Active Networking and End-To-End Arguments

Comment by David P. Reed, Jerome H. Saltzer, and David D. Clark[1]
dpreed@reed.com, Saltzer@mit.edu, ddc@lcs.mit.edu

Some twenty years have elapsed since we identified and named end-to-end arguments[1], a class of system design principles that organize and guide the placement of function within a system. These arguments and the underlying principles have now been invoked in many contexts, becoming part of the vocabulary of network protocol and operating system designers. Like other general design principles, end-to-end arguments impose a structure on the design space, rather than solving the design problem. This structure provides a basis for discussion and analysis of trade-offs, and suggests a strong rationale to justify design choices.

This note comments on a current design controversy that can be framed partly in terms of end-to-end arguments. One form of active networking[2], a novel category of communication network architectures, comprises attaching programs to data packets, with the intent that those programs be executed at points within the network. The purpose is to better match the behavior of the network to the requirements of the application. The question being raised is whether or not this idea directly violates an end-to-end argument.

On the one hand, programmability appears to contradict the end-to-end principle that a function or service should be carried out within a network layer only if it is needed by all clients of that layer, and it can be completely implemented in that layer[2]. On the other hand, programmability may allow a network client to implement precisely the service it needs, an outcome that is consonant with end-to-end arguments. Contradiction and consonance aside, because programmability enables such a wide range of possibilities, applying end-to-end arguments in a general, yet definitive, way may be impossible. Instead, the specifics of each particular active networking idea would benefit from evaluation in light of the end-to-end principle.

Programmability's Effect on Design-time Function Placement

End-to-end arguments address design more than implementation and implementation more than execution—that is, they suggest who should provide the code, not which box it should run on. Moving functions and services upward in a layered design, closer to the application(s) that use them, increases the flexibility and autonomy of the application designer to apply those functions and services to the specific needs of the application. With that view, programmability in a lower layer can be seen as a means to defer design choices upwards in the layering, closer to the application, and later in time, even though the resulting functions may actually take place deep inside the network.

At the same time, one can raise a concern that a general programming interface can lead to complex and unpredictable interactions among independently designed applications and independently acting users. Part of the context of an end-to-end argument is the idea that a lower layer of a system should support the widest

---

1. Author's affiliations: David P. Reed, Techburst, 16 Schaffner Lane, Dover, MA 02030; Jerome H. Saltzer and David D. Clark, M. I. T. Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139.
2. There are some situations where applying an end-to-end argument is counter–productive. One category is cost–related. For example, many-to-many communications can be done by forwarding packets among multicast servers located outside the network, but it appears to be much more effective to add some support at the lowest layers of the network. Another category is where for some reason the end–points are not in a position to cooperate. For example, version 5 of the Kerberos authentication system pulls inside the application programming interface the function of replay prevention, because experience with version 4 showed that few application programmers understood how to do that critical function correctly. The interesting thing is how few examples have turned up in 20 years of experience with systems like the Internet.

possible variety of services and functions, so as to permit applications that cannot be anticipated. That is, minimize the lower–layer function, get out of the way, and let the higher layer do its thing. But this flexibility actually implies that end-to-end arguments have not one, but two complementary goals:

• Higher-level layers, more specific to an application, are free to (and thus expected to) organize lower-level network resources to achieve application-specific design goals efficiently. (application autonomy)

• Lower-level layers, which support many independent applications, should provide only resources of broad utility across applications, while providing to applications usable means for effective sharing of resources and resolution of resource conflicts. (network transparency)

While making lower layers more active or programmable is likely to enhance application autonomy, the risk is that programmable lower layers may reduce network transparency. The reason is that a key element of transparency is some ability to predict how the network will behave.

Since lower-level network resources are shared among many different users with different applications, the complexity of potential interactions among independent users rises with the complexity of the behaviors that the users or applications can request. For example, when the lower layer offers a simple store-and-forward packet transport service, interactions take the form of end-to-end delay that can be modeled by relatively straightforward queueing models. Adding priority mechanisms (to limit the impact of congestion) that are fixed at design time adds modest complexity to models that predict the behavior of the system. But relatively simple programming capabilities, such as allowing packets to change their priority dynamically within the network, may create behaviors that are intractable to model, in the same way that the simple rules of cellular automata such as Conway's Game of Life[3] can lead to remarkably complex behavior.

To maintain the largest degree of network transparency, then, the end-to-end principle requires that the semantics of any active features be carefully constrained so that interactions among different users of a shared lower level can be predicted by a designer who is using the services and functions of that active layer. Lack of predictability thus becomes a cost for all users, including those that do not use the programmability features. Getting the semantics of active enhancements right is a major challenge, and wrong active enhancements are likely to be worse than none at all, since everyone helps pay the cost of something that is used by only a few but reduces transparency for everyone else.

Thus, even though active network ideas are not ruled out by end-to-end arguments, we have not seen practical, high-impact examples of a sufficiently simple, flexible, and transparent programming semantics suitable for use in lower levels of networks. Until such examples are developed in detail, the existence of active networks that meet the end-to-end criteria should perhaps be classified as theoretical.

Keep it Simple, Stupid

End-to-end arguments arose from our work on secure operating system kernels in the Multics project[4,5], and our work on end-to-end transport protocols in LAN's and the Internet experiment[6]. Similar thinking by John Cocke and his colleagues on the role of compilers in simplifying processor architecture led to the RISC approach[7] to processor architecture, which also suggests moving function from lower layers to more application-specific layers. In the past 20 years, systems designers have only begun to explore and demonstrate the profound implications of this kind of architectural approach on design in large scale systems.

An end-to-end argument is similar to the argument for RISC: it serves to remind us that building complex function into a network implicitly optimizes the network for one set of uses while substantially increasing the cost of a set of potentially valuable uses that may be unknown or unpredictable at design time. A case in point: had the original Internet design been optimized for telephony-style virtual circuits (as were its contemporaries SNA and TYMNET), it would never have enabled the experimentation that led to protocols that could support the World-Wide Web, or the flexible interconnect that has led to the flowering of a million independent Internet service providers (ISP's). Preserving low-cost options to innovate outside the network, while keeping the core network services and functions simple and cheap, has been shown to have very substantial value. In this way, an end-to-end argument does not oppose active networks, per se, but instead strongly suggests that enthusiasm for the benefits of optimizing current application needs by making the network more complex may be misplaced.

This idea of "stupid operating systems", "stupid networks", and "stupid processors" hasn't yet had its full run[3]. The telephone company still seems to think that all users want the illusion of a copper pair from the user's house to some ISP point of presence in another city. In its Internet access approach, the cable company places the real network one step closer, but neither architecture is really prepared for the household with three computers and two network access providers. Politicians want both the cable company and the ISP to filter packets for things children shouldn't see, and the FBI asks them to make copies of specific data streams to simplify wiretapping (these may be examples of non–cooperating end–points). Political arguments aside, even if one accepts these requirements, the corresponding implementation proposals are sometimes stunning in the way they fail to consider end-to-end arguments.

Take it case-by-case

It is important to keep in mind that end-to-end arguments are one of several important organizing principles for systems design. While there will be situations where other principles or goals have greater weight, an end-to-end argument can facilitate the design conversation that leads to a more flexible and scalable architecture. We suggest that, as researchers investigate various ideas that are lumped into the research area called active networking, it is important to consider and understand the specific applications of end-to-end arguments to the designs under consideration. Those active interfaces that survive close inspection under this light should benefit from such analysis.

References

[1] J[erome] H. Saltzer, D[avid]. P. Reed, and D[avid]. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems 2*, 4 (November 1984) pages 277-288. An earlier version appeared in the *Second International Conference on Distributed Computing Systems* (April, 1981) pages 509-512.

[2] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *IEEE Communications Magazine 35*,1 (January 1997) pages 80-86.

[3] Elwyn R. Berlekamp, John H. Conway, Richard K. Guy. *Winning ways for your mathematical plays. Volume 2: Games in particular.* Academic Press, 1982. Pages 817-849.

[4] Michael D. Schroeder, David D. Clark, and Jerome H. Saltzer. The Multics kernel design project. *Sixth ACM Symposium on Operating Systems Principles*, in *ACM Operating Systems Review 11*, 5 (November 1977) pages 43-56.

[5] David P. Reed. Processor multiplexing in a layered operating system. S.M. and E.E. thesis, M. I. T. Department of EECS, 1976. Available as M. I. T. Project MAC Technical Report MAC-TR-164, July 1976.)

[6] D.D. Clark, K.T. Pogran, and D.P. Reed. An introduction to local area networks. *Proceedings of the IEEE 66*, 11 (November 1978) pages 1497-1516.

[7] George Radin. The 801 Minicomputer. *Proceedings of the first ACM Symposium for Programming Languages and Operating Systems* (1982), in *Computer Architecture News 10*, 2 (March, 1982) pages 39-47.

3. David Isenberg of AT&T has recently re-invented end-to-end arguments in an entertaining paper called "The Rise of the Stupid Network" (http://www.computertelephony.com/ct/att.html) that criticizes the telephone industry's concept of "Intelligent Network" based on ideas very similar to those in the original end-to-end arguments paper.