CSE 461: Protocols and Layering

Some Administravia

1. Office hours for both instructors and TAs finalized & on web site
2. Homework will be given out Wednesday and due the following week
3. Project 1 is available now and will be due October 17

This Lecture

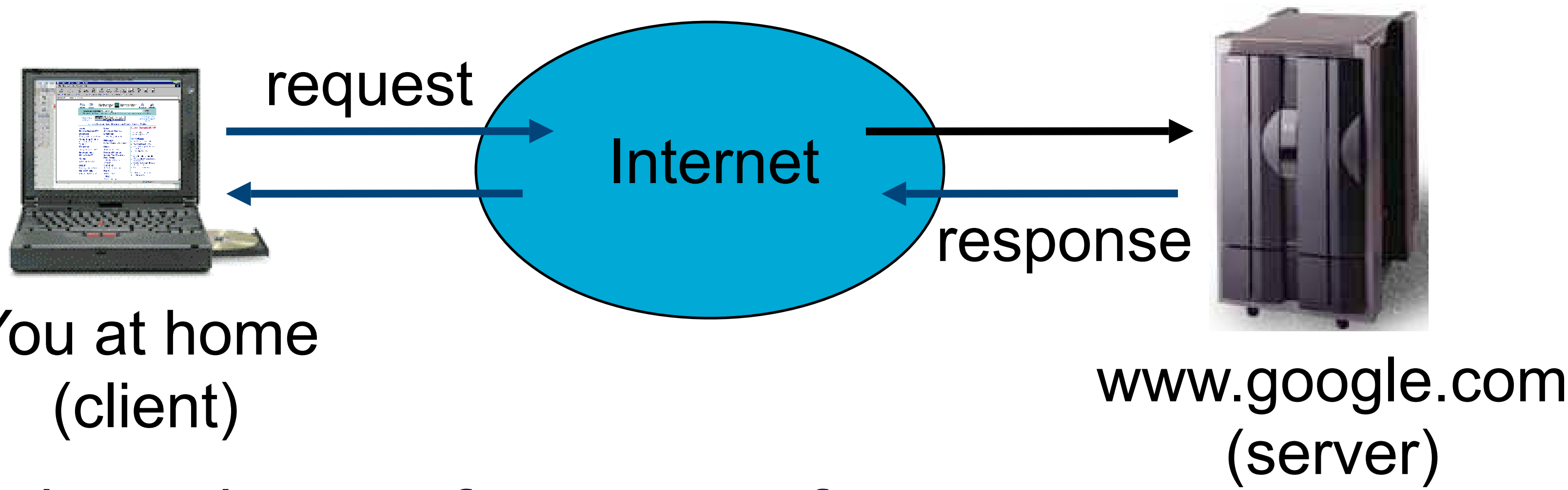
1. The entire course in 10 slides
   a/k/a, a top-down look at the Internet

2. Protocols, layering and standards

3. The end-to-end principle

1. A Brief Tour of the Internet

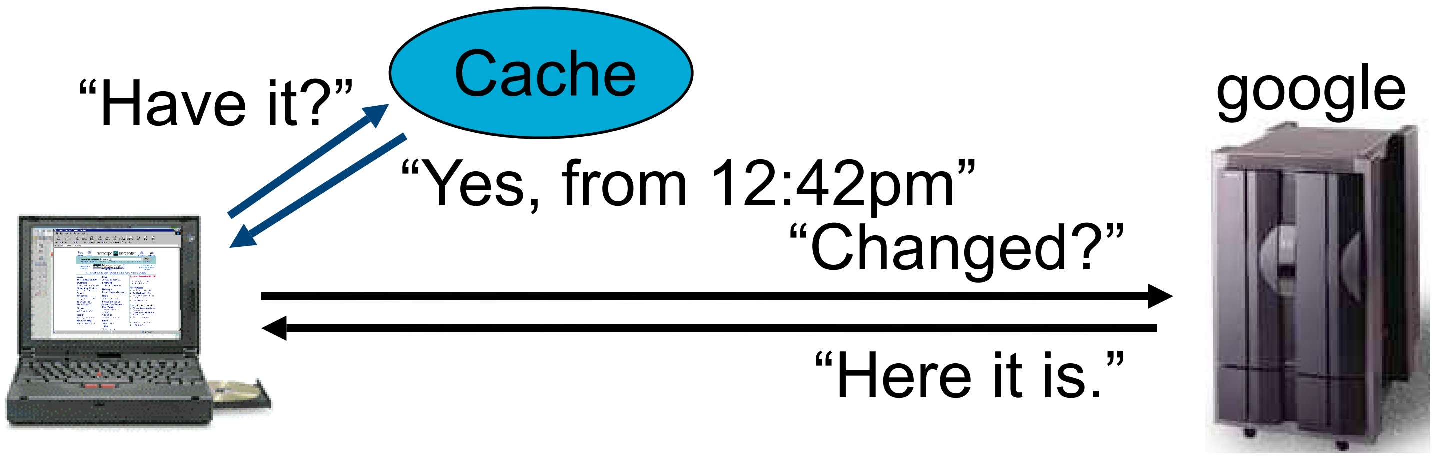
- What happens when you "click" on a web link?



- This is the view from 10,000 ft …

9,000 ft: Caching

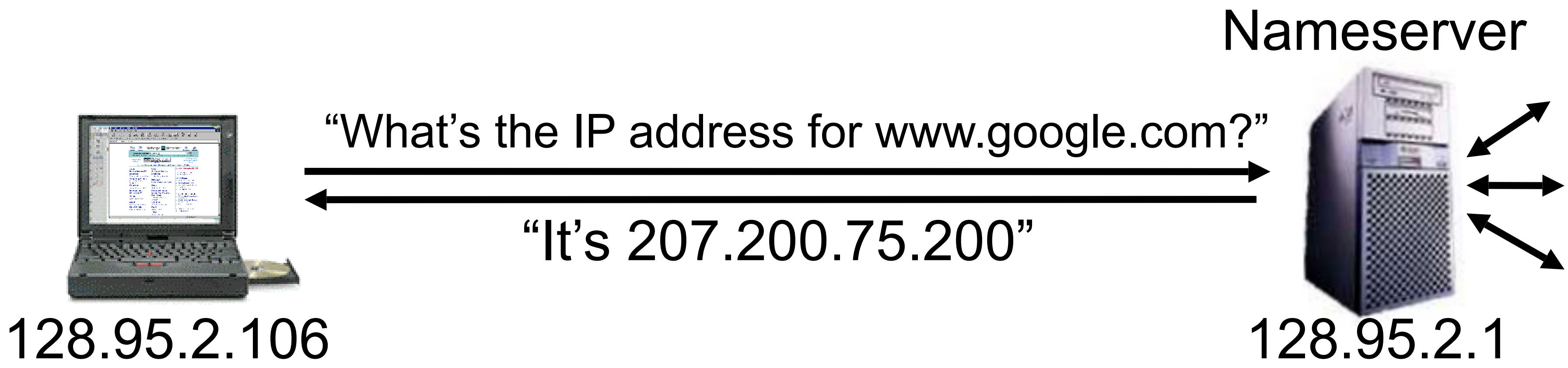
- Lookup a cache before making the full request



- Check cache (local or proxy) for a copy
- Check with server for a new version
- Question: what does caching improve?

8,000 ft: Naming (DNS)

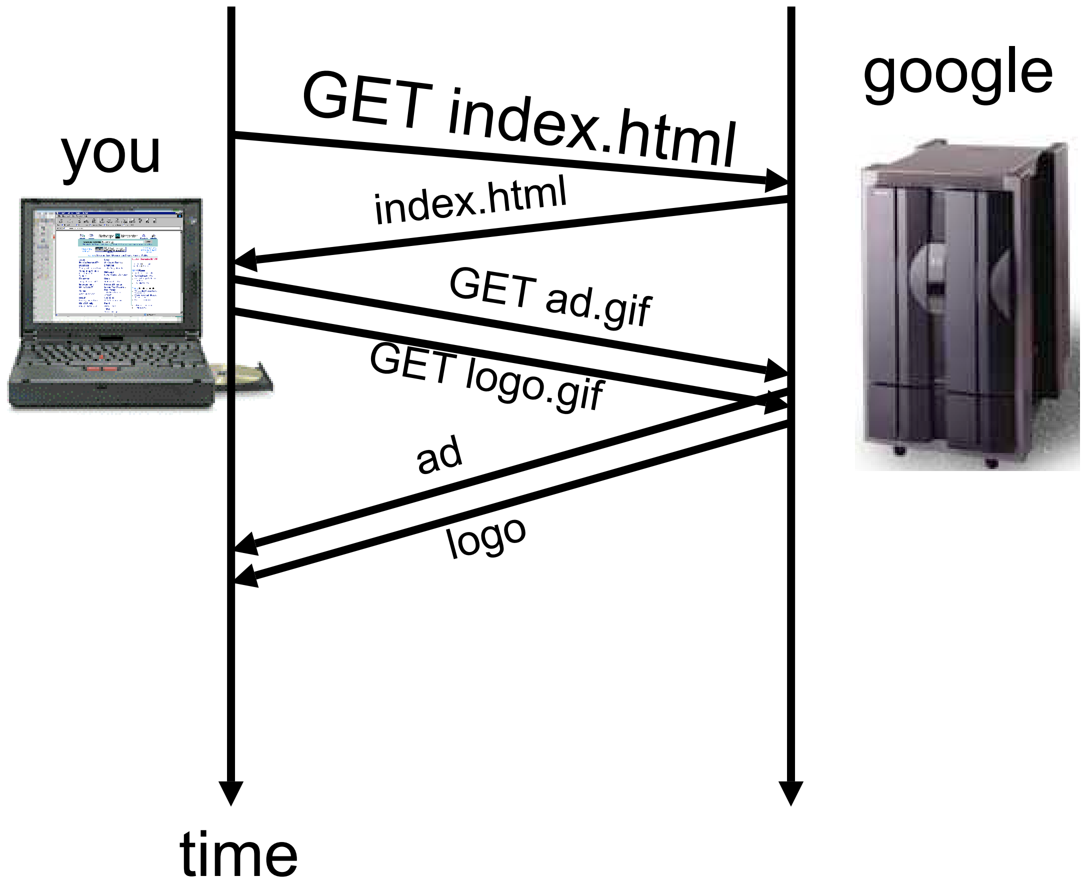
- Map domain names to IP network addresses



- All messages are sent using IP addresses
  - So we have to translate names to addresses first
  - But we cache translations to avoid doing it next time (how do we check for consistency?)

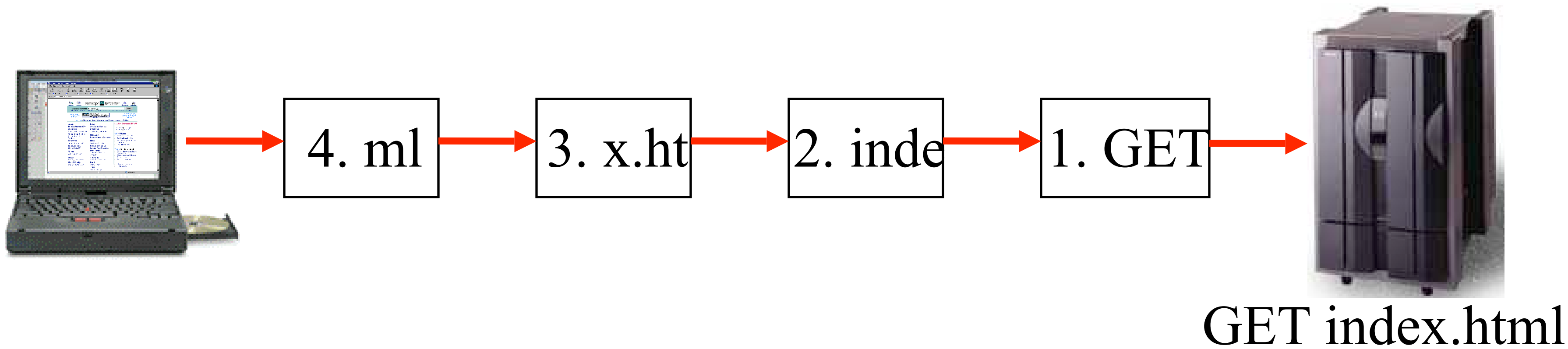
7,000 ft: Sessions (HTTP)

- A single web page can be multiple "objects"
- Fetch each "object" either sequentially or in parallel
- Parallel requests often called "pipelining"



6,000 ft: Packets (TCP)

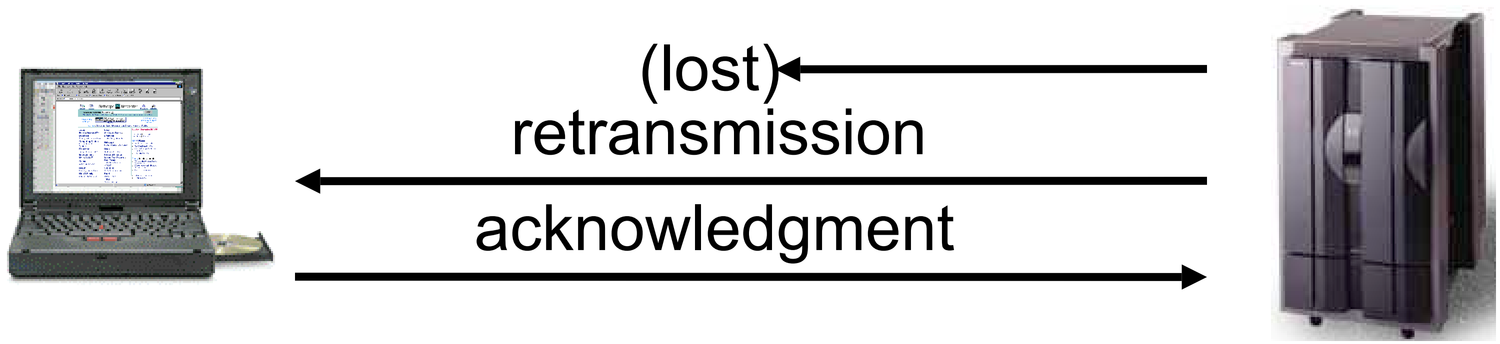
- Long messages are broken into packets
  - Maximum Ethernet packet is 1.5 Kbytes
  - Typical web page is 10 Kbytes



- Number the segments for reassembly and loss detection

5,000 ft: Reliability (TCP)

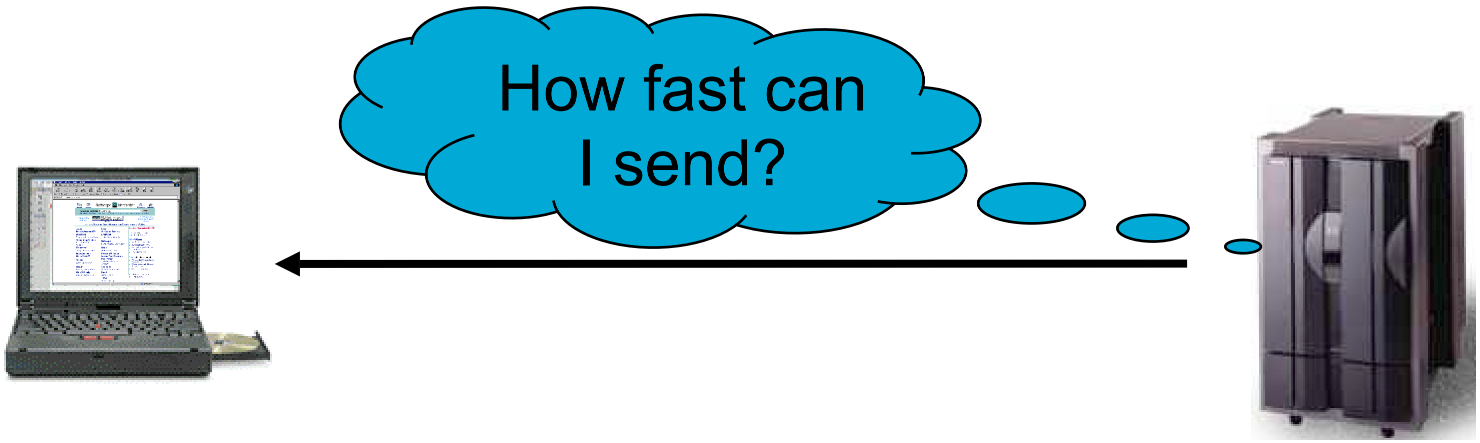
- Packets can (and do) get lost



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

4,000 ft: Congestion (TCP)

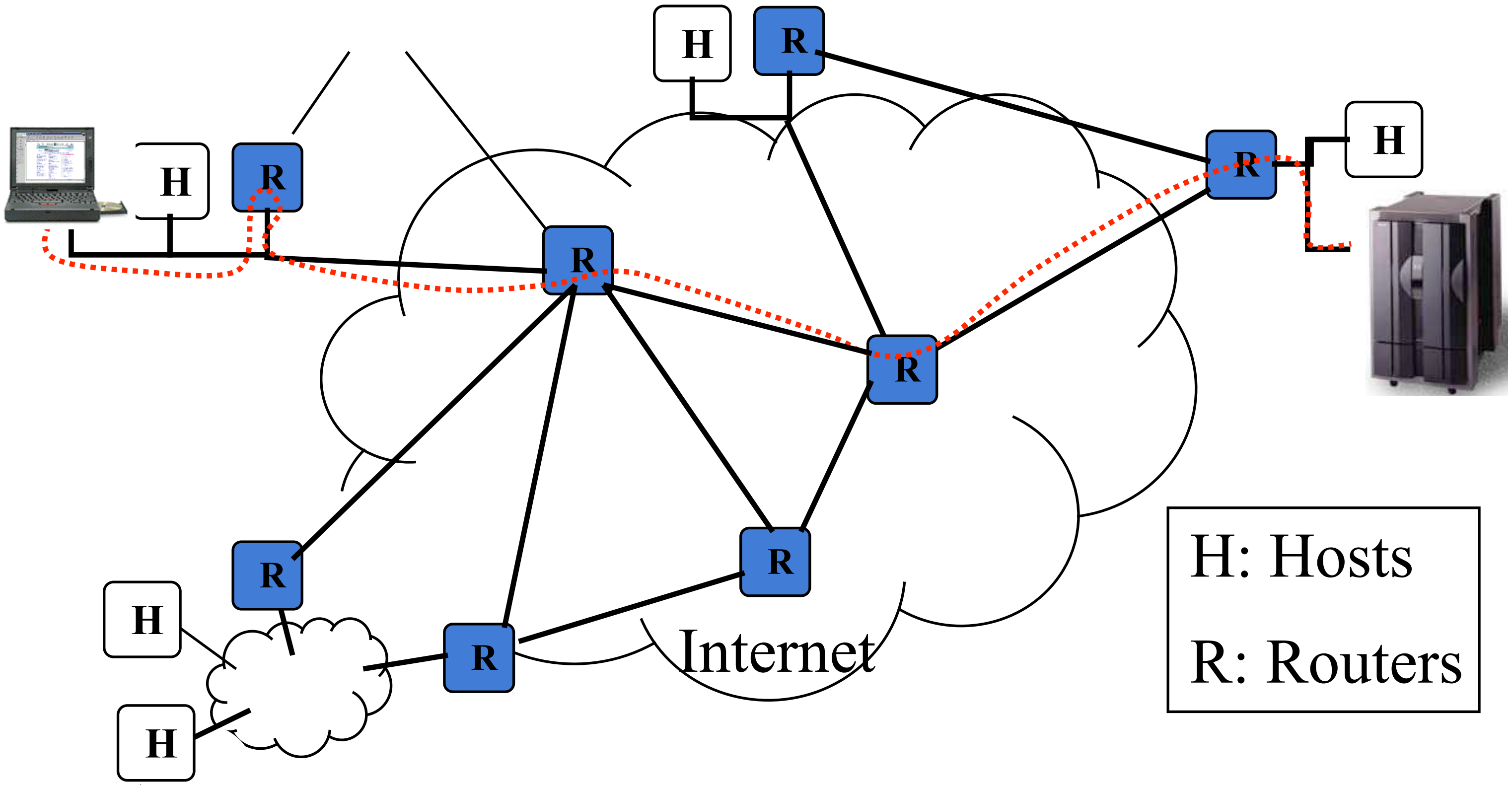
- Need to “allocate” bandwidth between users
  - The magic of statistical multiplexing
  - **Statistical Multiplexing**: key concept in networking
  - Queuing: alien concept in circuit switched networks



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

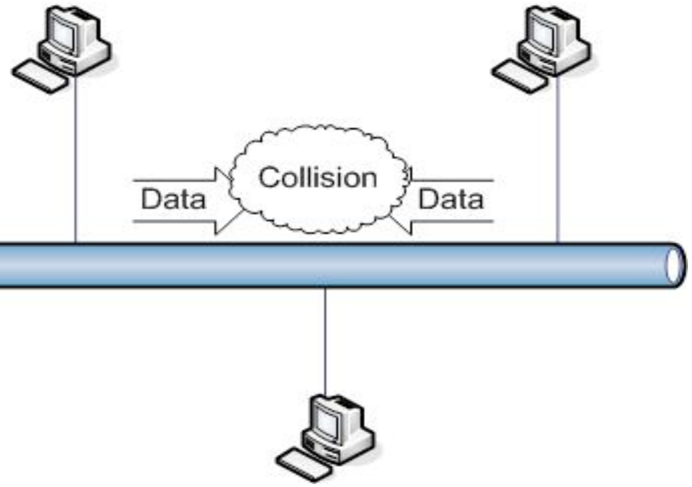
3,000 ft: Routing (IP)

- Packets are directed through many routers
- "IP addresses" tell each packet its destination
- The maze is traversed using protocols like BGP



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

- May need to share links with other senders

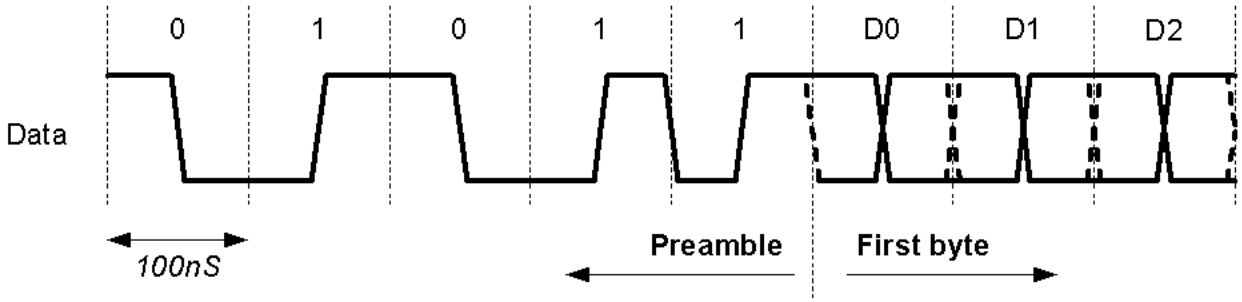


- Send Ethernet “frame”. Collisions can occur if more than one node sends at once (back in “The Day” when Ethernet was a bus)
  - Why is minimum allowed packet length determined by max allowed cable length and transmit speed?
- Ethernet “addresses” (really, identifiers) vs. IP addresses

1,000 ft: Framing/Modulation

- Protect, delimit and modulate payload as 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

2. Protocols and Layering

- We need abstractions to handle all this system complexity

A protocol is an agreement dictating the form and function of data exchanged between parties to effect communication

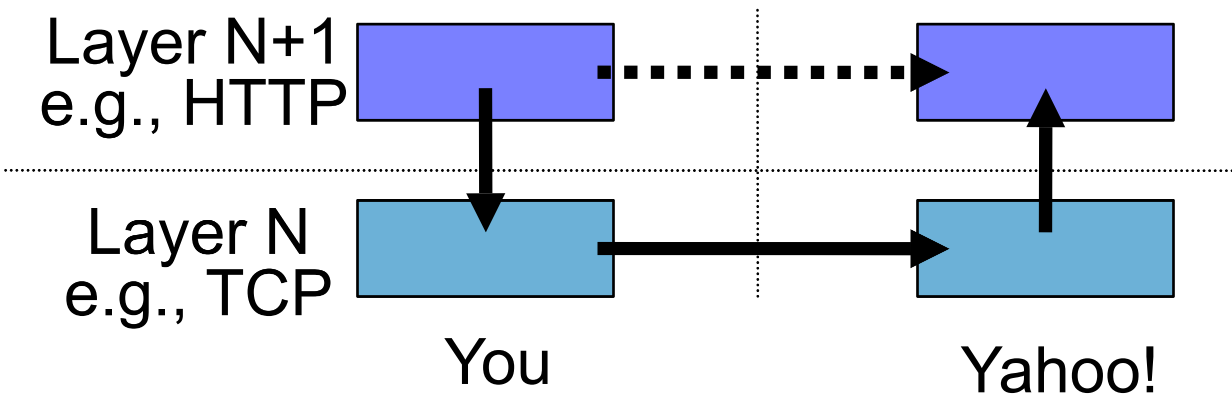
- Two parts:
  - Syntax: format -- where the bits go
  - Semantics: meaning -- what the words mean, what to do with them
- Examples:
  - IP, the Internet protocol; TCP and HTTP, for the Web
  - You can make up your own

Protocol Standards

- Different functions require different protocols
- A "standard" protocol is one that has been carefully specified so that different implementations can interoperate.
  - Standardized: screws, batteries
  - Not standardized: Appliances, furniture
- 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.

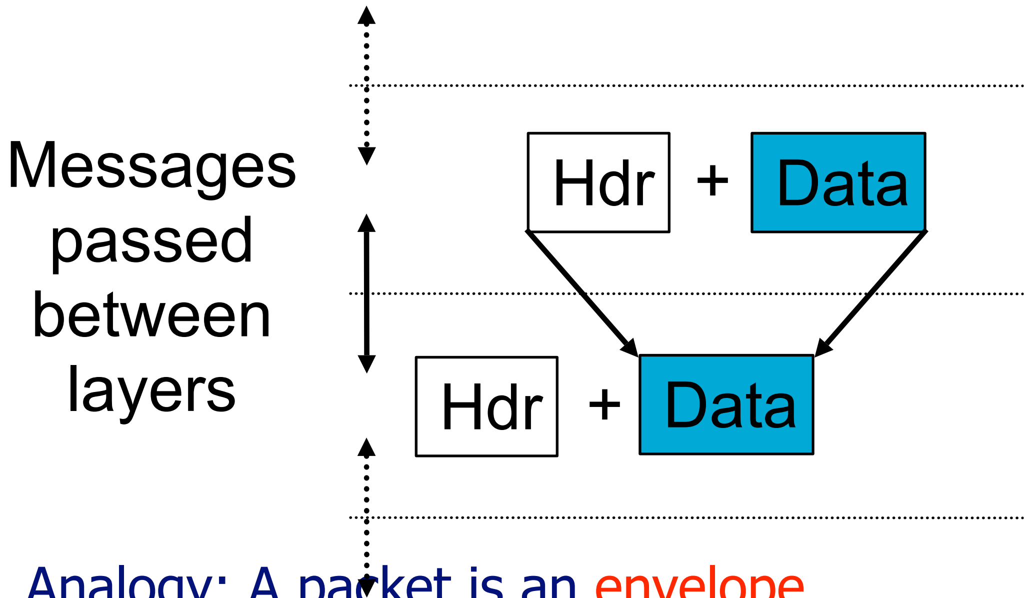
Layering and Protocol Stacks

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



Layering Mechanics

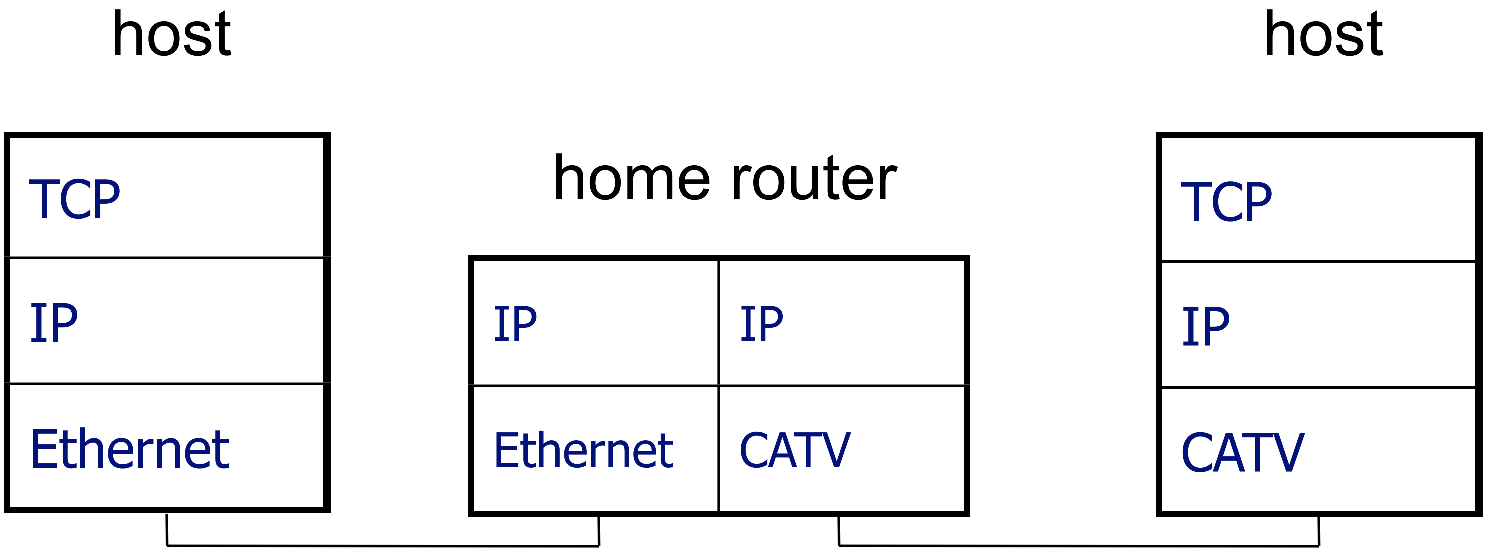
- Encapsulation and de(en)capsulation



Analogy: A packet is an envelope.

- What's written on the outside is the header
- What's contained on the inside is the payload
- The payload may, itself, be another envelope
- Each layer understands (and acts on) the writing on the outside, but doesn't understand what it contains – just delivers it.

Example – Layering at work



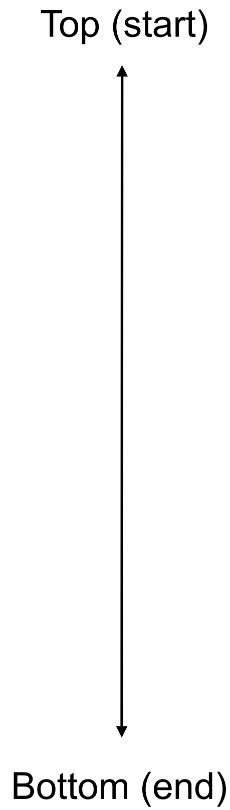
A Packet on the Wire

- Starts looking like an onion!

- Each layer treats all layers above/after it as opaque payload – the contents of the envelope
- We're still leaving out some details (segmentation and reassembly, for
- Layering adds overhead

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

What's Inside a Packet



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

What’s Inside a Packet

FROM=00:30:65:0a:ea:62,
TO=00:30:64:9a:11:22,
SIZE=200,...

Top (start)

Bottom (end)

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

What's Inside a Packet

Ethernet Header: FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,...

Top (start)

Bottom (end)

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

What's Inside a Packet

Ethernet Header: FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,...

IP Header:

Top (start)

Bottom (end)

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

What's Inside a Packet

Ethernet Header: FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,...

IP Header: FROM=128.95.1.32, TO=28.2.5.1, SIZE=200-SIZEOF(Ehdr)

Top (start)

Bottom (end)

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

What's Inside a Packet

Top (start)

Ethernet Header: FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,...

IP Header: FROM=128.95.1.32, TO=28.2.5.1, SIZE=200-SIZEOF(Ehdr)

TCP Header:

Bottom (end)

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

What’s Inside a Packet

Top (start)

Ethernet Header: FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,...

IP Header: FROM=128.95.1.32, TO=28.2.5.1, SIZE=200-SIZEOF(Ehdr)

TCP Header: FROM=Port 5000, TO=Port 80, Byte#=23, SIZE=200-SIZEOF(Ehdr)-SIZEOF(IPHdr)

Bottom (end)

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

What's Inside a Packet

Ethernet Header:	FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,...	Top (start)
IP Header:	FROM=128.95.1.32, TO=28.2.5.1, SIZE=200-SIZEOF(Ehdr)	↕
TCP Header:	FROM=Port 5000, TO=Port 80, Byte#=23, SIZE=200-SIZEOF(Ehdr)-SIZEOF(IPHdr)	
HTTP Hdr:		Bottom (end)

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

What's Inside a Packet

Top (start)

Ethernet Header:	FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,...
IP Header:	FROM=128.95.1.32, TO=28.2.5.1, SIZE=200-SIZEOF(Ehdr)
TCP Header:	FROM=Port 5000, TO=Port 80, Byte#=23, SIZE=200-SIZEOF(Ehdr)-SIZEOF(IPHdr)
HTTP Hdr:	HTTP v.1.0, Internet Explorer v5.1,...

Bottom (end)

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

What's Inside a Packet

Ethernet Header:	FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,...	Top (start)
IP Header:	FROM=128.95.1.32, TO=28.2.5.1, SIZE=200-SIZEOF(Ehdr)	↑
TCP Header:	FROM=Port 5000, TO=Port 80, Byte#=23, SIZE=200-SIZEOF(Ehdr)-SIZEOF(IPHdr)	↓
HTTP Hdr:	HTTP v.1.0, Internet Explorer v5.1,...	
Good Stuff		Bottom (end)

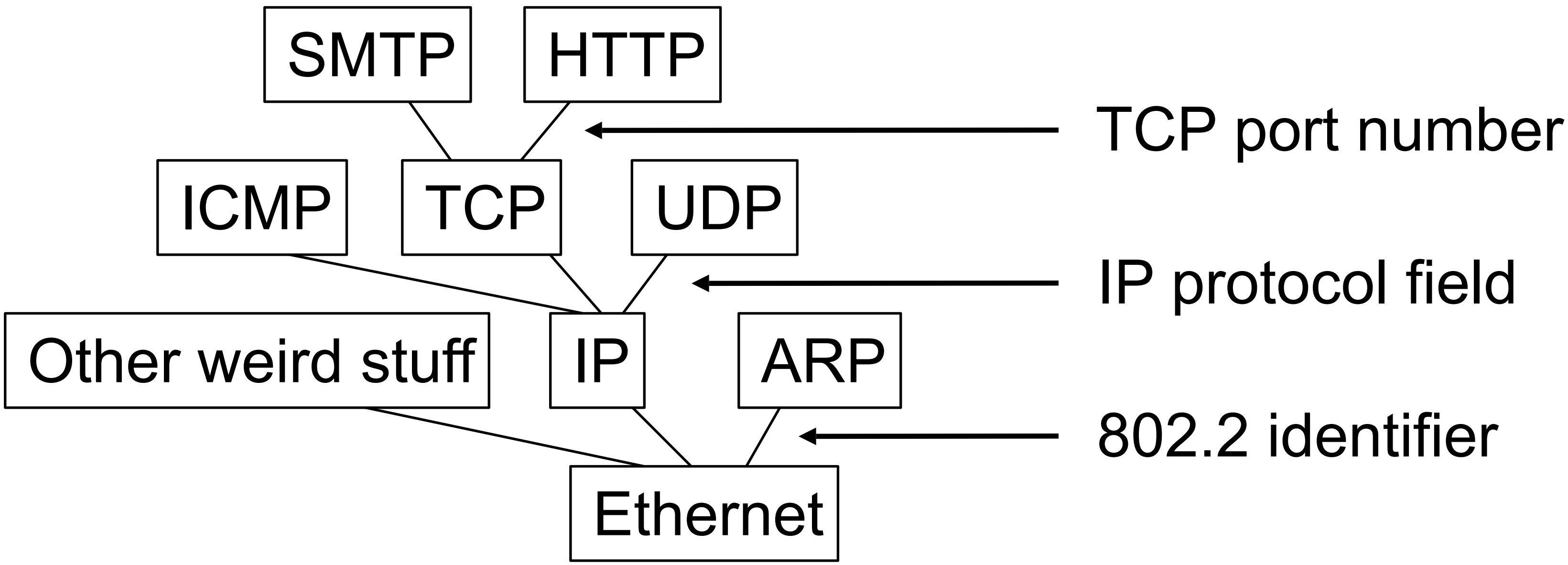
Ethernet Hdr	IP Hdr	TCP Hdr	HTTP Hdr	Payload (Web object)

What's Inside a Packet

Ethernet Header:	FROM=00:30:65:0a:ea:62, TO=00:30:64:9a:11:22, SIZE=200,…	Top (start)
IP Header:	FROM=128.95.1.32, TO=28.2.5.1, SIZE=200-SIZEOF(Ehdr)	↑
TCP Header:	FROM=Port 5000, TO=Port 80, Byte#=23, SIZE=200-SIZEOF(Ehdr)-SIZEOF(IPHdr)	↕
HTTP Hdr:	HTTP v.1.0, Internet Explorer v5.1,…	↓
Good Stuff	GET http://www.google.com	Bottom (end)

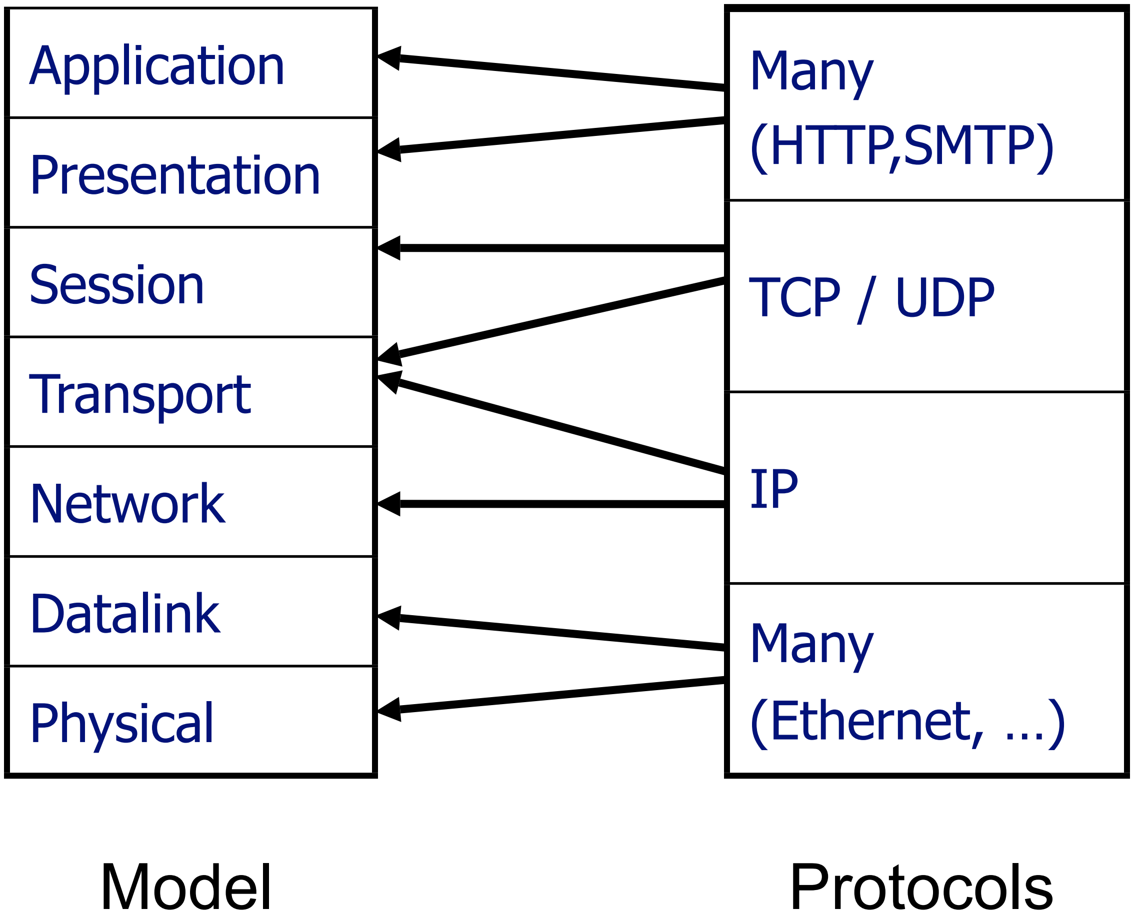
More Layering Mechanics

- Multiplexing and demultiplexing in a protocol graph



The "OSI" Model

To be honest, mostly obsolete, but I feel obligated to tell you about it in case someone important asks you.



3. The End-to-End Principle

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 extensibility of the Internet.

The End-to-End Principle

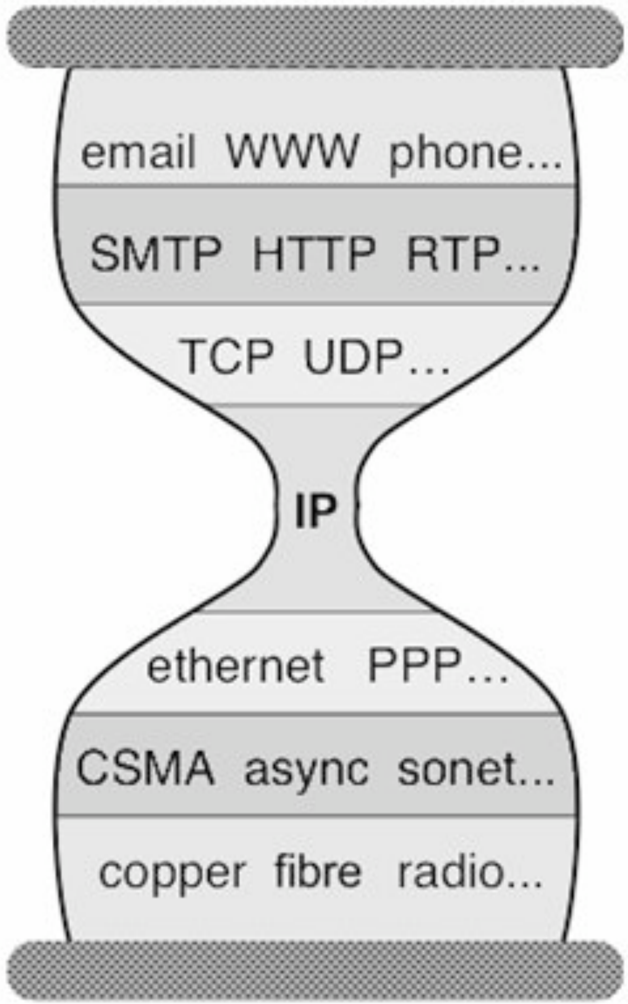
- The inside (network) is usually considered dumb and stateless
- The end-points (hosts) are smart and stateful

- Examples:
  - Reliability. Re-transmission is done by endpoints.
    - What would the advantages of in-network re-tx be?
  - Congestion control.
  - Name resolution. DNS resolution is a separate step between endpoints.
    - They could have integrated naming and routing.

The Internet Hourglass: A Narrow Waist

The "narrow waist"
is crucial to
letting the
network evolve.

Comparison:
Telephone
network.



Pros of the End-to-End

- Don’t impose performance penalty on applications that don’t need it
  - If we put reliability into Ethernet, IP, etc., then apps that don’t need reliability pay for it

- Complex middle = complex interface. Specifying policy is HARD!
  - ATM failed. This is part of the reason.

- You need it at the end-points anyway; may as well not do it twice
  - Checksums on big files

- By keeping state at the end-points, not in the network, the only failure you can’t survive is a failure of the end-points

Cons of the End-to-End principle

- Loss of efficiency
  - The end-to-end principle is sometimes relaxed
- End-points are also hard to change en masse
  - New versions of TCP can't "just be deployed"
- End-points no longer trust each other to be good actors. Result?
  - Routers now enforce bandwidth fairness (RED)
  - Firewalls now impose security restrictions
  - Caches now intercept your requests and satisfy them
    - Akamai and other CDNs ("reverse caching"): good design
    - Transparent proxy caching: breaks things

Key Concepts

- The Internet is complicated
- Protocol layers are the modularity that is used in networks to handle complexity
- The end-to-end principle gives us general guidance that complicated things should go at the edges of the network
- The simple, “narrow waist” lets technology evolve both above and below it without throwing everything out.

Project Description (if time)

- You will implement a simple protocol using both TCP and UDP
- Write a client in C that talks to a server and extracts its sweet, sweet secrets
- Can be done on any host/OS, but Linux preferred – that's what the future stages of the project will use
- The "Berkeley socket interface" is covered by Ivan
- Due Friday, October 17