Network Applications: Overview, EMail

Y. Richard Yang

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

1/27/2016

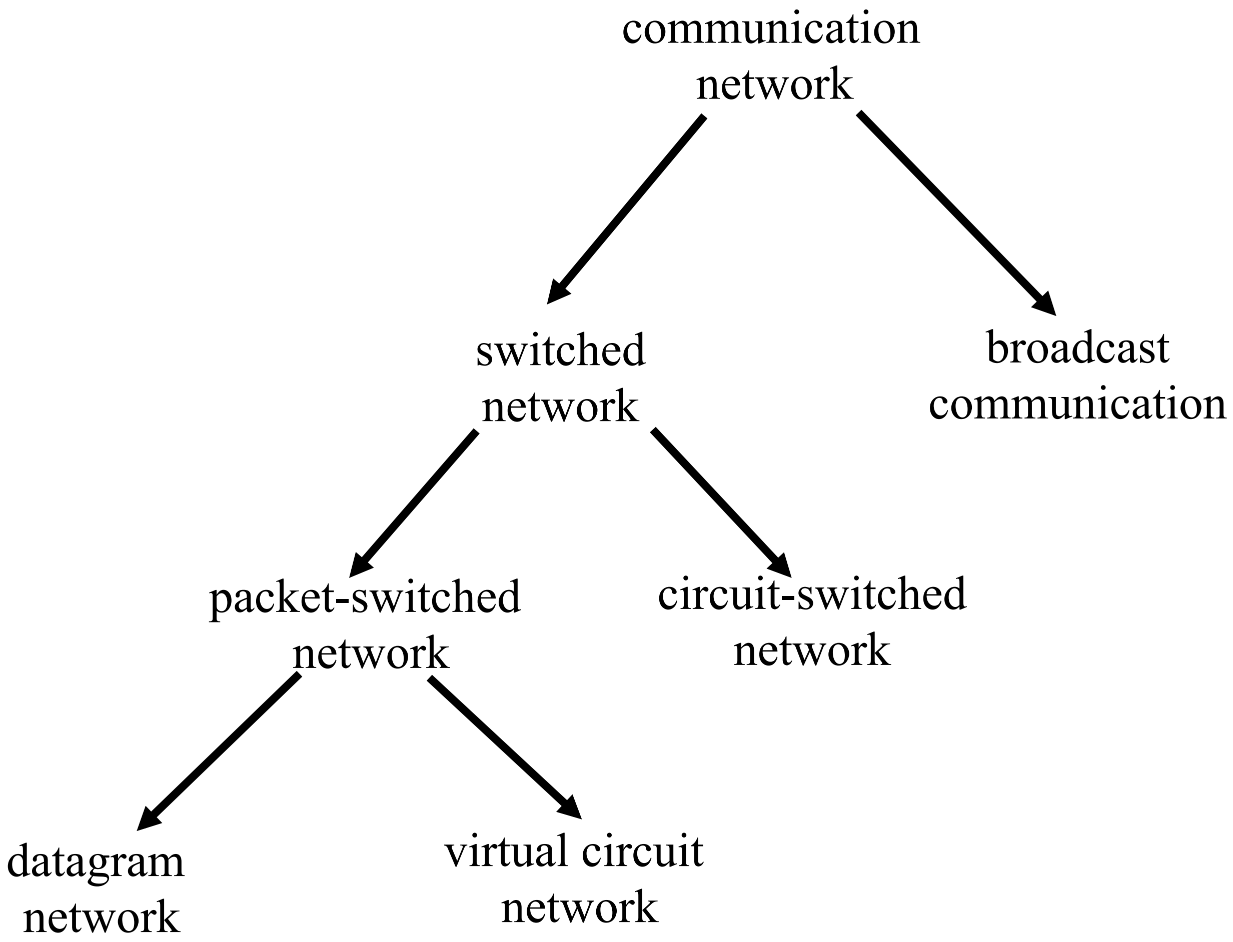
Outline

- ➢ Admin and recap
- ❒ ISO/OSI Layering and Internet Layering
- ❒ Application layer overview
- ❒ Network applications
  - ❍ Email

Admin

- Questions on Assignment One

Recap: Summary of the Taxonomy of Communication Networks


communication network

switched network

broadcast communication

packet-switched network

circuit-switched network

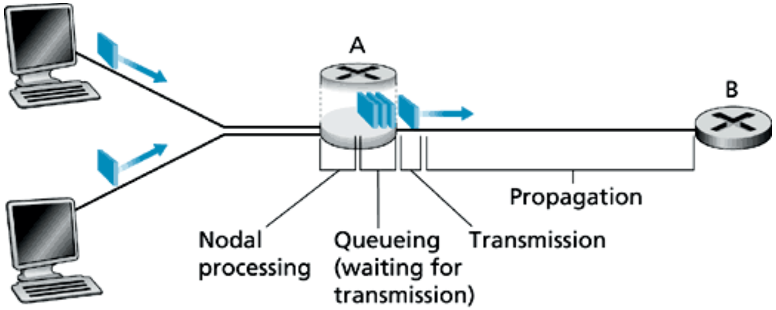
datagram network

virtual circuit network

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
- 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:

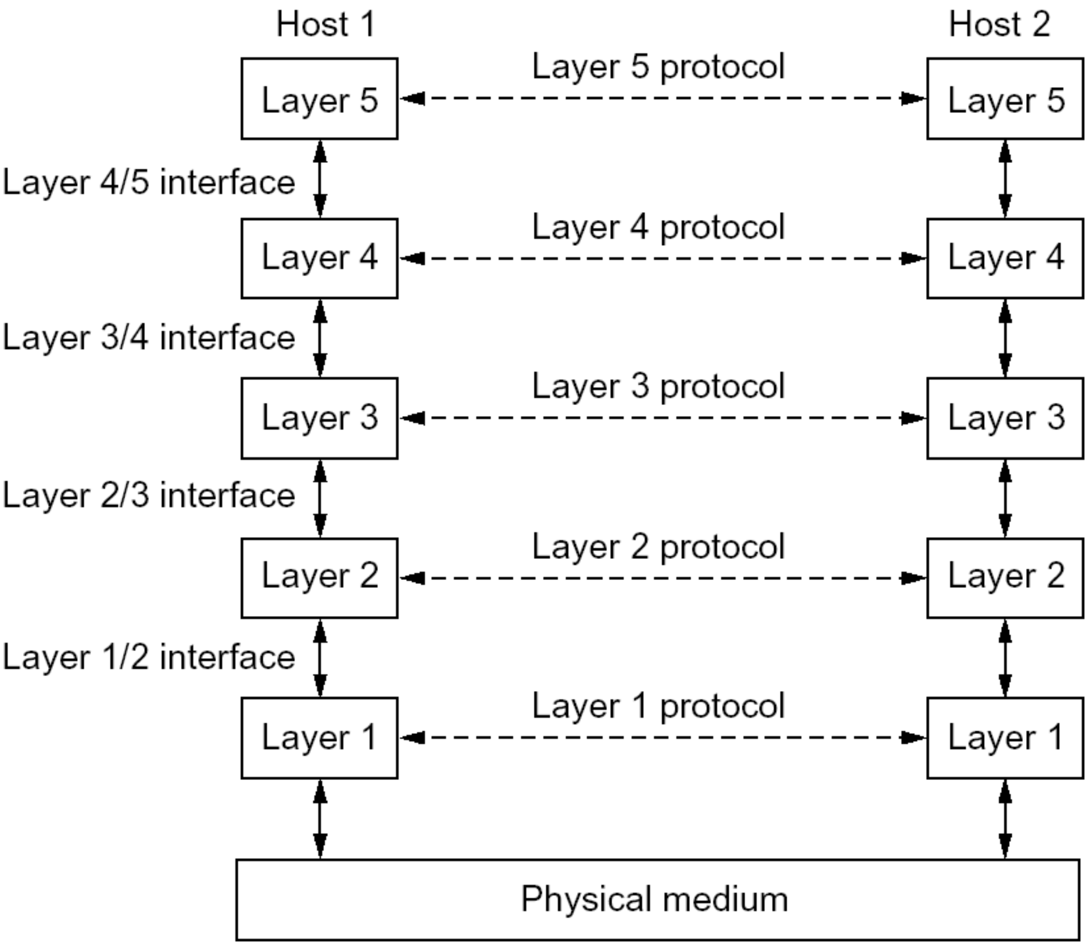
$$\frac{L}{R}\frac{1}{1-\rho}$$

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

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

Recap: Layering

- Why layering
  - reference model
  - modularization
- Concepts
  - service, interface, and protocol
  - physical vs logical communication
- Key design decision: what functionalities to put in each layer: End-to-end arguements

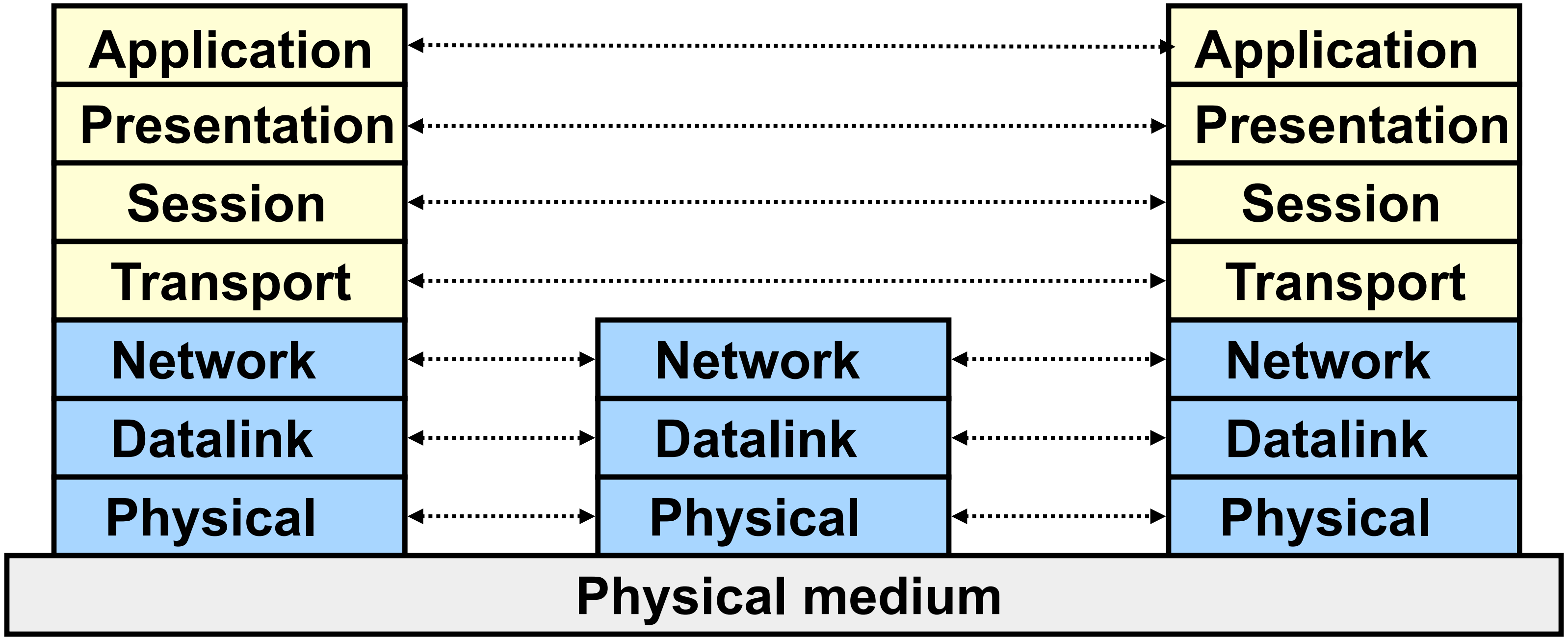


Outline

- Recap
- *ISO/OSI Layering and Internet Layering*
- Application layer overview

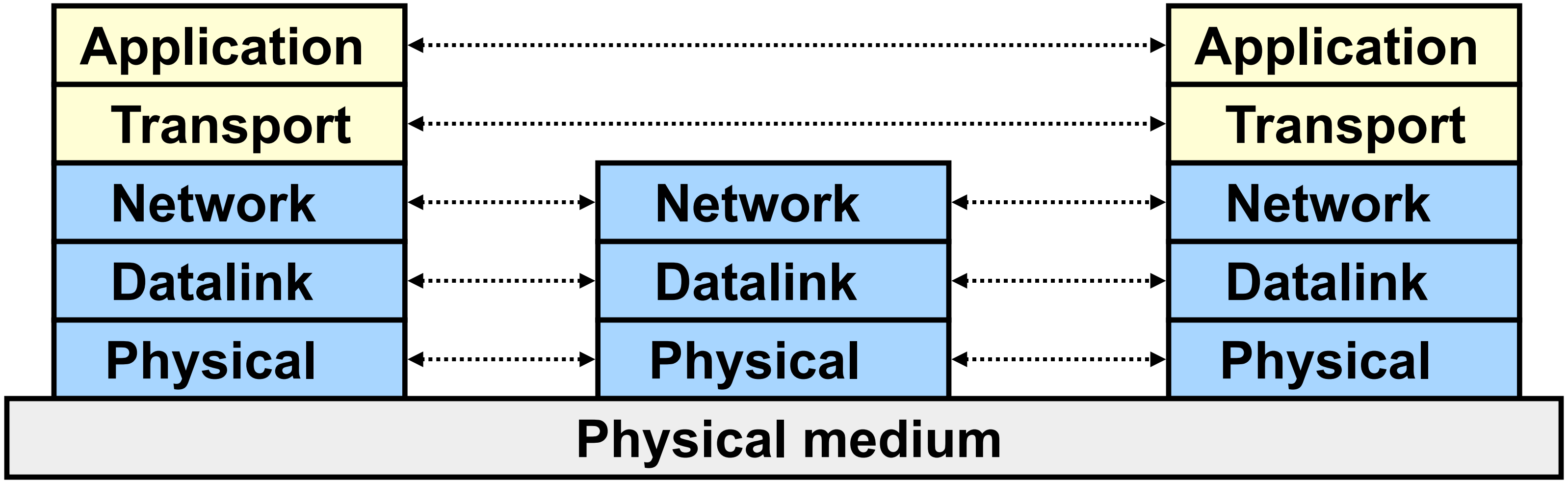
ISO/OSI Reference Model

- Seven layers
  - lower three layers are hop-by-hop
  - next four layers are end-to-end (host-to-host)



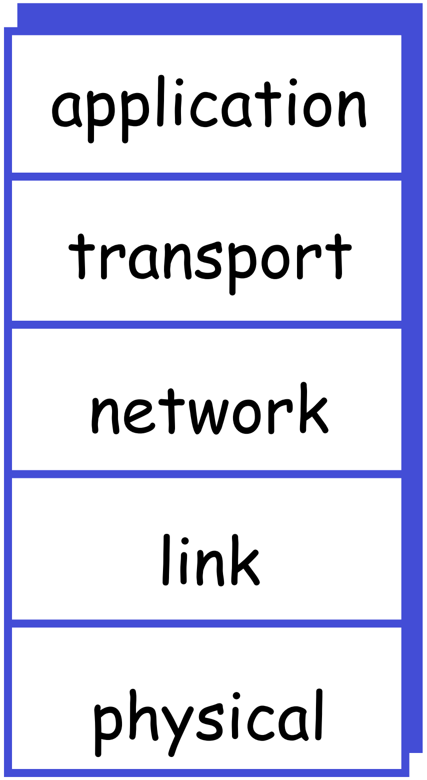
Internet Layering

- Lower three layers are hop-by-hop
- Next two layers are end-to-end



Internet Protocol Layers

- Five layers
  - Application: specific network applications
    - ftp, smtp, http, p2p, IP telephony, …
  - Transport: host-host data transfer
    - tcp (reliable), udp (not reliable)
  - Network: routing of datagram from source to destination
    - ipv4, ipv6
  - Link: data transfer between neighboring network elements
    - ethernet, 802.11, cable, DSL, …
  - Physical: bits “on the wire”
    - cable, wireless, optical fiber


application

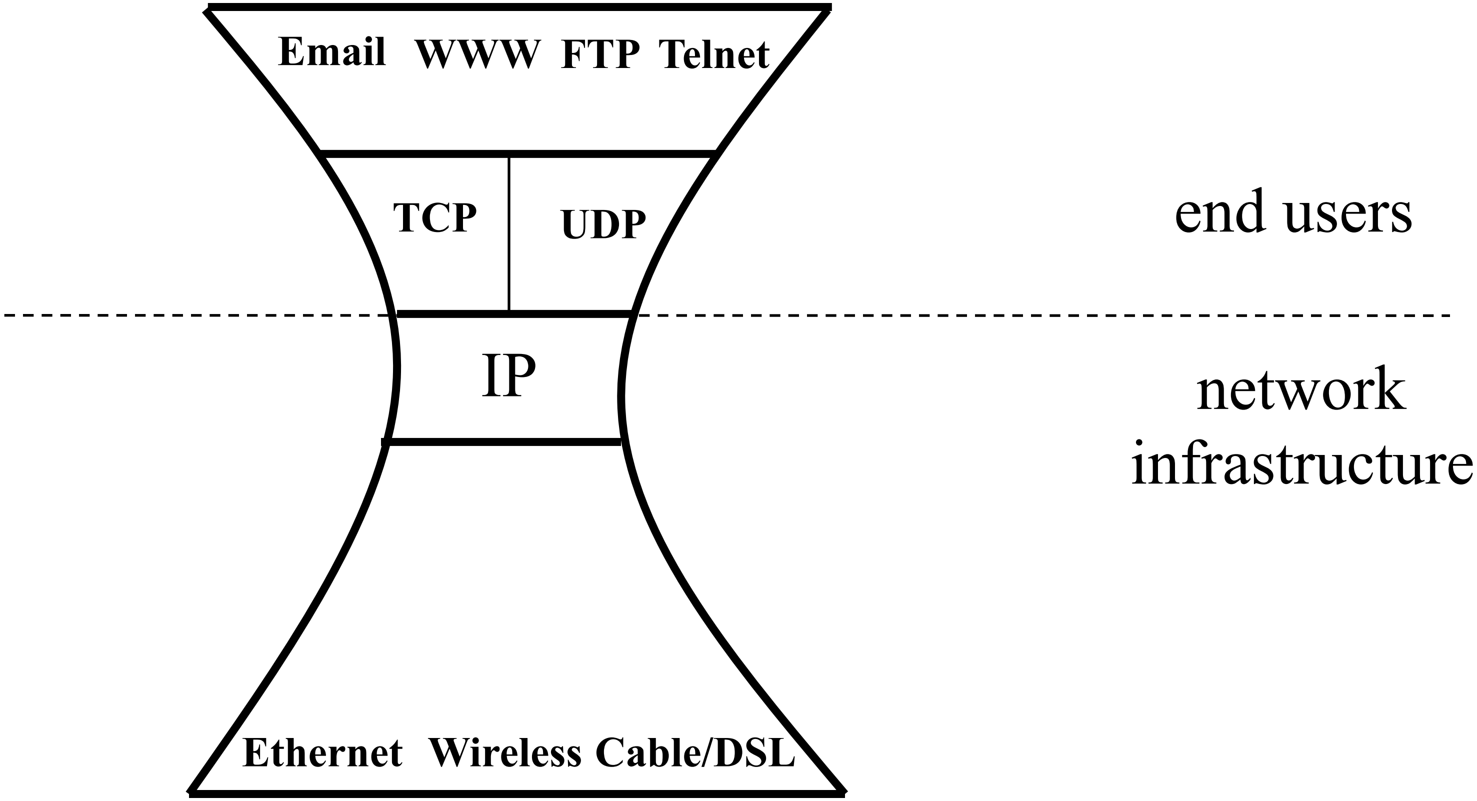
transport

network

link

physical

The Hourglass Architecture of the Internet



Email WWW FTP Telnet

TCP

UDP

IP

Ethernet Wireless Cable/DSL

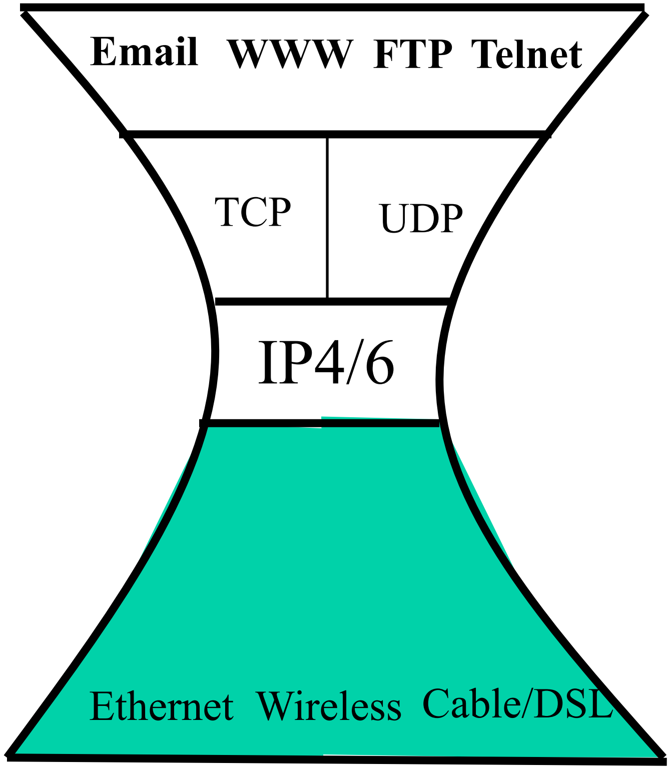
end users

network infrastructure

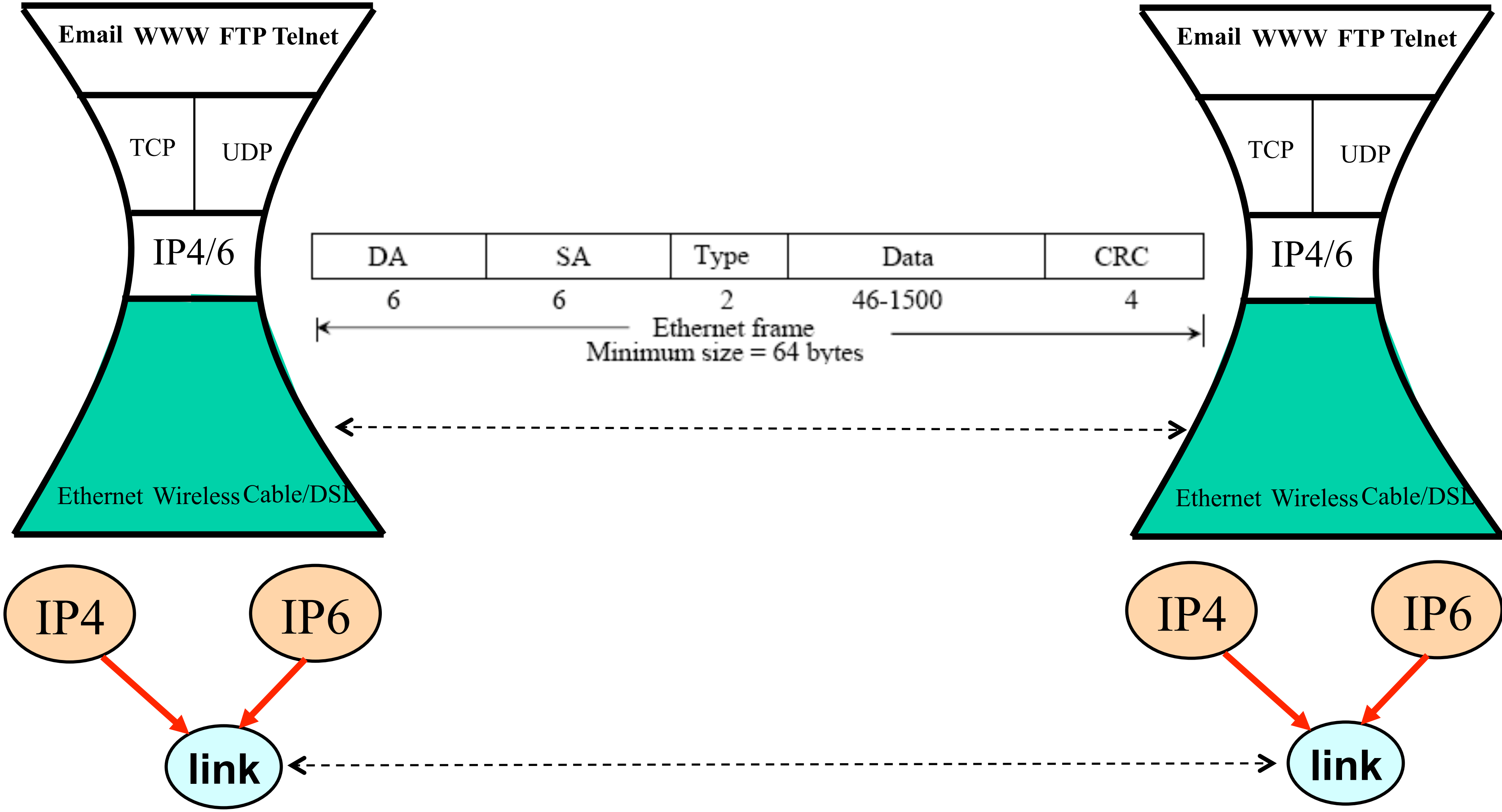
Link Layer (Ethernet)

- Services
  - multiple access control
    - arbitrate access to shared medium
  - multiplexing/ demultiplexing
    - from/to the network layer
  - error detection

- Interface
  - send frames to a directly reachable peer



Link Layer: Protocol Header (Ethernet)

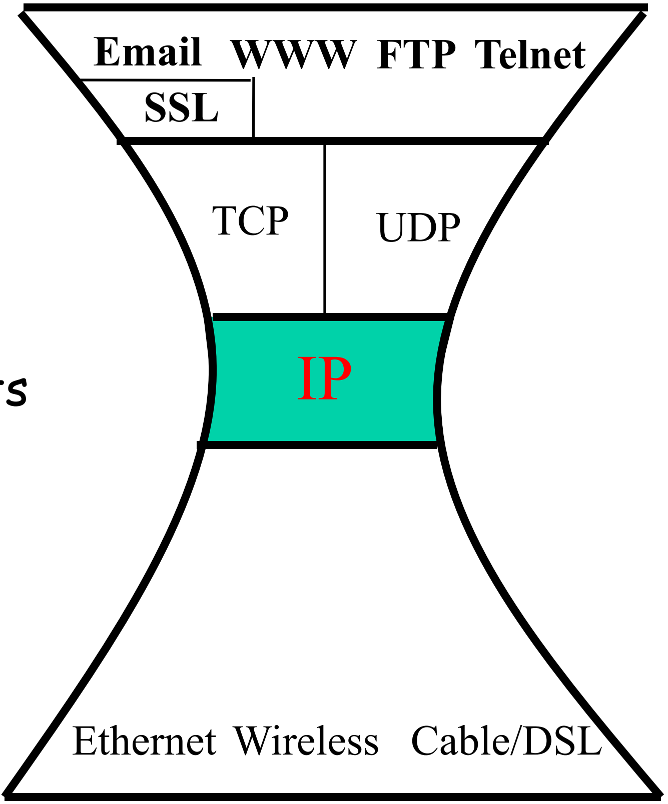


DA	SA	Type	Data	CRC
6	6	2	46-1500	4

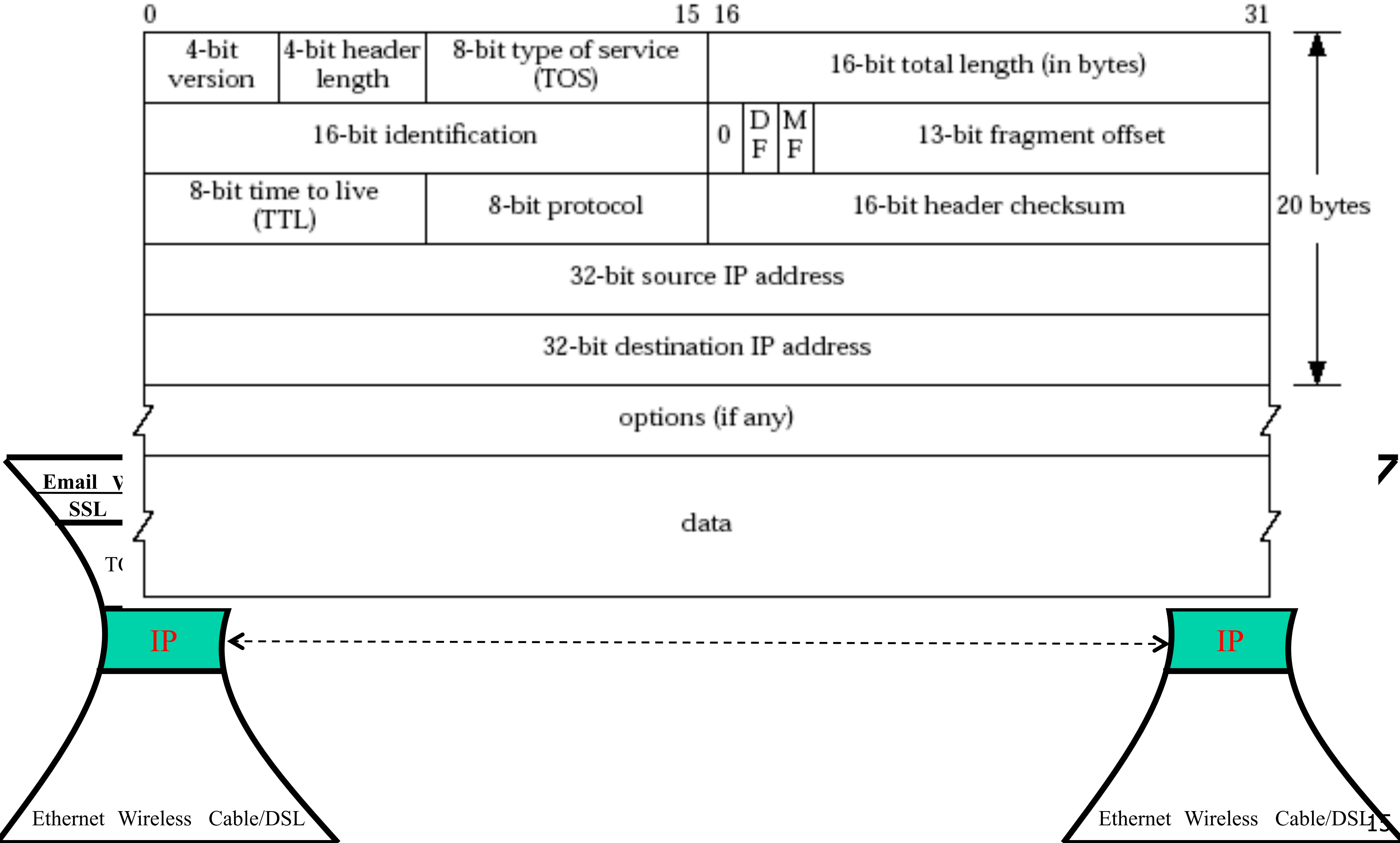
Ethernet frame
Minimum size = 64 bytes

Network Layer: IP

- Services
  - routing: best-effort to send packets from source to destination
  - multiplexing/demultiplexing from/to the transport
  - fragmentation and reassembling: partition a fragment into smaller packets
    - removed in IPv6
  - error detection
  - 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



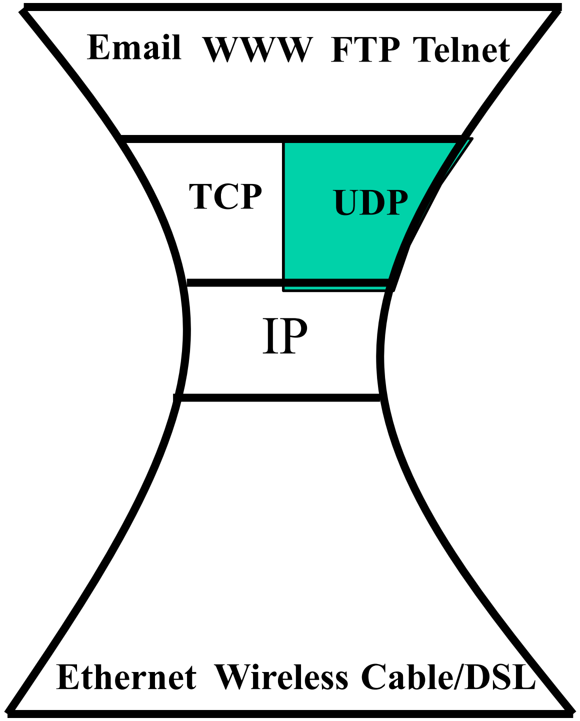
Network Layer: IPv4 Header



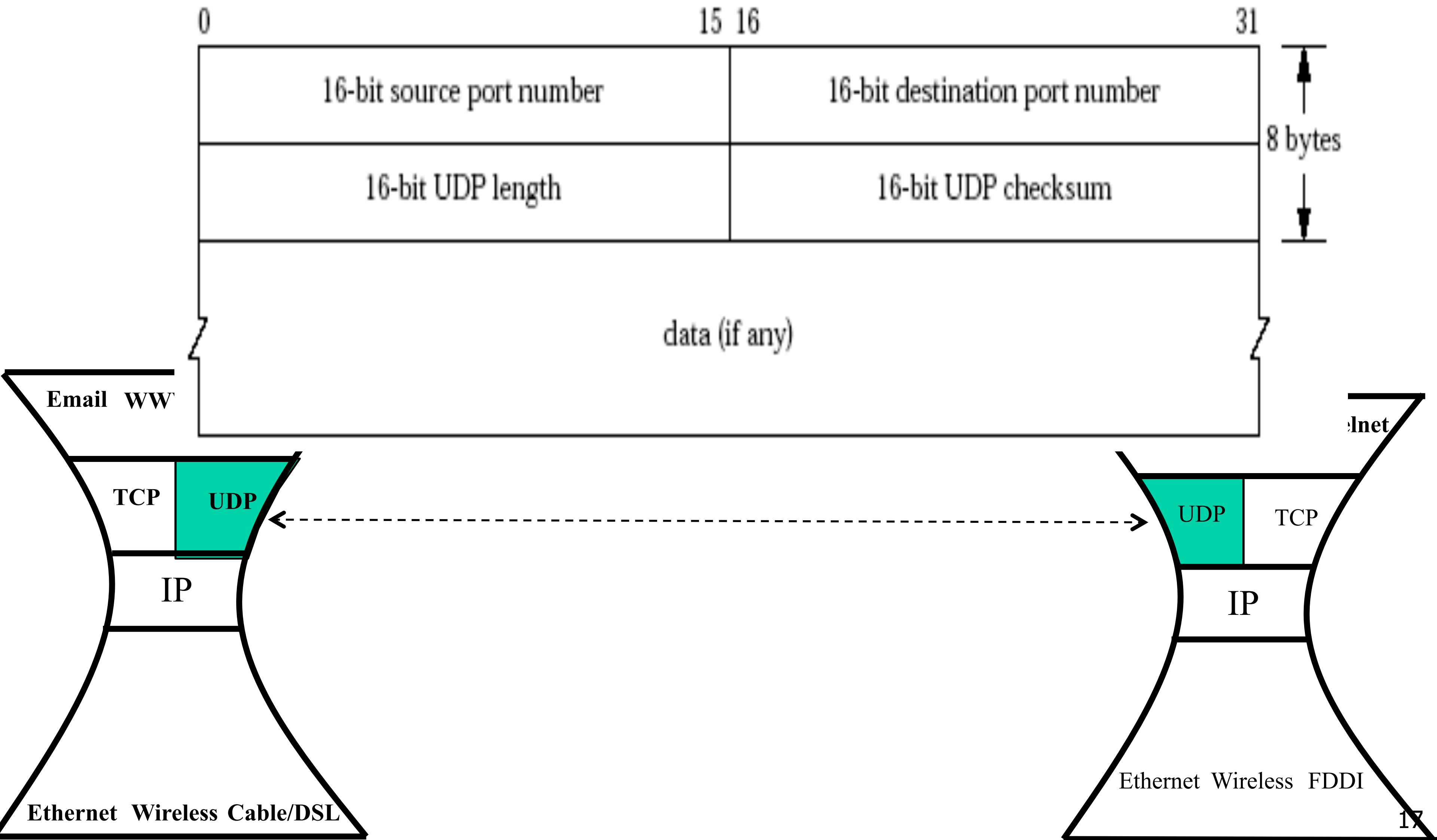
<table>
<tr><td colspan="4">4-bit version</td><td colspan="4">4-bit header length</td><td colspan="8">8-bit type of service (TOS)</td><td colspan="16">16-bit total length (in bytes)</td></tr>
<tr><td colspan="16">16-bit identification</td><td>0</td><td>D F</td><td>M F</td><td colspan="13">13-bit fragment offset</td></tr>
<tr><td colspan="8">8-bit time to live (TTL)</td><td colspan="8">8-bit protocol</td><td colspan="16">16-bit header checksum</td></tr>
<tr><td colspan="32">32-bit source IP address</td></tr>
<tr><td colspan="32">32-bit destination IP address</td></tr>
<tr><td colspan="32">options (if any)</td></tr>
<tr><td colspan="32">data</td></tr>
</table>

Services Provided by UDP

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



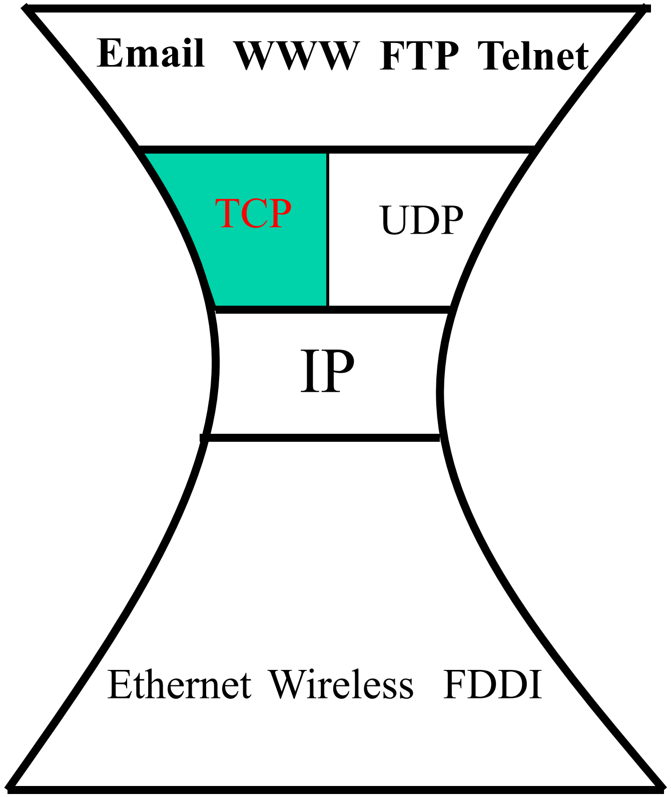
Transport Layer: UDP Header



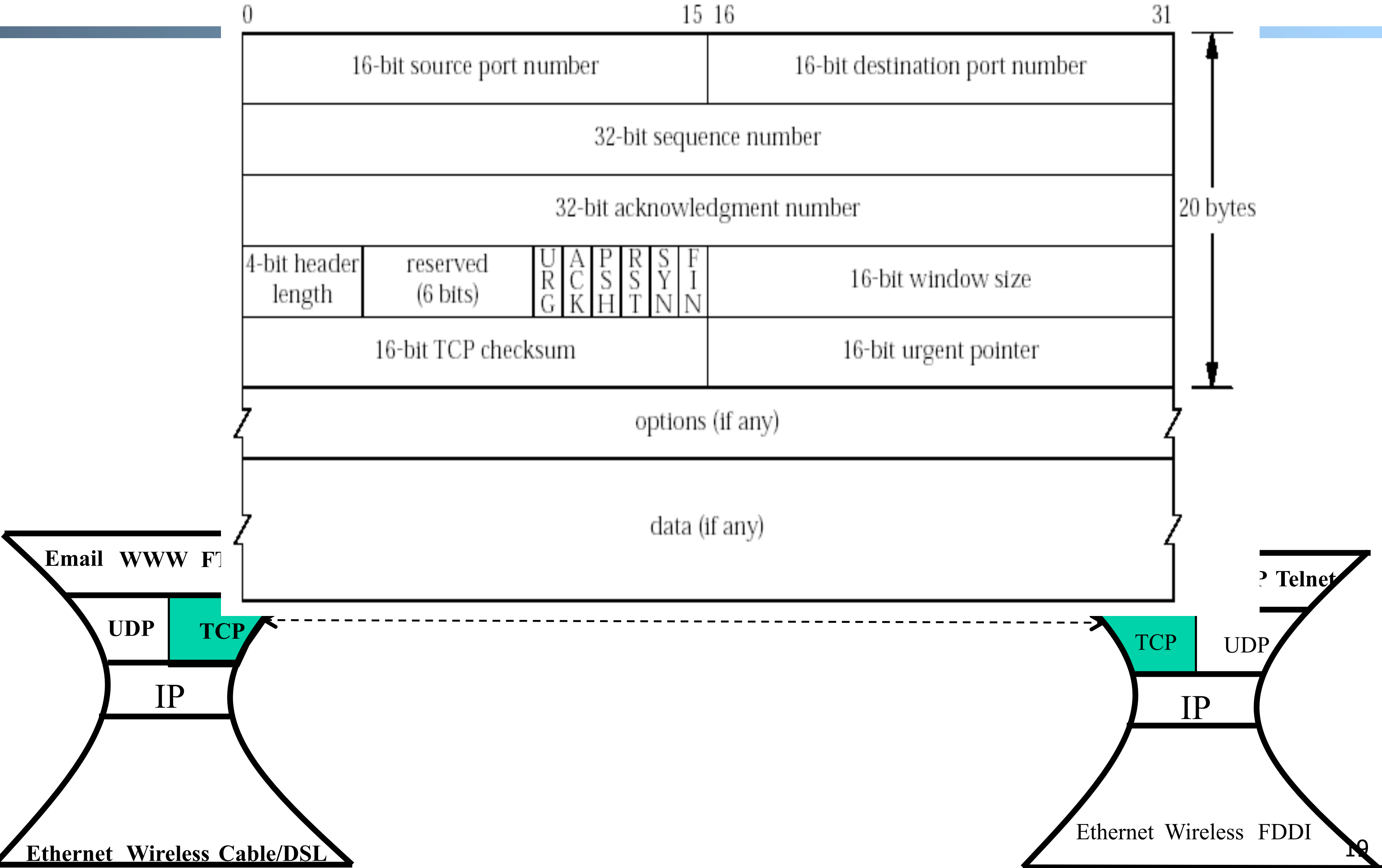
0 15	16 31
16-bit source port number	16-bit destination port number
16-bit UDP length	16-bit UDP checksum
data (if any)	

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



0 15 16 31

16-bit source port number

16-bit destination port number

32-bit sequence number

32-bit acknowledgment number

4-bit header length

reserved (6 bits)

URG ACK PSH RST SYN FIN

16-bit window size

16-bit TCP checksum

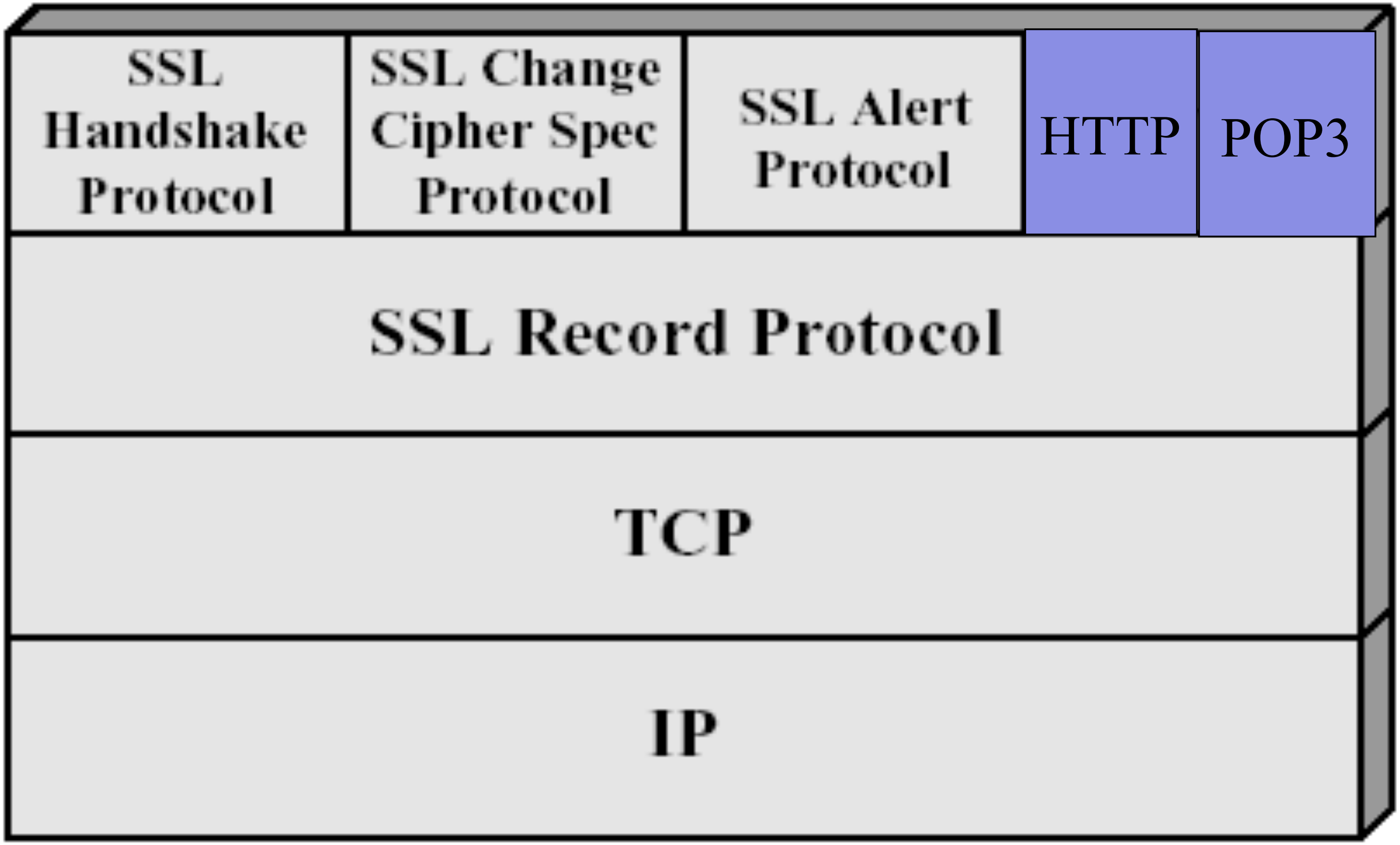
16-bit urgent pointer

20 bytes

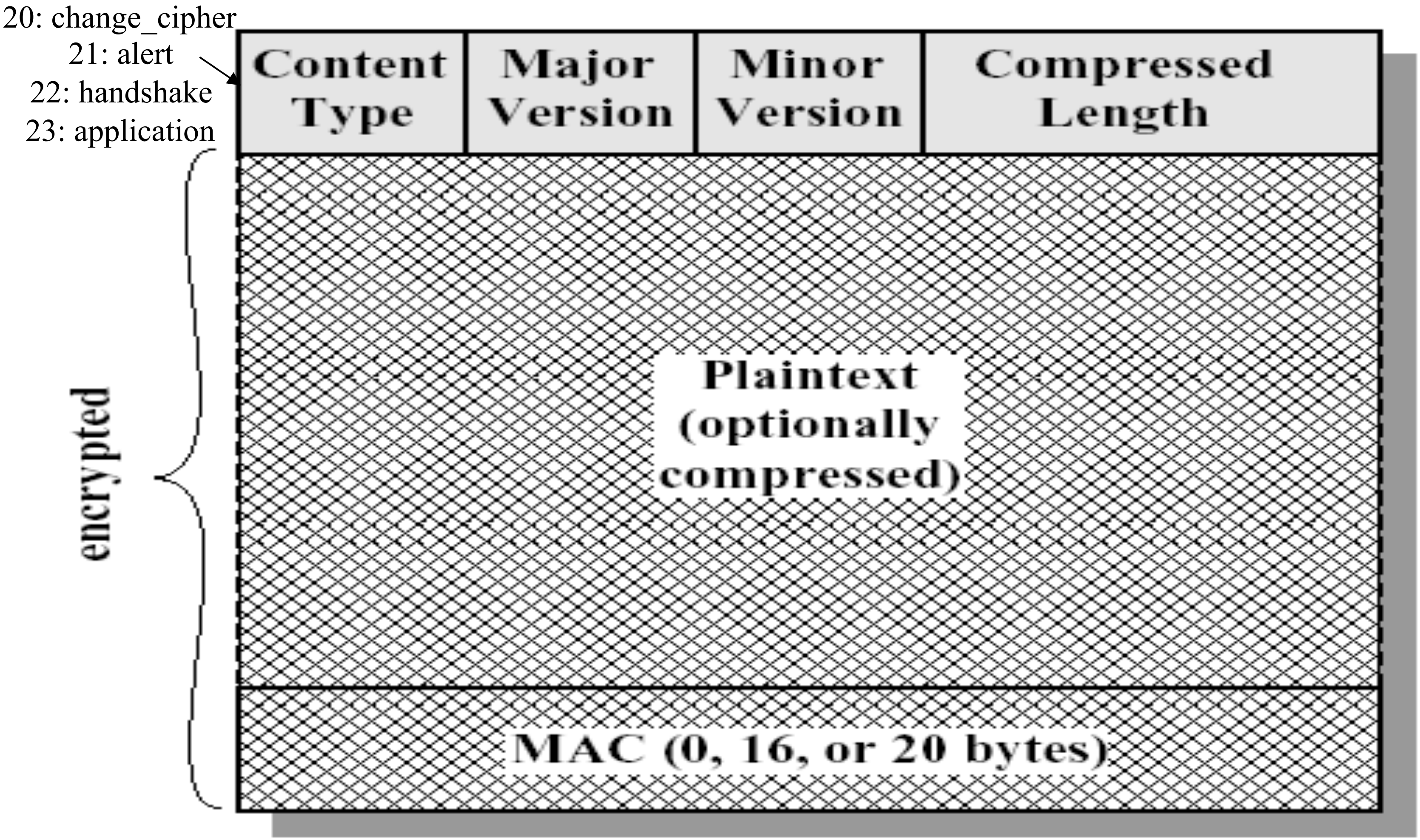
options (if any)

data (if any)

Secure Socket Layer Architecture



SSL Record-Layer Packet Format



20: change_cipher
21: alert
22: handshake
23: application

Content Type	Major Version	Minor Version	Compressed Length

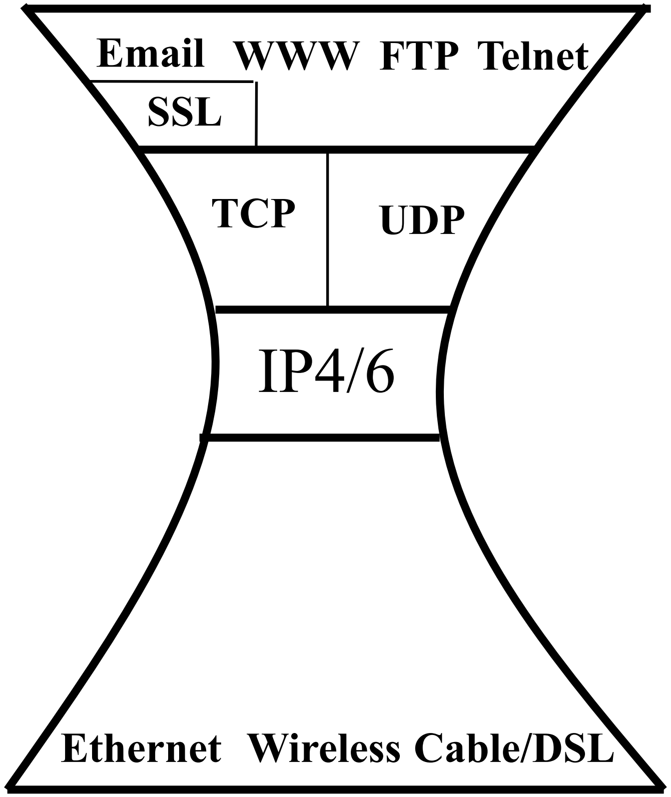
Plaintext (optionally compressed)

MAC (0, 16, or 20 bytes)

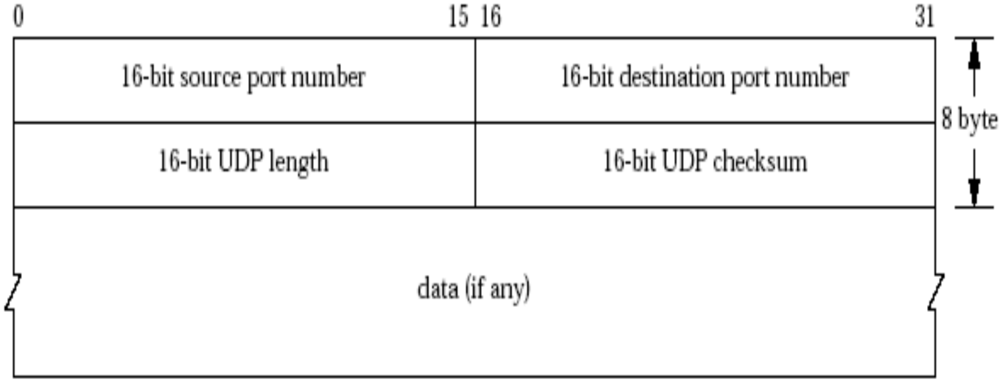
encrypted

Summary: The Big Picture of the Internet

- Hosts and routers:
  - ~ 1 bil. hosts (July 2015)
  - 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



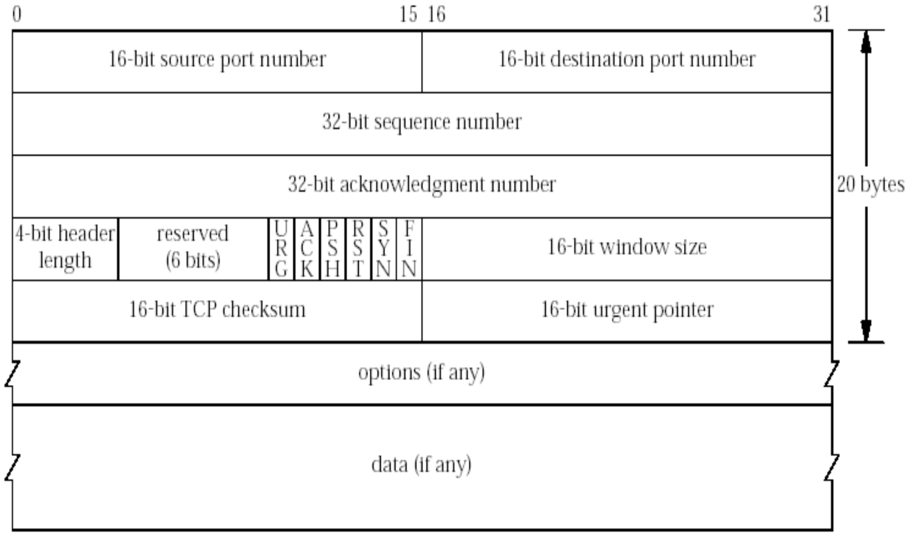
0 15 16 31

16-bit source port number | 16-bit destination port number

16-bit UDP length | 16-bit UDP checksum

8 byte

data (if any)



0 15 16 31

16-bit source port number | 16-bit destination port number

32-bit sequence number

32-bit acknowledgment number

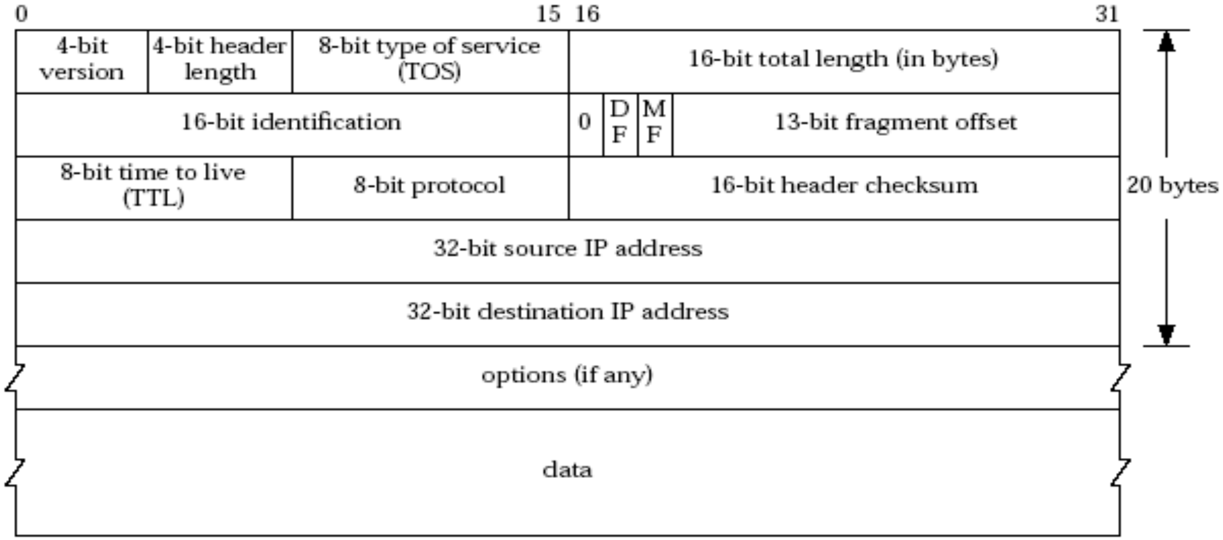
4-bit header length | reserved (6 bits) | URG | ACK | PSH | RST | SYN | FIN | 16-bit window size

16-bit TCP checksum | 16-bit urgent pointer

20 bytes

options (if any)

data (if any)



0 15 16 31

4-bit version | 4-bit header length | 8-bit type of service (TOS) | 16-bit total length (in bytes)

16-bit identification | 0 | DF | MF | 13-bit fragment offset

8-bit time to live (TTL) | 8-bit protocol | 16-bit header checksum

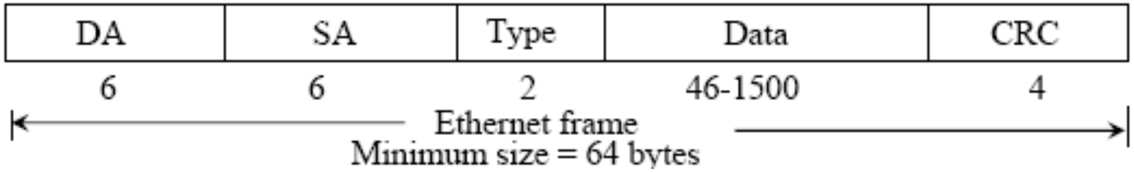
32-bit source IP address

32-bit destination IP address

20 bytes

options (if any)

data



DA	SA	Type	Data	CRC
6	6	2	46-1500	4

Ethernet frame
Minimum size = 64 bytes

Outline

- Recap
- ISO/OSI Layering and Internet Layering
- *Application layer overview*

Application Layer: Goals

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

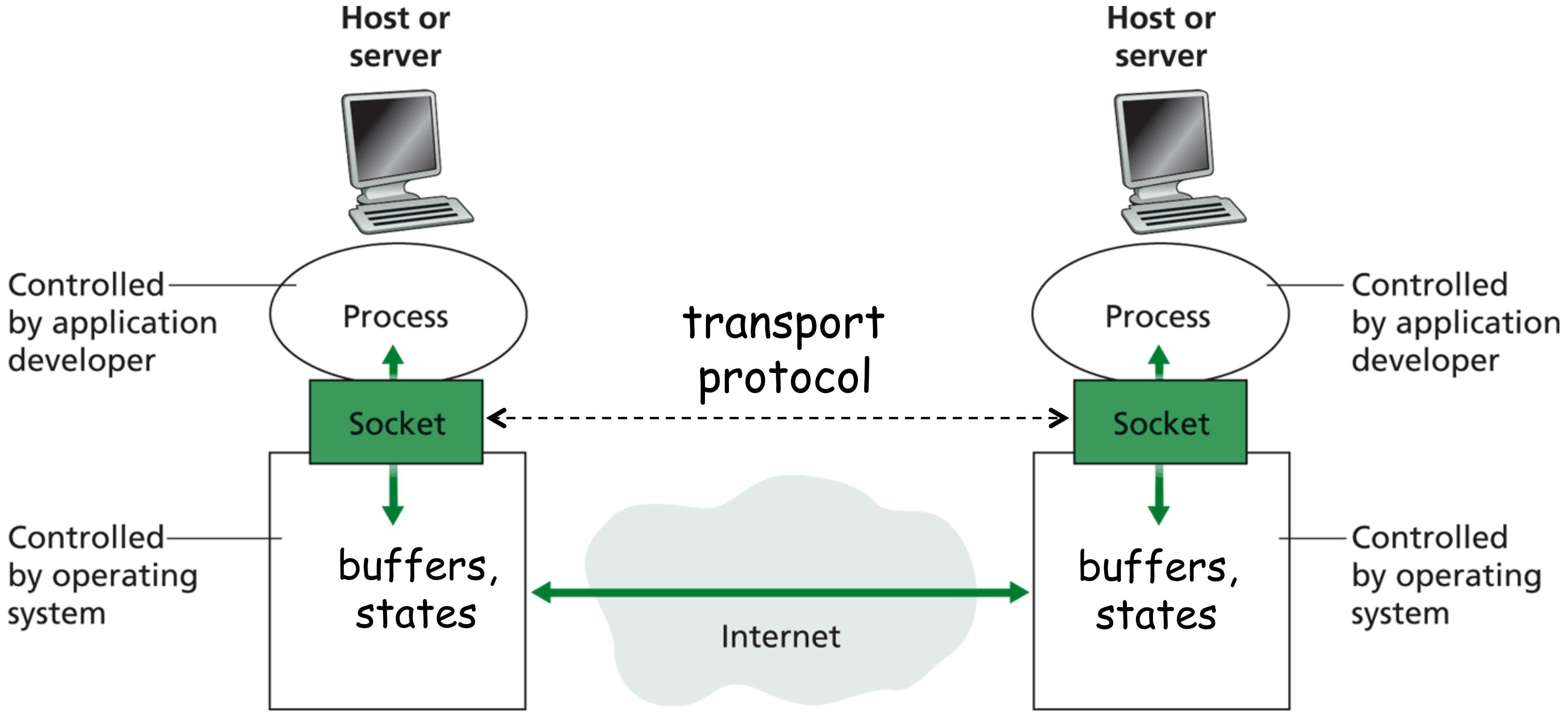
- Learn about applications by examining common applications
  - smtp/pop
  - dns
  - http
  - content distribution

How does an Application Access the Transport Service?

API: application programming interface

- Defines interface between application and transport layer
- Multiple APIs proposed in history
  - XTI (X/Open Transport Interface), a slight modification of the Transport Layer Interface (TLI) developed by AT&T.
- Commonly used: Socket API
  - sometimes called "Berkeley sockets" acknowledging their heritage from Berkeley Unix
  - a socket has a network-layer host IP address and a transport-layer local port number
    - e.g., email (SMTP) port number 25, web port number 80
  - an application process binds to a socket
    - %netstat or lsof
  - two processes communicate by sending data into socket, reading data out of socket

Socket API


Host or server
Host or server
Controlled by application developer
Process
transport protocol
Process
Controlled by application developer
Socket
Socket
Controlled by operating system
buffers, states
Internet
buffers, states
Controlled by operating system

App. and Trans.: 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

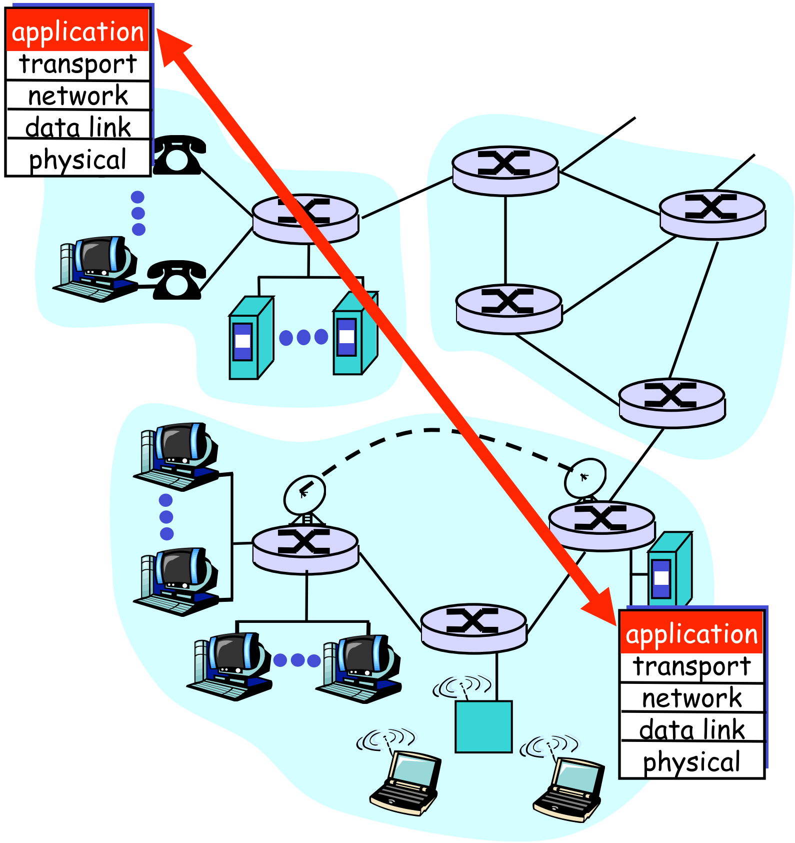
Network Applications vs. Application-layer Protocols

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



Client-Server Paradigm

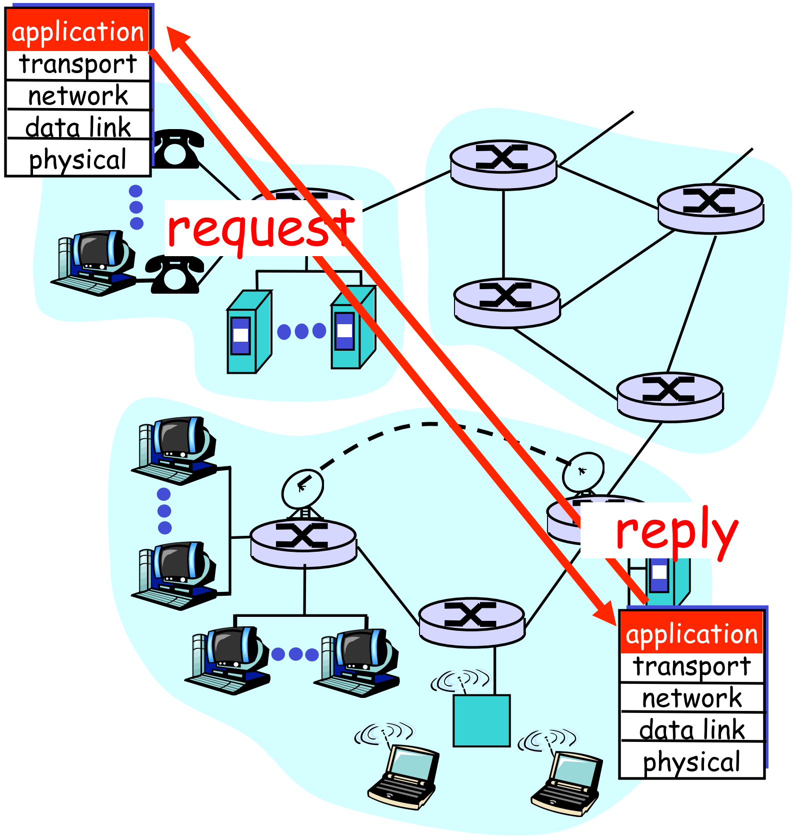
Typical network app has two pieces: *client* and *server*

Client (C):

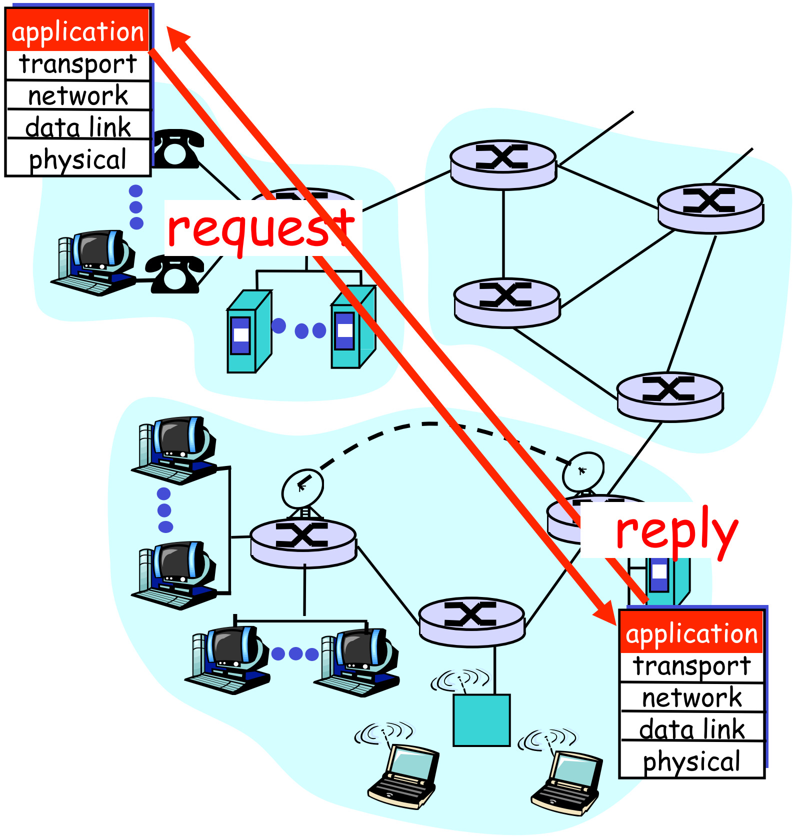
- initiates contact with server ("speaks first")
- typically requests service from server
- for Web, client is implemented in browser; for e-mail, in mail reader

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

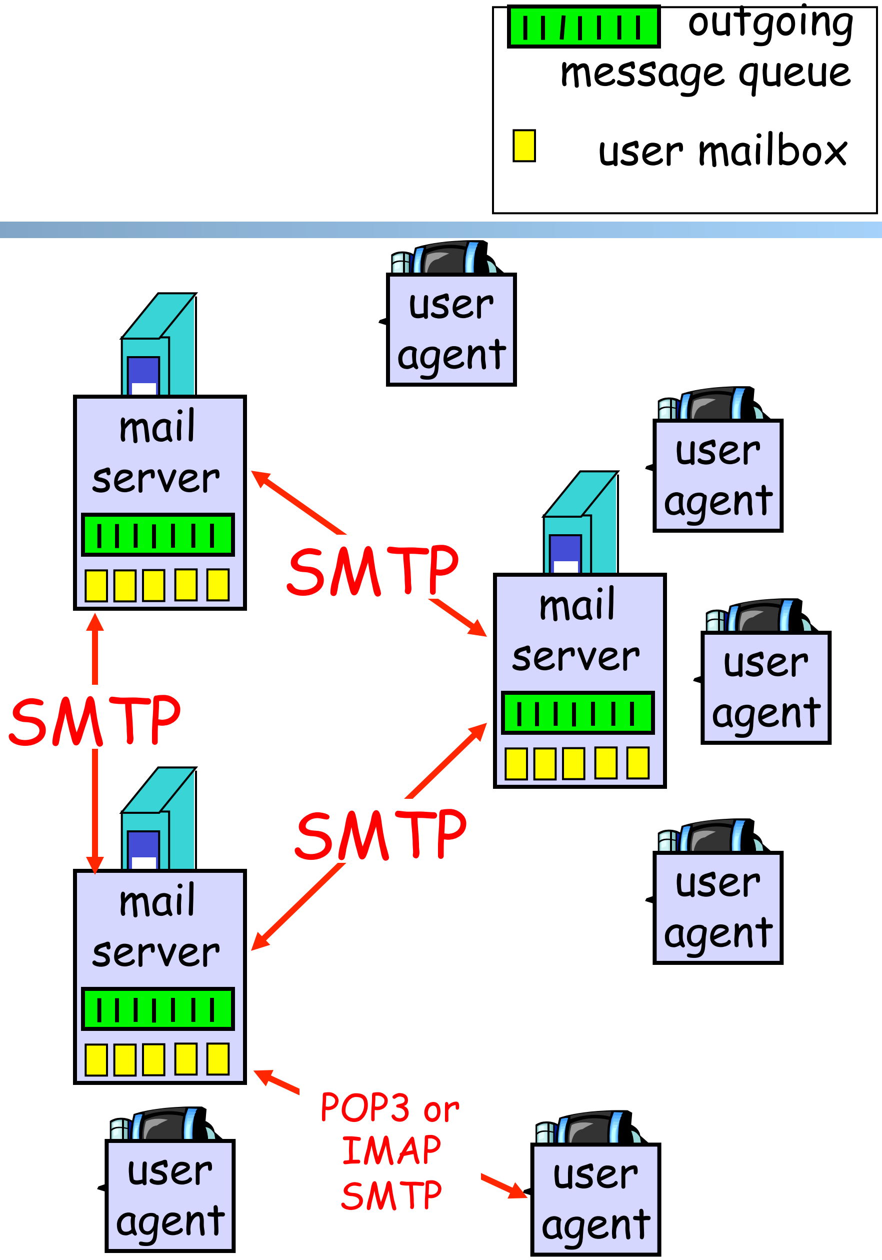
- Is the application **extensible**?
- Is the application **scalable**?
- How does the application handle server failures (being **robust**)?
- How does the application provide **security**?

Electronic Mail

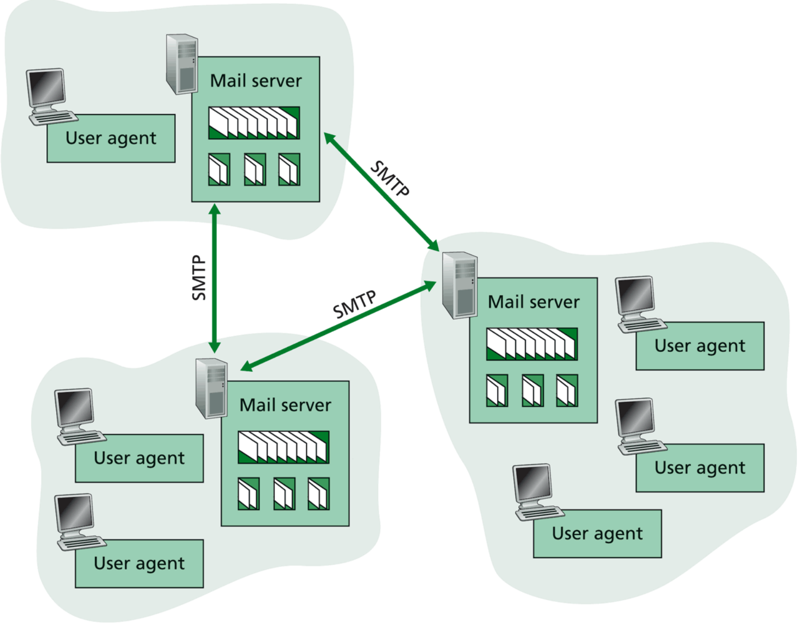
- Still active
  - 80B emails/day
  - 3.9B active email boxes

Three major components:

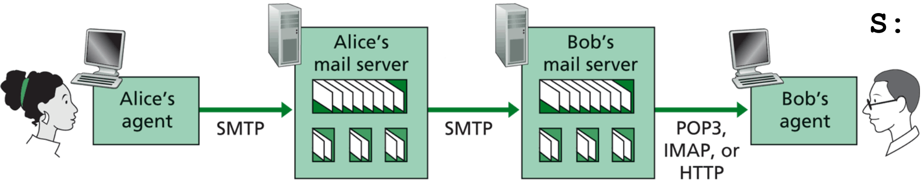
- User agents
- Mail servers
- Protocols
  - Outgoing email
    - SMTP
  - Retrieving email
    - POP3: Post Office Protocol [RFC 1939]
    - IMAP: Internet Mail Access Protocol [RFC 1730]



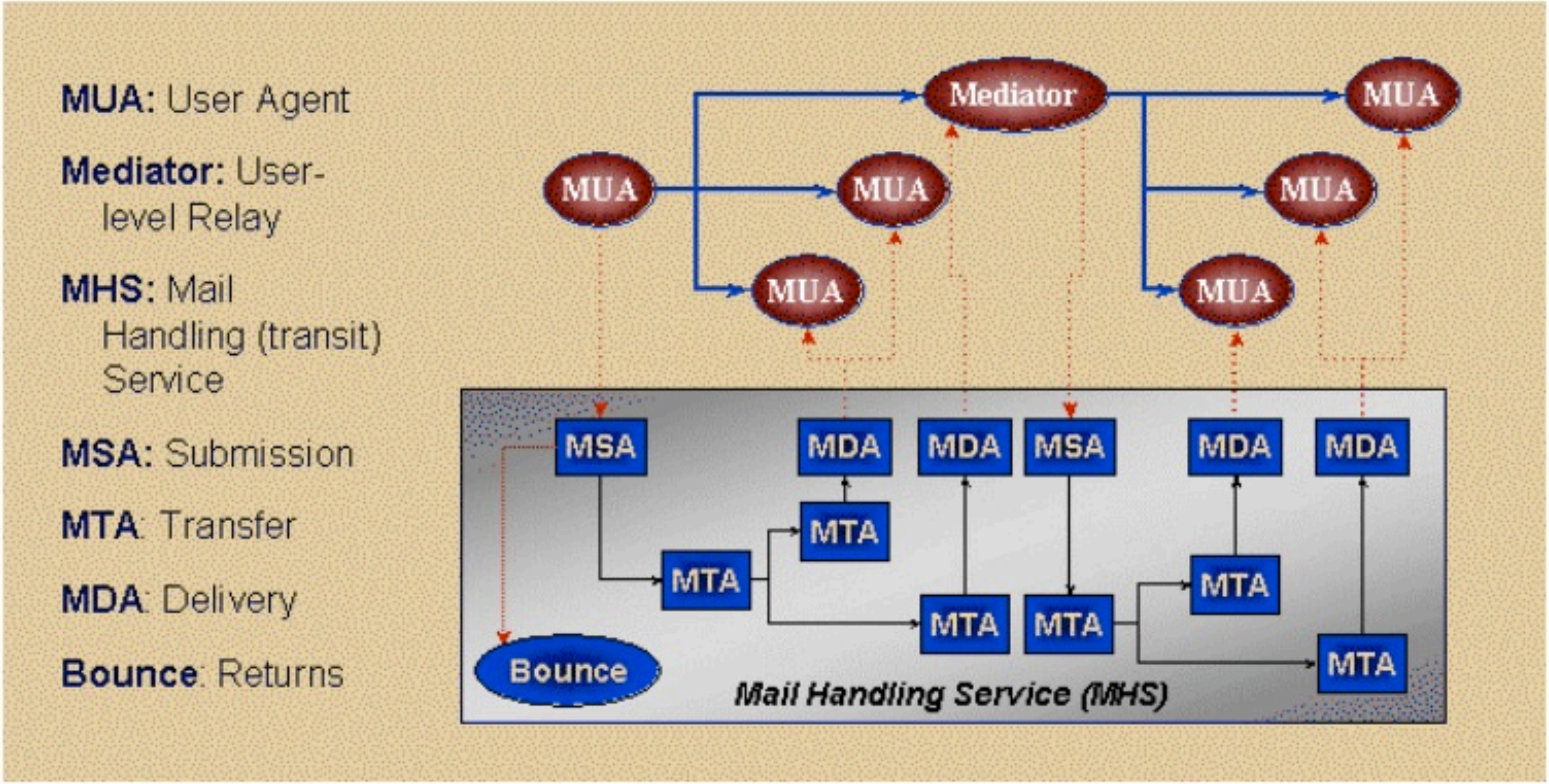
SMTP: Outgoing Email as a Client-Server Application



```
S: 220 mr1.its.yale.edu
C: HELO cyndra.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
```



Email Transport Architecture



MUA: User Agent

Mediator: User-level Relay

MHS: Mail Handling (transit) Service

MSA: Submission

MTA: Transfer

MDA: Delivery

Bounce: Returns

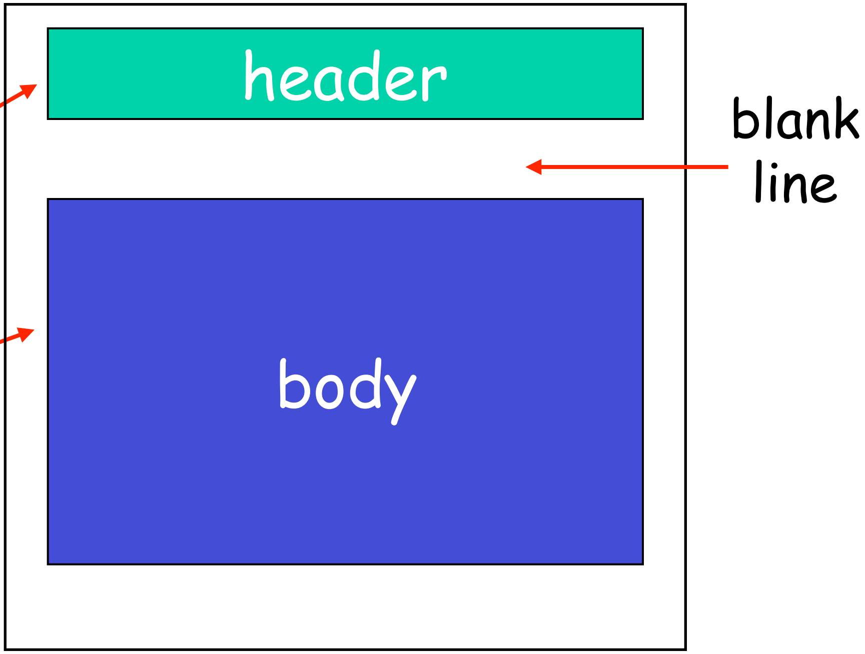
http://www.maawg.org/sites/maawg/files/news/MAAWG_Email_Authentication_Paper_2008-07.pdf

Mail Message Data Format

SMTP: protocol for exchanging email msgs

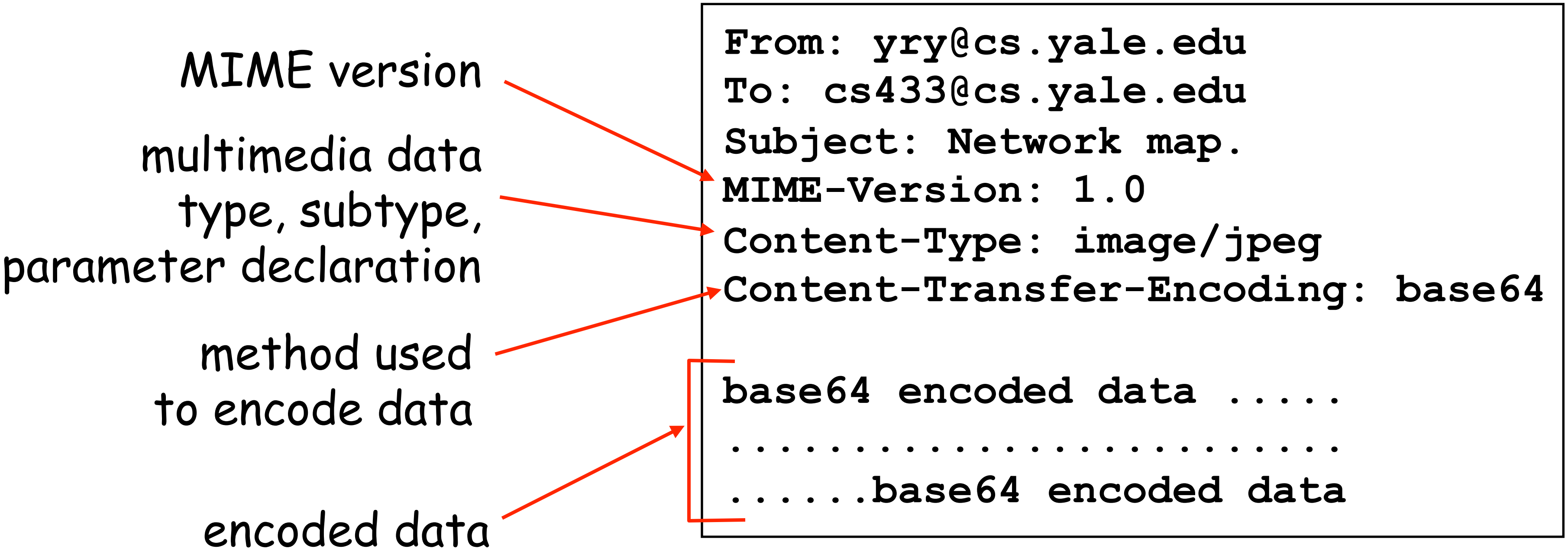
RFC 822: standard for text message format:

- Header lines, e.g.,
  - To:
  - From:
  - Subject:
- Body
  - the "message", ASCII characters only (any problem?)



Message Format: Multimedia Extensions

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



MIME version

multimedia data type, subtype, parameter declaration

method used to encode data

encoded data

```
From: yry@cs.yale.edu
To: cs433@cs.yale.edu
Subject: Network map.
MIME-Version: 1.0
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

base64 encoded data .....
.........................
......base64 encoded data
```

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--
```

Design Review

```
S: 220 mr1.its.yale.edu
C: HELO cyndra.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: From: yry@cs.yale.edu
C: To: cs433@cs.yale.edu
C: Subject: Network map.
C: MIME-Version: 1.0
C: Content-Type: image/jpeg
C: Content-Transfer-Encoding: base64
C:
C: base64 encoded data .....
C: .........................
C: ......base64 encoded data
C:
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 mr1.its.yale.edu closing connection
```

Why not make the msg headers smtp headers?

POP3 Protocol: Mail Retrieval

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`

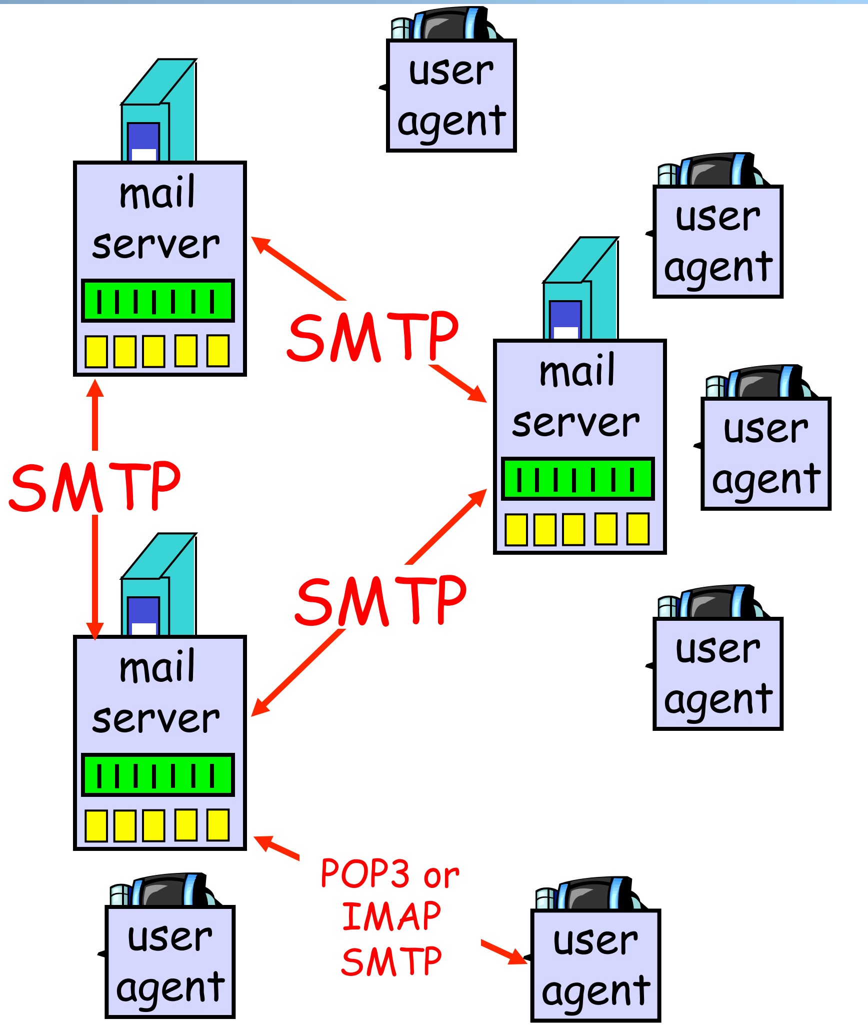
```
C:  list
S:  1 498
S:  2 912
S:  .
C:  retr 1
S:  <message 1 contents>
S:  .
C:  dele 1
C:  retr 2
S:  <message 1 contents>
S:  .
C:  dele 2
C:  quit
S:  +OK POP3 server signing off
```

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

Evaluation of SMTP/POP/IMAP

Key questions to ask about a C-S application

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

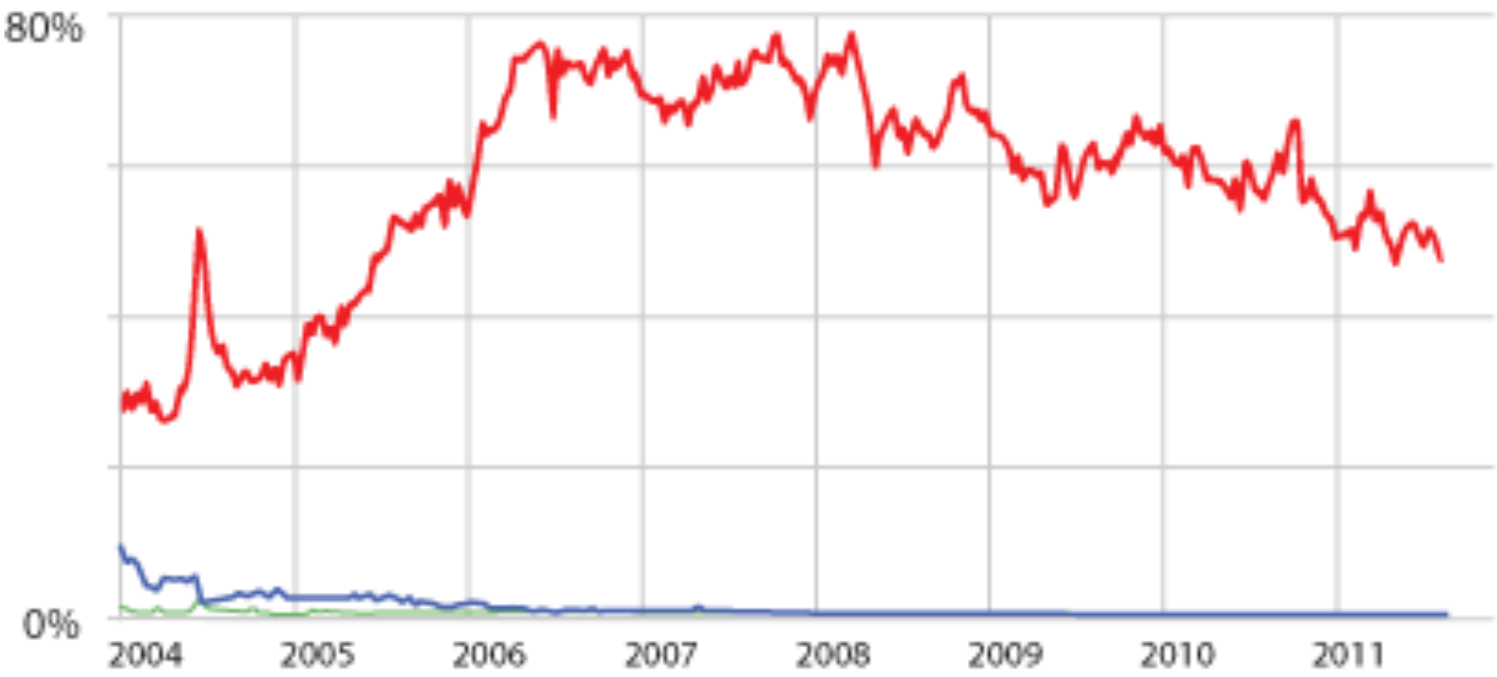


Email: Positive

- Some nice design features we can learn from the design of the email
  - separate protocols for different functions
    - email retrieval (e.g., POP3, IMAP)
    - email transmission (SMTP)
  - simple/basic requests to implement basic control; fine-grain control through ASCII header and message body
    - make the protocol easy to read/debug/extend (analogy with end-to-end layered design?)
  - status code in response makes message easy to parse

Email: Challenge

- Spam (Google)

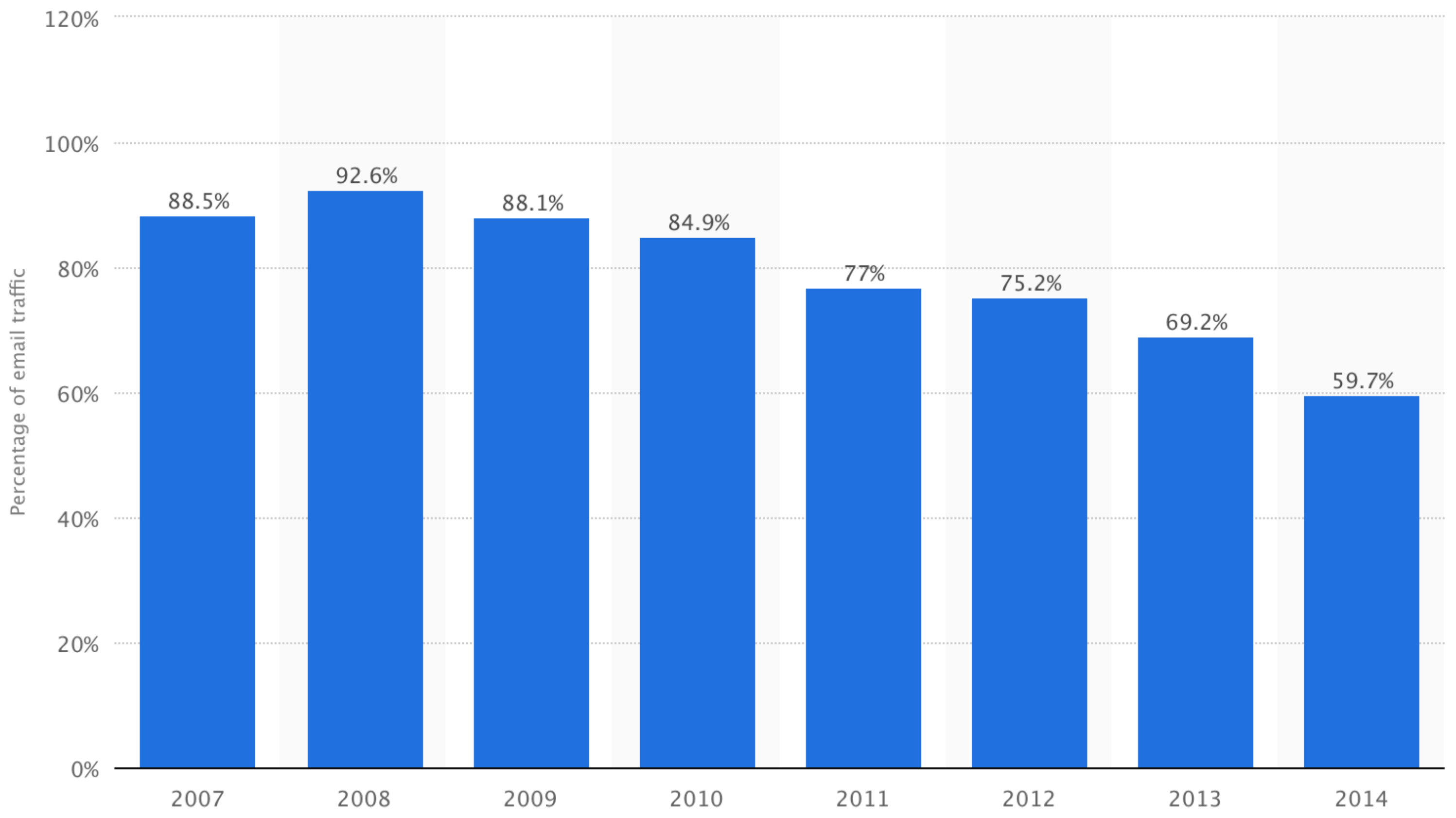


https://mail.google.com/intl/en/mail/help/fightspam/spamexplained.html

Email: Challenge

- A large percentage of spam/phish

Global spam volume as percentage of total e-mail traffic from 2007 to 2014

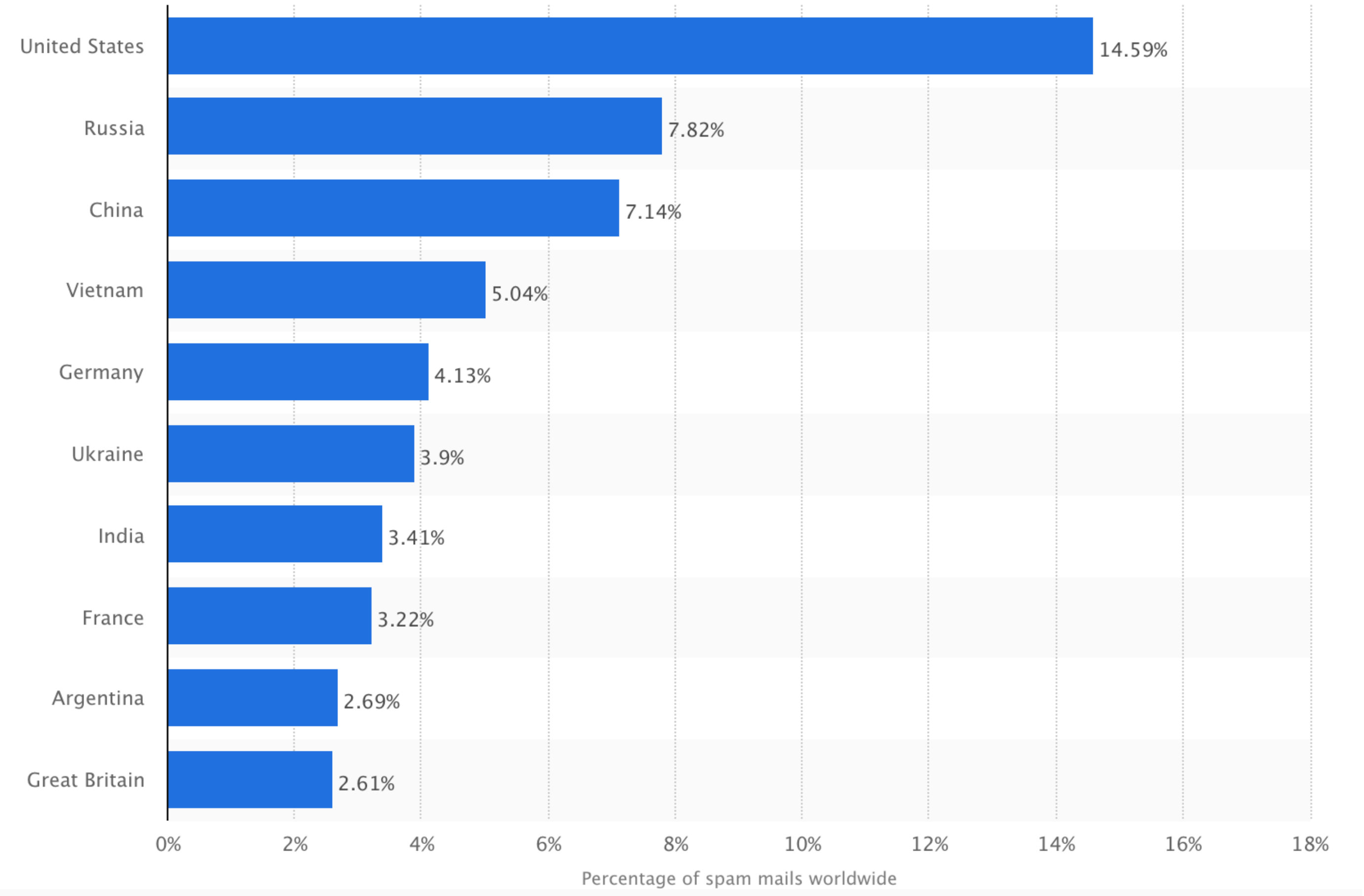


Source: http://www.statista.com/statistics/420400/spam-email-traffic-share-annual/

© Statista 2016

Email: Challenge

Leading countries of origin for unsolicited spam emails as of 2nd quarter 2015, by share of worldwide spam volume



Source: http://www.statista.com/statistics/263086/countries-of-origin-of-spam/

Discussion: How May Email Spams Be Detected?

Detection Methods Used by GMail

- Known phishing scams
- Message from unconfirmed sender identity
- Message you sent to Spam/similarity to suspicious messages
- Administrator-set policies
- Empty message content

https://support.google.com/mail/answer/1366858?hl=en