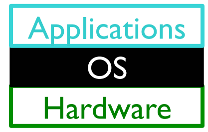
Today's Objectives

- Networking Layers

Discussion

- What is the end-to-end argument?

Review:
Layering in Network Design



- Fundamental Question: What services do applications require from the bare hardware?
  - Don't make each app implement the same functionality
  - OSs should implement these services/abstractions
- Network links hand a *frame* to the operating system
- But what abstraction does the application desire?
- Do all applications need the same abstraction?
- What abstractions do intermediate hosts (**routers**) in the network need?

Protocols

- ***Protocol***: An agreement between two "parties" as to how information is to be transmitted
  - More valuable with more users
  - How do we introduce new protocols if everyone must agree?
- Network protocols are typically implemented in software
  - Adds overhead to communication
  - Network bandwidth vs. Application throughput
  - Small matter of code to deploy new protocol
- Examples
  - **Internet Protocol (IP):** global packet transmission/addressing
  - **Transmission Control Protocol (TCP):** reliable byte stream

The Theory: OSI (Open Systems Interconnection)Model

Function			Example
Ultimate data destination	Application	Application	Web browser
Format conversion	Presentation	Presentation	ASCII/XDR
Interaction across presentation	Session	Session	Restartable file transfer
Reliable, ordered delivery	Transport	Transport	TCP
Routing/ Internetworking	Network	Network	IP
Data framing over links	Data link	Data link	Ethernet, WiFi
Bits on the wire	Physical	Physical	SONET, 100BT

Each layer adds some more data/bits

OSI Model

Function	
Ultimate data destination	Application
Format conversion	Presentation
Interaction across presentation	Session
Reliable, ordered delivery	Transport
Routing/ Internetworking	Network
Data framing over links	Data link
Bits on the wire	Physical

Where does security go?

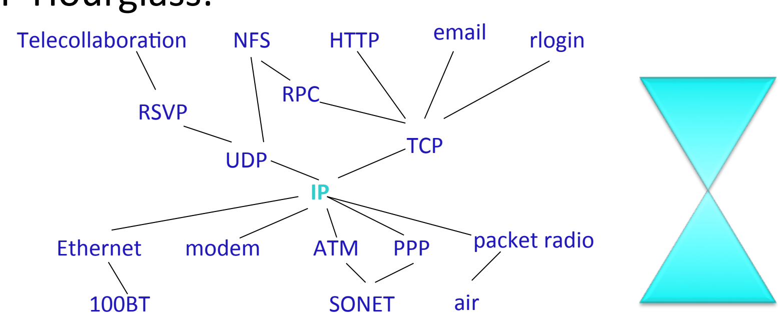
What about reliability?

OSI Model Discussion

- OSI standardized before implemented
- On the other hand… Internet Engineering Task Force
  - IETF philosophy: "We reject kings, presidents, and voting. We believe in rough consensus and working code"
  - IETF requires two working/interoperable versions before considering a standard
- Modular design but some boundaries are arbitrary
  - Why *seven* layers?
  - What exactly is the session layer?
  - Basic network functionality exists at multiple layers

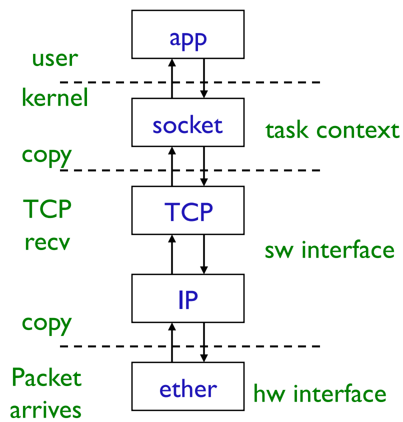
Internet Architecture

- IP Hourglass:



- Layering not strict
  - Can define new abstractions on any existing protocol

Layering in Internet Applications



- Bottlenecks
  - Boundary crossings
  - Copies
  - Context switches
- Layering nice way to logically consider protocols
  - May not lead to fastest implementation
  - But! Processors are getting faster… people are more expensive

A little OS

Layering in Network Design

- Encapsulate each layer in lower-layer format
- Ethernet's view of the packet:

Header	Payload

- IP View:

Header	Payload

- TCP View:

Header	Payload

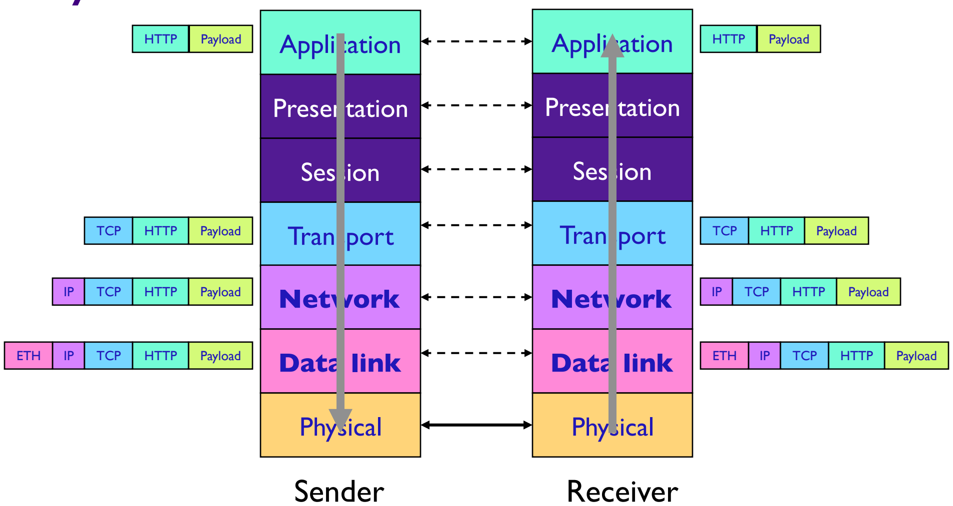
Reality:

Eth	IP	TCP	HTTP	Payload

OSI Layering: HTTP Example

Communication abstraction

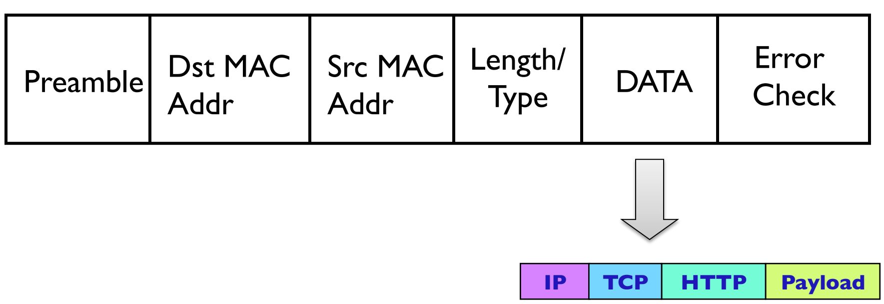
Payload: data to be sent



Each protocol adds its own headers, metadata to facilitate delivery

Datalink Layer (Layer 2): Ethernet/WiFi

- Ethernet frame format



Network Layer (Layer 3): Internet Protocol (IP)

- Service mode: *best effort*
  - No guarantees about reliable, in-order, or error-free delivery
  - Enables IP to "run over anything" (e.g., any type of network)

<table>
<tr><td>Version</td><td>HLen</td><td>TOS</td><td colspan="2">Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Offset</td></tr>
<tr><td colspan="2">TTL</td><td>Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source IP Addr</td></tr>
<tr><td colspan="5">Destination IP Addr</td></tr>
<tr><td colspan="4">Options (variable)</td><td>Pad (variable)</td></tr>
<tr><td colspan="5">Data [TCP | HTTP | Payload]</td></tr>
</table>

IP Address Allocation

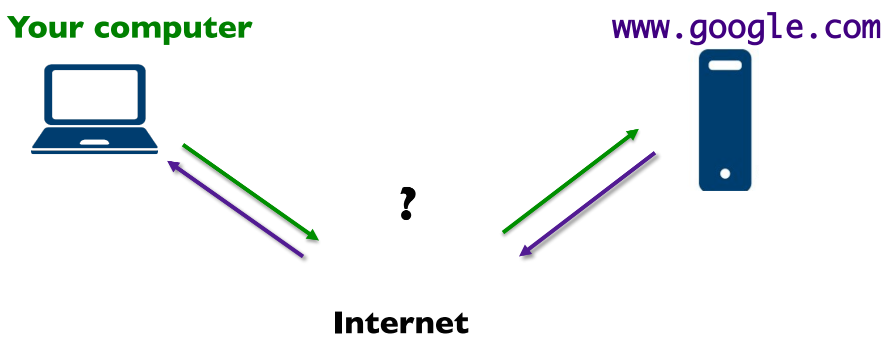
- Originally, 4 address classes
  - A: 0 | 7 bit network | 24 bit host (1M each)
  - B: 10 | 14 bit network | 16 bit host (64K)
  - C: 110 | 21 bit network | 8 bit host (255)
  - D: 1110 | 28 bit multicast group #
- Assign network # centrally, host # locally
- W&L probably has class B address
  - Prefix = 137.113 = 10001001. 01110001

End-to-End **Principle** Summary

- Argument evolved into a principle
- Idea: Keep the internet (middle) simple, end points can be complex
  - Right abstraction → Powerful!
- IP protocol allows a variety of applications to be built on top of it

EXAMPLE: WHAT HAPPENS WHEN YOU CLICK ON A WEB LINK?

What Happens When You Click on a Web Link?



Steps in Retrieving a Web Page

1. Extract hostname from URL
   - http://www.google.com/foo ➔ www.google.com
2. Use **DNS** to translate www.google.com to IP address
   - IP address used for Internet routing
   - Result: 172.217.9.196
3. Establish a **TCP** (socket) connection to 172.217.9.196
   - Protocol agreement for browser and server to speak HTTP
   - TCP handles network problems (drops, corruption, etc.)
   - TCP layered on top of IP/Ethernet
4. Routers determine efficient path to 172.217.9.196

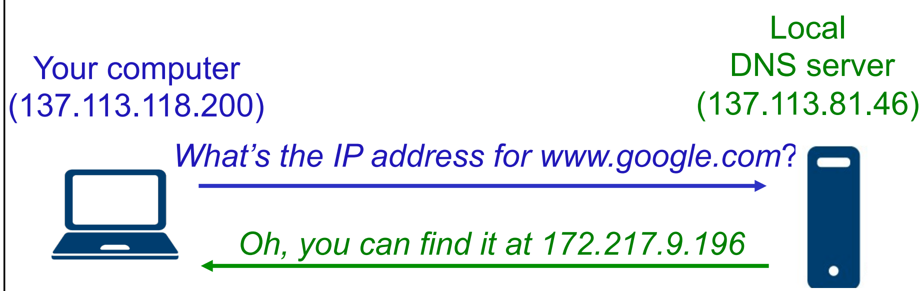
Different Kinds of Identifiers

- Domain name (e.g., *www.google.com)*
- IP Address (e.g., 172.217.9.196)
- Ethernet (e.g., *08-00-2b-18-bc-65)*

Different Kinds of Identifiers

- Domain name (e.g., *www.google.com)*
  - Global, human readable
- IP Address (e.g., 172.217.9.196)
  - Global, works across all networks
  - Some IP addrs are not global (e.g., 192.168…)
- Ethernet (e.g., *08-00-2b-18-bc-65)*
  - Also known as the Media Access Control (MAC) addres
  - Local, works on a particular network
  - BUT each Ethernet address is globally unique!

Domain name to IP address: Domain Name System (DNS)

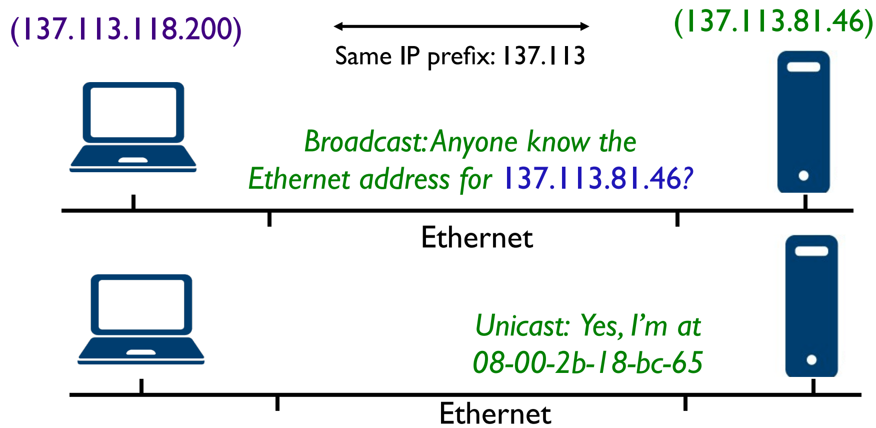


But how do we know how to find 137.113.81.46?

How Do We Know Our DNS Server?

- IP cannot be configured into hosts when they are manufactured
- **DHCP** (Dynamic Host Configuration Protocol) allows for dynamic assignment of IP addresses
  - Admins don't need to manually configure hosts
- When host needs config info, broadcast `DHCPDISCOVER`
  - DHCP server responds with IP address, Router's IP address, and DNS server address
- But we still need to know the MAC address in order to communicate
  - need to get to the lowest layer

IP address to MAC Address: Address Resolution Protocol (ARP)

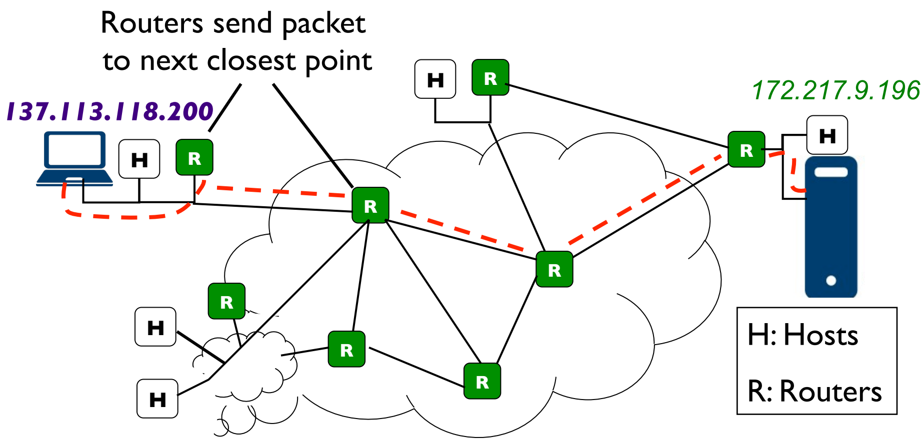


Ethernet addresses are only helpful on our local network.
Matching IP address prefixes tell us that we are on the same network!

Finding Google

1. Know Google's IP address
   - Update IP header of our packet with correct destination IP address
2. Google's IP address prefix doesn't match our IP address prefix
   - We are NOT on the same network
3. We must go through our gateway router
   - Router will pass our packet out to the Internet
4. Update Ethernet address of our packet so that destination Ethernet addr is the address of our router
   - IP address never changes!

How Does a Packet Get Through the Internet?



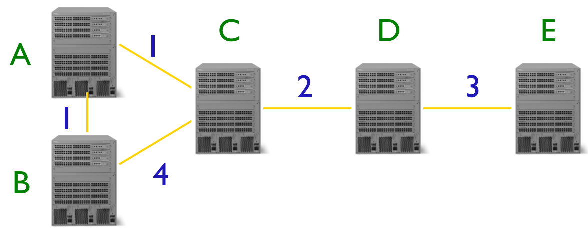
Each router uses IP prefix matching to get our packet one step closer to Google. Dest *Ethernet* addr in packet gets updated with each hop! Dest *IP* addr never changes…

How Do Routers Know Where to Send Data?

- Forwarding tables at each router
- Original Internet: manual update
- Today: automatic update based on "cost"
  - **Link State Protocol**
    - Broadcast local connectivity information throughout the network
    - Maintain table of cost to get to other destinations
    - Choose path with smallest cost

Link State Example

- Flood following information through the network
  - A: (A→C,1) (A→B,1)
  - B: (B→A,1) (B→C,4)
  - C: (C→A,1) (C→B,4) (C→D,2)
  - D: (D→C,2) (D→E,3)
  - E: (E→D,3)
- Each host runs Dijkstra's algorithm to find shortest path to all other hosts



Why Do We Need All Three?

- Domain name (e.g., *www.google.com*)
  - Global, human readable
- IP Address (e.g., 74.125.91.147)
  - Global, works across all networks
  - Some IP addrs are not global (e.g., 192.168…)
- Ethernet (e.g., *08-00-2b-18-bc-65)*
  - Also known as the Media Access Control (MAC) addres
  - Local, works on a particular network
  - BUT each Ethernet address is globally unique!

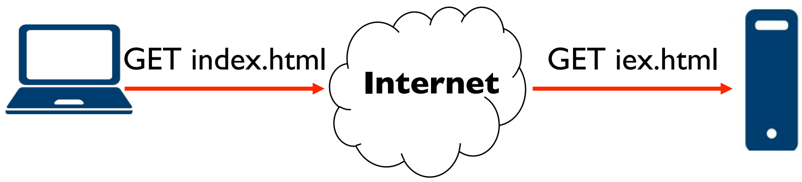
Why Do We Need All Three?

- Domain name (e.g., *www.google.com*)
  - Humans remember
- IP Address (e.g., 74.125.91.147)
  - Routing
  - Can't be assigned by manufacturer
- Ethernet (e.g., *08-00-2b-18-bc-65)*
  - Assigned by manufacturer
  - Used locally

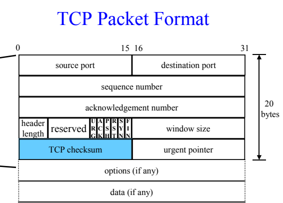
Have Address, Now Send Data

- So now we know how to find Google and send data
- Real networks must deal with congestion, bit errors, etc. that occur during data transmission
  - Data can be corrupted
  - Data can get lost
  - Data might not fit in a single packet
  - Data can be delivered in the wrong order
  - …

What If the Data Gets Corrupted?

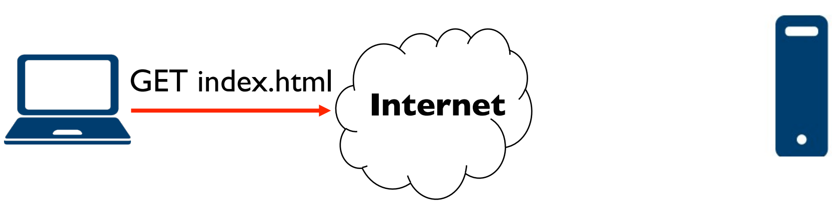


IP	TCP	Data



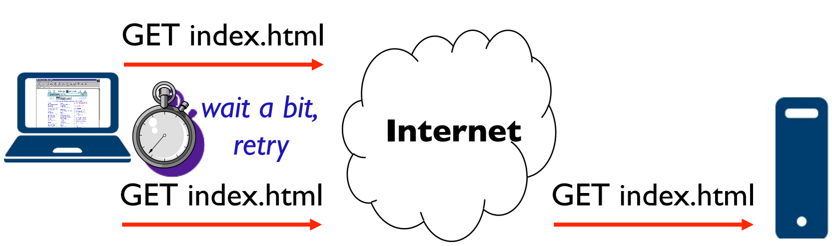
Solution: Add a *checksum*

What If the Data Gets Lost?



No ACK that data was received

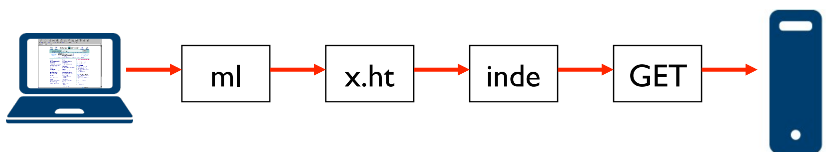
Solution: *Timeout* and *retransmit*



What If the Data Does Not Fit in a Single Packet?

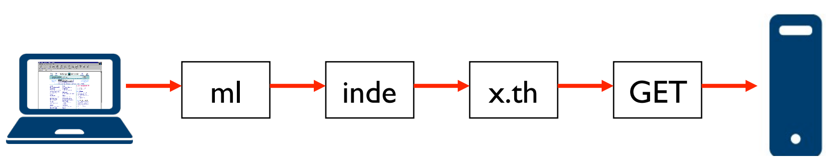
On Ethernet, max IP packet size is 1.5kbytes
Typical web page is 10kbytes

Solution: *Fragment* data across packets



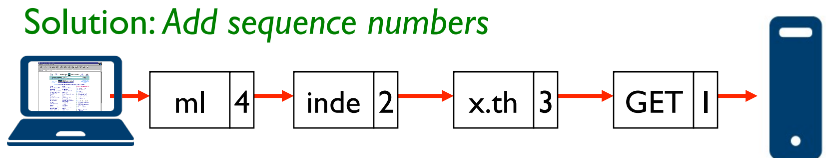
GET index.html

What If the Data is Out of Order?



GET x.thindeml
???

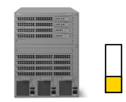
Solution: *Add sequence numbers*



GET index.html

What If Network is Overloaded?

- Data can arrive at router faster than it can be forwarded
- Short bursts: buffer at router
- What if buffer overflows?
  - Packets dropped and retransmitted
  - Sender adjusts rate until load matches available resources
- Called "Congestion control"
  - Broadcast network (Ethernet): bus arbitration
  - Wide-area networks: typically handled by TCP
  - Necessary for global fairness, but in the interests of individuals to ignore



Looking Ahead

- Internet Services – Friday
- Wed: Threads and Synchronization
- Web Server Implementation