CSEP 561 – Introduction

David Wetherall
djw@cs.washington.edu

Administrivia

- Teaching staff: David, Yanping and Seungyeop
- Web: http://www.cs.washington.edu/p561/
  - Points to everything (mailing list, discussion boards, …)
- Schedule and readings
  - Note makeup class on Tue 11/9 (optional)
- Textbook
  - Arrives at UW bookstore on Monday
- Homeworks
  - Four of them, spread over the quarter
- Final
  - Take-home out on 12/9 and due Tue 12/14

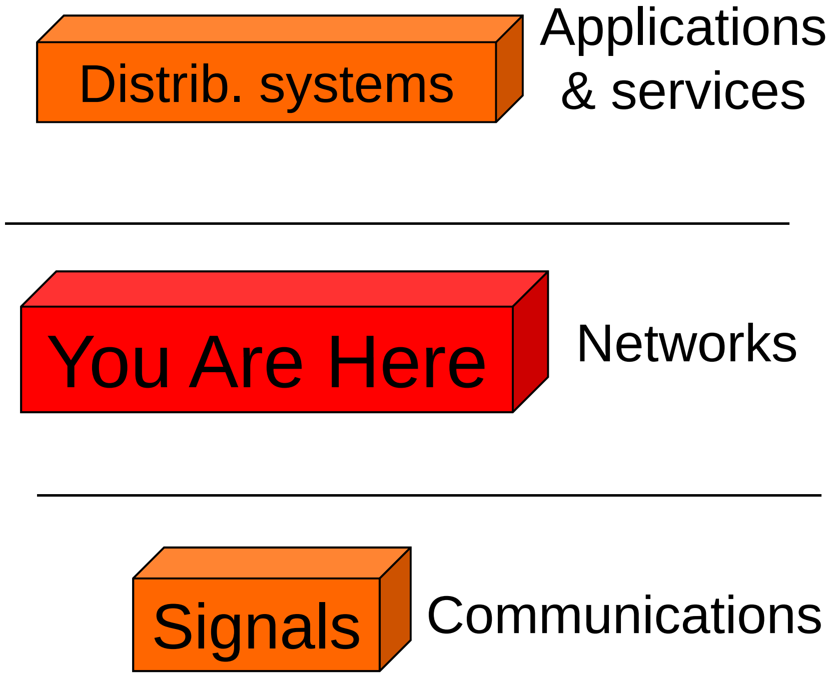
Your job

- Read the research papers before class
  - Submit a short post
- Read the assigned text before class
  - Bring your questions
- Do the homeworks on time
- Do the take-home final on time

- Learn all you want about network systems and have some fun along the way

Goal of this Course

- To understand how to design/build a variety of computer networks
  - Fundamental problems
  - Design principles
  - Implementation technologies
- This is a systems course, not queuing theory, signals, or hardware design.
- We focus on networks, and a bit on applications or services that run on top of them.



561 Syllabus and Key Concepts (Joke!)

- IP – internetworking
- …
- BGP – routing
- …
- TCP – reliability and congestion control
- …
- HTTP – the Web
- …

561 Syllabus and Key Concepts

- **Reliability – reliable distributed services from unreliable parts**
- *Soft-state. Fate-sharing. Error detection codes (checksums, CRCs). Acknowledgements and retransmissions (AR*Q). *Sliding window. Error correcting codes or FEC. TCP's three-way handshake. Link-state and distance vector routing*
- **Resource Sharing – cost-effective support for multiple users**
- *Statistical multiplexing. CSMA. AIMD. TCP congestion avoidance. TCP slow start. RED. Weighted fair queuing (WFQ). Token buckets. Generalized processor sharing (GPS) Load-sensitive routing. Adaptive applications. Over-provisioning*
- **Growth and Evolution – accommodating scale and heterogeneity**
- *Protocols* and *layering. Internetworking. E2E argument. Sliding window. Hierarchy. Naming. Caching. Replication..*
- **Different Interests – accommodating greed and malice**
- *Policy. Cookies. ECN nonce. Routing areas. TTL filtering. (need more here!)*

Networks and network technologies you know of …

- "The Internet"
- Wireless (802.11)
- DSL and cable
- Powerline networks
- RFID
- "3G cellular" mobile phone networks (UMTS)
- Old-fashioned phone network
- Enterprise networks
- ISPs and datacenters
- Satellites, space networks

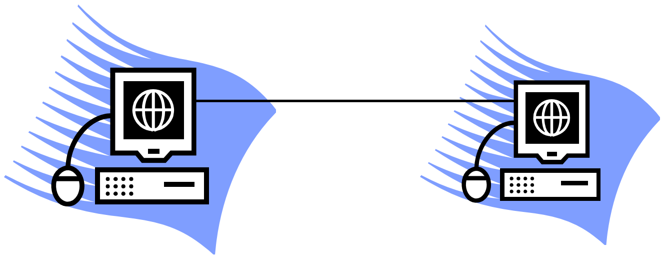
- This course is relevant to all these networks/technologies

Uses of networks

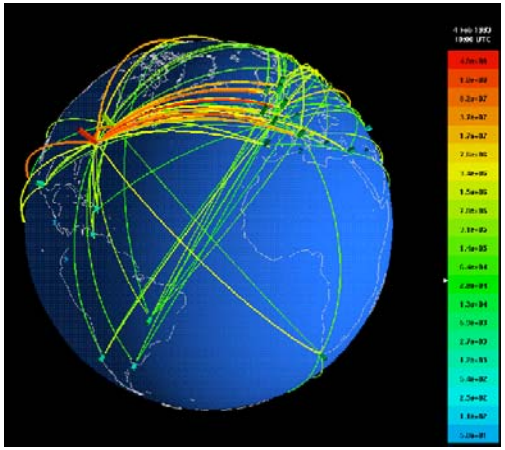
- Remote communication
  - Voice, video, email, text/instant messaging
  - For people, and for computers
- Information sharing
  - The Web, content distribution, P2P, social network apps
- Resource sharing
  - 3D printer, dataset in the cloud
- Connectivity between devices
  - E.g., consumer electronics in the home
- To link computers and the physical world?
  - Embedded computing/sensing, e.g., RFID

A Network from our point of view

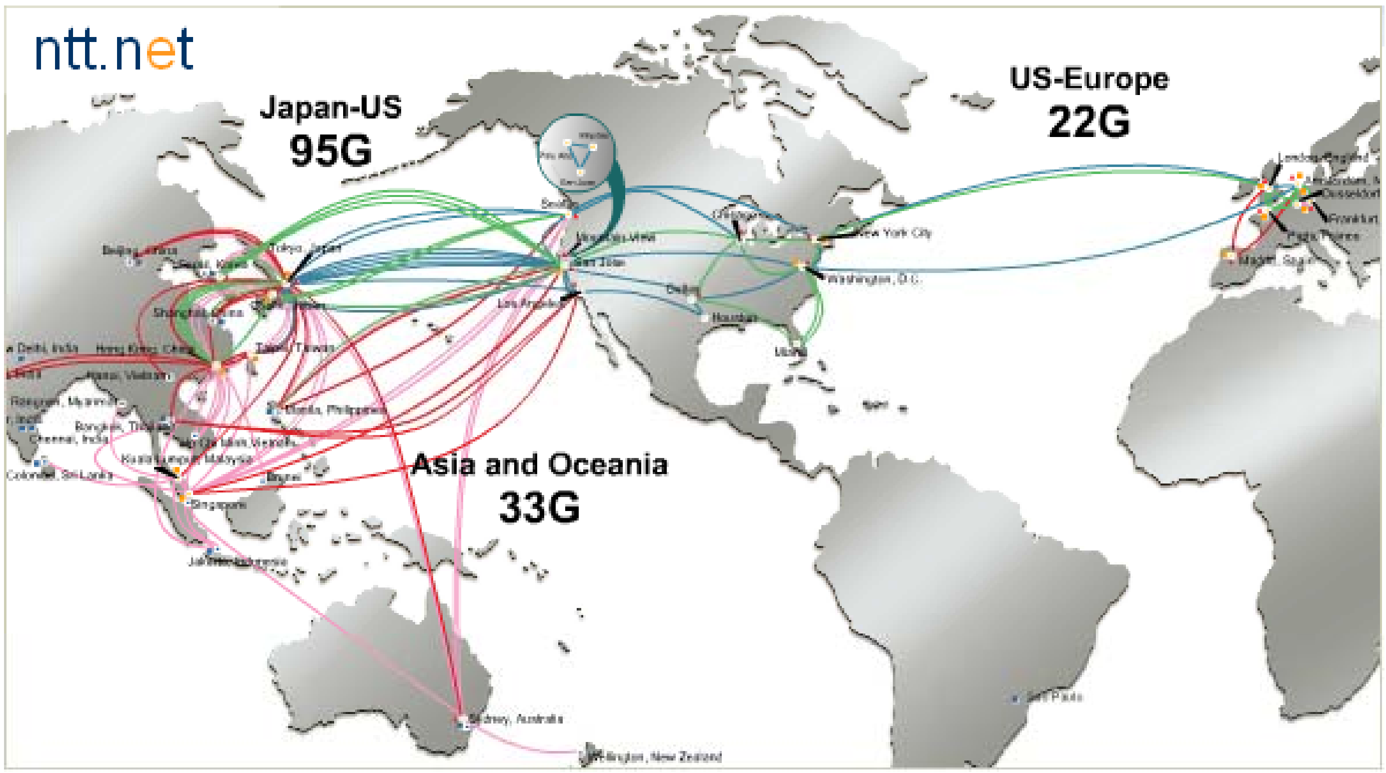
- A network is what you get anytime you connect two or more computers together by some kind of a link.
  - An internet is a "network of networks"



OR



Example piece: NTT backbone



Components of a Network ("Hardware")

- Links carry information (bits)
  - Wire, wireless, fiber optic, smoke signals …
  - May be point-to-point or broadcast
- Switches move bits between links
  - Routers, gateways, bridges, CATV headend, PABXs, …
- Hosts are the communication endpoints
  - PC, PDA, cell phone, tank, toaster, …
  - Hosts have names
- Applications make use of the network at hosts
  - Facebook, iTunes, VoIP phones, cameras, …

- Note much other terminology:
  - channels, nodes, intermediate systems, end systems, etc.

Technical challenges for network design

- Scale (PAN, LAN, MAN, WAN, …)
  - Number of devices and speed
  - Some designs will break at scale
- Heterogeneity
  - Mix of technologies
  - Weakens assumptions that can be made
- Distributed nature
  - Parts of the whole will fail
  - Have to tolerate for reliability
- Decentralized nature
  - Independent actors; no single party in control
  - Have to share resources in a reasonable way

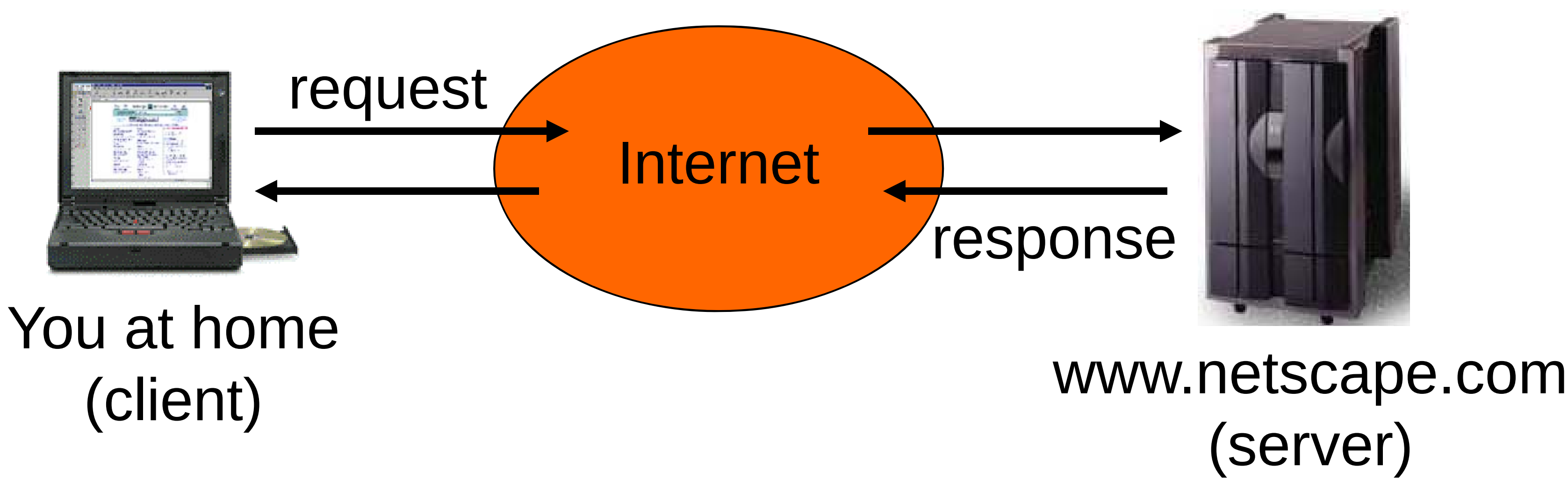
Functionality of Networks ("Software")

- We need structure to handle the complexity. What?

- Key idea: modularity in the form of layered protocols
- Protocols are modules that provide specific functionality
  - But composed in a constrained way
- Higher layers build on (hide) lower layers
  - Provide virtual communication at higher levels

Informal Example: Brief Tour of the Internet

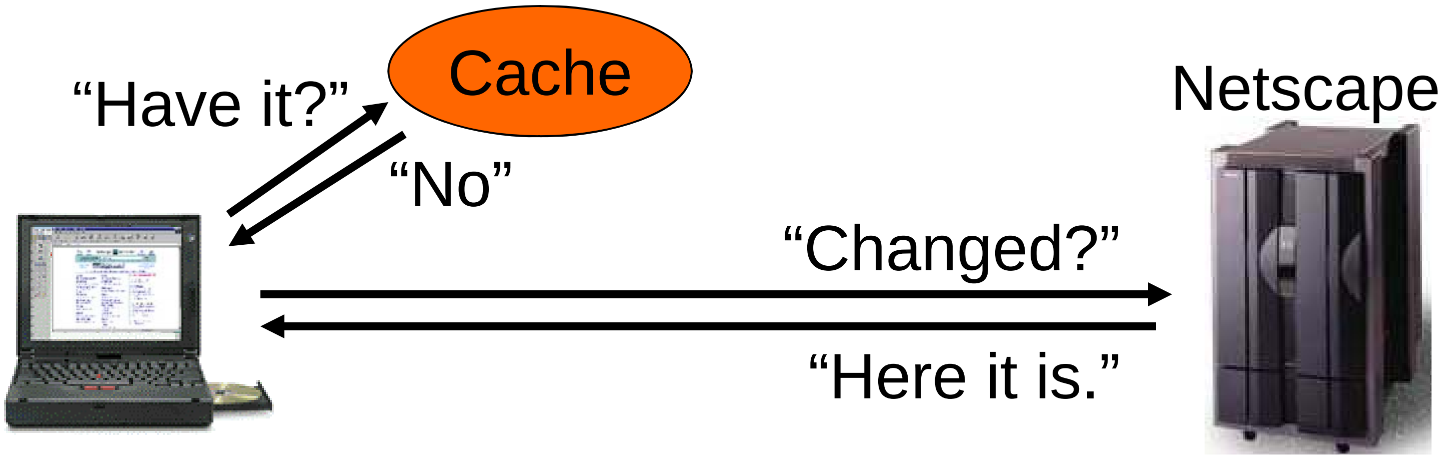
- What happens when you “click” on a web link?



- This is the view from 10,000 ft …

9,000 ft: Scalability

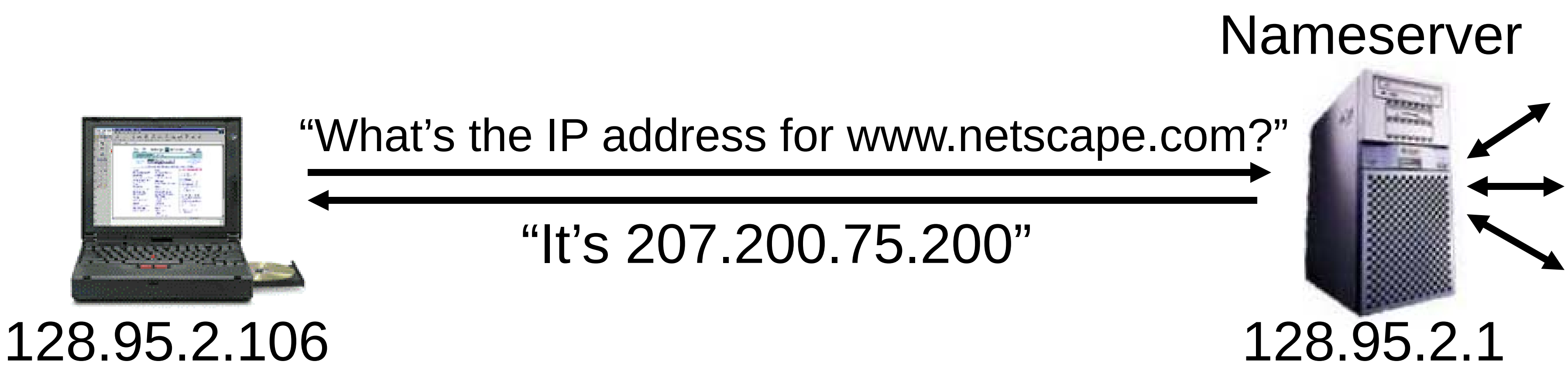
- Caching improves scalability



- We cut down on transfers:
  - Check cache (local or proxy) for a copy
  - Check with server for a new version

8,000 ft: Naming (DNS)

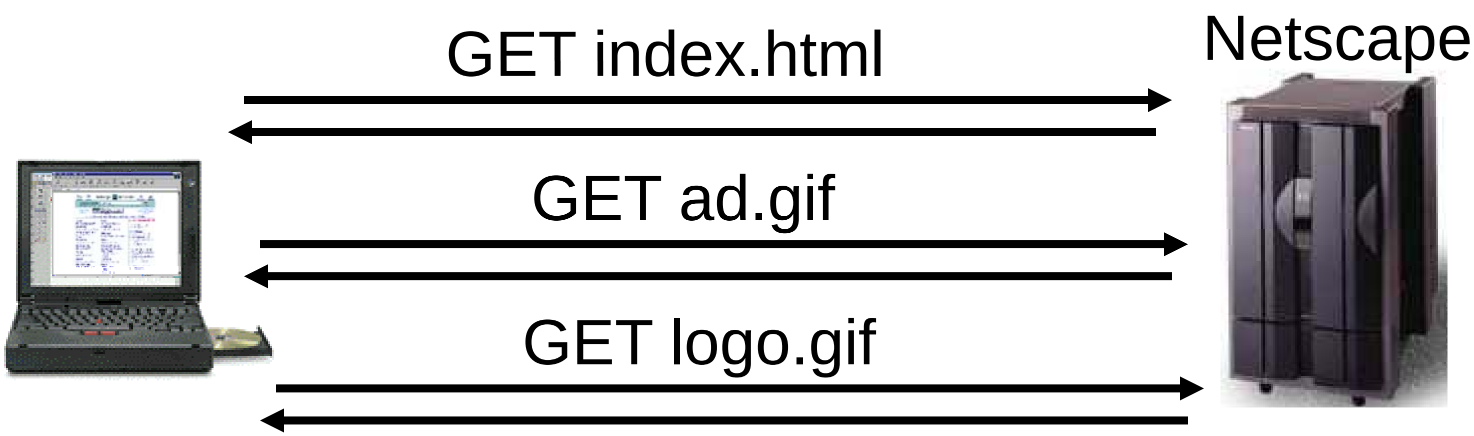
- Map domain names to IP network addresses



- All messages are sent using IP addresses
  - So we have to translate names to addresses first
  - But we cache translations to avoid next time

7,000 ft: Sessions (HTTP)

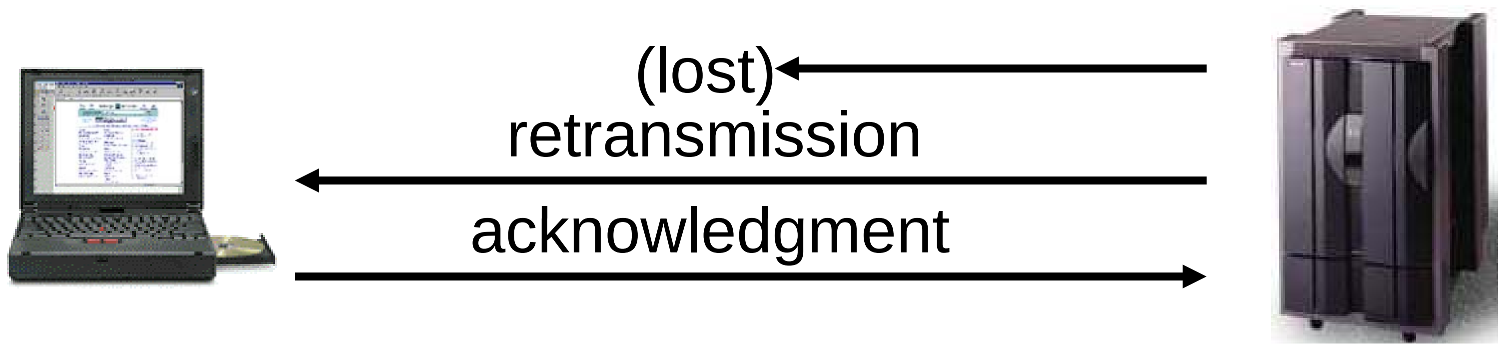
- A single web page can be multiple “objects”



- Fetch each “object”
  - either sequentially or in parallel

6,000 ft: Reliability (TCP)

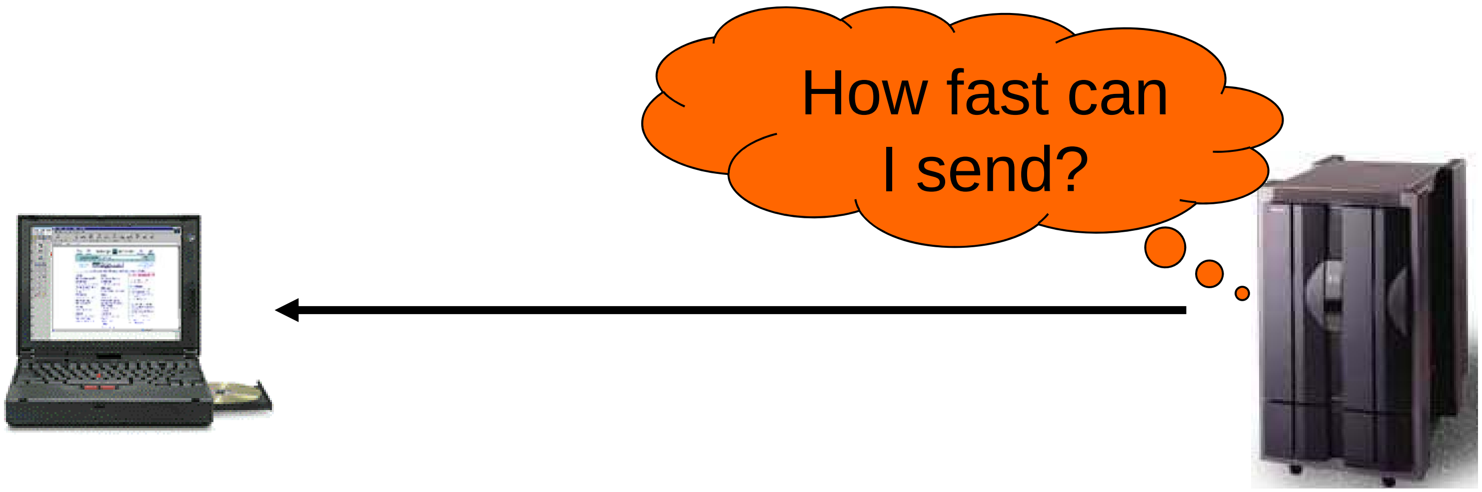
- Messages can get lost



- We acknowledge successful receipt and detect and retransmit lost messages (e.g., timeouts)

5,000 ft: Congestion (TCP)

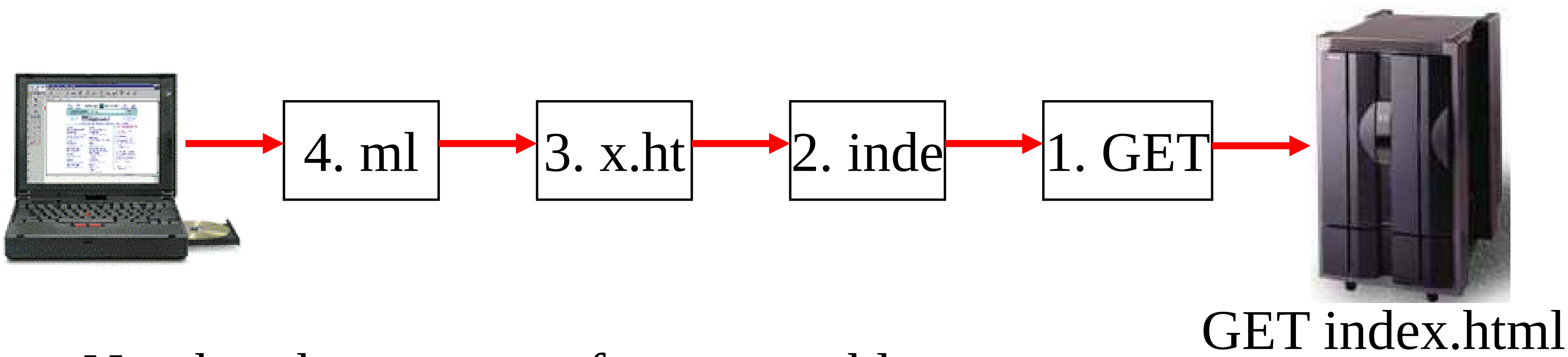
- Need to allocate bandwidth between users



- Senders balance available and required bandwidths by probing network path and observing the response

4,000 ft: Packets (TCP/IP)

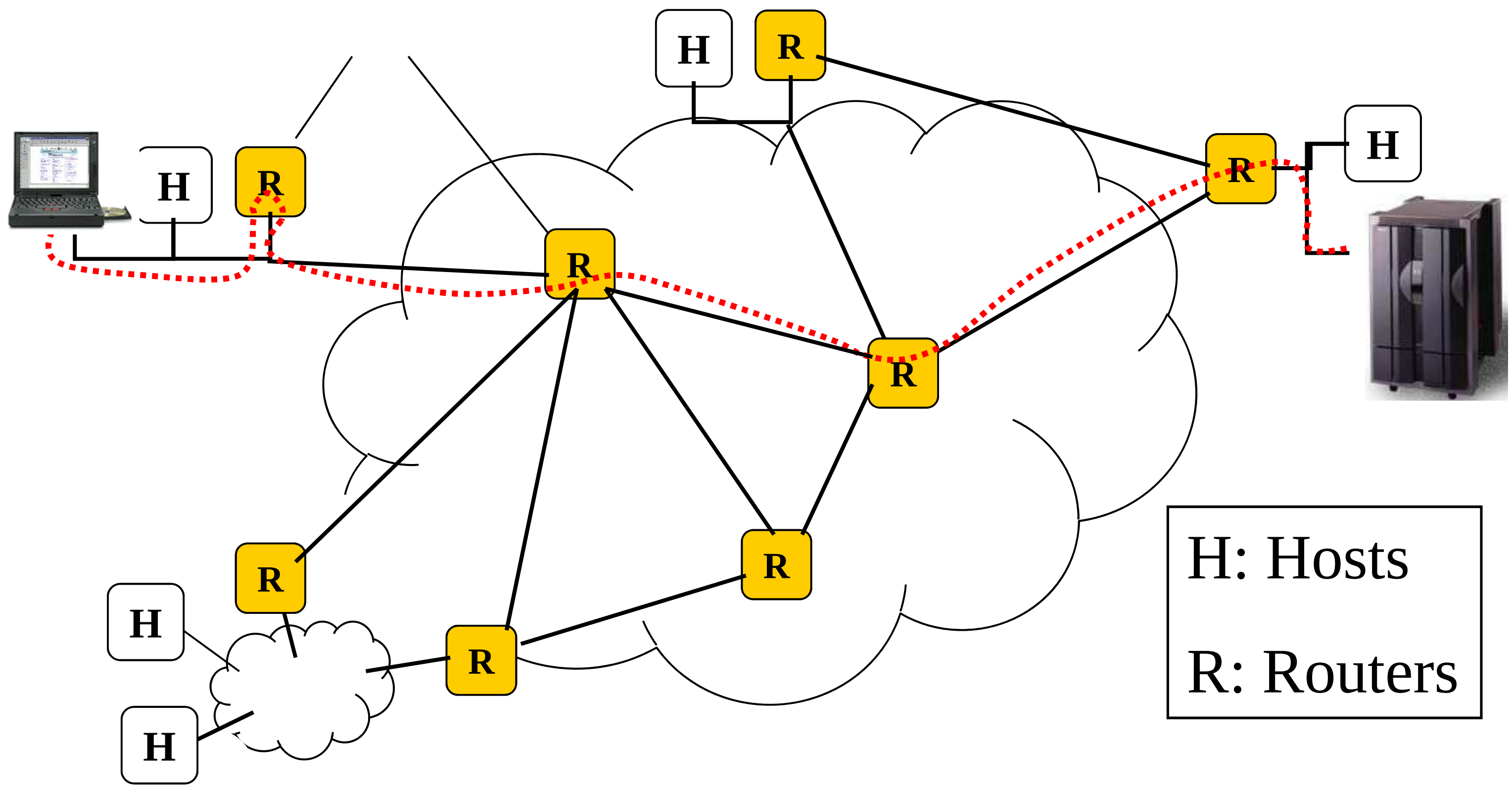
- Long messages are broken into packets
  - Maximum Ethernet packet is 1.5 Kbytes
  - Typical web page is 10 Kbytes



- Number the segments for reassembly

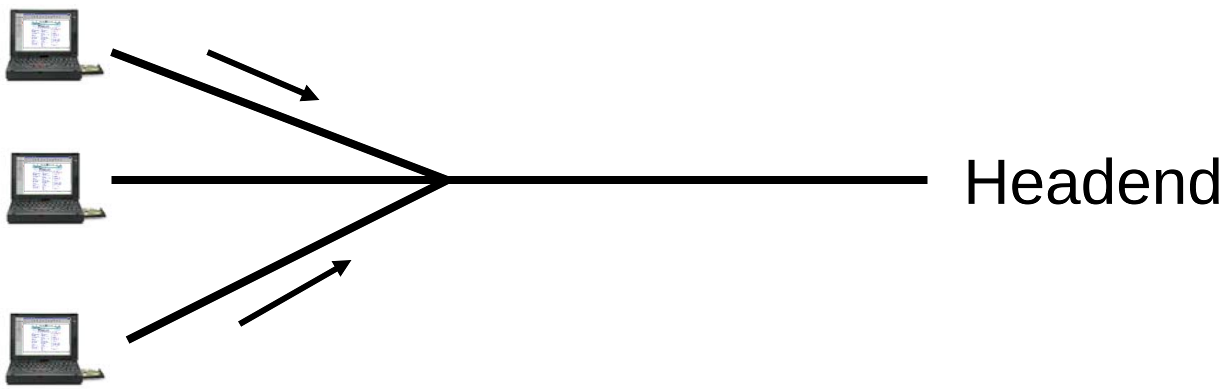
3,000 ft: Routing (IP)

- Packets are directed through many routers



2,000 ft: Multi-access (e.g., Cable)

- May need to share links with other senders



- Poll headend to receive a timeslot to send upstream
  - Headend controls all downstream transmissions
  - A lower level of addressing is used …

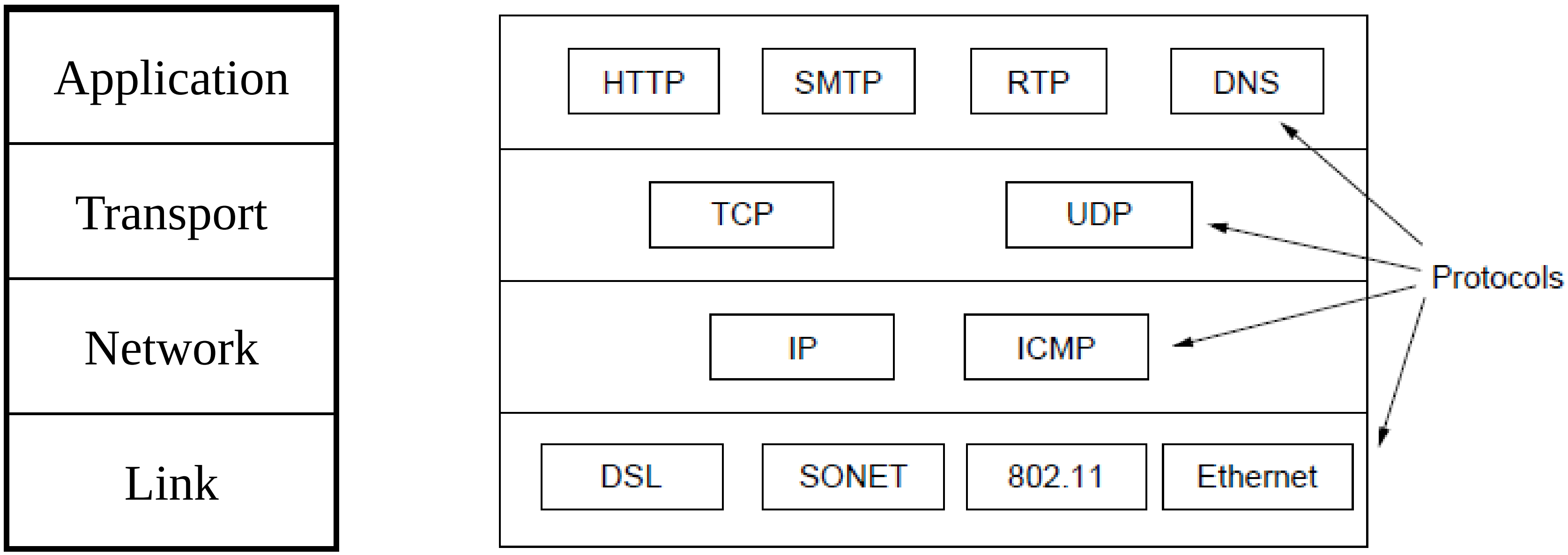
1,000 ft: Framing/Modulation

- Protect, delimit and modulate payload as a signal

Sync / Unique	Header	Payload w/ error correcting code

- E.g, for cable, take payload, add error protection (Reed-Solomon), header and framing, then turn into a signal
  - Modulate data to assigned channel and time (upstream)
  - Downstream, 6 MHz (~30 Mbps), Upstream ~2 MHz (~3 Mbps)

Internet Protocol Framework



Model

Protocols

Larger scope for higher layers

OSI "Seven Layer" Reference Model

- Seven Layers:

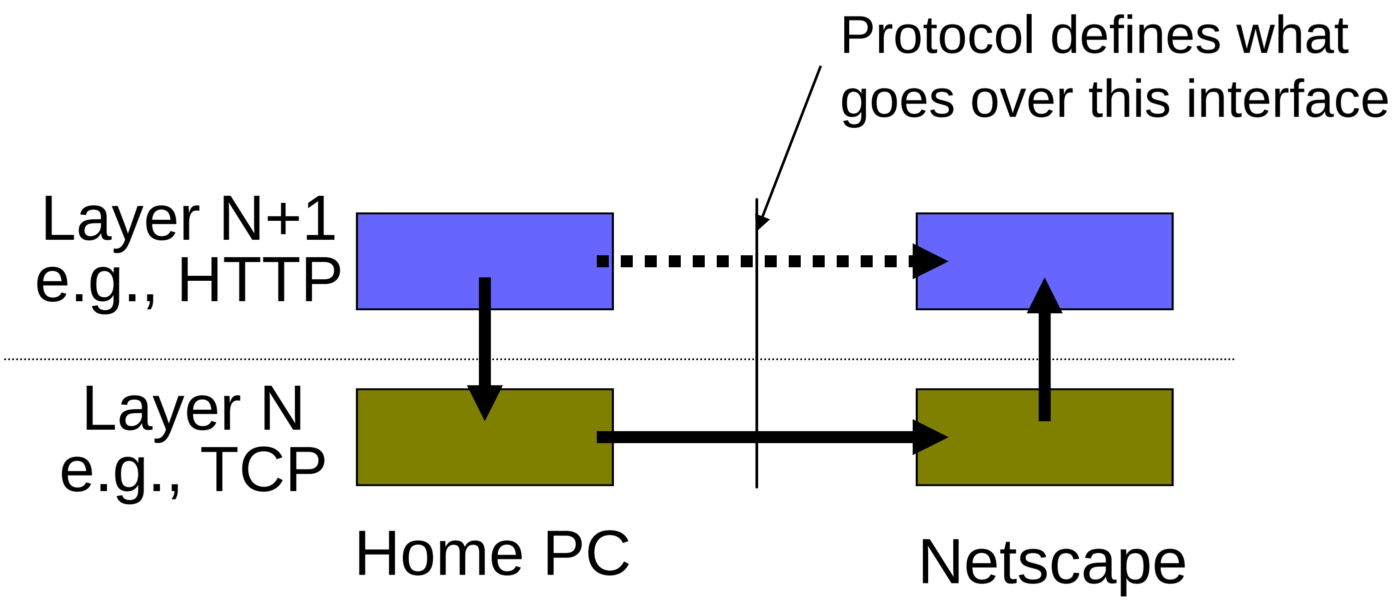
Application
Presentation
Session
Transport
Network
Link
Physical

Their functions:

- Your call
- Encode/decode messages
- Manage connections
- Reliability, congestion control
- Routing
- Framing, multiple access
- Symbol coding, modulation

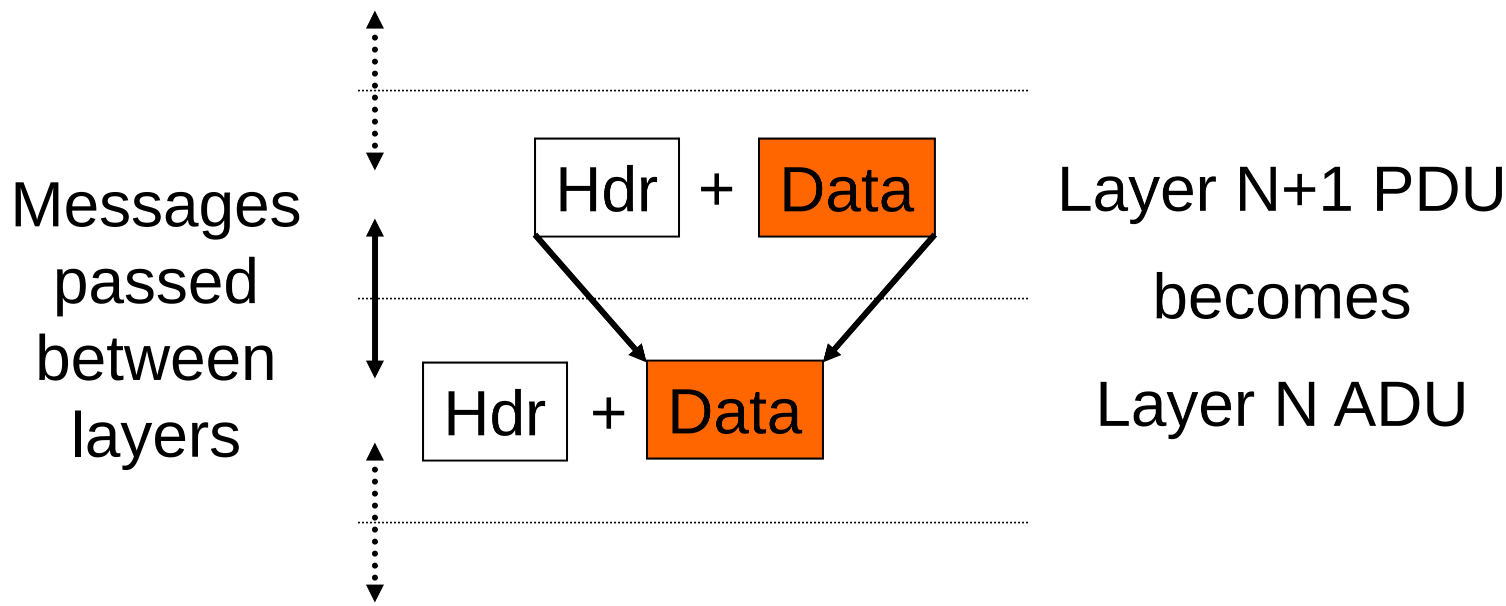
Layering and Protocol Stacks

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



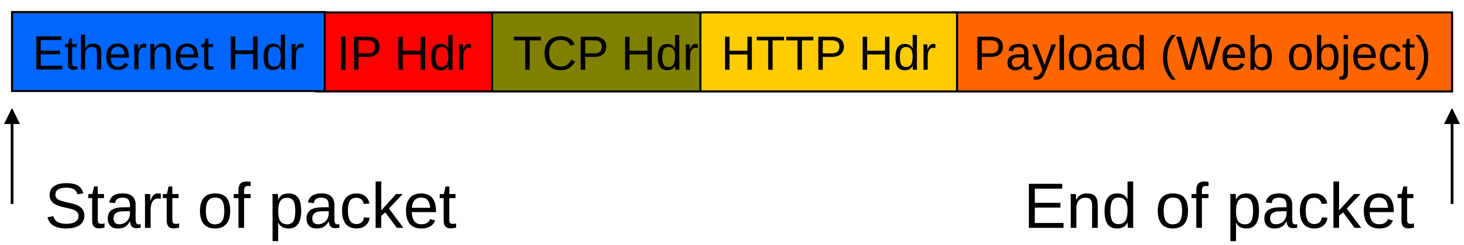
Layering Mechanics

- Encapsulation and decapsulation



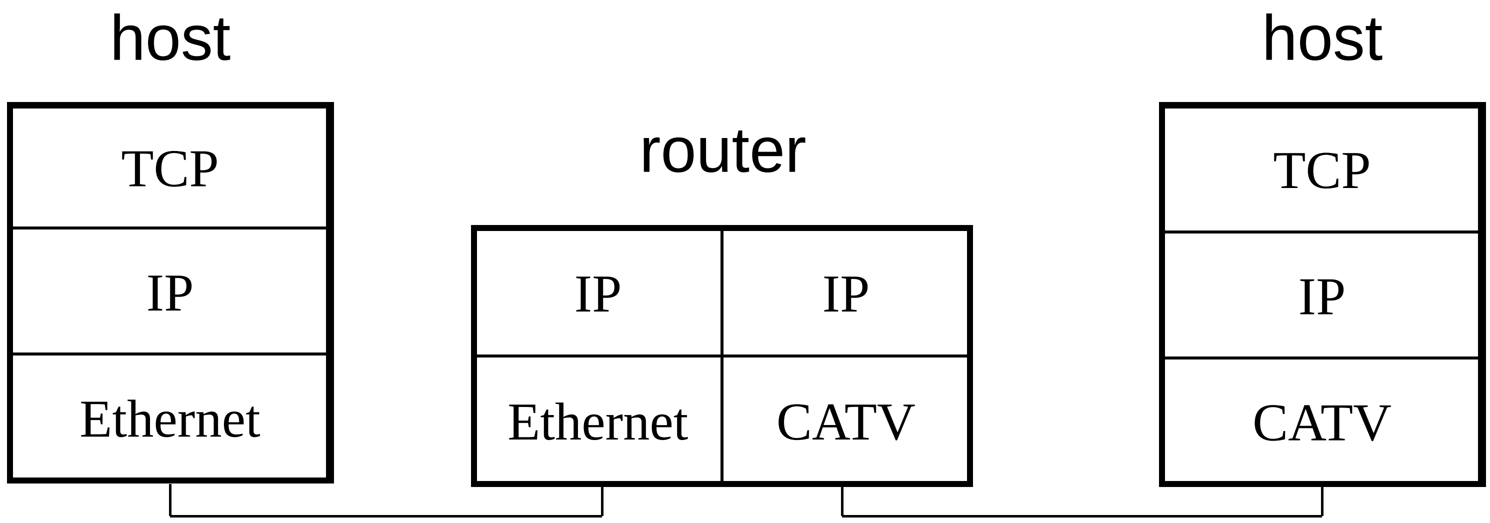
A Packet on the Wire

- Starts looking like an onion!



- This isn't entirely accurate
  – ignores segmentation and reassembly, Ethernet trailers, etc.
- But you can see that layering adds overhead

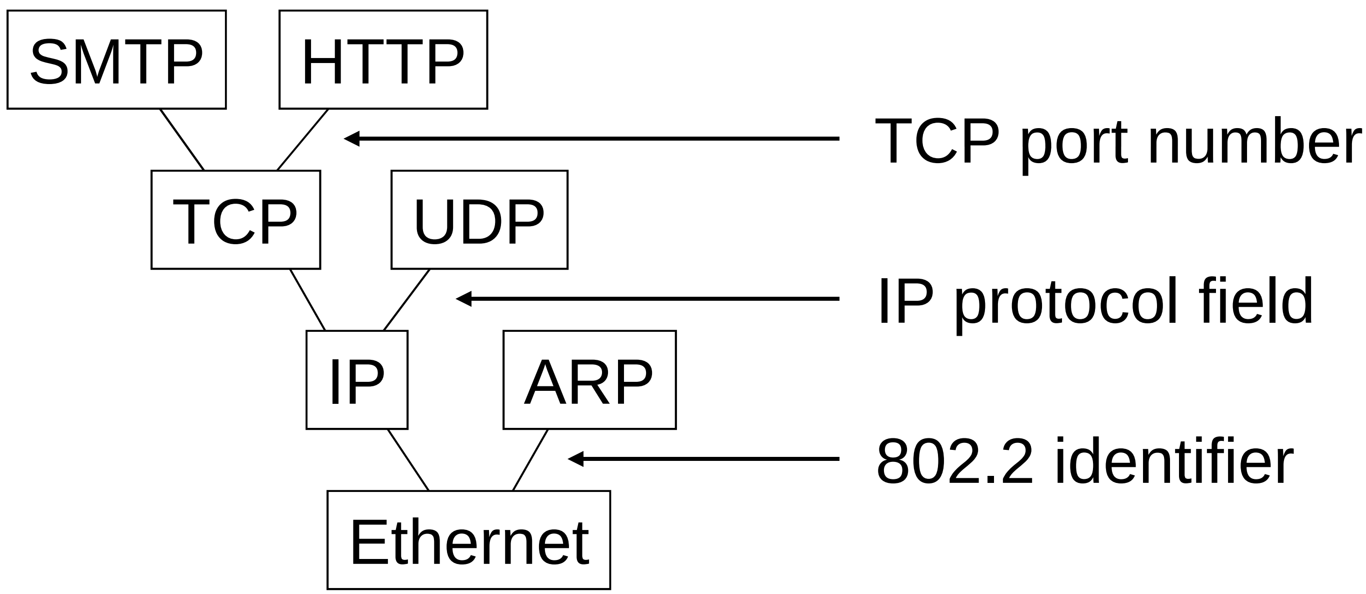
Example – Layering at work



- We can connect different systems because of virtual communication between peers.

More Layering Mechanics

- Multiplexing and demultiplexing in a protocol graph
- Demultiplexing requires a <u>demux key</u>



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, …
- Key concern is interoperability
  - Not how to build a good product. Why?
- Organizations: IETF, IEEE, ITU
  - RFCs, e.g., RFC 2460 is IPv6
  - 802 standards, e.g.,802.11 is WiFi
  - “letter recommendations”, e.g., G.992.5 is ADSL

Questions

- What are the advantages and disadvantages of protocols and layering?
- How do we decide what functions belong in which layers?

Pros and Cons

- Protocols break apart a complex task into simpler and reusable pieces.
- Interoperability promotes markets

- Layers drag down efficiency
- Layers can hide important information (e.g., wireless)

E2E argument

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.

E2E example: reliable file transfer

- We need reliability mechanisms for two purposes.
- 1. Correctness. Must be at the ends (app, host)
- 2. Performance. Can be in the middle (routers, links)

- In practice:
  - Links: lower residual error rate, e.g., CRCs, 802.11 ARQ
  - Routers: don't do much; don't expect many errors
  - Host: key for correctness, e.g., TCP checksum, retransmit
  - Apps: don't do much, at least for short transfers

E2E versus software engineering

- There is an apparent tension
  - E2E pushes the implementation of a function to higher layers
  - Logically this is the application itself, e.g., file transfer app.
  - Code reuse benefits from implementation below the app
- But there is not much tension in practice
  - Can still obtain code reuse, e.g., with libraries
  - Even easier are checks near the end (on hosts, but not part of app.) that may be “good enough”, e.g., applications use TCP