CSE/EE 461
Protocols and Layering
Protocols

• We need abstractions to handle communication system complexity
• A protocol is an agreement dictating the form and function of data exchanged between parties to affect communication
• Two parts:
  – Syntax: Words.
    • where the bits go
  – Semantics: Meaning
    • what the words mean, what to do with them
• Examples:
  – Ordering pizza
  – International relations
  – IP, the Internet protocol
  – TCP and HTTP, for the Web
Protocol Standards

• Different functions require different protocols
• Thus there are many protocol standards
  – E.g., IP, TCP, UDP, HTTP, DNS, FTP, SMTP, NNTP, ARP, Ethernet/802.3, 802.11, RIP, OPSF, 802.1D, NFS, ICMP, IGMP, DVMRP, IPSEC, PIM-SM, BGP, ...
• Organizations: IETF, IEEE, ITU
• IETF (www.ietf.org) specifies Internet-related protocols
  – RFCs (Requests for Comments)
  – “We reject kings, presidents and voting. We believe in rough consensus and running code.” – Dave Clark.
Layering and Protocol Stacks

- Layering is how we combine protocols
  - Higher level protocols build on services provided by lower levels
  - Peer layers communicate with each other
Example – Layering at work

<table>
<thead>
<tr>
<th>host</th>
<th>router</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>IP</td>
<td>TCP</td>
</tr>
<tr>
<td>IP</td>
<td>Ethernet</td>
<td>IP</td>
</tr>
<tr>
<td>Ethernet</td>
<td>CATV</td>
<td>Ethernet</td>
</tr>
</tbody>
</table>

The diagram illustrates the layering at work in a network, with hosts, routers, and various network protocols such as TCP, IP, Ethernet, and CATV.
Layering Mechanics

• Encapsulation and deencapsulation

Messages passed between layers

\[ \text{Hdr} + \text{Data} \]

\[ \text{Hdr} + \text{Data} \]
Speaking Abstractly

• Suppose you have something to express
• Must have an utterance (**message**)  
  – Must choose a language (**protocol**) with which to utter  
  – Must identify a person (**destination**) to whom to utter  
  – Must identify self (**source**) in order to receive a response  
  – Must include some control information “outside” the utterance
• Use the **protocol** to emit the **message** from the **source** to the **destination** with the **control** information
• All the way down
Abstract Send

- Think “M”, S and D.
- Call Protocol P with Message M to be sent from endpoint S to endpoint D
  - \( P_{\text{down}}(S, D, M) \)
- Protocol P adds header information to M
  - \( H = [S, D, C, P] \)
    - P is the type of protocol; used on delivery
    - S,D endpoints
    - C is control information.
      - S,D influence
    - \( M' = [H,M] \)
- Protocol P invokes a new protocol \( P' \) that can be used to communicate with P
  - \( P'_{\text{down}}(S', D', M') \)
    - With endpoints \( S' = \text{“this” end of P’} \), \( D' = \text{“that” end of P’} \)
- Continue until call chain terminates
  - Including just “dropping the message”
Abstract Deliver

- P\_up is a request from a lower level protocol to deliver a message using protocol P
- P\_up(M')
  - M' = [H, M]
  - Extract H = [S, D, C, P']
  - Process C accordingly (S, D are important here)
  - Invoke (Choose\_P\_up(P'))(M)
- Type identifier for P allows multiple protocols to “ride on top” of
  - Choose P\_up method based on type of next level up protocol.
For example

- Sample chains
  - HTTP->TCP->IP->ETHERNET
  - DNS->UDP->IP->FIBER

Type information (P) allows for lower level to “route” to higher level, e.g. lower level delivery reads a field written by a higher level send.
Protocol Graphs

- Multiplexing and demultiplexing in a protocol graph

```
  SMTP  HTTP
   |     |
   TCP  UDP
   |     |
   IP   ARP
   |
Ethernet
```

TYPE: IP protocol field

TYPE: 802.2 identifier
A Packet on the Wire

• Starts looking like an onion!

- Ethernet Hdr
- IP Hdr
- TCP Hdr
- HTTP Hdr
- Payload (Web object)

Start of packet

End of packet

• This isn’t entirely accurate
  - ignores segmentation and reassembly, Ethernet trailers, etc.
What’s Inside a Packet

A Protocol is an ENCAPSULATION.
One protocol’s payload may be another encapsulated protocol.
Deliver vs. Receive

- Networks deliver messages asynchronously
- When a message arrives, some program must service it
  - Interrupt handler
    - Synchronous with message delivery
  - OS Protocol
    - Asynchronous with message delivery
  - Application protocol
    - Asynchronous with message delivery
- Buffering is used to store messages between when they are delivered and when they are received (read) by an OS protocol or an Application protocol
  - Eg, recvfrom can block
    - Blocks if message has not yet arrived.
    - Does not block if message has already arrived and has been buffered by OS
OSI/Internet Protocol Stacks

Key Question: What functionality goes in which protocol?

The “End to End Argument” (Reed, Saltzer, Clark, 1984):

- Functionality should be implemented at a lower layer only if it can be correctly and completely implemented. (Sometimes an incomplete implementation can be useful as a performance optimization.)
- Tends to push functions to the endpoints, which has aided the transparency and extensibility of the Internet.
OSI “Seven Layer” Reference Model

- Seven Layers:
  - Application
  - Presentation
  - Session
  - Transport
  - Network
  - Link
  - Physical

Their functions:
- Your call
- Encode/decode messages
- Manage connections
- Reliability, congestion control
- Routing
- Framing, multiple access
- Symbol coding, modulation
Internet Protocol Framework

Model

- Application
- Transport
- Network
- Link

Protocols

- Many (HTTP, SMTP)
- TCP / UDP
- IP
- Many (Ethernet, …)
Key Concepts

- Protocol layers are the modularity that is used in networks to handle complexity
- Structure of a protocol layer is well-defined
- The Internet/OSI models give us a roadmap of what kind of function belongs at what layer