ECE4574 – Large-Scale SW Development for Engineering Systems
Lecture 6 – Review of Networks

Creed Jones, PhD

BRADLEY DEPARTMENT
OF ELECTRICAL & COMPUTER ENGINEERING

VIRGINIA TECH.

Course Updates

- Your project proposal is due TODAY, 11:59 PM
  - It's a group assignment – only one person (the Scrum Master) need submit it
- Homework 1 is due Friday, September 22
- Quiz 2 is TODAY, September 11
  - 7 PM to 1 AM Eastern time
  - 20 minute time limit
  - open notes, no help from anyone else
  - covers lectures 3 and 4
- No office hours on Tuesday, Sept 12
  - schedule conflict

Today's Objectives

Brief review of networking

- The layered model
- Application layer
  - HTTP
  - port numbers
- Transport layer
- Network layer
- Link and physical layers

A Network is a connection between computers that can be used to exchange data; network interactions are defined in *protocols*

Standard – not custom
Multi-use – not dedicated to one app

Packet-based – not byte by byte
Multiple connections – not one-to-one

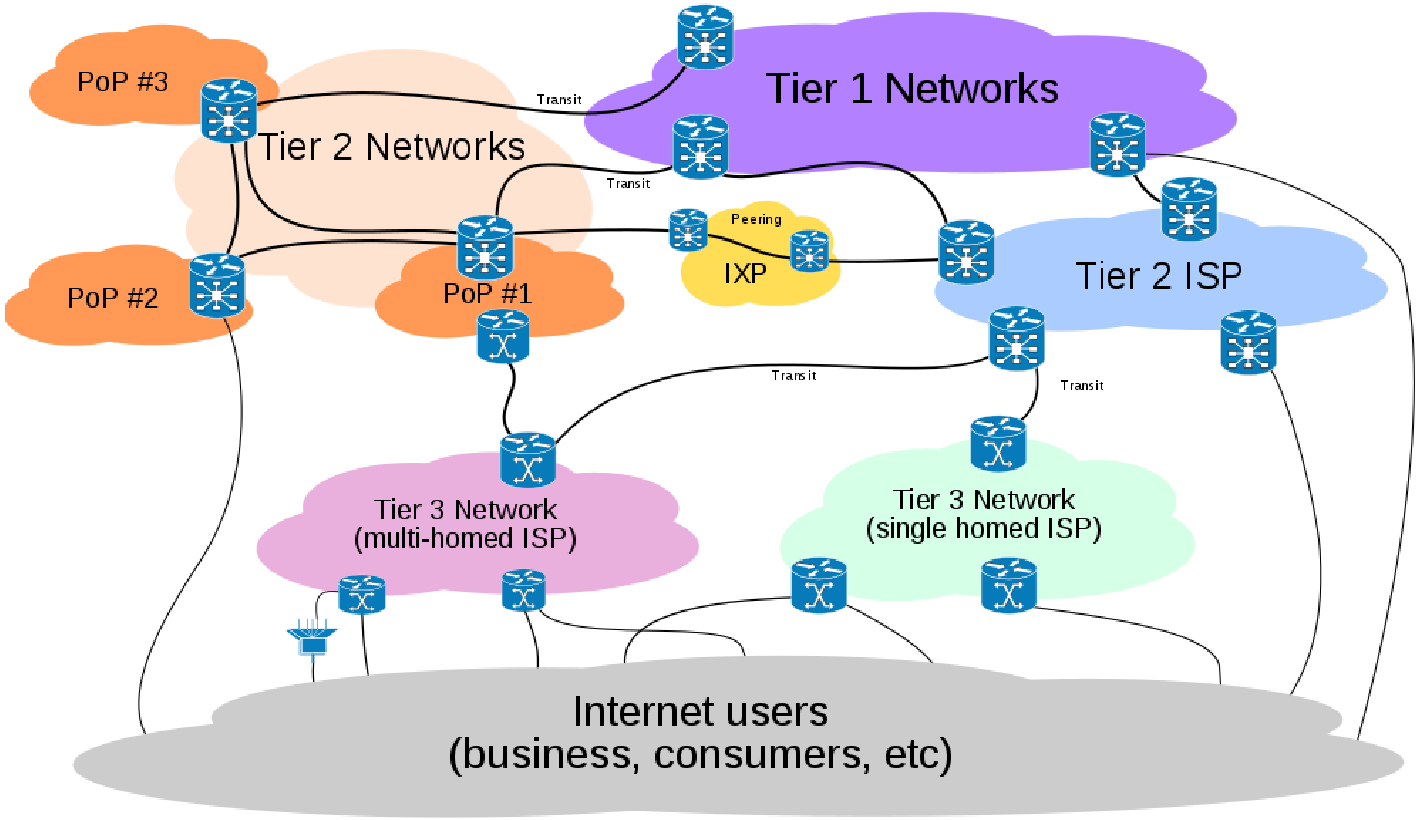
- A protocol is set of rules governing the exchange of data between two entities
  - used for communication between entities that can exchange information
  - for two entities to communicate successfully they must "speak the same language"
- Key elements of a protocol are:

Syntax	Semantics	Timing
• includes such things as data format and signal levels	• includes control information for coordination and error handling	• includes speed matching and sequencing

A network can be divided into systems at the edge and the core; the core exists to provide connections between nodes on the edge

- This is the ***end-to-end network principle***
  - The application is characterized by what happens in system on the edge
  - Core systems just support message passing

- network edge: applications and hosts
- network core:
  - routers
  - gateways
  - network of networks
- access networks, physical media: communication links



We can connect edge nodes in several ways: for example, TCP is a connection-oriented service

Goal: data transfer between end systems
- *handshaking:* setup (prepare for) data transfer ahead of time
  - Hello, hello back human protocol
  - *set up "state"* in two communicating hosts
- TCP - Transmission Control Protocol
  - Internet's connection-oriented service

TCP service [RFC 793]
- *reliable, in-order* byte-stream data transfer
  - loss: acknowledgements and retransmissions
- *flow control:*
  - sender won't overwhelm receiver
- *congestion control:*
  - senders "slow down sending rate" when network congested

While TCP is based on establishing and using a connection, UDP is a connectionless service

Goal: data transfer between end systems

- same as before!

- UDP - User Datagram Protocol [RFC 768]:
  - connectionless
  - unreliable data transfer
  - no flow control
  - no congestion control

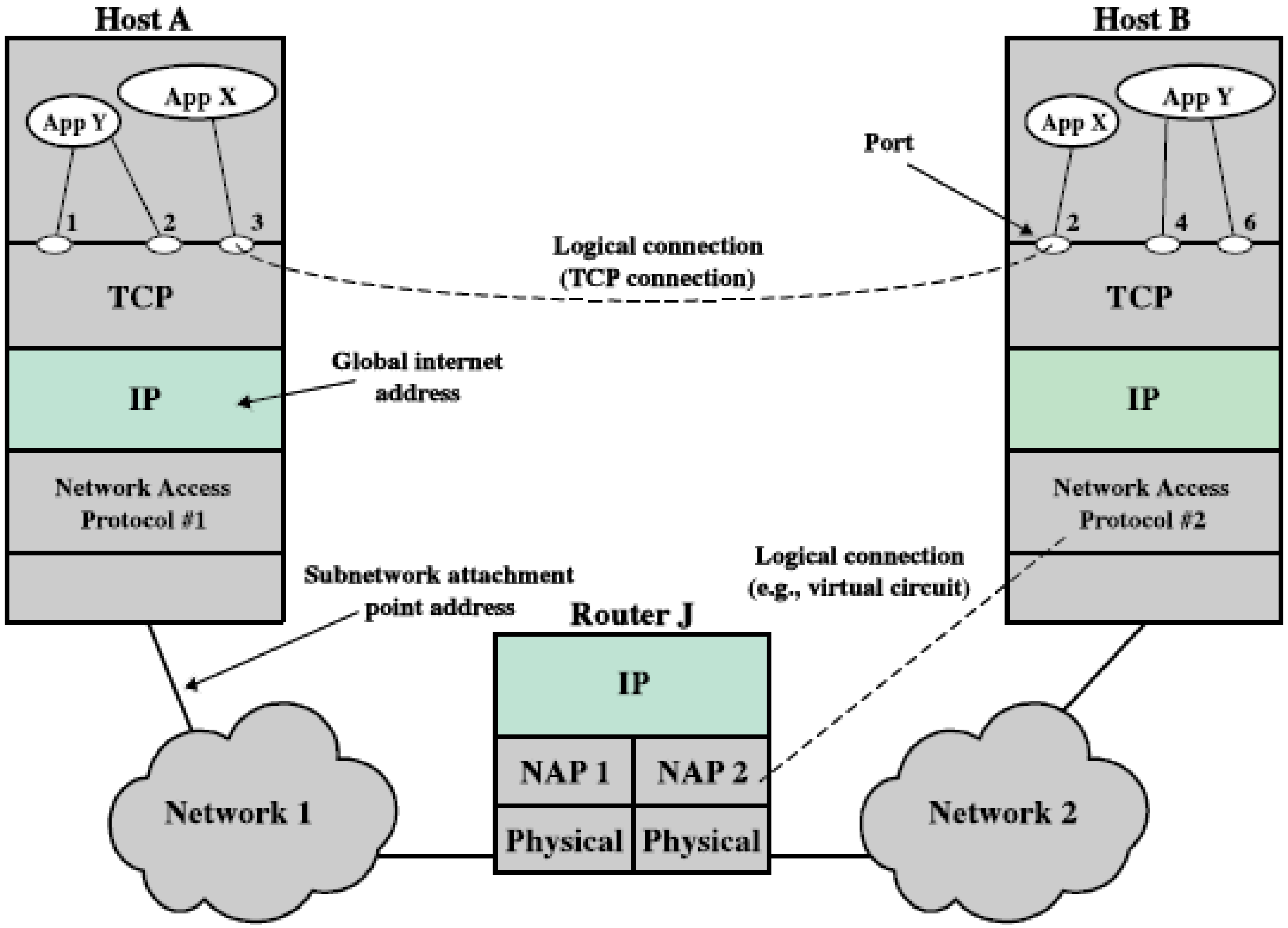
Apps using TCP:

- HTTP/HTTPS (Web), FTP (file transfer), Telnet (remote login), SMTP/IMAP (email)

Apps using UDP:

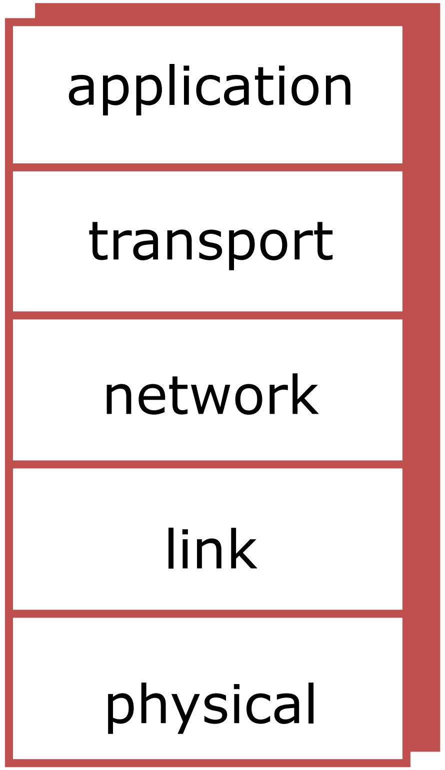
- streaming media, teleconferencing, DNS, Internet telephony

Network Protocols are <u>Layered</u>; each layer handles certain tasks and connects with layers above and below

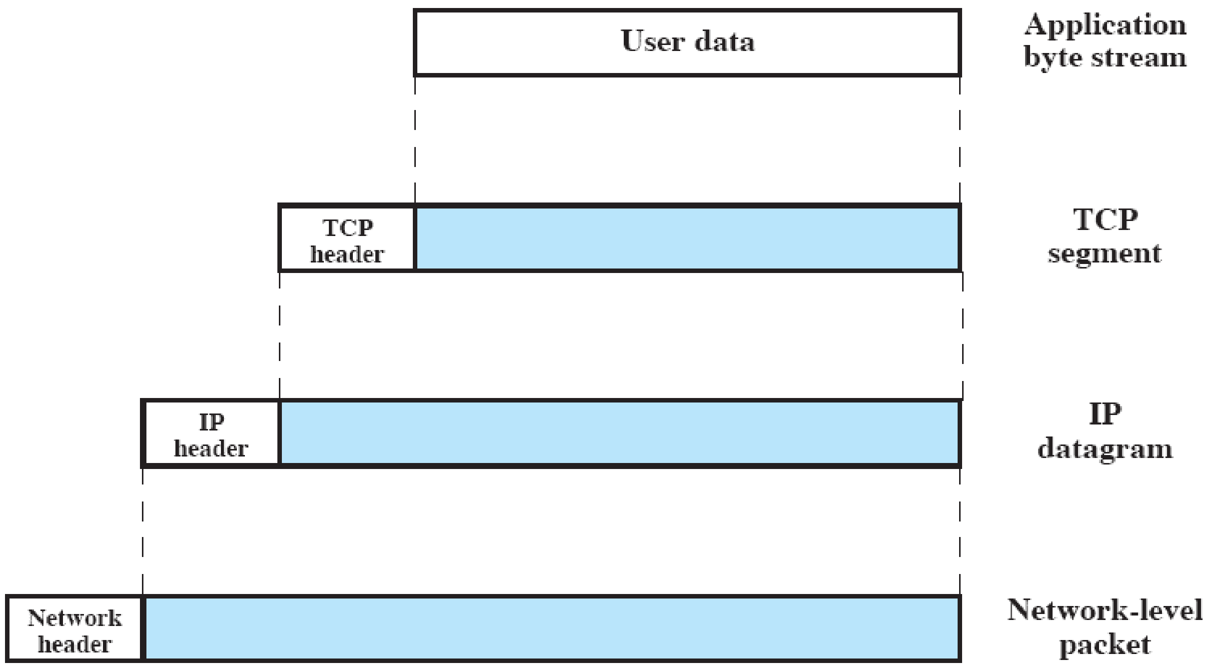


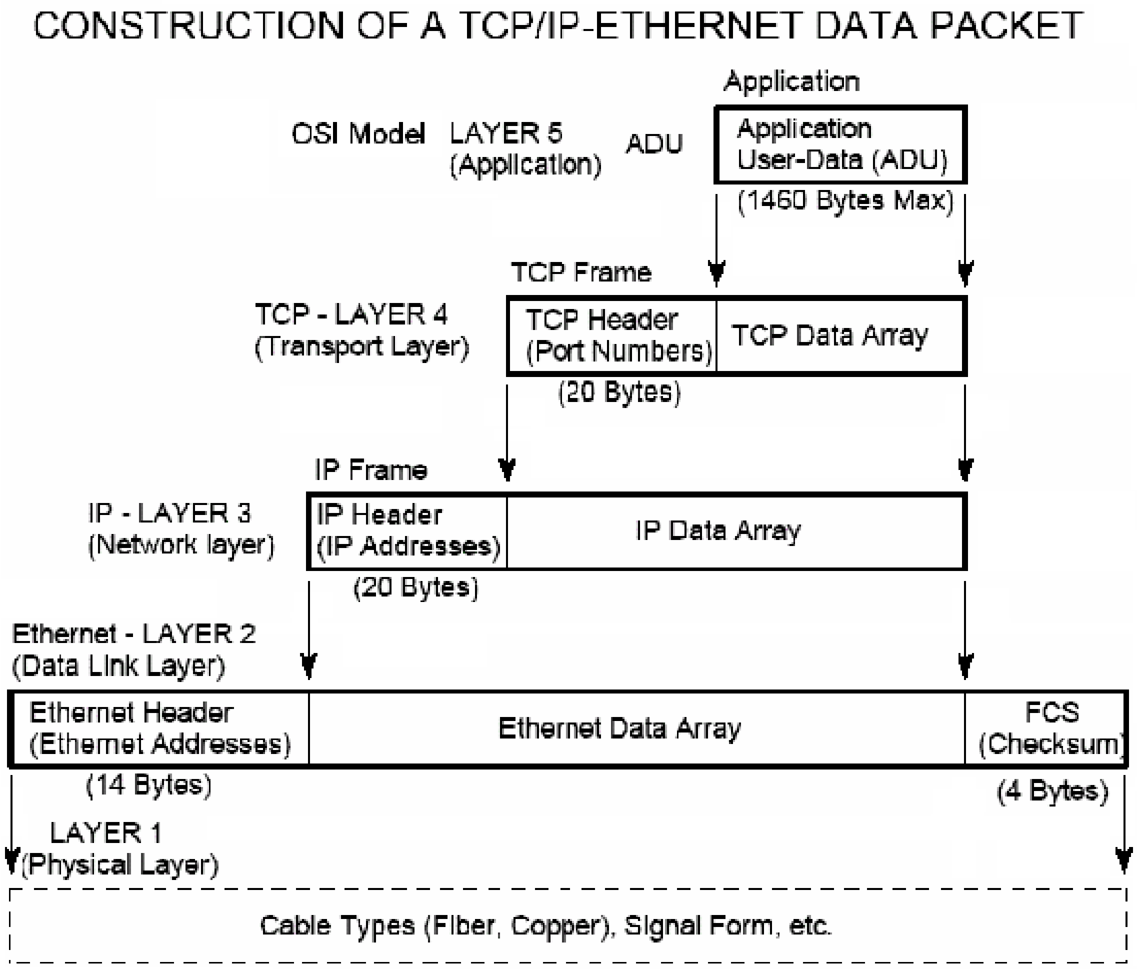
The Internet protocol stack consists of layers

- application: supporting network applications
  - FTP, SMTP, HTTP
- transport: host-host data transfer
  - TCP, UDP
- network: routing of datagrams from source to destination
  - IP, routing protocols
- link: data transfer between neighboring network elements
  - PPP, Ethernet
- physical: bits on a cable, on Wifi or cell network

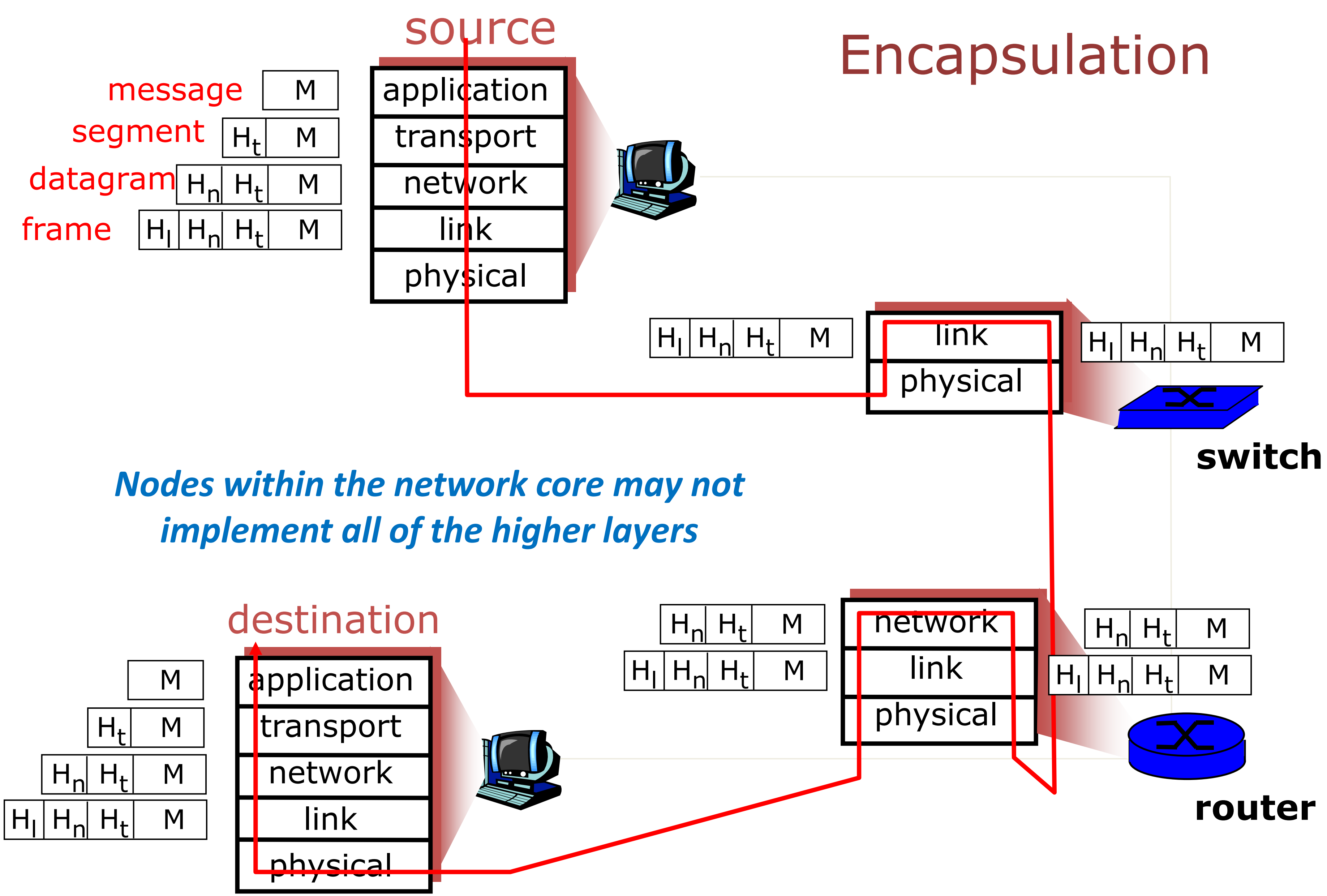


Data is encapsulated by lower layers



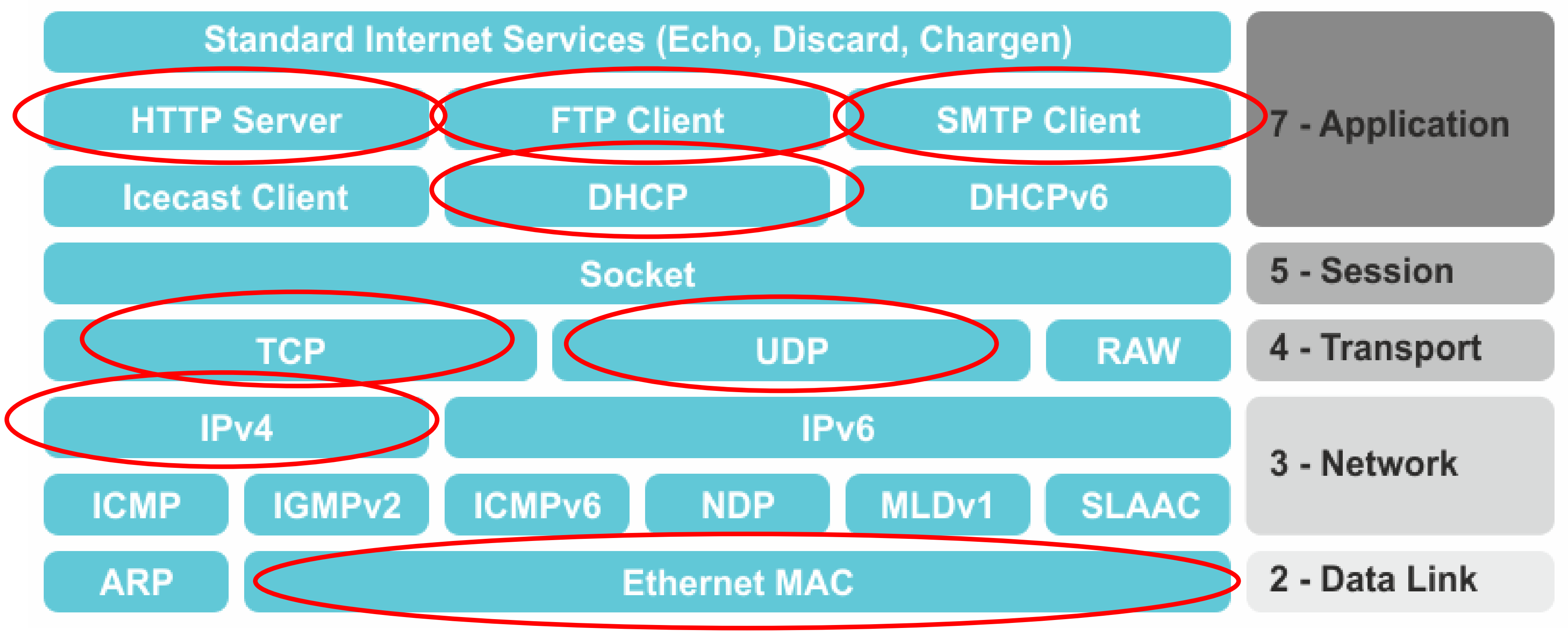


Encapsulation



Nodes within the network core may not implement all of the higher layers

Many protocols exist at most levels of the TCP/IP Protocol stack: these are the most important ones



Internet protocols are defined in documents called RFCs (Request for Comments)

VIRGINIA TECH.

BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

- These standard definitions of the protocols are not laid down by some authority, in most cases
- They were developed and are followed by common consent
- Initial RFCs were issued and changes made
  - When the changes settled down, the RFC was considered the defining document
- You can see them at http://www.rfc-editor.org/rfc.html
  - Search for "HTTP" – the key docs are RFC 1945, RFC 7230 and RFC 2660

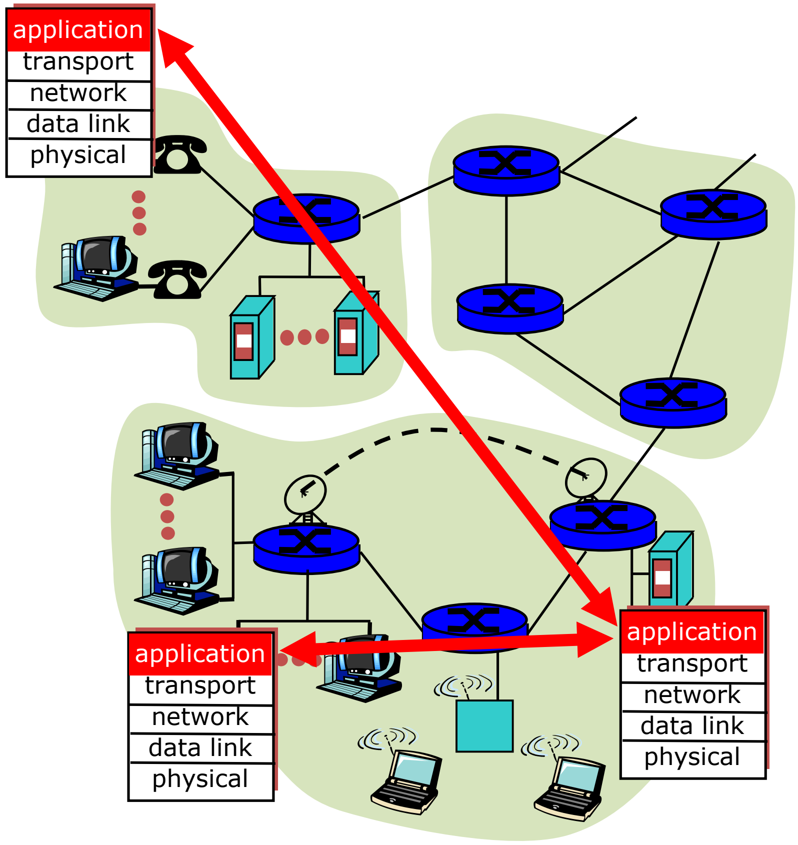
Creating a network application

Write programs that

- run on different end systems and
- communicate over a network.
- e.g., Web: Web server software communicates with browser software

Less software is written for devices in network core

- network core devices do not run user application code
- application on end systems allows for rapid app development, propagation

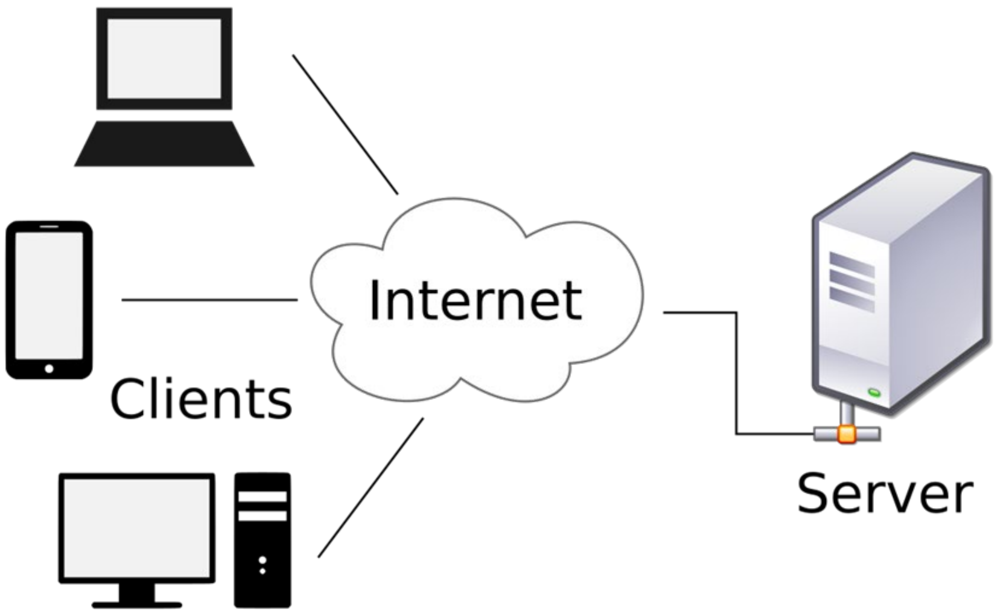


APPLICATION LAYER

Client-server architecture

server:
- always-on host
- often, permanent IP address
- server farms for scaling

clients:
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

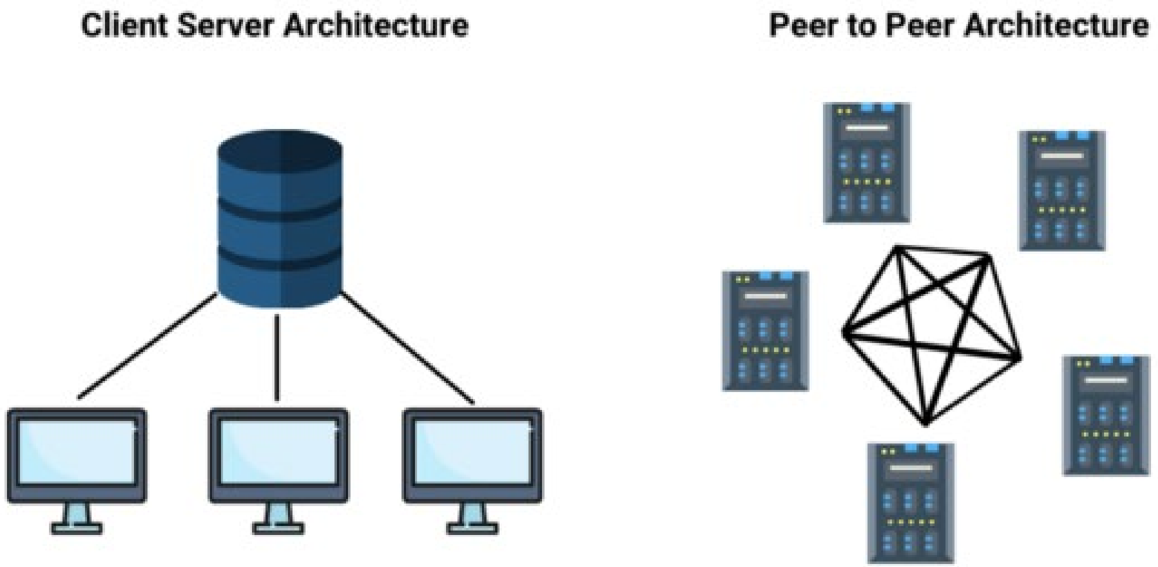


Pure Peer to Peer (P2P) architecture

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses
- examples: BitTorrent and Akamai

Highly scalable

But difficult to manage



Hybrid of client-server and P2P

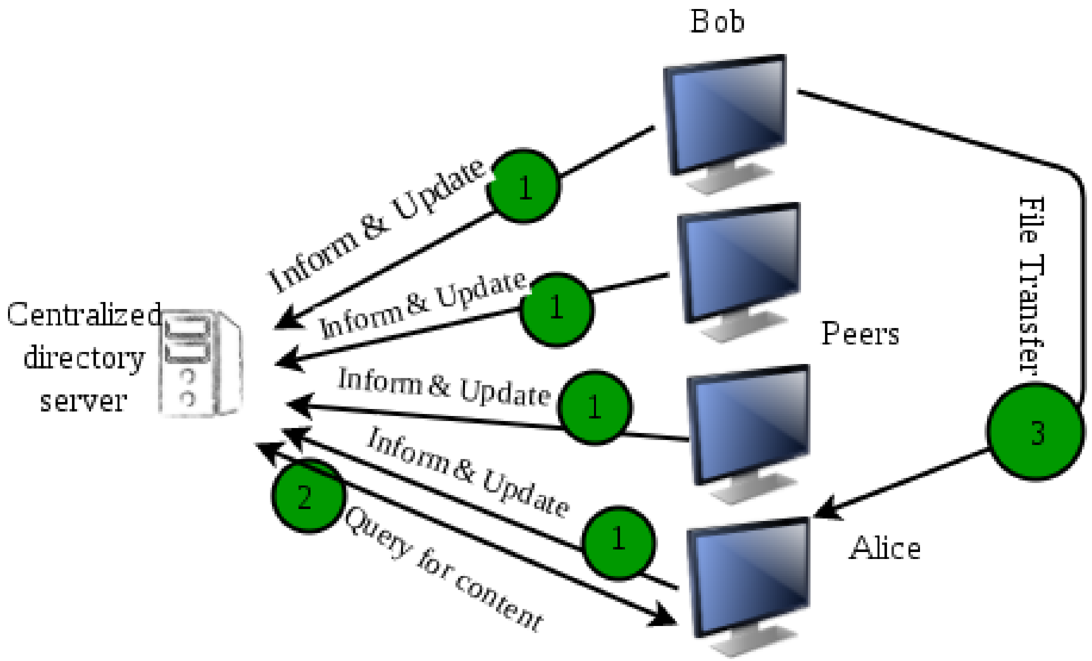
Instant messaging

- Chatting between two users can be P2P
- Presence detection/location centralized:
  - User registers its IP address with central server when it comes online
  - User contacts central server to find IP addresses of buddies

File sharing

- Linux distribution

Bitcoin



P2P paradigm with a centralised directory

Processes communicating

Process: program running within a host.

- within same host, two processes communicate using inter-process communication (defined by OS).
- processes in different hosts communicate by exchanging messages

Client process: process that initiates communication

Server process: process that waits to be contacted

- Note: applications with P2P architectures have client processes & server processes

Addressing processes uses both the IP address and the *port number*

- Identifier includes both the IP address and port numbers associated with the process on the host.
- Example port numbers:
  - HTTP server: 80
  - Mail server: 25
- In programs, a network *socket* is opened on a particular address and port number

- For a process to receive messages, it must have an identifier
- A host has a unique32-bit IP address
- Q: does the IP address of the host on which the process runs suffice for identifying the process?
- Answer: No, many processes can be running on same host

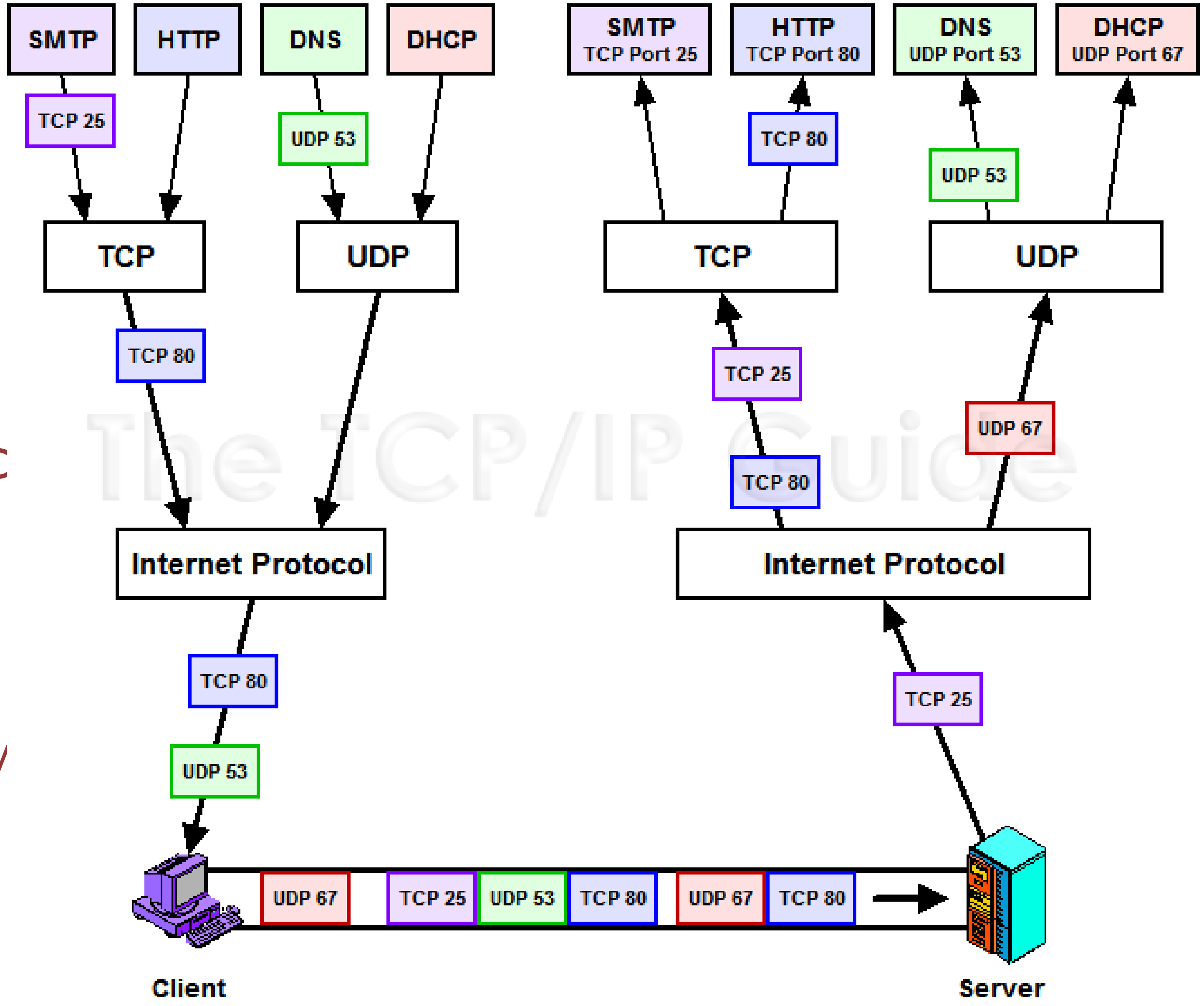
PORT NUMBERS

Recall that port numbers are used to route incoming network traffic to the right application

- The IP address (derived from the URL using the Domain Name Service, DNS) will get traffic to the right host
- On that host, different applications are looking for traffic that carries the proper port number
- More specifically, a network *socket* is created by the application code on the destination, and is associated with a single port number and destination IP address
- Many port numbers are universally connected with certain types of traffic
  - these are called the well-known port numbers

Port numbers are processed by the transport layer (the IP implementation) and allow direction of traffic to and from different processes on the hosts

- from http://www.tcpipguide.com/



The "well-known" port numbers are universally agreed to route traffic to specific applications – NOTE that the application can still split data up by other means!

PORT NUMBER	TRANSPORT PROTOCOL	SERVICE NAME	RFC
20, 21	TCP	File Transfer Protocol (FTP)	RFC 959
22	TCP and UDP	Secure Shell (SSH)	RFC 4250-4256
23	TCP	Telnet	RFC 854
25	TCP	Simple Mail Transfer Protocol (SMTP)	RFC 5321
53	TCP and UDP	Domain Name Server (DNS)	RFC 1034-1035
67, 68	UDP	Dynamic Host Configuration Protocol (DHCP)	RFC 2131
69	UDP	Trivial File Transfer Protocol (TFTP)	RFC 1350
80	TCP	HyperText Transfer Protocol (HTTP)	RFC 2616
110	TCP	Post Office Protocol (POP3)	RFC 1939
119	TCP	Network News Transport Protocol (NNTP)	RFC 8977
123	UDP	Network Time Protocol (NTP)	RFC 5905
135-139	TCP and UDP	NetBIOS	RFC 1001-1002
143	TCP and UDP	Internet Message Access Protocol (IMAP4)	RFC 3501
161, 162	TCP and UDP	Simple Network Management Protocol (SNMP)	RFC 1901-1908, 3411-3418
179	TCP	Border Gateway Protocol (BGP)	RFC 4271
389	TCP and UDP	Lightweight Directory Access Protocol	RFC 4510
443	TCP and UDP	HTTP with Secure Sockets Layer (SSL)	RFC 2818
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) / Internet Key Exchange (IKE)	RFC 2408 - 2409
636	TCP and UDP	Lightweight Directory Access Protocol over TLS/SSL (LDAPS	RFC 4513
989/990	TCP	FTP over TLS/SSL	RFC 4217
			https://ipwithease.com

7 Echo	**554** RTSP	**2745** Bagle.H	**6891-6901** Windows Live
19 Chargen	**546-547** DHCPv6	**2967** Symantec AV	**6970** Quicktime
20-21 FTP	**560** rmonitor	**3050** Interbase DB	**7212** GhostSurf
22 SSH/SCP	**563** NNTP over SSL	**3074** XBOX Live	**7648-7649** CU-SeeMe
23 Telnet	**587** SMTP	**3124** HTTP Proxy	**8000** Internet Radio
25 SMTP	**591** FileMaker	**3127** MyDoom	**8080** HTTP Proxy
42 WINS Replication	**593** Microsoft DCOM	**3128** HTTP Proxy	**8086-8087** Kaspersky AV
43 WHOIS	**631** Internet Printing	**3222** GLBP	**8118** Privoxy
49 TACACS	**636** LDAP over SSL	**3260** iSCSI Target	**8200** VMware Server
53 DNS	**639** MSDP (PIM)	**3306** MySQL	**8500** Adobe ColdFusion
67-68 DHCP/BOOTP	**646** LDP (MPLS)	**3389** Terminal Server	**8767** TeamSpeak
69 TFTP	**691** MS Exchange	**3689** iTunes	**8866** Bagle.B
70 Gopher	**860** iSCSI	**3690** Subversion	**9100** HP JetDirect
79 Finger	**873** rsync	**3724** World of Warcraft	**9101-9103** Bacula
80 HTTP	**902** VMware Server	**3784-3785** Ventrilo	**9119** MXit
88 Kerberos	**989-990** FTP over SSL	**4333** mSQL	**9800** WebDAV
102 MS Exchange	**993** IMAP4 over SSL	**4444** Blaster	**9898** Dabber
110 POP3	**995** POP3 over SSL	**4664** Google Desktop	**9988** Rbot/Spybot
113 Ident	**1025** Microsoft RPC	**4672** eMule	**9999** Urchin
119 NNTP (Usenet)	**1026-1029** Windows Messenger	**4899** Radmin	**10000** Webmin
123 NTP	**1080** SOCKS Proxy	**5000** UPnP	**10000** BackupExec
135 Microsoft RPC	**1080** MyDoom	**5001** Slingbox	**10113-10116** NetIQ
137-139 NetBIOS	**1194** OpenVPN	**5001** iperf	**11371** OpenPGP
143 IMAP4	**1214** Kazaa	**5004-5005** RTP	**12035-12036** Second Life
161-162 SNMP	**1241** Nessus	**5050** Yahoo! Messenger	**12345** NetBus
177 XDMCP	**1311** Dell OpenManage	**5060** SIP	**13720-13721** NetBackup
179 BGP	**1337** WASTE	**5190** AIM/ICQ	**14567** Battlefield
201 AppleTalk	**1433-1434** Microsoft SQL	**5222-5223** XMPP/Jabber	**15118** Dipnet/Oddbob
264 BGMP	**1512** WINS	**5432** PostgreSQL	**19226** AdminSecure
318 TSP	**1589** Cisco VQP	**5500** VNC Server	**19638** Ensim
381-383 HP Openview	**1701** L2TP	**5554** Sasser	**20000** Usermin
389 LDAP	**1723** MS PPTP	**5631-5632** pcAnywhere	**24800** Synergy
411-412 Direct Connect	**1725** Steam	**5800** VNC over HTTP	**25999** Xfire
443 HTTP over SSL	**1741** CiscoWorks 2000	**5900+** VNC Server	**27015** Half-Life
445 Microsoft DS	**1755** MS Media Server	**6000-6001** X11	**27374** Sub7
464 Kerberos	**1812-1813** RADIUS	**6112** Battle.net	**28960** Call of Duty
465 SMTP over SSL	**1863** MSN	**6129** DameWare	**31337** Back Orifice
497 Retrospect	**1985** Cisco HSRP	**6257** WinMX	**33434+** traceroute
500 ISAKMP	**2000** Cisco SCCP	**6346-6347** Gnutella	**Legends**
512 rexec	**2002** Cisco ACS	**6500** GameSpy Arcade	
513 rlogin	**2049** NFS	**6566** SANE	Chat
514 syslog	**2082-2083** cPanel	**6588** AnalogX	Encrypted
515 LPD/LPR	**2100** Oracle XDB	**6665-6669** IRC	Gaming
520 RIP	**2222** DirectAdmin	**6679/6697** IRC over SSL	Malicious
521 RIPng (IPv6)	**2302** Halo	**6699** Napster	Peer to Peer
540 UUCP	**2483-2484** Oracle DB	**6881-6999** BitTorrent	Streaming

This PDF is available at: https://www.stationx.net/common-ports-cheat-sheet/

VIRGINIA TECH

BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Here is a partial list of port numbers – note that these include:

- protocols
- applications
- OS utilities
- comm (VOIP)
- games

App-layer protocol defines

- Types of messages exchanged, e.g., request & response messages
- Syntax of message types: what fields in messages & how fields are delineated
- Semantics of the fields, i.e., meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

Proprietary protocols:

- e.g., Skype

What transport service does an app need?

Data loss

- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Bandwidth

- some apps (e.g., multimedia) require minimum amount of bandwidth to be "effective"
- other apps ("elastic apps") make use of whatever bandwidth they get

Timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Transport service requirements of common apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Internet transport protocols services

TCP service:

- *connection-oriented:* 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 bandwidth guarantees

UDP service:

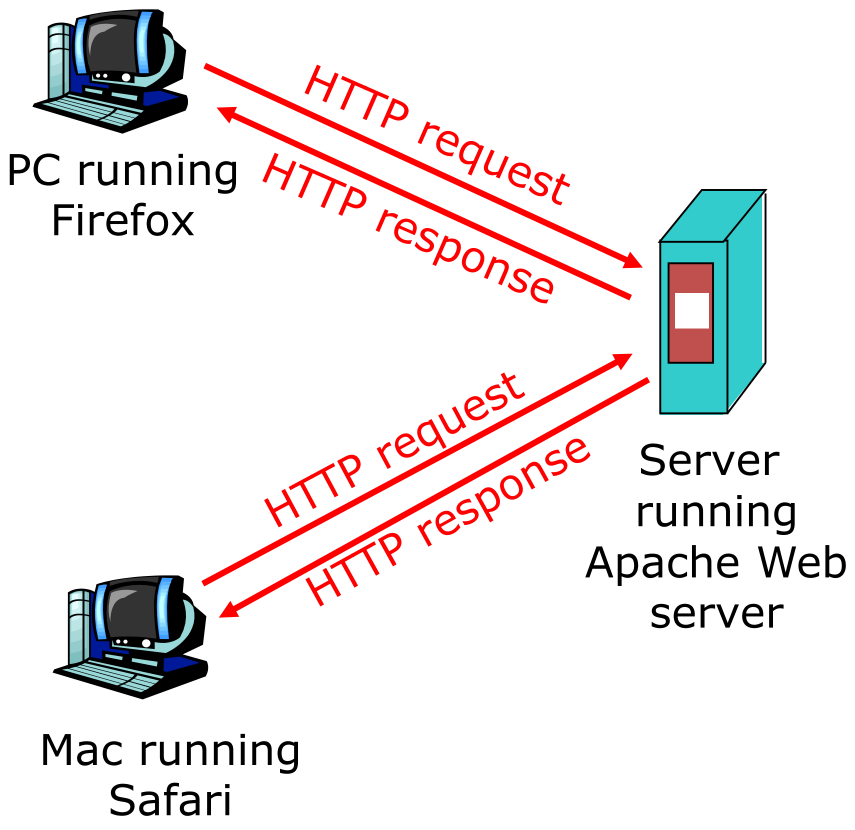
- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee

Internet apps: application, transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP [RFC 793]
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 7540]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Vonage,Dialpad)	typically UDP

HTTP – the hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests
- HTTP 1.1: [RFC 2330]
- HTTP 2.0: [RFC 7540]



HTTP overview (continued)

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests

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

HTTP request message

- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII text (human-readable format)

request line (GET, POST, HEAD commands)

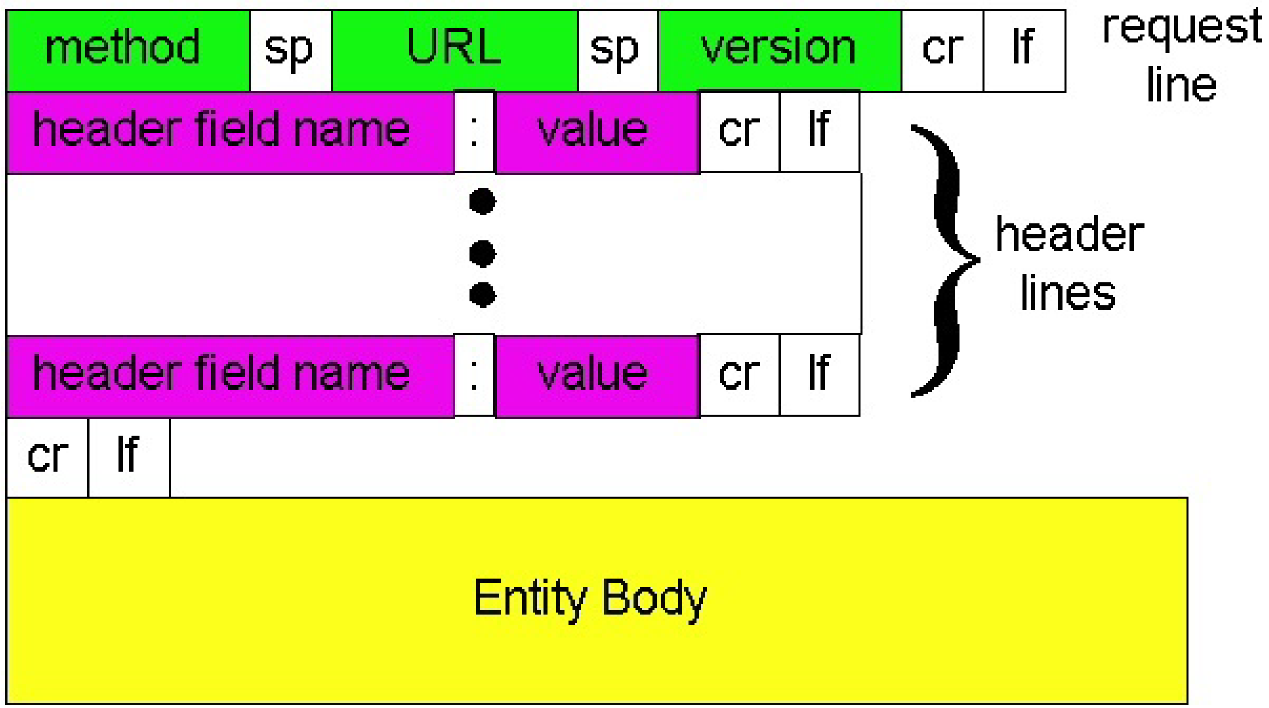
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

header lines

(extra carriage return, line feed)

Carriage return, line feed indicates end of message

HTTP request message: general format




```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

Clients can specify various operations, and provide data to servers, in several ways

HTTP supports several message types

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

Post method of data upload:

- Web page often includes form input
- Input is uploaded to server in entity body

URL method of data upload:

- Uses GET method
- Input is uploaded in URL field of request line:
- `www.somesite.com/animalsearch?monkeys&banana`

HTTP response message

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

HTTP response status codes

In first line in server->client response message.

A few sample codes:

`200 OK`
- request succeeded, requested object later in this message

`301 Moved Permanently`
- requested object moved, new location specified later in this message (Location:)

`400 Bad Request`
- request message not understood by server

`404 Not Found`
- requested document not found on this server

`505 HTTP Version Not Supported`

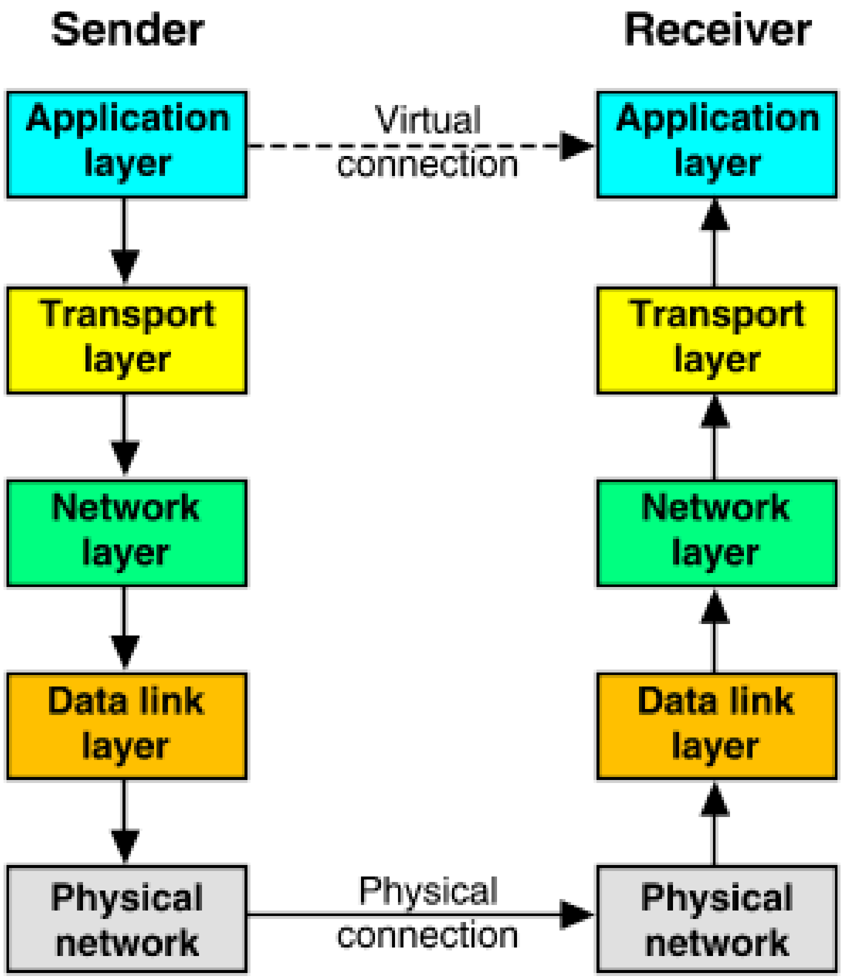
HTTP Status Codes

Level 200 (Success)	Level 400	Level 500
200 : OK	400 : Bad Request	500 : Internal Server Error
201 : Created	401 : Unauthorized	503 : Service Unavailable
203 : Non-Authoritative Information	403 : Forbidden	501 : Not Implemented
204 : No Content	404 : Not Found	504 : Gateway Timeout
	409 : Conflict	599 : Network timeout
		502 : Bad Gateway

THE TRANSPORT LAYER

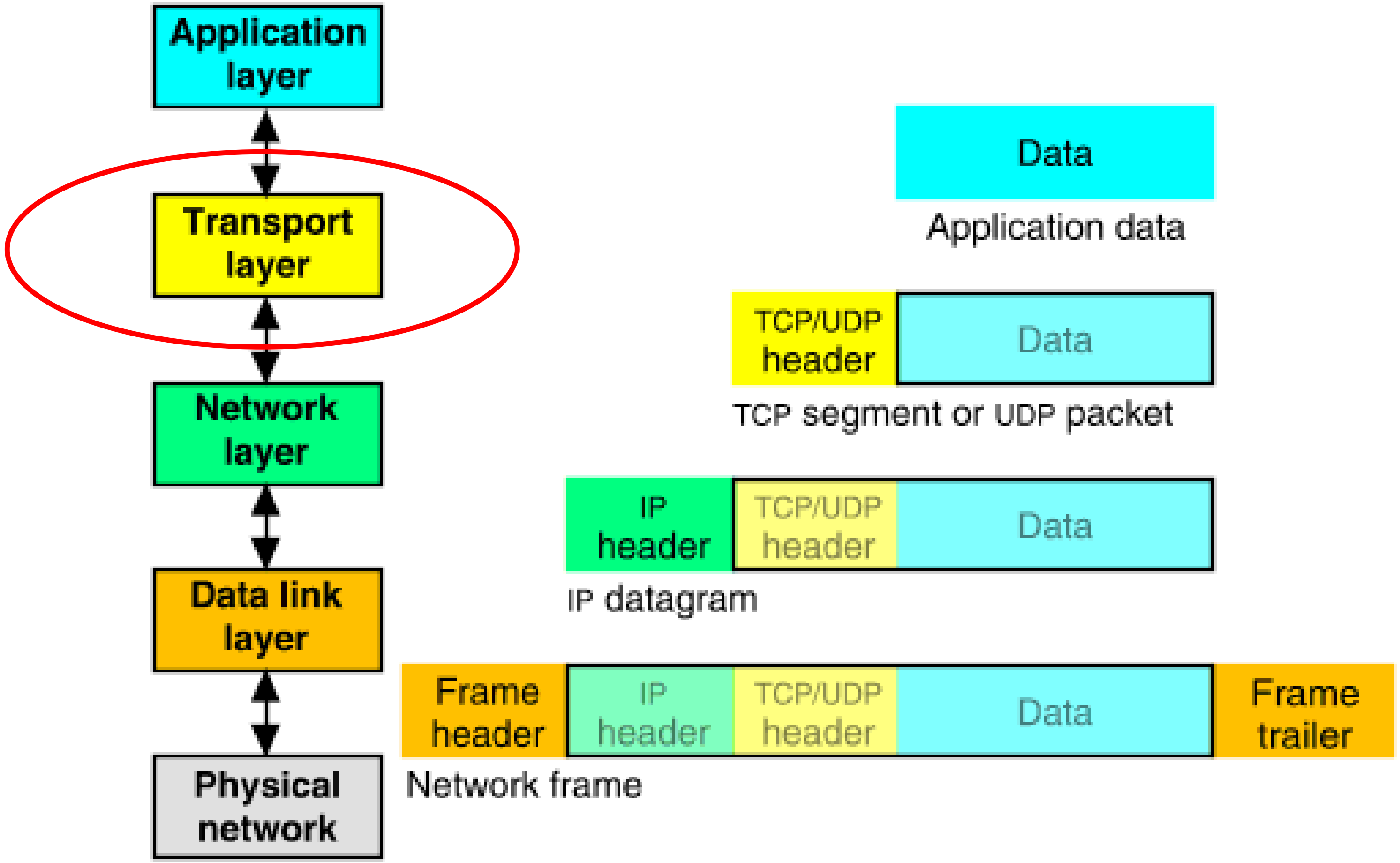
TCP/IP protocol stack

- Recall the nature of the TCP/IP network stack
  - An example of any layered set of protocols
- Each layer consists of a <u>protocol</u>
  - a set of messages and data formats
- Data is <u>encapsulated</u> as it passes between layers



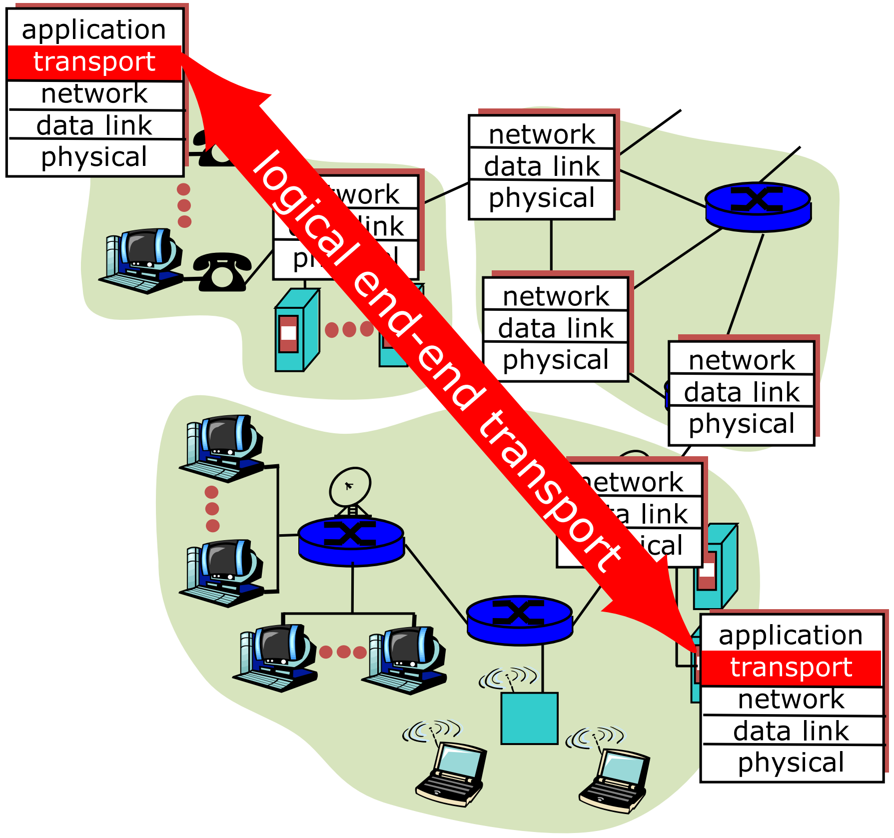
from http://uw713doc.sco.com/en/NET_tcpip/graphics/encapsulation.gif

TCP or UDP work at the transport layer



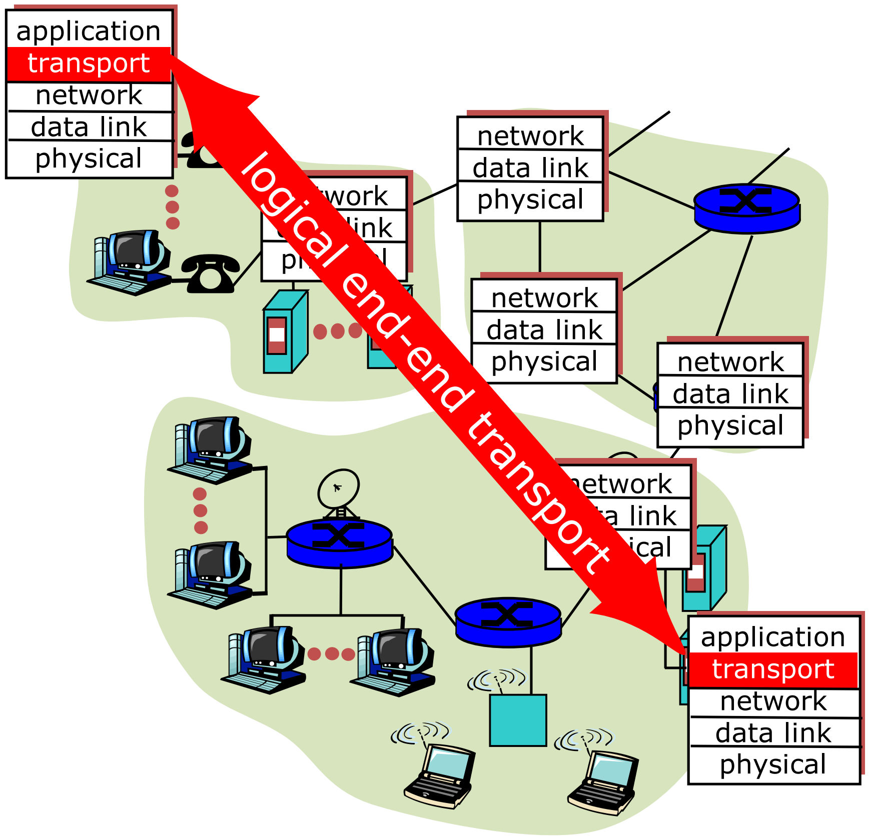
Transport services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - no-frills extension of "best-effort" IP
- services not available:
  - delay guarantees
  - bandwidth guarantees



UDP: User Datagram Protocol [RFC 768]

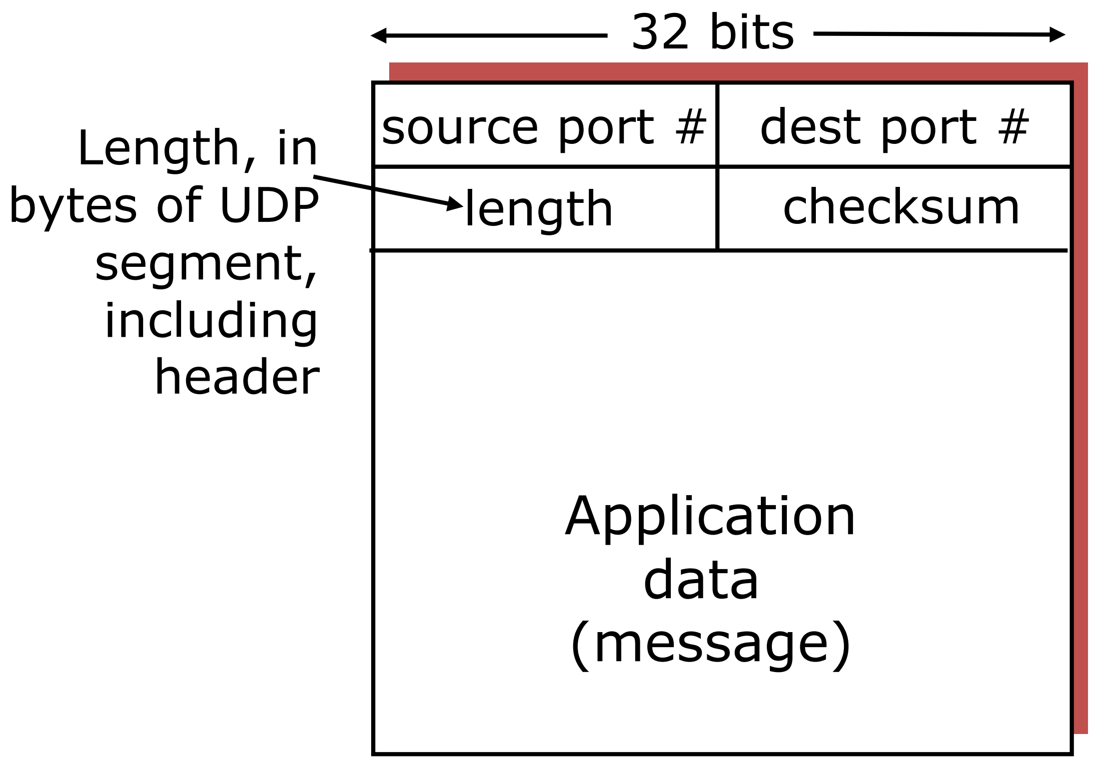
- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - lost
  - delivered out of order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

Why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired

User Datagram Protocol

- often used for streaming multimedia apps
  - loss tolerant
  - rate sensitive
- other UDP uses
  - DNS
  - SNMP
- reliable transfer over UDP: add reliability at application layer
  - application-specific error recovery!



UDP segment format

UDP checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted segment

Sender:

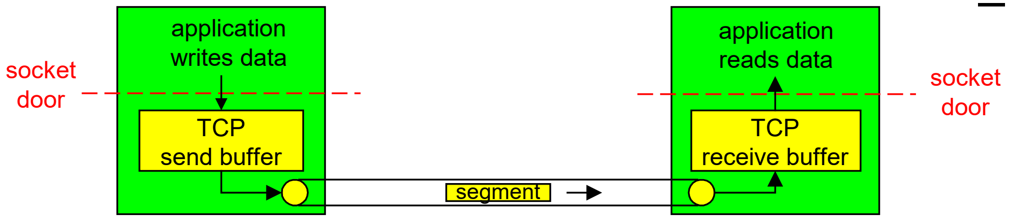
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?* More later ….

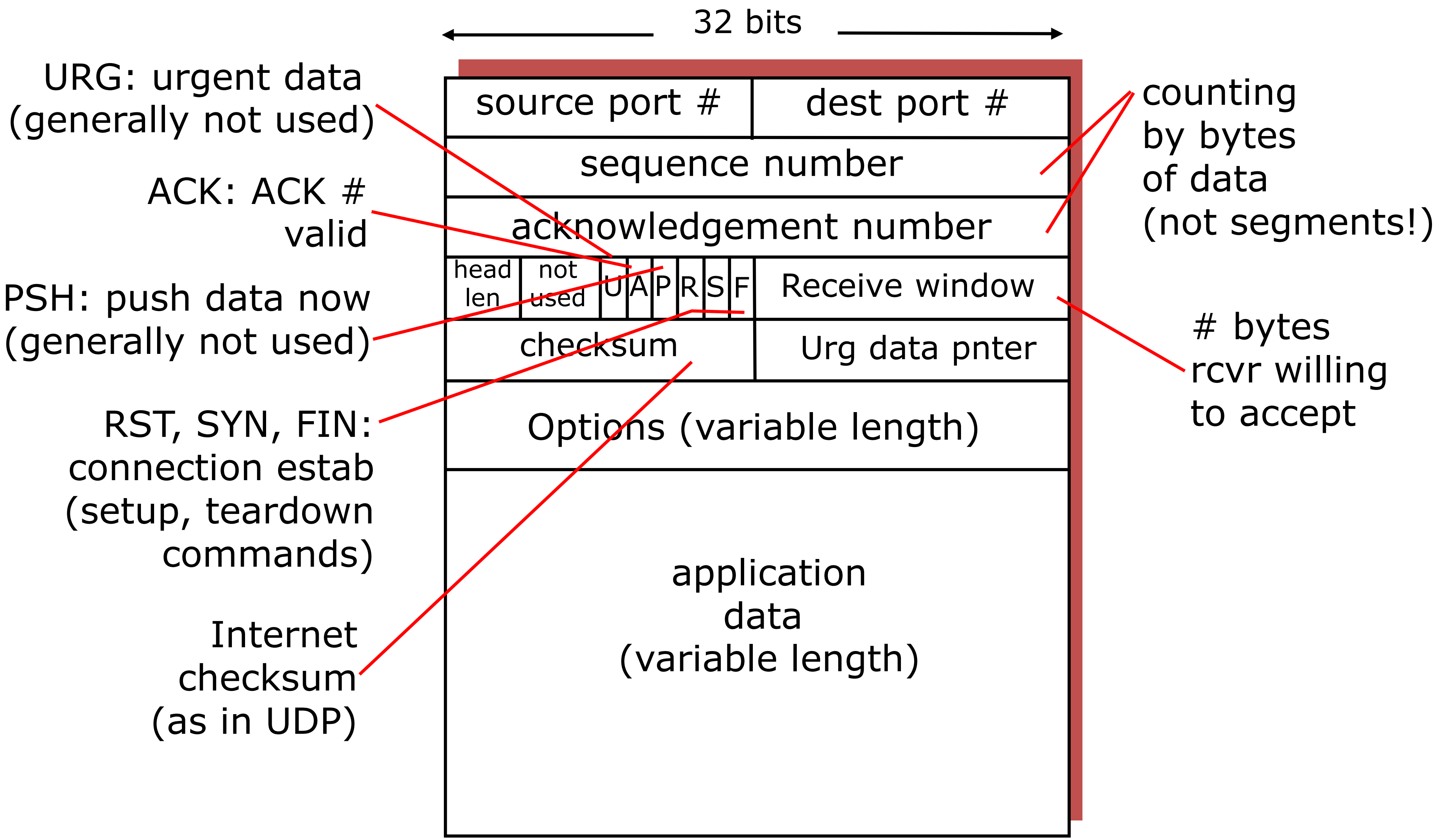
TCP: Overview RFCs: 793, 1122, 1323, 2018, 2581

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte stream:*
  - no “message boundaries”
- pipelined:
  - TCP congestion and flow control set window size
- *send & receive buffers*



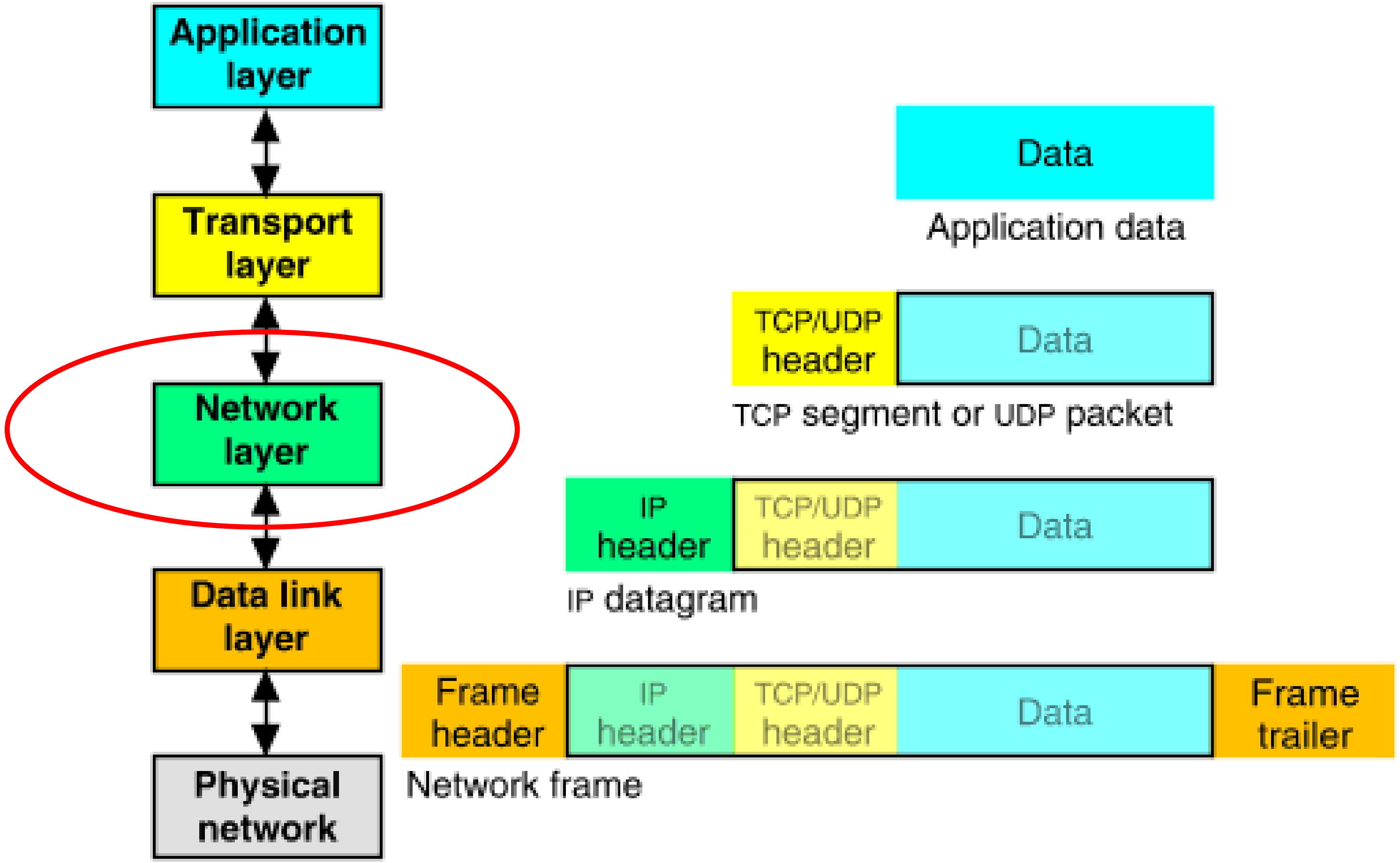
- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) init’s sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

TCP segment structure



THE NETWORK LAYER

IP works at the network layer



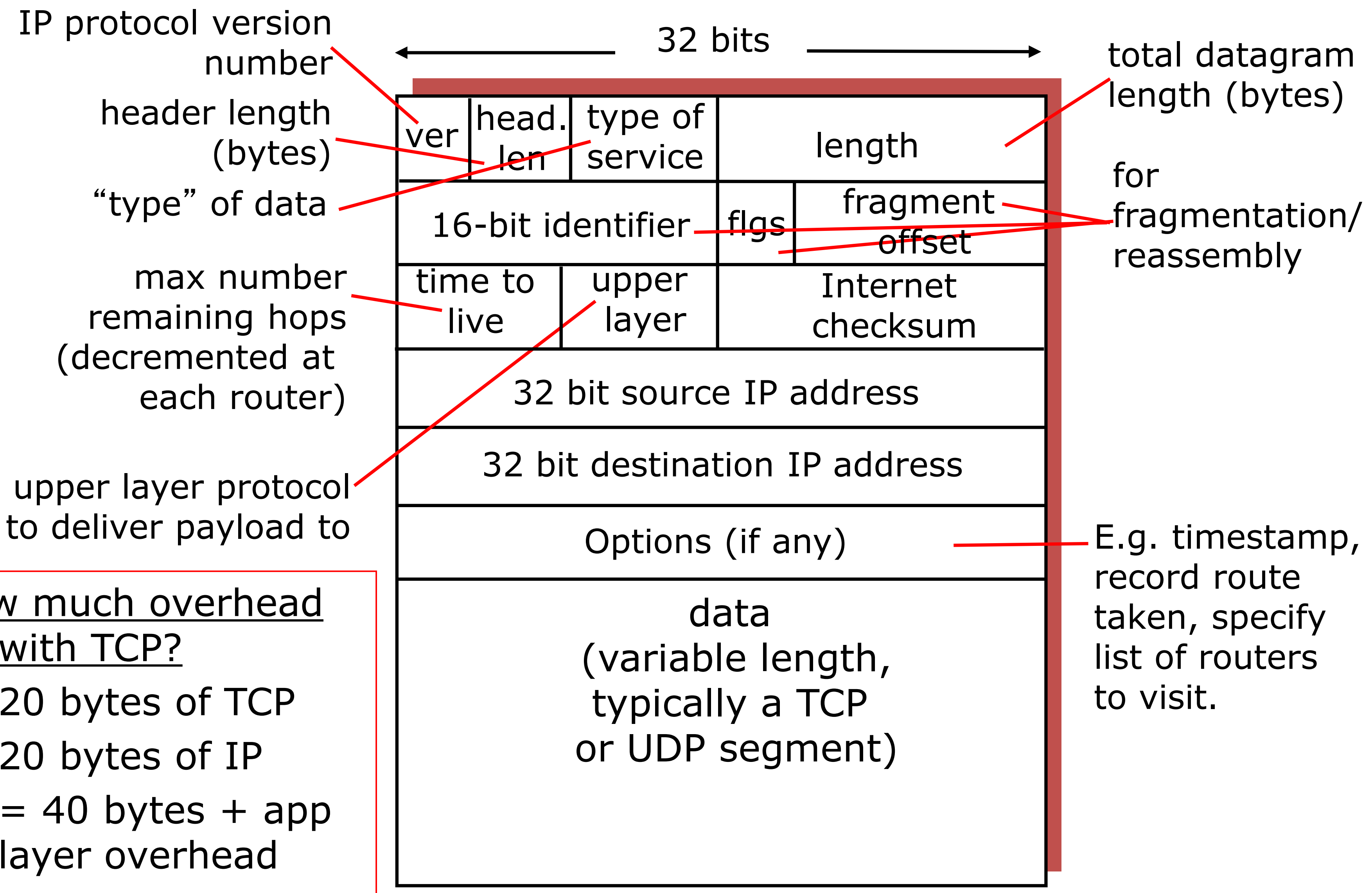
Key Network-Layer Functions

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to dest.
  - *Routing algorithms*

analogy:

- routing: process of planning trip from source to dest
- forwarding: process of getting through single interchange

IPv4 datagram format

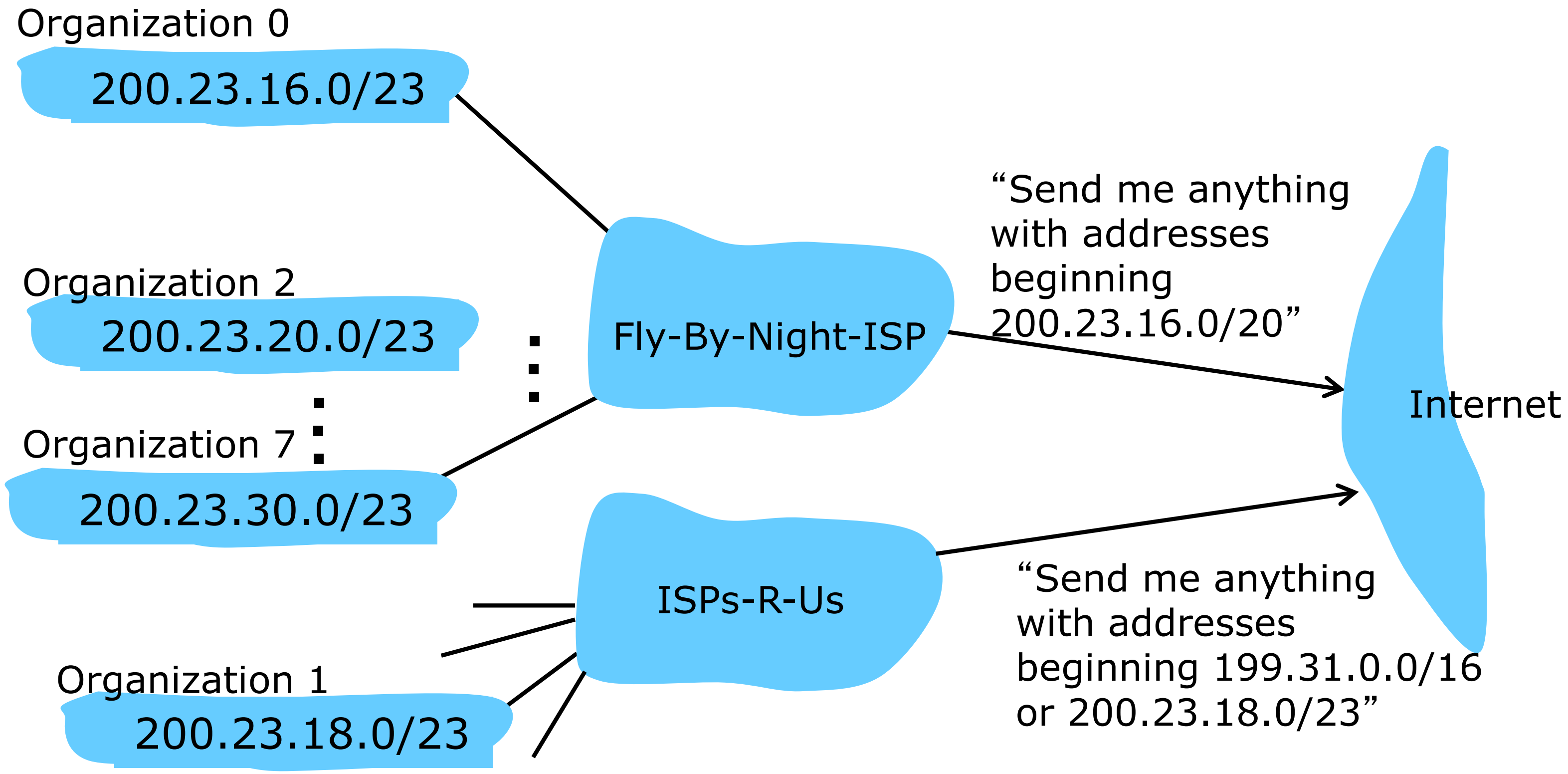


how much overhead with TCP?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1



IP addressing: the last word...

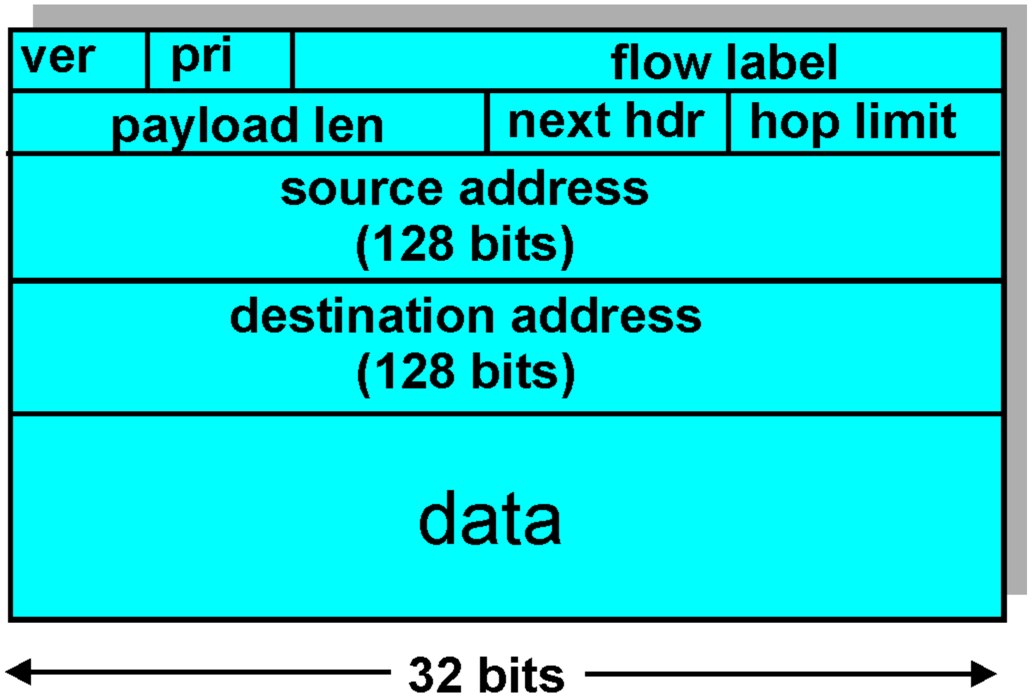
Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

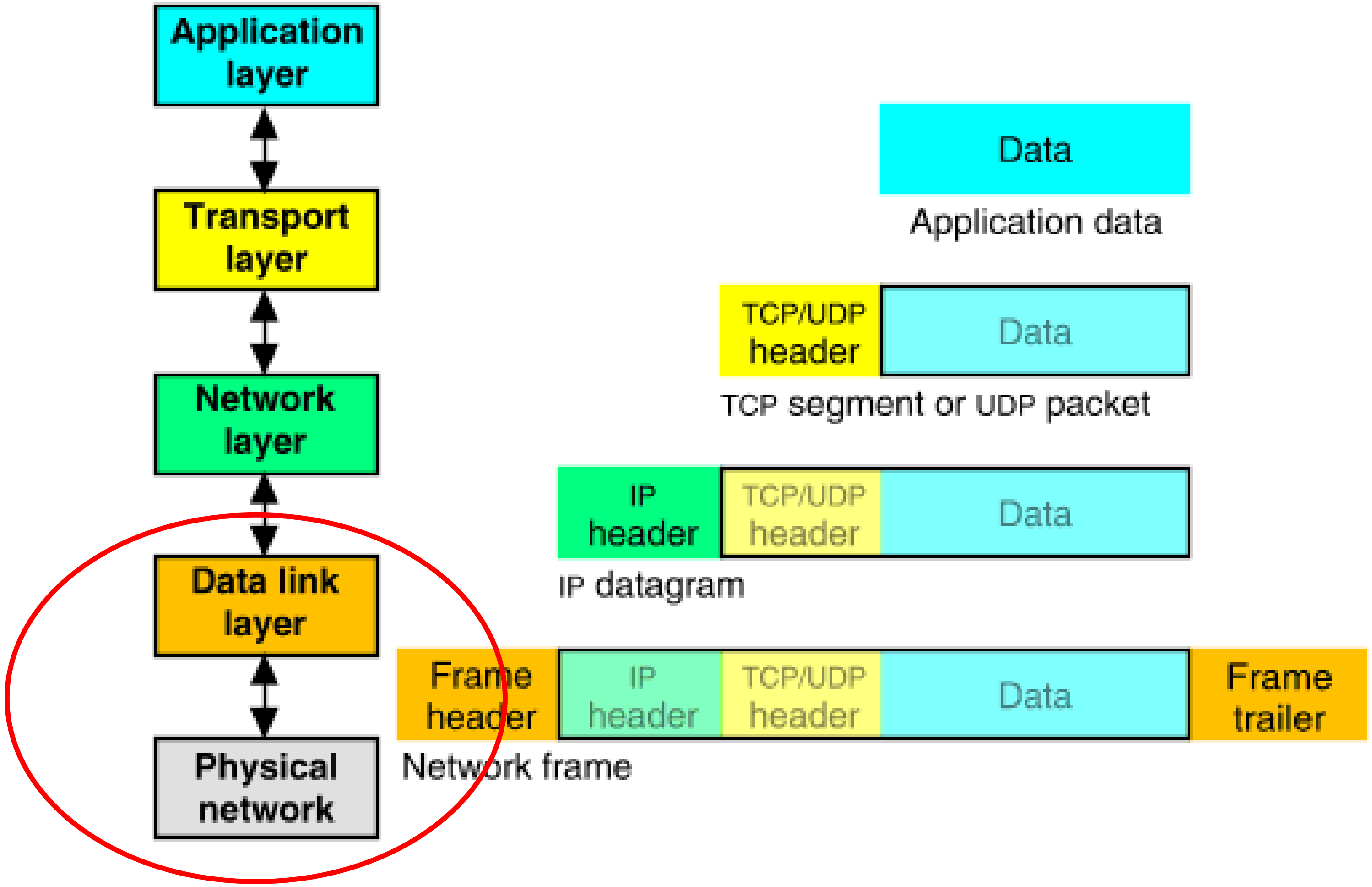
IPv6 datagram format

Priority: identify priority among datagrams in flow
Flow Label: identify datagrams in same "flow."
(concept of"flow" not well defined).
Next header: identify upper layer protocol for data



THE LINK AND PHYSICAL LAYERS

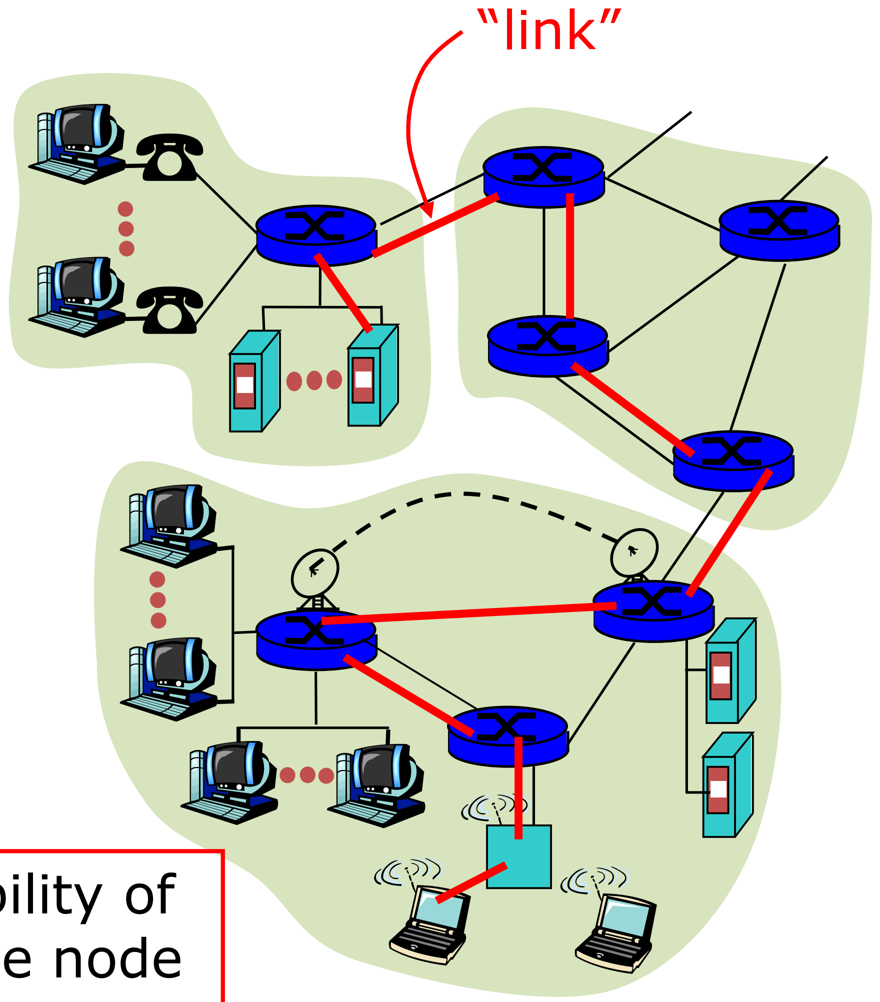
Various links and media are possible at the link and physical layers



Link Layer: Introduction

Some terminology:

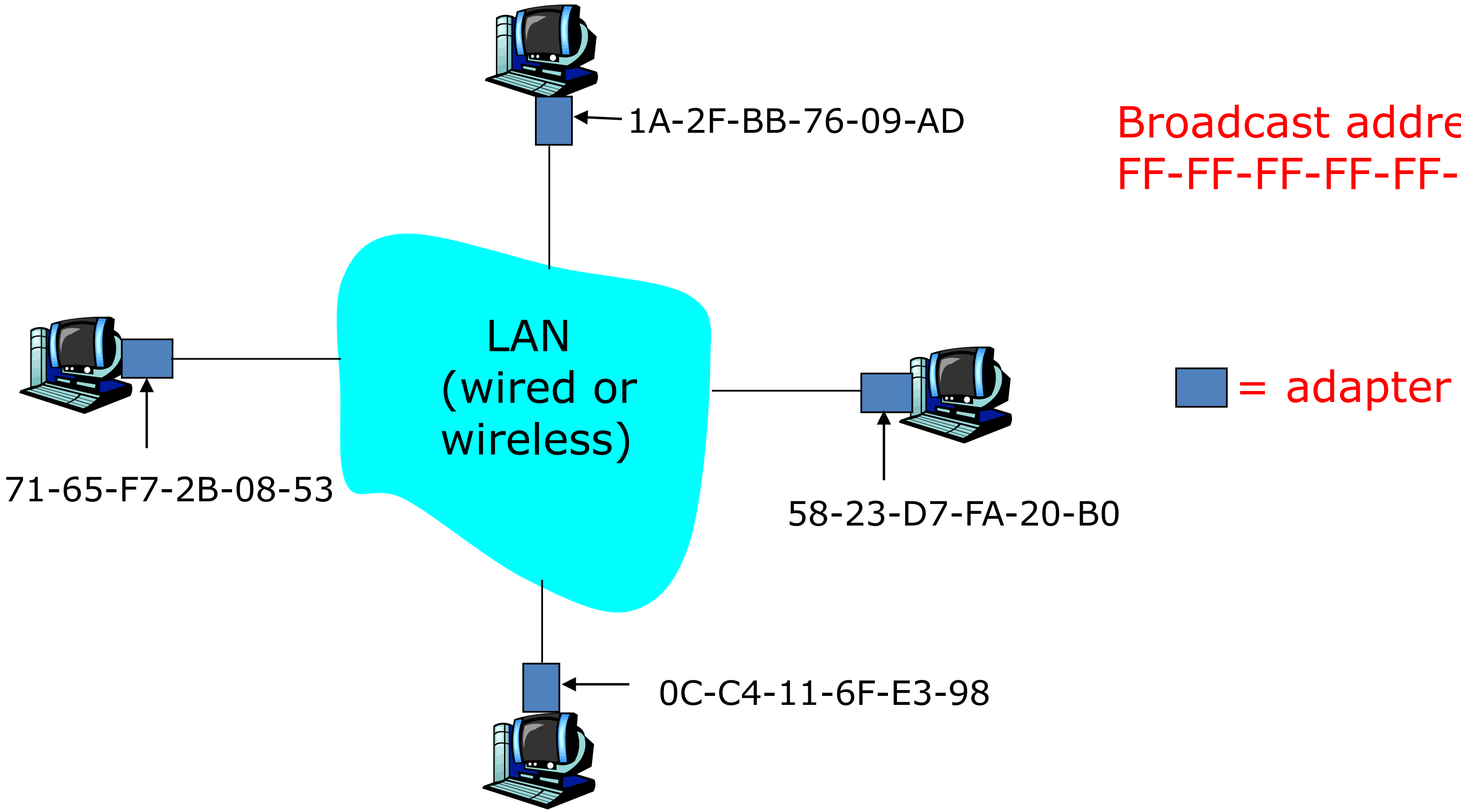
- hosts and routers are **nodes**
- communication channels that connect adjacent nodes along communication path are **links**
  - wired links
  - wireless links
  - LANs
- layer-2 packet is a **frame**, encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to adjacent node over a link

LAN Addresses and ARP

Each adapter on LAN has unique LAN address



Broadcast address = FF-FF-FF-FF-FF-FF

Today’s Objectives

Brief review of networking

- The layered model
- Application layer
  - HTTP
  - port numbers
- Transport layer
- Network layer
- Link and physical layers