CSE 561 – Introduction

David Wetherall
djw@cs.washington.edu

561 Syllabus and Key Concepts by Theme

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

561 Syllabus and Key Concepts by Theme

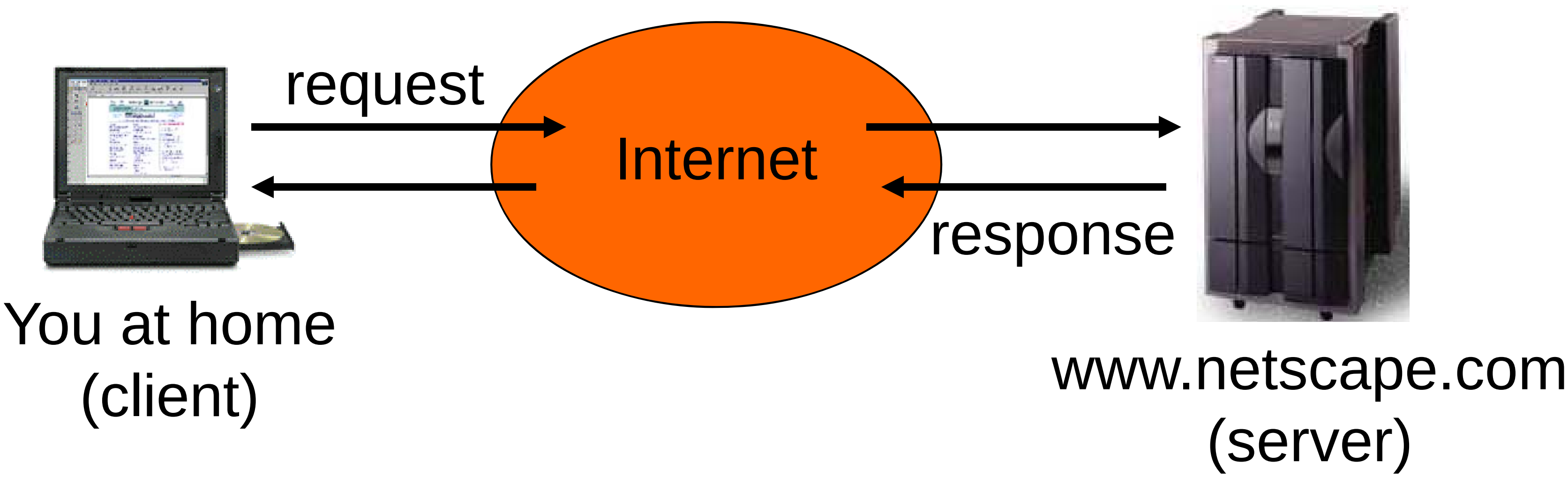
- **Reliability – reliable distributed services from unreliable parts**
- *Soft-state. Fate-sharing. Error detection codes (checksums, CRCs). Acknowledgements and retransmissions (ARQ). Sliding window. Error correcting codes or FEC. TCP's three-way handshake. Link-state and distance vector routing*
- **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 are Wonderful!

- We use them to communicate with people remotely, to find and share songs, movies, Web pages/blogs, information, play games, and more.
- We use them at work, at home, in social networks.
- …
- The Internet has expanded to pretty much everything (laptops, and toasters, tanks, lightbulbs)
- The fixed and mobile phone networks are in flux
- Many new technologies bring networking to new domains (RFID, powerline, 60GHz, underwater, satellite)

A Brief Tour of the Internet

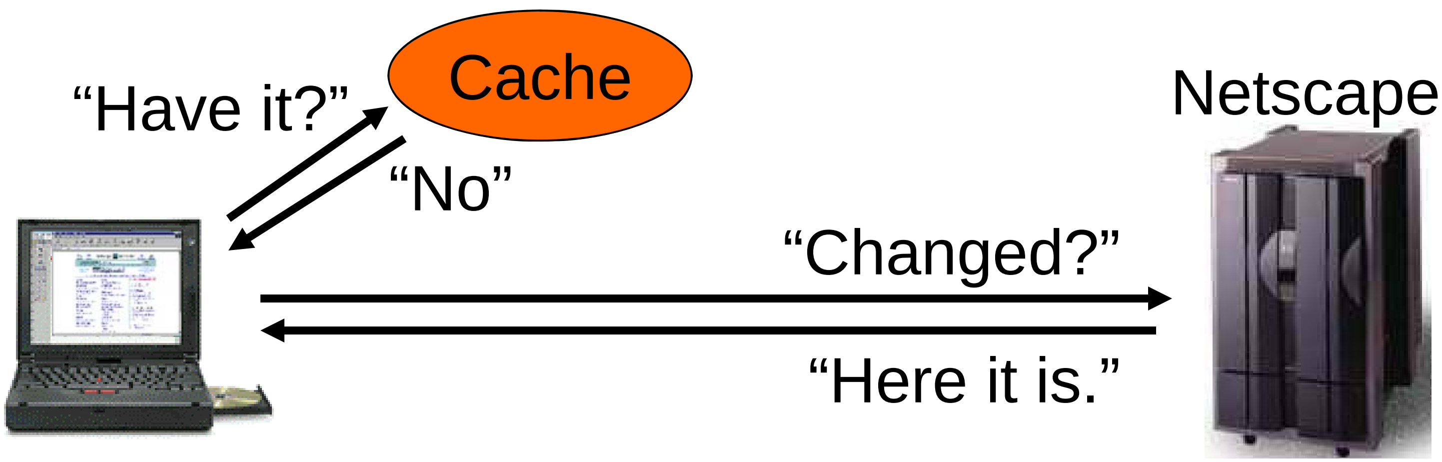
- What happens when you “click” on a web link?



- This is the view from 10,000 ft …

9,000 ft: Scalability

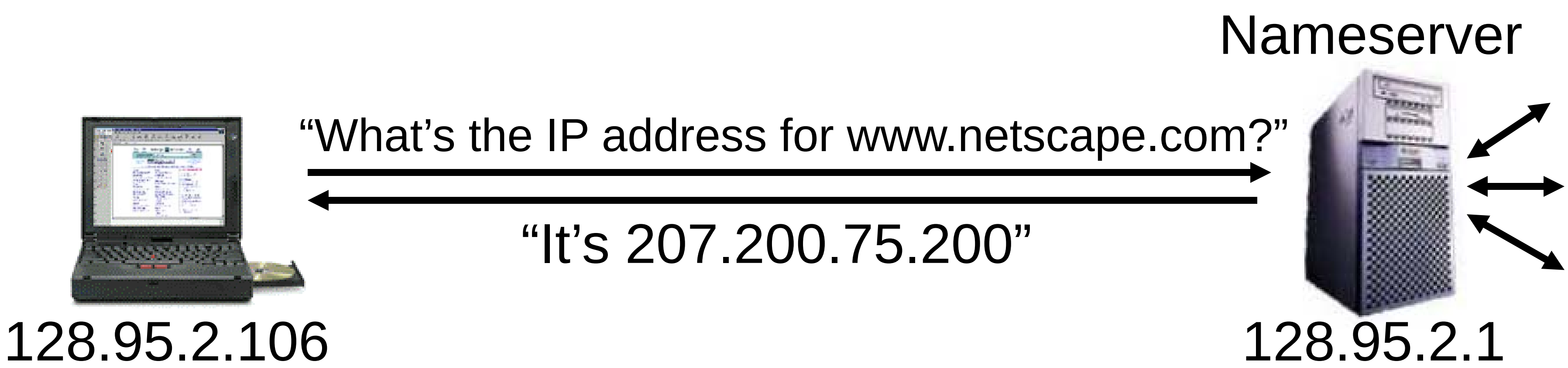
- Caching improves scalability



- 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)

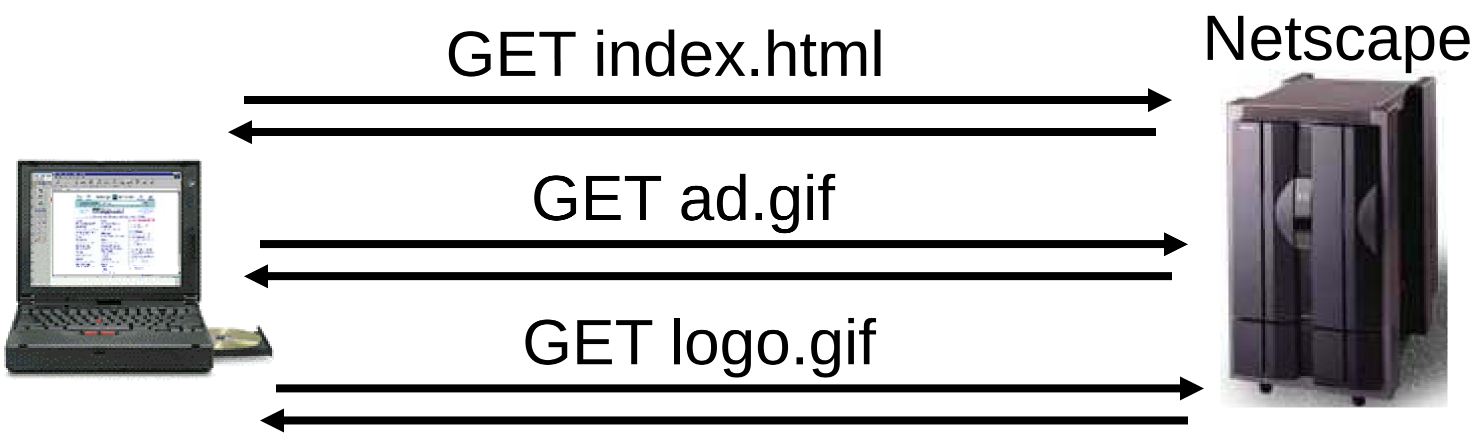
- Map domain names to IP network addresses



- 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)

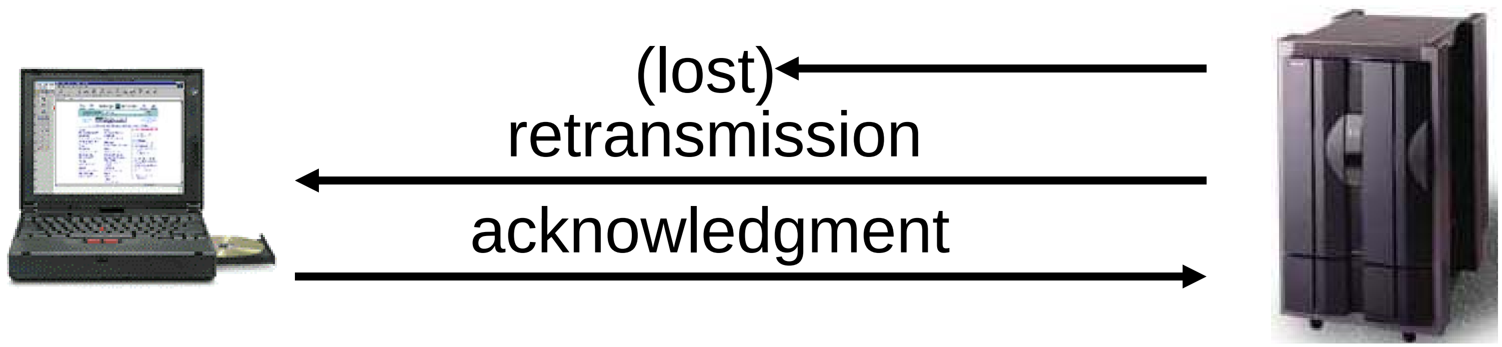
- A single web page can be multiple “objects”



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

6,000 ft: Reliability (TCP)

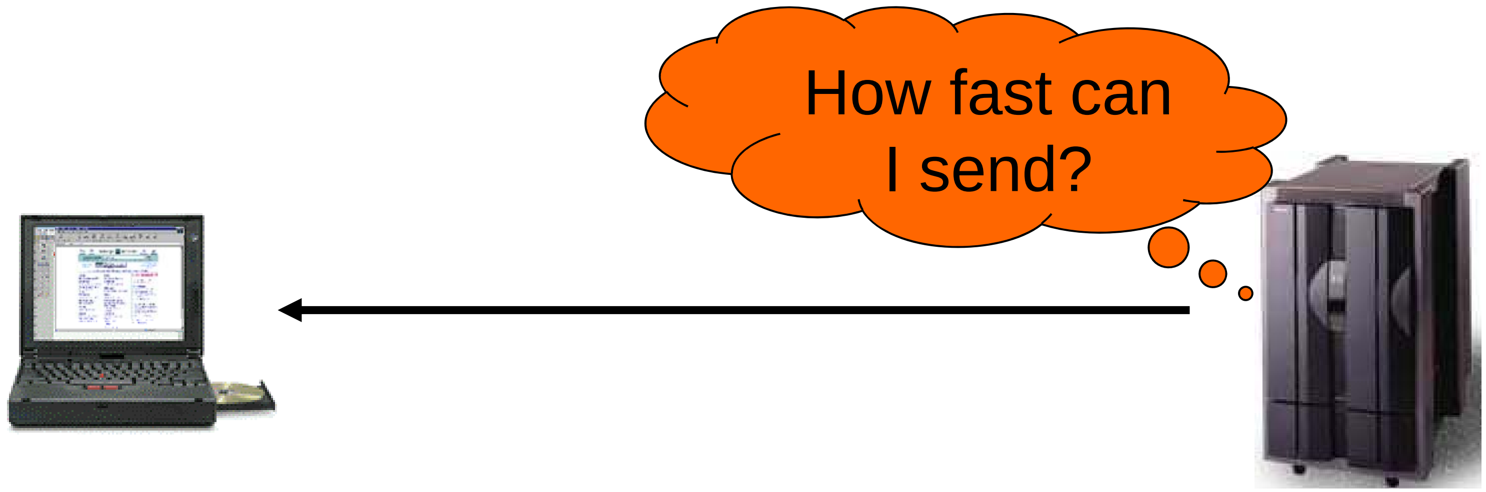
- Messages can get lost



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

5,000 ft: Congestion (TCP)

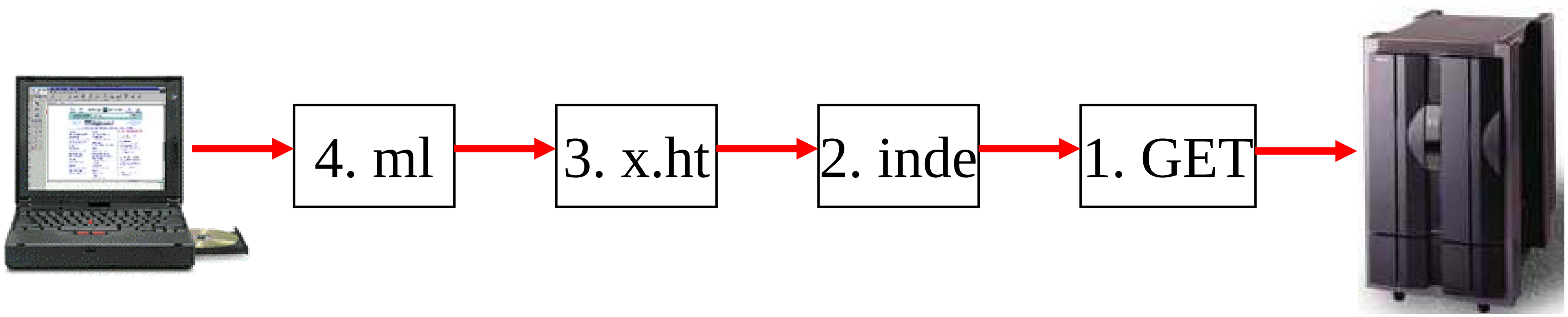
- Need to allocate bandwidth between users



- 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

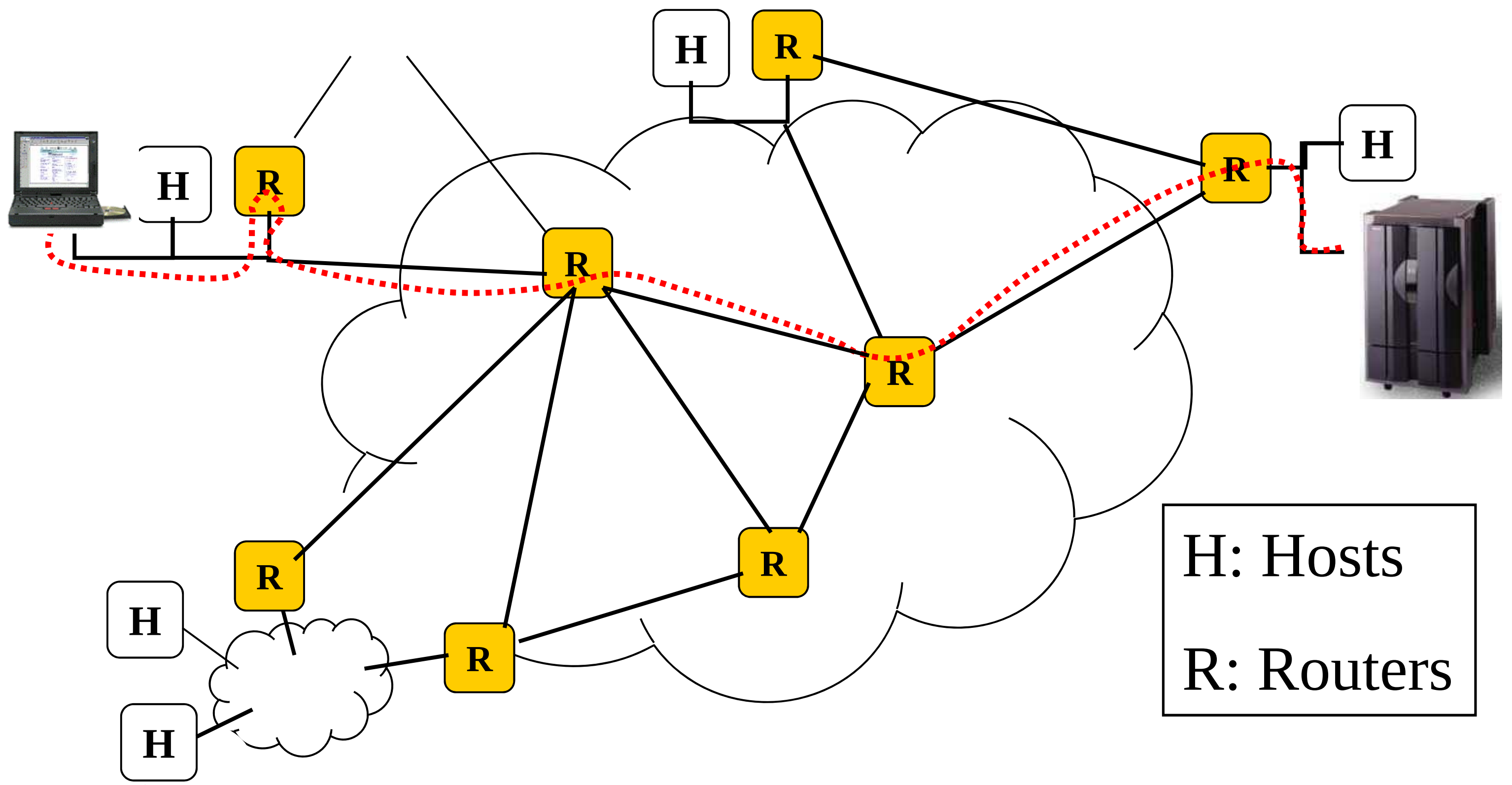


GET index.html

- Number the segments for reassembly

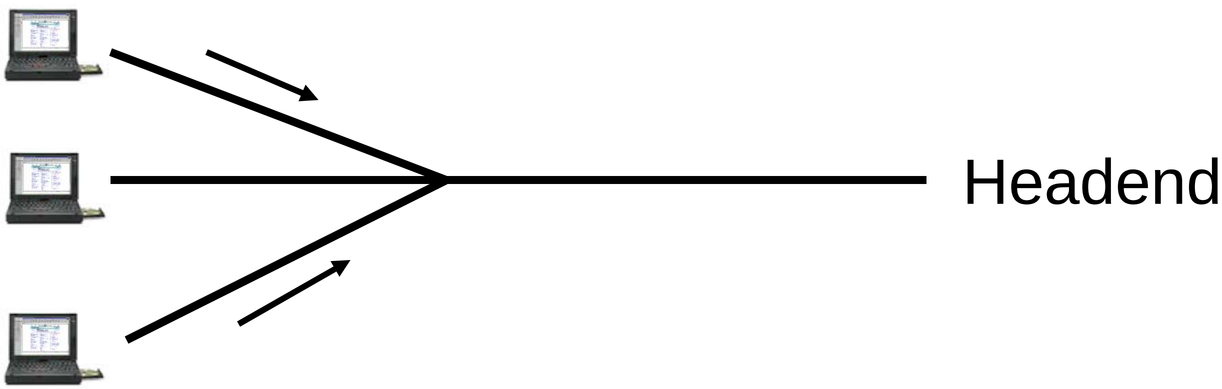
3,000 ft: Routing (IP)

- 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)

2. Protocols and Layering

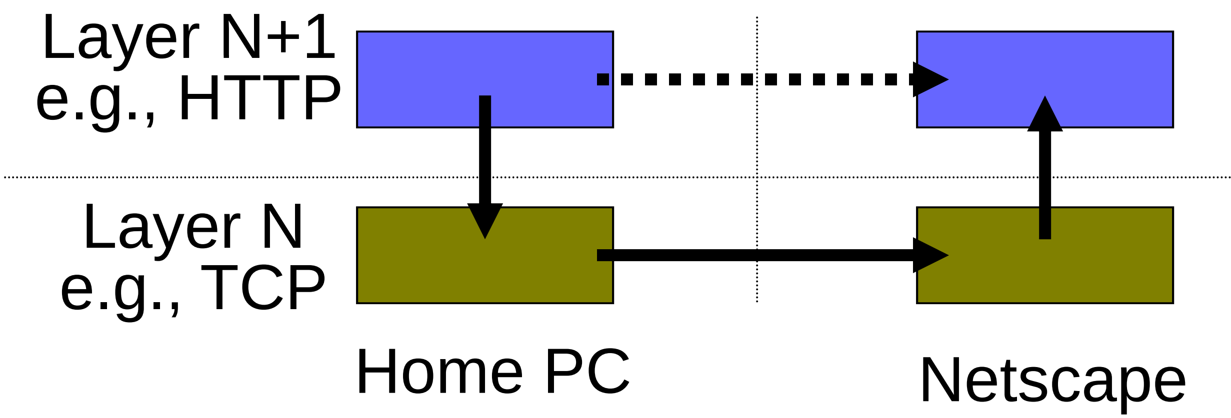
- We need abstractions to handle complexity
- A protocol is an agreement dictating the form and function of data exchanged between parties to effect communication
- Two parts:
  - Syntax: where the bits go
  - Semantics: what they mean, what to do with them
- Examples:
  - IP, the Internet protocol
  - TCP and HTTP, for the Web

Protocol Standards

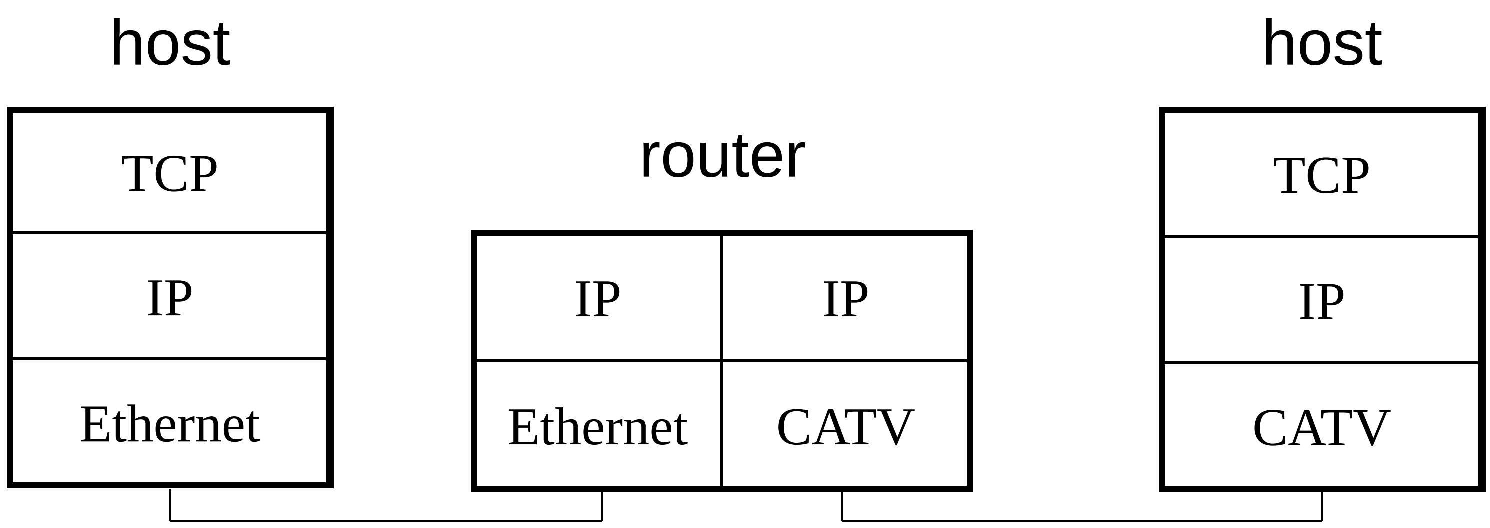
- Different functions require different protocols
- Thus there are many protocol standards
  - E.g., IP, TCP, UDP, HTTP, DNS, FTP, SMTP, NNTP, ARP, Ethernet/802.3, 802.11, RIP, OPSF, 802.1D, NFS, ICMP, IGMP, DVMRP, IPSEC, PIM-SM, BGP, …
- Organizations: IETF, IEEE, ITU
- Key driver is interoperability
- 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



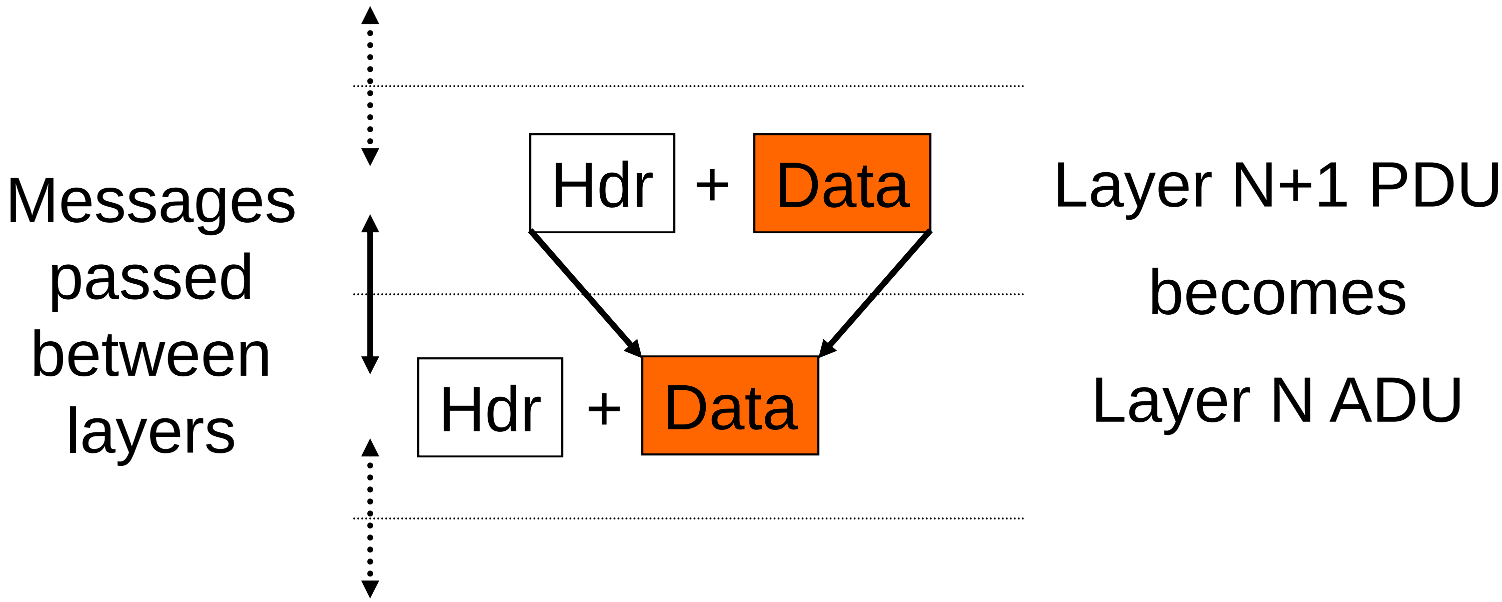
Example – Layering at work



- We can connect different systems

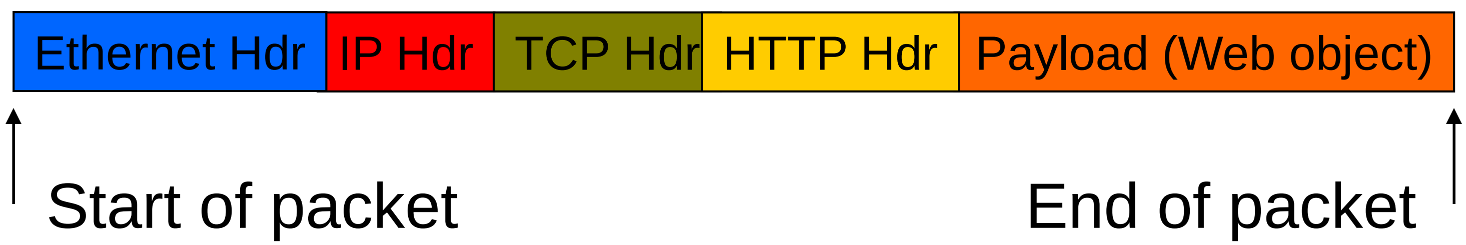
Layering Mechanics

- Encapsulation and decapsulation



A Packet on the Wire

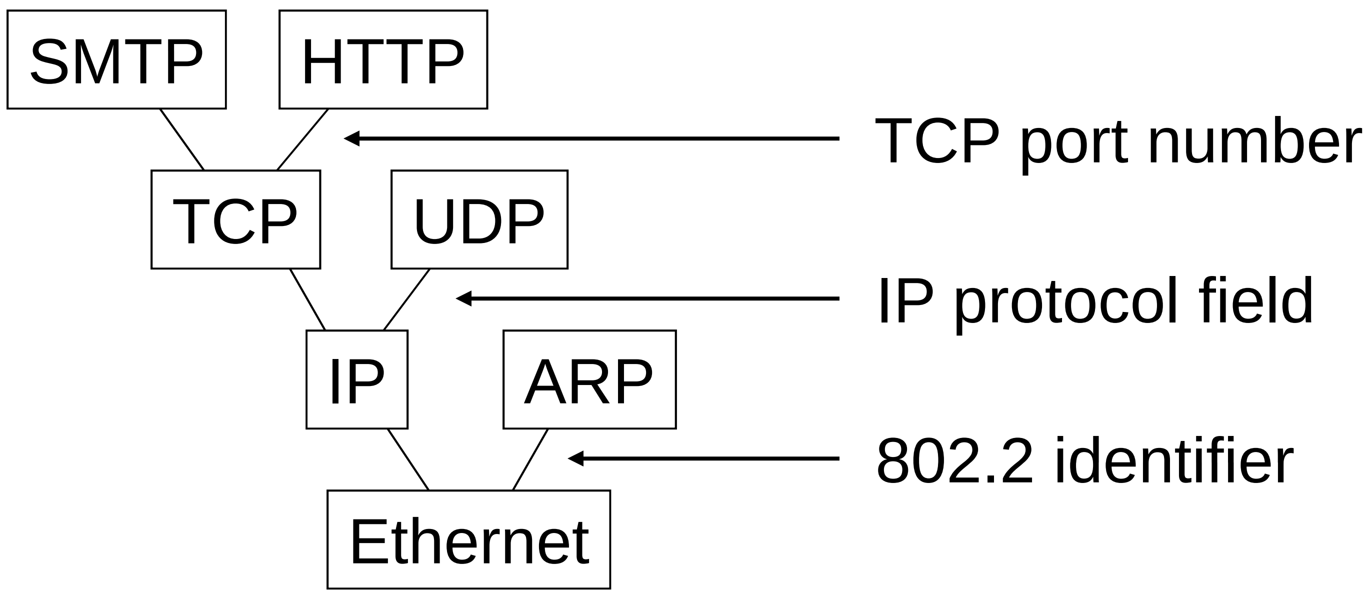
- Starts looking like an onion!



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

More Layering Mechanics

- Multiplexing and demultiplexing in a protocol graph



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 (wireless)

Internet Protocol Framework

Model	Protocols
Application	Many (HTTP,SMTP)
Transport	TCP / UDP
Network	IP
Link	Many (Ethernet, …)

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

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.