
Network Applications: Email, DNS

Qiao Xiang, Congming Gao, Qiang Su

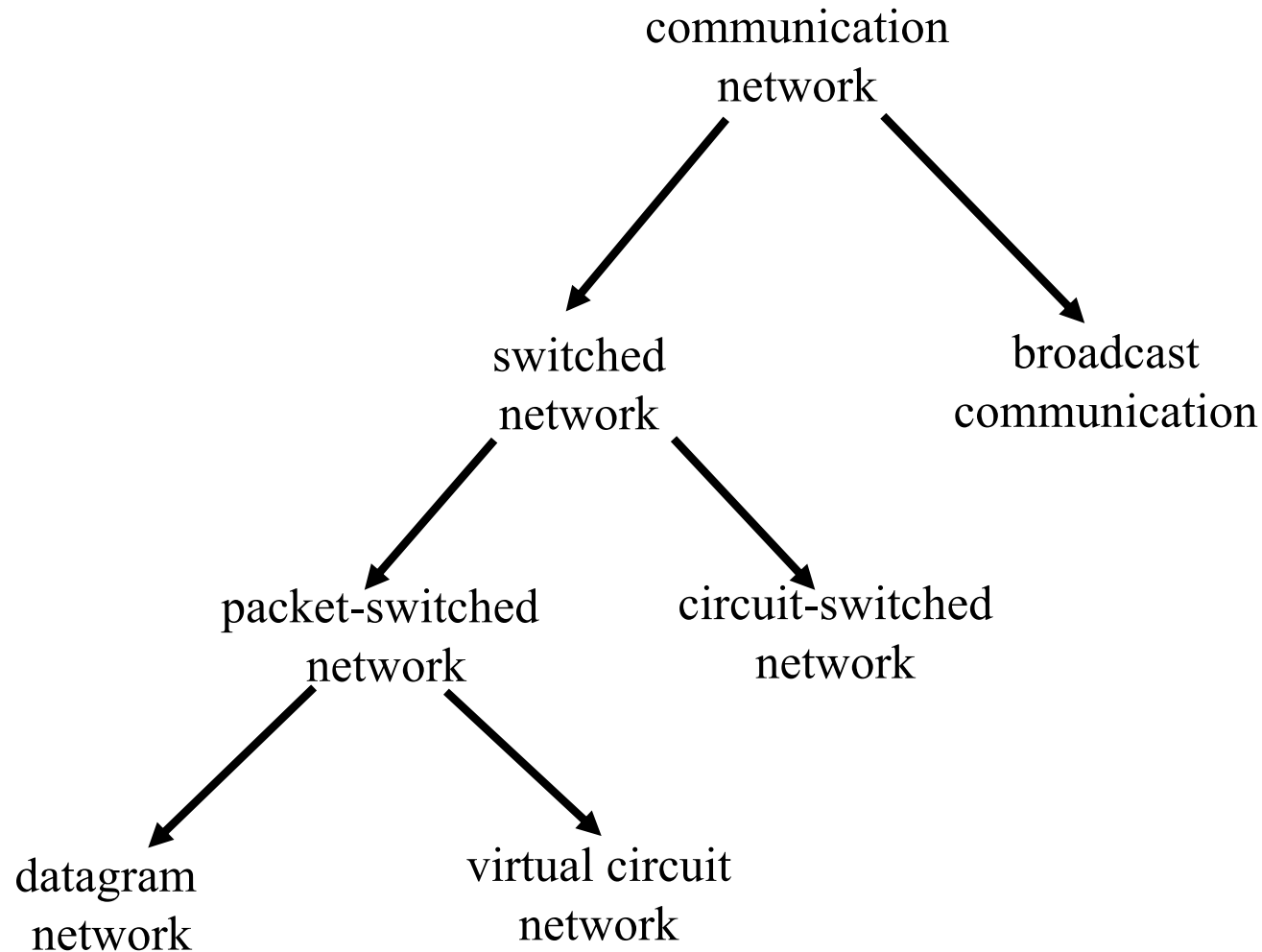
<https://sngroup.org.cn/courses/cnns-xmuf25/index.shtml>

09/11/2025

Outline

- Admin. and recap
- ❑ Layered architecture
 - Internet layering
- ❑ Application layer overview
- ❑ Network applications
 - Email
 - DNS

Recap: Summary of the Taxonomy of Communication Networks



Recap: Circuit Switching vs. Packet Switching

	circuit switching	packet switching
resource usage	use a single partition bandwidth	use whole link bandwidth
reservation/setup	need reservation (setup delay)	no reservation
resource contention	busy signal (session loss)	congestion (long delay and packet losses)
charging	time	packet
header	no per-pkt header	per packet header
fast path processing	fast	per packet processing

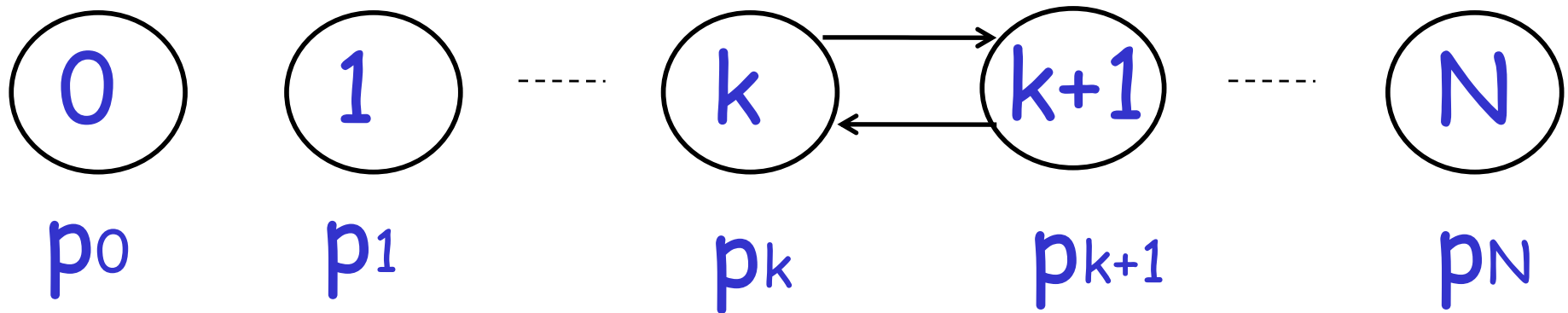
Recap: Queueing Theory

- ❑ Model **system state**
- ❑ Introduce **state transition** diagram
- ❑ Focus on **equilibrium**: state trend neither growing nor shrinking

Recap: Queueing Theory

Analysis of Circuit-Switching

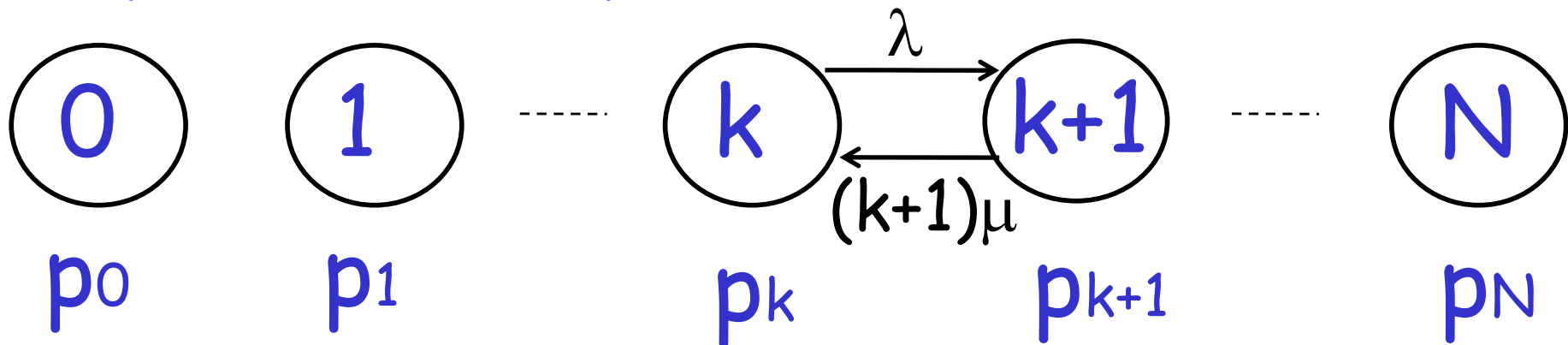
system state: # of busy lines



Recap: Queueing Theory

Analysis of Circuit-Switching

system state: # of busy lines



at equilibrium (time reversibility) in one unit time:

#(transitions $k \rightarrow k+1$) = #(transitions $k+1 \rightarrow k$)

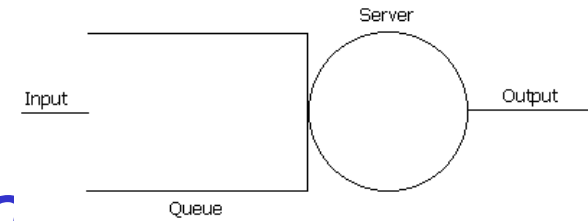
$$p_k \lambda = p_{k+1} (k+1) \mu$$

$$p_{k+1} = \frac{1}{k+1} \frac{\lambda}{\mu} p_k = \frac{1}{(k+1)!} \left(\frac{\lambda}{\mu} \right)^{k+1} p_0$$

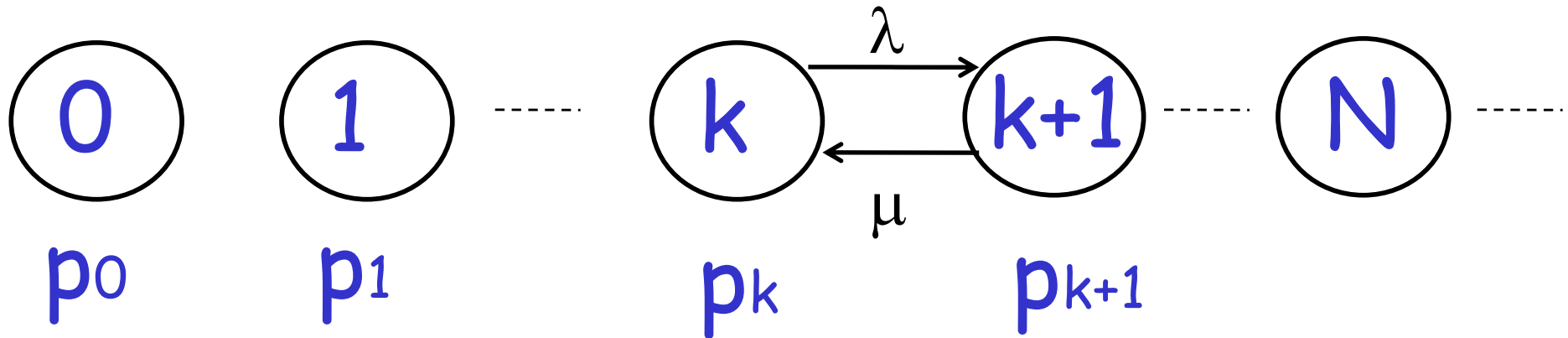
$$p_0 = \frac{1}{1 + \frac{1}{1!} \frac{\lambda}{\mu} + \frac{1}{2!} \left(\frac{\lambda}{\mu} \right)^2 + \dots + \frac{1}{N!} \left(\frac{\lambda}{\mu} \right)^N}$$

Recap: Queueing Theory

Analysis of Packet Switching



system state: #packets in queue



at equilibrium (time reversibility) in one unit time:

$\#(\text{transitions } k \rightarrow k+1) = \#(\text{transitions } k+1 \rightarrow k)$

$$p_k \lambda = p_{k+1} \mu$$

$$\sum_{k=0}^{\infty} p_k = 1$$

$$p_{k+1} = \frac{\lambda}{\mu} p_k = \left(\frac{\lambda}{\mu}\right)^{k+1} p_0 = \rho^{k+1} p_0$$

$$p_0 = 1 - \rho$$

$$\rho = \frac{\lambda}{\mu}$$

Recap: Layering

❑ Why layering

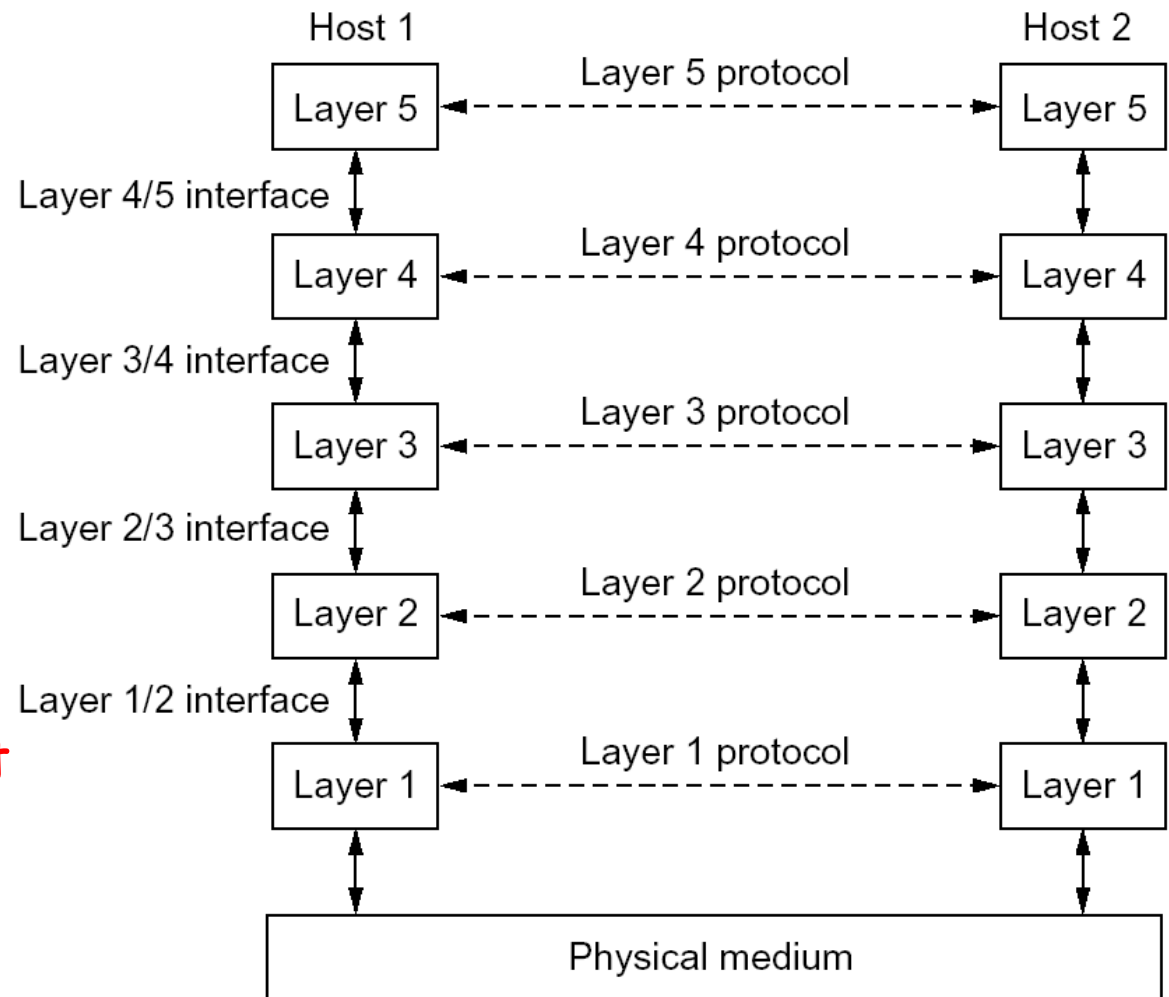
- reference model
- modularization

❑ Concepts

- service, interface, and protocol
- physical vs logical communication

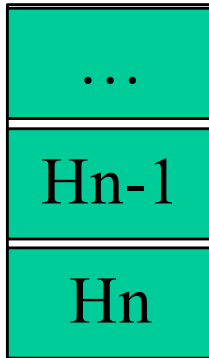
❑ Key design decision

- end-to-end argument to place functions in layers

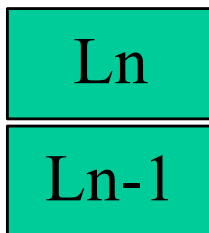


Some Implications of Layered Architecture

- A packet as a stack container

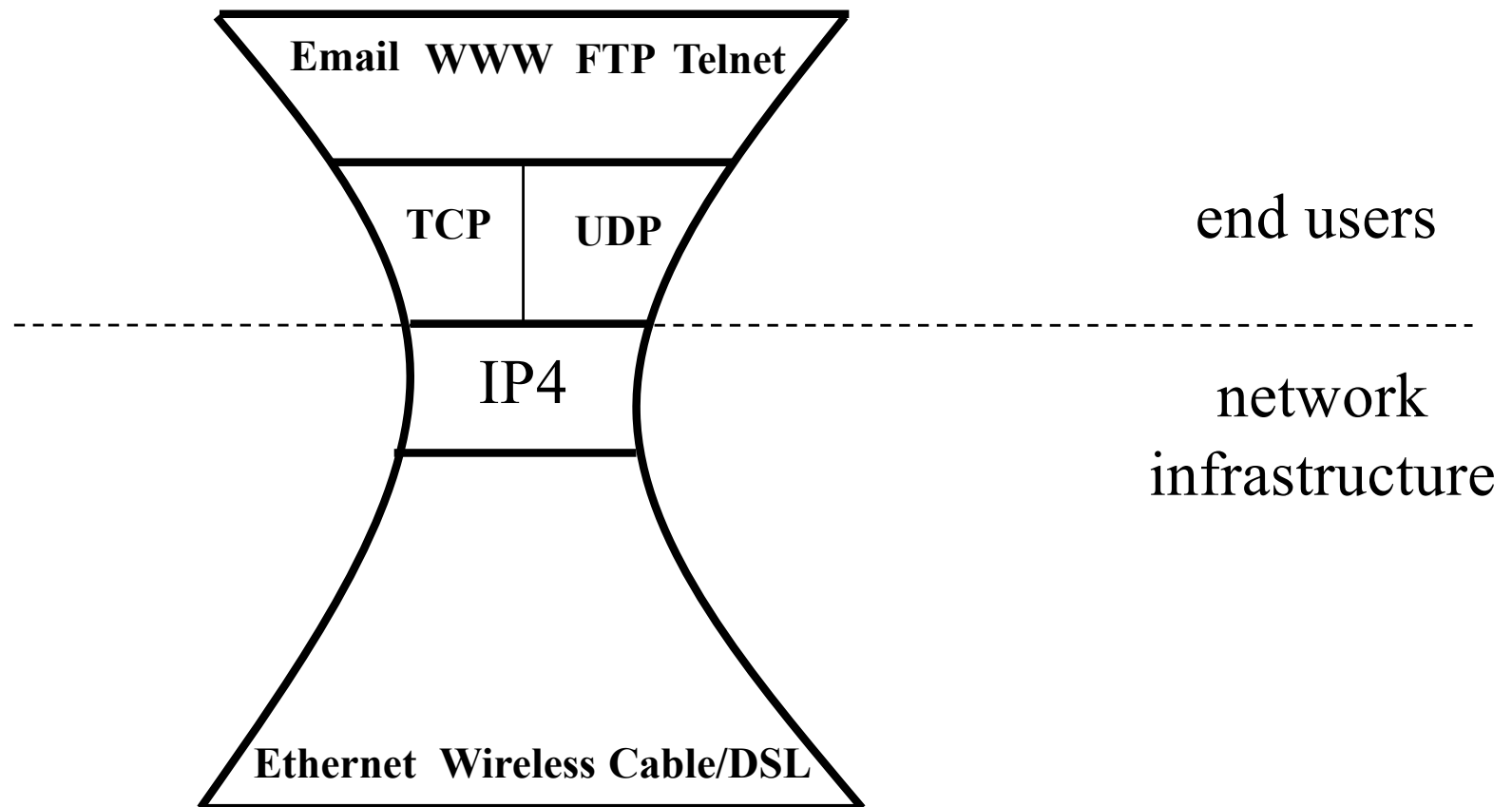


- Each layer needs multiplexing and demultiplexing to serve layer above



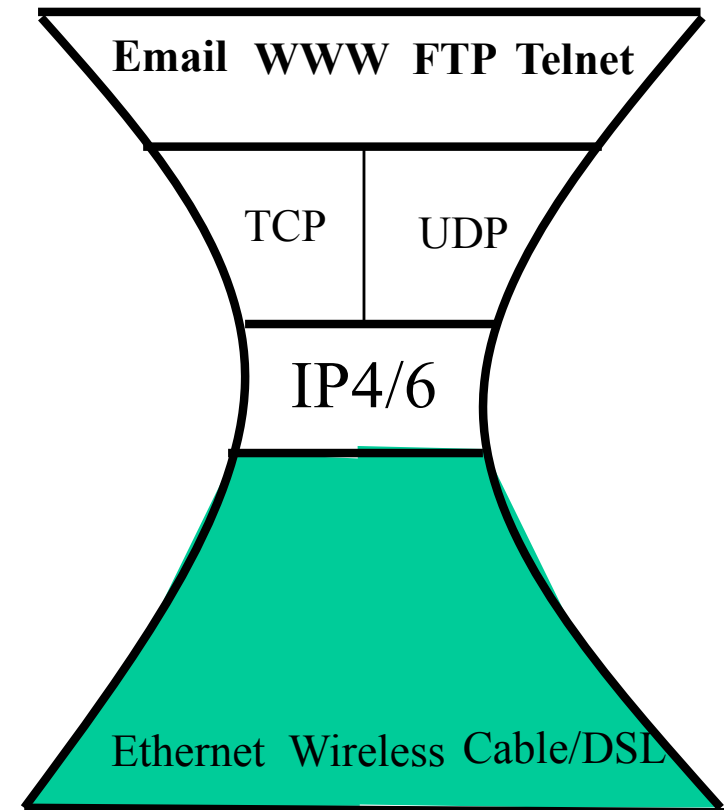
Has a field to indicate
which higher layer
requires the service

The Hourglass Architecture of the Internet

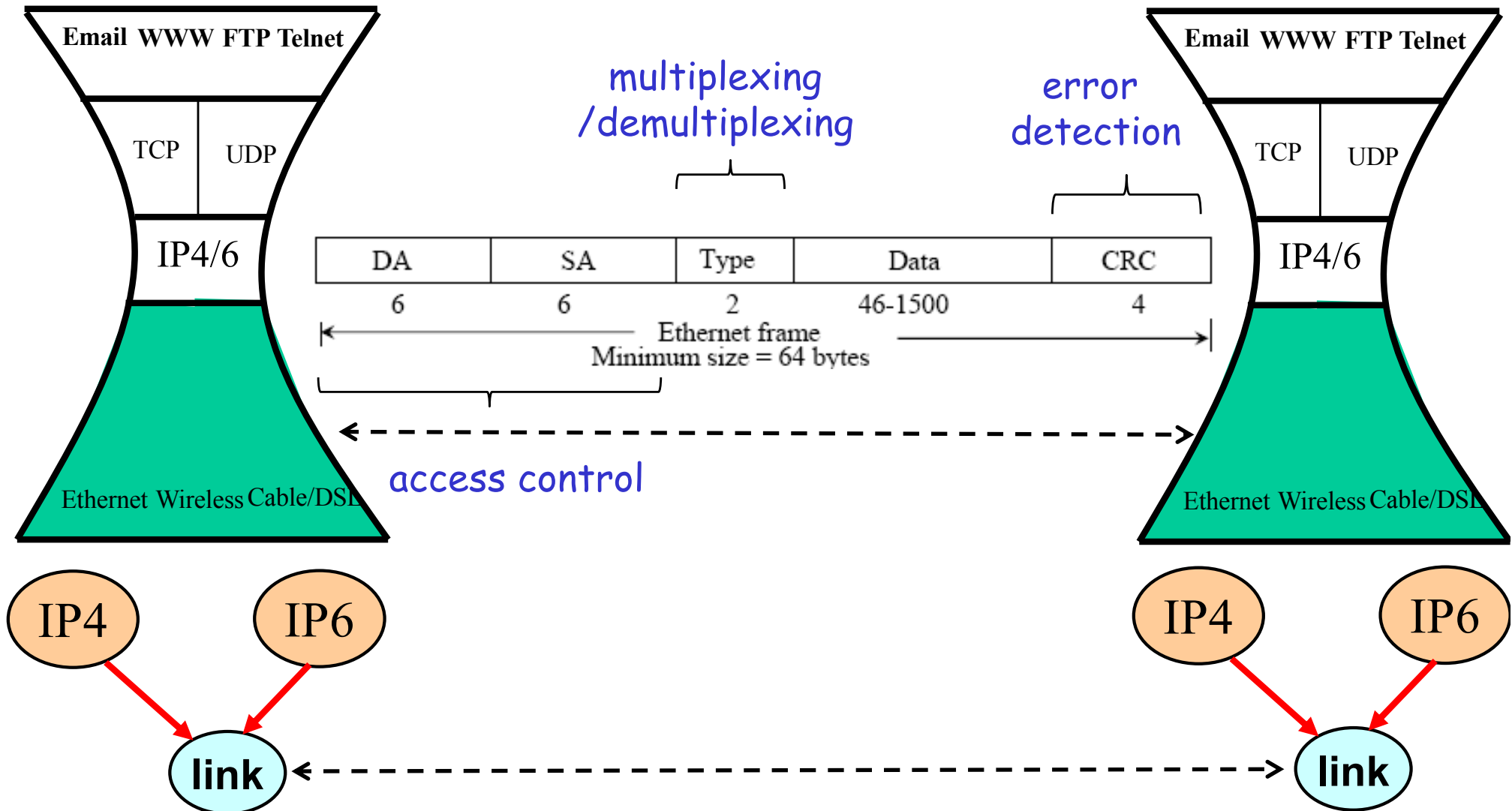


Link Layer (Ethernet)

- ❑ 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

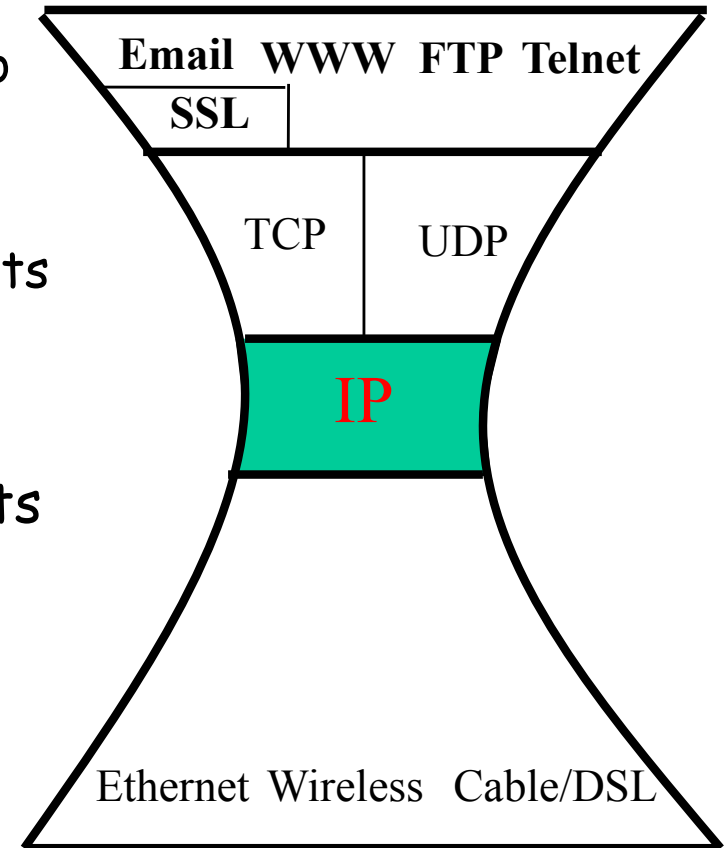


Link Layer: Protocol Header (Ethernet)

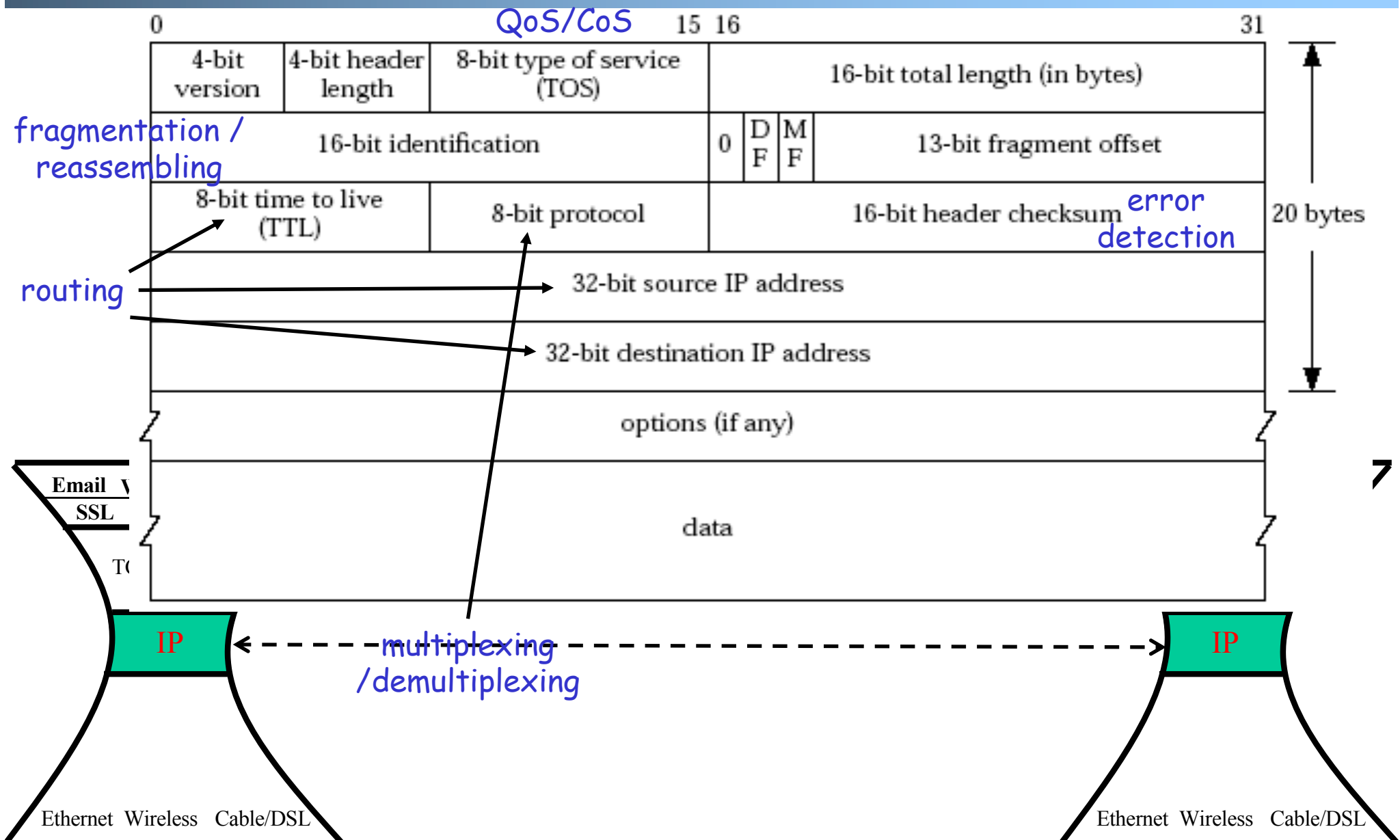


Network Layer: IP

- ❑ 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

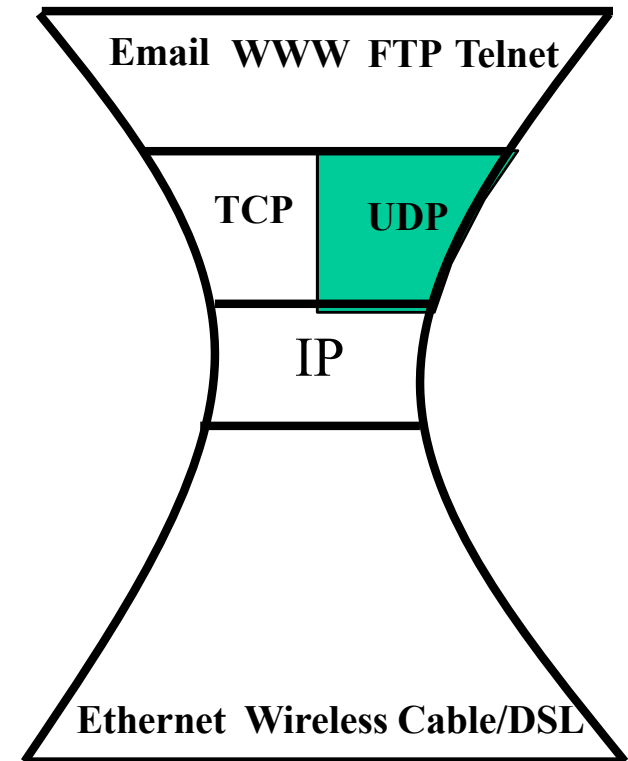


Network Layer: IPv4 Header



Transport Layer: UDP

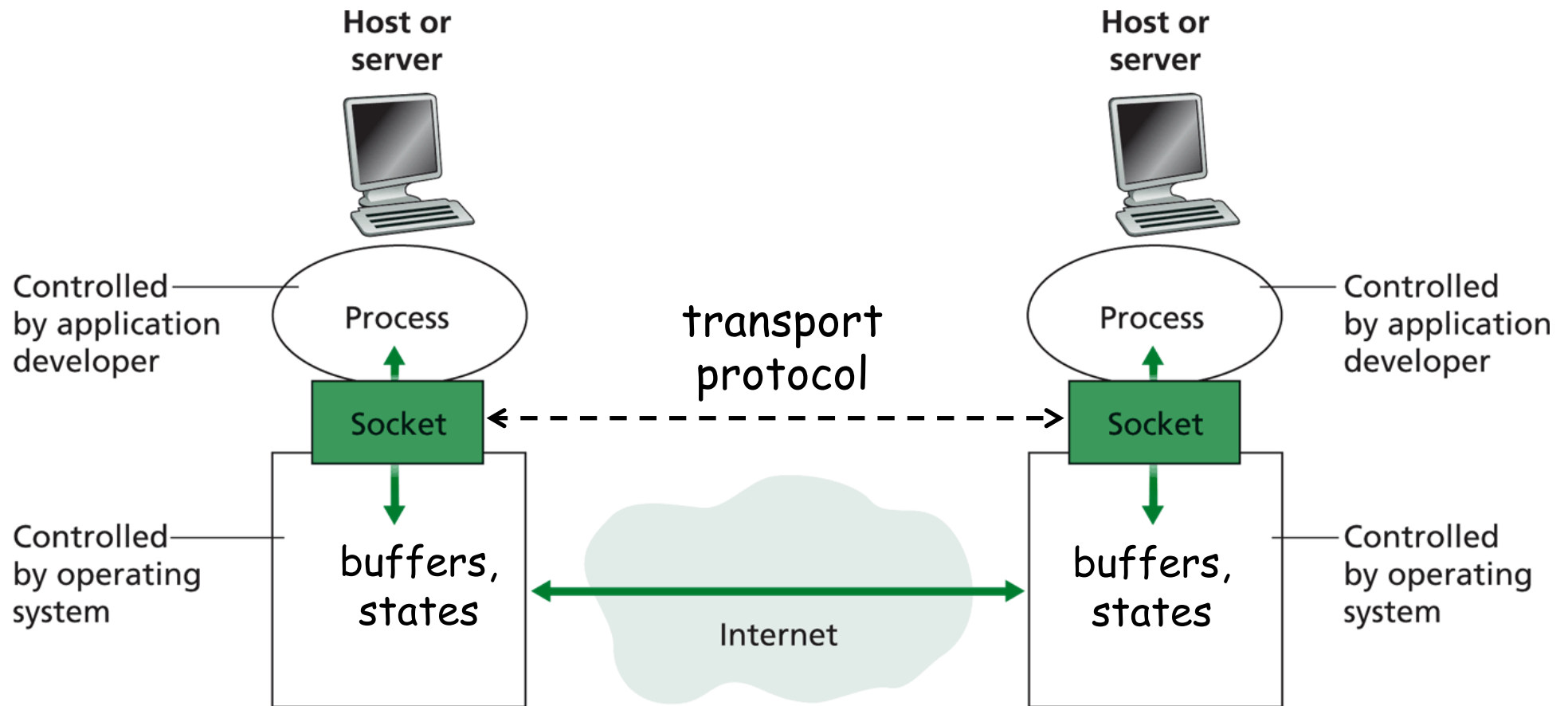
- ❑ A connectionless service
- ❑ Does not provide:
connection setup, reliability,
flow control, congestion
control, timing, or
bandwidth guarantee
 - why is there a UDP?



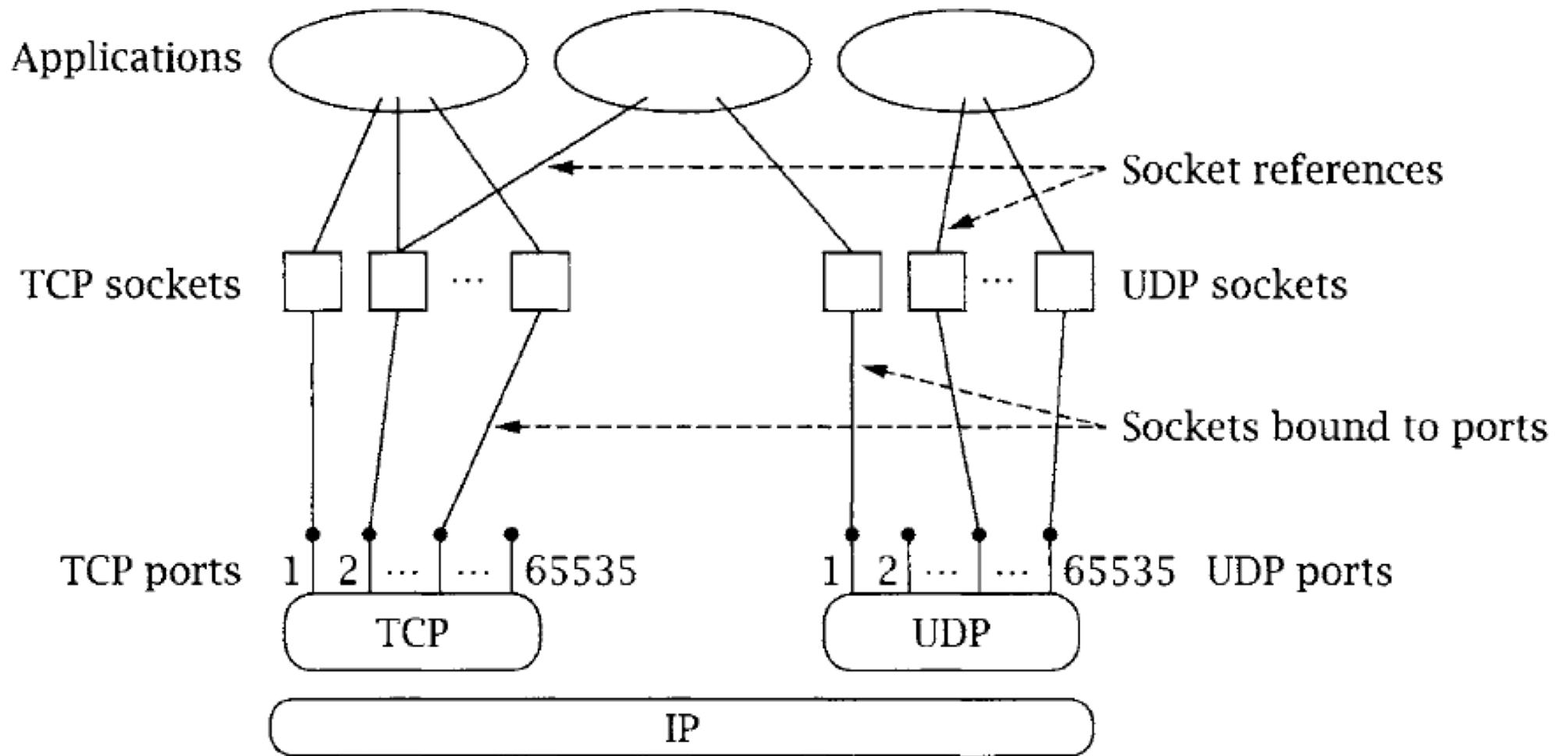
Transport Services and APIs

- ❑ 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

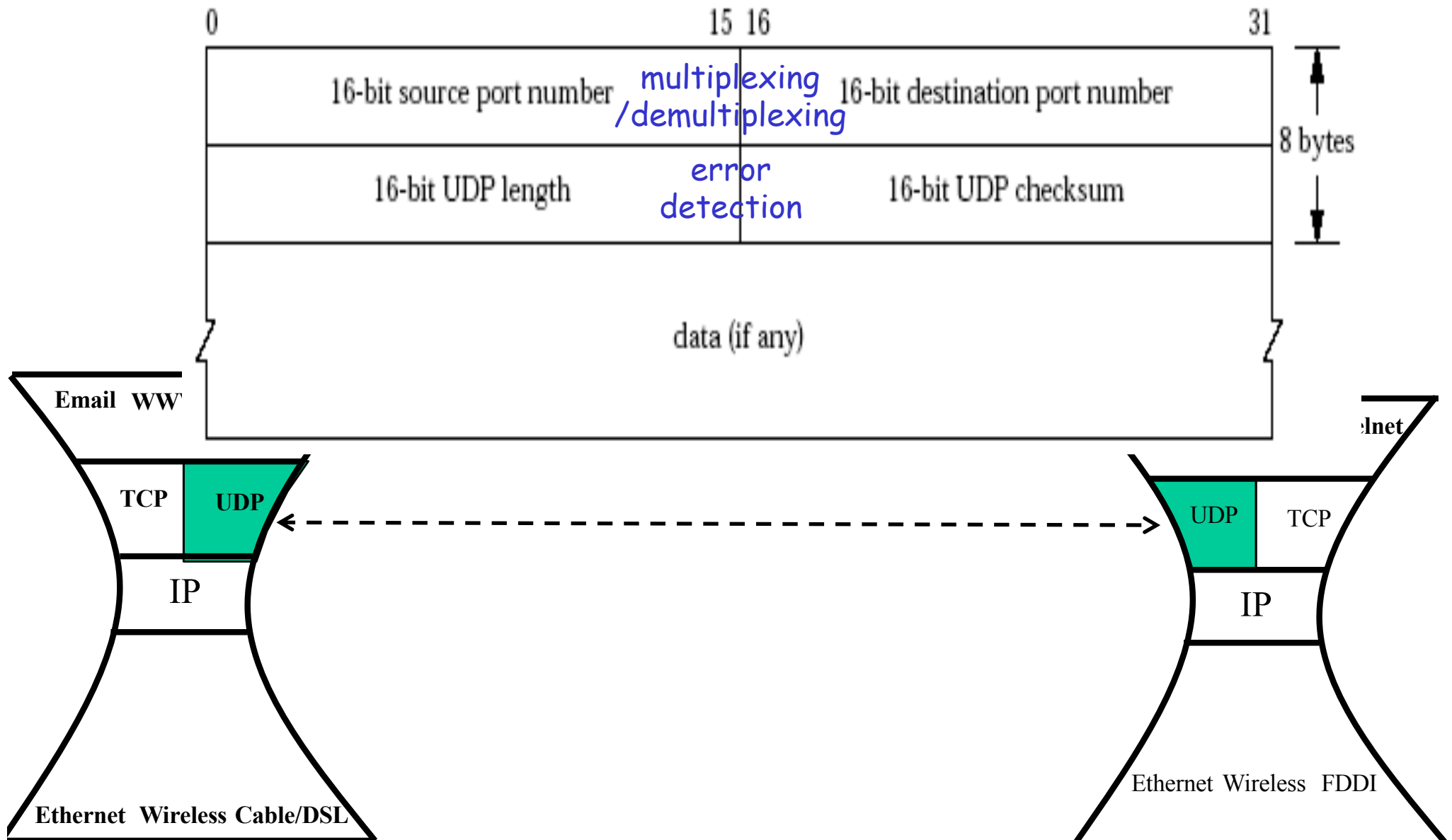
Socket Service Model and API



Multiplexing/Demultiplexing



Transport Layer: UDP Header



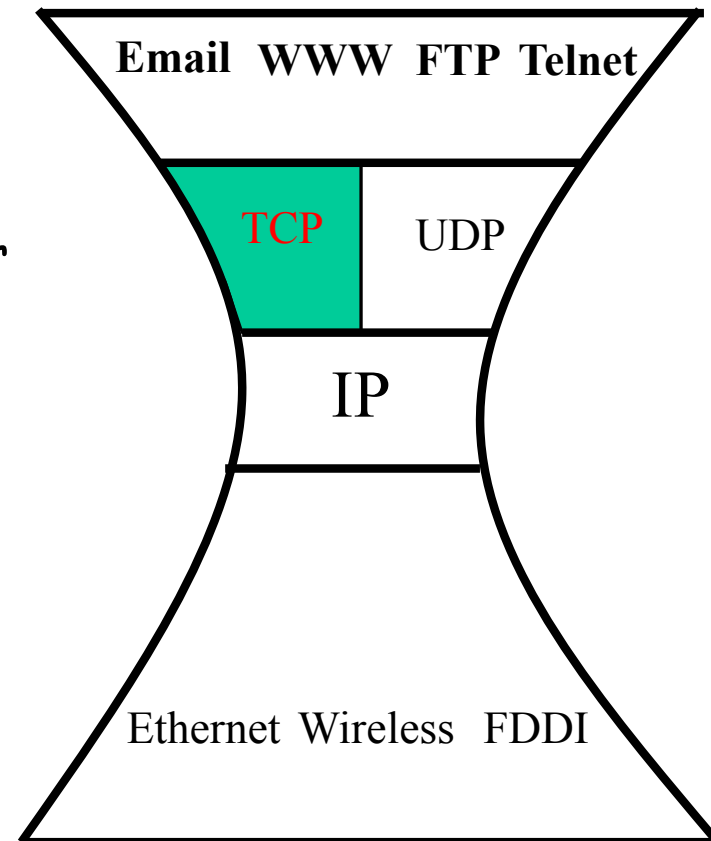
Transport Layer: TCP

□ Services

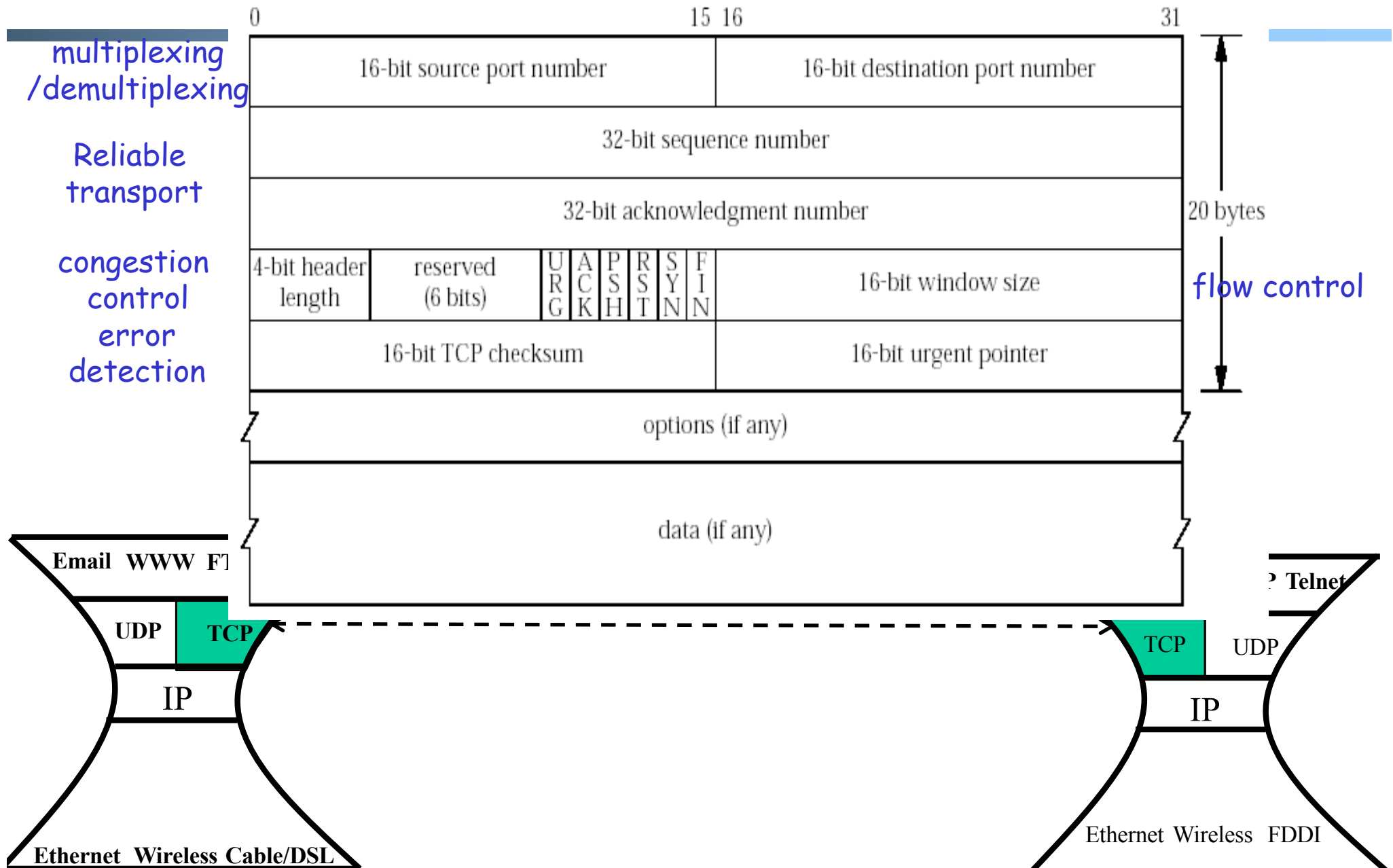
- multiplexing/demultiplexing
- reliable transport
 - between sending and receiving processes
 - setup required between sender and receiver: a **connection-oriented service**
- flow control: sender won't overwhelm receiver
- congestion control: throttle sender when network overloaded
- error detection
- does not provide timing, minimum bandwidth guarantees

□ Interface:

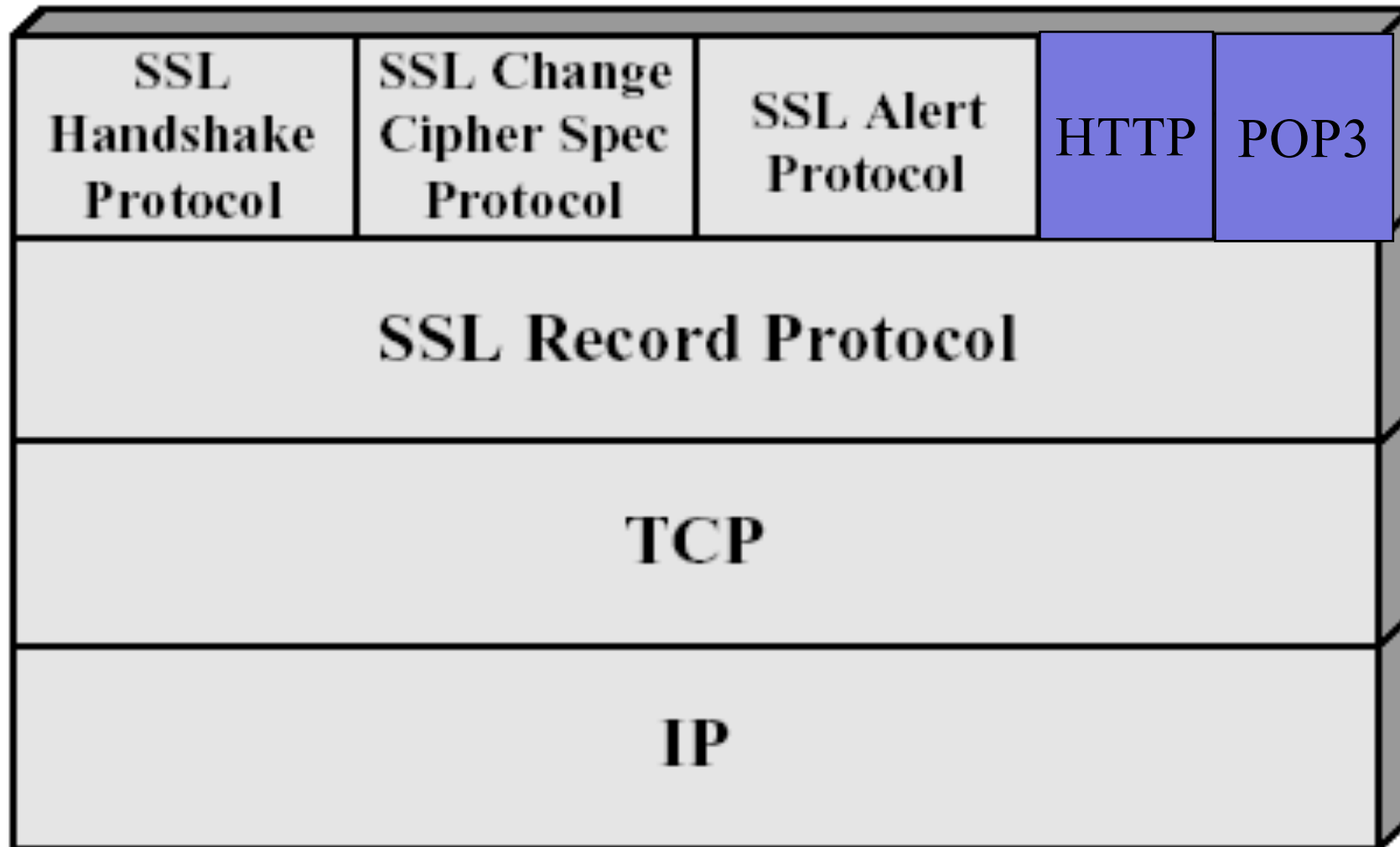
- send a packet to a (app-layer) peer



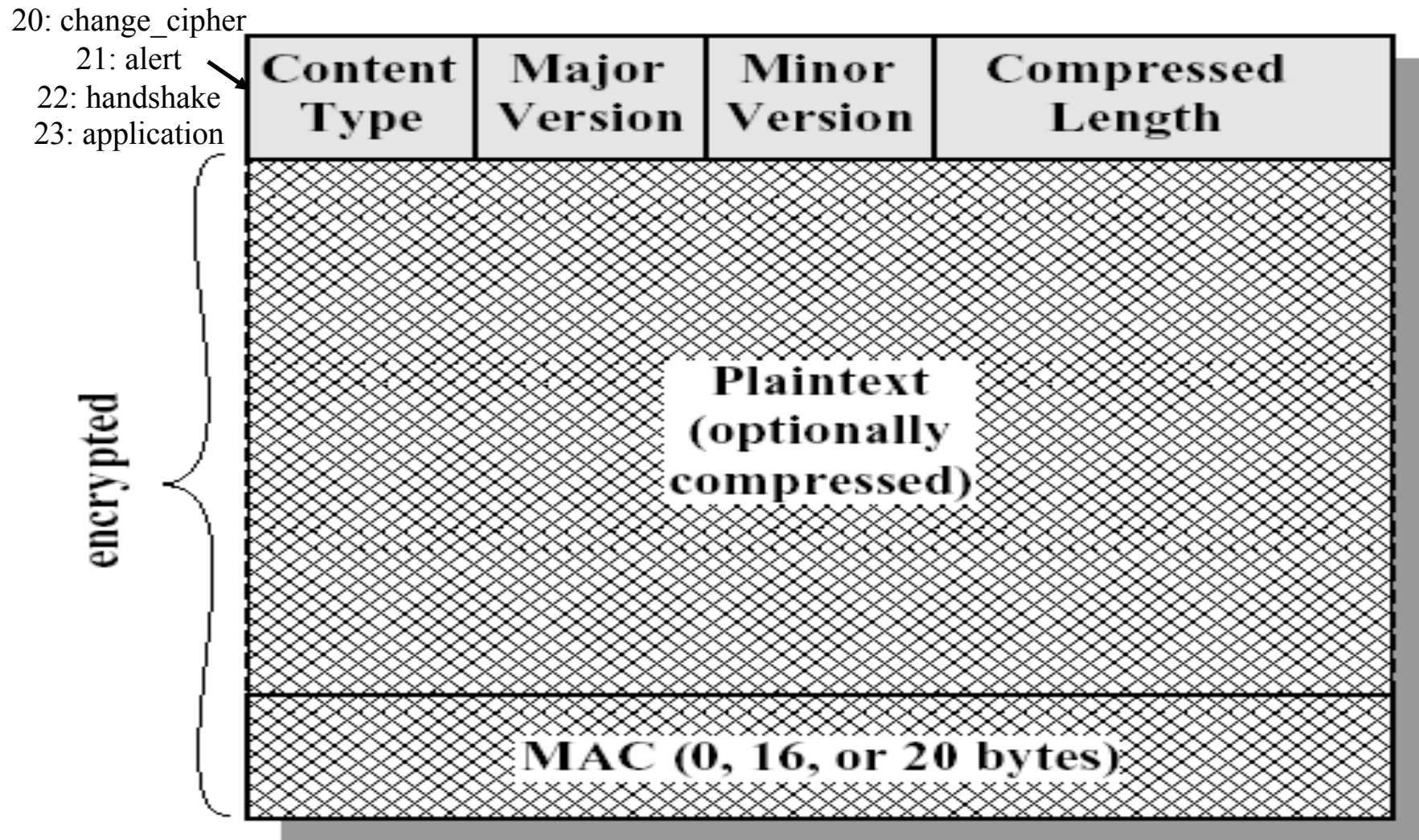
Transport Layer: TCP Header



Secure Socket Layer Architecture



SSL Record-Layer Packet Format



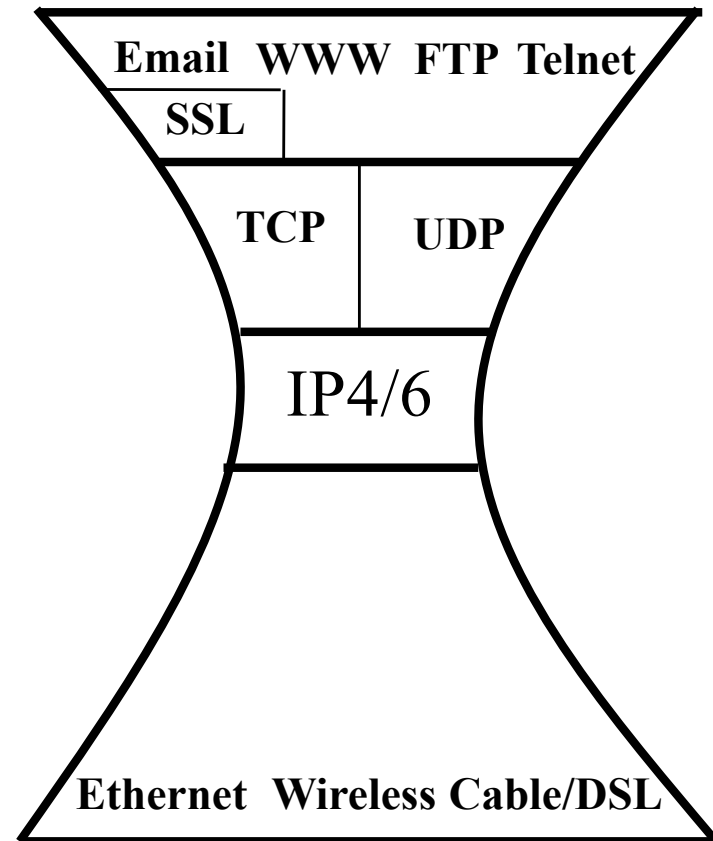
Summary: The Big Picture of the Internet

□ Hosts and routers:

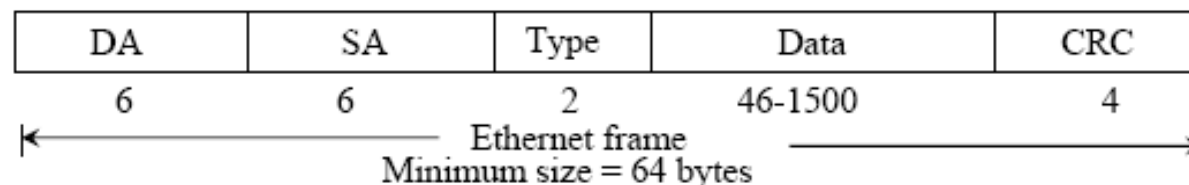
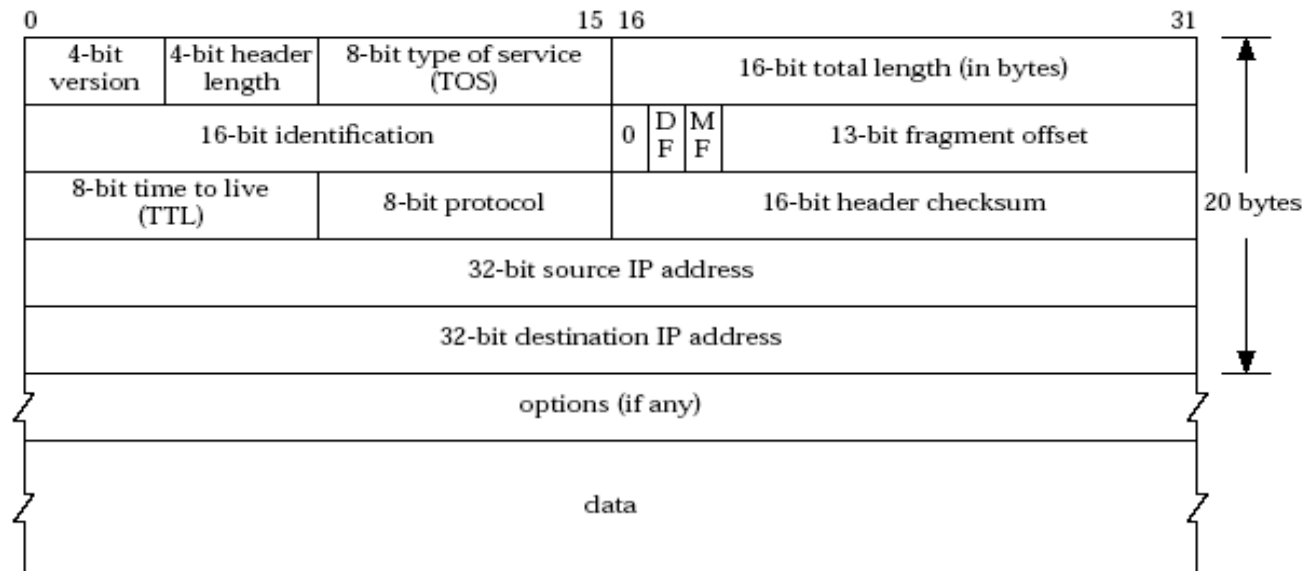
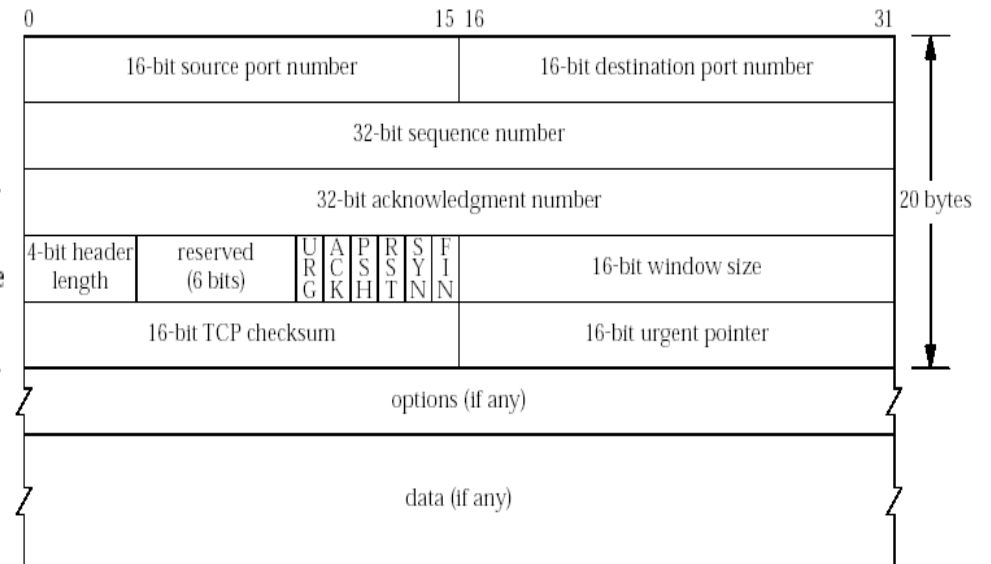
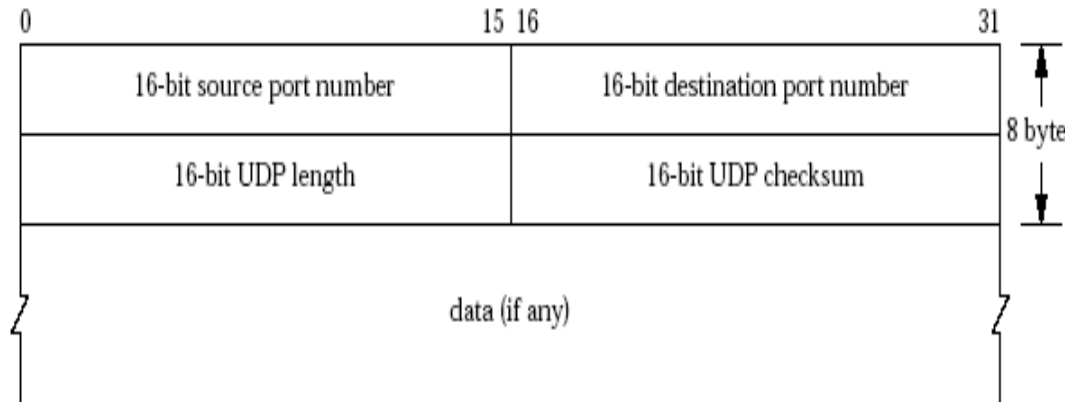
- ~ 1 bil. hosts
- autonomous systems organized roughly hierarchical
- backbone links at 100 Gbps

□ 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 paradigm
 - network app. programming

- ❑ Learn about applications by examining common applications
 - smtp/pop
 - dns
 - http (1, 1.1, /2)
 - content distribution
 - peer-to-peer

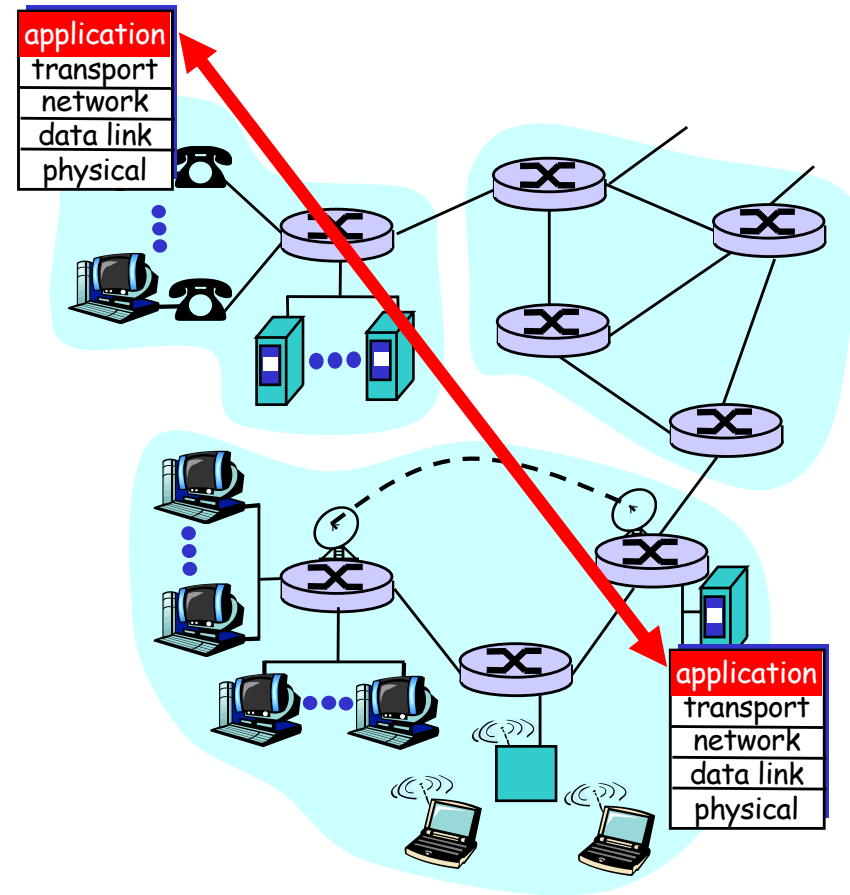
Network Applications vs. Application-layer Protocols

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. and Trans.: App. Protocols and their Transport Protocols

- An application needs to choose the transport protocol

<u>Application</u>	<u>Application layer protocol</u>	<u>Underlying transport protocol</u>
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

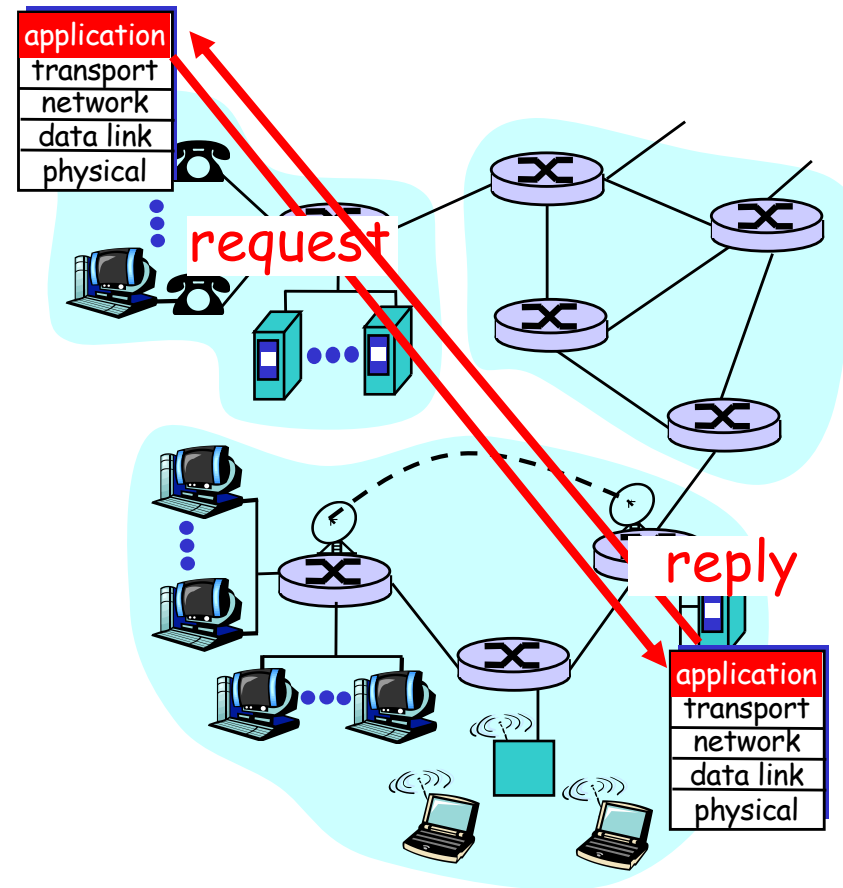
Typical network app has two pieces: *client* and *server*

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):

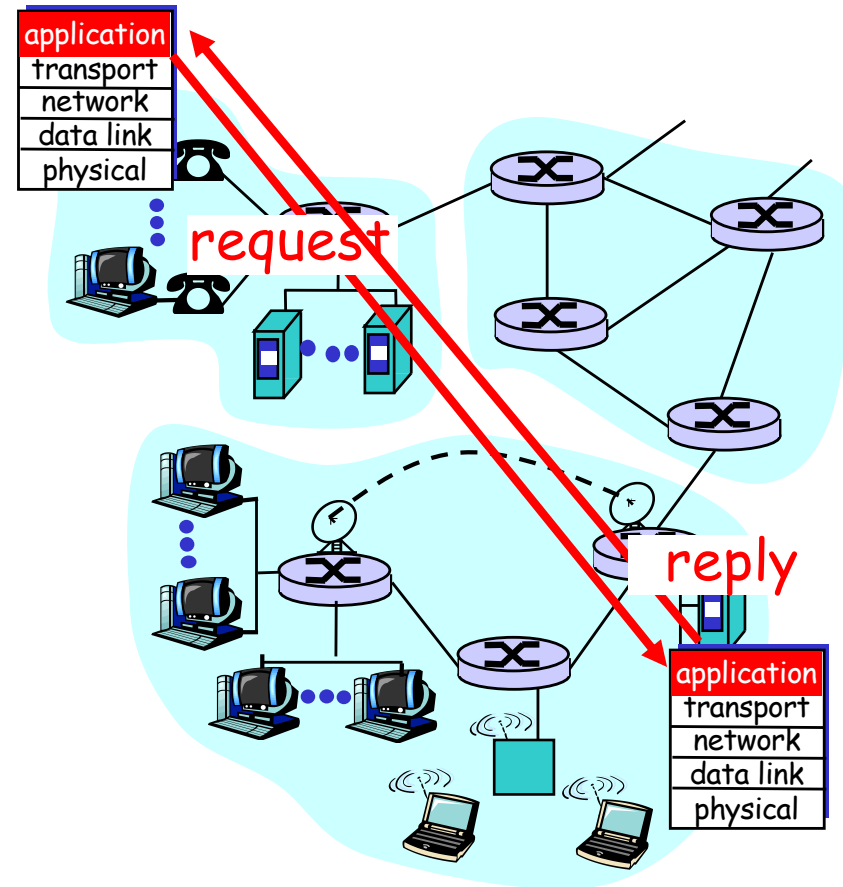
- provides requested service to client
- e.g., Web server sends requested Web page; mail server delivers e-mail



Client-Server Paradigm: Key Questions

Key questions to ask about a C-S application

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



Outline

- ❑ Admin. and recap
- ❑ Application layer overview
- ❑ Network applications
 - *Email*

Electronic Mail

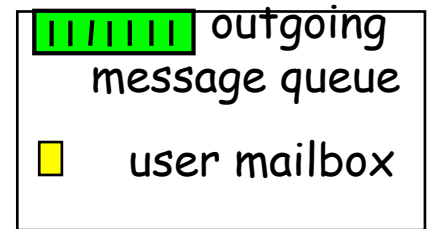
- ❑ Still active
 - 80B emails/day
 - 3.9B active email boxes

- ❑ A highly recommended reading: a history of Email development
 - linked on the Schedule page

Demo: SMTP

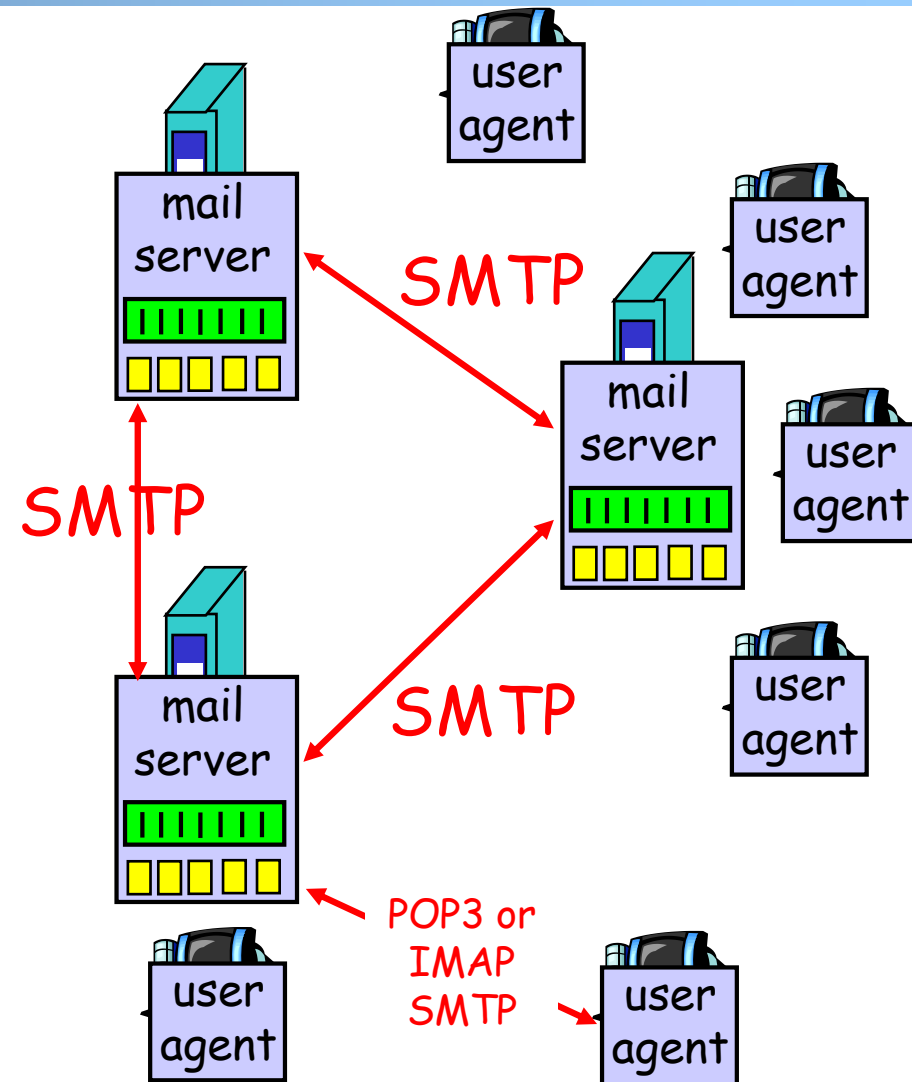
C: [auth login](#)
S: 334 VXNlcm5hbWU6
C: eG11Y25ucw==
S: 334 UGFzc3dvcmQ6
C: MzM0ZjU2MDVkZjE1MDRmOQ==
S: 235 OK Authenticated
C: [mail from](#):xmucnns@sina.com
S: 250 ok
C: [rcpt to](#):qiaoxiang@xmu.edu.cn
S: 250 ok
C: [data](#)
S: 354 End data with <CR><LF>.<CR><LF>
C: Date:2022-9-22 12:36
C: From:xmucnns@sina.com
C: To:qiaoxiang@xmu.edu.cn
C: Subject:test smtp
C:
C: Hello, Qiao.
C:
C: .
S: 250 ok queue id 11479549283321
C: [quit](#)
S: 221 smtp-97-27.smtpmail.fmail.bx.sinanode.com
S: Connection closed by foreign host.

Electronic Mail: Components

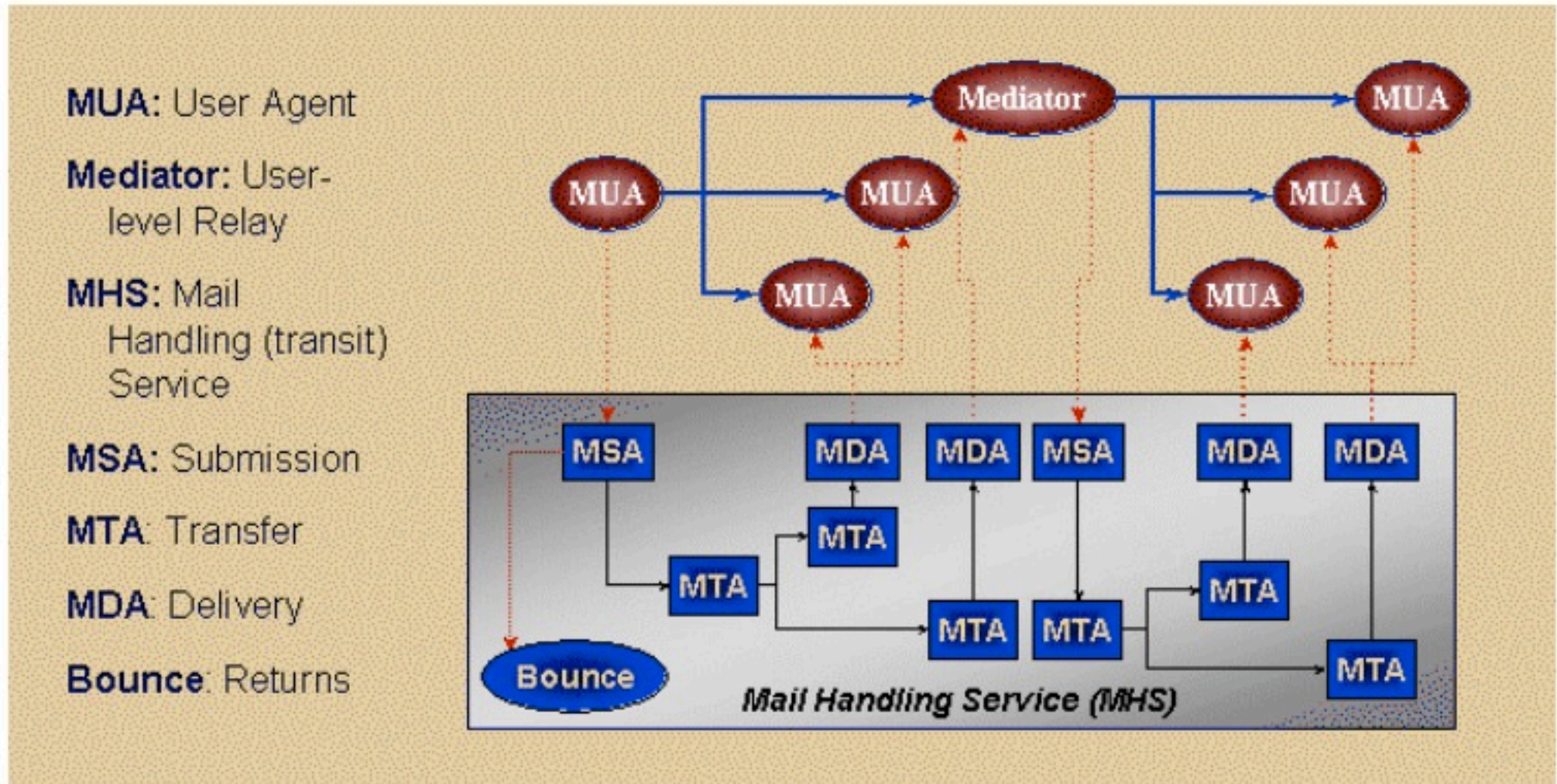


Three major components:

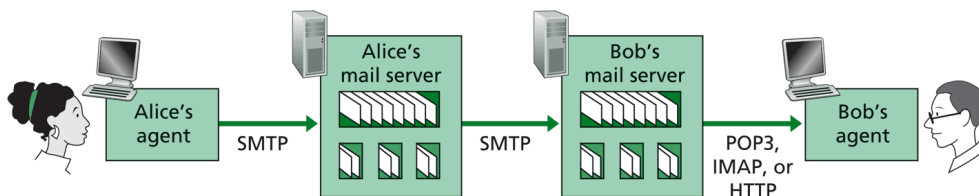
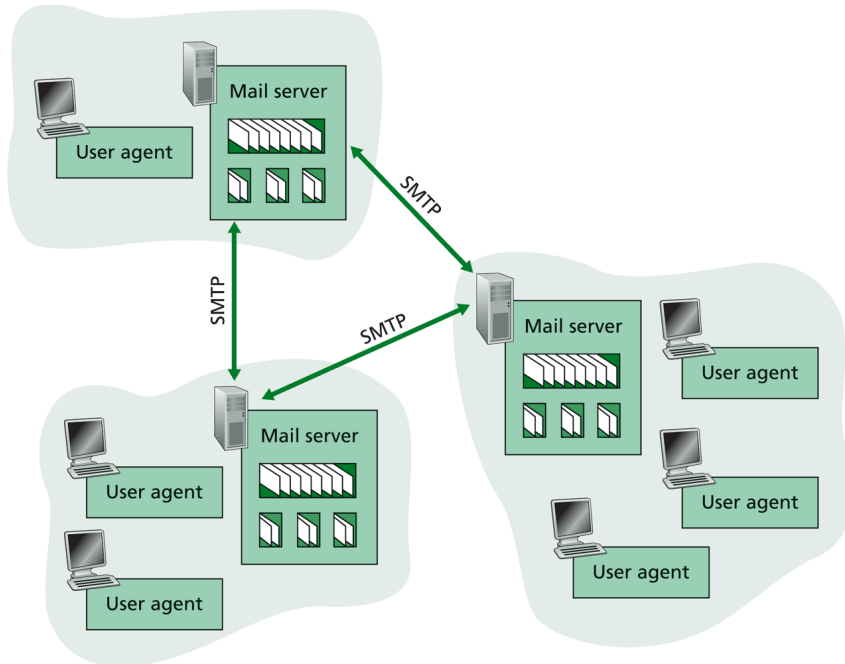
- User agents
- Mail servers
- Protocols
 - Mail transport protocol
 - SMTP
 - Mail access protocols
 - POP3: Post Office Protocol [RFC 1939]
 - IMAP: Internet Mail Access Protocol [RFC 1730]



Email Transport Architecture



SMTP: Mail Transport Protocol Messages (Envelop Messages)



```

C: auth login
S: 334 VXNlcm5hbWU6
C: eG11Y25ucw==
S: 334 UGFzc3dvcmQ6
C: MzM0ZjU2MDVkJzE1MDRmOQ==
S: 235 OK Authenticated
C: mail from:xmucnns@sina.com
S: 250 ok
C: rcpt to:qiaoxiang@xmu.edu.cn
S: 250 ok
C: data
S: 354 End data with <CR><LF>.<CR><LF>
C: Date:2021-9-22 12:36
C: From:xmucnns@sina.com
C: To:qiaoxiang@xmu.edu.cn
C: Subject:test smtp
C:
C: Hello, Qiao.
C:
C:
S: 250 ok qu
C: quit
S: 221 smtp-
S: Connection
  
```

Email text different from
SMTP protocol message

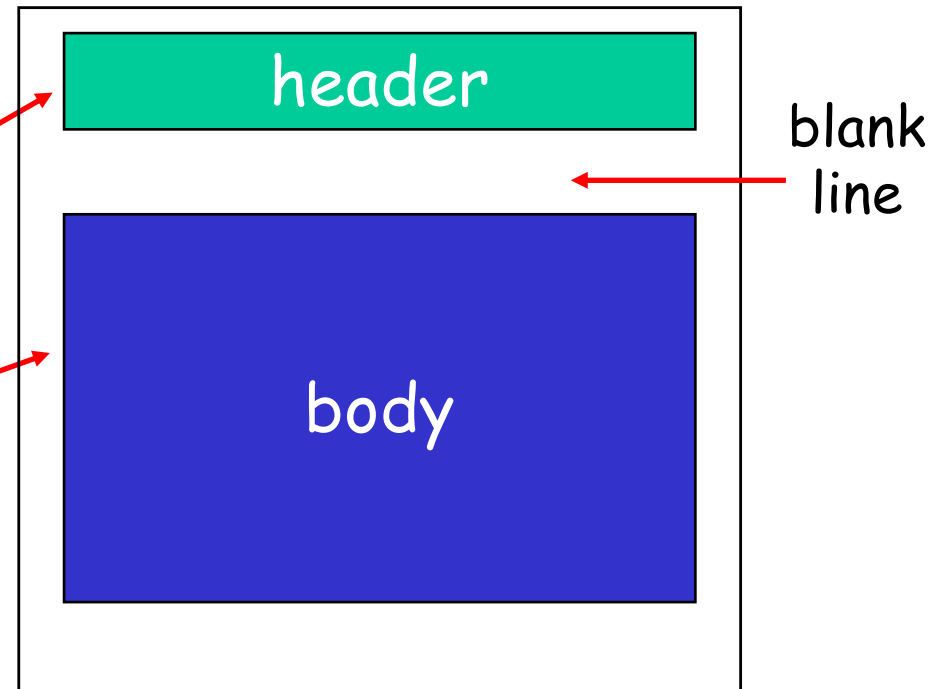
%telnet smtp.sina.com 25

Mail Message Data

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



Benefit of separating protocol and msg: easier extensibility

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: xmucnns@sina.com
To: qiaoxiang@xmu.edu.cn
Subject: Network map.
MIME-Version: 1.0
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

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

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

Multipart Type: How Attachment Works

From: xmucnns@sina.com
To: qiaoxiang@xmu.edu.cn
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--

POP3 Protocol: Mail Retrieval

Authorization phase

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

S: +OK sina pop3 server ready
C: user xmucnns
S: +OK welcome to sina mail
C: pass 334f5605df1504f9
S: +OK 4 messages (32377 octets)

user successfully logged in

Transaction phase, client:

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

C: list
S: +OK 4 messages (32377 octets)
S: 1 10410
S: 2 10748
S: 3 7859
S: 4 3360
S: .
C: retr 4
S: +OK 3360 octets
C: dele 2
C: quit
S: +OK

POP3 server signing off

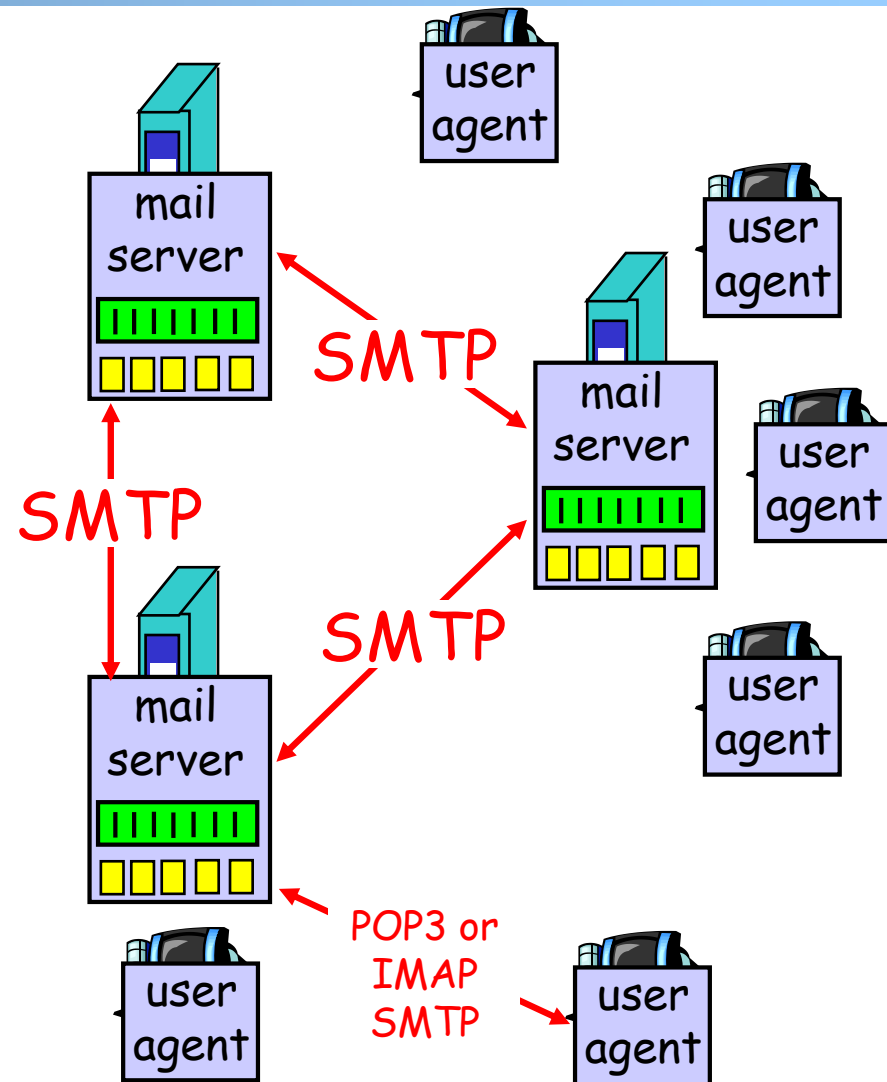
Exercise

- ☐ Register an email address at sina.com
- ☐ Send an email to the registered email address using smtp
- ☐ Retrieve using pop

Evaluation of SMTP/POP/IMAP

Key questions to ask about a C-S application

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

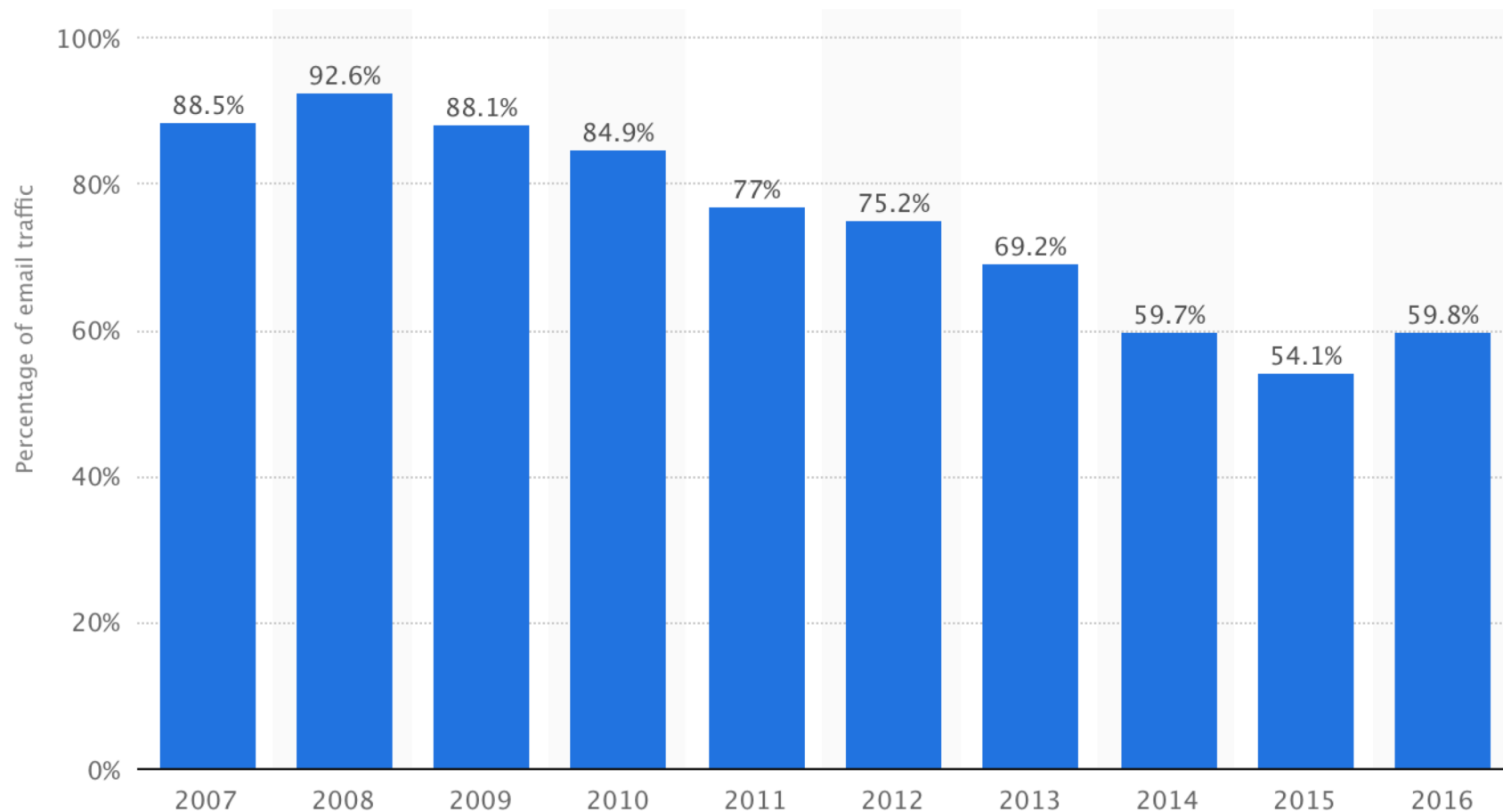


Email Security: Spam

□ Spam (Google)



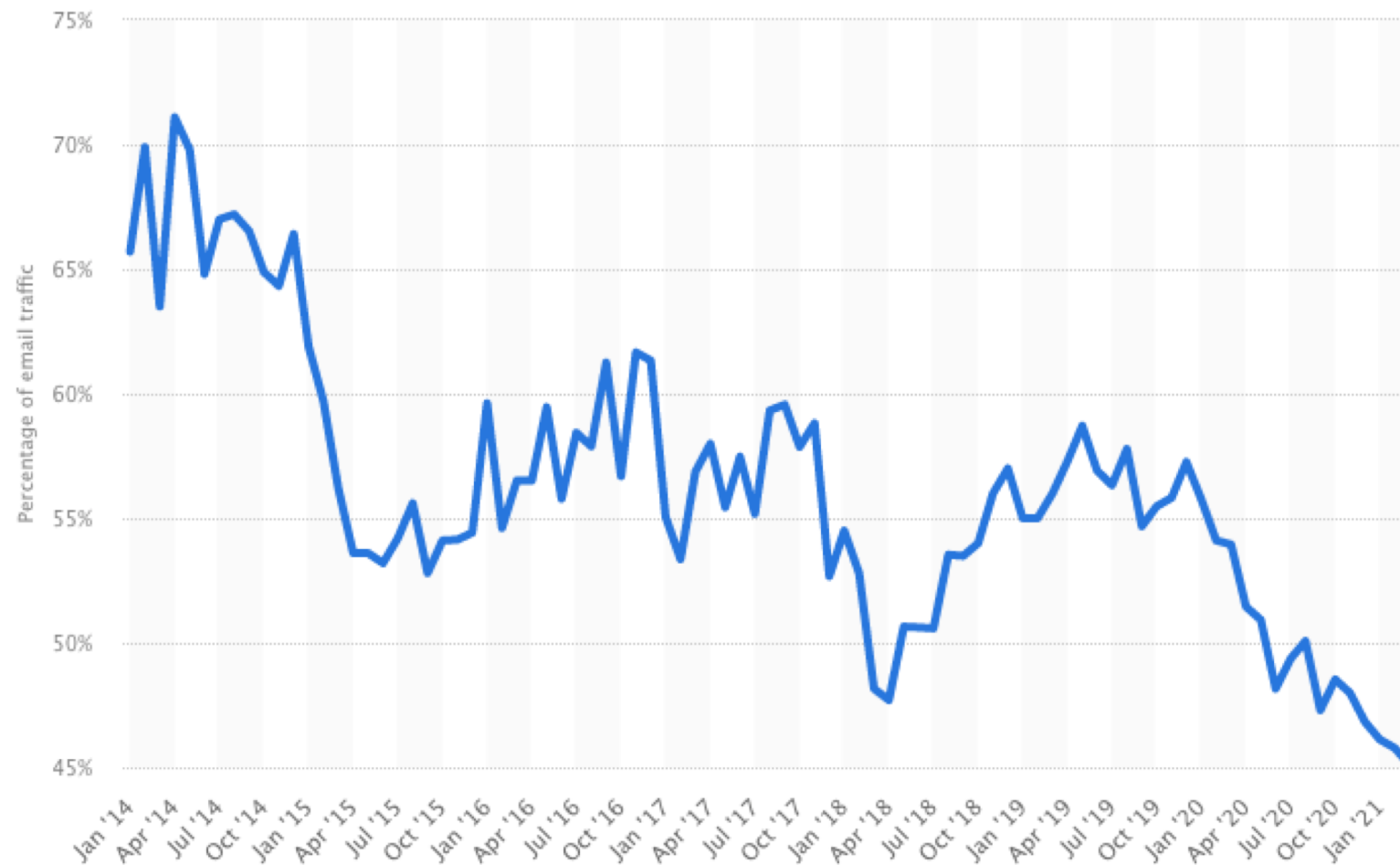
Email Security Issue: Spam



© Statista 2017

Source: <https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/>

Email Security Issue: Spam



Source: <https://www.statista.com/statistics/420391/spam-email-traffic-share/>

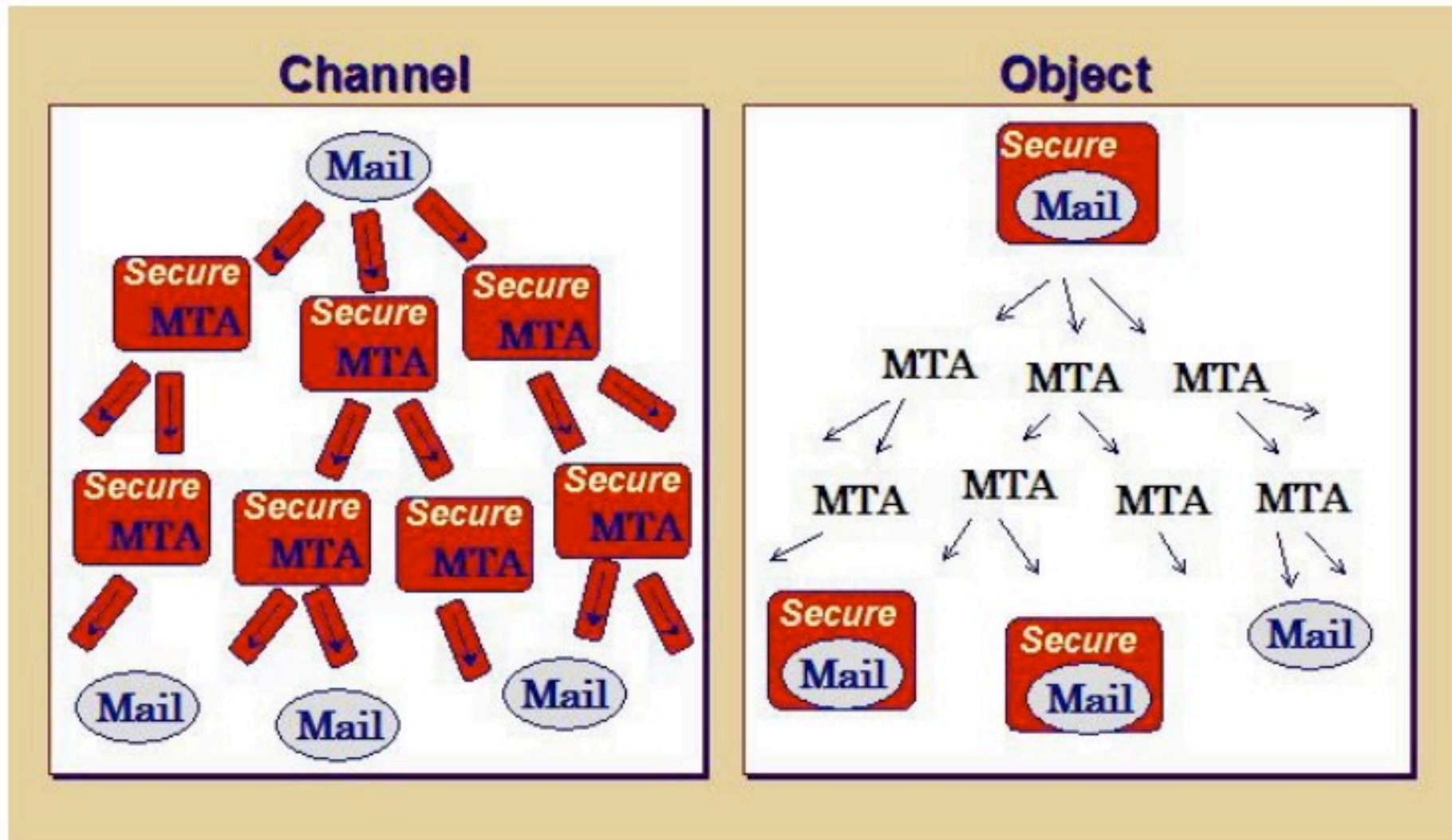
Discussion: How May One Handle Email Spams?

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

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

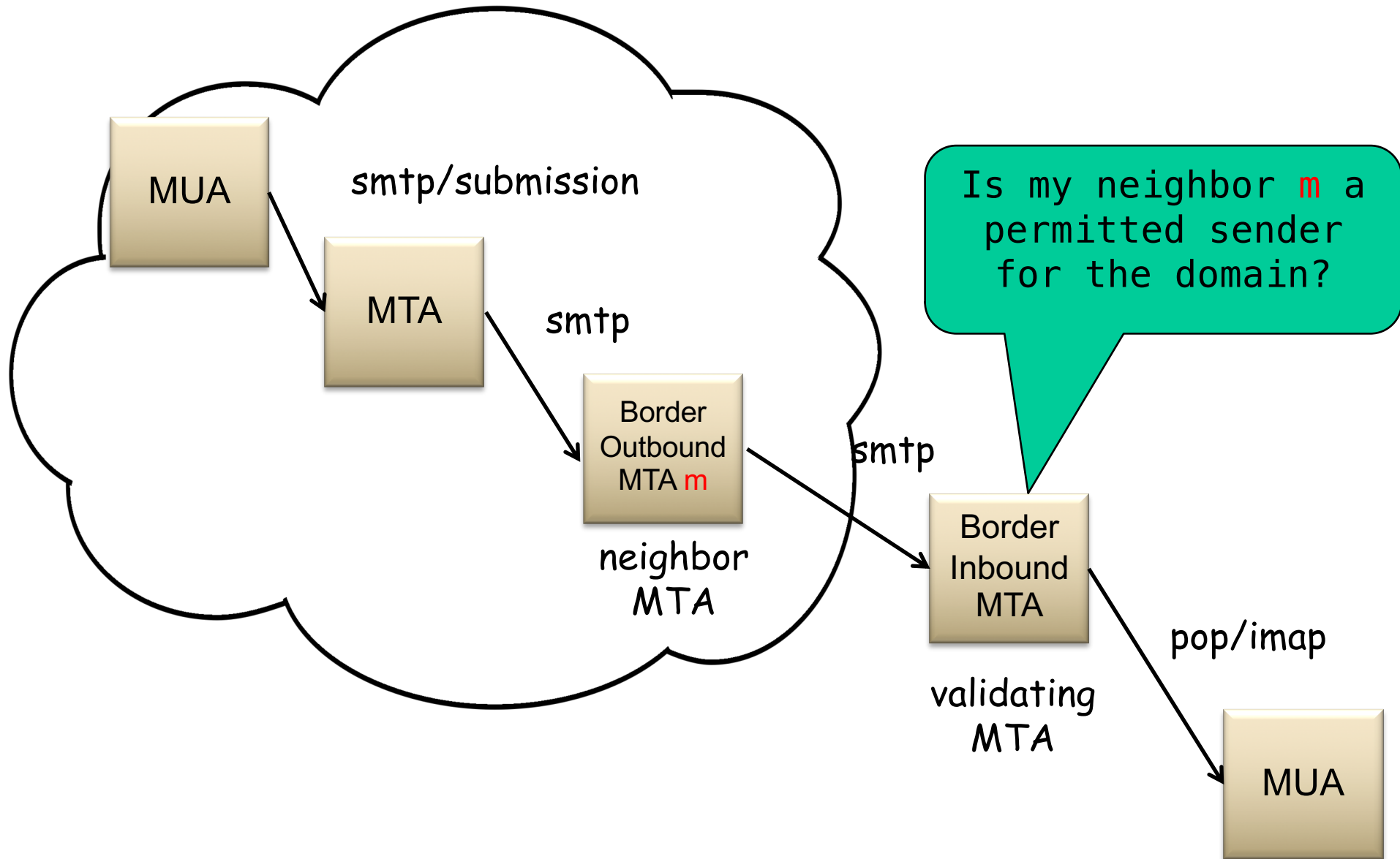
Email Authentication Approaches



Sender Policy Frame (SPF)

DomainKeys Identified Mail (DKIM)
Authenticated Results Chain (ARC)

Sender Policy Framework (SPF RFC7208)



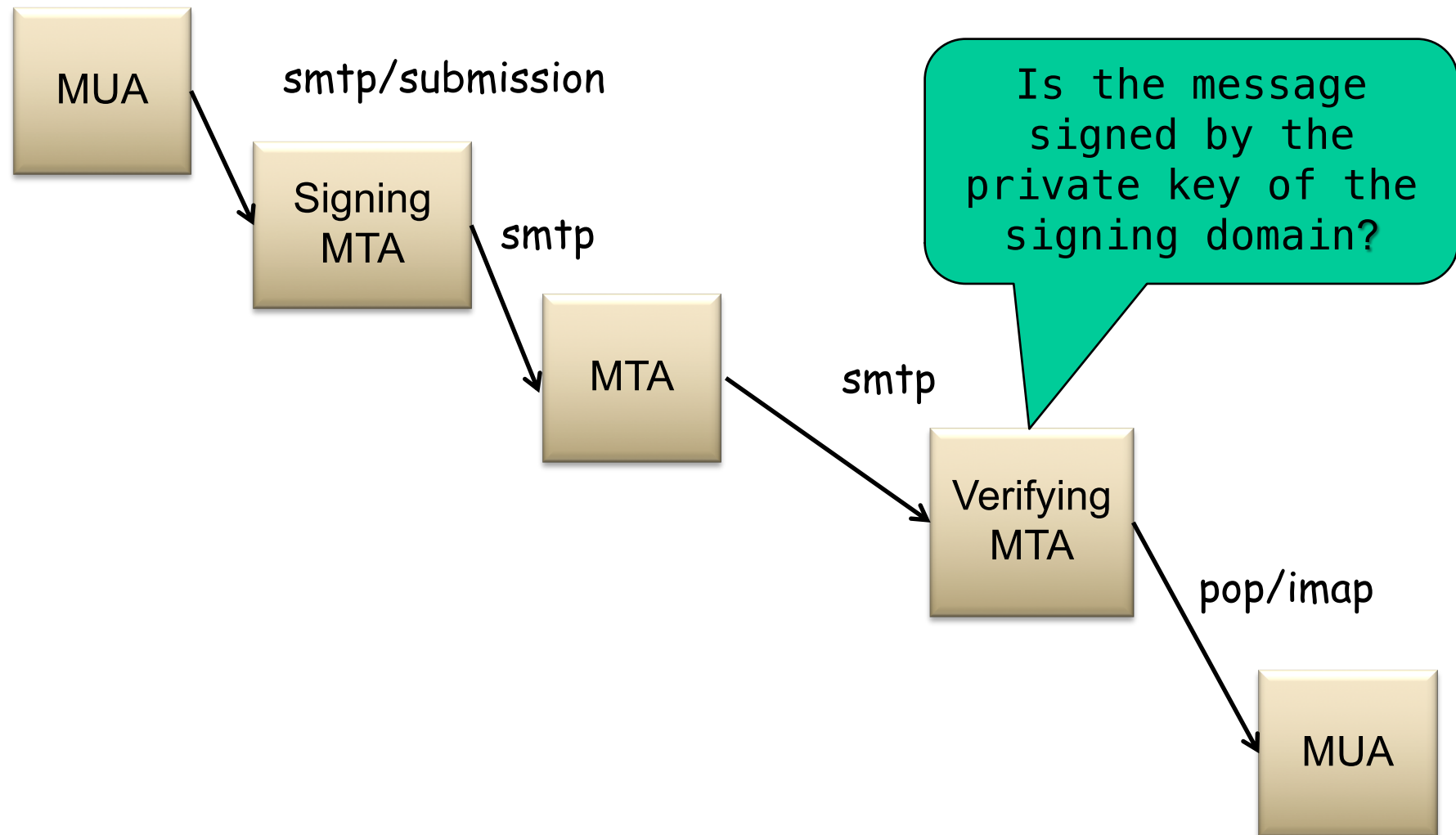
Key Question for SPF?

- ❑ How does SPF know if its neighbor MTA is a permitted sender of the domain?

DomainKeys Identified Mail (DKIM; RFC 5585)

- ❑ A domain-level digital signature authentication framework for email, using public key crypto
 - E.g., mail.sina.com signs that the message is sent by mail.sina server
- ❑ Basic idea of public key signature
 - Owner has both public and private keys
 - Owner uses private key to sign a message to generate a signature
 - Others with public key can verify signature
 - Assumption: difficult to get private key even w/ public key distributed

DomainKeys Identified Mail (DKIM)



Example: RSA

1. Choose two large prime numbers p, q .
(e.g., 1024 bits each)
2. Compute $n = pq$, $z = (p-1)(q-1)$
3. Choose e (with $e < n$) that has no common factors with z . (e, z are "relatively prime").
4. Choose d such that $ed-1$ is exactly divisible by z .
(in other words: $ed \bmod z = 1$).
5. Public key is (n, e) . Private key is (n, d) .

RSA: Signing/Verification

0. Given (n,e) and (n,d) as computed above
1. To sign message, m , compute $h = \text{hash}(m)$, then sign with private key
 $s = h^d \bmod n$ (i.e., remainder when h^d is divided by n)
2. To verify signature s , compute
 $h' = s^e \bmod n$ (i.e., remainder when s^e is divided by n)

Magic
happens!

$$h = (h^d \bmod n)^e \bmod n$$

The magic is a simple application of Euler's generalization of Fermat's little theorem

Key Question about DKIM?

- ❑ How does DKIM retrieve the public key of the author domain?

Summary: Some Key Remaining Issues about Email

- ❑ Basic: How to find the email server of a domain?
- ❑ Scalability/robustness: how to find multiple servers for the email domain?
- ❑ Security
 - SPF: How does SPF know if its neighbor MTA is a permitted sender of the domain?
 - DKIM: How does DKIM retrieve the public key of the author domain?