CS551
End to End Argument
[Saltzer81a]

Bill Cheng

http://merlot.usc.edu/cs551-f12

The End-to-end Argument

- Deals with *where* to place protocol functionality (e.g., encryption, reliability, ordering, duplication surpression):
  - *inside* the network (in switching elements), or
  - at the edges
- Not an arhchitecture in itself, but an architectural principle
  - other architecture can use this principle
    - e.g., architectures for transaction management

Key Ideas

- The end-to-end argument
  - don't duplicate functionality in multiple levels if you *have to* do it at the top anyway
  - apply to networking: the lower layers of the network are not the right place to implement *application-specific* functions (the lower network layers should implement basic and general functions)
    - move these functions *up* and *out*
    - the network should be as transparent as technology permits
- Duplicate functionality has a cost associated with it
  - better spend it on other things
- Need to be general: Additional functionality may help some but may actually hurt other applications

Example: Reliability

- Consider copying a file
  - want an end-to-end checksum, even if network guarantees reliable delivery
- Steps:
  - A reads from disk to memory; sends over network
  - network moves data from A to B
  - B gets data from network; writes to disk
- Possible faults:
  - disk I/O errors, buffer overruns in NIC, memory errors, network corruption or congestion, computer crashes
- Recommendation: in order to achieve reliable file transfer, application program must supply a file-transfer-specific, end-to-end reliable guarantee (and not rely on the data communication system)

Other Examples

- Encrypted data transfer (cannot trust the network)
- Duplicate message suppression (did I just double-click the button or single-clicked it?)
- Guaranteed FIFO message delivery
- Transactions in a DB

Caveat: Performance

- Consider file copy again
- Reliability at physical, link, network, transport, application layers
  - need some physical redundancy (coding)
  - sometimes want link repair (Ethernet retransmission after collision, wireless links)
  - network level repair (TCP)
  - application level checks (checksum)
- multiple levels may be needed for *performance*, not correctness

End2End: A Broader View

- What breaks end2end connectivity?
  - NAT
  - Web caches
  - Transparent web proxies
  - Others?
- Is this bad and why?
  - NAT - who’s talking?
  - Web caches - web page out of date, server wants to keep a hit count
  - Transparent web proxies - where are the ads? what else is filtered out?

Difficulty: What Is the "End"?

- Consider secure communication:
  - me to my bank over HTTPS: browser to commerce server
  - me from home to USC over ssh: app-level
  - my computer to USC over a virtual private network (VPN): network-layer on my computer to USC network
  - my computer to the wireless base-station over 802.11 with WEP: link-layer on my computer to wireless LAN
  - my PIN number in my head to the ATM (?)
- Lower-layers have benefits (wider coverage)
  - but may increase risks
- End-to-end argument is *not* an absolute rule (like Occam’s razor)
  - rather a *guideline* that helps in application and protocol design analysis

Discussion

- Summary
  - don’t put functionality inside the network when it would have to be duplicated at the ends anyway
- Context
  - came well before much of the Internet had been built
- Impact
  - arguably the most influential paper in the history of networking
  - measure of worth: not many papers are remembered after 20 years
  - helpful for understanding the success of the Internet
  - people tend to use it to justify/dispute everything
    - active networks, sensor networks, etc.