CSE 123A
Computer Networks

Fall 2005

Lecture 2:
Protocols & Layering

Amin Vahdat
(standing in for Stefan Savage)
Today

- What’s a protocol?
- Organizing protocols via *layering*
- Encoding layers in packets
- The OSI & Internet layering models
- The end-to-end argument
There’s more to networking than sending bits? Like What?

- Sending bits to a specific destination
- Sending a long message to a particular destination
- Detecting if there was an error
- Fixing the error
- Deciding how fast to send
- Making sure the message is kept private
Definitions

- **Service**: A particular networking function (e.g. message delivery)
- **Protocol**: An implementation of a service (e.g. UDP)
- **Interface**: How a protocol is manipulated (e.g. packet format)

**Layering**
- Technique for organizing protocols into an *ordered* series of distinct abstractions
- The services provided by a layer depend **only** on the services provided by the previous, less abstract, layer
Why Layering?

- Example: without layering each new application has to be re-implemented for every network technology!

![Layering Diagram]

Courtesy Ion Stoica
Why Layering?

- Solution: introduce an intermediate layer that provides a unique abstraction for various network technologies.
Benefits of layering

- **Encapsulation**
  - Functionality inside a layer is self-contained; one layer doesn’t need to reason about other layers

- **Modularity**
  - Can replace a layer without impacting other layers
  - Lower layers can be reused by higher layers (e.g. TCP and UDP both are layered upon IP)

- One obvious drawback
  - Information hiding can produce *inefficient implementations*
  - Encryption layer and compression layer – independent?
Who decides what goes in the layers?

- **ISO OSI Reference Model**
  - ISO – International Standard Organization
  - OSI – Open System Interconnection
  - Designed by committee in 1978
  - Goal: open standard to support a market for vendors to compete on protocol implementation and design

- **Internet** layering model
  - Backporting of experience from ARPAnet (1969) and TCP/IP protocols (1974)
  - Shares much of OSI design
  - Roughly managed by the Internet Engineering Task Force (IETF)
The OSI layering Model

- Top 4 layers are end-to-end
- Lower 3 layers are hop-by-hop
Physical Layer (1)

- **Service**: move the 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:* where piece of data begins and ends
  - *Local addressing and delivery:* send data frames between peers attached to the same physical media
  - Others (sometimes):
    - Shared media access
    - Reliable transmission (resend missing packets)

- **Interface:** send a data unit (packet) to a machine connected to the same physical media

- **Protocol:** MAC addresses, media access control (MAC) implementation (e.g. Ethernet)
Network Layer (3)

- **Service:**
  - Deliver a packet to specified destination
  - Perform segmentation/reassemble (fragmentation/defragmentation)
  - Sometimes:
    - Packet scheduling: order packets are sent
    - Buffer management: what if there are too many packets?

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

- **Protocol:** global unique addresses, construct routing tables, forward packets towards destination
Transport Layer (4)

- Service:
  - Provide an **error-free** and **flow-controlled** end-to-end connection
  - Multiplex multiple transport connections to one network connection

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

- **Protocol**: implement reliability and flow control (e.g. TCP and UDP)
Session Layer (5)

- **Service**
  - Session management
  - Synchronization, e.g., between audio/video streams

- **Interface**: depends on service

- **Protocols**: full duplex connection
  - setup/teardown, restart and checkpointing, inter-session synchronization (e.g. SMIL)
Presentation Layer (6)

- **Service**: data format conversion
- **Interface**: depends on service
- **Protocol**: define data formats, and rules to convert from one format to another (e.g. NFS’s XDR)
Application Layer (7)

- **Service**: any service provided to the end user
- **Interface**: depends on the application
- **Protocol**: depends on the application

- Examples: FTP, Kazaa, SSHt, HTTP
The Internet layering model

- Application (Web, FTP, SMTP)
- Network (IP)
- Transport (TCP, UDP)
- Datalink (Ethernet, 802.11)
- Physical (100BaseTX, 1000BaseSX)

• So-called “hourglass” model
  • One network layer protocol
  • Significant diversity at other layers

• No presentation or session layers

• Implementations more important than interfaces
Layer Encapsulation via Packet Headers

- Typical Web packet

```
| EthernetHdr | IPHdr | TCPhdr | HTTPHdr | Payload (Web object) |
```

- Notice that layers add overhead
  - Space (headers), effective bandwidth
  - Time (processing headers, peeling the onion), latency
Layer encapsulation via packet headers

Layer N+1 packet becomes Layer N data
Layering by example...

- **ROUGHLY**, what happens when I click on a Web page from UCSD?

![Diagram showing a flow from My computer to Internet to www.yahoo.com](Image)
Application layer (HTTP)

- Turn click into HTTP request

GET http://www.yahoo.com/r/mp HTTP/1.1
Host: www.yahoo.com
Connection:keep-alive

...
Application layer?
Name resolution (DNS)

- Where is www.yahoo.com?

My computer (132.239.9.64)

Local DNS server (132.239.51.18)

What’s the address for www.yahoo.com

Oh, you can find it at 64.58.76.177
Transport layer (TCP)

- Break message into packets (TCP segments)
- Should be delivered reliably & in-order

GET http://www.yahoo.com/r/mp HTTP/1.1
Host: www.yahoo.com
Connection: keep-alive

"and let me know when they got there"
Network layer: IP Addressing

- Address each packet so it can traverse network and arrive at host

My computer (132.239.9.64)

www.yahoo.com (64.58.76.177)

<table>
<thead>
<tr>
<th>Destination</th>
<th>Source</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>64.58.76.177</td>
<td>132.239.9.64</td>
<td>GET http</td>
</tr>
</tbody>
</table>
Network layer: IP Routing

- Each router forwards packet towards destination
Datalink layer (Ethernet)

- Break message into frames
- Media Access Control (MAC)
  - Can I send now? Can I send now?
- Send frame
Physical layer

2.4Ghz Radio
DS/FH Radio
(1-11Mbps)

802.11b Wireless Access Point

Cat5 Cable (4 wires)
100Base TX Ethernet
100Mbps

Ethernet switch/router

To campus backbone

62.5/125um 850nm MMF
1000BaseSX Ethernet
1000Mbps
Key Design Decision

- How do you divide functionality across the layers?

- End-to-end argument [Saltzer84]
  - Functionality should be implemented at a lower layer iff it can be correctly and completely implemented there
  - Incomplete versions of a function can be used as a performance enhancement, but not for correctness

- Early, and still relevant, example
  - ARPAnet provided reliable link transfers between switches
  - Packets could still get corrupted on host-switch link, or inside of the switches
  - Hence, still need reliability at higher layers
Example: Reliable File Transfer

- Where can data be corrected?
- How to tell if data has been corrupted?
- Is there any value in lower-layer reliability?
Violating the E2E argument for performance optimization

- Functionality at lower layer can enhance performance
  - Not required for correct operation
  - Can be required for reasonably efficient operation
- Back of the envelope
  - $N$ hops (average hops on Internet route = 15 hops)
  - $\text{Prob (corrupted packet per link)} = p$
  - $\text{Prob (packet lost end to end)}$
    - $p = 0.0001\% \rightarrow \text{Prob (e2e loss)} = 0.0015\%$
    - $p = 1\% \rightarrow \text{Prob (e2e loss)} = 14\%$
    - Reasonable to implement additional reliability in the datalink layer for the 2nd case
Optimization trade-offs

- Higher layers have more semantic information about service needs
  - (e.g. video: bits comprising MPEG I frames are much more important that bits in MPEG B frames)
- Lower layers have more information about true capabilities
  - (e.g. packet size, bandwidth, error rate)
- This tension is the subject of countless papers…
Summary: layering

- Key technique to implement communication protocols; provides
  - Modularity
  - Abstraction
  - Reuse
- Key design decision: what functionality to put in each layer?

- Read Chap 2-2.2
- Stefan’s going to cover signaling, coding and clock recovery