Network Applications: Overview, Email

Y. Richard Yang

http://zoo.cs.yale.edu/classes/cs433/

9/13/2018

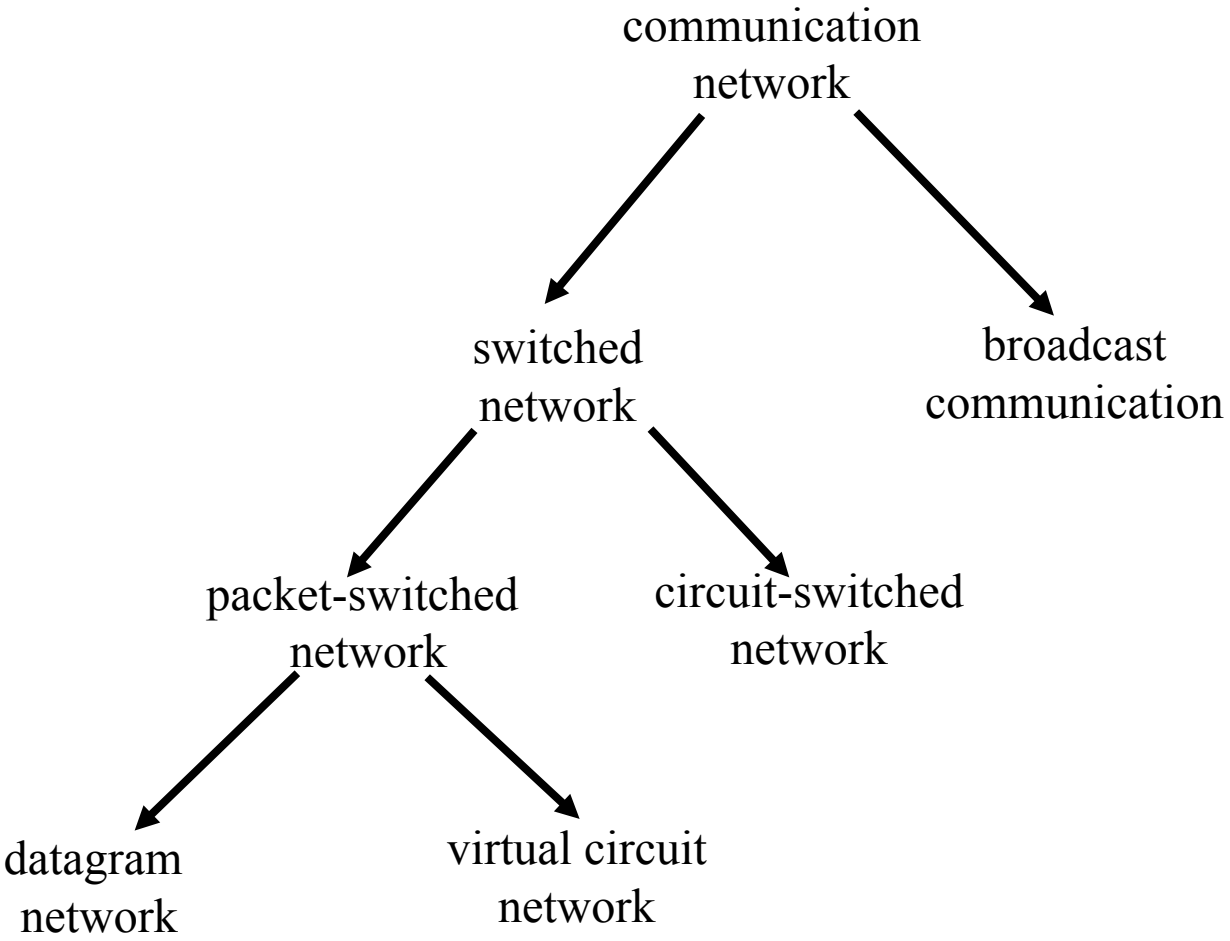
Outline

- Admin and recap
- Application layer overview
- Network applications
  - Email

Admin

- Office hours today and tomorrow posted on class home page
- Questions on Assignment One

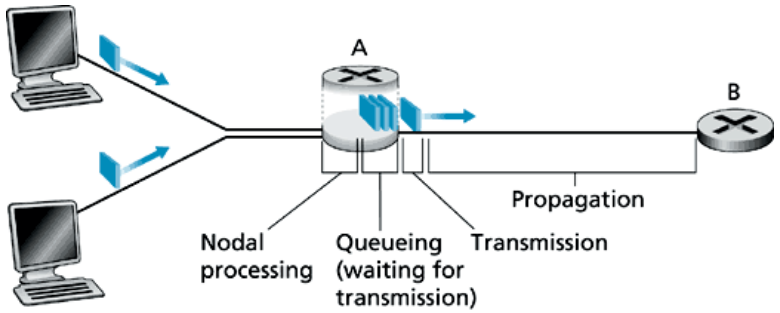
Recap: Summary of the Taxonomy of Communication Networks



Recap: Statistical Multiplexing

A simple model to compare bandwidth efficiency of
- reservation/dedication (aka circuit-switching) and
- no reservation (aka packet switching)

setup
- a single bottleneck link with rate R (L/R to trans. L bits)
- n flows; each flow has an arrival rate of a/n



- no reservation: all arrivals into the single link with rate R, the queueing delay + transmission delay:

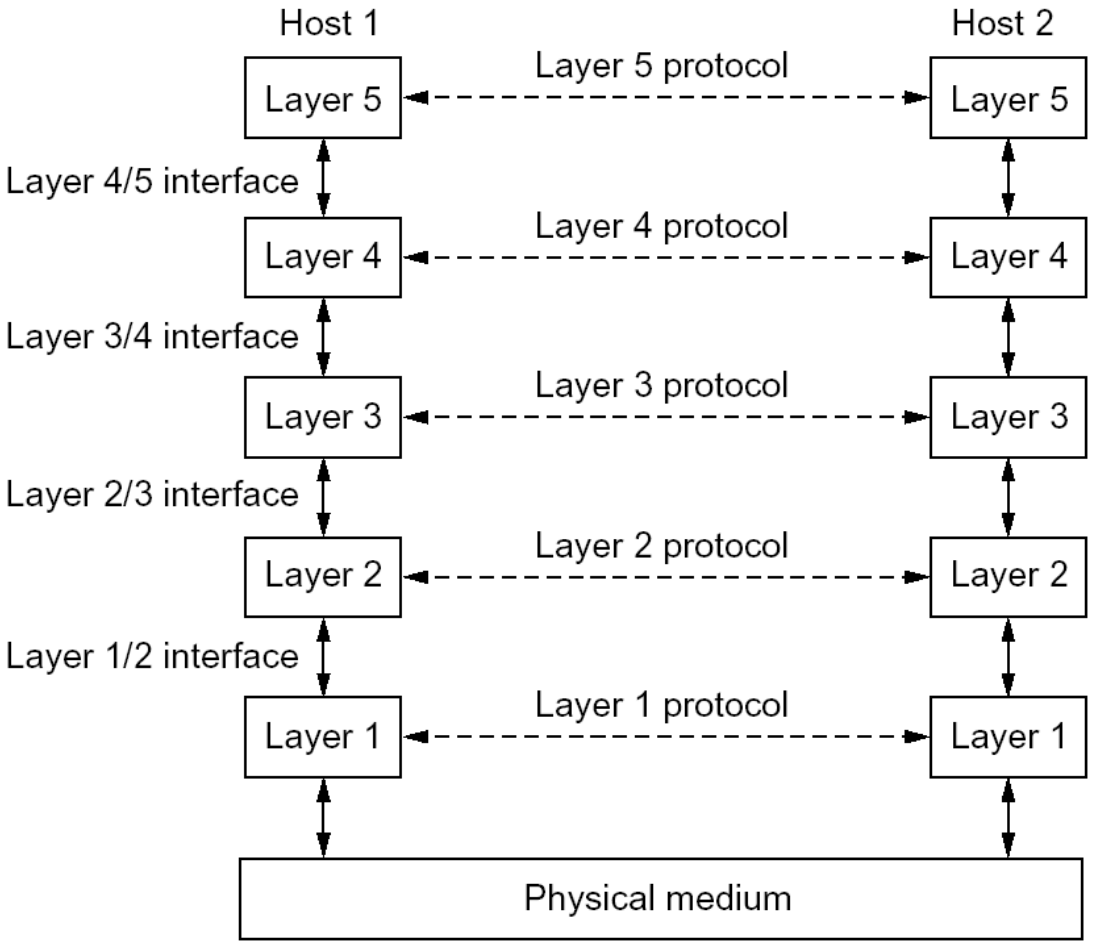
$$\frac{L}{R}\frac{1}{1-\rho}$$

- reservation: each flow uses its own reserved (sub)link with rate R/n, the queueing delay + transmission delay:

$$n\frac{L}{R}\frac{1}{1-\rho}$$

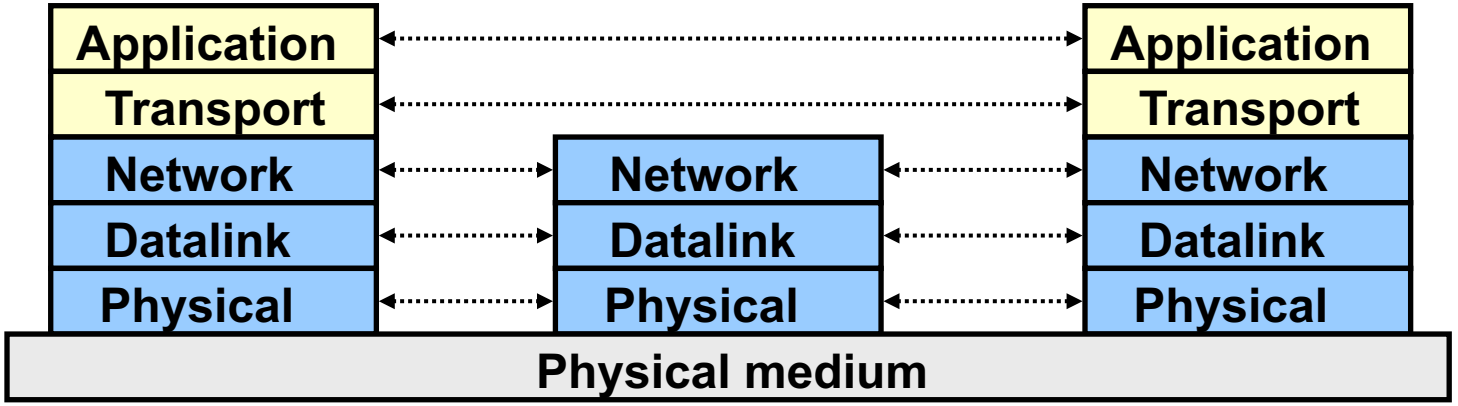
Recap: Layering

- Why layering
  - reference model
  - modularization
- Concepts
  - service, interface, and protocol
  - physical vs logical communication
- Key layering principle
  - end-to-end arguments to place functions in layers

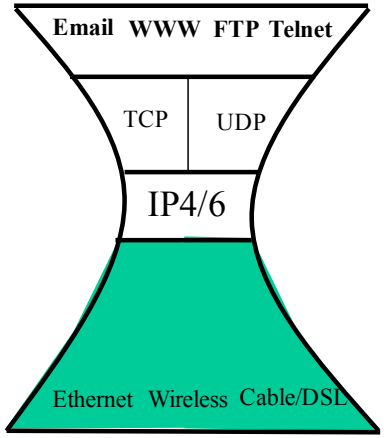


Recap: Internet Layering

- Five layers
  - highest two layers are implemented in host

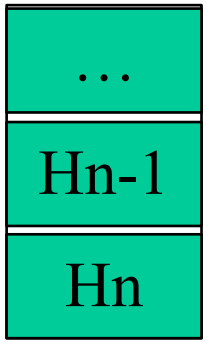


- Form an hourglass structure

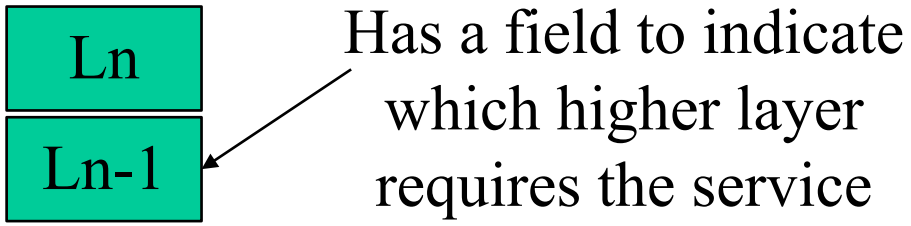


Some Implications of Layered Architecture

- A packet as a stack container

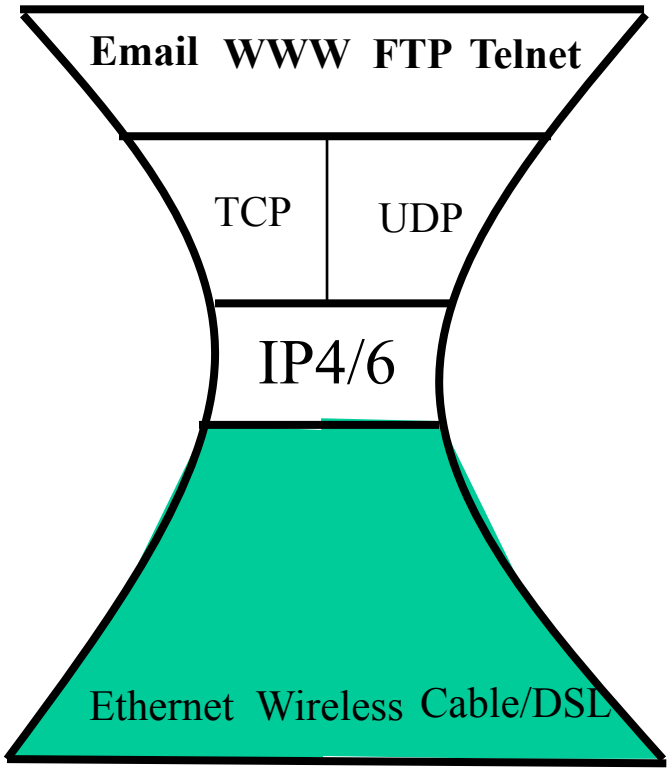


- Each layer needs multiplexing and demultiplexing to serve layer above

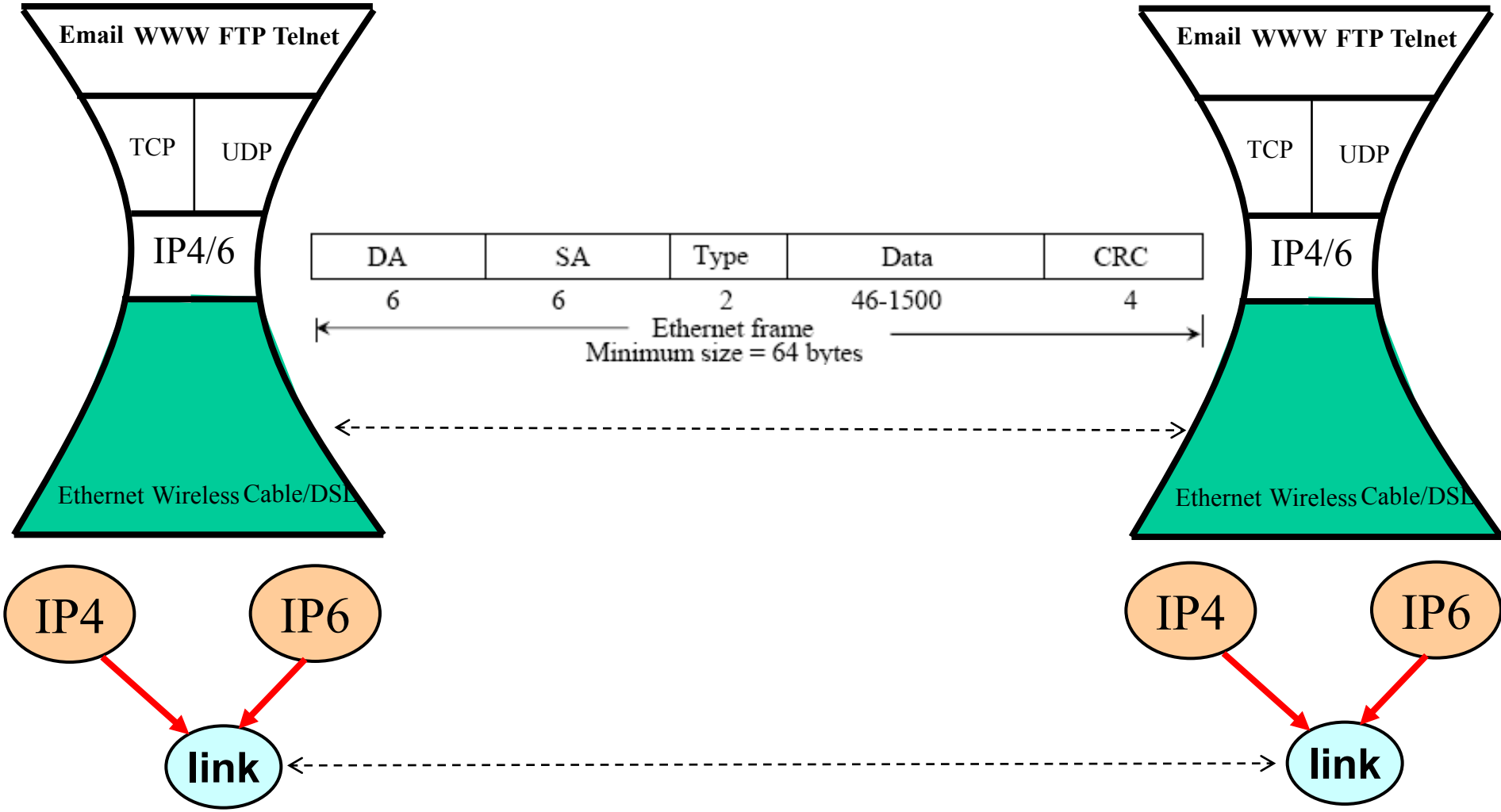


Link Layer (Ethernet)

- Services (to network layer)
  - multiplexing/demultiplexing
    - from/to the network layer
  - error detection
  - multiple access control
    - arbitrate access to shared medium
- Interface
  - send frames to a directly reachable peer

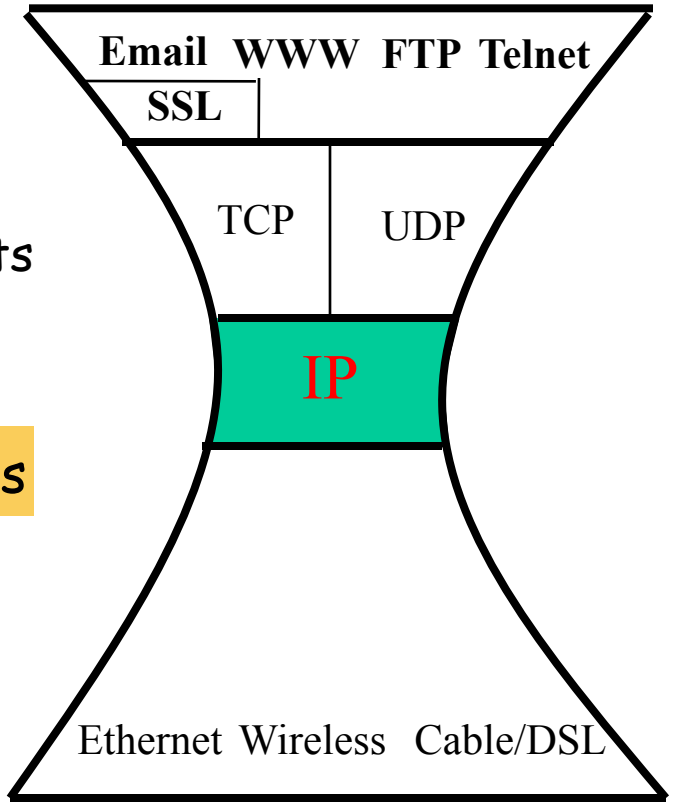


Link Layer: Protocol Header (Ethernet)

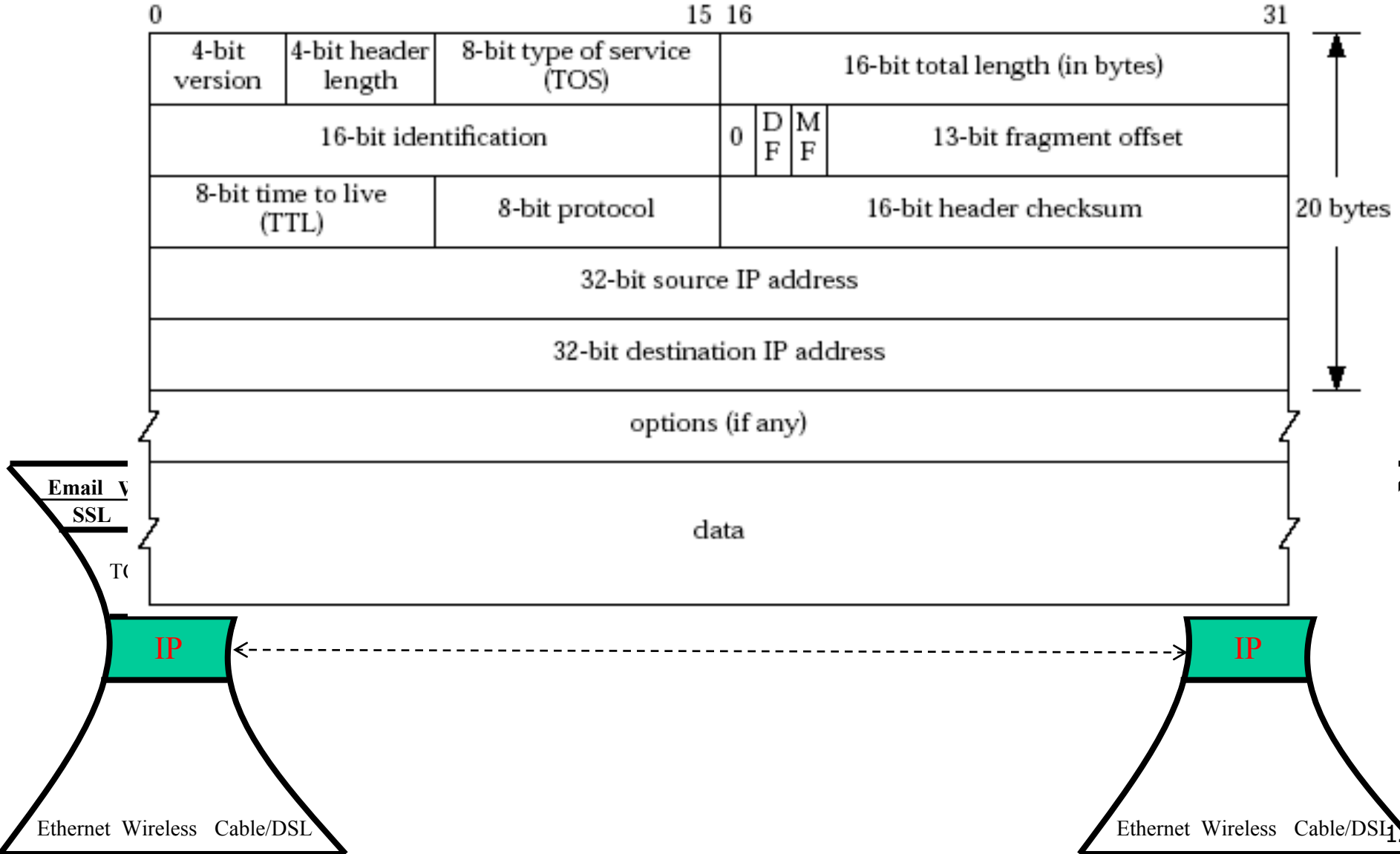


Network Layer: IP

- Services (to transport layer)
  - multiplexing/demultiplexing from/to the transport
  - fragmentation and reassembling: partition a fragment into smaller packets
    - removed in IPv6
  - error detection
  - routing: best-effort to send packets from source to destination
  - certain QoS/CoS
  - does not provide reliability or reservation
- Interface:
  - send a packet to a (transport-layer) peer at a specified global destination, with certain QoS/CoS

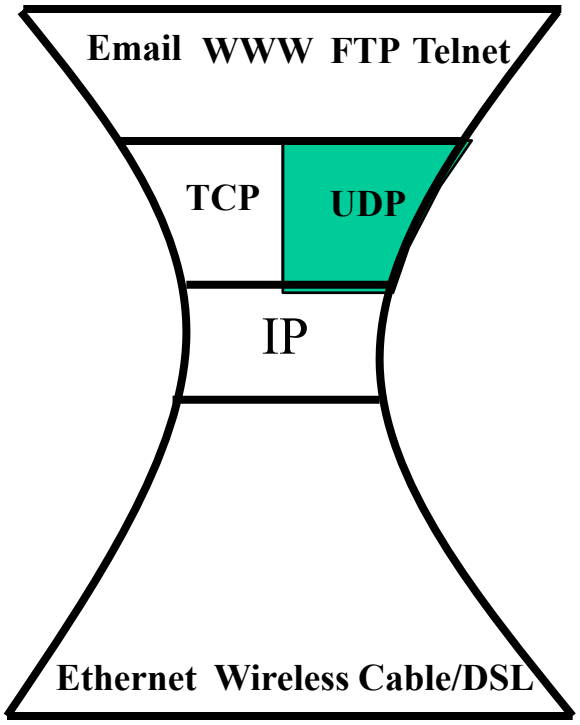


Network Layer: IPv4 Header



Transport Layer: UDP

- A connectionless service
- Does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee
  - why is there a UDP?

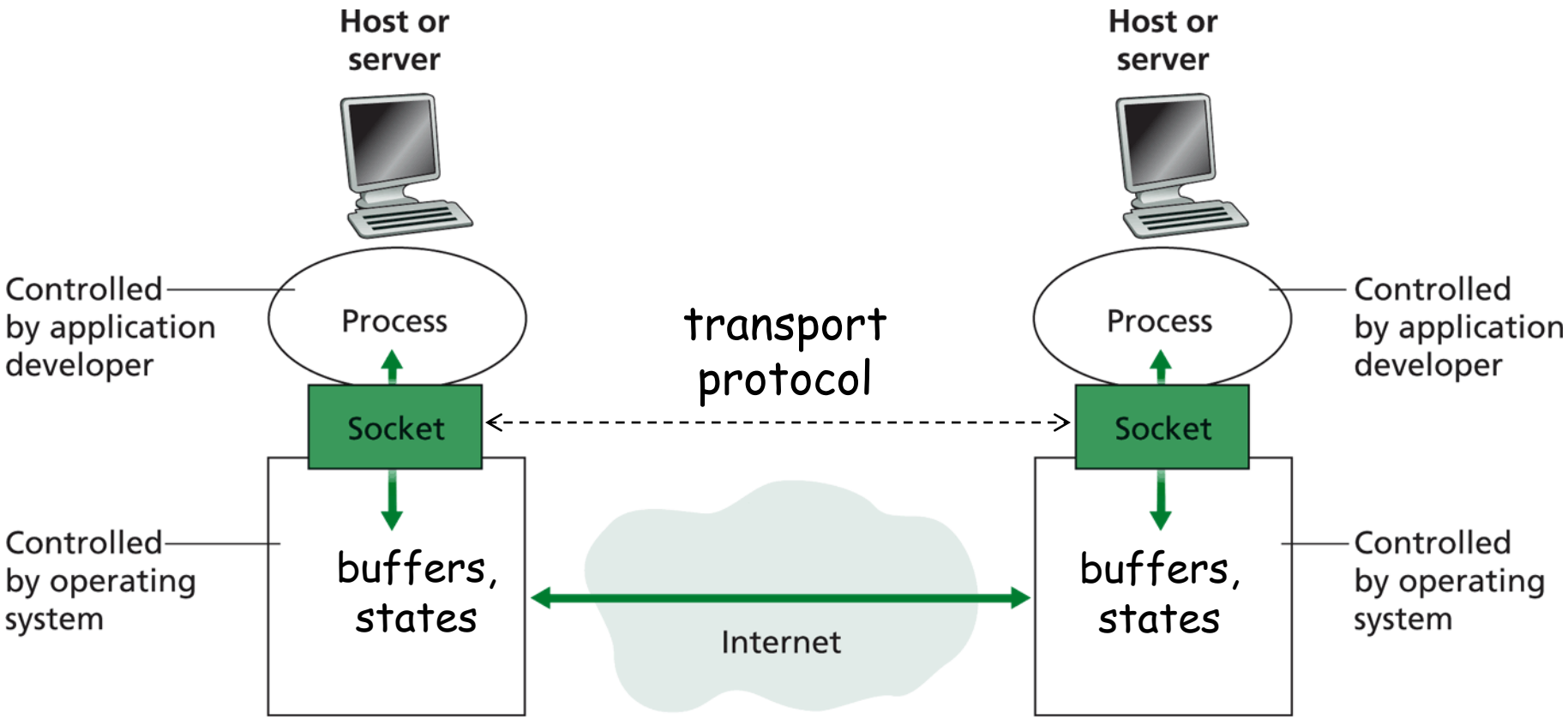


Transport Services and APIs

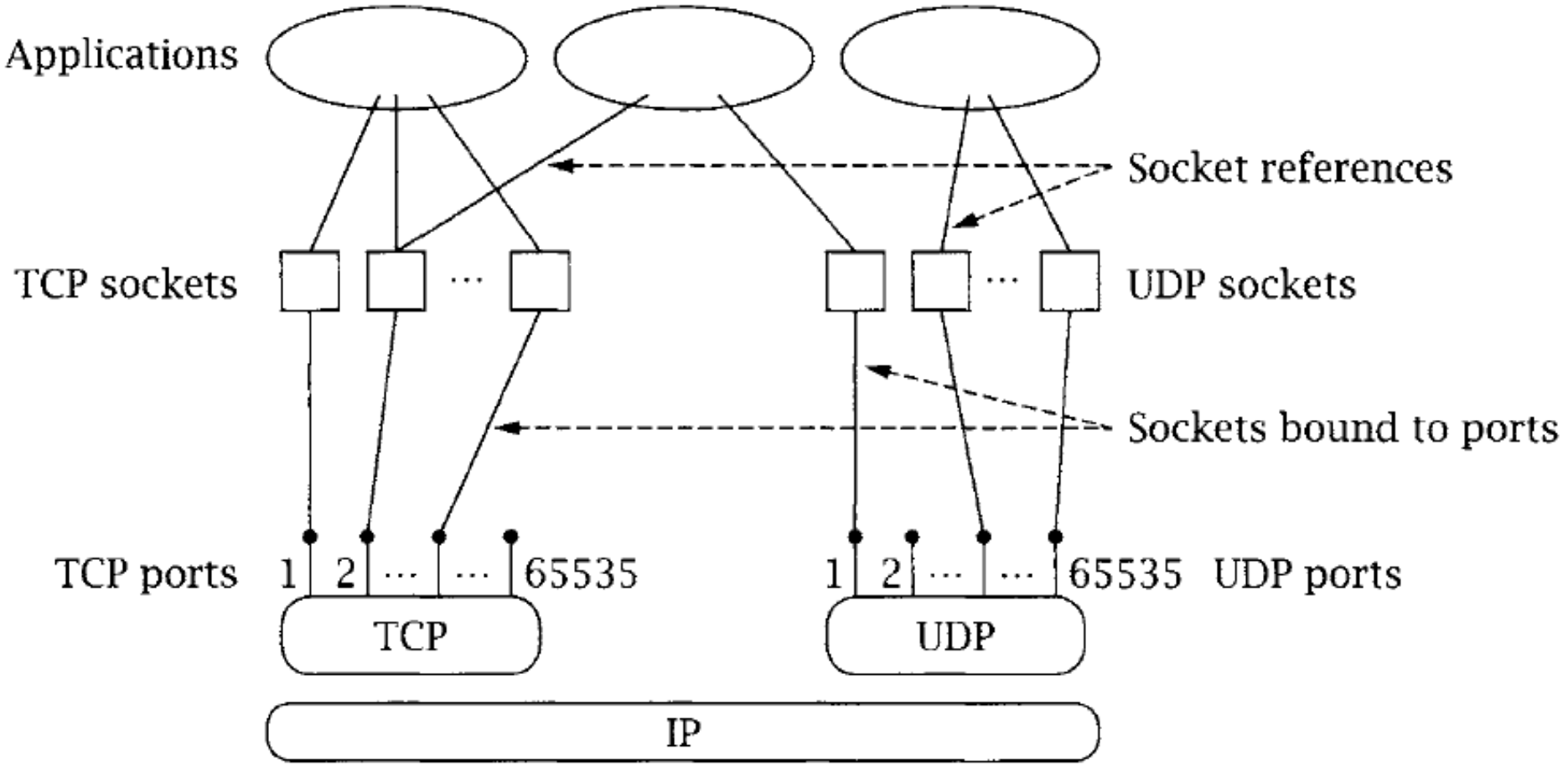
- Multiple services and APIs proposed in history
  - XTI (X/Open Transport Interface), a slight modification of the Transport Layer Interface (TLI) developed by AT&T.

- Commonly used transport-layer service model and API: Socket
  - sometimes called "Berkeley sockets" acknowledging their heritage from Berkeley Unix
  - a socket has a transport-layer local port number
    - e.g., email (SMTP) port number 25, web port number 80
  - Application can send data into socket, read data out of socket
  - an application process binds to a socket (-a all; -u udp; -n number)
    - %netstat -aun

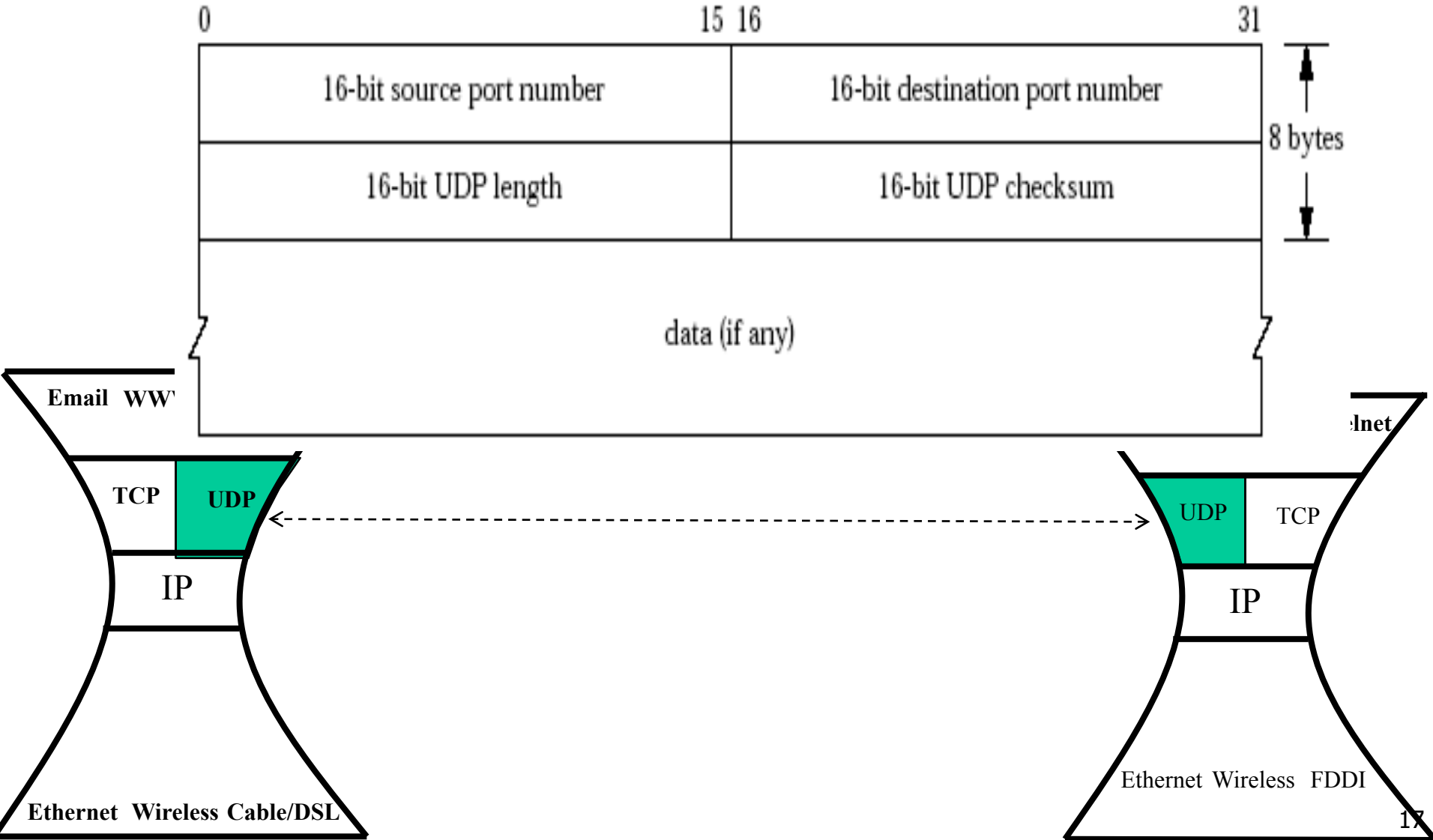
Socket Service Model and API



Multiplexing/Demultiplexing

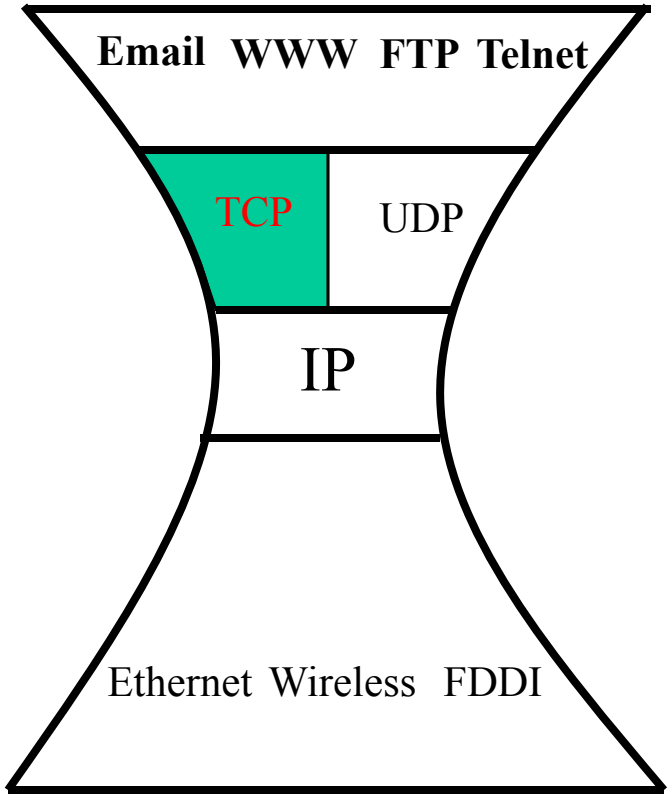


Transport Layer: UDP Header

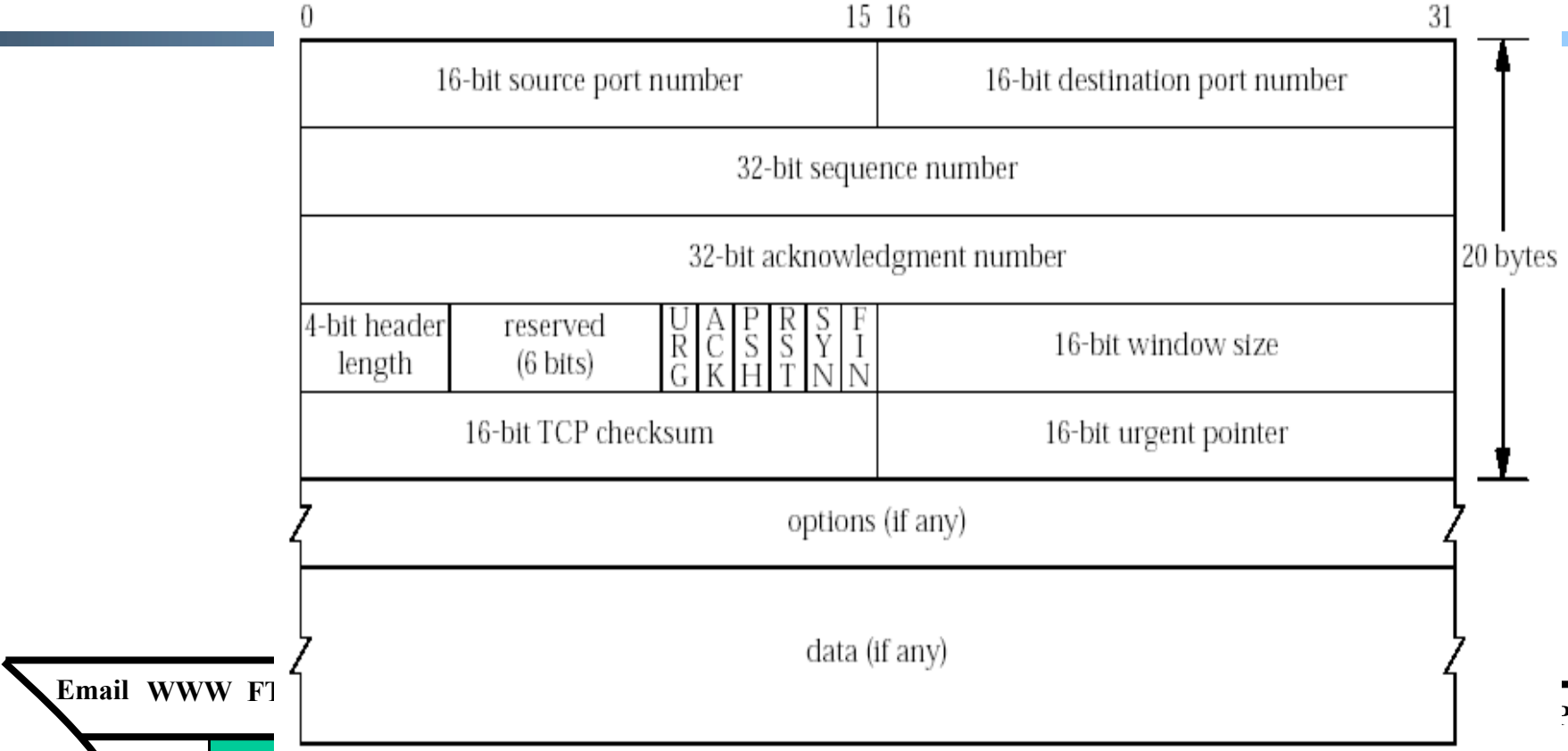


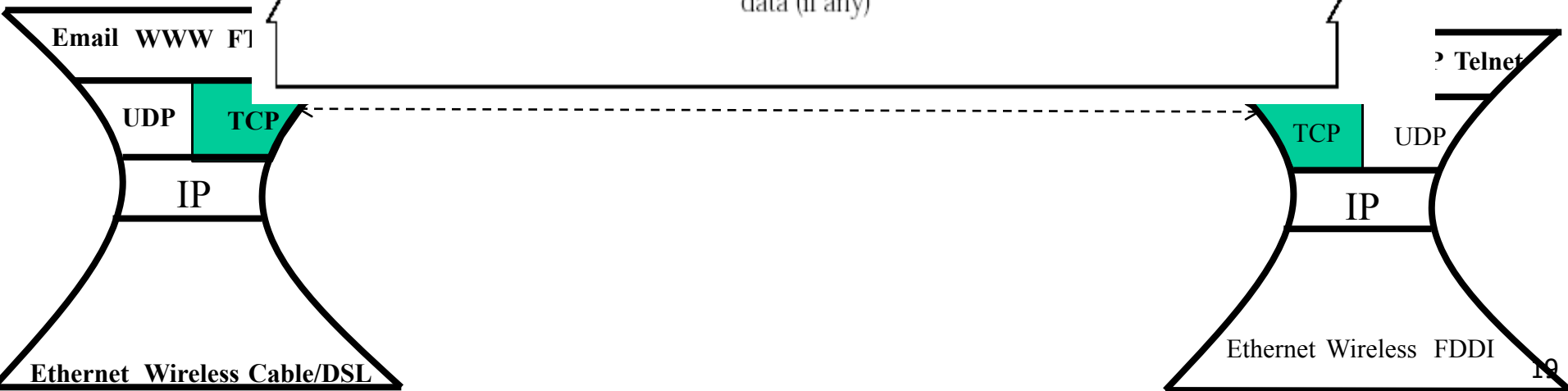
Transport Layer: TCP

- Services
  - multiplexing/demultiplexing
  - reliable transport
    - between sending and receiving processes
    - setup required between sender and receiver: a connection-oriented service
  - flow control: sender won't overwhelm receiver
  - congestion control: throttle sender when network overloaded
  - error detection
  - does not provide timing, minimum bandwidth guarantees
- Interface:
  - send a packet to a (app-layer) peer

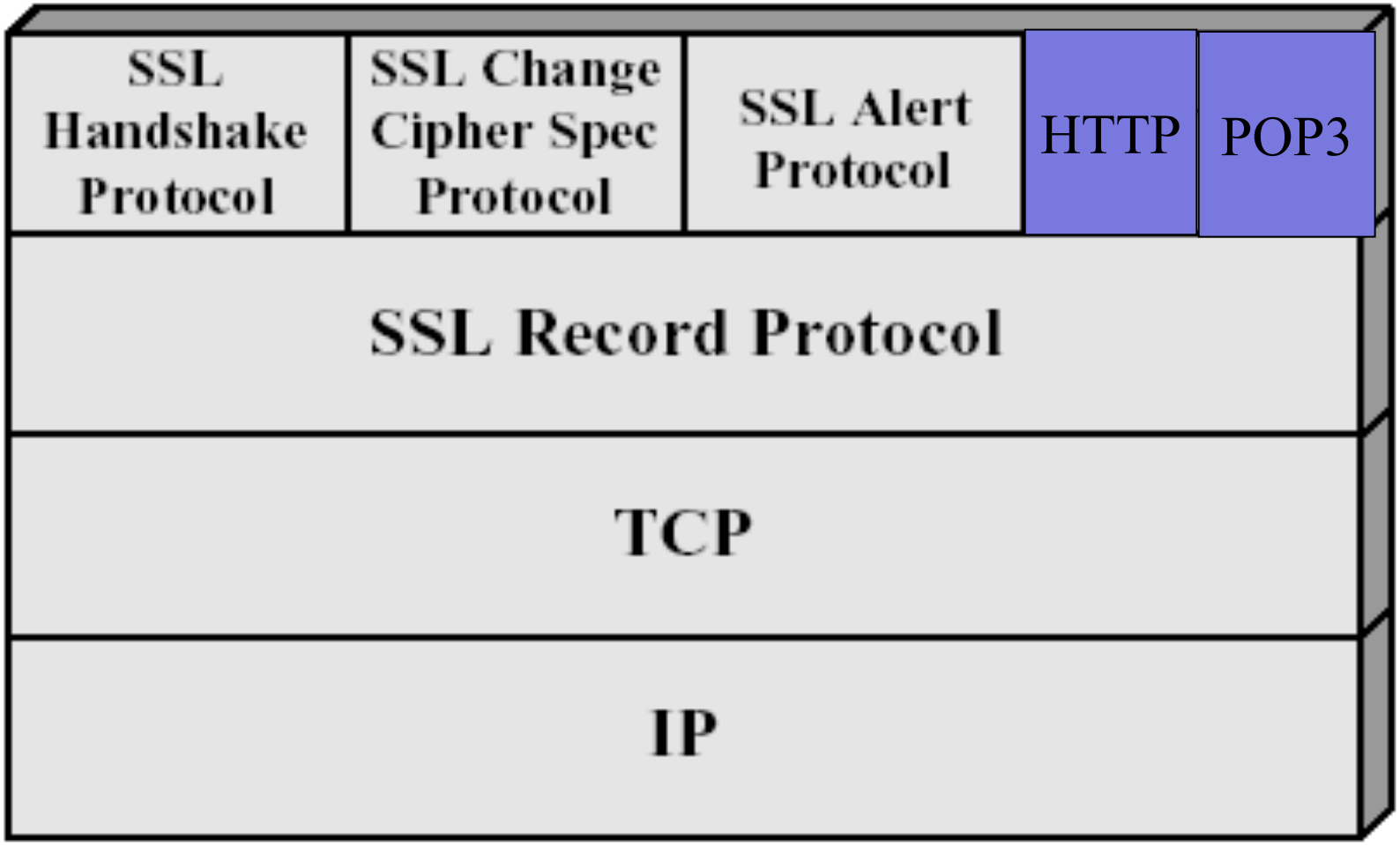


Transport Layer: TCP Header



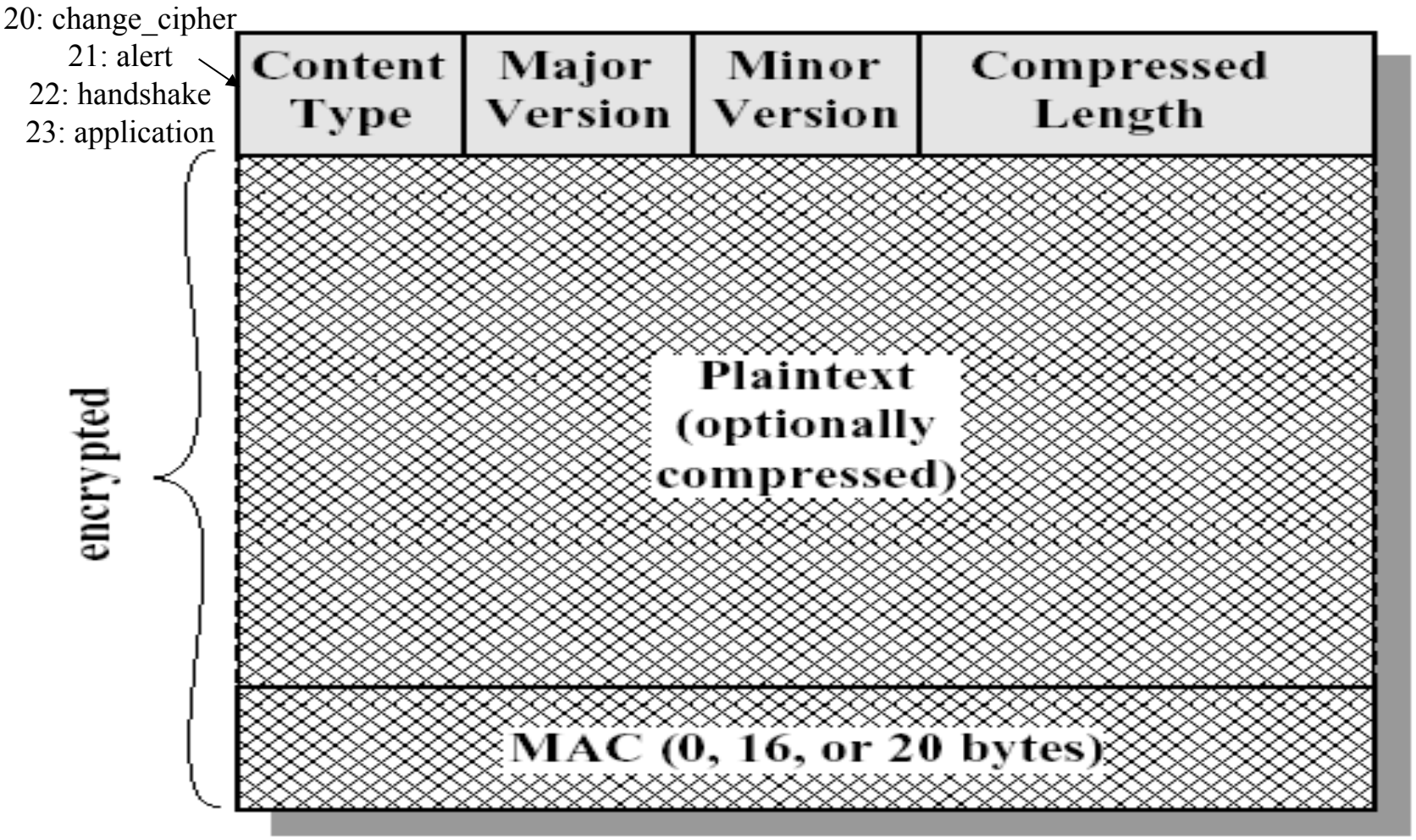


Secure Socket Layer Architecture



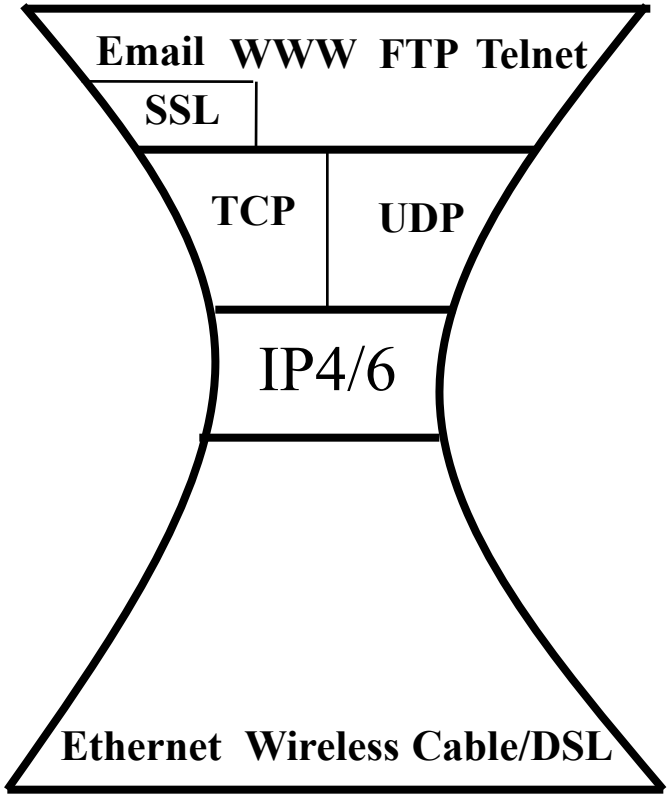

```
%openssl s_client -connect pop.gmail.com:995
```

SSL Record-Layer Packet Format

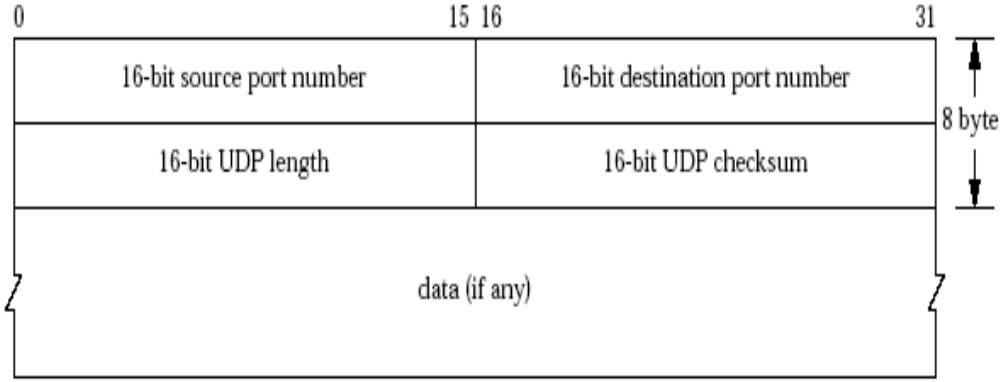


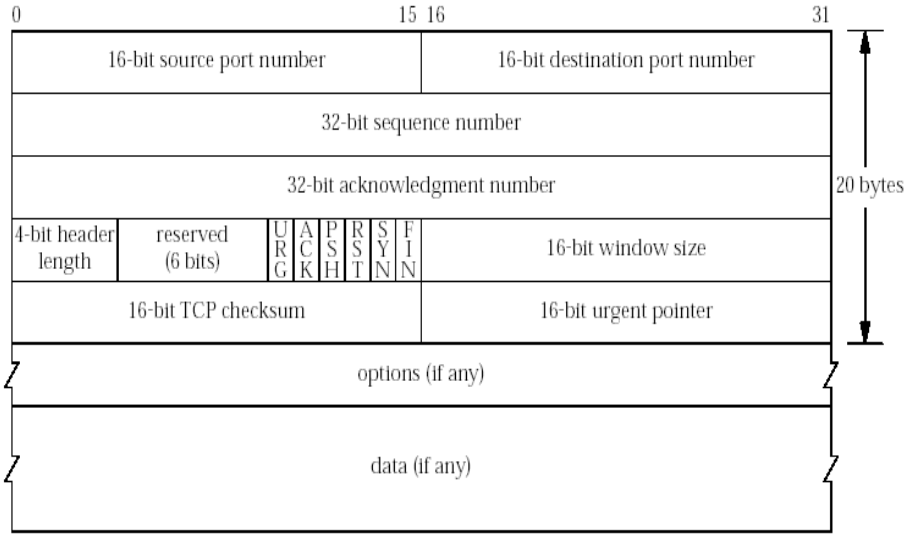
Summary: The Big Picture of the Internet

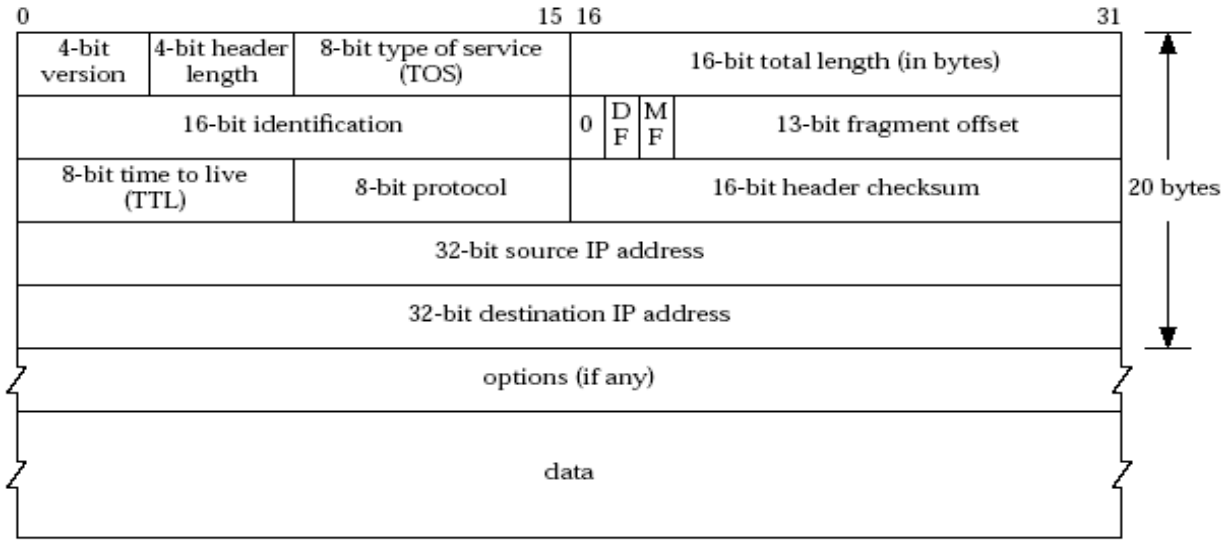
- Hosts and routers:
  - ~ 1 bil. hosts
  - autonomous systems organized roughly hierarchical
  - backbone links at 100 Gbps
- Software:
  - datagram switching with virtual circuit support at backbone
  - layered network architecture
    - use end-to-end arguments to determine the services provided by each layer
    - the hourglass architecture of the Internet

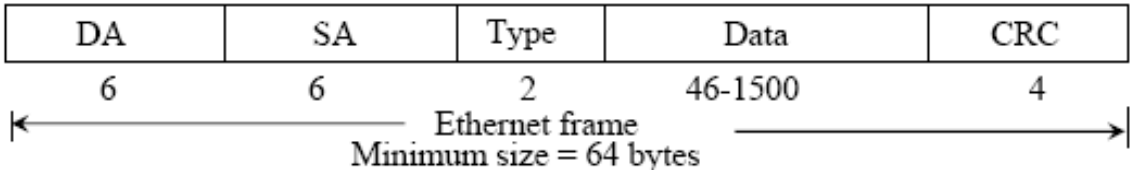


Protocol Formats









Outline

- Admin and recap
- *Application layer overview*

Application Layer: Goals

- Conceptual + implementation aspects of network application protocols
  - client server paradigm
  - peer to peer (distributed) paradigm
  - network app. programming

- Learn about applications by examining common applications
    - pop/smtp
    - dns
    - ftp, http (1, 1.1, /2), content distribution
    - freenet, gossiping, BT, consensus

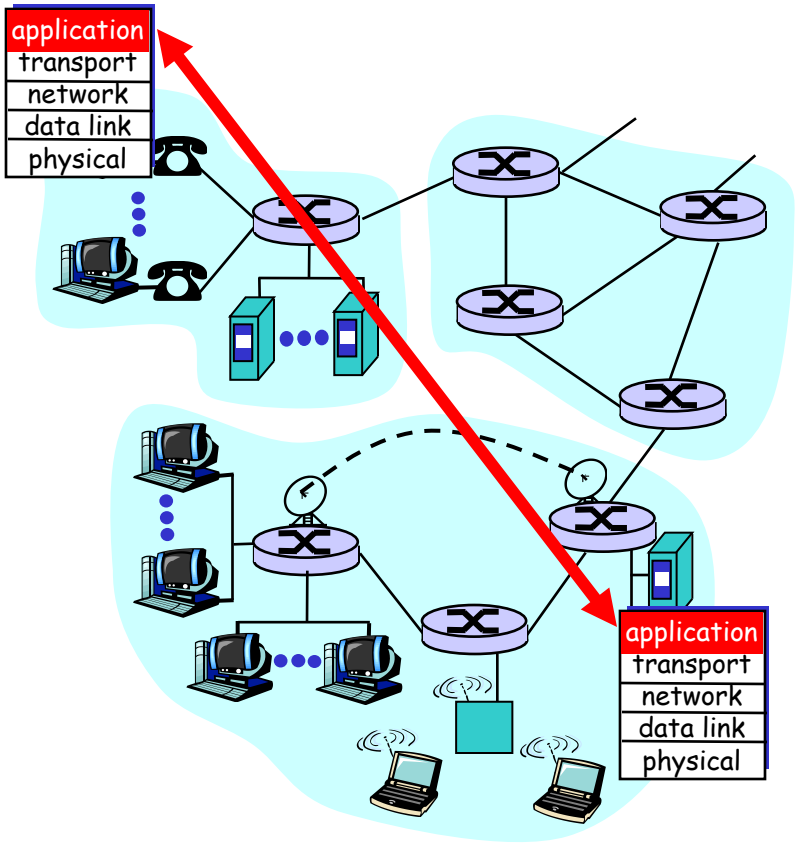
Network Applications vs. Application-layer Protocols

Network application: communicating, distributed processes

- a process is a program that is running within a host
  - a user agent is a process serving as an interface to the user
    - web: browser
    - streaming audio/video: media player
- processes communicate by an application-layer protocol
  - e.g., email, Web

Application-layer protocols

- one "piece" of an app
- define messages exchanged by apps and actions taken
- implementing services by using the service provided by the lower layer, i.e., the transport layer



App. Protocols and their Transport Protocols

- An application needs to choose the transport protocol

Application	Application layer protocol	Underlying transport protocol
e-mail	smtp [RFC 821]	TCP/SSL
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP/SSL
file transfer	ftp [RFC 959]	TCP
Internet telephony	proprietary (e.g., Vocaltec)	typically UDP
remote file server	NFS	TCP or UDP
streaming multimedia	proprietary	typically UDP but moving to http

Client-Server Paradigm

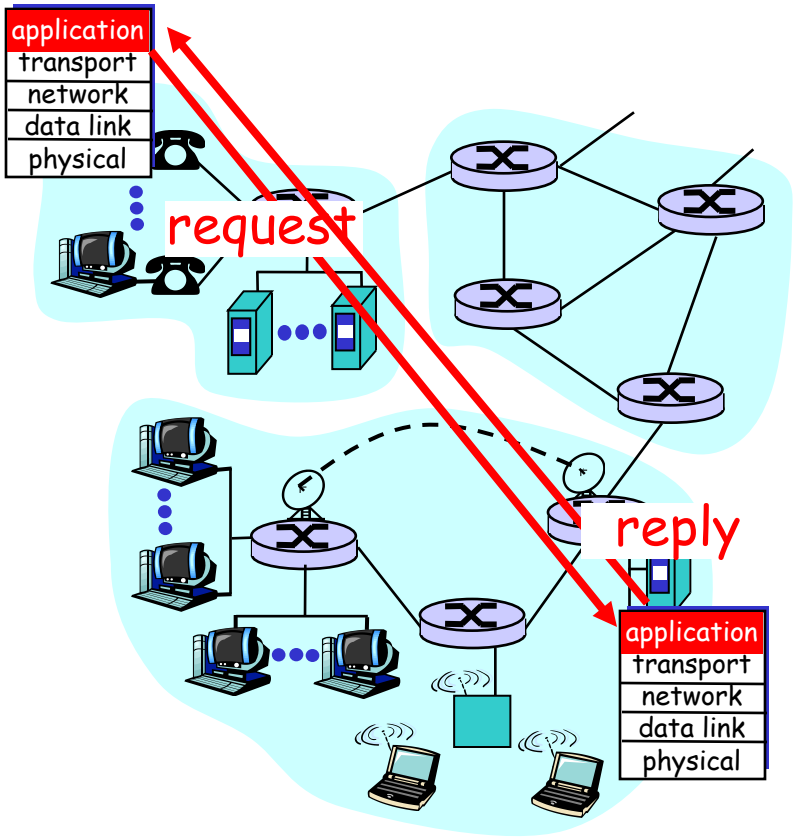
Typical network app has two pieces: *client* and *server*

Client (C):

- initiates contact with server ("speaks first")
- typically requests service from server
- for Web, client is implemented in browser; for e-mail, in mail reader

Server (S):

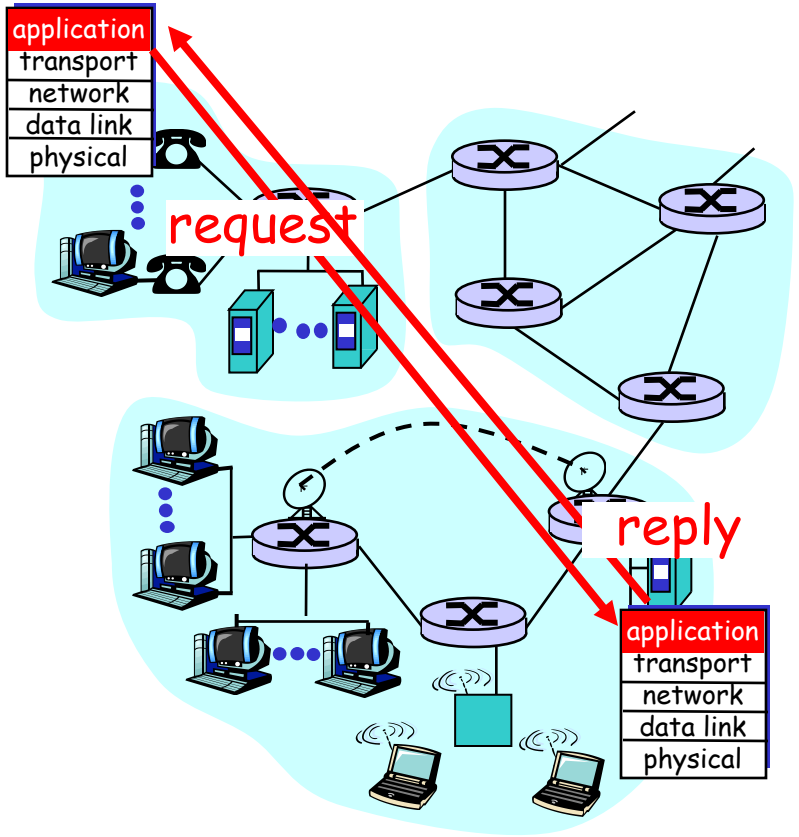
- provides requested service to client
- e.g., Web server sends requested Web page; mail server delivers e-mail



Client-Server Paradigm: Key Questions

Key questions to ask about a C-S application design

- Is the design **extensible**?
- Is the design **scalable**?
- Is the design **robust** (e.g., server crashes)?
- Is the design **secure** (e.g., what kinds of security issues does it have/address)?



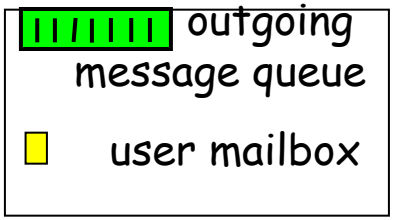
Outline

- Admin and recap
- Application layer overview
- Network applications
  - Email

Electronic Mail

- Still active
  - 80B emails/day
  - 3.9B active email boxes
- As simple as email, its app structure has complexities
  - A highly recommended reading: a history of Email development
    - linked on the Schedule page

Electronic Mail: Components

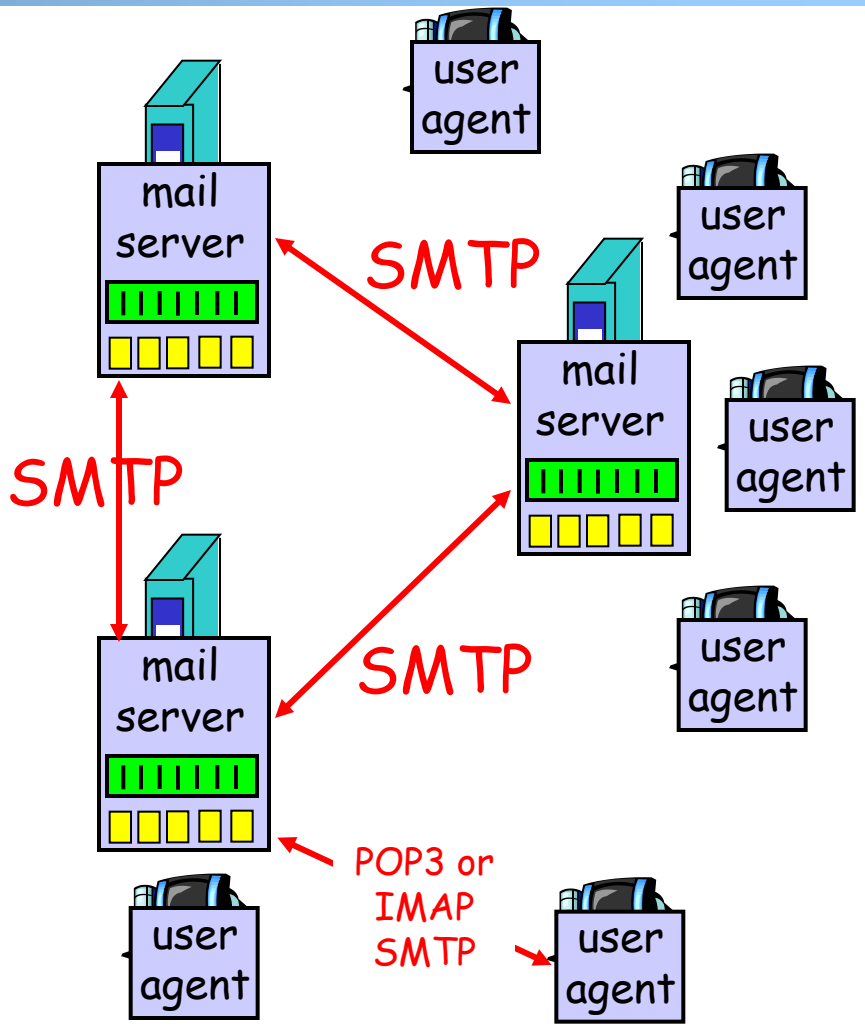


Two subsystems

- Message handling system (MHS)
- User agents

Two types of protocols

- Mail access protocols
  - POP3: Post Office Protocol [RFC 1939]
  - IMAP: Internet Mail Access Protocol [RFC 1730]
- Mail transport protocol
  - SMTP: Simple Mail Transport Protocol [RFC5321]



Email Transport Architecture

MUA: User Agent

Mediator: User-level Relay

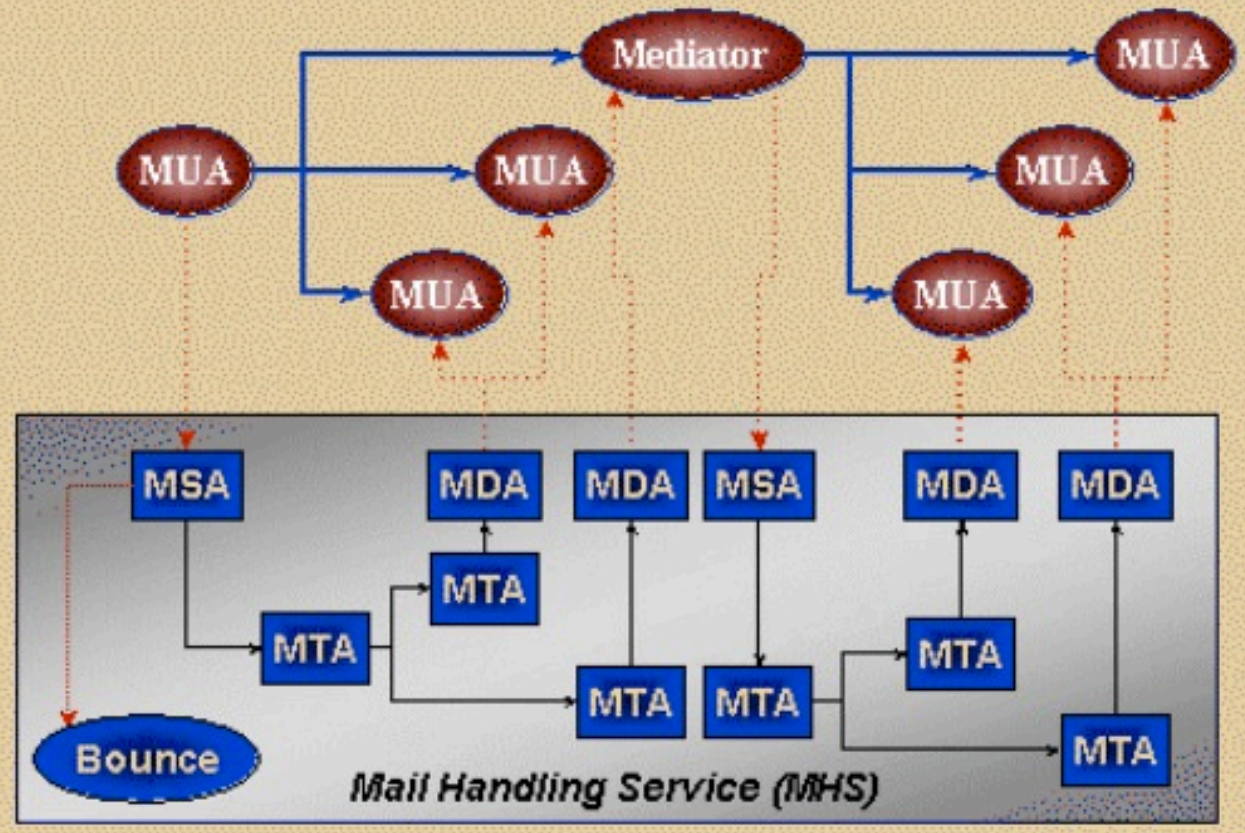
MHS: Mail Handling (transit) Service

MSA: Submission

MTA: Transfer

MDA: Delivery

Bounce: Returns



http://www.maawg.org/sites/maawg/files/news/MAAWG_Email_Authentication_Paper_2008-07.pdf

POP3 Protocol: Mail Retrieval (a Basic C-S Protocol)

Authorization phase

- client commands:
  - **user**: declare username
  - **pass**: password
- server responses
  - **+OK**
  - **-ERR**

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

Transaction phase, client:

- **list**: list message numbers
- **retr**: retrieve message by number
- **dele**: delete
- **quit**

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

```
%openssl s_client -connect pop.gmail.com:995
```

Exercise

- Send email to yalecs433533
- Retrieve using pop3

See pop3-trace.txt

Some Observations

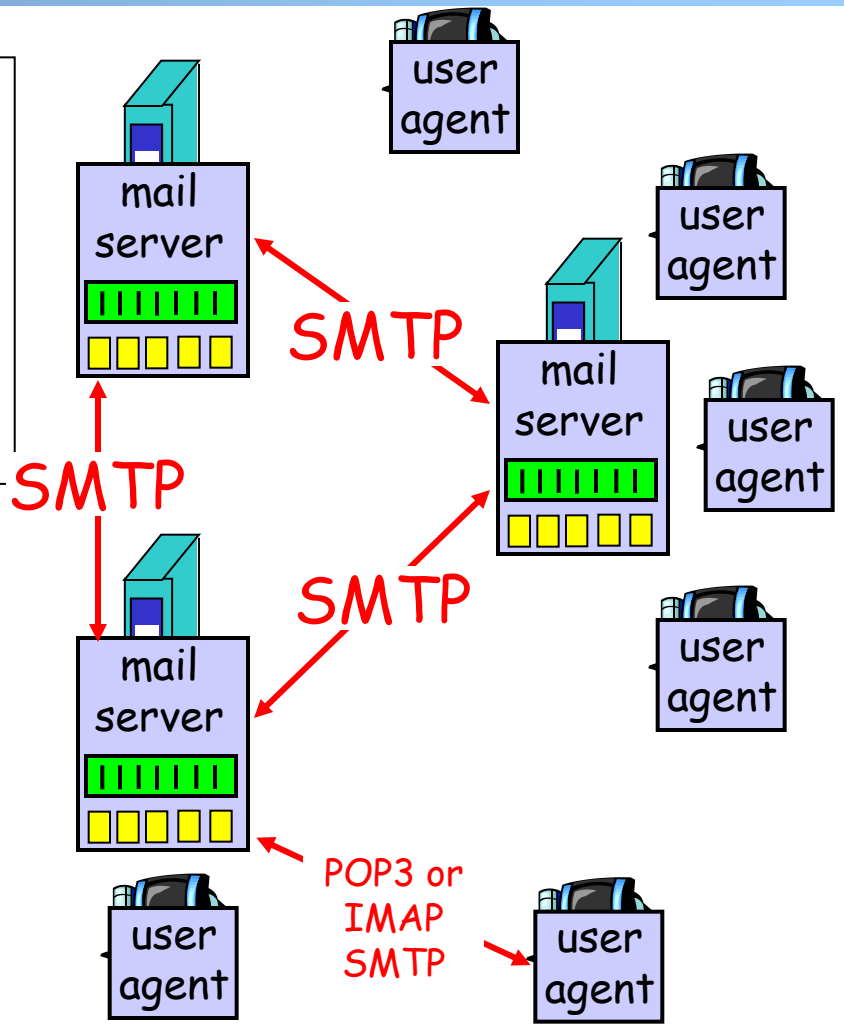
- An application protocol can be designed on top of different transport protocols (TCP, SSL)

- An application C-S protocol typically has state (e.g., first user before pass)—a sequence of commands forming a transaction

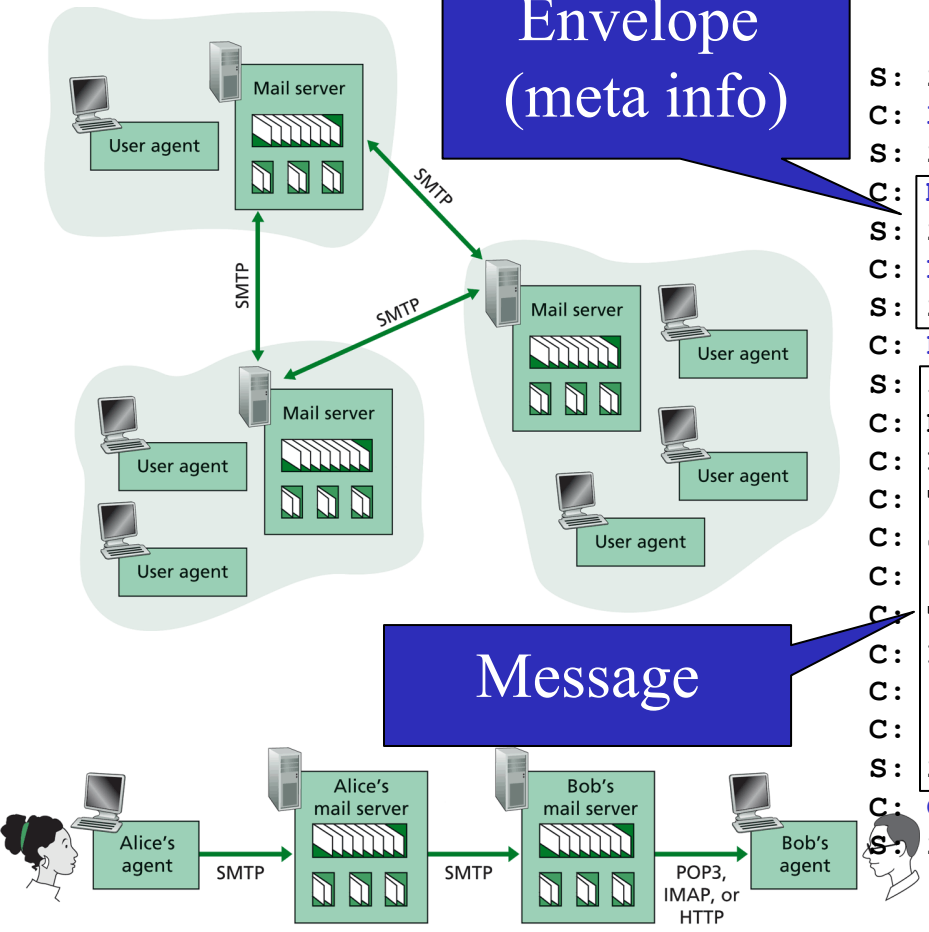
Evaluation of POP

Key questions to ask about a C-S application

- **extensible?**
- **scalable?**
- **robust?**
- **security?**



SMTP: Simple Mail Transport Protocol



Envelope (meta info)

Message

```
S: 220 mr1.its.yale.edu
C: HELO cyndra.cs.yale.edu
S: 250  Hello cyndra.cs.yale.edu, pleased to meet you
C: MAIL FROM: <spoof@cs.yale.edu>
S: 250 spoof@cs.yale.edu... Sender ok
C: RCPT TO: <yry@yale.edu>
S: 250 yry@yale.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Date: Wed, 23 Jan 2008 11:20:27 -0500 (EST)
C: From: "Y. R. Yang" <yry@cs.yale.edu>
C: To: "Y. R. Yang" <yry@cs.yale.edu>
C: Subject: This is subject
C:
C: This is the message body!
C: Please don't spoof!
C:
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 mr1.its.yale.edu closing connection
```

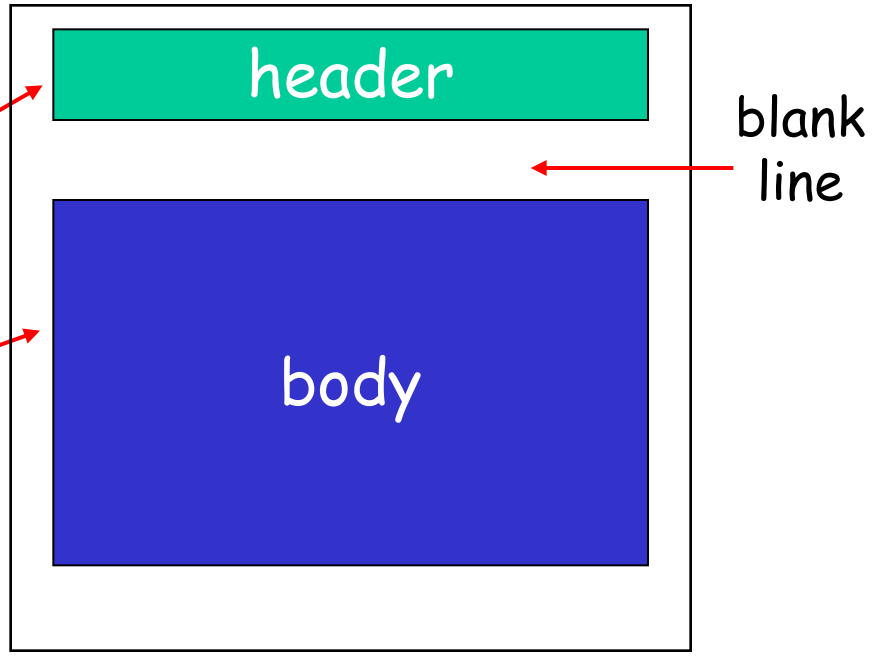
The envelope and message define a mail transport transaction, specifying one client request.

Mail Message

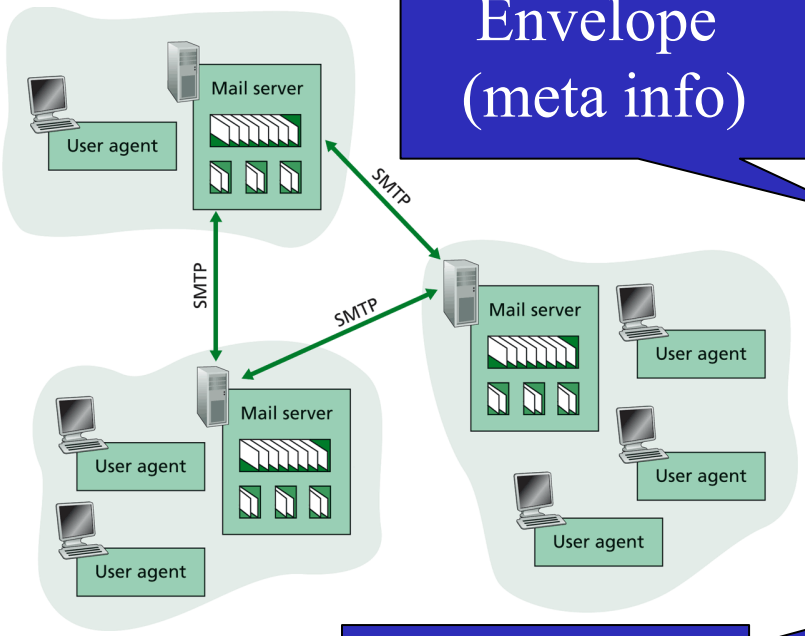
SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

- Header lines, e.g.,
  - To:
  - From:
  - Subject:
- Body
  - the "message", ASCII characters only



SMTP: Simple Mail Transport Protocol



Envelope (meta info)

Message

```
S: 220 mr1.its.yale.edu
C: HELO cyndra.cs.yale.edu
S: 250  Hello cyndra.cs.yale.edu, pleased to meet you
C: MAIL FROM: <yry@cs.yale.edu>
S: 250 yry@cs.yale.edu... Sender ok
C: RCPT TO: <yry@yale.edu>
S: 250 yry@yale.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Date: Wed, 23 Jan 2008 11:20:27 -0500 (EST)
C: From: "Y. R. Yang" <yry@cs.yale.edu>
C: To: "Y. R. Yang" <yry@cs.yale.edu>
C: Subject: This is subject
C:
C: This is the message body!
C: Please don't spoof!
C:
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 mr1.its.yale.edu closing connection
```

Discussions

- Aren't the fields in envelope duplicate of those in message headers?
- What is a main architecture benefit of the envelope-message separation design?
- SMTP is derived from MTP, which has command as MAIL <from@host1> TO <remote@host2>. Which implementation experience might motivate the separation?
- Historically SMTP often produced duplicates [RFC1047]. Why?

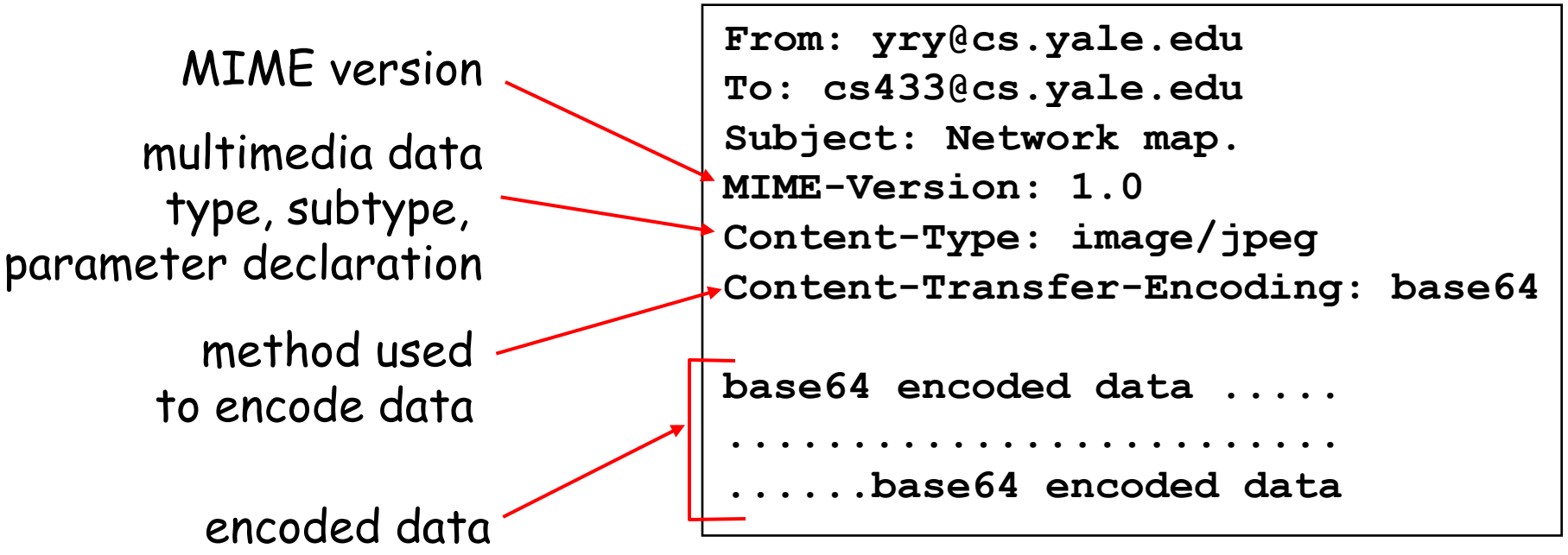
Exercise

- Look at structure of Mail message format

See pop3-trace.txt

Mail Message Extension

- MIME: multimedia mail extension, RFC 2045, 2056
  - Additional lines in msg header declare MIME content type



Benefit of MIME type: self-describing data type, adding extensibility.

Multipart Type: How Attachment Works

```
From: yry@cs.yale.edu
To: cs433@cs.yale.edu
Subject: Network map.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Hi,
Attached is network topology map.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

Evaluation of SMTP

Key questions to ask about a C-S application

- extensible?
  separat. of envelope and msg;
  self-describing message;
  ehlo negotiation

- scalable?
  have not seen mechanism yet

- robust?
  have not seen mechanism yet

- security?
  authentication/authorization
  (spoof, spam) are major issues
  of mail transport