Networking Concepts and Tools

Monday, December 1, 2014

Sources:
Kurose & Ross, *Computer Networking* (the source of many illustrations)
Randy Shull's CS242 *Computer Networks* slides;
Daniel Bilar's Fall'07 CS242 slides

CS342 Computer Security

Department of Computer Science
Wellesley College

Installing Network Tools in your Ubuntu VM

```
scp gdome@cs.wellesley.edu:/home/cs342/download/install-network-tools.sh .

./install-network-tools.sh
```

Notes:
- Execute the installer in an account with sudo privileges
- When presented with [Y/n] choice, type Y
- When presented with Yes/No GUI, navigate to Yes and press Enter/Return
- The installer will log you out of Ubuntu in order to add you to the wireshark group. Just log back in.

Overview of this week

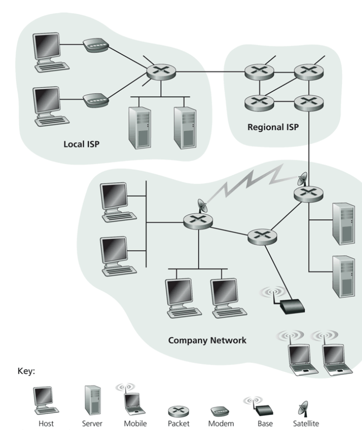
Today's lecture
- Introduction to how the Internet works, with an emphasis on layers of the Internet protocol stack.
- How these layers are used in common applications
- Linux commands for experimenting with networking concepts
- Wireshark: collect and analyze network packets

Tue/Wed lab
- EDURange recon exercise. Use nmap to explore structure of network

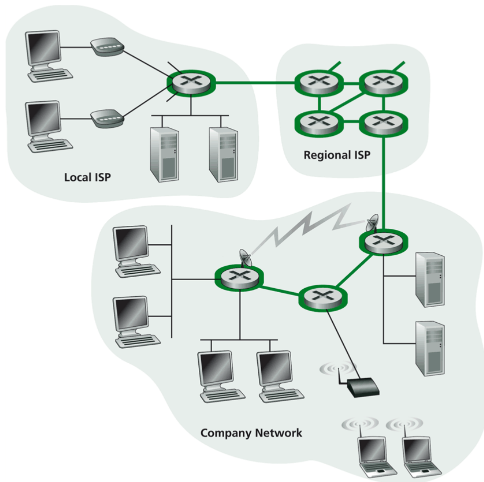
Thursday lecture:
- Vulnerabilities at each layer, how attackers exploit them, and how they can be defended.

The Internet: A nuts and bolts view

- The Internet is a network of networks consisting of:
  - hosts or end systems;
  - communication links of varying bandwidths;
  - switching devices known as routers.
- Communication paths are shared using packet switching.

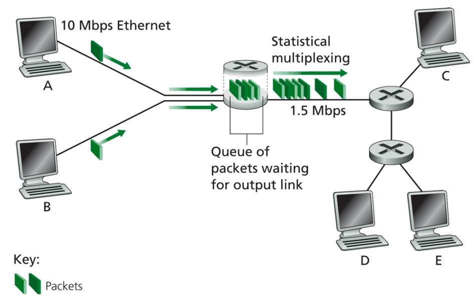


The network core



Packet switching

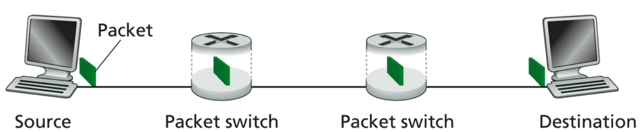
- Messages are broken into packets each of which travels from the source to destination through a maze of routers and links.



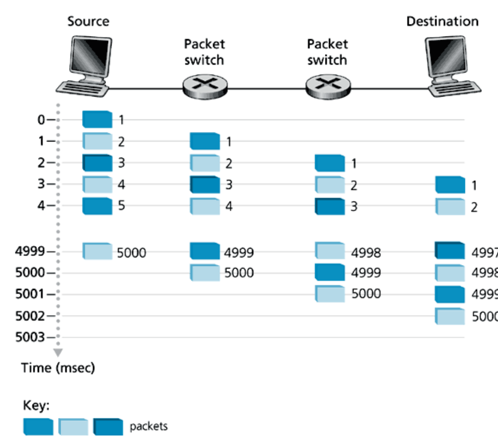
- Most routers are store-and-forward, meaning the switch must receive the entire packet before it can transmit it outbound link.

Message pipelining

- When the message is segmented into packets, the network is said to pipeline message transmission.

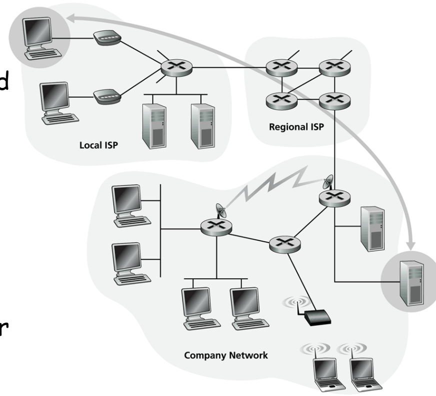


A message transfer w/ 5000 segments

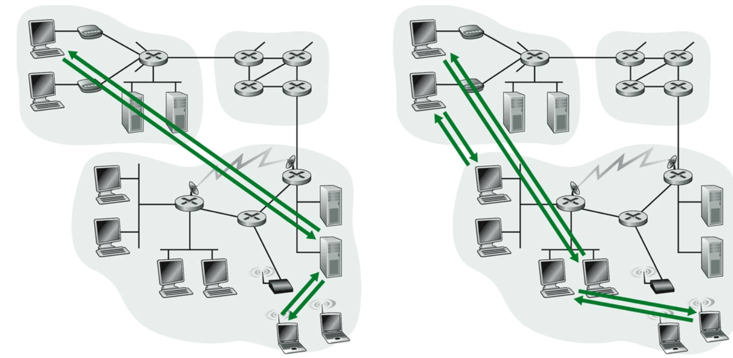


Hosts, clients and servers

- End systems or hosts are addressed by hostnames (e.g. cs.wellesley.edu) that stand for IP addresses (e.g., 149.130.136.19)
- A client program running on one host requests and receives a service from a server program running on another host.
- A single host can play client in some communications and server in others.



Network application architectures



a. Client-server application

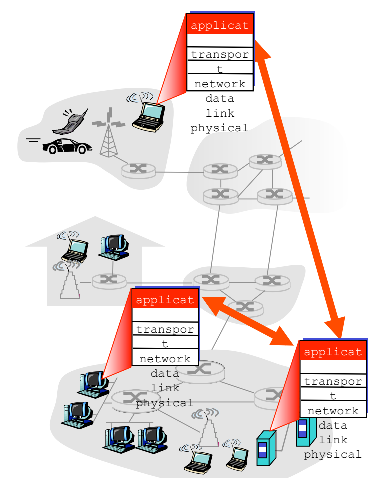
b. Peer-to-peer application

Some network apps

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips
- social networks
- voice over IP
- real-time video conferencing
- grid/cloud computing

The beauty/power of network apps

- Network apps are programs running on *end systems* at network edge
  - e.g., browser server software communicates with web server software
- Network an abstraction used for communication; network-core devices do **not** run user software.
- Applications on end systems allows for rapid app development, propagation

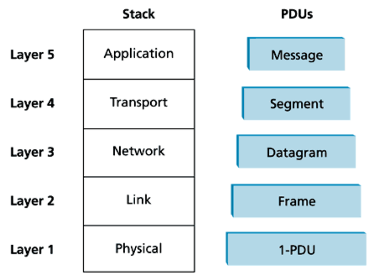


The Internet protocol stack

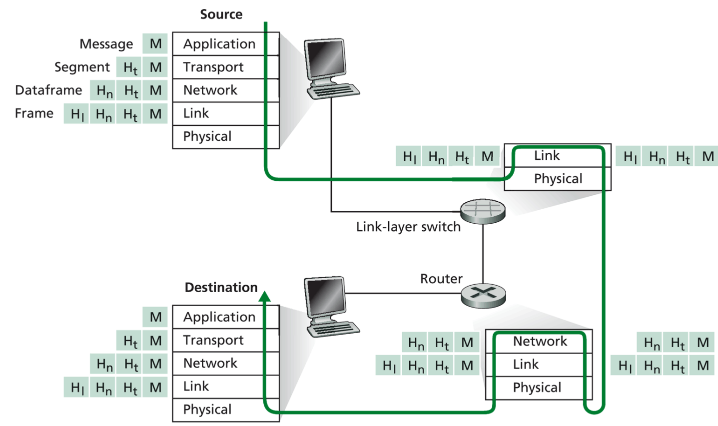
A protocol stack is a software implementation of a computer networking protocol suite

- Application:
  - What: messages between network applications
  - How: HTTP, SMTP, FTP, DNS, IMAP, POP, SSL
- Transport:
  - What: process-to-process data transfer
  - How: TCP, UDP, Appletalk
- Network:
  - What: routing of datagrams from source to dest.
  - How: IP, RIP, IPX
- Link
  - What: data transfer between neighboring network elements
  - How: Ethernet, PPP, 802.11 WiFi, ATM,
- Physical
  - What: put bits "on the wire"
  - How: Firewire, Ethernet (physical), IEEE 802.11x Wi-Fi physical layers

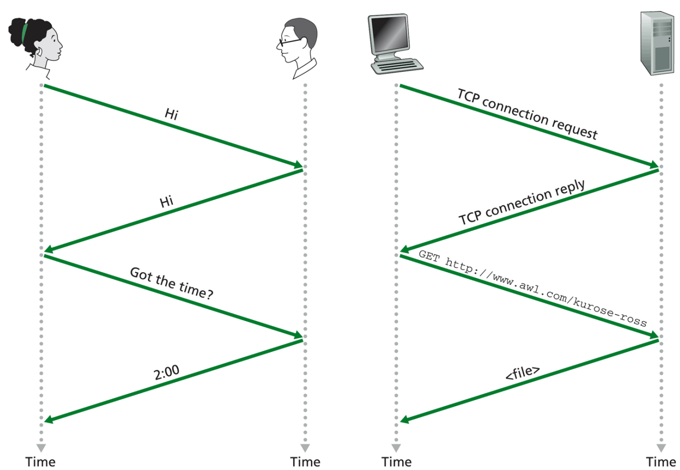
TCP/IP protocol stack



Up and down the protocol stack

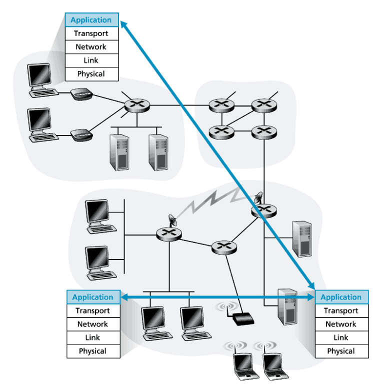


Sending and receiving information



Application-layer protocols define

- Type of message exchanged.
- The syntax of the various message types.
- The semantics of each message field.
- Rules for determining when and how a process sends messages and responds to messages.



telnet & ssh

telnet is a simple communication protocol that can be used to connect to a port on a remote machine. E.g.:

telnet cs.wellesley.edu 80

telnet sends **all** messages in cleartext. Bad for logging in remotely!

SSH (Secure Shell) is a communication protocol that encrypts all messages.

Problem: host names vs. IP addresses

- Host names
  - Mnemonic name appreciated by humans, such as

`www.wellesley.edu`

Internet resource (Web server name): www

Domain name: wellesley.edu

  - Variable length, alpha-numeric characters
  - Provide little information about location
- IP addresses
  - Numerical address appreciated by routers, such as 149.130.12.213
  - Fixed length, binary number
  - Hierarchical, can relate to host location

Names as Abstraction Barriers

Not only are names easier for humans to remember, but the level of indirection they provide is a flexible abstraction barrier that solves many problems:

- **Addresses can change underneath**: Move www.cnn.com to 173.15.201.39 when say changing providers.
- **Name could map to multiple IP addresses:** Map www.cnn.com to different addresses in different places. E.g., could return a nearby copy of the website to reduce latency or return different content.
- **Multiple names for the same address**: Aliases like cs.wellesley.edu and puma.wellesley.edu.
- **Same name for different purposes:** can use same name both for web server and mail server.

Goal: a Name -> IP Address Table

Abstractly, all we need is a table that maps names to (lists of) IP addresses:

Name	IP Address
cs.wellesley.edu	149.130.136.19
puma.wellesley.edu	149.130.136.19
www.wellesley.edu	149.130.12.213
charlotte.wellesley.edu	149.130.12.213
www.google.com	66.249.81.104
gmail.google.com	66.249.80.100
www.cnn.com	157.166.224.25 157.166.224.26 157.166.226.25 157.166.226.26 157.166.255.18 157.166.255.19

DNS: An Efficient Table Implementation

Fundamentally, the Domain Name System (DNS) is just an efficient implementation of the Name -> IP address table.

In practice, it provides other useful services, such as:

- Determining the canonical name of a site. E.g. puma.wellesley.edu for cs.wellesley.edu
- Performing reverse lookups. E.g., determining that 149.130.136.19 is known as puma.wellesley.edu
- Finding the mail server associated with a domain. E.g., the mail server for the domain wellesley.edu is cliff.wellesley.edu.
- Finding other info relevant to the DNS implementation. E.g., the authoritative name server(s) for a domain.

DNS from Linux: host and dig

```
[lynux@localhost cs342]$ host cs.wellesley.edu
cs.wellesley.edu is an alias for puma.wellesley.edu.
puma.wellesley.edu has address 149.130.136.19

[lynux@localhost cs342]$ dig www.wellesley .edu    # Some parts of response edited out

;; QUESTION SECTION:
;www.wellesley.edu.                  IN      A

;; ANSWER SECTION:
www.wellesley.edu.      86400        IN      CNAME   charlotte.wellesley.edu.
charlotte.wellesley.edu. 86400       IN      A       149.130.12.213

;; AUTHORITY SECTION:
wellesley.edu.                86400  IN      NS      cheers.wellesley.edu.
wellesley.edu.                86400  IN      NS      jeers.wellesley.edu.

;; ADDITIONAL SECTION:
jeers.wellesley.edu.    86400        IN      A       149.130.70.177
cheers.wellesley.edu.   86400        IN      A       149.130.10.16

[lynux@localhost cs342]$ dig -t MX wellesley.edu    # Most parts of response edited out
;; ANSWER SECTION:
wellesley.edu.                86400  IN      MX      1 cliff.wellesley.edu.
```

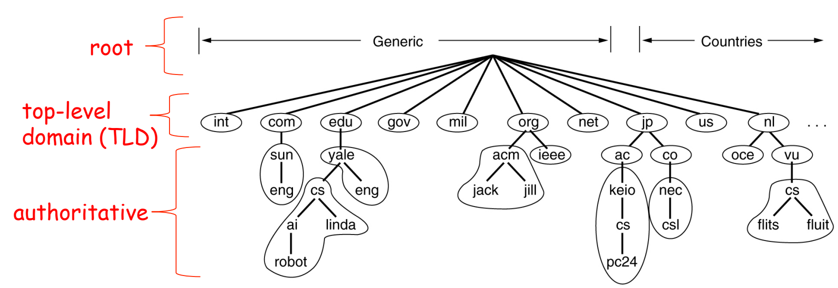

Who uses DNS?

Everyone! Almost all application-layer programs and socket programming models use DNS. E.g.

- Web browser uses DNS to determine IP address for URL hostname
- SMTP uses DNS to determine IP addresses for mail servers associated with email names.
- FTP, TELNET, etc. use DNS to determine IP address of the host they're connecting to.
- Java Socket interface. E.g. new Socket("lark.wellesley.edu", 6789)

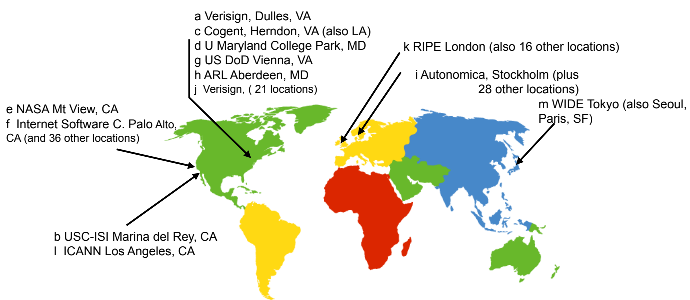
Name Server Hierarchy

Part of the DNS name space showing the division into zones.



Root name servers

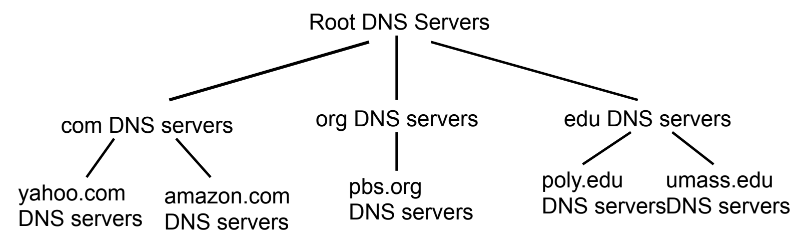
- "13" root servers (see http://www.root-servers.org/) named A through M
- Replicated for reliability/security
- Locates top-level domain (TLD) servers



DNS name lookup: first cut

What happens when client wants IP address for www.amazon.com?

- client queries a root server to find com DNS server
- client queries com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com



Actually, client uses local name server

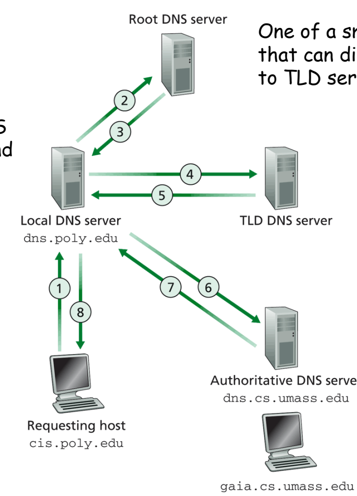
- Does not strictly belong to DNS hierarchy
- Each ISP (residential ISP, company, university) has one.
  - Also called "default name server"
- When host makes DNS query, query is sent to its local DNS server
  - Acts as proxy, forwards query into hierarchy
  - Makes things go faster because can cache results
- Lots of hosts use the same local name server

How does a DNS lookup work?

One of a small number of servers that can direct local DNS server to TLD server.

Wellesley's local DNS servers are cheers and jeers.

Results of lookup are cached here to avoid some future lookups.

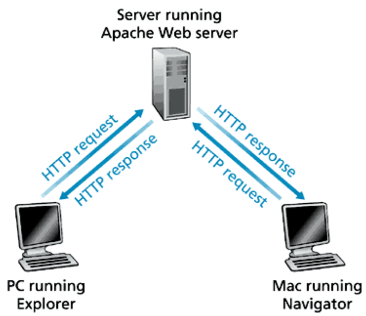


TLD (Top-level Domain) server handles a domain like .edu, .com, .uk,, .fr

Each organization has an authoritative DNS server that knows names in the organization. Wellesley's is cheers.

Hyper-Text Transfer Protocol

- HTTP is the Web's client/ server protocol.
- User agent (browser) implements the client side of HTTP.
- Web pages generally consist of a base HTML file which references other objects (JPEG, GIF, Java applet, audio clips).



HTTP GET Protocol: First Cut

Browser (client) | Web server

1. When browser wants a web page, sends HTTP GET request message to web server hostname, specifying pathname of desired page.
2. Web server at hostname replies with contents of requested page in HTTP response
3. Browser receives response message, extracts page contents, examines HTML, and requests embedded objects.

Note: protocol specifies requests & response messages, not how browser displays page!

HTTP GET example: Lyn's home page

Contents of http://cs.wellesely.edu/~fturbak/index.html

```
<html>
 <head>
  <TITLE>Franklyn Turbak Home Page</TITLE>
 </head>
 <body bgcolor=white>
  <h1> Franklyn Turbak </h1>
        ⋮
   <img align="left" valign="top" src="lyn.gif">
        ⋮
   <img src="dcpl-cover-medium.jpg" width=150>
        ⋮
 </body>
</html>
```

embedded objects

Wireshark: Protocol Capture & Analysis

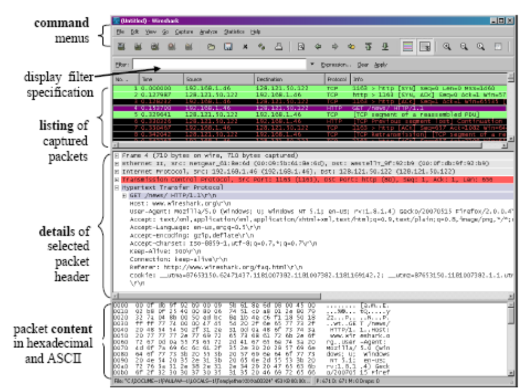


Figure 2: Wireshark Graphical User Interface

- See protocols in action
- See all layers
- Can filter packets of interest. E.g.
  - http
  - ip.addr == 149.130.136.19

HTTP GET example: Wireshark

Edited version of packet summary created by print-to-file option:

```
Source          Destination     Info
192.168.1.3     149.130.136.19  GET /~fturbak/index.html HTTP/1.1
149.130.136.19  192.168.1.3     HTTP/1.1 200 OK  (text/html)
192.168.1.3     149.130.136.19  GET /~fturbak/lyn.gif HTTP/1.1
192.168.1.3     149.130.136.19  GET /~fturbak/dcpl-cover-medium.jpg HTTP/1.1
149.130.136.19  192.168.1.3     [TCP Retransmission] HTTP/1.1 200 OK  (GIF89a)
149.130.136.19  192.168.1.3     HTTP/1.1 200 OK  (JPEG JFIF image)
```

Note that embedded objects can be requested in parallel!

HTTP GET example: request message

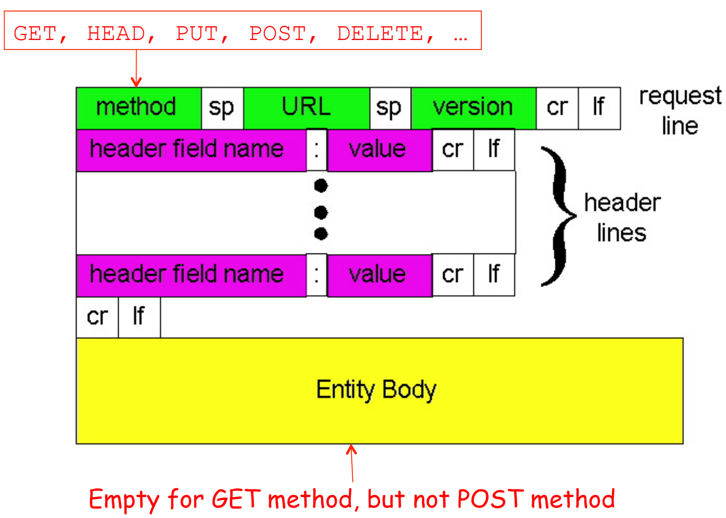
request line

header lines

```
GET /~fturbak/index.html HTTP/1.1\r\n
Host: cs.wellesley.edu\r\n
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3)
Gecko/20090824 Firefox/3.5.3 (.NET CLR 3.5.30729)\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-us\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 300\r\n
Connection: keep-alive\r\n
\r\n
```

two CRLFs indicate end of header lines

HTTP request message: general format



HTTP GET example: response message

status line

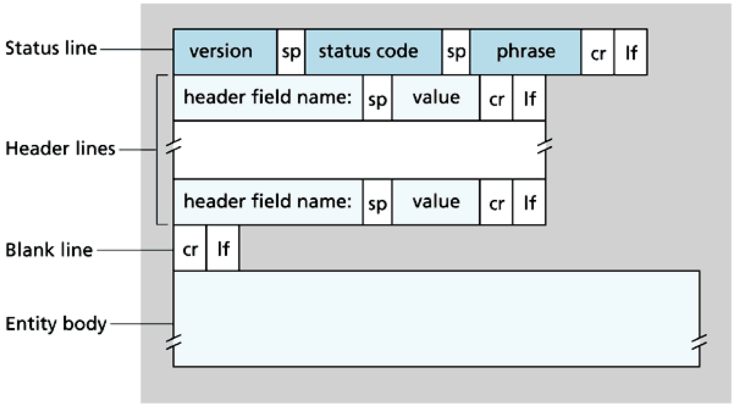
header lines

```
HTTP/1.1 200 OK\r\n
Date: Mon, 21 Sep 2009 02:24:51 GMT\r\n
Server: Apache/2.2.3 (Red Hat)\r\n
Last-Modified: Wed, 13 Aug 2008 14:19:54 GMT\r\n
Accept-Ranges: bytes\r\n
Content-Length: 3237\r\n
Connection: close\r\n
Content-Type: text/html\r\n
\r\n
<html>\n
\n
  <head>\n
    <TITLE>Franklyn Turbak Home Page</TITLE>\n
  </head>\n
\n
  <body bgcolor=white>\n
 \n
  <h1> Franklyn Turbak </h1>\n \r\n
      •
      •
      •
  </body>\n
</html>\n
\n
```

two CRLFs separate header lines from entity body

entity body

HTTP response message: general format



Some HTTP response status codes

200 OK
request succeeded, requested object in this message

301 Moved Permanently
requested object moved, new location specified in this message

400 Bad Request
request message not understood by server

404 Not Found
requested document not found on this server

505 HTTP Version Not Supported

Everything you wanted to know about HTTP

- Where is HTTP protocol specified?
- What are valid request methods?
- What do response status codes mean?
- What are valid request/response header fields?
- What goes in request/response entitiy body?

All answers are in RFC 2616:

http://www.w3.org/Protocols/rfc2616/rfc2616.html

Experimenting with HTTP via telnet

1. Telnet to your favorite Web server:

```
telnet cs.wellesley.edu 80
```

Opens TCP connection to port 80 (default HTTP server port) at cs.wellesley.edu. Anything typed in sent to port 80 at cs.wellesley.edu

2. Type in a GET HTTP request:

```
GET /~fturbak/index.html HTTP/1.1
Host: cs.wellesley.edu
```

By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. Look at response message sent by HTTP server!

Some telnet experiments

```
GET /~fturbak/index.html HTTP/1.1
```

```
GET /~fturbak/foobarbaz.html HTTP/1.1
Host: cs.wellesley.edu
```

```
GET /~fturbak HTTP/1.1
Host: cs.wellesley.edu
```

```
FOO /~fturbak/foobarbaz.html HTTP/1.1
Host: cs.wellesley.edu
```


Experimenting with HTTP via curl

```
curl http://cs.wellesley.edu/~fturbak
```

```
curl http://cs.wellesley.edu/~fturbak/
```

```
curl http://cs.wellesley.edu/~fturbak/foobarbaz.html
```


HTTP request methods

- GET: Requests a representation of the specified resource. By far the most common method used.
- HEAD: Like GET, but response has empty body. Useful for retrieving meta-information, without having to transport the entire content
- POST: Submits data to be processed (e.g. from an HTML form). The data is included in the body of the request
- PUT: uploads file in request body to path specified in URL field
- DELETE: deletes file specified in the URL field

POST example

- Use Wireshark to analyze the HTTP packets sent when processing the forms at

  http://cs.wellesley.edu/~cs242/form-post.html

  http://cs.wellesley.edu/~cs242/form-get.html
- Aside: such forms are processed by CGI scripts, PHP programs, ASP programs, Java servlets, etc.

HTTP is stateless

- Stateless protocol: server maintains no information about past client requests
- Why? Protocols that maintain state are complex!
  - past history (state) must be maintained
  - if server/client crashes, their views of state may be inconsistent, must be reconciled

Maintaining state in HTTP

- Although HTTP is stateless, sometimes the server wants to carry on a longer-lived conversation in which it remembers some things from previous requests. Examples?
- Can maintain state in HTTP via a variety of mechanisms:
  - Cookies can be used to create a 'user session' layer on top of the stateless HTTP
  - HTTP authorization can associate long-lasting conversation with user
  - Other tracking methods include hidden form fields and IP addresses.

Maintaining state: cookies

Many major Web sites use cookies

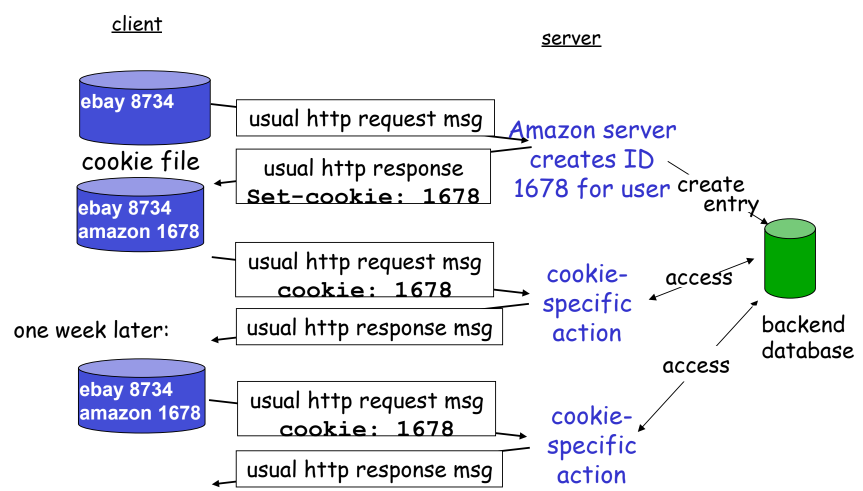
Four components:

1) cookie header line of HTTP *response* message
2) cookie header line in HTTP *request* message
3) cookie file kept on user's host, managed by user's browser
4) back-end database at Web site

Example:

- User always accesses Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

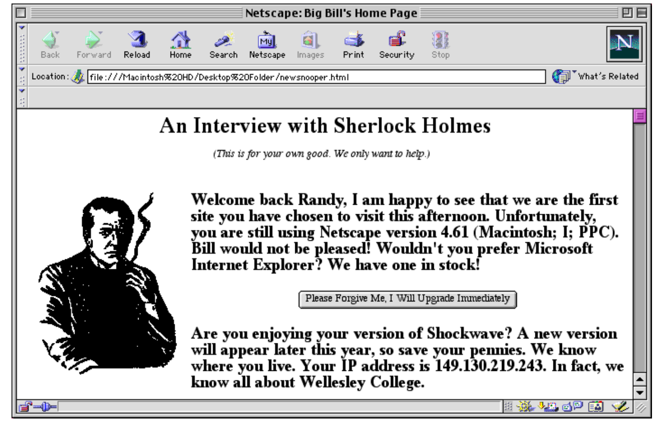
Cookie example



Firefox cookie example

- In one Firefox window, log into gmail.
- In another Firefox window, visit gmail page – why don't you have to log in?
- Use Wireshark to view the cookies being sent with the GET request.
- In Firefox, can examine google cookies via Tools>Options>Privacy>remove individual cookies
- Delete all Google cookies, and try experiment again.

Cookies and privacy



Cookies: benefits & drawbacks

What cookies do for you:
- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

What cookies do for them (and your privacy):
- Cookies permit sites to learn a lot about you

➔ What all this electronic tracking ability means to our cherished right to privacy (The **Monitored** and the **Searchable**)
"The Architecture of Privacy" (Lessig, 1998) at http://cyber.law.harvard.edu/works/lessig/architecture_priv.pdf

Maintaining State: Authorization

1. Client issues HTTP request message.
2. Server at host returns response 401: Authorization Required.
3. Client requests name and password through user agent and resends message with authorization header line including username and password.
4. Server receives response, verifies user and returns requested file.

- State here is credentials: Client resends credentials with every request

Authorization: more details

- Mechanism has to be setup on webserver
- Say user name "Aladdin" and password "open sesame"
- Combine as "Aladdin:open sesame"
- Equivalent to base-64:

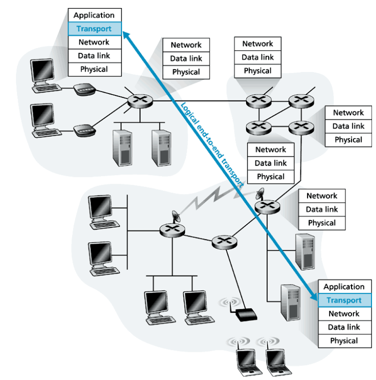
QWxhZGRpbjpvcGVuIHNlc2FtZQ==

```
GET /private/index.html HTTP/1.1
Host: cis.poly.edu
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

- Client continues to send cached username/password in subsequent requests for objects on the server.
- In this manner, the site can identify the user for every request

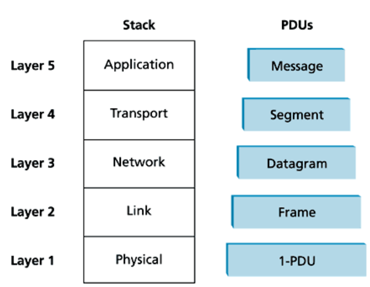
Transport-layer services

- Provides for logical communication between application processes running on different hosts.
- Transport-layer protocols are implemented in the end systems, not in the network routers.



Transport layer
runs on top of network layer

- Transport-layer protocols provide logical communication between processes.
- Network-layer protocols provides provide logical communication between hosts.
- Examples: TCP and UDP



Internet transport layer services

TCP service:

- *connection-oriented:* full-duplex (both processes can messages to each other at same time); setup required between client and server processes
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantees, security

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security
- No-frills service can be fast because of low overhead, but can only be used for services that can tolerate data loss.

Sheridan Sunier's UDP joke

"I tell UDP jokes because
I don't care if you get them"

Who uses what?

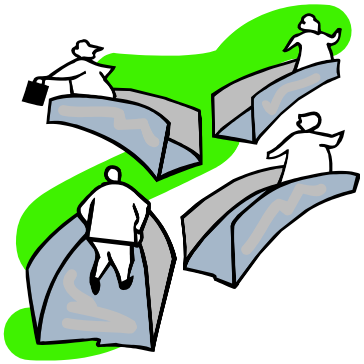
Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (eg Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP
domain name lookup	DNS	UDP
routing protocols	RIP, BGP	UDP

Process communication

- Programs don't communicate, processes do.
- Interprocess communication is governed by the end system's operating system (CS341).
- Communication between end systems is accomplished by exchanging messages across a computer network (CS242).

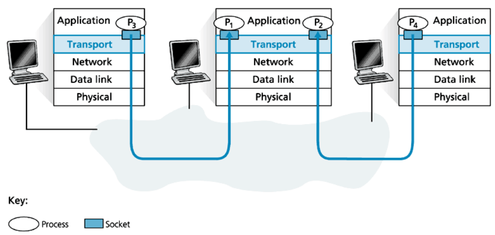
There is one problem

- Properly addressed, the network protocol can get the message to the right host.
- But what if the host is running STMP, FTP, DNP, and HTTP all at the same time?



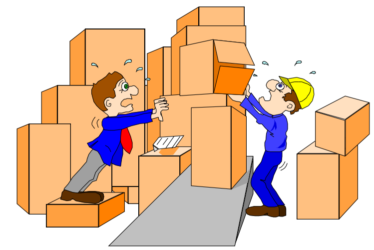
Sockets

- Each process has a socket through which data passes from the process to the network and back.
- The transport layer in the receiving host does not deliver data directly to a process, but to a socket.



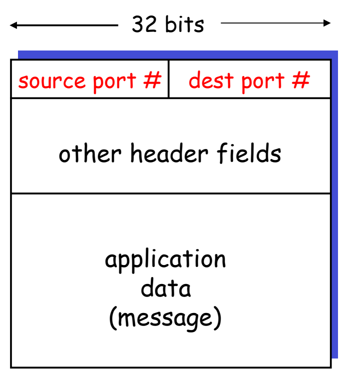
Sorting and gathering

- The task of gathering data chunks from different sockets, packaging each chunk with header information, and passing it to the network is called multiplexing.
- The task of unpacking and delivering the data to the correct socket is called demultiplexing.



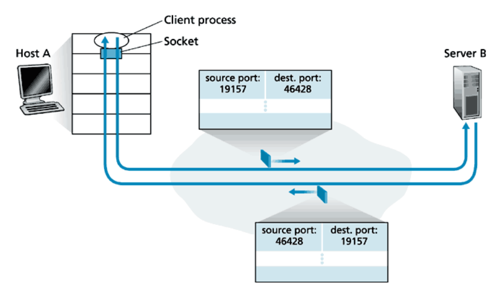
Source and destination ports

- A socket is identified with its port number.
- Application data are encapsulated into segments which are addressed to the socket they are to be delivered.



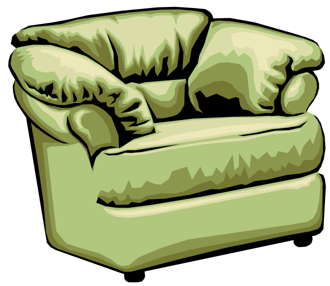
UDP

- A UDP socket is fully identified by a two-tuple consisting of a destination IP address and a destination port number.
- So why include a source port number?

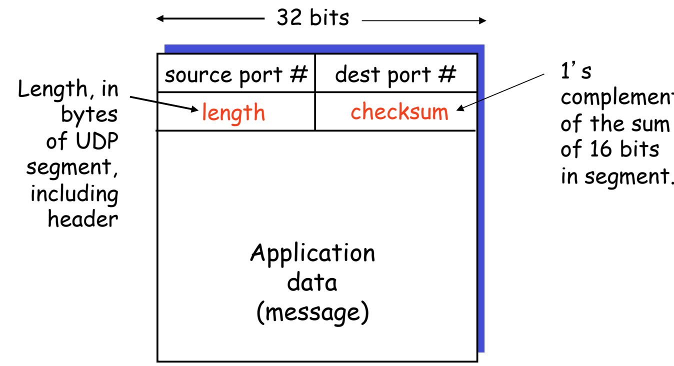


UDP does as little as possible

- Aside from multiplexing and demultiplexing and some light error checking, UDP adds nothing to IP.
- So why use it?
  - No connection establishment.
  - No connection state.
  - Small packet header overhead.
  - Finer application-level control over what data is sent.



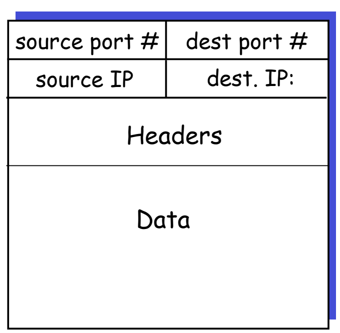
UDP segment structure



UDP segment format

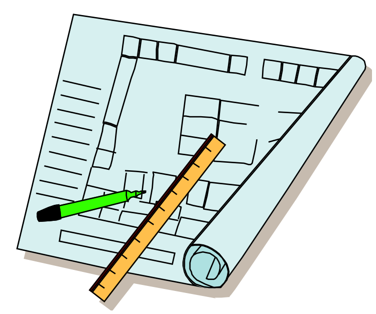
How does TCP keep all those connections straight?

- For starters, a TCP socket is identified by a 4-tuple:
  - Source IP;
  - Source port number;
  - Destination IP;
  - Destination port number.
- In particular, arriving TCP segments with different source IP addresses or source port numbers go to different sockets.



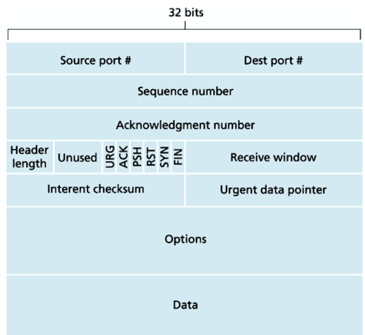
The TCP connection

- Two processes must handshake before exchanging data.
- TCP connection provides full-duplex, point-to-point, reliable communication.



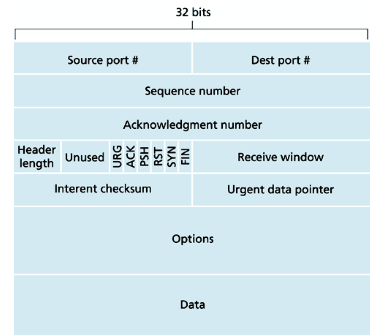
The man behind the curtain

- TCP pairs each chunk of client data with a TCP header, forming a TCP segment.
- Segments are passed down the network layer, where they are separately encapsulated within network-layer IP datagrams and sent out over the network.

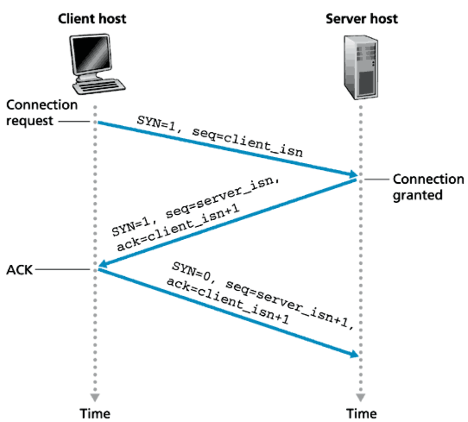


Setting up a TCP connection

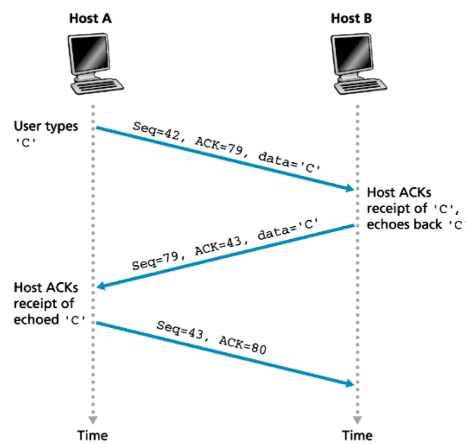
1. Client host sends TCP SYN segment to server specifying initial sequence number.
2. Server host receives SYN, replies with SYNACK and its initial sequence number.
3. Client receives SYNACK, replies with ACK segment which may contain data.



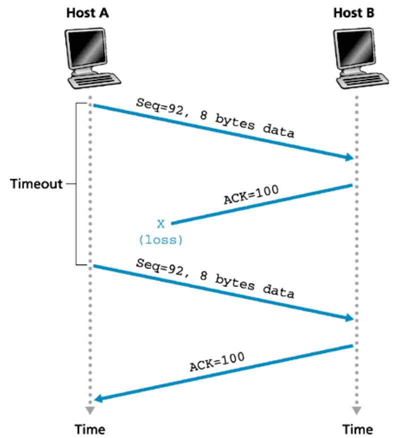
TCP three-way handshake



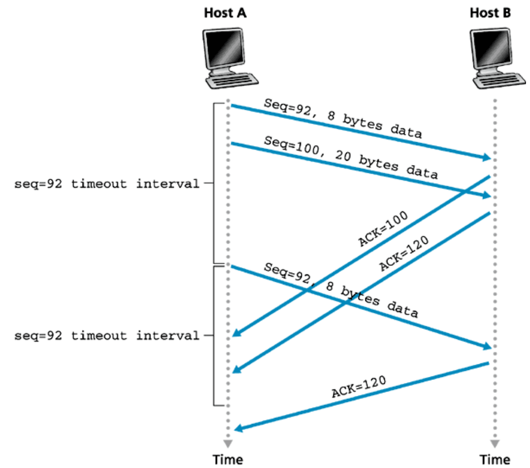
Telnet application over TCP



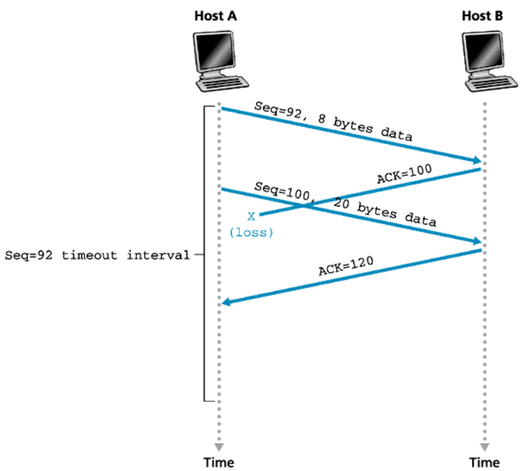
Retransmission due to lost ACK



Segment 100 not retransmitted

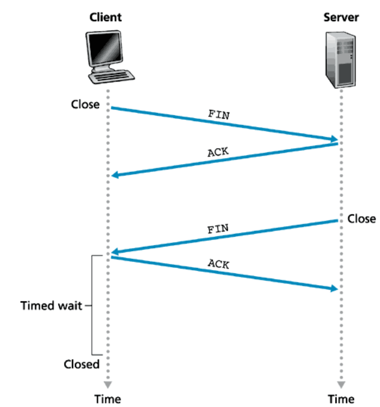


Cumulative ACK avoids retransmission



Closing a connection

1. Client end system sends FIN control segment to server.
2. Server receives FIN, replies with ACK, begins closing.
3. Server sends its own FIN.
4. Client receives FIN, replies with ACK. Enters timed wait to respond to additional FINs.
5. Server receives ACK. Connection closed.



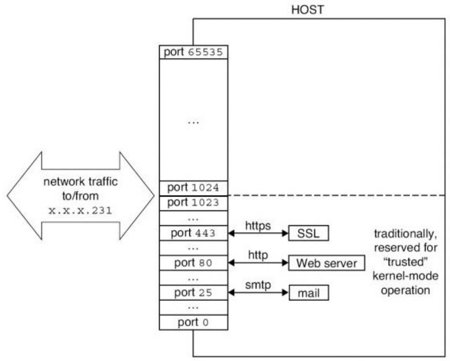
Port Scanning (nmap, basis of recon lab)

Goals of scanning:

- network topology of target s
- OSs used by target system
- which ports/services running individual machines

Techniques:

- send TCP/UDP packets to st port numbers to see if activ
- banner grabbing from active
- TCP stack fingerprinting to target machine OS
- search for application-level remote procedure calls (RPC) on target
- ping sweeping/traceroute to determine network topology
- *nmap* and other high-level tools automate some steps



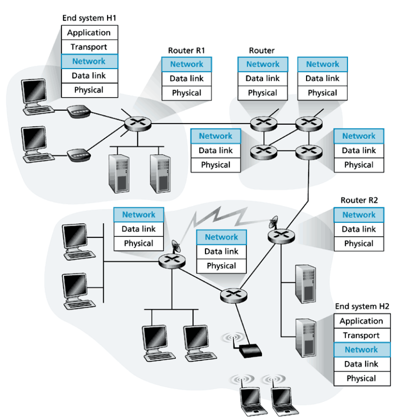
Network Layer: Internet Protocol (IP)

	Stack	PDUs	Examples
Layer 5	Application	Message	HTTP, FTP, SMTP, POP3 IMAP, DNS
Layer 4	Transport	Segment	UDP, TCP
Layer 3	Network	Datagram	IP
Layer 2	Link	Frame	Ethernet, 802.11 WiFi
Layer 1	Physical	1-PDU	

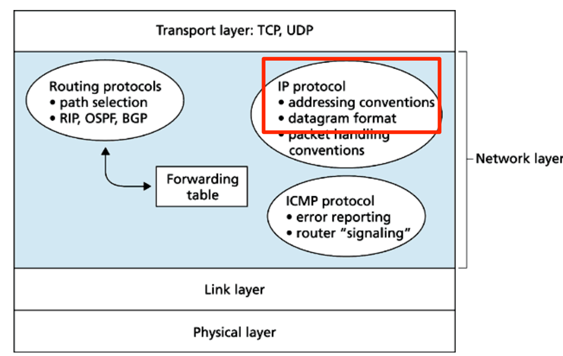
You are here!

Network layer services

- The transport layer is responsible for application to application.
- The network layer is responsible for host to host.
  - Determine the path taken by packets.
  - Forwards packets from one router to the next in the path.
- Internet Protocol (IP) service model is best-effort delivery, but it makes no guarantees. Can drop packets!

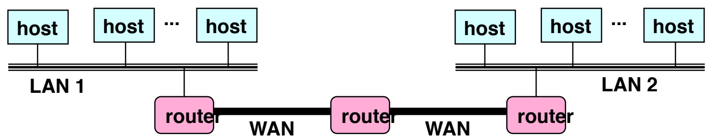


Major IP components



Grouping related hosts

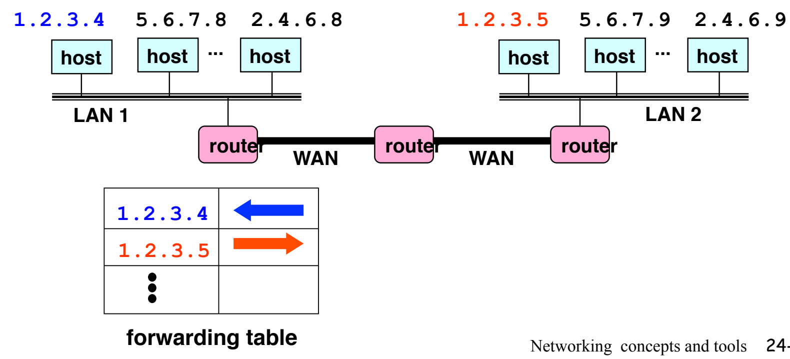
- The Internet is an “inter-network”
  - Used to connect (*sub*)*networks* together, not *hosts*
  - Needs a way to address a network (i.e., group of hosts)



LAN = Local Area Network
WAN = Wide Area Network

Scalability challenge

- Suppose hosts had arbitrary addresses
  - Then every router would need a lot of information
  - …to know how to direct packets toward the host



Classless Inter-Domain Routing (CIDR)

Use two 32-bit numbers to represent a network.
Network number = IP address + Mask

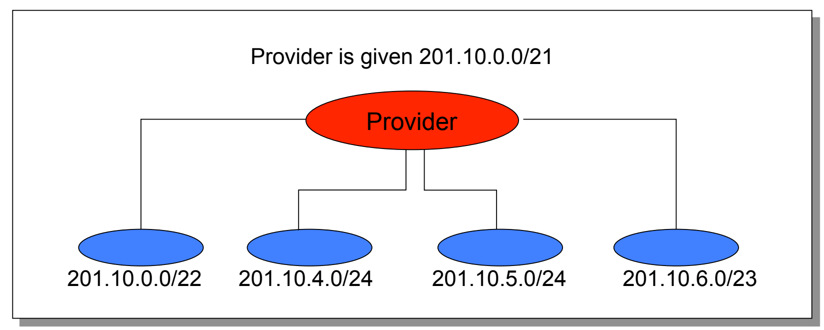
IP Address : 12.4.0.0 IP Mask: 255.254.0.0

Address	00001100	00000100	00000000	00000000
Mask	11111111	11111110	00000000	00000000

|← **Network Prefix** →|← **for hosts** →|

Written as 12.4.0.0/15

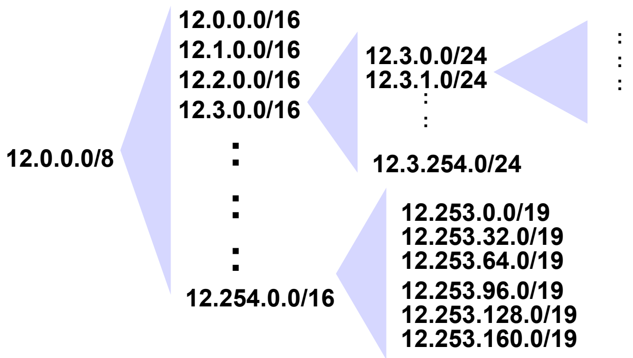
Scalability: Address Aggregation



Routers in the rest of the Internet just need to know how to reach 201.10.0.0/21.
The provider can direct the IP packets to the appropriate customer.

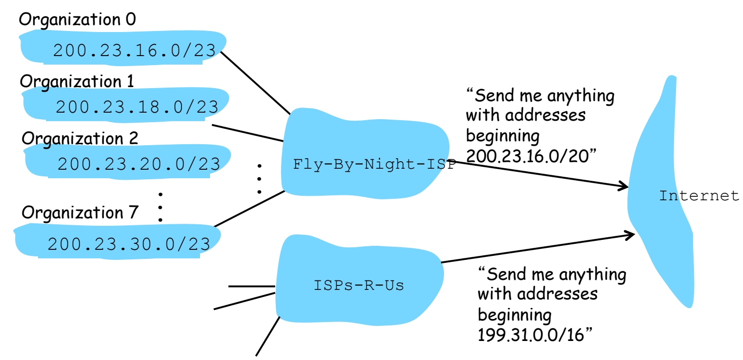
CIDR: Hierarchal Address Allocation

- **Prefixes** are key to Internet scalability
- Address allocated in contiguous chunks (prefixes)
- Routing protocols and packet forwarding based on prefixes
- Today, routing tables contain ~150,000-200,000 prefixes



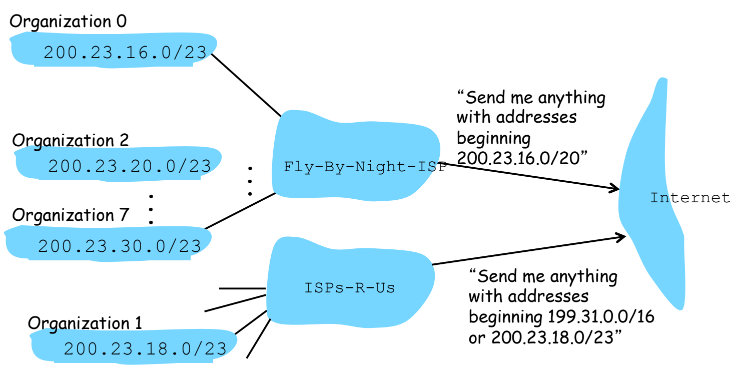
CIDR: Address aggregation

Hierarchical addressing allows efficient advertisement of routing information:

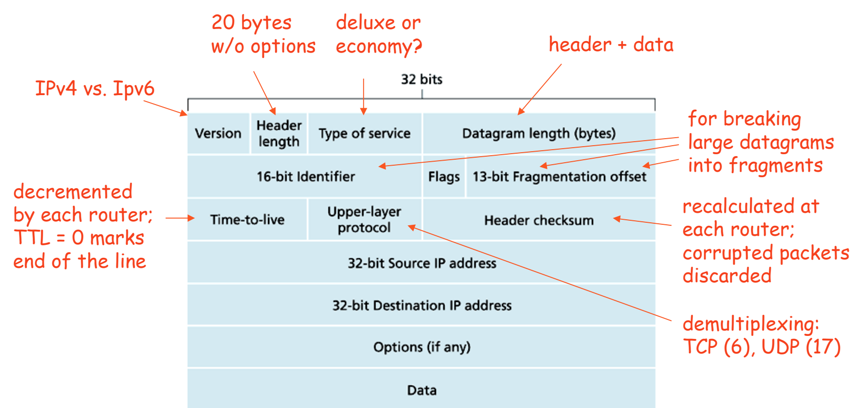


CIDR: More specific address

Suppose Organization 1 moves to ISPs-R-Us:

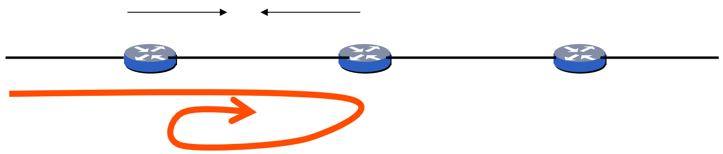


IPv4 datagram format



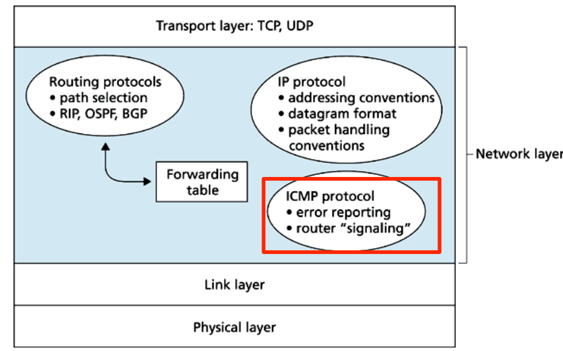
Time-to-Live (TTL)

- Potential robustness problem
  - Forwarding loops can cause packets to cycle forever
  - Confusing if the packet arrives much later



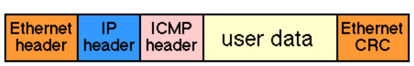
- Time-to-live field in packet header
  - TTL field decremented by each router on the path
  - Packet is discarded when TTL field reaches 0…
  - …and “time exceeded” message is sent to the source

Major IP components



ICMP (Internet Control Message Protocol)

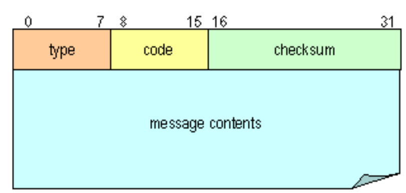
- IP network “feedback” messages
- Used to report problems with delivery of IP packets within IP networks, also for queries
- Encapsulated in an IP packet



- Not authenticated!

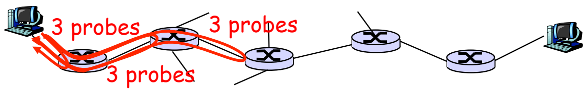
Basic ICMP Message Types

Type	Code	Desc	Query/Error
0	0	**Echo reply** e.g. ping	Q
3	1	**Host unreachable**	E
3	3	**Port unreachable** (see traceroute)	E
8	0	**Echo request** e.g. ping	Q
11	0	**Time-to-live is zero** during transit (see traceroute)	E



Message types: 40 assigned, 255 possible, ~ 25 in use

ICMP: traceroute



- **Trace route** attempts to measure delay from source to each router along an Internet path towards destination.
- Traceroute sends ordinary messages to dest with TTLs of 1, 2, 3, … and times them until notified of their demise. The host where the message expires phones home (type 11 code 0) with the sad news. Sends three packets for each TTL value.
- One of the datagrams will eventually make it all the way to the destination host. Because this datagram contains a UDP segment with an unlikely port number, the destination host sends a port unreachable port ICMP message (type 3 code 3) back to the source. When the source receives this ICMP message, it knows it does not need to send additional probe packets.

Traceroute from gaia.cs.umass.edu

3 delay measurements

```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 ms 18 ms
6  abilene-vbns.abilene.ucaid.edu (198.32.11.9)  22 ms  18 ms  22 ms
7  nycm-wash.abilene.ucaid.edu (198.32.8.46)  22 ms  22 ms  22 ms
8  62.40.103.253 (62.40.103.253)  104 ms 109 ms 106 ms
9  de2-1.de1.de.geant.net (62.40.96.129)  109 ms 102 ms 104 ms
10  de.fr1.fr.geant.net (62.40.96.50)  113 ms 121 ms 114 ms
11  renater-gw.fr1.fr.geant.net (62.40.103.54)  112 ms  114 ms  112 ms
12  nio-n2.cssi.renater.fr (193.51.206.13)  111 ms  114 ms  116 ms
13  nice.cssi.renater.fr (195.220.98.102)  123 ms  125 ms  124 ms
14  r3t2-nice.cssi.renater.fr (195.220.98.110)  126 ms  126 ms  124 ms
15  eurecom-valbonne.r3t2.ft.net (193.48.50.54)  135 ms  128 ms  133 ms
16  194.214.211.25 (194.214.211.25)  126 ms  128 ms  126 ms
17  * * *
18  * * *
19  fantasia.eurecom.fr (193.55.113.142)  132 ms  128 ms  136 ms
```

trans-oceanic link (hop 8)

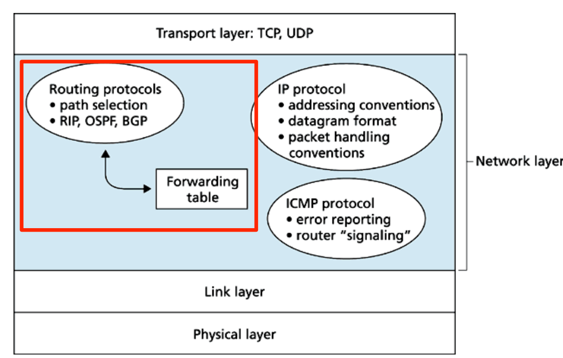
no response (hop 17)

ICMP: echo (a.k.a. ping)

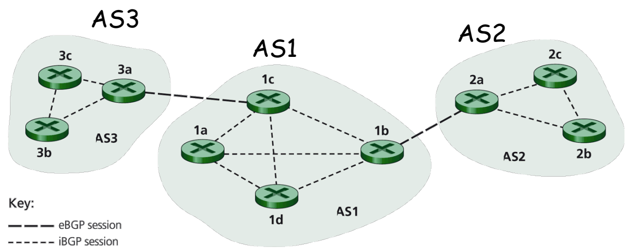
- Source host sends an echo request (“ping”, type 8 code 0)
- The destination host replies to source IP of request with echo reply (“pong”, type 0 code 0)
- Data received in the echo message must be returned in the echo reply.
- How can this be abused? (ping flood!)

```
[fturbak@puma ~] ping cardinal.wellesley.edu
PING cardinal.wellesley.edu (149.130.136.43) 56(84) bytes of data.
64 bytes from cardinal.wellesley.edu (149.130.136.43): icmp_seq=1 ttl=64
    time=1.01 ms
64 bytes from cardinal.wellesley.edu (149.130.136.43): icmp_seq=2 ttl=64
    time=0.466 ms
64 bytes from cardinal.wellesley.edu (149.130.136.43): icmp_seq=3 ttl=64
    time=0.390 ms
64 bytes from cardinal.wellesley.edu (149.130.136.43): icmp_seq=4 ttl=64
    time=0.292 ms
```


Major IP components



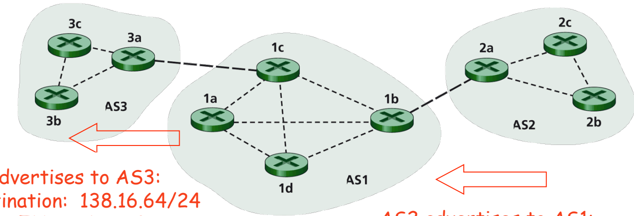
Routing Protocols



- For scalability reasons, networks are decomposed into Autonomous Systems (ASes). ISP may have one or many of these.
- The forwarding tables that routers use to forward packets are determined by two kinds of routing protocols:
  - Intra-AS routing protocols (e.g., RIP, OPSF) for internal dests.
  - Inter-AS routing protocols (e.g., BGP = Border Gateway Protocol) for external dests.

BGP: AS Advertisements

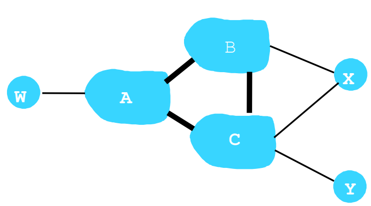
- BGP allows subnet to advertise "I am here" to rest of Internet. "
- BGP determines "good" routes to subnets based on reachability information and policy.
- When AS2 advertises a prefix to AS1:
  - AS2 promises it will forward datagrams towards that prefix.
  - AS2 can aggregate prefixes in its advertisement

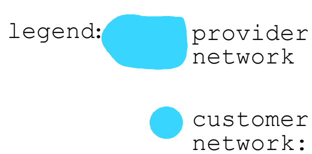


AS1 advertises to AS3:
Destination: 138.16.64/24
AS-PATH: AS1; AS2
NEXT-HOP: IP address of 1c's interface to AS3.

AS2 advertises to AS1:
Destination: 138.16.64/24
AS-PATH: AS2
NEXT-HOP: IP address of 2a's interface to AS3.

BGP Routing Policy





- Inter-AS routing determined by a combination of performance and policy.
- Suppose X does not want to route from B via X to C. Then it will not advertise to B a route to C
- Suppose A advertises path AW to B and B advertises path BAW to X. Should B advertise path BAW to C?
  - No way! B wants to route *only* to/from its customers! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - Instead, B wants to force C to route to w via A

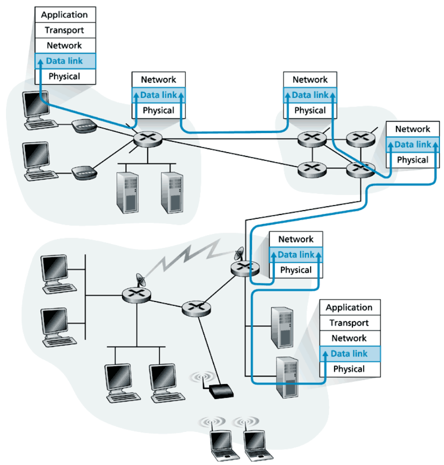
Link Layer

	Stack	PDUs	Examples
Layer 5	Application	Message	HTTP, FTP, SMTP, POP3 IMAP, DNS
Layer 4	Transport	Segment	UDP, TCP
Layer 3	Network	Datagram	IP
Layer 2	Link	Frame	Ethernet, 802.11 WiFi
Layer 1	Physical	1-PDU	

You are here! (Layer 2)

The link layer

- The transport layer provides communication of segments between two processes.
- The network layer provides communication of datagrams between two hosts.
- The link layer provides communication of frames between two network nodes (routers or hosts) connected by a link (i.e. can communicat directly with each other).

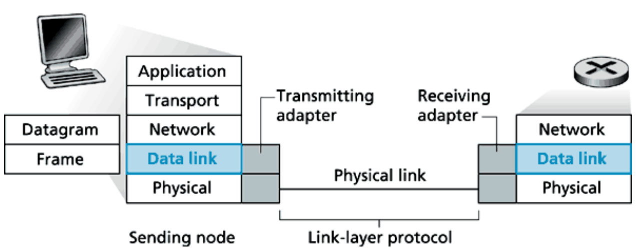


Link layer protocols

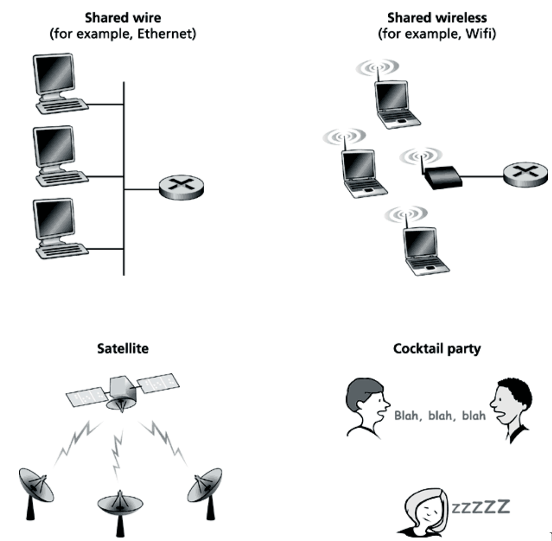
- Lots of them, including Ethernet, 802.11 wireless LAN (WiFi), token ring, PPP, HDLC, and ATM.
- Different links in a path may use different protocols.
- Responsibilities include one or more of following:
  - framing,
  - link access
  - reliable delivery
  - flow control
  - bit-level error detection (and possibly error correction).
  - half-duplex vs. full-duplex.

Adapters (NICs)

- The link-layer protocol is implemented in an adapter, a board containing RAM, DSP chips, host bus interface, and a link interface.
- Adapters sometimes called network interface cards (NICs)



Multiple Access Protocols

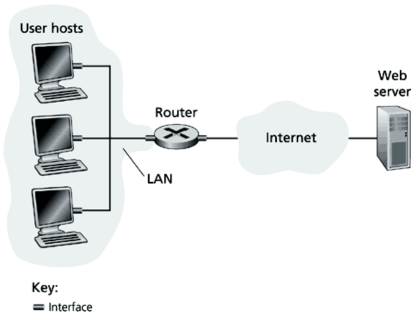


Key technical problem: when two or more nodes transmit frames at the same time, the frames collide and both transmissions are lost.

There are several solutions to this problem, which involve detecting collisions and retransmitting. See CS242 for details.

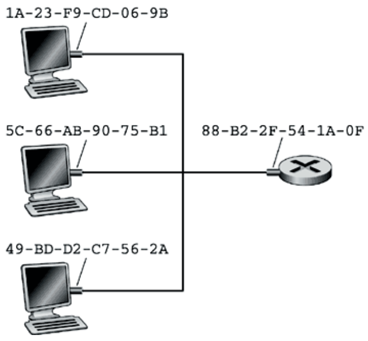
LAN Addressing

- LANs transmit frames over a broadcast channel using LAN addresses.
- On the receiving end,
  - If a destination address matches the node's LAN address, it extracts the network-layer datagram and passes it up the protocol stack.
  - If the destination address doesn't match, the node discards the frame.



MAC address

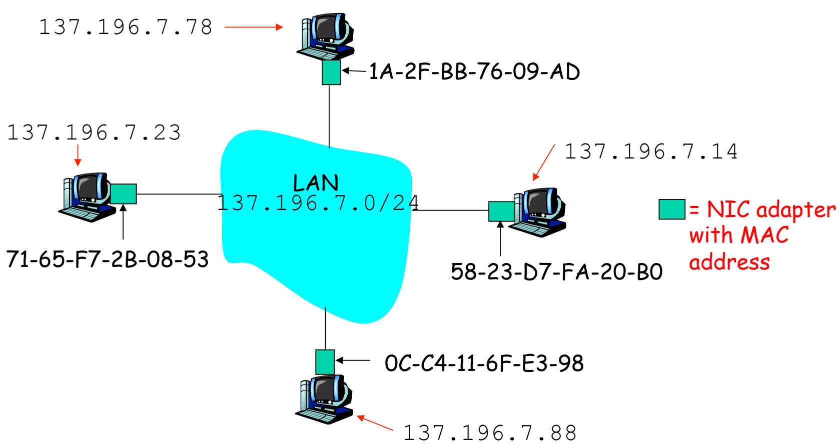
- A LAN node's MAC (Medium Access Control) address (a.k.a physical, Ethernet or LAN) properly belongs its adapter.
- Generally 48 bits long, the address is intended to be permanent unique ID burnt into the adapter's ROM. (But we'll see that in practice it's changeable!)
- LAN addresses have a flat structure (portable), as opposed to the IP hierarchical structure (routable).
- For Ethernet and token-passing LANs, broadcast MAC address is string of 48 1s: FF-FF-FF-FF-FF-FF.
- IEEE manages address space – allocates 1st 24 bits to manufacturers, who can use last 24 bits



MAC Address vs. IP Address

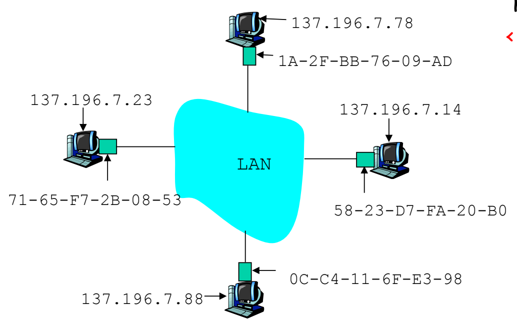
- MAC addresses "Physical address", Layer 2
  - Hard-coded in ROM of network interface card
  - Similar to social security number (*almost* unique, immutable)
  - .. but flat name space of 48 bits (e.g., 00-0E-9B-6E-49-76)
  - Stays the same when host moves
  - Used to get packet between interfaces on *same* network
- IP addresses "Logical address", Layer 3
  - Can be configured manually or learned dynamically
  - Similar to postal mailing address (change of address is easy)
  - Hierarchical name space of 32 bits (e.g., 12.178.66.9)
  - May change depending on where the host is attached
  - Used to get a packet to *any* destination IP subnet

Example: MAC/IP addresses



ARP: Address Resolution Protocol

<u>Question:</u> how to determine MAC address of B knowing B's IP address?



- Each IP node (host, router) on LAN has ARP table
- ARP table: IP/MAC address mappings for some LAN nodes

  < IP address; MAC address; TTL>

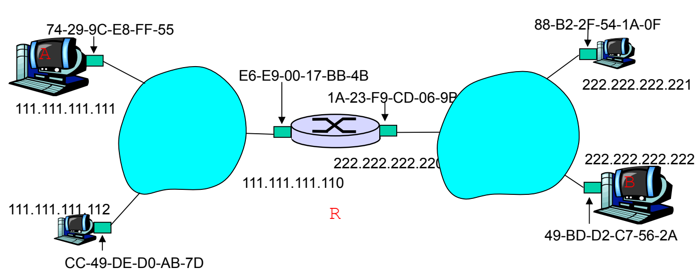
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

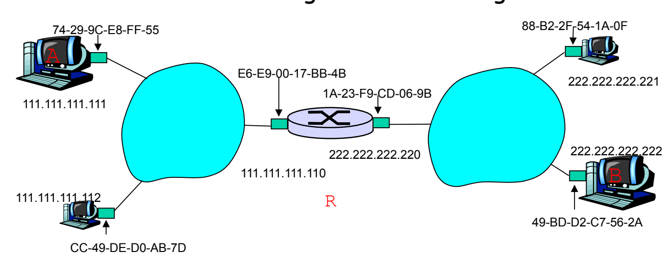
Addressing: routing to another LAN

- send datagram from A to B via R
- assume A knows B's IP address

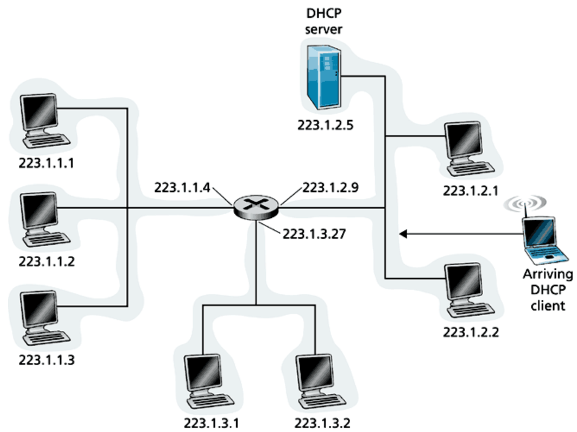


- two ARP tables in router R, one for each IP network (LAN)
- Should A address the message to B's physical address, 49-BD-D2-C7-56-2A?

- A creates IP datagram with source A, destination B
- A uses ARP to get R's MAC address for 111.111.111.110
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram
- A's NIC sends frame
- R's NIC receives frame
- R removes IP datagram from Ethernet frame, sees its destined to B
- R uses ARP to get B's MAC address
- R creates frame containing A-to-B IP datagram sends to B

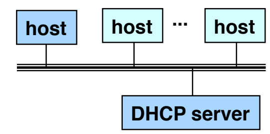


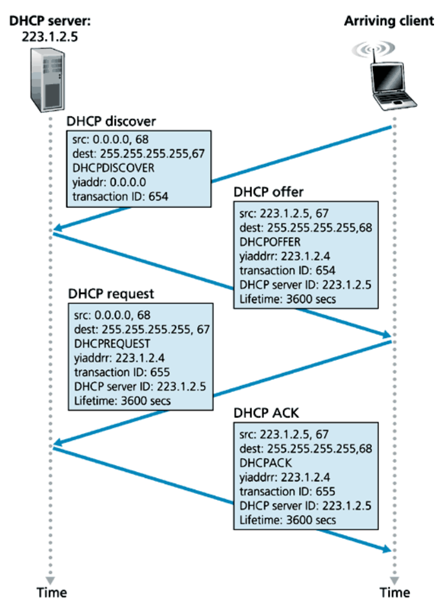
Dynamic Host Configuration Protocol (DHCP)



DHCP: Bootstrapping

- Host doesn't have an IP address yet
  - So, host doesn't know what source address to use
- Host doesn't know who to ask for an IP address
  - So, host doesn't know what destination address to use
- Solution: **Shout** to discover server who can help
  - Broadcast a server-discovery message
  - Server sends a reply offering an address





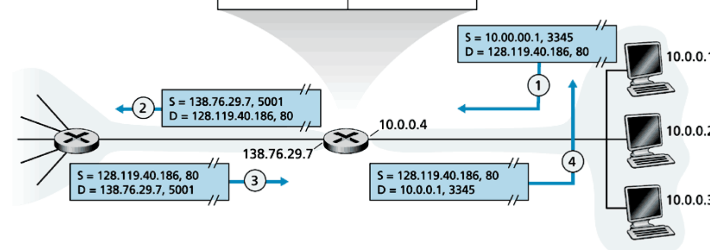
DHCP client-server interaction

Network address translation (NAT)

Used to set up small LAN network behind a single IP address (home/small business)

NAT translation table

WAN side	LAN side
138.76.29.7, 5001	10.00. 00.1, 3345
. . .	. . .

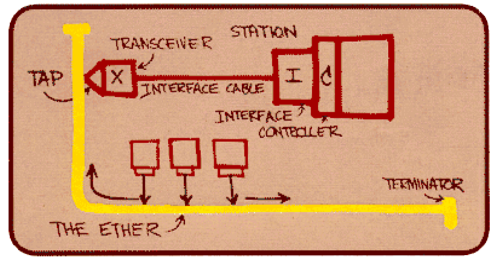


NAT problems

- Port numbers are meant for addressing processes, not for addressing hosts.
- Routers are suppose to process packets only up to layer 3.
- Nat protocol violates the so-called "end-to-end argument"; that is, hosts should be talking directly with each other, without interfering nodes modifying IP addresses and port numbers.
- Interferes with P2P applications --- peers behind a NAT cannot act as server and accept TCP connections.

Ethernet

- Invented in mid 1970s by Bob Metcalfe and David Boggs at Xerox PARC.
- Ethernet has dominated the LAN market because:
  - First LAN technology to be widely deployed.
  - Generally cheaper and simpler than its competitors (token rings, ATM, FDDI = Fiber Distributed Data Interface),
  - Always managed to maintain comparable data rates with emerging technologies: 10Mbps – 10 Gbps



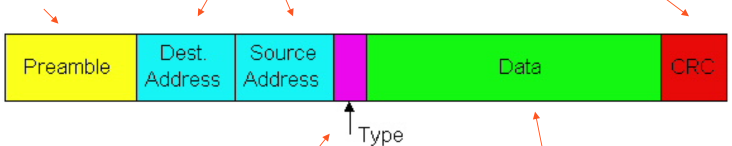
Metcalfe's Ethernet sketch

Ethernet frame structure

Seven bytes of 10101010 and one byte of 10101011; used to synchronize sender & receiver clock rates

MAC addresses, 6 bytes each; receiving adapter discards unless it matches dest. address or broadcast address (except if in permiscuous mode for sniffing!)

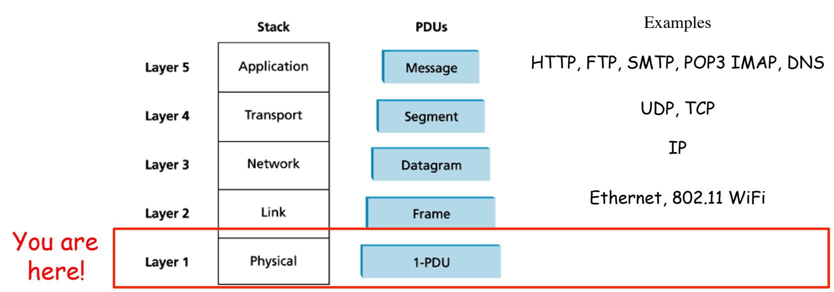
Our friend from previous lecture 4 bytes; if error detected, frame dropped



Two bytes used for multiplexed network -layer protocols; who do I pass the data up to? Usually IP, But could also be AppleTalk, Novell IPX, DecNet, …

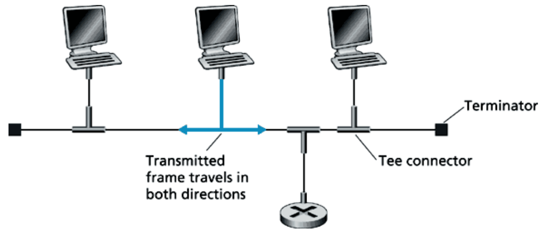
Carries IP datagram; has MTU of 1500 bytes and minimum of 46 (if less, it is stuffed)

Physical Layer



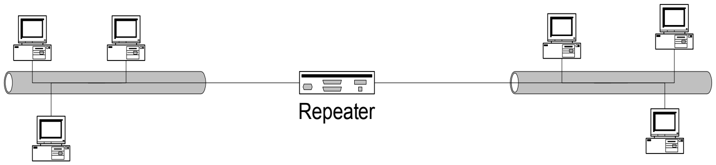
Physical Layer: Buses

- In early Ethernet implementations, nodes were "tapped into" coaxial cable
- Remained popular through mid 90s
- All nodes in same collision domain (can collide with each other)
- Limitation in bus length (often only up to 100 meters)
- Cable problems can cut off one part of network from another.



Physical Layer: Repeaters

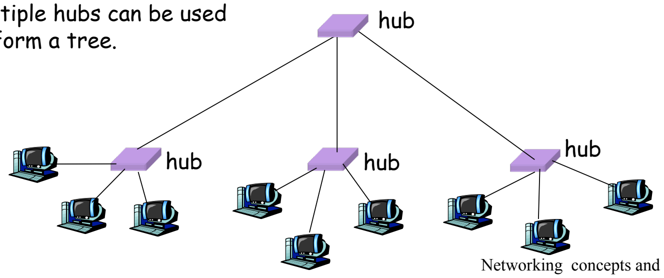
- Distance limitation in local-area networks
  - Electrical signal becomes weaker as it travels
  - Propagation delays interfere with collision detection
- Repeaters join LANs together
  - Analog electronic device
  - Continuously monitors electrical signals on each LAN
  - Transmits an amplified copy
- Example:
  - Without repeater, 10Base2 is limited to 30 nodes and 185 meters.
  - Up to four repeaters can be used to create a bus up to 925 meters.



Physical Layer: Hubs

- Hub is an unsophisticated broadcast device; when bit received on any link, broadcast it to all links at same rate.
- Often (but not always) amplifies signal, so can act like a repeater.
- Operates at the physical layer; does not examine frames or buffer them.

- Permits star topology in which each host connected separately to hub, reducing impact of wire problems.
- Multiple hubs can be used to form a tree.

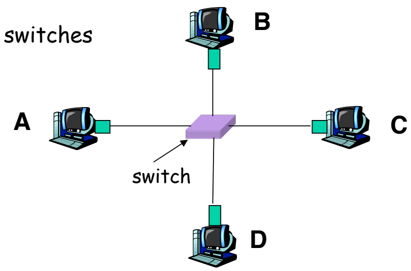


Limitations of Repeaters and Hubs

- One large collision domain
  - Every bit is sent everywhere
  - So, aggregate throughput is limited
  - E.g., three departments each get 10 Mbps independently
  - … and then connect via a hub and must share 10 Mbps
- Cannot support multiple LAN technologies
  - Does not buffer or interpret frames
  - So, can't interconnect between different rates or formats, e.g., 10 Mbps Ethernet and 100 Mbps Ethernet
- Limitations on maximum nodes and distances
  - Does not circumvent the limitations of shared media

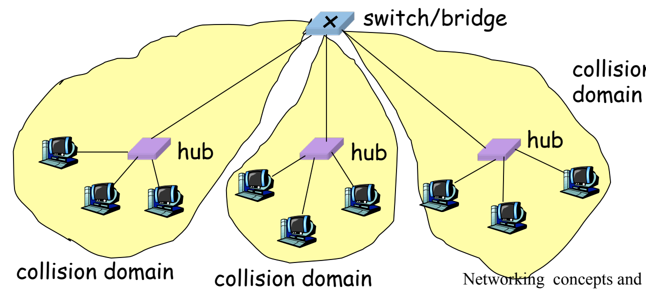
Link Layer: Switches

- Unlike "dumb" hubs, switches are smart and active,
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links
  - when frame is to be forwarded on link, uses CSMA/CD to access link
  - buffers frames, allowing links with different bandwidths
  - Also called bridges; sometimes "switch" used when connecting hosts and "bridge" used when connecting LANs.
- transparent
  - hosts are unaware of presence of switches
- concurrent communication
  - Host A can talk to C, while B talks to D, without collisions!
- plug-and-play, self-learning
  - switches do not need to be configured



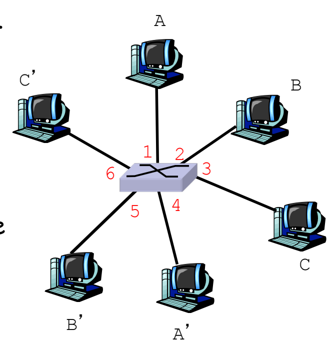
Switches: Traffic Isolation

- Breaks subnet into LAN segments
- Filters packets
  - Frame only forwarded to the necessary segments
  - Segments become separate collision domains



Switch Table

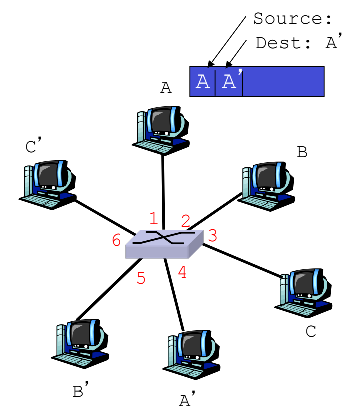
- Q: how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- A: each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- Q: how are entries created, maintained in switch table?
  - Self-learning rather than routing protocols or manual configuration.



switch with six interfaces (1,2,3,4,5,6)

Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	1	60

Switch table (initially empty)

Switch: frame filtering/forwarding

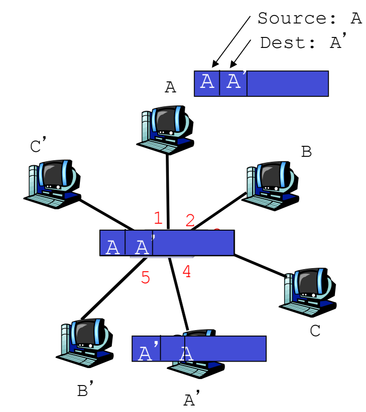
When frame received:

1. record link associated with sending host
2. index switch table using MAC dest address
3. **if** entry found for destination
   then {
   if dest on segment from which frame arrived
   then drop the frame
   else forward the frame on interface indicated
   }
   else flood

forward on all but the interface on which the frame arrived

Self-learning, forwarding: example

- frame destination unknown: flood
- destination A location known: selective send

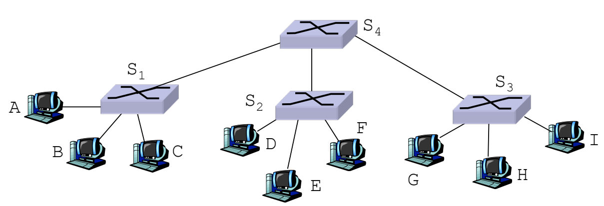


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table (initially empty)

Interconnecting switches

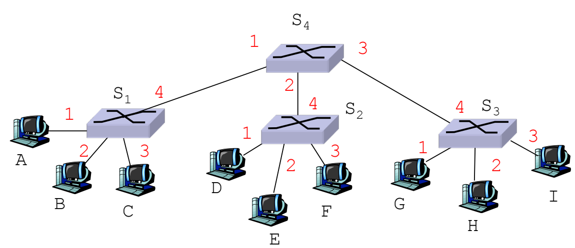
- switches can be connected together



- Q: sending from A to G - how does S_1 know to forward frame destined to F via S_4 and S_3?
- A: self learning! (works exactly the same as in single-switch case!)

Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



- Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Switches: Advantages Over Hubs/Repeaters

- Only forwards frames as needed
  - Filters frames to avoid unnecessary load on segments
  - Sends frames only to segments that need to see them
- Extends the geographic span of the network
  - Separate collision domains allow longer distances
- Improves privacy by limiting scope of frames
  - Hosts can "snoop" the traffic traversing their segment but not all the rest of the traffic
- Applies carrier sense and collision detection
  - Does not transmit when the link is busy
  - Applies exponential back-off after a collision
- Joins segments using different technologies
  - E.g., can join 10 Mbps Ethernet and 100 Mbps Ethernet

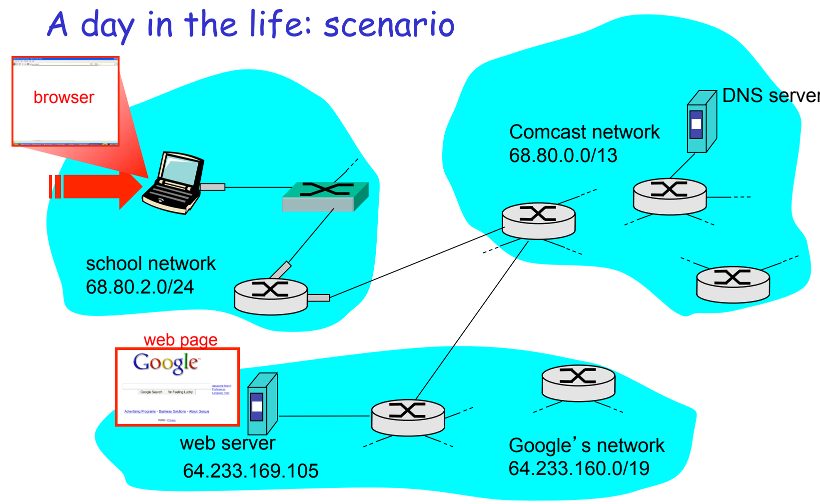
Switches: Disadvantages Over Hubs/Repeaters

- Delay in forwarding frames
  - Bridge/switch must receive and parse the frame and perform a look-up to decide where to forward
  - Storing and forwarding the packet introduces delay
- Need to learn where to forward frames
  - Bridge/switch needs to construct a forwarding table
  - Ideally, without intervention from network administrators
- Higher cost
  - More complicated devices that cost more money

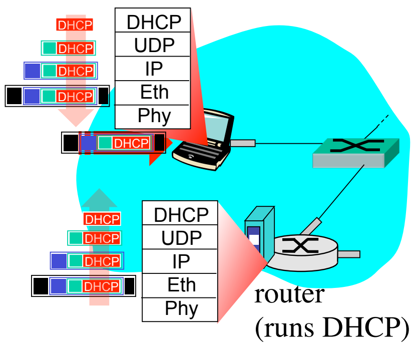
Synthesis: a day in the life of a web request

- journey down protocol stack complete!
  - application, transport, network, link
- putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

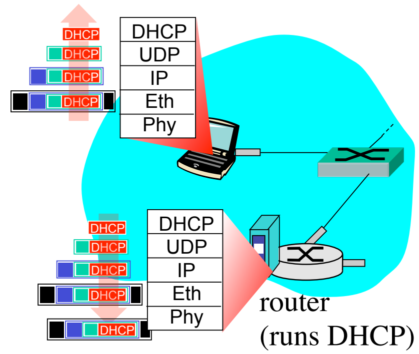


A day in the life... connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use ***DHCP***
- DHCP request **encapsulated** in **UDP**, encapsulated in **IP**, encapsulated in **802.1** Ethernet
- Ethernet frame **broadcast** (dest: FFFFFFFFFFFF) on LAN, received at router running **DHCP** server
- Ethernet **demux'ed** to IP demux'ed, UDP demux'ed to DHCP

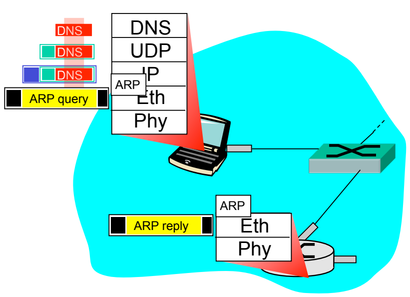
A day in the life... connecting to the Internet



- DHCP server formulates ***DHCP ACK*** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded (**switch learning**) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

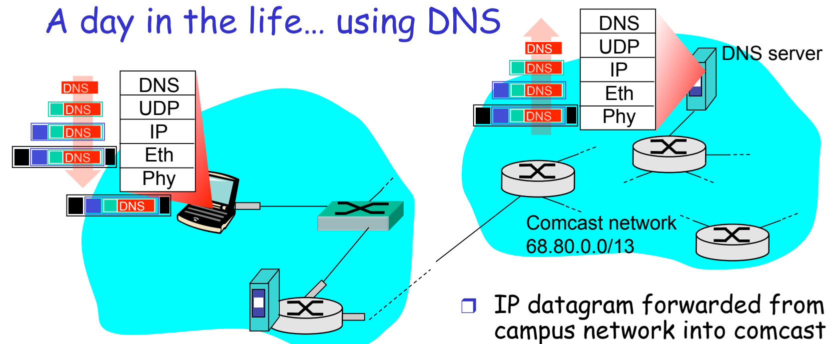
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life... ARP (before DNS, before HTTP)



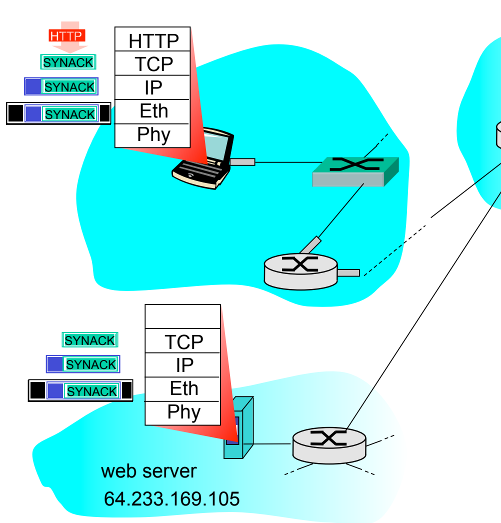
- before sending ***HTTP*** request, need IP address of www.google.com: ***DNS***
- DNS query created, encapsulated in UDP, encapsulated in IP, encasulated in Eth. In order to send frame to router, need MAC address of router interface: ***ARP***
- ***ARP query*** broadcast, received by router, which replies with ***ARP reply*** giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life... using DNS



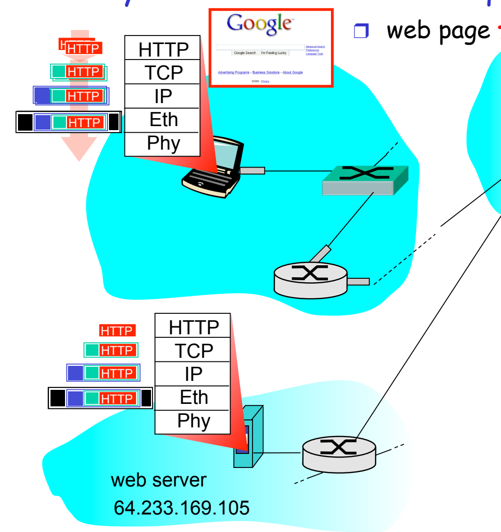
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed (tables created by **RIP**, **OSPF**, **IS-IS** and/or **BGP** routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life... TCP connection carrying HTTP



- to send HTTP request, client first opens **TCP socket** to web server
- TCP **SYN segment** (step 1 in 3-way handshake) **inter-domain routed** to web server
- web server responds with **TCP SYNACK** (step 2 in 3-way handshake)
- TCP **connection established!**

A day in the life... HTTP request/reply



- web page **finally (!!!)** displayed
- **HTTP request** sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with **HTTP reply** (containing web page)
- IP datgram containing HTTP reply routed back to client