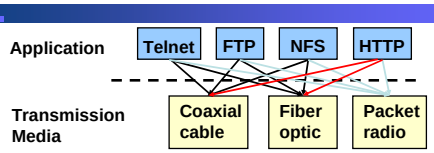
Internet Protocols
Fall 2005

Lecture 2
Andreas Terzis
terzis@cs.jhu.edu

Outline

- Internet Architecture
  - Layering
  - Encapsulation
  - E2E Principle

The Problem



- Many different applications and underlying transmission technologies
- Do we re-implement every application for every technology?
- Obviously not, but how does the Internet architecture avoid this?

Today's Lecture: Architecture

- The assignment of tasks and knowledge
  - Who does what, and where is the state kept
- 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

Internet Architecture

- Routers do routing, and almost nothing else
  - No application-specific functions
- Hosts do all application-specific processing
- This allowed wide variety of applications to flourish on Internet

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

Outline

- 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)
- (Session)
- Transport
- Network
- Datalink
- Physical

Layering Solves Problems

- 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!
- Caveat: not quite....

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

Next Five (~~Seven~~) Slides

- Will summarize each layer
- A good time for a nap....

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 process
- 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)

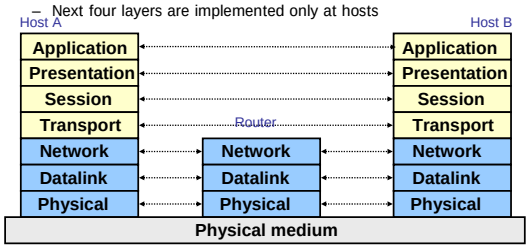
- **Service**: convert data between various representations
- **Interface**: depends on service
- **Protocol**: define data formats, and rules to convert from one format to another

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

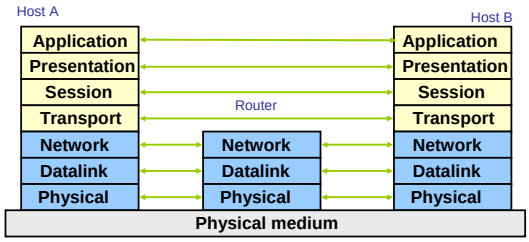
Who Does What?

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



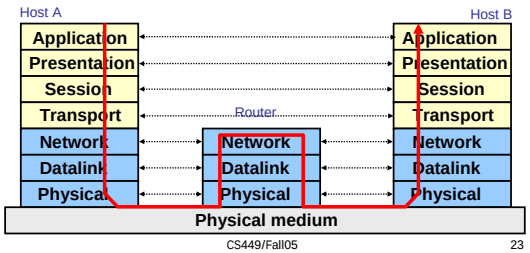
Logical Communication

- Layers interacts with corresponding layer on peer



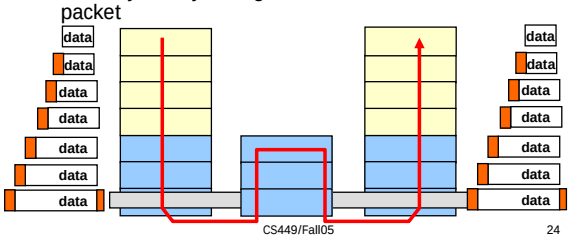
Physical Communication

- 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:

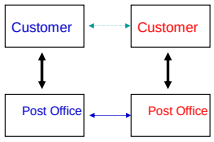
- Letter writing/reading
- Delivery

Information Hiding:

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

Encapsulation:

- Envelope



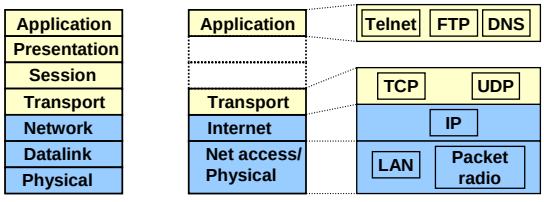
Standards Bodies

- 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

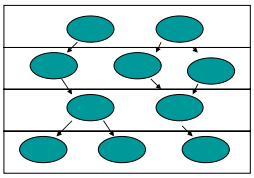
- OSI: conceptually define services, interfaces, protocols
- Internet: provide a successful implementation



Multiple Instantiations

- Can have several instantiations for each layer
  - many applications
  - many network technologies
  - transport can be reliable (TCP) or not (UDP)
- Applications dictate transport
  - In general, higher layers can dictate lower layer
- But this is a disaster!
  - applications that can only run certain networks

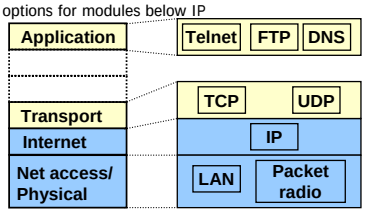
Multiple Instantiations of Layers



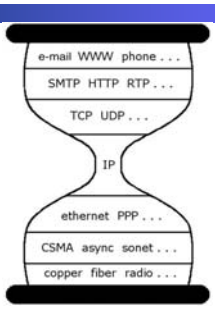
Solution

A universal Internet layer:

- Internet has only IP at the Internet layer
- Many options for modules above IP
- Many options for modules below IP



Hourglass



Implications of 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

Reality

- Layering is a convenient way to think about networks
- But layering is often violated
  - Firewalls
  - Transparent caches
  - NAT boxes
- Questions?

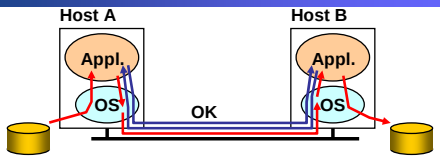
Placing Functionality

- 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

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

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

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

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 threaten to undo all of this...