CS 525
Advanced Distributed Systems
Spring 2010

Indranil Gupta (Indy)

The Middle or the End?

April 22, 2010

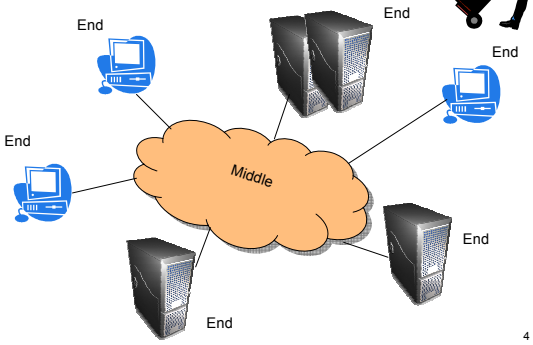
End to End Argument

- Made by J. H. Saltzer, D. P. Reed, and D. D. Clark.
- ACM Transactions on Computer Systems (TOCS) paper in 1984
- Was formulated in the early days of the Arpanet
- Has influenced almost all networked and distributed systems developed during, and since, the early Internet days
- Is a principle used by many researchers and developers in industry, and academia
- Often becomes a religious question
- Has come under challenge in the last few years

Through pristine eyes

- For a few minutes, forget everything you know about existing networks
- Imagine you are in a world where the first Internet (Arpanet) is being assembled
- It is being built for just a small set of users (ARPA is the US army's research organization)
- TCP has not been invented
- No one knows how popular this "Internet" will be, what applications will be run on top of it
- But everyone can think of a large number of applications that such an Internet "ought to support"
- Folks are at a loss, and often confused about what "principles" they should use to build the Internet

Simplistic View: where do I put..



The End to End Argument

- "Functions placed at low levels of a system may be redundant or of little *value* when compared with the *cost* of providing them at that low level."
- Examples: error recovery, security, duplicate suppression, message acknowledgement, etc.:
  - "Low level mechanisms to support these functions are justified only as *performance enhancements*." (as opposed to functionality)

Reasoning behind it

- "The function .. can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system."
- Putting something in the middle may not eliminate something bad from happening; it may only lower probability of something bad happening
- In deciding which functionality to put in the middle, need to consider issues of
  1. $ cost + engineering effort
     - The lower it is, the simpler is the network to build (and to use)
     - (Funding was limited!)
  2. Is the probability of the threat "low enough that the system allows useful work to be accomplished"?
     - If yes, put it at the ends
     - If no, put it in the middle (in the network)

"Probability of Threat…

- "If failures are fairly rare, then [the end to end technique] will normally work on the first try; occasionally a second or even third try might be required."
- E.g., Careful File Transfer example: correcting lost packets
  - Is the failure rate in the network so high (e.g., 50%) that the application has to be correcting errors most of the time?
  - Or is it so low (e.g., 10%-1%) that the end can correct it without impeding useful work?
  - Does the failure rate affect convenience at the ends, i.e., the resources available to do useful work (for the user and her processes)?

Other Reasons to Prefer Ends

- For a complex service that an end desires, one option is to have multiple vendors provide a chain of services which together provide the desired service
  - E.g., packet correction
  - OK if all vendors implement same functionality: but this rarely happens in real-life!
    - As a result, putting something in the middle may actually increase the probability of something going wrong!
    - One broken link in a chain may kill QoS!
  - E.g., encryption chains: PKI has succeeded, but it has only a few vendors that are really trusted

What is the Right Tradeoff?

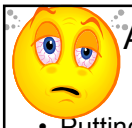
- "Using performance to justify placing functions in a low-level subsystem must be done carefully."
- Correctness not an issue, since you can only lower (but not reduce to zero) the probability of failures by doing something inside the network
- Rather, performance is important, insofar as it allows you to do useful work
- Has to be traded off with $ cost + engineering effort.
- So the question becomes: can you add the functionality to the lower layers with low enough cost+engineering effort?
  - (What about future implications of a new functionality, i.e., how might it affect future applications? How would anyone know?)

Additional Concerns

- Lower Level is common to many applications
  - Applications that don't need additional functionality that is in the middle will need to pay for it!
  - E.g., if reliable messaging were inside the Internet, would you be able to design a real-time multimedia transport protocol?
- Lower levels (middle) may not have as much information as the higher levels (ends).
- So the tradeoff here is: need across multiple applications vs. complexity at the end (if the e2e argument is followed).
- A kind of Occam's Razor in design for the Internet

Identifying the Ends

- In a file transfer protocol, the ends are the processes receiving and sending the file.
- In a protocol that involves a buffer (e.g., email), the storage buffer may also be a part of the end of the receiver.
- In a system model (e.g., BFT) where failure of a link is the same as failure of the sending node, that link is a part of the end as well.
- So, be careful in deciding what your "end" is in the e2e argument.



After-Effects of End to End Argument

- Putting less functionality in the network led to the evolution of an Internet that was so flexible that a wide variety of applications were able to use it
  - Arguably, this allowed HTTP, P2P, VoIP, file transfer, and a host of other applications to co-exist atop the Internet
  - It allowed TCP and the "narrow waist" to evolve (as we'll see next)

Prelude to the Argument

- 1950's: Tape drives were the rage in storage devices
  - Lot of researchers attempted to design an absolutely reliable tape subsystem
  - They kept failing
  - Eventually everyone realized that applications would always have to check for errors in data obtained from tape drives
- Banks use ATMs, multiple servers, etc.: however high-level auditing procedures are needed to serve as a final check

The Religion

- End to end argument often becomes a religious argument
- What is your opinion of the end to end argument?
- With growing power of hardware (ends), should we put more functionality at the ends?
- Do the following violate or satisfy the end to end argument? Should they violate it?
  A. Telephone system (think: voice quality)
  B. VoIP
  C. Active Networks
  D. Peer to peer systems
  E. Wireless ad-hoc networks/DTNs
  F. Sensor networks
  G. Cloud services/datacenters

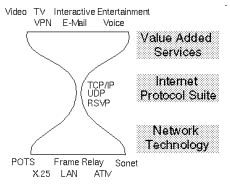
Middleboxes: Taxonomy and Issues

B. Carpenter, RFC 3234

- Middlebox (as defined by Lixia Zhang): an intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and a destination host.
- Everyone of us has used a middlebox.

The Hourglass model

- Widely accepted story of network stack evolution
- A variety of applications
- A variety of low-level networking hardware and protocols
- A small set of mid-level networking protocols



Source: http://www.isoc.org

Why are Middleboxes challenging?

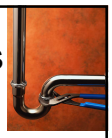
Middleboxes (MBs):

- Challenge old protocols
  - Protocols designed without MBs in mind (i.e., built with the end to end principle) might fail
- Introduce new failure modes
  - E.g., MB crashes, path rerouting
- Configuration is no longer end to end
- Diagnosis of failures and misconfigurations is complex

Examples of Middleboxes

- NAT: Network Address Translator
  - Assigns a host a unique IP address without its knowledge
  - Rewrites packets on the fly to and from that host
  - E.g., most wireless access points (at home)
  - Usually accompanied by a gateway
  - NAT-PT also translates between different protocols, e.g, IPv4 and IPv6.

Examples of Middleboxes



- IP Tunnel Endpoints
  - Consider two IPv6 hosts communicating over an IPv4 network
  - Two tunneling servers reformat IPv6 packets from these hosts, and IPv4 packets to these hosts
- Packet Classifiers, Markers, Schedulers: Affect the way packets are routed
- SOCKS server: for traversing firewalls; stateful

Examples of Middleboxes

- Transport Relays: Rewrite TCP packets,
- TCP performance enhancing proxies: TCP spoofer that changes timing of TCP packets for improving throughput
  - TCP packet shapers have overcome TCP's shortcomings
  - TCP-friendliness?
- E.g., Resilient Overlay Networks (RON) rewrites TCP packets to route them via other RON nodes
- E.g., Variety of techniques on TCP rerouting

Examples of Middleboxes

- Proxies: Most popular are web proxies, which allow a client to access it like a server for all webpages, and which in turn are a client to the outside world's servers
  - Makes configurations easy
- Caches: cache results, e.g., web caches
- IP Firewalls: protect a network by disallowing traffic from coming inside
  - Enhancing security without changing the OS/end
- Application Firewalls
- Application-level Gateways: generalization of proxies, e.g., you could use UIUC's PlanetLab nodes as ALGs

Examples of Middleboxes

- Modified DNS servers: Akamai's original protocol added several entries to DNS, so that a given URL was mapped to "nearby" Akamai servers (throughout the world)
- CDN boxes: Content Distribution Boxes, e.g., IPTV servers, DNS servers that rewrite URLs.

Middleboxes vs. End to end

- Are middleboxes a violation of the end to end principle?
- Apparently Yes, but let's go back to the E2E arg:
  - Do they enhance convenience?
  - Permit more useful work that was not otherwise possible at the ends?
  - Or do they affect performance so negatively that they are undesirable?
- Does the end to end argument foretell middleboxes?
  - Do NATs, proxies, caches, IP tunnels, transport relays improve functionality from a place where no work can be done to a place where useful work can be done?
  - Or are they merely a performance improvement technique?

Coming back to the End-to-end argument

- RFC's conclusion: Need to design future application protocols to be aware of the existence of Middleboxes.
- Your conclusion: ?
- Example -- For botnets: is the probability of threat high enough nowadays that we should add anti-botnet functionality to the network?

An End to the Middle

C. Dixon et al, HotOS 2009

Disadvantages of Middleboxes

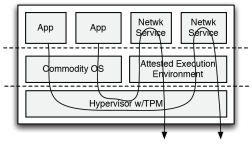
- Costs:
  - upfront hardware
  - management personnel
  - scaling under increased loads
  - overprovisioning
- Deployment:
  - throughout the network, rather than just at edges
  - dealing with mobile devices
- As per the E2E argument paper, the $ cost + management seems high for middleboxes!

Do we need Middleboxes at all?

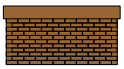
- Why not get rid of them completely?
- Instead, have intelligent endpoints
  - Endpoints decide, inject, and enforce policies
  - Middleboxes implement policies

End to the Middle (ETTM)

- Key idea: run trusted code in a separated VM on each end node (e.g., proxy)
- Trusted code decides and enforces policies for middleboxes
- Separate and modularized from other OS functionalities
- Like active networks, gives control to end nodes over data @ routers



More Concretely



- For NAT boxes
  - The main criticism is that they perform stateful routing
  - ETTM:
    - Keep logical address translation tables at each end host (or a subset)
    - Run an agreement/consensus protocol across these hosts
      - Cost? Overhead?
    - This also makes MBs and ends consistent so either can do the work.

Caveats?

Going back to the E2E argument:

- Doesn't requiring all ends to run the same policy and VMs actually violate the end to end principle?
  - If it's commonly needed by many nodes, put it inside the network!
- Does moving policy enforcement from middleboxes to ends actually increase functionality and useful work?
- Will eliminating middleboxes really reduce functionality at the ends to such an extent that useful work can no longer be done?

Your opinions…

- How do you feel about the argument?
- Do you use it? Have you ever used it? Will you never use it?
- Is it still valid today?
- Is/are variant(s) of the original argument valid today?