EECS 122: Introduction to Computer Networks

Network Architecture

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1776

A Quick Review

- 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

- Re-implement every application for every technology?
- No! But how does the Internet architecture avoid this?

Today’s Lecture: Architecture

- Architecture is \textit{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
  - Change implementation of modules
  - 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:
  - How to break system into modules
  - Where modules are implemented

- We will address these questions in turn
Layering

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

- Introduce an intermediate layer that provides a single abstraction for various network technologies
  - A new app/media implemented only once
  - Variation on “add another level of indirection”

ISO OSI Reference Model for Layers

- Application
- Presentation
- Session
- Transport
- Network
- Datalink
- Physical

ISO: International Standards Organization
OSI: Open System Interface
Layering Solves Problem

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

- Information about network is hidden from higher layers

- Ensures that we only need to implement an application once!

- Caveat: not quite....

---

OSI Model Concepts

- **Service**: what a layer does

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

- **Protocol** (peer interface): how peers communicate
  - 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
Physical Layer (1)

- **Service**: move information between two systems connected by a physical link
- **Interface**: specifies how to send a bit
- **Protocol**: coding scheme used to represent a bit, voltage levels, duration of a bit
- Examples: coaxial cable, optical fiber links; transmitters, receivers

Datalink Layer (2)

- **Service**:  
  - Framing (attach frame separators)  
  - Send data frames between peers  
  - Others:  
    - arbitrate the access to common physical media  
    - per-hop reliable transmission  
    - per-hop flow control
- **Interface**: send a data unit (packet) to a machine connected to the same physical media
- **Protocol**: layer addresses, implement Medium Access Control (MAC) (e.g., CSMA/CD)...
## Network Layer (3)

- **Service:**
  - Deliver a packet to specified network destination
  - Perform segmentation/reassemble
  - Others:
    - packet scheduling
    - buffer management

- **Interface:** send a packet to a specified destination

- **Protocol:** define global unique addresses; construct routing tables

## Transport Layer (4)

- **Service:**
  - Demultiplexing
  - Optional: error-free and flow-controlled delivery

- **Interface:** send message to specific destination

- **Protocol:** implements reliability and flow control

- **Examples:** TCP and UDP
Session Layer (5)

- **Service:**
  - Full-duplex
  - Access management (e.g., token control)
  - Synchronization (e.g., provide check points for long transfers)

- **Interface:** depends on service

- **Protocol:** token management; insert checkpoints, implement roll-back functions

Presentation Layer (6)

- **Service:** convert data between various representations

- **Interface:** depends on service

- **Protocol:** define data formats, and rules to convert from one format to another
Application Layer (7)

- **Service**: any service provided to the end user
- **Interface**: depends on the application
- **Protocol**: depends on the application
- **Examples**: FTP, Telnet, WWW browser

Who Does What?

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

- Layers interacts 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

Example: Postal System

Standard process (historical):
- Write letter
- Drop an addressed letter off in your local mailbox
- Postal service delivers to address
- Addressee reads letter (and perhaps responds)
Postal Service as Layered System

Layers:
- Letter writing/reading
- Delivery

Information Hiding:
- Network need not know letter contents
- Customer need not know how the postal network works

Encapsulation:
- Envelope

Standards Bodies

- ISO: International Standards Organization
  - Professional bureaucrats writing standards
  - Produced OSI layering model

- IETF: Internet Engineering Task Force
  - Started with early Internet hackers
  - More technical than bureaucratic

“*We reject kings, presidents, and voting. We believe in rough consensus and running code*” (David Clark)
OSI vs. Internet

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

OSI (formal) Internet (informal)

Hourglass

e-mail WWW phone...
SMTP HTTP RTP...
TCP UDP...

IP

ethernet PPP...
CSMA async sonet...
copper fiber radio...
Implications of Hourglass

Single Internet layer module:

- Allows networks to interoperate
  - Any network technology that supports IP can exchange packets

- Allows applications to function on all networks
  - 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
  - .......
- More on this later....on to part two of this lecture

- Questions?
Placing Functionality

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

- “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

- 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

Discussion

- Solution 1 not complete
  - What happens if any network element misbehaves?
  - 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?
Summary

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
  - E.g., very lossy link

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 Interpretation

- Don’t implement anything in the network that can be implemented correctly by the hosts
  - E.g., multicast

- Make network layer absolutely minimal
  - Ignore performance issues

Moderate Interpretation

- Think twice before implementing functionality in the network

- If hosts can implement functionality correctly, implement it 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
## Summary

- Layering is a good way to organize networks
- Unified Internet layer decouples apps from networks
- E2E argument encourages us to keep IP simple
- Commercial realities may undo all of this...