CS 268: Internet Architecture & E2E Arguments

Scott Shenker and Ion Stoica
(Fall, 2010)

Today’s Agenda

- Design goals
- Layering (review)
- End-to-end arguments (review)
Internet Design Goals

Goals

0 Connect existing networks
   • Initially ARPANET and ARPA packet radio network

1. Survivability
   • Ensure communication service even in the presence of network and router failures

2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Must be cost effective
6. Allow host attachment with a low level of effort
7. Allow resource accountability
Connect Existing Networks

- Existing networks: ARPANET and ARPA packet radio
- Decision: packet switching
  - Existing networks already were using this technology
- Packet switching $\rightarrow$ store and forward router architecture

- Internet: a packet switched communication network consisting of different networks connected by store-and-forward routers

Survivability

1. As long as the network is not partitioned, two endpoints should be able to communicate
2. Failures (excepting network partition) should not interfere with endpoint semantics (why?)

- Maintain state only at end-points
  - Fate-sharing, eliminates network state restoration
  - Stateless network architecture (no per-flow state)

- Routing state is held by network (why?)
- No failure information is given to ends (why?)
Types of Services

- Use of the term “communication services” already implied that they wanted application-neutral network

- Realized TCP wasn’t needed (or wanted) by some applications

- Separated TCP from IP, and introduced UDP
  - What’s missing from UDP?

Variety of Networks

- Incredibly successful!
  - Minimal requirements on networks
  - No need for reliability, in-order, fixed size packets, etc.

- IP over everything
  - Then: ARPANET, X.25, DARPA satellite network...
  - Now: ATM, SONET, WDM…
Why Datagrams?

- No connection state needed
- Good building block for variety of services
- Minimal network assumptions

Questions

- What priority order would a commercial design have?
- What would a commercially invented Internet look like?
- What goals are missing from this list?
- Which goals led to the success of the Internet?
Key Advantages

- The service can be implemented by a large variety of network technologies
- Does not require routers to maintain any fine grained state about traffic. Thus, network architecture is
  - Robust
  - Scalable

Layering and other General Mutterings about Internet Architecture

Repeats122 material
The Big Question

- Many different network styles and technologies
  - circuit-switched vs packet-switched, etc.
  - wireless vs wired vs optical, etc.

- Many different applications
  - ftp, email, web, P2P, etc.

- How do we organize this mess?

The Problem

- Do we re-implement every application for every technology?
- Obviously not, but how does the Internet architecture avoid this?
Architecture

- Architecture is not the implementation itself
- Architecture is how to “organize” implementations
  - what interfaces are supported
  - where functionality is implemented
- Architecture is the modular design of the network

Software Modularity

Break system into modules:

- Well-defined interfaces gives flexibility
  - can change implementation of modules
  - can extend functionality of system by adding new modules
- Interfaces hide information
  - allows for flexibility
  - but can hurt performance
Network Modularity

Like software modularity, but with a twist:

- Implementation distributed across routers and hosts

- Must decide both:
  - how to break system into modules
  - where modules are implemented

- Lecture will address these questions in turn

Two Aspects to Architecture

- Layering
  - how to break network functionality into modules

- The End-to-End Argument
  - where to implement functionality
Layering

- Layering is a particular form of modularization
- The system is broken into a vertical hierarchy of logically distinct entities (layers)
- The service provided by one layer is based solely on the service provided by layer below
- Rigid structure: easy reuse, performance suffers

ISO OSI Reference Model for Layers

- Application
- Presentation $\rightarrow$ (data conversion, encryption, decryption)
- Session $\rightarrow$ (connection management, e.g., open, close)
- Transport $\rightarrow$ (proc-to-proc comm., reliability, in-order delivery, flow ctrl)
- Network $\rightarrow$ (addressing, routing, fragmentation)
- Datalink $\rightarrow$ (framing, addressing, error detection & correction)
- Physical $\rightarrow$ (media, bit-level encoding and transmission)
Where Do These Fit?

- IP
- TCP
- Email
- Ethernet

Layering Solves Problem

- Application layer doesn’t know about anything below the presentation layer, etc.

- Information about network is hidden from higher layers

- This ensures that we only need to implement an application once!
OSI Model Concepts

- Service: what a layer does

- Service interface: how to access the service
  - interface for layer above

- Peer interface (protocol): how peers communicate
  - a set of rules and formats that govern the communication between two network boxes
  - protocol does not govern the implementation on a single machine, but how the layer is implemented between machines

Who Does What?

- Seven layers
  - Lower three layers are implemented everywhere
  - Next four layers are implemented only at hosts

```plaintext
Host A               Host B
Application          Application
Presentation         Presentation
Session              Session
Transport            Transport
Network             Network             Network
Datalink             Datalink             Datalink
Physical             Physical             Physical
                       Physical medium
```
Logical Communication

- Layers interact with corresponding layer on peer

Physical Communication

- Communication goes down to physical network, then to peer, then up to relevant layer
Encapsulation

- A layer can use only the service provided by the layer immediate below it.
- Each layer may change and add a header to data packet.

OSI vs. Internet

- OSI: conceptually define services, interfaces, protocols.
- Internet: provide a successful implementation.
Hourglass

A single Internet layer module:

- Allows all networks to interoperate
  - all networks technologies that support IP can exchange packets
- Allows all applications to function on all networks
  - all applications that can run on IP can use any network
- Simultaneous developments above and below IP
Back to Reality

- Layering is a convenient way to think about networks

- But layering is often violated
  - Firewalls
  - Transparent caches
  - NAT boxes
  - .......

- What problems does this cause?

Endless Arguments about End-to-End
Placing Functionality

- The most influential paper about placing functionality is “End-to-End Arguments in System Design” by Saltzer, Reed, and Clark

- The “Sacred Text” of the Internet
  - endless disputes about what it means
  - everyone cites it as supporting their position

Basic Observation

- Some applications have end-to-end performance requirements
  - reliability, security, etc.

- Implementing these in the network is very hard:
  - every step along the way must be fail-proof

- The hosts:
  - can satisfy the requirement without the network
  - can’t depend on the network
Example: Reliable File Transfer

- Solution 1: make each step reliable, and then concatenate them
- Solution 2: end-to-end check and retry

Example (cont’d)

- Solution 1 not complete
  - What happens if any network element misbehaves?
  - The receiver has to do the check anyway!

- Solution 2 is complete
  - Full functionality can be entirely implemented at application layer with no need for reliability from lower layers

- Is there any need to implement reliability at lower layers?
Conclusion

Implementing this functionality in the network:
- Doesn’t reduce host implementation complexity
- Does increase network complexity
- Probably imposes delay and overhead on all applications, even if they don’t need functionality
- However, implementing in network can enhance performance in some cases
  - very lossy link

What the Paper Says

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points 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.)
Conservative Interpretation

- “Don’t implement a function at the lower levels of the system unless it can be completely implemented at this level” (Peterson and Davie)

- Unless you can relieve the burden from hosts, then don’t bother

Radical Interpretations

- Don’t implement anything in the network that can be implemented correctly by the hosts
  - e.g., multicast
  - Makes network layer absolutely minimal
  - Ignores performance issues

- Don’t rely on anything that’s not on the data path
  - E.g., no DNS
  - Makes network layer more complicated
**Moderate Interpretation**

- Think twice before implementing functionality in the network

- If hosts can implement functionality correctly, implement it at a lower layer **only** as a performance enhancement

- But do so only if it does not impose burden on applications that do not require that functionality

**Extended Version of E2E Argument**

- Don’t put application semantics in network
  - Leads to loss of flexibility
  - Cannot change old applications easily
  - Cannot introduce new applications easily

- Normal E2E argument: performance issue
  - Introducing more functionality imposes more overhead
  - Subtle issue, many tough calls (e.g., multicast)

- Extended version:
  - Short-term performance vs long-term flexibility
Do These Belong in the Network?

- Multicast?
- Routing?
- Quality of Service (QoS)?
- Name resolution? (is DNS “in the network”?)
- Web caches?

Back to Reality (again)

- Layering and E2E Principle regularly violated:
  - Firewalls
  - Transparent caches
  - Other middleboxes

- Battle between architectural purity and commercial pressures
  - extremely important
  - imagine a world where new apps couldn’t emerge
Challenge

- Install functions in network that aid application performance…

- …without limiting the application flexibility of the network