rates. Delay is caused by the conditions that lead to queuing.

Figure 5: Creating Congestion by Over-Provisioniong



One of the functions of Internet routers is to aggregate traffic from multiple ingress ports onto a single egress port headed to the network's core. The only way to eliminate the requirement for occasional queuing is to arbitrarily limit ingress port speed, an undesirable reduction in performance. Priority queuing provides an opportunity to allocate network resources intelligently: if some packets are more sensitive to delay than others, they can be placed at the head of the queue; if some are more sensitive to cost, they can be placed at the tail. Network systems that can classify packets and queue them intelligently don't need to resort to bandwidth caps and metering schemes; they provide the optimal mix of price, performance, and reliability by design. Isenberg's insistence on the single service level raises the cost of networks with no real benefit to the user.

Isenberg's greatest error was his radical underestimation of the amount of intelligence a network must have in order to appear stupid. A connectionless network has to know where every end system is at all times, or it can't route packets. This capability requires more intelligence in each edge router than the entire telephone network has. Edge routers have to know how to handle everything that can happen, not just what is happening. The Internet is many things, but "stupid" is not one of them.

From the Fringe to the Mainstream

Yet another Berkman fellow, Lawrence Lessig, seeking regulatory principles in the structure of Internet protocols, carried Isenberg's formulation into the academic mainstream. In a 1999 book entitled *Code and Other Laws of Cyberspace,*[74] Lessig developed the idea that "architecture"—in the sense of the design of engineered systems—places constraints on behavior that are similar in power to legal restrictions.

Although Lessig didn't flesh out this notion with respect to the Internet's end-to-end functions (that would come later), he established the notion that network structure enables and disables specific human behaviors. In Lessig's view, Internet protocols were designed to protect privacy and to optimize network efficiency:[75]

> Like a daydreaming postal worker, the network simply moves the data and leaves interpretation of the data to the applications at either end. This minimalism in design is intentional. It reflects both a political decision about disabling control and a technological decision about the optimal network design. The designers were not interested in advancing social control; they were concerned with network efficiency. Thus, the design pushes complexity out of the basic Internet protocols, leaving it to applications, or ends, to incorporate any sophistication that a particular service may require.

Lessig's assertion that the network designers were concerned with efficiency is groundless. As we've seen, the designers of datagram protocols in both France and the United States had no delusion that their systems of layered protocols split across multiple machines bound by unreliable datagrams were "efficient." The reason they adopted this architecture was to give control over network functions to researchers; they were willing to accept a loss of efficiency for the increase in flexibility it gave them.

Lessig's 1999 book "Code and Other Laws of Cyberspace" continues down the trail blazed by Barlow and Isenberg, constructing a mythical history for the Internet and imbuing its structure with magical properties.

Lessig's assertions about the network designers' hostility to "social control" are similarly fanciful. Authoritarian states routinely use the Internet to snoop on their citizens and tailor the information they're able to see; the naïve assumption that the network is transparent makes manipulation of data a piece of cake. TCP may not expect to get packets in order, but network applications assume that TCP delivers the genuine article if it delivers anything at all. Hence, the Internet is just as capable of serving the ends of authoritarian regimes as liberal ones, as we saw recently in Iran's blocking Internet access following the 2009 election controversy. Twitter continued to flow in and out of Iran despite the Internet block because it's capable of bypassing the Internet, thanks to the cell network's Short Message Service (SMS).[76] Datagram network designers could well have been wild-eyed activists, but their work product was sublimely indifferent to social questions.

In sum, Lessig's 1999 book *Code and Other Laws of Cyberspace* continued down the trail blazed by Barlow and Isenberg, constructing a mythical history for the Internet and imbuing its structure with magical properties and its creators with a revolutionary, liberation-based political program. Although the book is insightful in many respects (network efficiency does enhance the social good, but that doesn't mean that the Internet's design is efficient), its flaws remain its most influential legacy.

The ability of the Internet to evolve and to adapt to the needs of new applications and new populations of users is central to its success. It's unwise to declare it a finished system that has to conform to regulations born out of conjecture and speculation in its very early stage.

After publishing *Code*, leaving Berkman, and co-founding the Stanford Center for Internet and Society, Lessig organized a workshop entitled "The Policy Implications of End-to-End" in December 2000 that took his arguments about the Internet's organization to the next level. That workshop connected the dots between the general notion that "architecture is law" and the specifics of Internet architecture as Lessig had come to understand it:[77]

> The "end-to-end argument" was proposed by network architects Jerome Saltzer, David Reed, and David Clark in 1981 as a principle for allocating intelligence within a large-scale computer network. It has since become a central principle of the Internet's design.
>
> In an increasing range of contexts, however, e2e is being questioned. Technologies that undermine e2e are increasingly being deployed; other essential services, such as quality of service, are being developed in ways that are inconsistent with e2e design.

Once again, we have to bypass a significant historical error to get to the assertion: the Internet did not *become* an end-to-end system after the 1981 publication of "End-to-End Arguments in System Design." If anything, the Internet became less "end-to-end", as the comment just cited about "e2e being questioned and undermined" indicates.

"The Policy Implications of End-to-End" workshop took place in 2000—25 years after the Internet was designed, nearly 10 after it had been opened to the public, and nearly 20 years after "End-to-End Arguments in System Design" was written. The erosion of end-to-end that Lessig bemoaned was a rational response to the repurposing of the Internet as a production network heavily intertwined with news delivery, politics, social life, and commerce.

Although it's appropriate to tinker and play with a research tool, general use networks need to be robust, efficient, and reliable. Lessig was told as much by Cisco Systems engineer Christine Hemerick:[78]

> …on this general subject of e2e-ness and whether or not by breaking the e2e model, we create a fatal, ethical, moral sin: I think it's important to remember that a lot of the growth in this whole industry, data networking, has really been driven [by] our ability inside the core network to break the e2e model in ways that would favor economically overall growth scale and development of other applications.
>
> ... [When] we talk about the e2e-ness, there's a lot of focus [on] what's best for a particular end user or a particular sub-class or of user when the trade offs are much more complex than that. Had [Cisco Systems] not been able to implement [bandwidth management] capabilities within the network, nine, ten years ago, a lot of applications that we see running on networks today would never even have been implemented because there were other applications that were not properly behaving and therefore did not give the opportunity for new applications to grow.
>
> … I just reject the notion that somehow e2e is always good and breaking e2e is always bad. The situation is much more complicated than that.

Packet networks create a dynamic in which users and applications compete for network resources, drawing bandwidth on demand from a common pool. Traffic management inside the network seeks to allocate bandwidth fairly or at least rationally. A purely end-to-end network would make bandwidth management difficult, if not impossible, so network operators do what they can to provide fairness. The erosion of end-to-end in the transition of the Internet from research tool to public network is real; Lessig fails to recognize the reasons for it, however. His viewpoint seems grounded in the philosopher-king's desire to enclose the *hoi polloi* in systems which will direct their simple minds toward noble pursuits, and he believes end-to-end is one of them. Consequently, he ignored expert advice and followed up this conference with additional books on threats to the Internet that ignored the major ones,[79] such as traffic inequality, limits to scalability, and malicious code.

Regulators have to distinguish "good discrimination" that enhances social goals from "bad discrimination" that serves less noble ends.

End-to-end arguments are a way to find the best allocation of functions in a distributed system that is thought to consist of multiple layers of communication protocols. Layered models of protocol design guided much of the network architecture work done for packet networks in the 1970s and onward. There are serious drawbacks to conceptualizing network functions in terms of layers when layer boundaries are not drawn correctly, as we find when we implement large-scale networks according to the TCP/IP layering model. Many functions cross layer boundaries, end up in places they shouldn't be, require duplicate implementation or cause a lot of "chatter" across layer boundaries. But these deficiencies weren't discovered until the 1990s and aren't widely understood outside the small community of network architects, as we saw in our review of the RFCs on Internet architecture and shall see again when we discuss recent attempts to move beyond the traditional Internet protocol structure.[80]

Fundamentally, network architects have used layering and end-to-end to ask questions about the assignment of functions in a network system, but in Lessig's hands they become answers to such questions, a dogma that's out of touch with the direction of leading-edge network engineering theory for the past 20 years and a threat to the Internet's fundamental dynamism. The ability of the Internet to evolve and to adapt to the needs of new applications and new populations of users is central to its success. It's unwise to declare it a finished system that has to conform to regulations born out of conjecture, speculation, and intellectual combat in its very early stage.

If, as Lessig claimed, the end-to-end arguments were crucial to the Internet's architecture, it's a mystery why his exploration of them was limited to a single, brief, 20-year-old paper. Surely a program of regulation aimed at protecting the Internet's foundation principle warrants a deeper exploration of the principle.

Network Neutrality: The Birth of a Notion

In 2003, Lessig's protégé, Columbia Law School professor Tim Wu, attempted to refine the debate and advance the campaign for network regulation in a seminal paper introducing the term "network neutrality." In "Network Neutrality, Broadband Discrimination," Wu conceded that a network forced to abide by Lessig's wishes would be less "neutral" and fair than one not so constrained:[81]

> Proponents of open access have generally overlooked the fact that, to the extent an open access rule inhibits vertical relationships, it can help maintain the Internet's greatest deviation from network neutrality. That deviation is favoritism of data applications, as a class, over latency-sensitive applications involving voice or video.

The reason is that a network bereft of the "idiot savant behaviors" with which Isenberg had struggled would by its nature favor applications that didn't need such services—specifically, content-oriented applications such as file sharing and static Web pages. Applications impaired by "stupid network" design span the space inhabited by real-time and communication applications such as VoIP, teleconferencing, sensor networks ("the Internet of things") and gaming, but also the "bulk data" space of applications that need to move enormous amounts of data with low cost. The one-size-fits-all delivery service impairs real-time applications technically and bulk data applications financially, as the network has to provide them with a higher service quality than they require, which has a cost.

Wu's paper proposed not a ban on active management but a system of strict scrutiny forbidding network operators from engaging in what he calls "discrimination" at their networks' Internet interface. Wu defines "discrimination" as priority routing. On its face, a ban on discrimination is very appealing: the civil rights movement taught us all that discrimination is bad in the social context,[82] so it must be bad in the network context as well.

Wu's 2003 paper "Network Neutrality, Broadband Discrimination" tacitly expresses the wish that the Internet were not what it is.

Unfortunately, Wu's attempts to define discrimination have him chasing his tail. Fundamentally, regulators have to distinguish "good discrimination" that enhances social goals from "bad discrimination" that serves less noble ends. Wu's paper lists a number of instances in which a network behavior that places an inordinately high traffic load is restricted by the network operator; he judges all of them discriminatory. On shared medium networks with dynamic bandwidth allocation (of which the Internet is one), heavy users of network bandwidth increase delay for others unless the network prioritizes based on application requirements (Isenberg's "idiot savant behavior.") This is exactly the anti-neutrality inherent in unmanaged networks that Wu claimed to reject. Networks can't be said to be neutral with respect to application requirements if they allow high bandwidth applications to degrade other ones.

Wu proposes a system in which network operators are free to police traffic on their residential networks in any way they please, as long as they don't manage traffic entering or leaving the portals that connect residential networks with the Internet core:[83]

> What follows is a proposed antidiscrimination principle (a rule, only if necessary). The effort is to strike a balance: to forbid broadband operators, absent a showing of harm, from restricting what users do with their Internet connection, while giving the operator general freedom to manage bandwidth consumption and other matters of local concern. The principle achieves this by adopting the basic principle that broadband operators should have full freedom to "police what they own" (the local network) while restrictions based on inter-network indicia should be viewed with suspicion.
>
> This non-discrimination principle works by recognizing a distinction between *local network* restrictions, which are generally allowable, and *inter-network* restrictions, which should be viewed as suspect. The principle represents ultimately an effort to develop *forbidden* and *permissible* grounds for discrimination in broadband usage restrictions.

Wu creates a zone where discrimination is permissible—services delivered and consumed strictly within the ISPs network; and a zone where it's not permissible—services delivered over the public Internet. From a technical perspective, Wu's separation of permitted discrimination from non-permitted discrimination works only if the residential network and the Internet portal have precisely equal capacity and performance. In order for the portal to be immune from management, it needs the capacity to deliver both inbound and outbound traffic without delay, no less and no more. This situation rarely occurs because the Internet is a dynamic system in which slower links are aggregated onto faster ones according to statistical estimates of peak traffic load. Any attempt to impose mandates for static allocation of bandwidth must be viewed with suspicion as they undermine networking economics as profoundly as Isenberg's insistence on the single service level would. It's also not reasonable to demand equal performance for Internet-based services and locally based ones; the Internet itself introduces delay and variation in service delivery over which the ISP has no control. The further a packet has to travel, the more likely it will encounter congestion.

Wu insists on an unmanaged Internet portal because he wants to have it both ways. ISPs manage broadband network services in order to achieve more predictable behavior than the Internet offers. A telephony or television service offered over a broadband network has to operate within performance bounds similar to those of the dedicated networks that were built for these services in the past. Internet access operates on a much looser set of performance boundaries, of course. But Wu believes that an unmanaged Internet portal can provide a high-performance path between service providers on the Internet and ISP customers that will permit greater consumer choice with respect to telephony and television services. This can only be true as a fact of engineering if the Internet portal is managed: it needs to assign higher priority to real-time streams than to those that carry content that's not time-sensitive. The demand for an unmanaged portal actually frustrates the goal that Wu wants to achieve

Wu's "Network Neutrality, Broadband Discrimination" tacitly expresses the wish that the Internet were not what it is; it hides the "stupid network" formulation in an unmanaged gateway, the precise place in which Internet engineers say active management code must reside. While its regulatory framework was a failure, Wu's insight that the absence of traffic management in private networks actually increases the network's structural bias was an enormously helpful contribution to the debate on network regulation. Less helpful is his attempt to draft legislation from the functional layering model.

Functional Layering as a Network Regulatory Model

After Lessig and Wu, a large number of Cyberlaw theorists and policy advocates jumped aboard the functional layering bandwagon with papers proposing systems of function-based network regulation following from their understanding of network structure. These arguments presume that the functional layering models of network protocol structure developed in the 1970s—such as the ISO Reference Model and the TCP/IP model—are complete, correct, and inviolable. Richard Whitt, then with MCI, made the following typical claim for the layering regulatory paradigm:[84]

> Layers analysis offers a fresh and compelling way to look at legacy U.S. legal and regulatory policies. In the e-commerce world, where ISPs face the prospects of legal liability for the actions of third parties using their network, a horizontal layers framework exposes the very real dangers of overzealous and ill-reasoned regulation at disparate layers. In the world of telecommunications regulation, the framework provides a valuable new mechanism for assessing harm from the exercise of undue market power, and suggesting practical alternatives. In particular, a layers model can assist policymakers in targeting regulation to foster needed competition at the core (or lower layers) of the network, while preserving and enhancing innovation at the edge (or upper layers). Well founded on *fundamental engineering principles*, and buttressed by economic analysis, the layers principle, as expressed in this Article's notion of a Network Layers Model, should be adopted by policymakers as a necessary and productive public policy tool. [Emphasis added]

Similar claims were made by Lawrence B. Solum and Minn Chung,[85] Yochai Benkler,[86] Kevin Werbach,[87] Rob Frieden,[88] and several others. The discovery of functional layering caused as much excitement in the regulatory policy community as it had in the engineering community 25 years previousy.

The network engineering community increasingly regards 1970s functional layering models as fundamentally flawed.

Unfortunately for the regulatory debate, scholars and advocates tended toward a common misunderstanding of layering in which layers function in relative isolation from one another, at most affected by lower layers. In real networks, applications are affected by the ability of networks to move information between systems on the application's behalf, but networks are also affected by the quantity of information that applications wish them to move. Applications are the source of nearly all the traffic that moves through a network (network functions move a bit of housekeeping information) and are therefore primarily responsible for the effects seen by other applications. The most that network functions can do for applications is move delays around according to some predetermined policy. Despite the fact that applications are ultimately in charge of any network, layering theorists placed ever-greater regulatory burdens on "lower" functional layers and attributed "innovativeness" exclusively to applications. The relationship between innovation and service is actually much more interconnected than this simplistic analysis suggests.

More important is the fact that the network engineering community increasingly tends to regard 1970s functional layering models as fundamentally flawed and a barrier to progress. This sentiment was expressed cogently in RFC 3439[89] (and the works it cites going back to the early 1990s) and continues to come up in architecture discussions. A recent e-mail thread on the IETF's tsv-area list on the distinction between application and transport protocols says the following:[90]

> The distinction between these two layers (transport and application) is quite blurry when it comes to "applications" that end up being used as middleware for other applications. Application protocols stacked on other application protocols 3 or 4 deep is no longer uncommon, and definitely makes our layering models look ridiculous. Personally, *the model we have where layers have particular names ("link", "network", "transport", etc.) has been useful to get this far, but needs to be thrown away to progress.* Does anyone really enjoy explaining to their interns and students that IPsec is layer 3 1/3 and MPLS is layer 2.5 after they've only just begun to grok the basic layered model?
>
> To me, if a piece of code is controlling transmission in terms of things like timing, segmentation/reassembly, error checking, ACK, retransmission, on behalf of some set of applications, then it smells a lot like a "transport." John Day[91] has noticed that the same thing can be said about part of the link layer, and realized that these are all just different flavors of the same flow and error-control parent class in his PNA. This fits reality much better than giving strict names to layers at what have now become arbitrary points within the stack. [Emphasis added]

At a time when network engineering is re-thinking the 30-year-old functional layering model that was never more than a conjecture in favor of simplified, recursive models, Cyberlaw scholars and activists are trying to freeze it in law as a *fundamental principle*. This disconnect should be cautionary to network architects: every time an architect sneezes, apparently, a 20-year chain of events is triggered that will ultimately result in a policy advocate insisting on a new regulation.

While these regulatory debates were taking place, the Internet was undergoing the most profound transformation in its history, the transition from a tool for network research to an open, commercial system available to anyone with a personal computer and a modem.

BIRTH OF THE PUBLIC INTERNET

From its advent in 1972 as part of the design of CYCLADES to its new role as the arbiter of good and evil in Cyberlaw, the end-to-end principle underwent a profound mutation. What was initially a mode of operation that facilitated experimentation and network diversity, enabling engineers to discover effective means of optimizing networks, had become a prohibition on the very engineering practices that network architects had always considered part of their mandate. Although this transformation was caused in part by nervousness about potential practices by telecom carriers and cable operators, it was accompanied by the transition of the Internet from a research network to a worldwide system used by hundreds of millions of unsophisticated users.

A mode of operation that initially facilitated experimentation had become a prohibition.

The early research networks were funded and built as learning vehicles, but the advent of the personal computer created the need for a general-use network to interconnect them all. There were four candidates to serve this need: Digital Equipment Corporation's DECNet, the Xerox Network Services (XNS)[92] architecture, the Open Systems Interconnection (OSI)[93] protocol suite created by the International Organization for Standardization (ISO), and TCP/IP.[94]

By the time Tim Berners-Lee implemented his World Wide Web we were 10 years into the PC era, and the XNS protocol suite was the accepted standard in network protocols. Vastly more popular than TCP/IP,[95] XNS was the standard upon which most corporate and university Local Area Networks (LANs) were deployed: versions of XNS were found in the LAN products of Novell, 3Com, Banyan, and Ungermann-Bass, the four dominant LAN companies of the 1980s. Digital's DECNet had been deployed by the company's customers for nearly 15 years, and had devised solutions to the common problems of congestion control and route synchronization. The OSI protocols were little used, despite a token investment by the U.S. Department of Commerce, because they were perceived to be inefficient, overly complex, and difficult to implement.[96]

If the choice of a protocol suite had been made on purely technical grounds, we'd be using a variant of XNS now (if not Digital's DECNet, or OSI, or CYCLADES itself). XNS properly distinguished the three principal objects in a network system, names, addresses, and routes, which TCP/IP lumps together. XNS was capable of supporting multi-homed services, mobility, and network-local addressing, three critical features that challenge TCP/IP. Berners-Lee happened to implement his Web on the TCP/IP network installed at CERN, his place of employment when he wrote the code. CERN's TCP/IP had only been connected to the Internet one year earlier, and this connection must have helped inspire Berners-Lee to build his system of networked hyperlinks. CERN in any case was the main Internet portal in Europe, connected to the United States by a T1 line subsidized by IBM.[97]

The Internet became the dominant packet network because it was in the right places at the right time.

Shortly after CERN connected to the Internet and Berners-Lee began fiddling with his Web, the Internet began to open up to the general-use marketplace. The first Commercial Internet Exchange (CIX) opened in 1991, connecting PSI, UUNet, and CERFnet. The Delphi bulletin board opened an e-mail gateway to the Internet in 1992, AOL and CompuServe followed shortly thereafter and by 1995 commercial use of the entire Internet was finally allowed.[98] By the time the last restrictions were lifted on Internet use, the Web had become a powerful pull and the online services all provided access to it. It was the existence of a network infrastructure and the combination of email and the Web that made TCP/IP the international standard for packet switching, not the structure of the protocols themselves.

The argument is frequently made that it would have been impossible to build the Web on a network that didn't have an end-to-end structure, but there's no reason why this should be the case. Had CERN connected to an NCP ARPANET instead of to the TCP/IP Internet, the Web would have been no harder to implement.[99] The Web is composed of client and server application programs communicating through virtual circuits, not customized transport protocols. NCP ARPANET was perfectly capable of supporting application programs: that was its primary purpose.

TCP/IP became the *de facto* standard for global packet communications in somedegree because the applications that ran on the Internet—e-mail and the Web—were more compelling or easier to deploy than those that ran on the competing networks. But more significantly, it had the advantage of being backed by the U.S. Department of Defense, which had paid a series of contractors to produce reference implementations of its protocols. OSI was supported by the U.S. Department of Commerce, but it didn't provide a comparable level of funding, leaving software to the private sector. Xerox didn't profit from the use of its XNS protocols in LAN products, and apparently saw no advantage in having its technology adopted as a worldwide standard; Digital Equipment Corporation offered DECNet to ISO, but was rejected in favor of a home-grown approach that was quite heavily influenced by DECNet.[100]

But the most important reason TCP/IP won the protocol wars had less to do with design than with material reality: unlike CYCLADES, XNS, DECNet, and OSI, TCP/IP had a far-reaching network infrastructure. The Department of Defense extended ARPANET to Europe in the early 1970s, and the National Science Foundation extended its NSFNet to Europe and Japan by the time the World Wide Web was created at CERN, while XNS and DECNet networks ended at corporate boundaries and CYCLADES had been decommissioned by the French government.

The operator of NSFNet provided a commercial service called ANS, and other commercial networks such as PSI, CERFNet, and UUNet had connected for-profit enterprises since the 1980s. In 1991, they formed the Commercial Internet Exchange (CIX) and connected with each other without settlement fees or payments to ANS. There were two Internets in that era, one supported by NSFNet and bounded by an Acceptable Use Policy forbidding commercial activities and another supported by mutual agreements among ISPs and explicitly friendly to commerce. The two Internets were connected to each other in an arrangement reminiscent of Hong Kong and mainland China, as one network with two acceptable use policies.

By the time the Web was ready for wide-scale deployment, there was one and only one worldwide infrastructure upon which it could be deployed, a significant portion of which was provided by commercial concerns. The advent of the Web caused the Internet to undergo yet another period of transformation.

The End-to-End Tax: "World Wide Wait"

Before the Web, commercial Internet use was driven by the desire to use e-mail across company boundaries. E-mail is the least challenging of network applications from an infrastructure perspective as it tends to consist of short files with extremely loose temporal delivery requirements: it doesn't matter whether your e-mail correspondent receives your message several seconds (or even minutes) after you send it, as long as it's delivered. It's the classic example of the "data application" Tim Wu says is favored by non-discrimination rules. The rise of the Web and the clamoring among customers of time-sharing services such as AOL and Delphi to use it created a pair of new problems for the Internet infrastructure.

TCP's strengths manifest themselves the longer the door remains open.

The first was easily predictable: unlike early Internet e-mail which as text only, Web pages contained graphic images that made them larger than e-mails by an order of magnitude. Large pages meant more traffic, and the interactive nature of the Web demanded that this increased volume of traffic be moved quickly. Dealing with an increase in traffic is easy: network operators simply obtained higher-speed links, just as the research Internet had increased its capacity by trading its early 56 Kb/s mesh for the 1.44 Mb/s (and ultimately much faster) NSF backbone. But upgrading link capacity didn't generally result in better perceived performance: Web pages on fat pipes didn't seem to load much faster than those on skinny pipes. Access to the World Wide Web quickly became an exercise in a frustrating "World-Wide Wait." Analysis of slow Web access by the W3C Consortium, the keepers of the Web standards, pointed the finger at poor interaction between the Web's Hypertext Transfer Protocol (HTTP) and TCP:[101]

> One of the goals of W3C Activities is to "save the Internet from the Web"—to alleviate Web-induced bottlenecks that might otherwise stifle further Web growth. To appreciate how some bottlenecks have been addressed, we must delve into the HTTP protocol, Cascading Style Sheets, and Portable Network Graphics.
>
> …To function efficiently, HTTP must take advantage of TCP/IP's strengths and avoid its weaknesses, something that HTTP/1.0 doesn't do very well. For example, in HTTP/1.0, every URI that appears in an HTML document initiates a new request by a client to connect to a server; such requests are part of TCP. Even if ten URIs in a row refer to the same server, it is not possible to "keep the TCP door open" for more than one request. Since TCP's strengths manifest themselves the longer the door remains open, it makes sense to eliminate unnecessary TCP open requests. After all, why spend all this time establishing a connection to a Web server only to close it and open another? HTTP/1.1 addresses this issue with an open door policy known as *persistent connections.*

The phrase "TCP's strengths manifest themselves the longer the door remains open" refers to one of the consequences of Jacobson's Algorithm: TCP connections now begin in a condition called "Slow Start", effectively the same as the "one packet at a time" condition of the old NCP ARPANET. Jacobson's Algorithm behaves this way in order to protect the Internet from overload, but the algorithm manifests a weakness inherent in nd-to-end management of congestion: each TCP connection comes into the world ignorant of the Internet's congestion state; it must learn the network's state from experience, and by the time it has gathered this knowledge it may be unable to apply it. As HTTP 1.0 opened and closed TCP connections willy-nilly (one for each graphic image, style sheet, and external reference plus one for the page's text), they didn't stay open long enough for Jacobson to graduate from Slow Start to full speed. Connections in Slow Start don't get much faster with fatter pipes; they're throttled by the round-trip transit time as well as processing delays between browser and web server, a figure that's often dominated by network-independent factors. Slow Start is a condition where the server application has to stop and wait for replies from the client, even though network capacity may be available for more efficient use; it's an inefficiency directly related to the time end-to-end functions spend learning about network conditions and performing application-oriented processing functions.

Tim Berners-Lee didn't design HTTP 1.0 the way he did because he had a desire to increase frustration on the part of web users, he designed it simply, in accordance with his cursory knowledge of the Internet protocols. Alleviating the World-Wide Wait required a redesign that added considerable complexity to web software, HTTP 1.1. The complexity of HTTP 1.1 was thus a side-effect of Jacobson's Algorithm, in effect an end-to-end tax in the form of a protocol redesign to align the application with transport protocol behavior. While Jacobson's patch was an expedient solution to the congestion collapse induced by ftp-over-Ethernet, it impaired the next big new application down the line.

The problems with adapting the Web to the Internet illustrate a primary vulnerability of end-to-end networks, the fact that application programmers need to have quite detailed knowledge of the network's traffic dynamics to avoid creating pathological conditions. In theory, functional layering is supposed to isolate network concerns from application design; in practice, new applications often produce dramatic side effects on network routers. Single-function layers are much more interdependent than independent.

Internet Innovation Cycle

The Web's impact on the Internet and the engineering community's response to it teach a lesson about the real dynamics of innovation on the Internet. Net neutrality advocates enjoy platitudes about the Internet enabling "innovation without permission," but the reality of major innovation is much more complex. Each new style of application alters the mix of traffic on the Internet, which is always tuned to support a particular range of traffic mixes reasonably well. In response to the new application, operators make adjustments in provisioning, and recommend changes in the application. When the applications make these changes, things work reasonably well, but not until. So the cycle is like this:

1. Internet is in equilibrium.
2. New application alters the mix.
3. Internet operators adjust to application.
4. Application adjusts to the network.
5. Equilibrium is regained.

This is precisely the cycle that saved the Internet from congestion collapse in the 1980s and from the World-Wide Wait in the 1990s: the Internet changed, and so did the application. A similar evolution is taking place around P2P file transfer applications, which will have to be domesticated to stop destabilizing the Internet.[102]

CHALLENGES TO THE INTERNET ARCHITECTURE

The Internet is a thoroughly impressive feat, interconnecting some 1.5 billion users worldwide and continuing to grow. More than two-thirds of the people in North America and Western Europe are already connected to the Internet, and penetration continues in the rest of the world. The four billion mobile phones deployed worldwide are being upgraded from simple phones with voice and SMS capability to smart phones capable of accessing the Web, and Internet service has sprung up all over the world thanks to Internet cafes and access via wireless links. The Internet faces a number of critical challenges to growth, not to mention day-to-day management.

Denial of Service Attacks

A network such as the contemporary Internet that is open to new applications and uses is inherently vulnerable to abuse. A network that's open to innovation, dependent on expert administrators, and largely unsecured—as the Internet was before it was opened to the general public—is even more vulnerable. Malicious code is among the *innovative new applications* the Internet has fostered, so putting an end to Denial of Service attacks, spam, identity theft, digital piracy, and botnets without sacrificing the flexibility to experiment with new applications is among the chief challenges facing the Internet.

Traffic Jams

Growth in the user population stresses the core links that carry Internet traffic across and between continents, dictating billions of dollars of annual investment simply to keep up with increased demand for ordinary services, and additional investment to provide new data-intensive services such as online video. Some governments are reacting to the collapse of the global financial system by funding infrastructure programs to increase the capacity of local access networks. The success of these projects will create a need for more capacity all the way from first-mile access networks to the regional aggregators known as the "middle mile" to the optical core and on to the transcontinental undersea cables. This activity will almost certainly result in an increase in network traffic[103] that exceeds the capacity of Moore's Law efficiencies to accommodate without increased investment. This problem is compounded by the fact that a small minority of Internet users account for the majority of usage. But traffic growth is not the most significant threat to the Internet, scalability is.

Address Unknown

The demand for Internet connections far outstrips the capability of existing Internet protocols to support such basic functions such as addressing and routing. Only 20% of the earth's population is connected to the Internet today, and well over half of all the addresses that the Internet Protocol can ever use are currently in use. The design of IP insists that addresses be globally unique, so the existing pool can't be cleverly reused in local pools as addresses in local area networks are reused by Network Address Translators (NATs).

The Internet's technical overseer, the Internet Engineering Task Force (IETF) proposes that the existing Internet Protocol, IPv4, be replaced by a new version, IPv6, which provides much larger addresses (128 bits vs. 32 bits). Ten years after its designation as the official successor to IPv4, IPv6 remains a curiosity in the United States and Western Europe, but is widely used in China. Part of the problem is the failure of the IETF to design IPv6 as a compatible upgrade to IPv4; transition is painful.

While the 128-bit address in IPv6 solves the "address exhaustion" problem, it creates another one that's at the same time more technically obscure and more difficult to resolve.

The Proliferation of Prefixes

There's reason to believe that the current addressing and routing architecture of the Internet is on a collision course with Moore's Law. The pool of addresses that can be used by the common version of the Internet Protocol, IPv4, will be exhausted in the next few years, at which point we will replace it with IPv6. While IPv6 will alleviate the address exhaustion problem, it will aggravate a more serious problem affecting the Internet, namely that of route fragmentation.

The global Internet currently consists of 1.5 billion computers on 60,000 distinct networks known as "Autonomous Systems" (ISPs, commercial firms of other types, public sector organizations, or multi-homed hosts) bound together by a system of 300 Internet Exchanges in which 300,000 routing prefixes are exchanged using BGP. The number of routes is significantly less than the number of computers because Autonomous Systems aggregate routes to computers whose addresses fall in consecutive ranges. If an ISP owns addresses 68.86.0.0 to 68.86.255.254, it only needs to advertise one route for each address in the range at the exchange points at which it accepts packets. Address exhaustion causes addresses to be allocated by Regional Internet Registries (RIRs) such as the American Registry of Internet Numbers (ARIN) in smaller groups, increasing the number of routing prefixes that must be exchanged. One of the objectives of IPv6 is to allocate Provider-Independent (PI) addresses so that firms can switch ISPs without renumbering systems. PI addressing defeats route aggregation, the principal method used to keep the number of routes that have to be remembered by edge routers manageable.

The net result of IPv4 address exhaustion, the transition to IPv6, and PI addressing is massive growth in the number of routes that have to be exchanged, retained, and processed. The semiconductor parts that do this processing are already outside the scope of Moore's Law as they're specialized, high-performance components not produced by the factories that reap the benefits of improvements in manufacturing processes; they're not going to get faster and larger at the rate we need in order to stay ahead of route proliferation.

So the cost of Internet routers is set to go up, perhaps alarmingly, in the near future, making it even more difficult for small ISPs to be competitive. These issues were discussed at an Internet Architecture Board Workshop on Routing and Addressing in 2006, which concluded:[104]

> The clear, highest-priority takeaway from the workshop is the need to devise a scalable routing and addressing system, one that is scalable in the face of multihoming, and that facilitates a wide

> spectrum of traffic engineering (TE) requirements. Several scalability problems of the current routing and addressing systems were discussed, most related to the size of the DFZ [Default Free Zone] routing table (frequently referred to as the Routing Information Base, or RIB) and its implications. Those implications included (but were not limited to) the sizes of the DFZ RIB and FIB (the Forwarding Information Base), the cost of recomputing the FIB, concerns about the BGP convergence times in the presence of growing RIB and FIB sizes, and the costs and power (and hence heat dissipation) properties of the hardware needed to route traffic in the core of the Internet.

The Internet's leading engineers and researchers believe concerns about the scalability of the routing system are real and will be aggravated by the addition of smart phones to the Internet, with a potential tripling of the Internet device population.

As the Internet continues to evolve into a multi-service, general use, reliable network, it will adopt the management mechanisms necessary to provide enhanced services. It's no longer adequate to rely on a network heavily optimized by design for short file transfers, we need a network that can support a rich set of real-time services such as telephony and conferencing on the one hand and large file transfers on the other. The means by which to do this are well understood. But in the final analysis, the current design of the Internet protocols contains the seeds of its own obsolescence. The IPv4 address pool is limited, the Internet's routing methods aren't scalable, and a one-size-fits-all delivery specification isn't adequate. There are several examples of these problems.

LTE Needs Quality of Service for Voice

Long Term Evolution (LTE)—the 3rd Generation Partnership Project's (3GPP) 4G wireless network of choice—is on its face a triumph for the Internet—an all-IP network that will ultimately support voice as well as video streaming, Web access, and bulk file transfer service. But it's not going to attempt this feat of convergence with the single, best-effort delivery class that we're used to on the consumer Internet. LTE is intended to use a system called IP Multimedia Subsystem or IMS. IMS is a work in progress, currently at revision 8, intended to allow full interoperability within and between IP-based wireless networks without sacrificing the QoS requirements of telephony or the billing opportunities that carriers depend on. IMS is a controversial system with net neutrality advocates since it violates the more radical definitions of net neutrality, giving special treatment to voice packets over all others and creating new opportunities for revenue collection.

InterStream Needs Quality of Service for Video

InterStream is a system that's intended to achieve high-quality delivery of streaming high-definition video over the public Internet through a system of expedited routing and edge-caching. InterStream provides premium delivery for premium content, and as such, may offend some ideas about end-to-end neutrality. It's appealing to content producers, though, because it provides a positive incentive for consumers to refrain from downloading pirated movies: InterStream offers a high-quality viewing experience on-demand, for a reasonable fee and largely eliminates the need for Deep Packet Inspection-based piracy mitigation.

RINA Rethinks Protocol Layering

The Recursive Internetwork Architecture—RINA—reconstructs the overall structure of Internet naming, addressing, routing, and functional decomposition along a model entirely different model from the model that evolved in the Internet.[105] Instead of breaking down network protocols into a "layer cake" structure in which each layer is performs a different function, RINA constructs a single recursive layer that repeats at different ranges of resource allocation and scope.

RINA might translate into one scope within a computing device; another within a private network; another across the ISP network; another across the public network core; and so on. This design makes solutions to such problems as attack mitigation, multi-homing, mobility, and security trivial to implement, because it provides an interchange function at the border of each administrative boundary where policies can be easily implemented. RINA replaces the "hour glass" structure of the current Internet protocols with a "barbell" structure, providing interoperability to diverse applications and network media through a common pair of protocols implemented multiple times.

RINA aims to partition resource allocation, but not functionality as the layer-cake system does. RINA might have a customized layer to handle extremely high-speed networks, for example, but its overall function would be the same as the handler for low-speed networks.

The fundamental insight of the RINA model is the realization that networking is a generalization of the inter-process communication (IPC) that takes place inside modern operating systems. An application requesting network communication is just doing IPC with somewhat longer delay. In both cases, the primary function is to pass messages between processes that happen to be running on different machines. When processes utilize IPC within the scope of their host system, they don't have to know the physical memory address of the message or the identifier of the companion process; they supply logical identifiers and rely on the operating system to do the rest. In network IPC, the process should be the same.

In operating systems, we learned that at least three levels of names were required: application names, logical addresses and physical addresses. The Internet only has the latter. This is at the root of most of the scaling problems in the Internet today. RINA recognizes that given the greater scope both in resources, numbers, and physical range and that no set of specific algorithms and parameters can effectively cover the entire range uses layers with the same protocols and different policies to manage smaller ranges. This same repeating structure also allows the scaling issues in routing table growth to yield to a divide and conquer strategy.

In principle, the RINA model could be implemented with TCP/IP, but with some loss of functionality. In practice it would be more likely to employ a protocol modeled on something like Richard Watson's Delta-t[106] which fits the model better and has greater simplicity and reliability.

RINA solves the problems of route proliferation previously mentioned, to the extent that it enriches the existing base of IP processing equipment. RINA, and similar but more traditional efforts such as the Information Sciences Institute's Recursive Network Architecture (RNA),[107] a traditional functional decomposition, raise an important issue for regulation of the current Internet: there's a strong desire among some Internet regulators to mandate specific handling of IP datagrams. What happens to this regulatory approach when an entirely different protocol begins to circulate on the infrastructure currently dedicated to IP? This eventuality is almost certain to happen within the next few years, so the regulatory issues it raises are urgent. In fact, a great deal of the inter-ISP traffic in the Internet's optical core is carried today by non-Internet protocols such as ISO CLNP and DECNet IS-IS.

We need a regulatory framework for the Internet that promotes innovation inside the network as well as outside, and discourages bad behavior by operators, application providers, and users alike.

The importance of RINA is that a recursive IPC model of allocating resources creates a more vibrant market environment that naturally encourages competition and innovation, fundamentally altering the regulatory environment. Advances in network architecture support innovation and increase competition in their own right.

LESSONS FROM END-TO-END'S EVOLUTION

In Louis Pouzin's original formulation, network engineering was a process of ongoing experimentation consisting of design and deployment followed by either standardization or redesign. He created a sandbox for experimentation on network protocols in the CYCLADES network by employing datagrams in his network infrastructure and virtual circuits in the host computers attached to the network. He believed this division of labor would allow the greatest degree of freedom to experiment with network protocols, and provide the best guidance as to which functions needed to be provided in the core of production networks and which were best left to the hots.

One of the unfortunate consequences of the connection of ARPANET to Western Europe and the rise of X.25 packet network service by the European PTTs was the decision by the French government to withdraw funding for CYCLADES, as TCP/IP thinking was well behind Pouzin. The second phase of protocol experimentation—core deployment for necessary functions—was overlooked by the TCP/IP community until the failure of "stupid network" control structures to work well over long, fat pipes, in the presence of malicious hosts, and in the absence of routing architecture forced a reconsideration. Internet engineering is now beginning to embrace an iterative Pouzin process.

Broadband network transport services create opportunities for innovation.

Legal theorists and policy advocates haven't kept pace with changes in the understanding of network architecture that have taken place since the publication of "End-to-End Arguments in System Design" in 1981. This is unfortunate, as they continue to argue for network operation mandates that have long since been discarded by the Internet operations and engineering community. It's in the public interest for the Internet to retain its dynamism and vibrancy, its ability to absorb upgrades that allow it to support an ever-more-diverse mix of applications without sacrificing cost and performance.

End-to-end arguments, however we define them, are fine as far as they go, but they don't go far enough to promote the same degree of innovation in network-enabled services as they do for network-independent applications. At the same time, it's inappropriate to allow network operators unbridled freedom to operate networks as they see fit. Too many people depend on the broadband Internet to connect them to sources of information and interactions that are vital to their lives. We need a network regulation framework that promotes innovation inside the network as well as outside, and discourages bad behavior by operators, application providers, and users alike. This is no easy task.

While the approaches that advocates of net neutrality have proposed are defective with respect to Internet architecture, they do at least represent an effort to regulate according to essential properties. Other regulators have simply tried to extend a system of constraints devised for monopoly telephone networks onto the Internet, an approach that does violence to its dynamic nature: Congressman Ed Markey's various Internet Freedom Preservation Acts fall into this category, as they would foist telephone network regulations on the Internet. The Internet isn't simply a telephone network with greater bandwidth; it's an entirely different approach to telecommunications. Hence, Internet regulations need to depart from the telephony model. The best option is to look toward non-technology-based frameworks, such as those developed in competition and consumer protection contexts.

In Search of a Regulatory Framework

Telecommunications regulation in the United States is shaped by an approach that's based on "regulatory silos" in which a unique set of rules loosely based on traditional notions of common carrier obligations is devised for each new communications technology. Former FCC commissioner Kathleen Abernathy explains:[108]

> In a world where multiple competitors offer bundles of IP-enabled services over broadband platforms—including voice, video, and data services—what is the appropriate regulatory framework? Should we continue to apply traditional common carrier principles at the federal and state level to wireline telecom carriers and perhaps *extend* such requirements to other network owners, such as cable operators, wireless broadband providers, and BPL system operators? Or should policymakers phase out these traditional forms of regulation—with the old regulatory silos for wireline, wireless, cable, and satellite services—and adopt instead a far more streamlined regulatory framework that concentrates on core social policy obligations? I have consistently argued that the latter approach is more rational, better for the industry, and, most importantly, better for consumers.

Regulatory silos create an inconsistent patchwork of telecommunications rules that invites special-interest intervention in the regulatory process and delays the adoption of new technologies. The functional layering model of network regulation proposed by network neutrality advocates is an attempt to correct the inconsistencies in the regulatory silos model, but simply rotates the silos 90 degrees to the right, replacing vertical stovepipes with horizontal ones. Functional layering advocates consult the Open Systems Interconnection Reference Model (OSI/RM)[109] to learn the proper roles of network layers, and seek to freeze these descriptions into regulation. This approach makes several critical errors, as we've explained. The OSI/RM was a conjecture about how functions might be modularized that was devised in the absence of experience. Modular implementation is sound engineering practice, but only after experience has shown how functions fit together. Ongoing experimentation with network implementation and operations tell us that we don't have the answers yet.

Striking the Balance

A more rational approach to network regulation is to recognize that networks are platforms for the delivery of services to people. Some of the services delivered by residential broadband networks are well understood, such as IP transport services, TV programming, and telephony. Some of these services, IP in particular, act as additional platforms for the delivery of additional services by third parties, such as social networks, search, auctions, and content delivery.

Bandwidth caps can be relaxed if the network is free to delay high bandwidth file transfers under conditions of congestion.

The addition of new transport services to a broadband network increases the opportunities for innovation by third parties. An enhanced IP transport service with limited latency would, for example, enable third parties to deliver wideband video conferencing reliably, regardless of transient network conditions that would otherwise impair it. Such transport services would also improve the experience of interactive gamers and improve the quality of telemedicine applications. At the other end of the latency spectrum, wireless providers tend to cap and meter high bandwidth applications in order to reduce overall network load. Bandwidth caps can easily be relaxed if the network is free to delay high bandwidth file transfer applications under conditions of congestion. If network operators are permitted to sell two, three, four, or more levels of IP transport services instead of one, innovation wins.

If innovators do their jobs, new services will emerge on these networks faster than regulators can anticipate them, but an effective, general purpose framework makes crafting of specific rules for new services unnecessary: the framework would rely on universals which can be expressed in a simple test, where most of the following conditions need to be satisfied:

1. Does the service perform as advertised?
2. Is the service offered for sale in a reasonable and non-discriminatory manner?
3. Does the service increase competition or consumer choice?
4. Are the specifics of the service available to third party innovators?
5. Does the service promote the public interest?
6. Does the service unfairly discriminate against other companies to benefit the company introducing the service?
7. Does the service promote improvement in network infrastructure?
8. Is the service consistent with current or emerging network standards?

The strongest argument for the services approach to network regulation is that it conforms to the dynamic and evolving nature of information networks better than any other approach. Rather than seeking to extract maximum public benefit from aging and obsolete networks, it promotes evolution toward better ones.

CONCLUSION

Network neutrality advocates argue that the Internet as we know it lives on borrowed time without the regulations they seek to impose on platform business models and packet management. These prophesies of doom are so old, repetitive, and shallow that they lack credibility.

Although fears of a telecom and cable industry takeover of the Internet are vastly overblown, one of the network neutrality advocates' intuitions is correct: the Internet as we know it most certainly will cease to exist one day, just as ARPANET, CYCLADES, and the very first worldwide network for digital communications, the telegraph network,[110] have ceased to exist. The Internet will one day become too expensive to operate and will be replaced by a better system. The fact that this will occur has more to do with limits to growth designed-in to the Internet and the general tendency of technologies to improve. The Internet as we know it will become a part of a larger network before it disappears, however.

Humans have a natural tendency to fear change. Complex technologies like the Internet probably exaggerate this tendency because they require a great deal of study to master— and people are reluctant to lose the benefit of all that work simply because a better system comes along. Arguments for freezing the Internet into a simplistic regulatory straightjacket often have a distinctly emotional character that frequently borders on manipulation.

The Internet is a wonderful system. It represents a new standard of global cooperation and enables forms of interaction never before possible. Thanks to the Internet, societies around the world reap the benefits of access to information, opportunities for collaboration, and modes of communication that weren't conceivable to the public a few years ago. It's such a wonderful system that we have to strive very hard not to make it into a fetish object, imbued with magical powers and beyond the realm of dispassionate analysis, criticism, and improvement.

At the end of the day, the Internet is simply a machine. It was built the way it was largely by a series of accidents, and it could easily have evolved along completely different lines with no loss of value to the public. Instead of separating TCP from IP in the way that they did, the academics in Palo Alto who adapted the CYCLADES architecture to the ARPANET infrastructure could have taken a different tack: They could have left them combined as a single architectural unit providing different retransmission policies (a reliable TCP-like policy and an unreliable UDP-like policy) or they could have chosen a different protocol such as Watson's Delta-t or Pouzin's CYCLADES TS. Had the academics gone in either of these directions, we could still have a World Wide Web and all the social networks it enables, perhaps with greater resiliency.

The glue that holds the Internet together is not any particular protocol or software implementation: first and foremost, it's the agreements between operators of Autonomous Systems to meet and share packets at Internet Exchange Centers and their willingness to work together. These agreements are slowly evolving from a blanket pact to cross boundaries with no particular regard for QoS into a richer system that may someday preserve delivery requirements on a large scale. Such agreements are entirely consistent with the structure of the IP packet, the needs of new applications, user empowerment, and "tussle."

The Internet's fundamental vibrancy is the sandbox created by the designers of the first datagram networks that permitted network service enhancements to be built and tested without destabilizing the network or exposing it to unnecessary hazards. We don't fully utilize the potential of the network to rise to new challenges if we confine innovations to the sandbox instead of moving them to the parts of the network infrastructure where they can do the most good once they're proven. The real meaning of end-to-end lies in the dynamism it bestows on the Internet by supporting innovation not just in applications but in fundamental network services. The Internet was designed for continual improvement: There is no reason not to continue down that path.

ENDNOTES

1. Tim Wu, "Network Neutrality, Broadband Discrimination," *Journal of Telecommunications and High Technology Law* (2) (2003): 141 <ssrn.com/abstract=388863> (accessed August 18, 2009).

2. Brian E. Carpenter, "Architectural Principles of the Internet," RFC 1958, maintained by the Internet Engineering Task Force, June 1996 <www.ietf.org/rfc/rfc1958.txt> (accessed September 18, 2009).

3. Jerome H. Saltzer, David P. Reed, and David D. Clark, "End-to-End Arguments in System Design," presentation at the Second International Conference on Distributed Computing Systems, sponsored by the Institute of Electrical and Electronic Engineers, (Paris, France, April 8-10, 1981): 509-512 <web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf> (accessed August 18, 2009).

4. Lawrence Lessig, *Code and Other Laws of Cyberspace* (New York: Basic Books, 1999).

5. Many people regard MIT's Multics operating system as the spiritual ancestor of UNIX©.

6. In the current milieu of excessive and intrusive advertising on the Web, it's easy to see their point.

7. Tim Berners-Lee's Web browser, called WorldWideWeb, only ran on the NeXTSTEP operating system.

8. Free Press, Inc., "Future of the Internet," Web essay, n.d. <freepress.net/media_issues/internet> (accessed June 9, 2009).

9. It actually doesn't, but this is a common misconception. The IP header specification includes three bits to express the desired Type of Service (ToS). The confusion arises because Border Gateway Protocol (BGP) doesn't describe how to implement ToS across domain boundaries.

10. Lawrence Lessig, Timothy Wu, and Susan Crawford have all made similar assertions.

11. Michael J. Copps, *Bringing Broadband to Rural America: Report on a Rural Broadband Strategy* (Washington, D.C.: Federal Communications Commission, May 22, 2009): 62 <hraunfoss.fcc.gov/edocs_public/attachmatch/DOC-291012A1.pdf> (accessed August 18, 2009).

12. Computer Control Company was ultimately acquired by Honeywell, so this system is sometimes called the Honeywell DDP-516.

13. "Honeywell DDP-516," Old-Computers.com Website <www.old-computers.com/MUSEUM/computer.asp?st=1&c=551> (accessed August 18, 2009).

14. Lawrence G. Roberts, "Multiple Computer Networks and Intercomputer Communication," ACM Symposium on Operating Systems Principles, *Proceedings of the first ACM symposium on Operating System Principles* (New York: ACM, 1967): 3.1-3.6. <portal.acm.org/ft_gateway.cfm?id=811680&type=pdf&coll=portal&dl=ACM&CFID=3817252&CFTOKEN=99895708> (accessed September 8, 2009).

15. There's essentially a revolving door between ARPA and BBN.

16. Requests for Comments (RFCs) are produced by the Network Working Group, which is now a part of the Internet Engineering Task Force (IETF), and they contain the Internet's specifications. Frustration with the contract engineering firm BBN is evident in this passage from the first RFC published on April 7, 1969: 'The point is made by Jeff Rulifson at SRI that error checking at major software interfaces is always a good thing. He points to some experience at SRI where it has saved much dispute and wasted effort. On these grounds, we would like to see some HOST to HOST checking. Besides checking the software interface, it would also check the HOST-Interface Message Processor (IMP) transmission hardware. (BB&N claims the HOST-IMP hardware will be as reliable as the internal registers of the HOST. We believe them, but we still want the error checking.). Steve Crocker, "Host Software," RFC 1, maintained by the Internet Engineering Task Force, April 7, 2009 <tools.ietf.org/html/rfc1> (accessed September 17, 2009).

17. Louis Pouzin, "Presentation and Major Design Aspects of the CYCLADES Computer Network," *Proceedings of the Third Data Communications Symposium,* Tampa, Florida, November 1973 (New York: Association for Computing Machinery, 1973) <rogerdmoore.ca/PS/CYCLB.html> (accessed August 18, 2009); David Walden, "Observations on the Beginnings of the Internet," Conference presentation, Asilomar State Park, California, April 24, 2002 <www.walden-family.com/dave/archive/net-history/internet-begin.pdf> (accessed August 20, 2009).

18. DECNet chief architect Tony Lauck tells me that Marek Irland, a member of Pouzin's team at INRIA, worked as a consultant with the DECNet team for two years in the 1970s, where he was instrumental in developing the DECNet architecture. The Internet protocols team included Gérard Le Lann, another of Pouzin's team. XNS was developed in close concert with the Internet, and everyone contributed to OSI.

19. Ibid.

20. See: Roy Longbottom, "Computer Speeds From Instruction Mixes Pre-1960 to 1971," Roy Longbottom's PC Benchmark Collection Website <www.roylongbottom.org.uk/cpumix.htm> (accessed August 18, 2009) and "ARM Architecture," Wikipedia <en.wikipedia.org/wiki/ARM_architecture> (accessed August 18, 2009).

21. Louis Pouzin, "CIGALE, The Packet Switching Machine of the CYCLADES Computer Network," *Proceedings of the International Federation for Information Processing,* Stockholm, Sweden, August 1974: 155-159 <rogerdmoore.ca/PS/CIGALE/CIGALE.html> (accessed August 18, 2009).

22. Pouzin, op. cit.

23. The current Internet doesn't actually support this mode of operation as a core function, but application-level P2P protocols provide something very similar.

24. Robert Cailliau, *How the Web Was Born: The Story of the World Wide Web* (Oxford: Oxford University Press, 2000): 40-41.

25. Vinton Cerf, Yogen Dalal, and Carl Sunshine, "Specification of Internet Transmission Control Program," RFC675, maintained by the Internet Engineering Task Force, December 1974 <tools.ietf.org/html/rfc675> (accessed September 17, 2009).

26. Vinton G. Cerf and Robert E. Kahn, "A Protocol for Packet Network Intercommunication," *IEEE Transactions on Communications* 22(5) (May 1974) <www.cs.princeton.edu/courses/archive/fall06/cos561/papers/cerf74.pdf> (accessed August 18, 2009).

27. While ARPANET did ultimately confront the internetworking problem when it connected with the ALOHA Net in Hawaii, internetworking was not among its design goals.

28. Lawrence G. Roberts, "The evolution of packet switching," *Proceedings of the IEEE* 66(11)(November 1978)<ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1455410&isnumber=31271> (accessed September 6, 2009).

29. An extreme analysis of the engineer's propensity for radicalism is presented in Diego Gambetta and Steffen Hertog's paper entitled "Engineers of Jihad," University of Oxford Sociology Working Paper Number 2007-10, 2007 <www.nuff.ox.ac.uk/users/gambetta/Engineers%20of%20Jihad.pdf> (accessed August 25, 2009).

30. The author was the vice-chairman of that effort, which produced the IEEE 802.3 1BASE5 standard in 1986.

31. Many people regard MIT's Multics operating system as the spiritual ancestor of Unix©.

32. Saltzer, Reed, and Clark, op. cit.

33. Ibid.

34. A 1970's programmer joke captures this dichotomy: Programmer 1: "My program takes half the memory and a tenth the time of your program." Programmer 2: "Yes, but mine gets the right answer." TCP gets the right answer.

35. Pouzin, op. cit.

36. The Low Extra Delay Background Transport (LEDBAT) protocol currently under development in the IETF is an attempt to reduce network congestion caused by P2P transfers by sensing congested paths and avoiding them. See: tools.ietf.org/wg/ledbat/.

37. Saltzer, Reed, and Clark, op. cit.

38. Crocker, op. cit.

39. Such failures were not uncommon in interface circuits that failed to include error-detection logic. Best practice was to check data with an additional "parity" line.

40. John Nagle, "Congestion Control in IP/TCP Networks," RFC 896, maintained by the Internet Engineering Task Force, January 6, 1984 <tools.ietf.org/html/rfc896> (accessed September 17, 2009).

41. John Nagle, comment on the Slashdot blog, March 24, 2008 <tech.slashdot.org/comments.pl?sid=497516&cid=22847764> (accessed June 1, 2009).

42. Van Jacobson, "Congestion Avoidance and Control," *Proceedings of SIGCOMM '88* (Stanford, California: Association for Computing Machinery: August 1988) <ee.lbl.gov/papers/congavoIbidpdf> (accessed August 18, 2009).

43. Raj Jain, K. K. Ramakrishnan, and Dah-Ming Chiu, *Congestion Avoidance in Computer Networks With a Connectionless Network Layer,* TR-506 (Littleton, Massachusetts: Digital Equipment Corporation, 1988) <arxiv.org/ftp/cs/papers/9809/9809094.pdf> (accessed August 18, 2009).

44. Jacobson, op. cit., 1988.

45. K. K. Ramakrishnan, Sally Floyd, and David L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, maintained by the Internet Engineering Task Force, January 6, 1984 <tools.ietf.org/html/rfc3168> (accessed September 17, 2009).

46. Microsoft's effective control of much of the Internet infrastructure is one side effect of end-to-end architecture.

47. Bob Braden et al., "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, maintained by the Internet Engineering Task Force, April 1998 <tools.ietf.org/html/rfc2309> (accessed September 18, 2009).

48. "Global Synchronization" is a phenomenon is which a large number of TCP flows cycle back and forth between half-speed and overload in lockstep with each other. See: Jamal Hadi Salim and Uvaiz Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks," RFC 2884, maintained by the Internet Engineering Task Force, July 2000 <tools.ietf.org/html/rfc2884> (accessed September 17, 2009).

49. "…several experiments and analyses have shown that [standard TCP] is not suitable for each and every kind of application and environment—e.g., bulk data transfer in high bandwidth, large round trip time networks" Mathieu Goutelle et al., "A Survey of Transport Protocols Other than Standard TCP," Global Grid Forum, May 10, 2005 <www.ogf.org/Public_Comment_Docs/Documents/May-2005/draft-ggf-dtrg-survey-1.pdf> (accessed August 18, 2009).

50. Bob Briscoe, "A Fairer, Faster Internet Protocol," *IEEE Spectrum* (December 2008): 38-43 <www.spectrum.ieee.org/dec08/7027> (accessed September 8, 2009).

51. David D. Clark, "The Design Philosophy of the DARPA Internet Protocols," *Proc SIGCOMM 88* (18)(4) (August 1988): 106-114 <nms.lcs.mit.edu/6829-papers/darpa-internet.pdf> (accessed August 18, 2009); Marjory S. Blumenthal and David D. Clark, "Rethinking the Design of the Internet: The End-to-End Arguments vs. The Brave New World," *ACM Transactions on Internet Technology* 1(1) (August 2001): 70-109 <www.caip.rutgers.edu/~virajb/readinglist/rethinking.pdf> (accessed August 18, 2009).

52. Carpenter, op. cit.

53. Ibid.

54. John Day, personal communication.

55. Tim Moors, "A Critical Review of End-to-End Arguments in System Design," *Proceedings of the 2000 IEEE International Conference on Communications* (April 2002): 1214-1219 <www.seas.upenn.edu/~farooq/tcom512/a-critical-review-of_e2e.pdf> (accessed August 18, 2009).

56. Randy Bush and David Meyer, "Some Internet Architectural Guidelines and Philosophy," RFC 3429, maintained by the Internet Engineering Task Force, December 2002 <tools.ietf.org/html/rfc3439> (accessed September 17, 2009).

57. Ibid.

58. Ibid.

59. James Kempf and Rob Austein, "The Rise of the Middle and the Future of End-to-End: Reflections on the Evolution of the Internet Architecture," RFC 3724, maintained by the Internet Engineering Task Force, March 2004 <tools.ietf.org/html/rfc3724> (accessed September 17, 2009).

60. Ibid., 5.

61. Sally Floyd and Leslie Daigle, eds., "IAB Architectural and Policy Considerations for Open Pluggable Edge Services," RFC 3238, maintained by the Internet Engineering Task Force, January 2002 <tools.ietf.org/html/rfc3238> (accessed September 17, 2009).

62. David D. Clark et al., "Tussle in Cyberspace: Defining Tomorrow's Internet," *IEEE/ACM Transactions on Networking* (13) (3) (June 2005): 471 <groups.csail.mit.edu/ana/Publications/PubPDFs/Tussle%20in%20Cyberspace%20Defining%20Tomorrows%20Internet%202005%27s%20Internet.pdf> (accessed August 18, 2009).

63. Ibid., 473.

64. Ibid.

65. Speaking before the U.S. Federal Communication Commission's workshop on wireless technology conducted in connection with the National Broadband Plan on August 13, 2009, net neutrality advocate Sascha Meinrath touted the merits of "open mesh" wireless networks operated by technically skilled hobbyists at very small scale relative to the Internet. Networks of dozens of nodes face an entirely different set of problems than those Internet operators address daily. See video at: www.fcc.gov/realaudio/mt081309b.ram.

66. Mark Handley, "Network Neutrality and the IETF," presentation to IETF 75, Stockholm, July 30, 2009: 3-4 <www.ietf.org/proceedings/75/slides/plenaryt-4.pdf> (accessed August 18, 2009)

67. See: Jonathan Rosenoer, *CyberLaw: The Law of the Internet* (North Communications, SSRN) <ssrn.com/abstract=11460> (accessed August 18, 2009).

68. Barlow is best known for his work as a songwriter for The Grateful Dead, the rock band immortalized in Tom Wolfe's New Journalism tale *The Electric Kool-Aid Acid Test* (New York: Farrar, Straus and Giroux, 1968).

69. John Perry Barlow, *A Declaration of the Independence of Cyberspace*, Web page <homes.eff.org/~barlow/Declaration-Final.html> (accessed August 18, 2009).

70. Joe Accardi*,* "Library Journal," *Northeastern Illinois Univ. Lib* (Chicago: Reed Business Information, Inc., 1995).

71. David S. Isenberg, "Rise of the Stupid Network," *Computer Telephony* (August 1997): 16-26 <www.isen.com/stupIbidhtml> (accessed May 26, 2009).

72. David S. Isenberg, "The Dawn of the Stupid Network," *ACM Networker 2.1* (February/March 1998): 24-31 <www.isen.com/papers/DawnstupIbidhtml> (accessed May 26, 2009).

73. Geoff Huston, "Faster," The ISP Column, Internet Society Website, June 2005 <ispcolumn.isoc.org/2005-06/faster.html> (accessed August 19, 2009).

74. Lawrence Lessig, op. cit.

75. Ibid., 32-33.

76. Angela Moscaritolo, "Iran election protesters use Twitter to recruit hackers," *SC Magazine* (June 15, 2009) <www.scmagazineus.com/Iranian-election-protestors-use-Twitter-to-recruit-hackers/article/138545/> (accessed August 18, 2009).

77. Stanford Center for Internet and Society, "The Policy Implications of End-to-End," workshop sponsored by the Stanford Program in Law, Science, and Technology, Stanford, CA, December 1, 2000 <cyberlaw.stanford.edu/e2e/> (accessed August 18, 2009).

78. Christine Hemerick, *Panel 4*, presentation to the Policy Implications of End to End conference, Stanford Law School Center for Internet and Society, Stanford, CA, December, 2000 <cyberlaw.stanford.edu/e2e/papers/e2e.panel4.pdf> (accessed August 18, 2009).

79. Lawrence Lessig, *The Future of Idea*s (New York: Random House, 2001); *Free Culture* (New York: Penguin Books, 2004); *Cod* (New York: Basic Books, 2006), etc.

80. Ian Wakeman, et al., "Layering considered harmful", *IEEE Network* (New York, January 1991).

81. Wu, op. cit., 141.

82. Except for "Affirmative Action," which liberals regard as good; in the United Kingdom, it's called "Positive Discrimination."

83. Wu, op. cit., at 197.

84. Richard S. Whitt, "A Horizontal Leap Forward: Formulating a New Communications Public Policy Framework Based on the Network Layers Model" *Federal Communications Law Journal* (56): 587-672, May 20, 2004 <www.law.indiana.edu/fclj/pubs/v56/no3/Whitt%20Final%202.pdf> (accessed August 18, 2009).

85. Lawrence B. Solum and Minn Chung, "The Layers Principle: Internet Architecture and the Law," University of San Diego School of Law, Public Law and Legal Theory Research Paper No. 55, June 16, 2003: 29 <ssrn.com/abstract=416263> (accessed August 18, 2009).

86. Yochai Benkler, "From Consumers to Users: Shifting the Deeper Structures of Regulation Toward Sustainable Commons and User Access," *Federal Communications Law Journal* (52) (2000): 561, 562.

87. Kevin Werbach, "A Layered Model for Internet Policy," *Journal on Telecommunications and High Technology Law* (1) (2002): 37, 39-40.

88. Rob Frieden, "Adjusting the Horizontal and Vertical in Telecommunications Regulation: A Comparison of the Traditional and a New Layered Approach," *Federal Communications Law Journal* (55) (2003): 207, 209.

89. Bush and Meyer, Ibid.

90. Wes Eddy, *Re: [tsv-area] TSVAREA presentation slots for Stockholm*, e-mail to the IETF tsv-area mailing list (June 4, 2009) <www.ietf.org/mail-archive/web/tsv-area/current/msg00460.html> (accessed August 18, 2009).

91. Day wrote *Patterns in Network Architecture: A Return to Fundamentals,* Prentice Hall (2008) an extremely significant work on the history and dynamics of network architectures, and organized the RINA initiative mentioned later.

92. *Xerox System Integration Standard - Internet Transport Protocols* (Xerox Corporation, Stamford, CT, 1981).

93. David M. Piscitello and A. Lyman Chapin, *Open Systems Networking* (Reading, PA: Addison-Wesley, 1993).

94. Jonathan Zittrain of the Berkman Center and others imagine that America Online and Compuserve were in the running; this is hardly possible as neither AOL nor Compuserve maintained an internetwork or had designed an internetworking architecture. These systems were more like today's social networks than alternative Internets.

95. By comparison with XNS, TCP was regarded as primitive with respect to naming and addressing, management, and directory services. The fragmentation that's about to beset the IPv4 address space would not have happened to XNS.

96. For a full post mortem on the political failure of OSI, see John Day, op. cit.

97. Ben Segal, "A Short History of Internet Protocols at CERN," CERN Website, April 1995 <ben.web.cern.ch/ben/TCPHIST.html> (accessed May 26, 2009).

98. Walt Howe, "A Brief History of the Internet," Walt Howe's Internet Learning Center Website <www.walthowe.com/navnet/history.html> (accessed May 26, 2009).

99. Except for the fact that NCP had been phased out before Berners-Lee had his revelation

100. The ISO IS-IS routing protocol was designed by Digital Eqipment Corporation engineers Radia Perlman and David Oran.

101. Rohit Khare and Ian Jacobs, "W3C Recommendations Reduce 'World Wide Wait,'" W3C Website, 1999 <www.w3.org/Protocols/NL-PerfNote.html> (accessed August 18, 2009).

102. This work is underway in the IETF.

103. *Cisco Visual Networking Index: Forecast and Methodology, 2008–2013* (San Jose, CA: Cisco Systems Inc., 2009) <www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf> (accessed August 18, 2009).

104. David Meyer, Lixia Zhang, and Kevin Fall, eds., "Report from the IAB Workshop on Routing and Addressing," RFC 4984 maintained by the Internet Engineering Task Force, September 2007 <www.ietf.org/rfc/rfc4984.txt>.

105. For detailed information about RINA, see the website of the Pouzin Society, 02d7097.netsolhost.com/documentspublications.html and Day, op. cit.

106. Richard W. Watson, "Delta-t Protocol Specifications," *Report UCRL-52881* (Washington, D.C.: Lawrence Livermore National Laboratory, November 1979); Richard W. Watson, "Delta-t Protocols Specification," (Washington, D.C.: Lawrence Livermore National Laboratory, April 15, 1983).

107. Joseph D. Touch, Yu-Shun Wang, and Venkata Pingali, *A Recursive Network Architecture* (ISI Technical Report ISI-TR-2006-626, October 2006) <www.isi.edu/touch/pubs/isi-tr-2006-626/> (accessed August 18, 2009).

108. Kathleen Q. Abernathy, "Remarks at Disruptive Technologies and Opportunities for Service Providers Panel," *Telecoms Transition World Summit* (Lisbon, Portugal, June 27, 2005) <fjallfoss.fcc.gov/edocs_public/attachmatch/DOC-260480A1.pdf> (accessed August 18, 2009).

109. Hubert Zimmermann, "OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications* 28(4) (April 1980):425- 432 <www.comsoc.org/livepubs/50_journals/pdf/RightsManagement_eid=136833.pdf> (accessed August 18, 2009).

110. Tom Standage, *The Victorian Internet* (New York: Walker and Co, 1998).

ABOUT THE AUTHOR

Richard Bennett is a Research Fellow with the Information Technology and Innovation Foundation (ITIF) in Washington, D.C. He was vice-chairman of the IEEE 802.3 1BASE5 task group that wrote the first standard for Ethernet over twisted pair, and a contributor to the IEEE 802.11 standard's original system architecture and designer of the IEEE 802.11n packet aggregation scheme. He was active in Open Systems Interconnection (OSI) standards work, the instigator of RFC 1001 and 1002, and the inventor of the Distributed Reservation Protocol for the Wi-Media Ultra-Wideband network. A frequent conference speaker, he has testified before the U.S. Federal Communications Commission as an expert witness in their network management inquiry and before Congress on Internet privacy.

ABOUT THE INFORMATION TECHNOLOGY AND INNOVATION FOUNDATION

ACKNOWLEDGMENTS

The author thanks Louis Pouzin, John Day, Rob Atkinson, Dave Walden, Chris Yoo, Matthias Bärwolff, and Tony Lauck for many helpful comments on early drafts of this document. All remaining errors are the responsibility of the author.

For more information contact ITIF at 202-449-1351 or at mail@itif.org, or go online to www.itif.org.
ITIF | 1101 K St. N.W. | Suite 610 | Washington, DC 20005