

Lecture 2:

Network Architecture

CSE 123: Computer Networks
Alex C. Snoeren



Our Problem

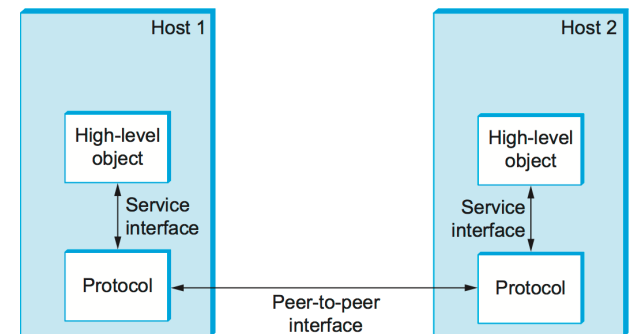


- Communications is complicated
 - ◆ Modulation and encoding bits
 - ◆ Splitting sequences of bits into packets
 - ◆ Fixing errors
 - ◆ Controlling access to network
 - ◆ Routing
 - ◆ Recovering from lost messages
 - ◆ Etc....
- Really hard to think about all of this and get it right
- Not all applications need all of it
- How to achieve interoperability?



Layering: A Modular Approach

- Sub-divide the problem
 - ◆ Each layer relies on services from layer below
 - ◆ Each layer exports services to layer above
- Interface between layers defines interaction
 - ◆ Hides implementation details (encapsulation)
 - ◆ Layers can change without disturbing other layers (modularity)
- Interface among peers in a layer is a **protocol**
 - ◆ If peers speak same protocol, they can interoperate





Key Design Decision

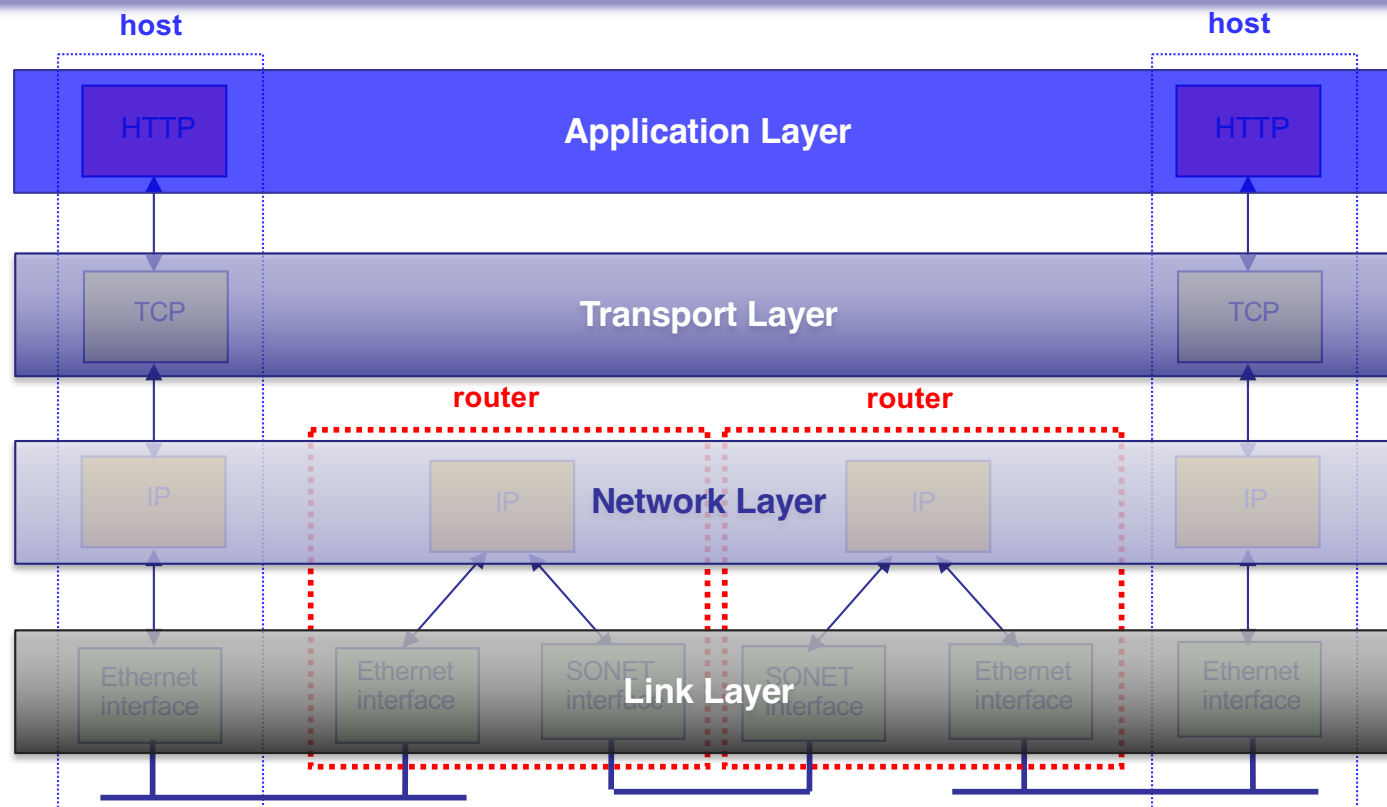
- How do you divide functionality across the layers?
- **End-to-end argument [Saltzer84]**
 - ♦ Functionality should be implemented at a lower layer **if and only if** it can be **correctly** and **completely** implemented there
 - ♦ Incomplete versions of a function can be used as a performance enhancement, but not for correctness
- Early, and still relevant, example
 - ♦ ARPAnet provided reliable link transfers between switches
 - ♦ Was this enough for reliable communication?
 - ♦ No, packets could still get corrupted on host-switch link, or inside of the switches
 - ♦ Hence, still need reliability at higher layers



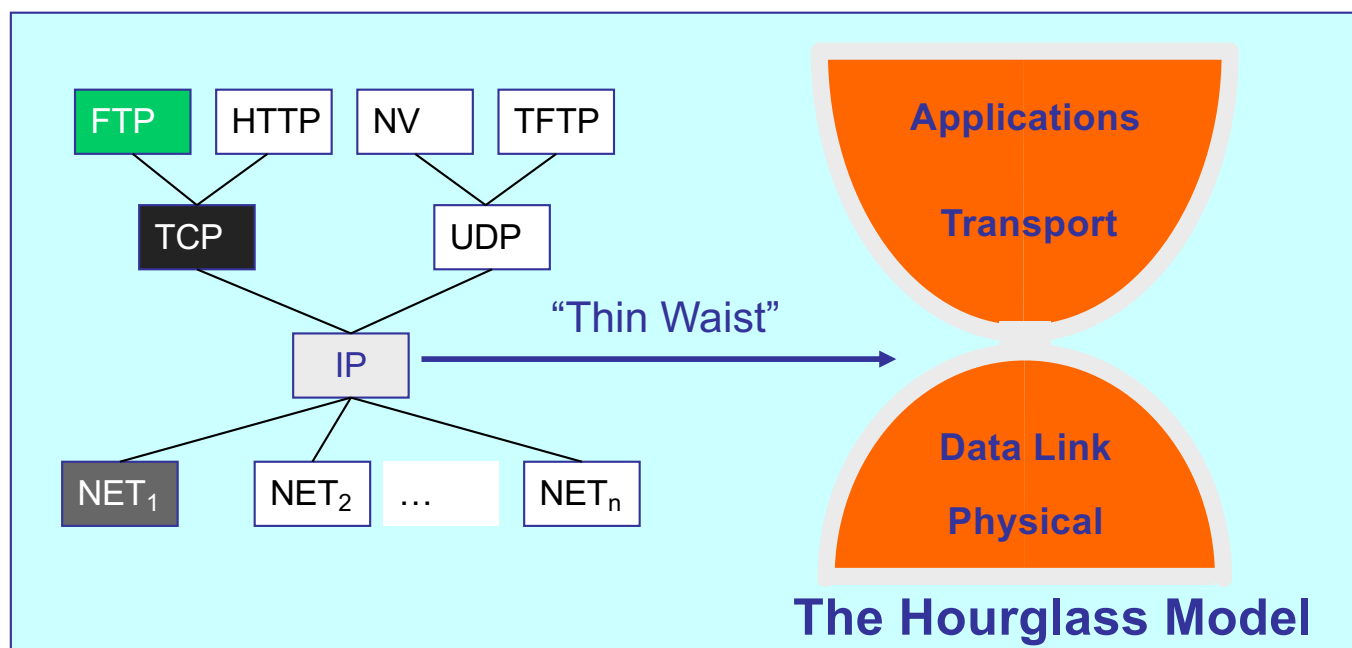
Protocol Standardization

- Communicating hosts speaking the same protocol
 - ◆ Standardization to enable multiple implementations
 - ◆ Or, the same folks have to write all the software
- Internet Engineering Task Force
 - ◆ Based on working groups that focus on specific issues
 - ◆ Produces “Request For Comments” (RFCs)
 - » Rough consensus and running code
 - » After enough time passes, promoted to Internet Standards
- Other standards bodies exist
 - ◆ ISO, ITU, IEEE, etc.

TCP/IP Protocol Stack



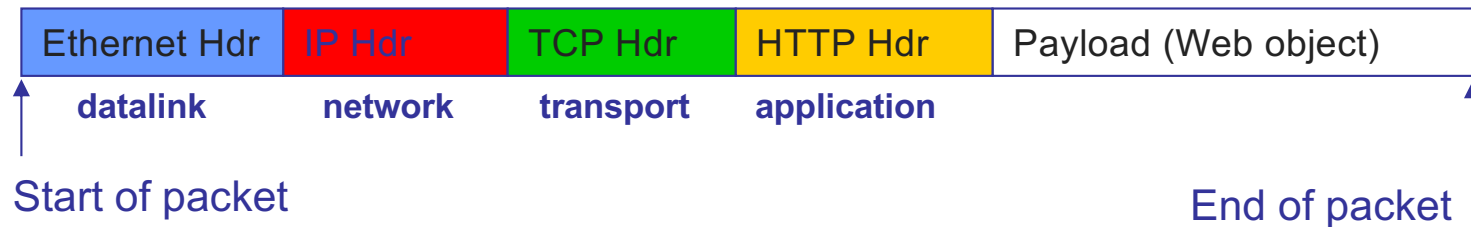
Internet Protocol Suite





Encapsulation via Headers

- Typical Web packet



- Notice that layers add overhead
 - ◆ Space (headers), effective bandwidth
 - ◆ Time (processing headers, “peeling the onion”), latency



Later: Phy/(MAC)Link layer

- Signal encoding
 - ◆ Encode binary data from source node into signals that physical links carry
 - ◆ Signal is decoded back into binary data at receiving node
 - ◆ Work performed by network adapter at sender and receiver
- Media access
 - ◆ Arbitrate which nodes can send messages at any point in time
 - ◆ Not always necessary; e.g. point-to-point duplex links



For now: (Data) Link Layer

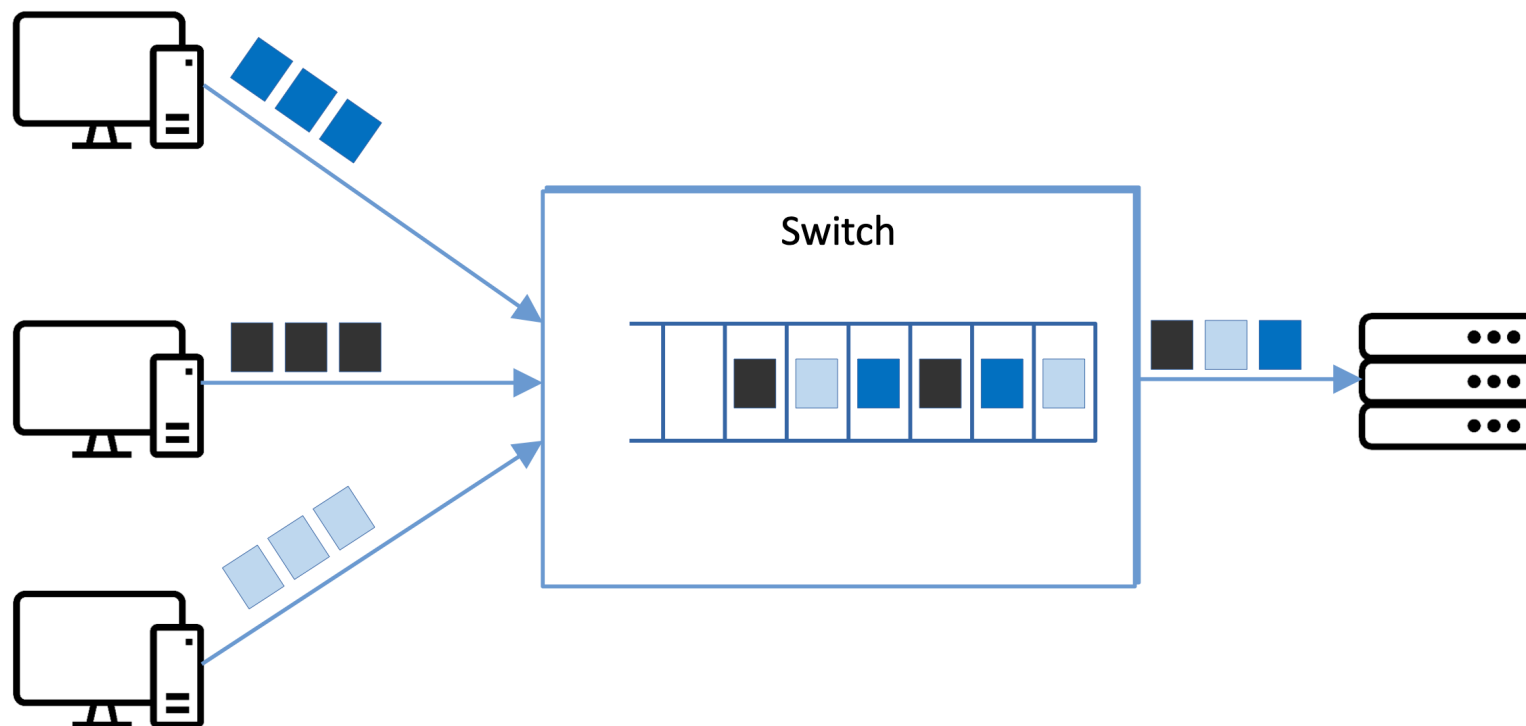
- **Framing**
 - ◆ Break messages up into discrete chunks called **frames** or **packets**
- Error handling
 - ◆ Detect and/or correct errors in received frames
- Multiplexing
 - ◆ Determine appropriate destination for a given frame
 - ◆ Also not always required; again, point-to-point

Framing



- Break down a message into smaller, digestible chunks called **frames**
 - ◆ Identify the beginning and the end of the message
- Allows the link to be shared
 - ◆ Multiple senders and/or receivers can **time multiplex** the link
 - ◆ Each frame can be separately addressed
- Provides manageable unit for error handling
 - ◆ Easy to determine whether something went wrong
 - ◆ And perhaps even to fix it if desired

Statistical multiplexing





What's a Frame?



- Wraps payload up with some additional information
 - ♦ Header usually contains addressing information
 - ♦ Maybe includes a trailer (w/checksum—next lecture)
- Basic unit of reception
 - ♦ Link either delivers entire frame payload, or none of it
 - ♦ Typically, some **maximum transmission unit (MTU)**
- Some link layers require absence of frames as well
 - ♦ I.e., minimum gaps between frames



Identifying Frames

- First task is to delineate frames
 - ◆ Receiver needs to know when a frame **starts** and **ends**
 - ◆ Otherwise, errors from misinterpretation of data stream
- Several different alternatives
 - ◆ Fixed length (bits) frames
 - ◆ Explicitly delimited frames
 - » Length-based framing
 - » Sentinel-based framing
 - ◆ Fixed duration (seconds) frames



Fixed-Length Frames

- Easy to manage for receiver
 - ♦ Well understood buffering requirements
- Introduces inefficiencies for variable length payloads
 - ♦ May waste space (padding) for small payloads
 - ♦ Larger payloads need to be **fragmented** across many frames
 - ♦ Very common inside switches
- Requires explicit design tradeoff
 - ♦ ATM uses 53-byte frames (cells)
 - ♦ Why 53? $48 + 5$. Why 48?

Length-Based Framing



- To avoid overhead, we'd like variable length frames
 - ♦ Each frame declares how long it is
 - ♦ E.g. Ethernet
- What's the issue with explicit length field?
 - ♦ Must correctly read the length field (bad if corrupted)
 - » Need to decode *while* receiving
 - ♦ Still need to identify the beginning...

For Next Class



- Read 3.1.3
- Project 1a out today; due in two weeks
- Introduced by Benny (project TA) in discussion at 4pm (WLH 2005)