End-to-end arguments: The Internet and beyond

David P. Reed
dpreed@reed.com

USENIX Security '10
13 August 2010

Agenda

Historical and personal perspective
Definition
Principled design and modularity
Sorting out some confusions
Controversies and challenges
Security in particular
The future of end-to-end arguments

History

1973 Saltzer – collected design principles for secure systems *kernel*

1973-1976 – Principles for crypto (Branstad...)

1976 – layering proc/mem abstraction in OS

1976-1978 principles for database recovery

1976-1977 TCP/Telnet design “factored” -> IP, TCP, UDP, ICMP, Telnet

1978 – IEEE Proc. Special Issue on Networking

1978 – coordination in autonomous decentralized systems

The Internet Created: Design Context

Clark, *The Design Philosophy of the Internet Protocols*. "The top level goal of the DARPA Internet Architecture was to develop an effective technique for the multiplexed utilization of existing interconnected networks. ... the top level assumption was that the top layer of interconnection would be provided by a layer of Internet packet switches, which were called gateways"

Clark, Reed, Pogran, *An Introduction to Local Area Networks.* "The utilization of a technological innovation often occurs in two stages. ... first stage, the innovation is exploited to perform better the same tasks that were already being performed.... second stage, new applications are discovered, which could not reasonably be performed or even foreseen prior to the innovation....

The greatest impact ... will come with...systems that integrate the idea of distribution and communication at a fundamental level.....

The impact...on the decentralization of computing is sociological as well as technological. ... decentralized computing greatly increases autonomy...

The paper

Saltzer identied non-intuitive structure of *some* systems design principles, named the "end to end argument" - I and Clark had been MIT's key participants in the DARPA *protocol* architecture team

What NOT to design into the "core" of a system?

Insight: that *resisting* function inclusion was often the correct design choice.

Definition in paper

In a system S including a shared communications subsystem C,

App. function F might be specified to be implemented either in C, or in the endpoints using S, or both.

F can only be completely and correctly implemented at the endpoints. Therefore providing F in C is not possible. (an incomplete F' in C may be useful for optimization).

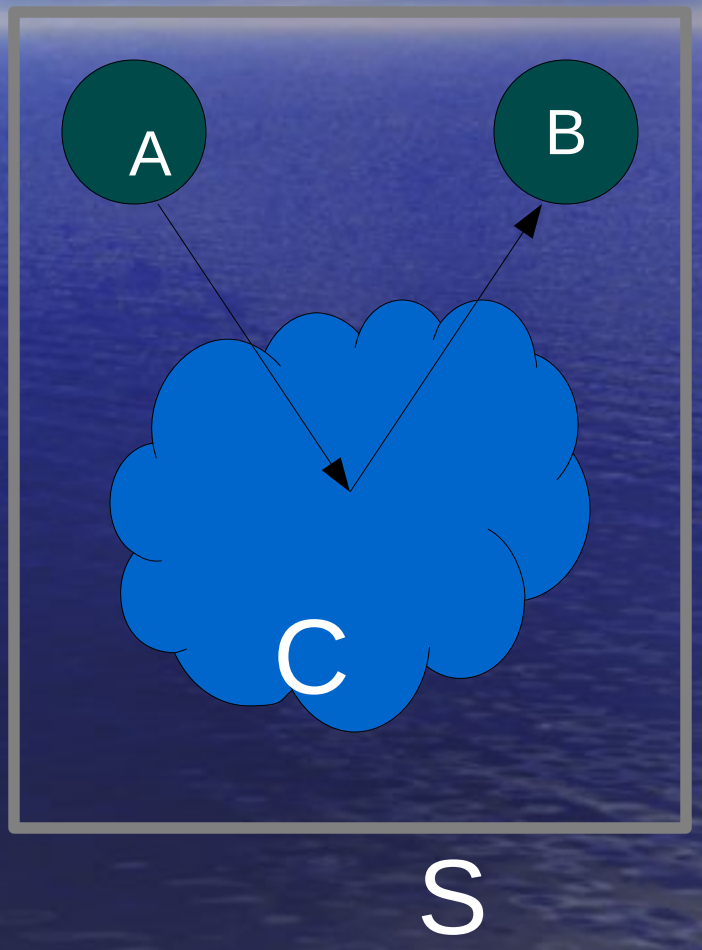
What we meant

F: Secure Message Delivery: only B can see contents

Using Internet – complete and correct SMD can only be ensured by end-to-end encryption

Therefore providing F in C is not possible. (an incomplete F' in C may be useful for optimization).

An End-to-end argument



What is this thing called End-to-end argumentation?

Class of arguments *against low level function implementation*

What is it an argument *for*?

When does it apply?

What's an “endpoint”?

Examples of the argument

Reliable delivery

Duplicate suppression

In-order delivery

Authentication/Accountability

Reputation maintenence

Fault tolerance

Non-examples

Traffic management

Capacity reservation

Multicast routing

Packet fragmentation and reassembly

What are the ends?

Doesn't cloud computing provide a counter example?

- Amazon EC2/S3 is not “in the communications system” - it's an end

- *functions* are not services or applications, and “in the net” is not geographical or topological.

What is necessarily in/of “the net”?

Problem for the model: *jurisdiction*.

Does it make sense to provide a function where a user says - “my traffic should/should not traverse jurisdiction X”? (Finland vs. Sweden)

How do we deal with UAE requirement to make Blackberry Messaging tappable?

In hindsight, term *function* led to confusion

Function == quality, property, attribute

Fallacy of Composition: quality of whole != quality of part

Expansive property (liquid) vs.

Emergent property (inexpensive)

E2EA as a design rubric

An exhortation to be careful in defining functions properly

- Be honest about what function *F* is.
- Avoid confusing technique w/function
  - (should we provide *T* “in the net”? = What function does *T* provide.)

Thinking evolves

- 1980-81 paper circulated & presented
- 1984 ACM TOCS
- 1988 Republished, called “core Internet principle”
- 1997 “Active Networking and the End-to-end ..”
- 1997 Isenberg “Rise of the Stupid Network”
- 2000 Lessig TNR: “...intelligence should lie in the applications, or "ends." This design principle has a consequence: It embeds a type of neutrality. Because the network is "simple," it is not in a position to discriminate against new applications or new content.”
- 2000 Lessig Conference on Policy implications of E2EA

Late binding

Saltzer at the Lessig policy conference:

It's a line of reasoning that simply says that: in the lower layers in your system, you are going to be supporting [uses] that you cannot predict at a higher level. And therefore, you should be very conservative about embedding function in the lower layers because if you embed a function down there, everybody up there above you has to live with it and has to work with it.

And if you get it just a little bit wrong, then it's not going to work well for at least some applications up above. And therefore, you should — in the interest of being conservative, in the interest of allowing future iteration, you should push function up. You should keep the bottom layers as general and straightforward as possible.

Preserving options

Long-lived designs must handle *uncertainty* with regards to requirements and challenges.

In financial theory, *Real Options* give a precise way of thinking about the economic value of options in a design.

(Clark & Baldwin: Design Rules, HBS Press)

In the Internet design we used the *hourglass model and end-to-end arguments* to instantiate options

Two kinds of options

E2EA: Options recognizing uncertainty about future applications function needs

“Neck” generality: Options recognizing uncertainty about future technical capabilities

The Big Bet: Good surprises can be incorporated, bad ones mitigated

Options vs. Optimization

Preserve options or optimize:

A design tradeoff

One point of view: Barbara van Schewick – Internet Architecture and Innovation

Well explored territory

The Controversies Ensue

There are functions that MUST be in the network!

The end-to-end rules are too austere, too pure, too idealistic!

The end-to-end rules block innovation!

The end-to-end rules are obsolete!

End-to-end is a cult!

Why challenge E2EA?

"Feeping creaturism" - sellers need claims to tempt buyers to buy new stuff, and flexibility/performance not enough

Poorly understood problems invite hyperbolic claims (DNSSEC makes Internet *completely safe*; "Firewalls" make systems "secure", SPI moreso)

Concerns I take seriously

- Security, Robustness, Safety require abandoning or modifying end-to-end arguments (Zittrain, others)
- Policy requirements are not compatible with the end-to-end argument (Clark, Blumenthal)
- The Internet Design Principles are/at near some limit (clean slate, GENI/FIND)

Zittrain: The Future of the Internet and How to Stop It

The Generative Dilemma: "Zittrain argues that the generative architecture of the Internet is necessarily insecure. The Internet puts the users in control of their computers and therefore in control of the network as a whole. ... free to cause harm....amplified by generativity itself"

Perfect Enforcement: *"technologies that lock down user systems so that the users cannot control anything about them, save trivial aspects of their functioning"*

Cloud/SaaS: *freedom to lose independence*

Privacy 2.0: *"the challenges to personal and social privacy arising from pervasive real-time sensing, data capture, permanent logging, inference, and dissemination."*

Security, Robustness, Safety

What about Zittrain's concerns:

Control at the edges means insecurity

Cloud/SaaS leads to dependency risk

Locked down edges incents function “in network”

Loss of privacy is inherent in E2EA

Policy requirements mean abandonment of E2EA

Example: CALEA-like rules, spam blocking

“Internet design embodies [pick your radical] values” (ACLU, Libertarian, American, anti-capitalist) and must change

Is E2EA a policy or political principle, disguised as technical argumentation?

Some thoughts

Authors of E2EA and systems architects were politically diverse

E2EA are principles relating to function placement, not function choice

It's worth struggling with whether function *can be completely and correctly defined without understanding the system S or function F.*

What function does CALEA, spam blocking implement?

Criminal activity detection, evidence gathering (forensics) vs. listening to phone calls; crime reduction/mitigation vs. data gathering; imposing unwanted marketing on users vs. blocking ports.

Does E2EA inhibit policies or mechanisms?

Sometimes – it may shine new light...

Functions involving three parties and beyond

Non-Discretionary Control Functions

- Agreed authority?
- Disputed authority?
- Competing authorities?

Multiple autonomous subdomains

We need extensions to the “logic” in which we can express these functions (properties)!

The clean slate argument

If we could start again, would we design a different architecture?

What principles would we use?

Are their reasons to put many functions into *C*, the communications subsystem?

Some possible reasons

We now understand how to define functions needed for most or all applications and it can be solved for all cases, entirely in *C*.

We know an *extensive* property of the elements of C that comes “for free”, that may meet future needs.

We are willing to “settle” for good approximations to *F*, despite flaws.

Closing thoughts

- Designs and design principles survive because of clear, systematic reasoning
- E2EA neither gospel nor prime directive, but a pattern to reason by
- E2E Argumentation has strong foundation
- E2E Approach helps manage uncertainties
- The primary issue with E2EA and its feature:
  - We are confused when defining function
  - We confuse techniques, mechanisms, features, and capabilities with functions