

---

# Network Applications: Email, DNS

Qiao Xiang

<https://qiaoxiang.me/courses/cnns-xmuf21/index.shtml>

9/30/2021

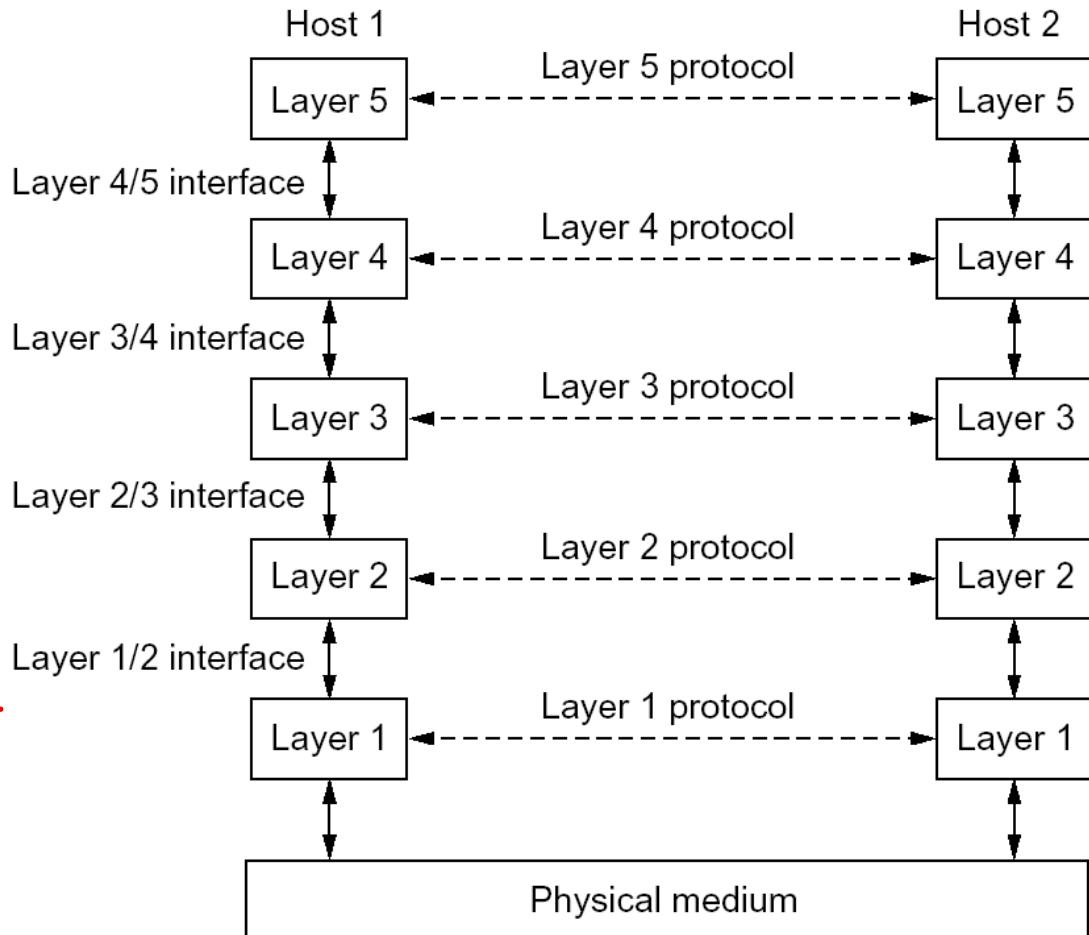
# Outline

---

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

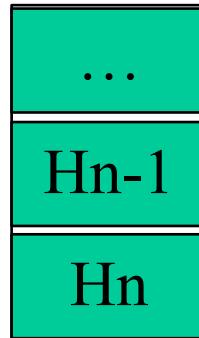
# Recap: Layering

- 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

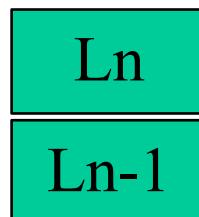


# Some Implications of Layered Architecture

- A packet as a stack container

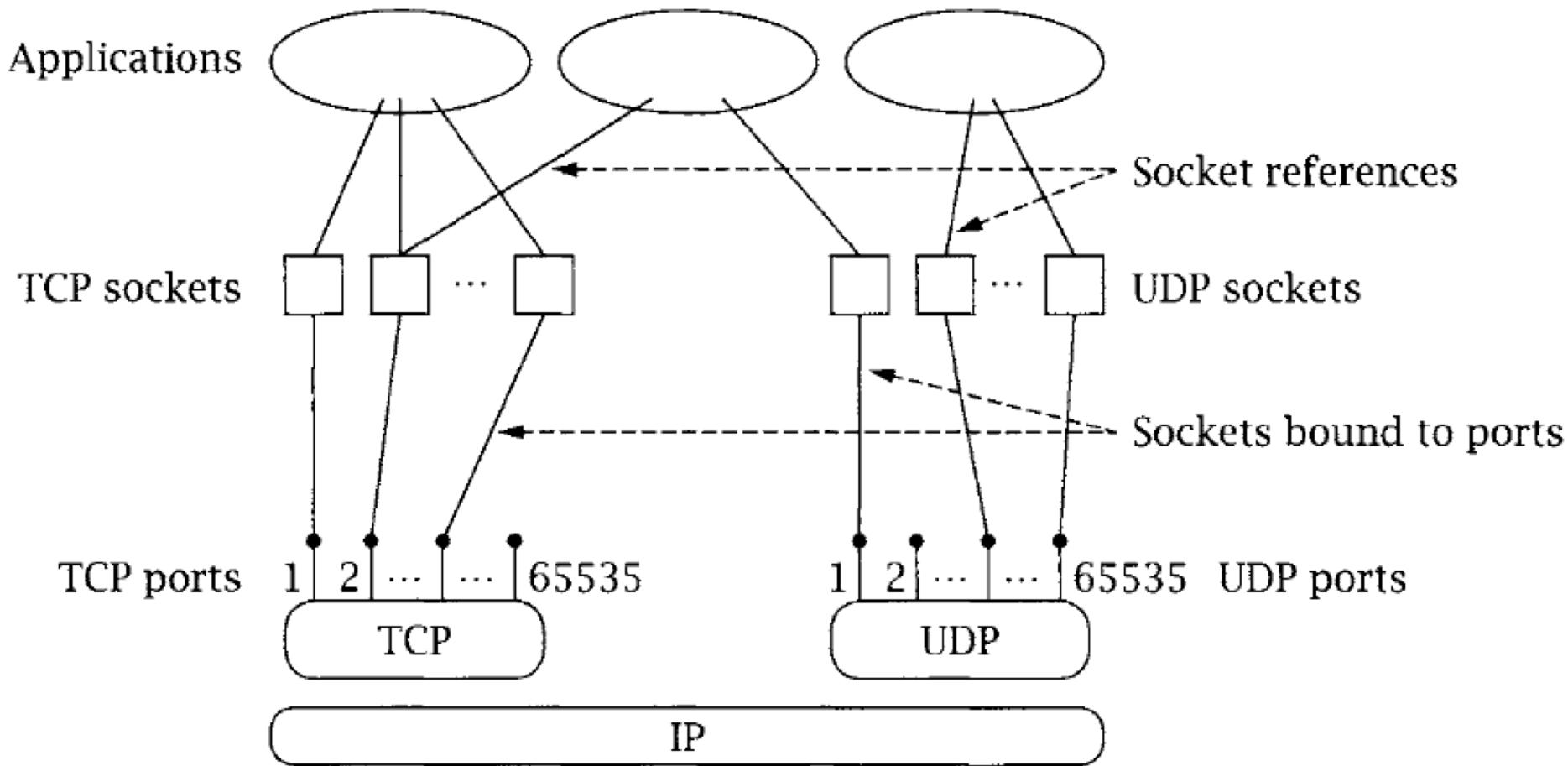


- Each layer needs multiplexing and demultiplexing to serve layer above



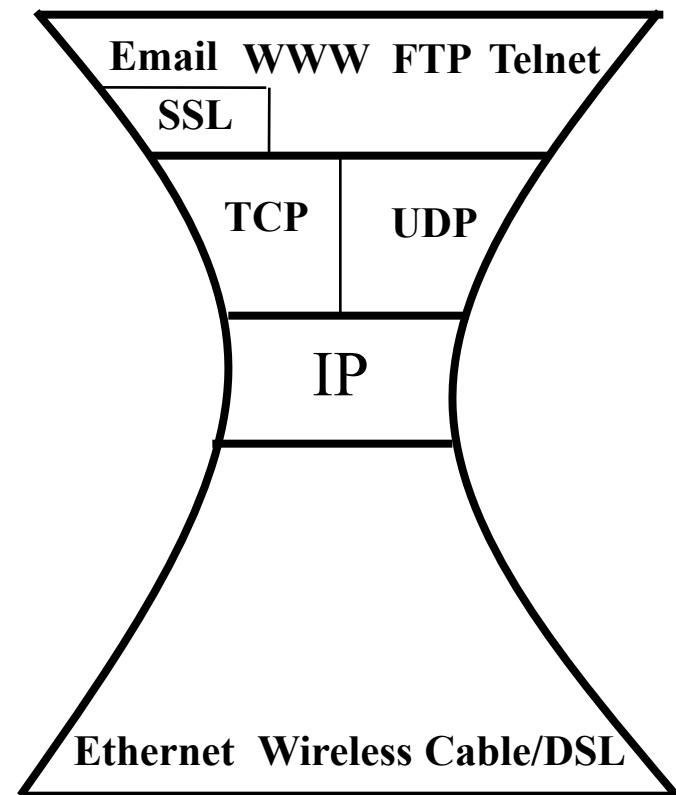
Has a field to indicate which higher layer requires the service

# Multiplexing/Demultiplexing

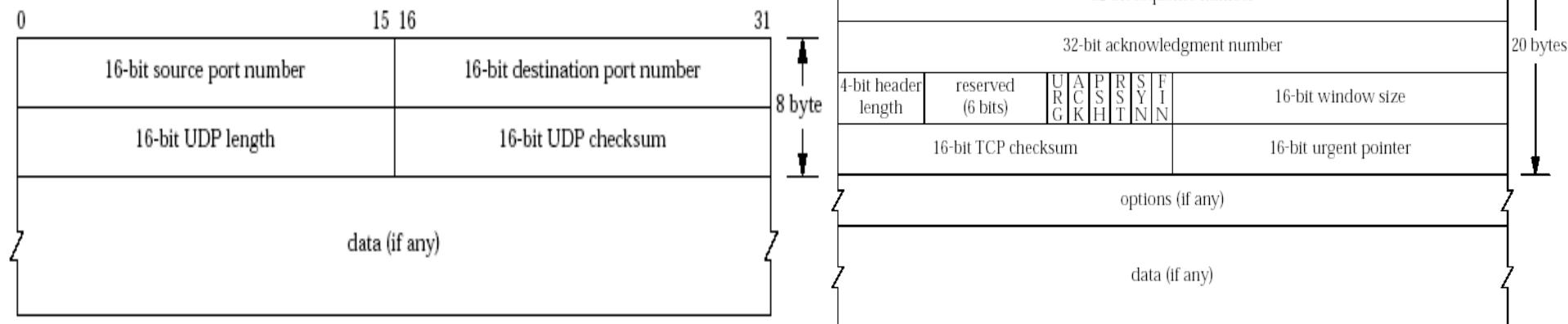


# Recap: The Big Picture of the Internet

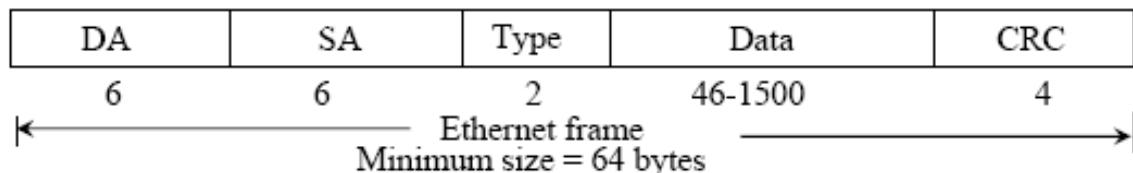
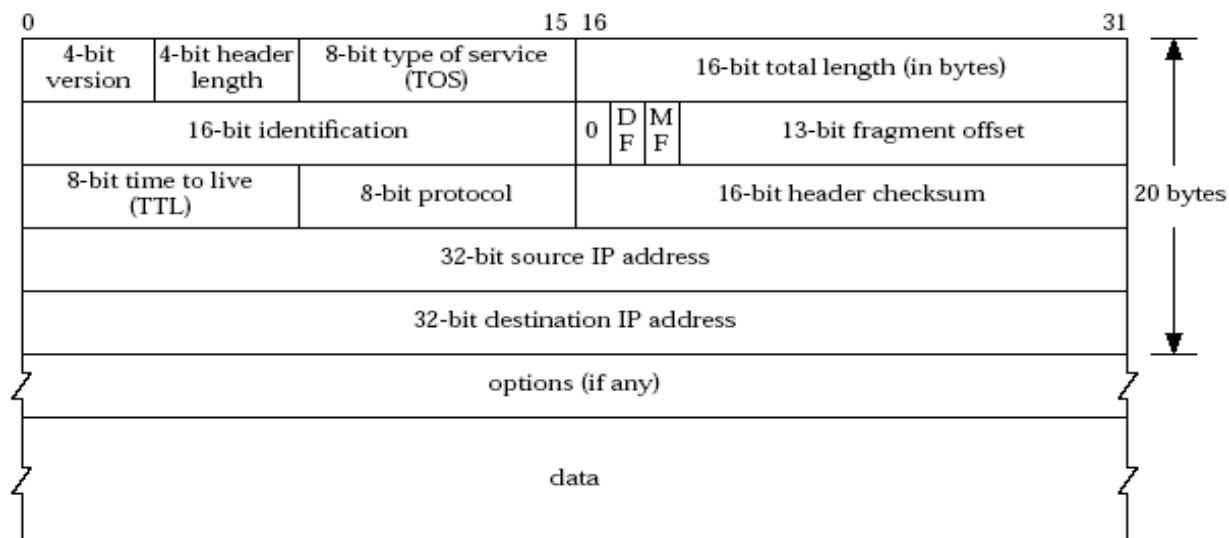
- Hosts and routers:
  - ~ 1 bill. hosts
  - organized into ~50K networks
  - backbone links 100 Gbps
- Software:
  - datagram switching with virtual circuit support
  - layered network architecture
    - use end-to-end arguments to determine the services provided by each layer
    - the 5-layer hourglass architecture of the Internet



# Formats of main protocols



**Tip: services map to header fields**



# 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

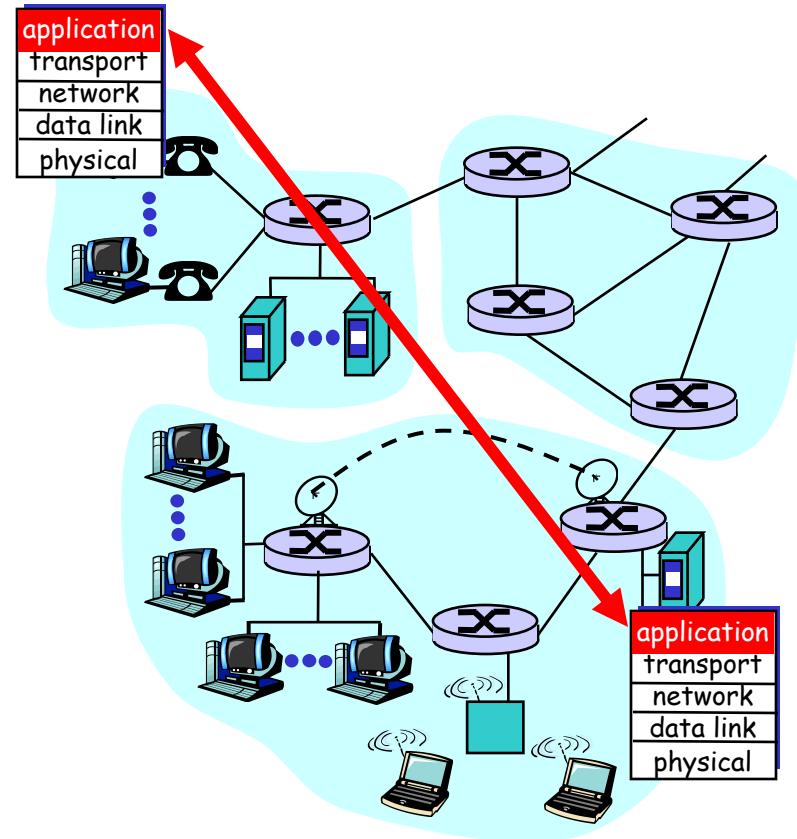
# Network Applications vs. Application-layer Protocols

**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

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

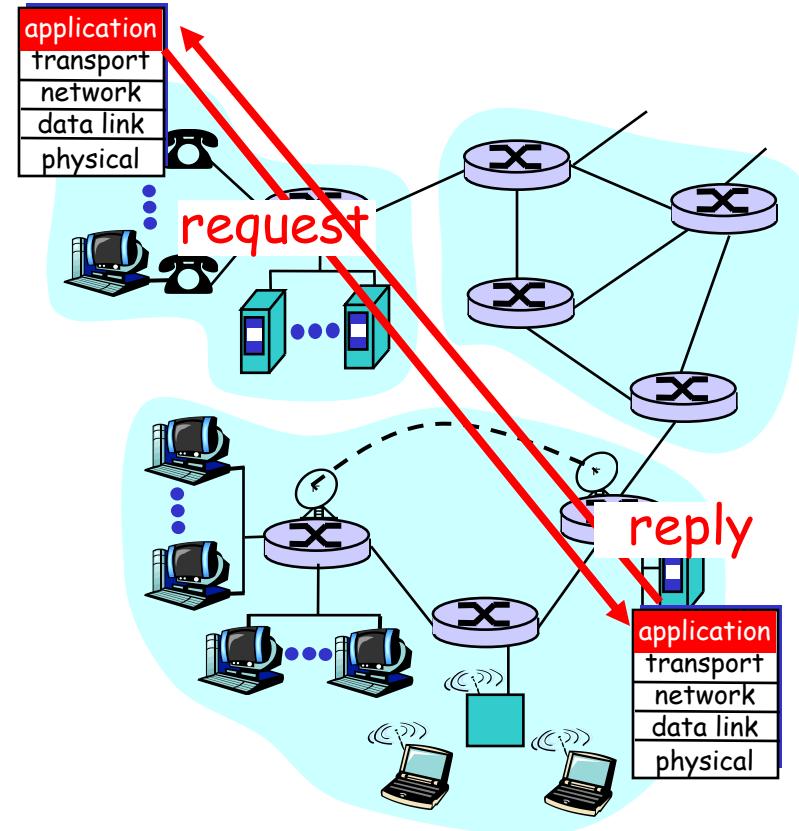
Typical network app has two pieces: *client* and *server*

## 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):

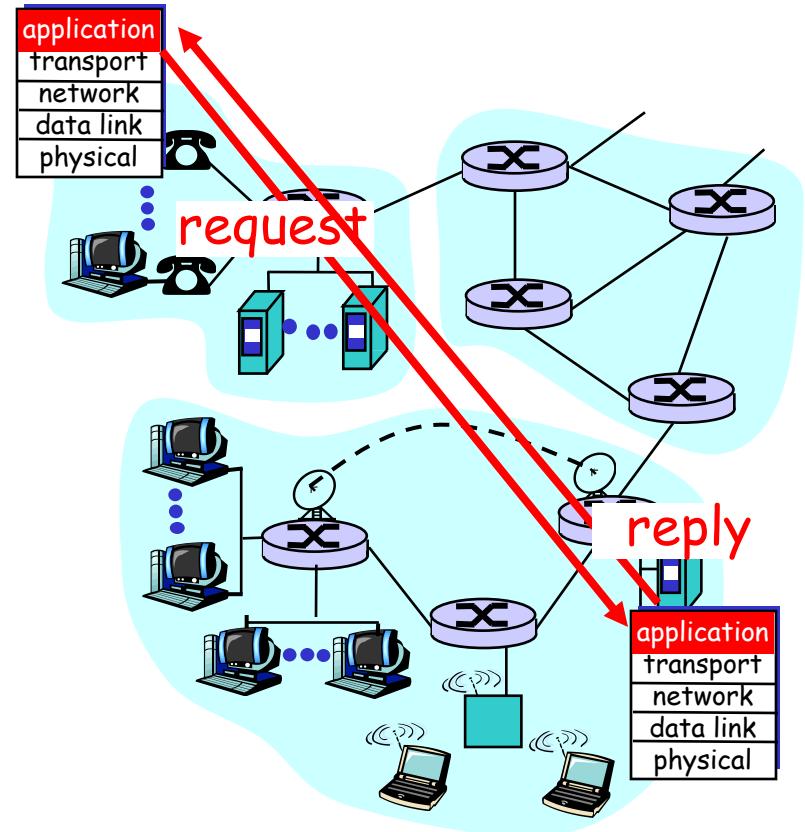
- provides requested service to client
- e.g., Web server sends requested Web page; mail server delivers e-mail



# 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

# Recall: SMTP

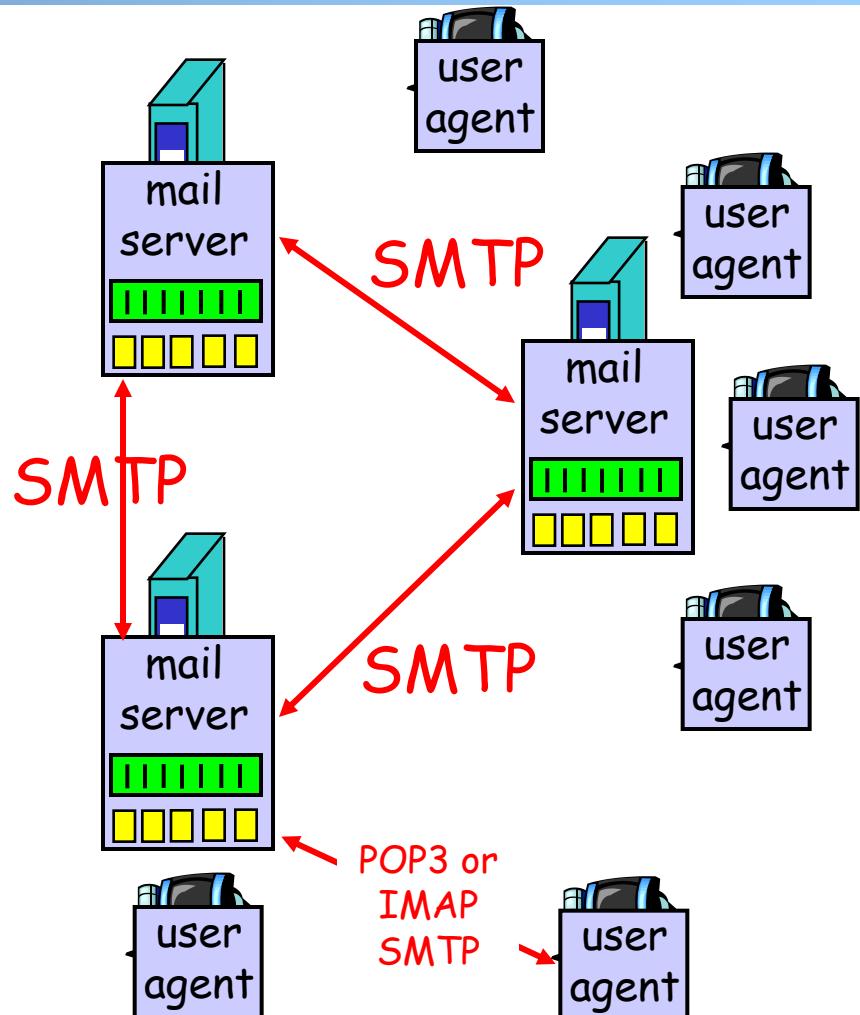
```
C: auth login
S: 334 VXNlcM5hbWU6
C: eG11Y25ucw==
S: 334 UGFzc3dvcmQ6
C: MzM0ZjU2MDVkJzE1MDRmOQ==
S: 235 OK Authenticated
C: mail from:xmucnns@sina.com
S: 250 ok
C: rcpt to:qiaoxiang@xmu.edu.cn
S: 250 ok
C: data
S: 354 End data with <CR><LF>.<CR><LF>
C: Date:2021-9-22 12:36
C: From:xmucnns@sina.com
C: To:qiaoxiang@xmu.edu.cn
C: Subject:test smtp
C:
C: Hello, Qiao.
C:
C: .
S: 250 ok queue id 11479549283321
C: quit
S: 221 smtp-97-27.smtpsmail.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

**MUA:** User Agent

**Mediator:** User-level Relay

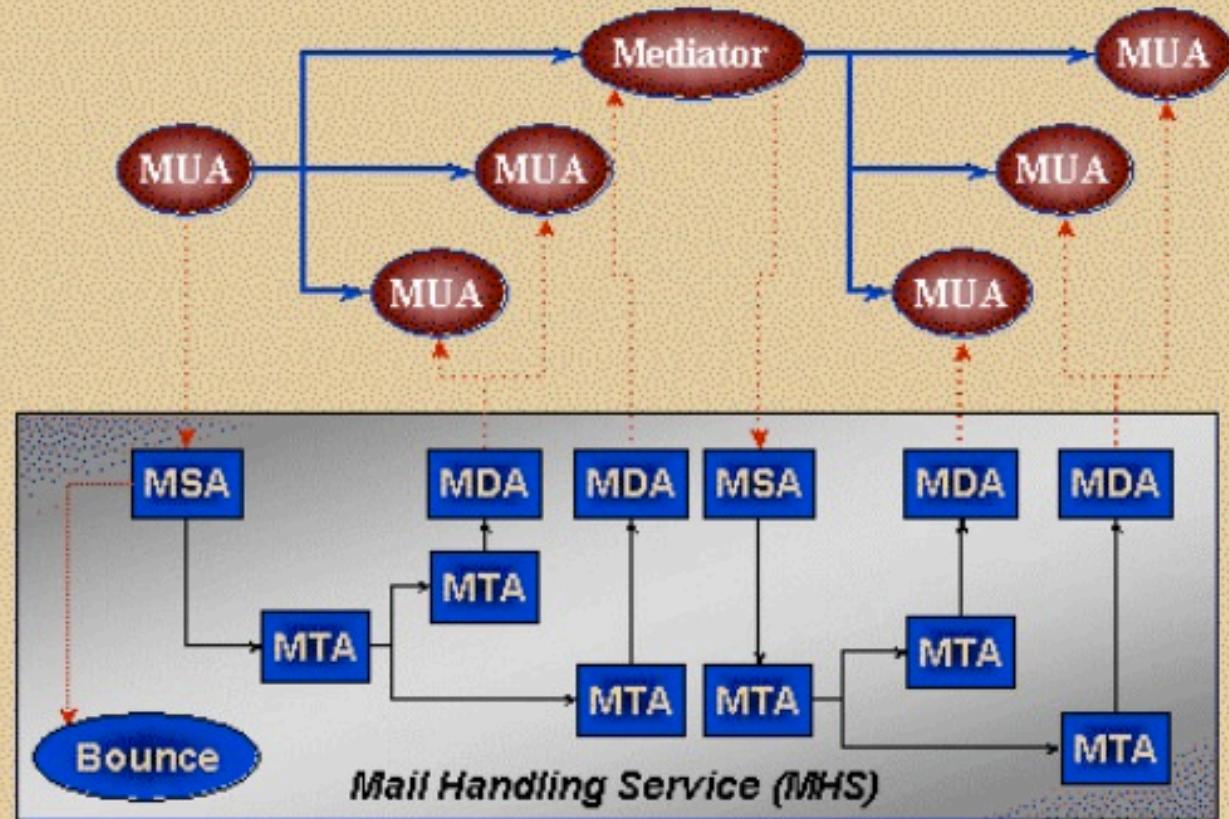
**MHS:** Mail Handling (transit) Service

**MSA:** Submission

**MTA:** Transfer

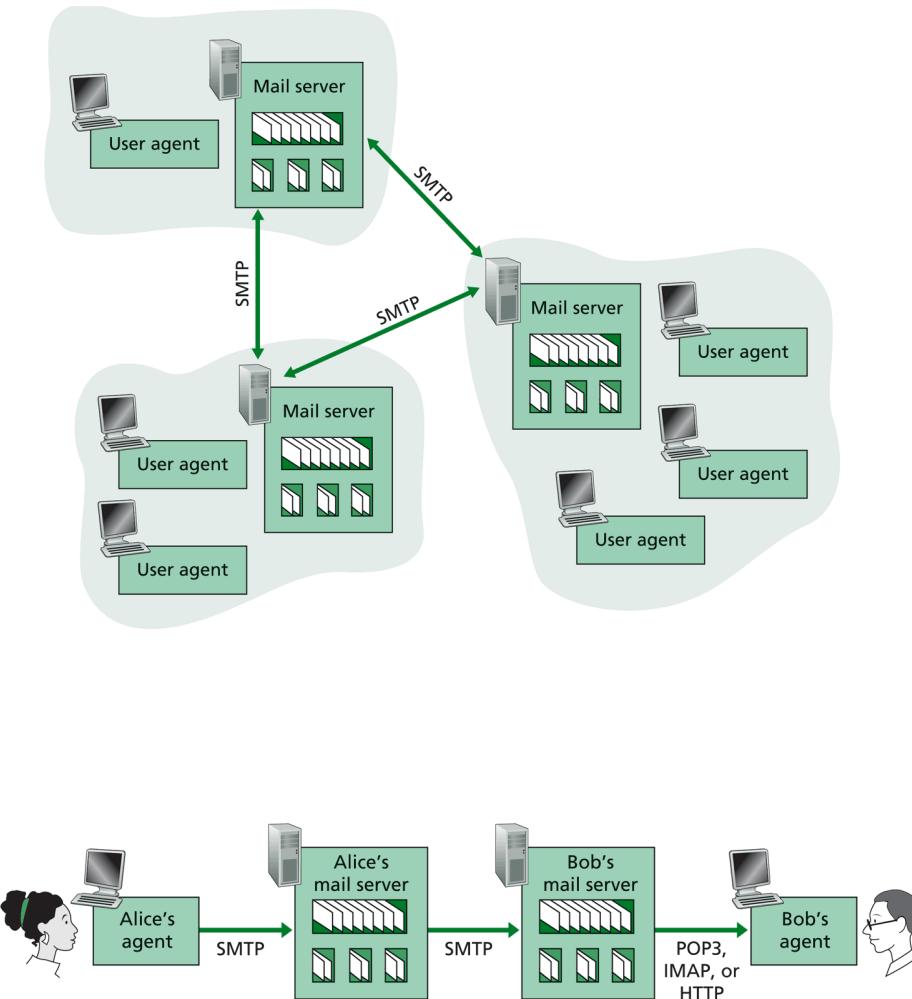
**MDA:** Delivery

**Bounce:** Returns



# SMTP: Mail Transport Protocol

## Messages (Envelop Messages)



C: auth login  
S: 334 VXNlcmlhbWU6  
C: eG11Y25ucw==  
S: 334 UGFzc3dvcmQ6  
C: MzM0ZjU2MDVkJE1MDRmOQ==  
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: Connectio

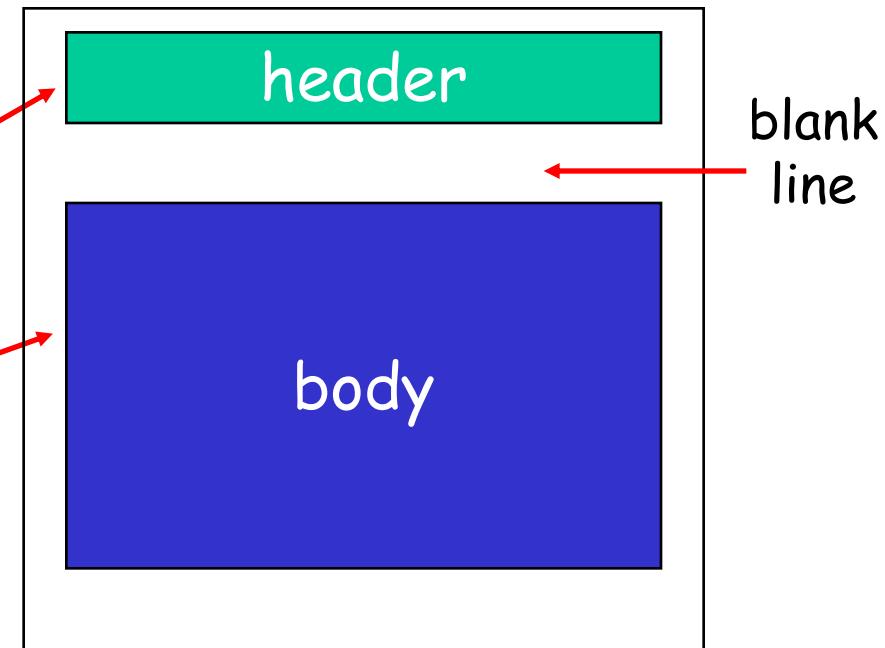
Email text different from  
SMTP protocol message

# Mail Message Data

SMTP: protocol for  
exchanging email msgs

RFC 822: standard for text  
message format:

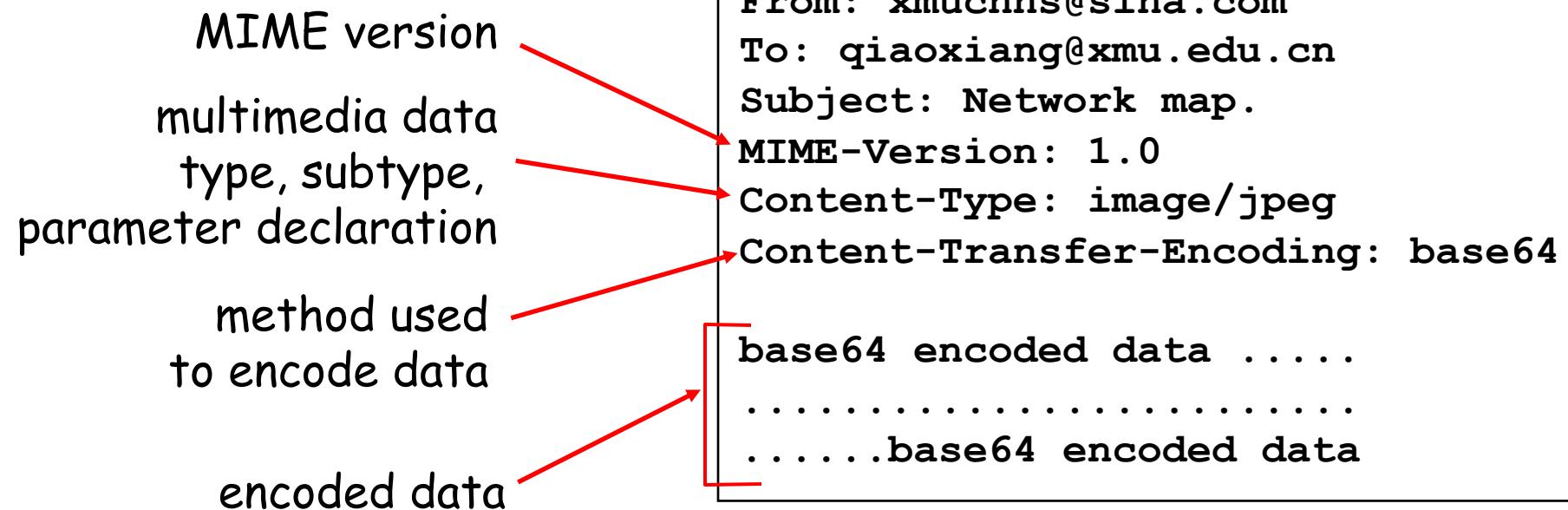
- Header lines, e.g.,
  - To:
  - From:
  - Subject:
- Body
  - the “message”, ASCII  
characters only



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



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*

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*

## Transaction phase, client:

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

# Exercise

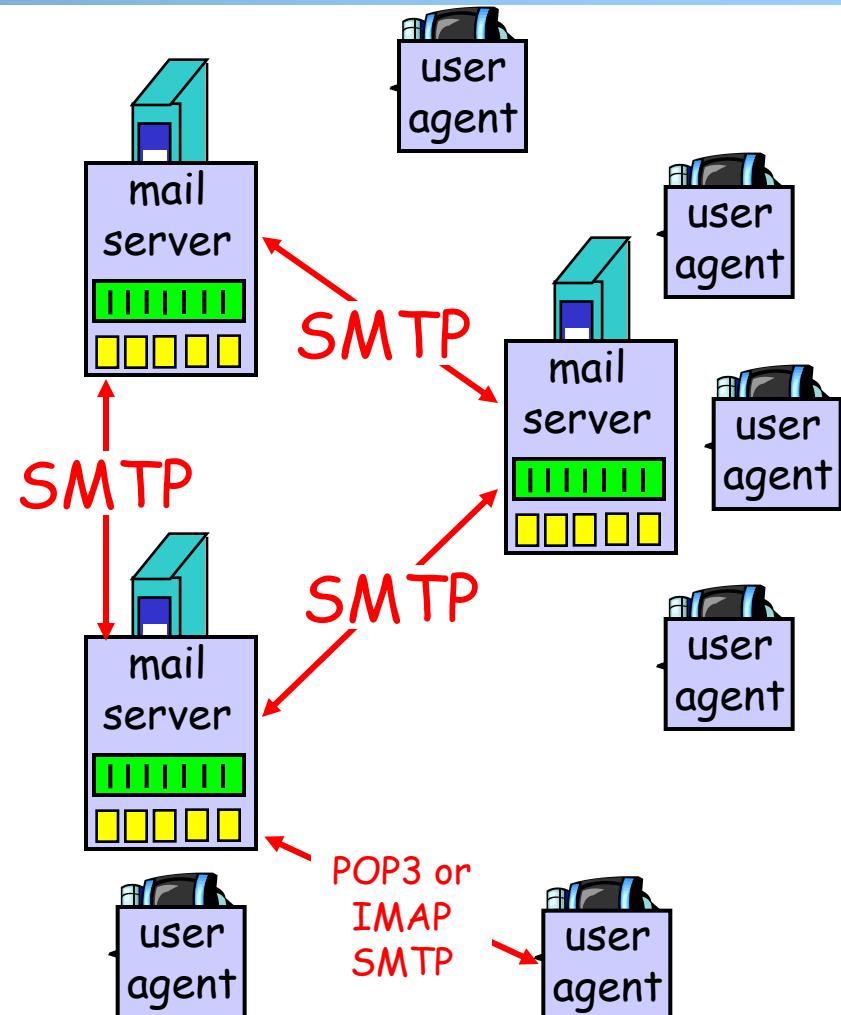
---

- 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

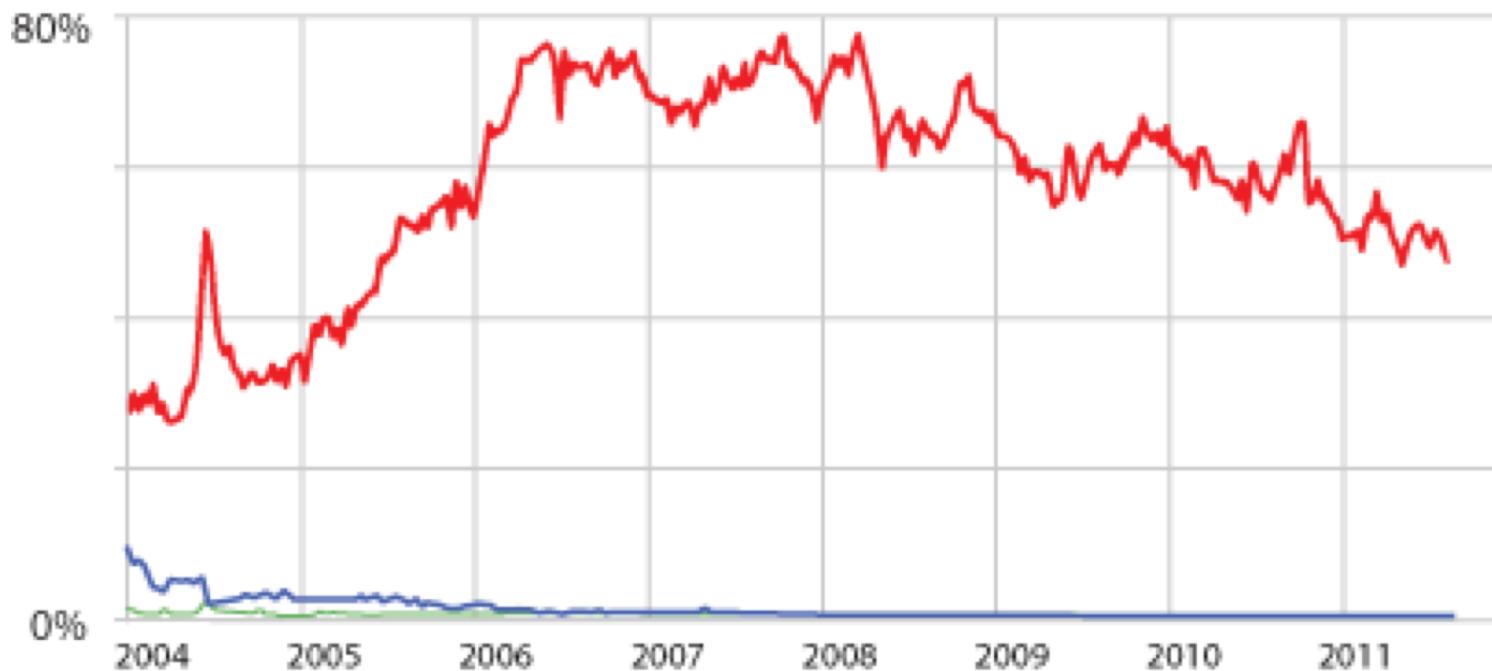
Key questions to ask about a C-S application

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

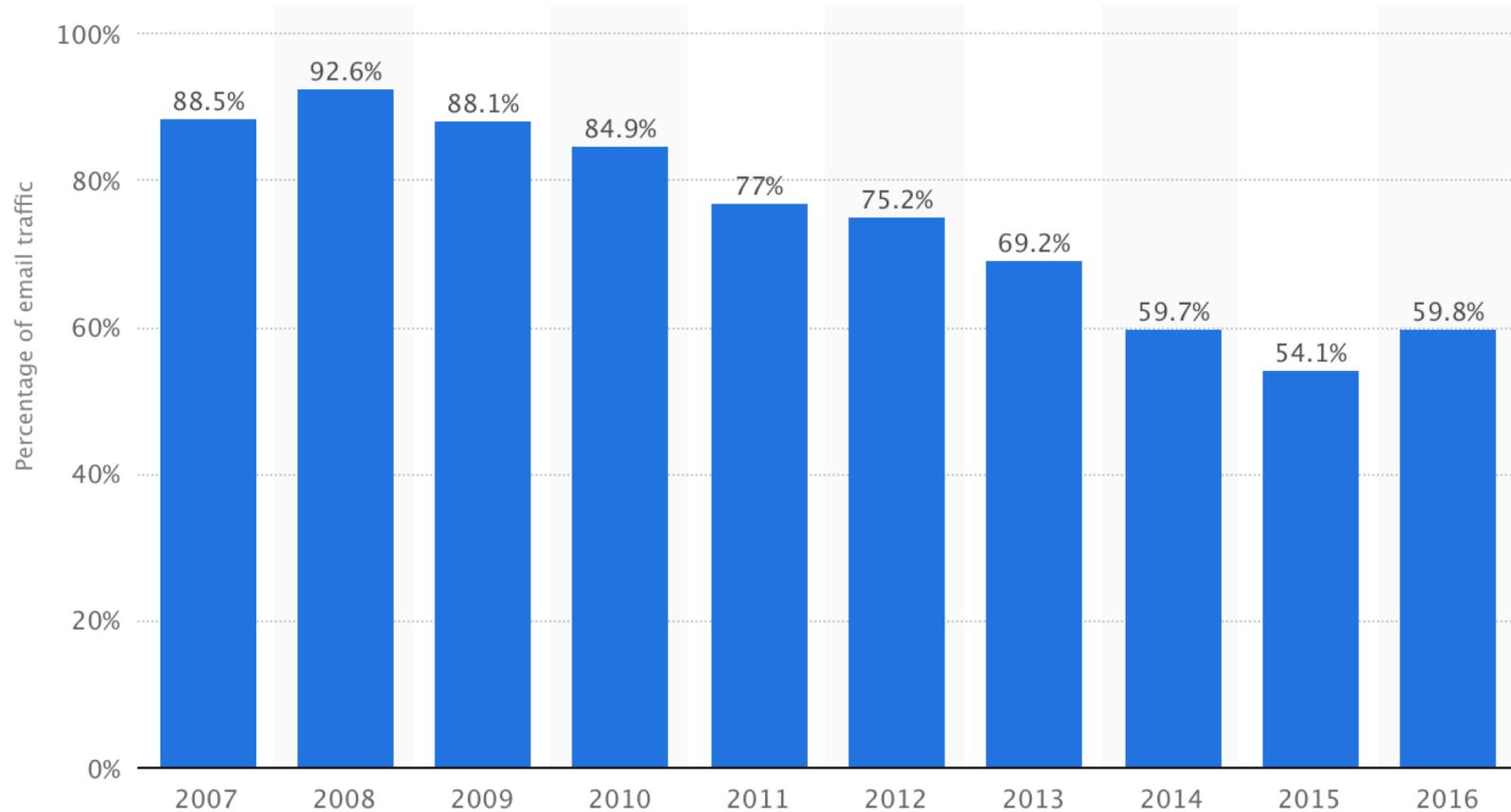


# Email Security: Spam

□ Spam (Google)



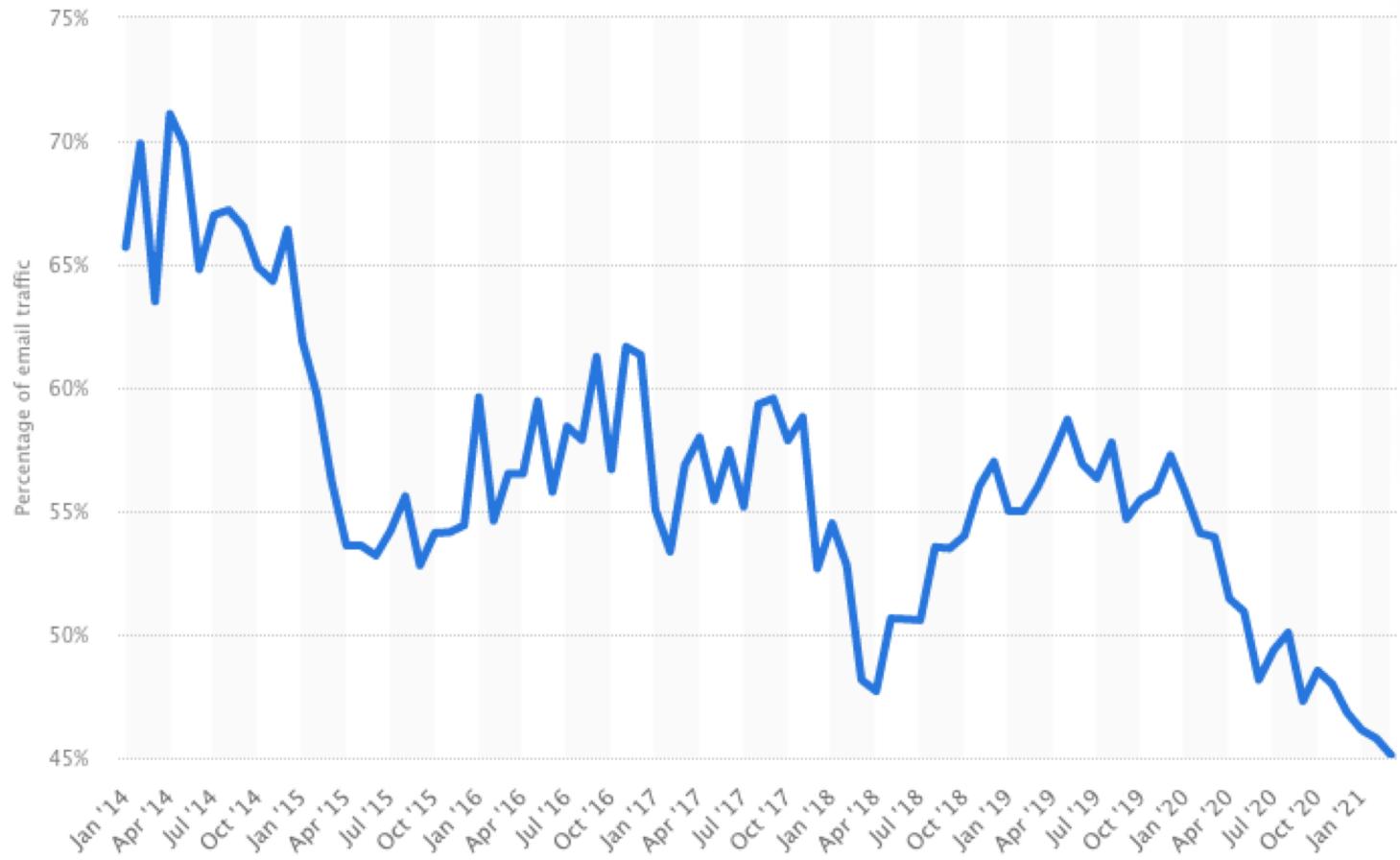
# Email Security Issue: Spam



© 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

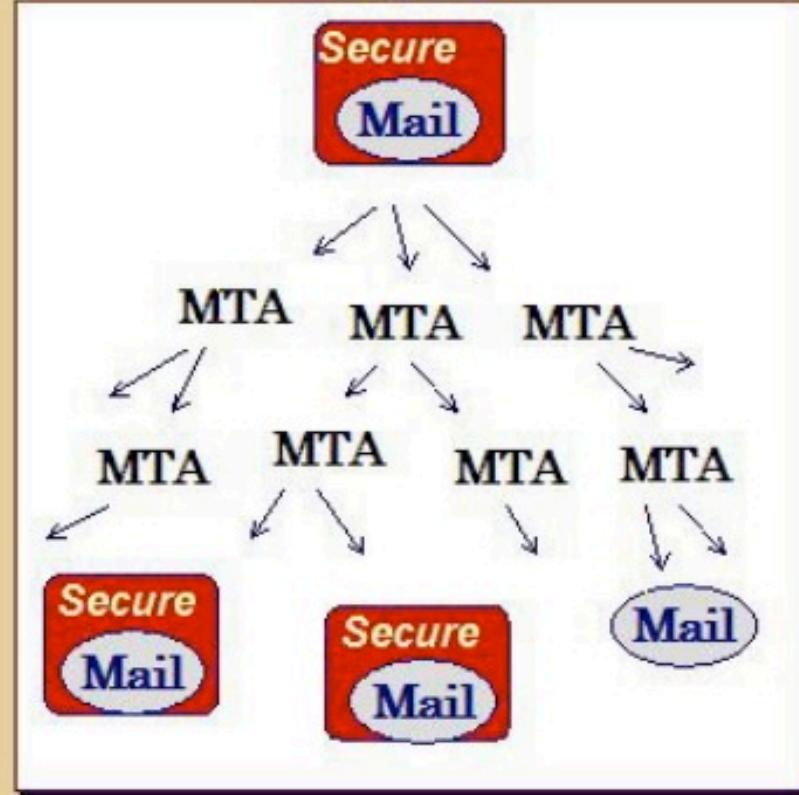
<https://support.google.com/mail/answer/1366858?hl=en>

# Email Authentication Approaches

**Channel**



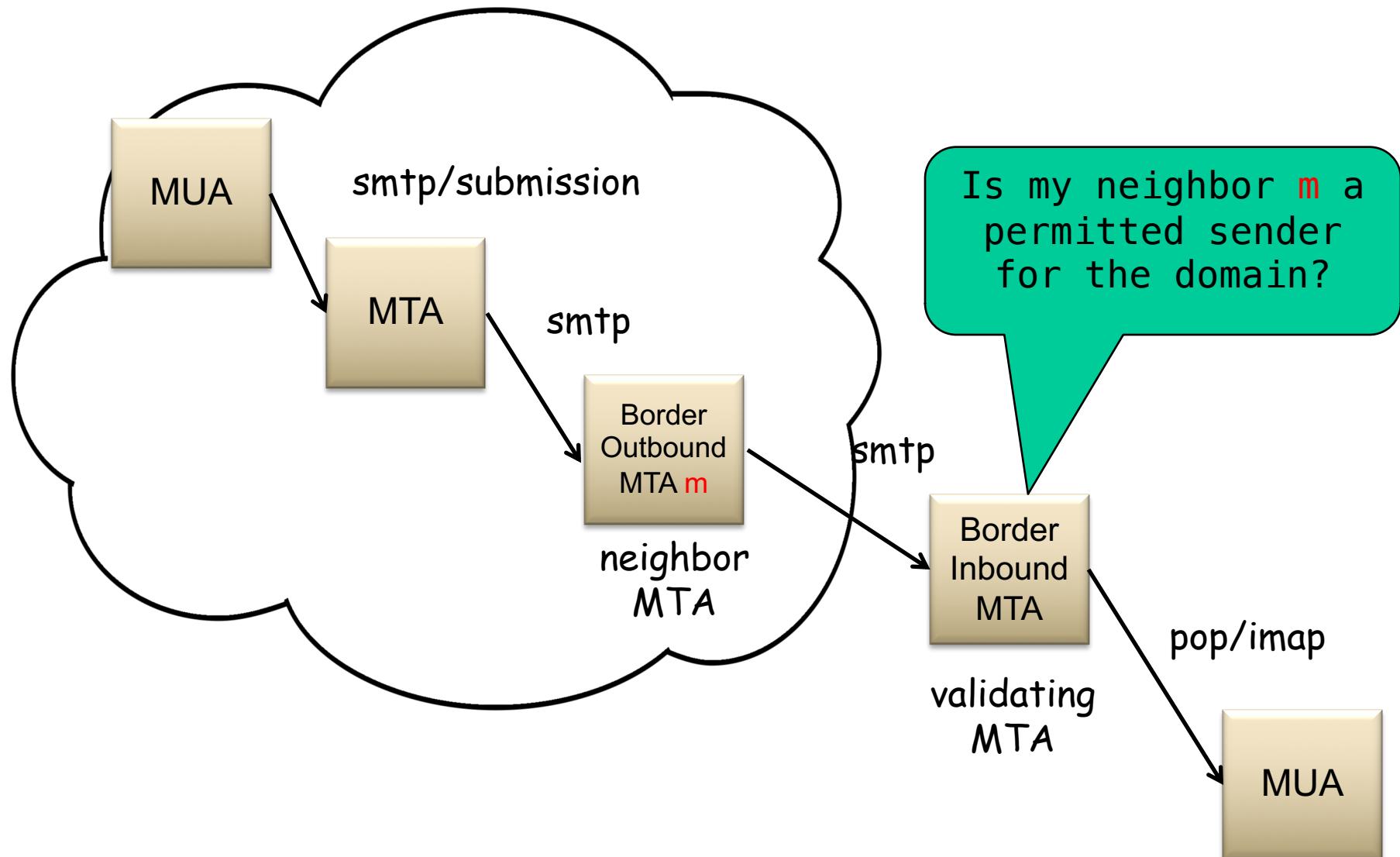
**Object**



Sender Policy Frame (SPF)

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

## Sender Policy Framework (SPF RFC7208)



## Key Question for SPF?

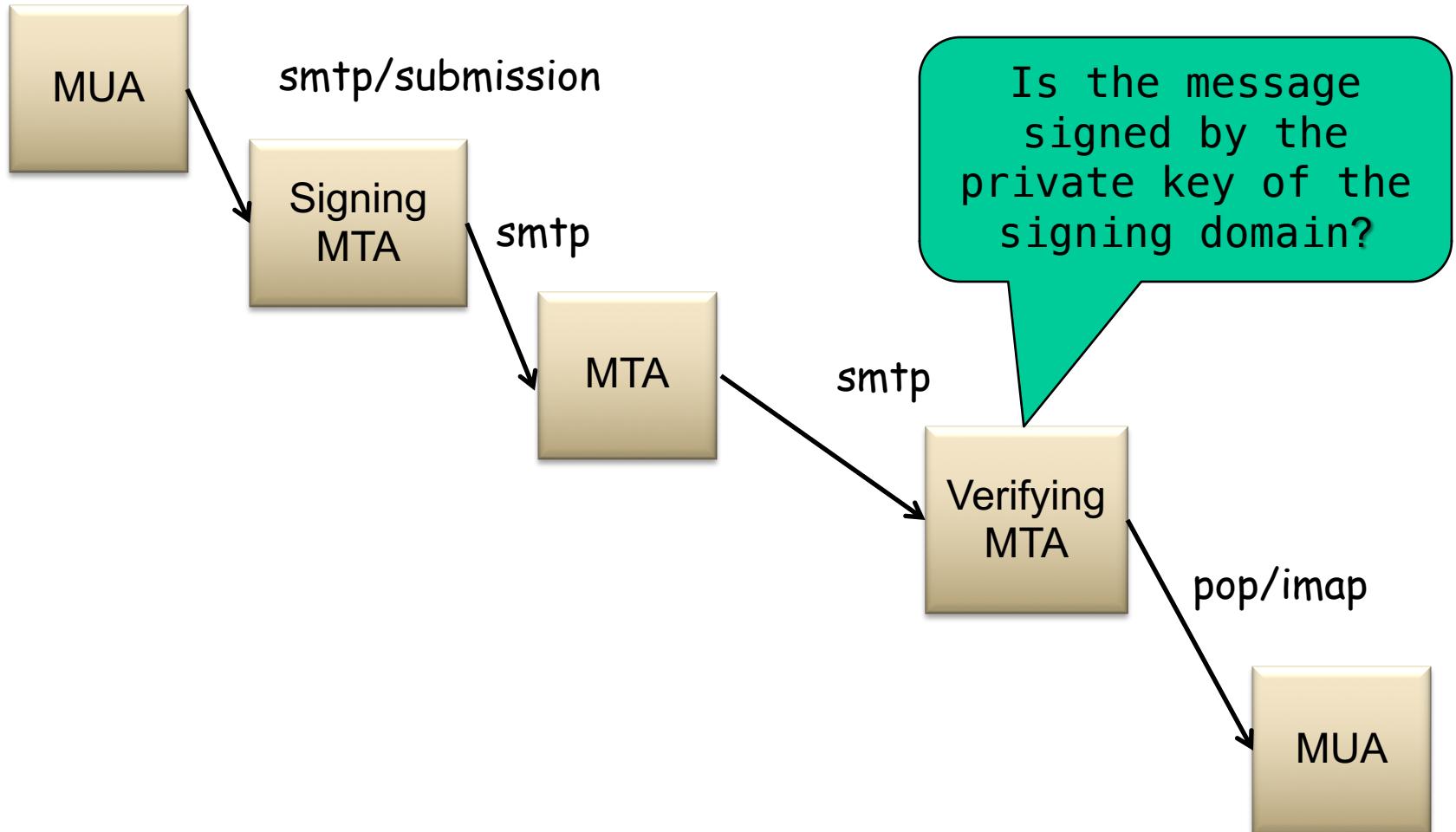
---

- 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

$s = h^d \text{mod } n$  (i.e., remainder when  $h^d$  is divided by  $n$ )

2. To verify signature  $s$ , compute

$h' = s^e \text{mod } n$  (i.e., remainder when  $s^e$  is divided by  $n$ )

Magic happens!

$$h = (h^d \text{mod } n)^e \text{ mod } n$$

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

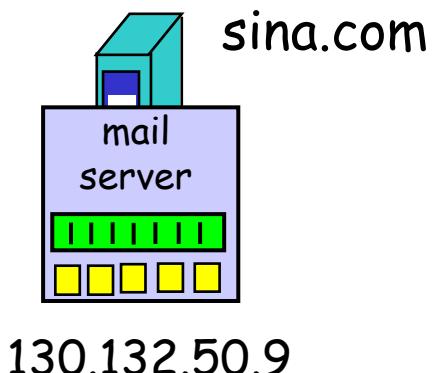
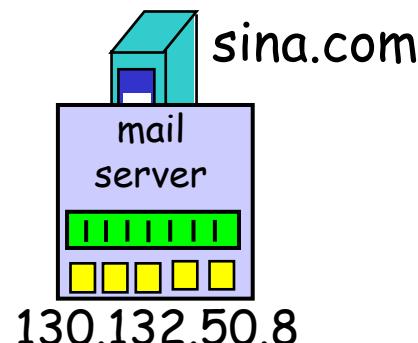
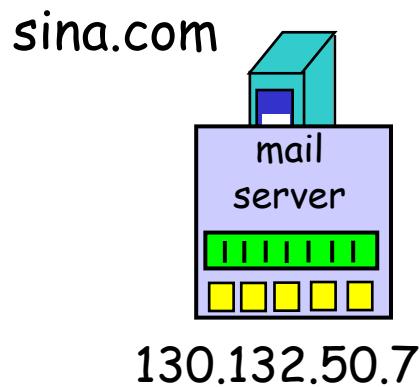
- Basic: How to find the email server of a domain?
- Scalability/robustness: how to find multiple servers for the email domain?
- Security
  - SPF: How does SPF know if its neighbor MTA is a permitted sender of the domain?
  - DKIM: How does DKIM retrieve the public key of the author domain?

# Scalability/R robustness

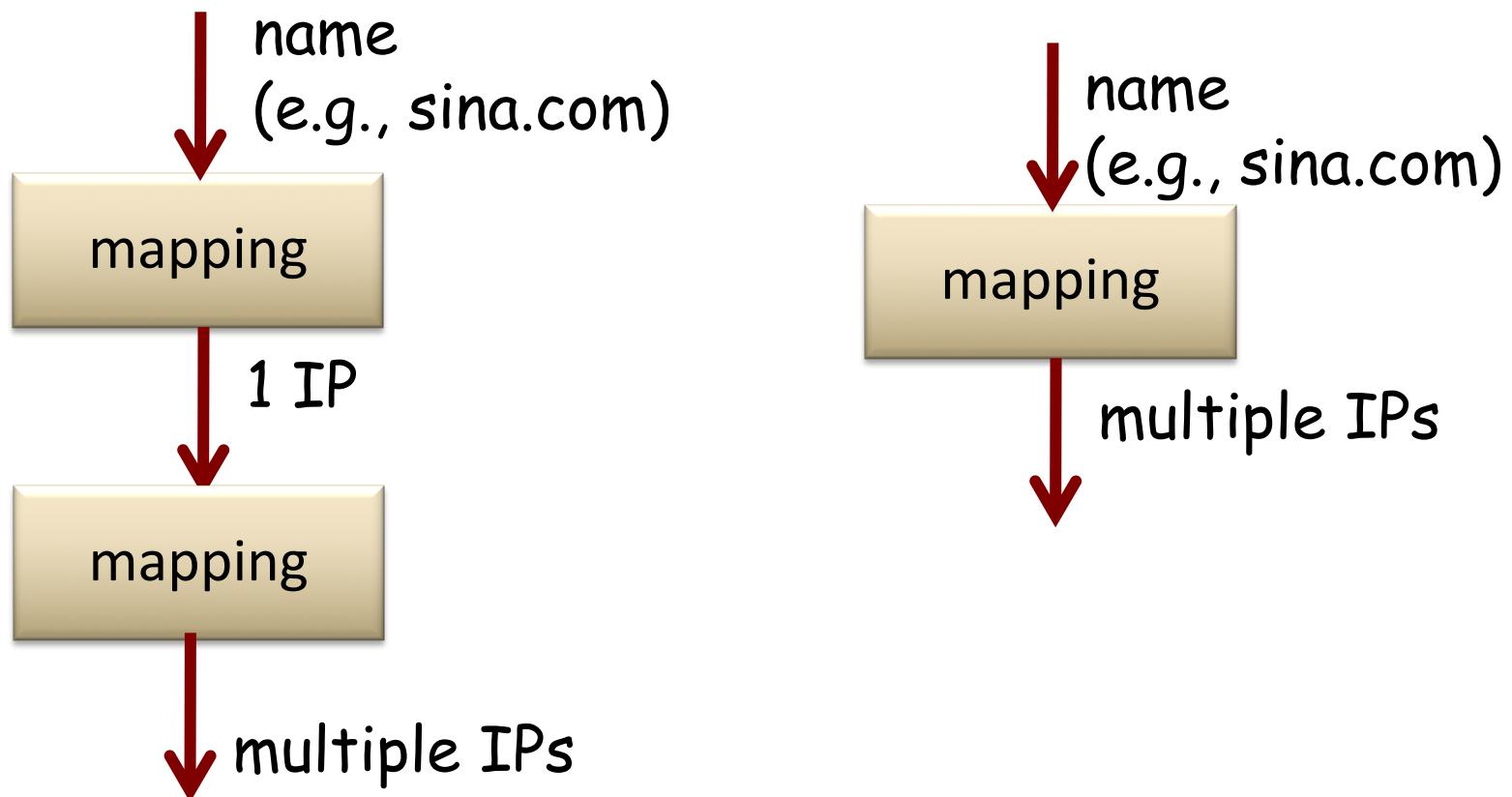
- ❑ Both scalability and robustness require that multiple email servers serve the same email address

 client need an email server's IP address

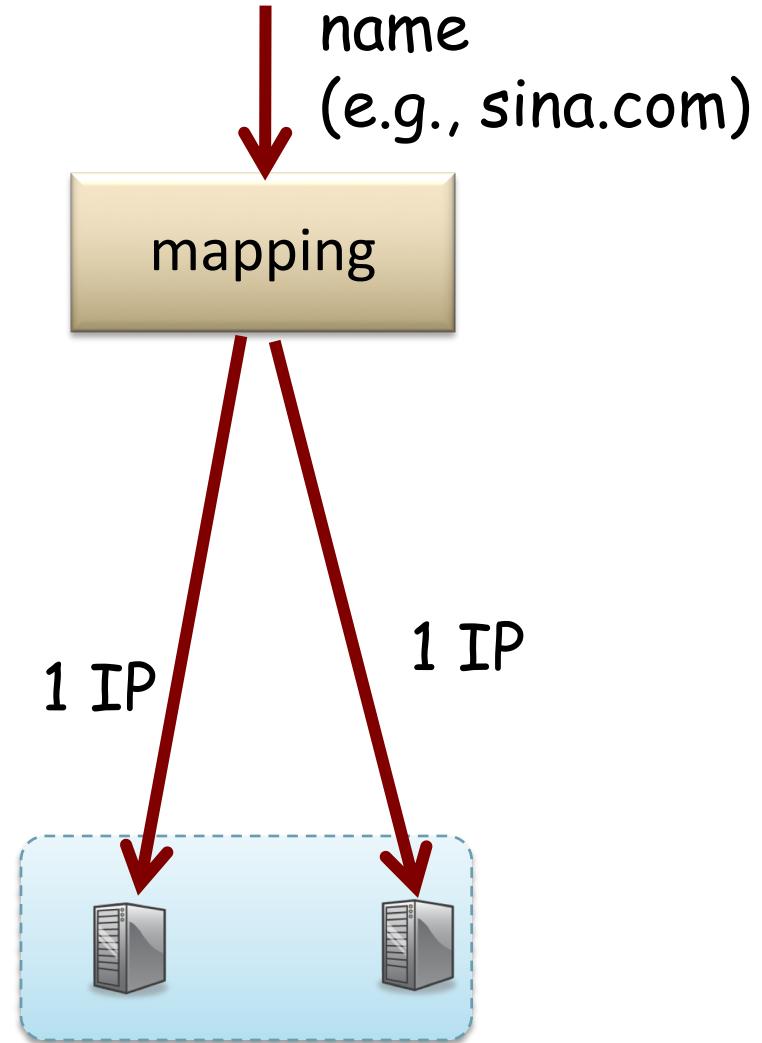
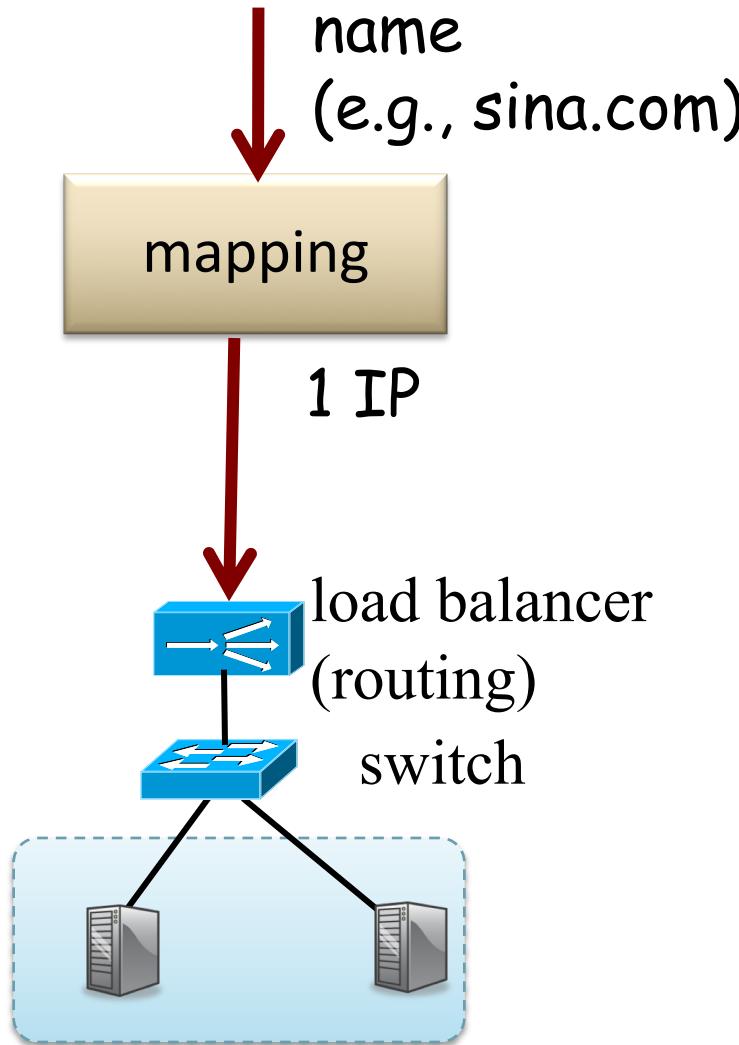
mapping



# Mapping Functions Design Alternatives



# Mapping Functions Design Alternatives



# Outline

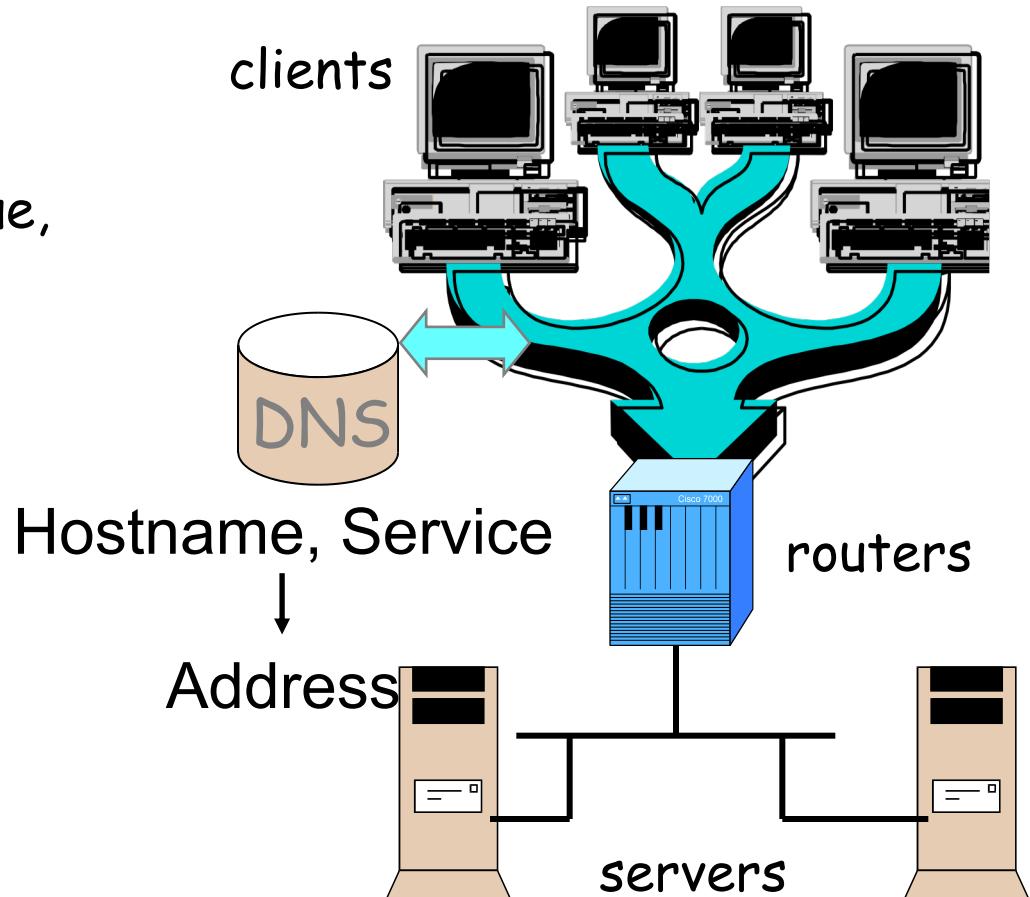
---

- Admin. and recap
- Layered network architecture
- Application layer overview
- Network applications
  - Email
  - *DNS*

# DNS: Domain Name System

## □ Function

- map between (domain name, service) to value,  
e.g.,
  - (xmu.edu.cn, addr)  
→ 210.34.0.35
  - (xmu.edu.cn, email)  
→ cmsn1.xmu.edu.cn



# DNS Records

<http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml>

DNS: stores resource records (**RR**)

RR format: (**name**, **type**, **value**, **ttl**)

- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. xmu.edu.cn)
  - **value** is the name of the authoritative name server for this domain
- Type=TXT
  - general txt
- Type=CNAME
  - **name** is an alias of a “canonical” (real) name
  - **value** is canonical name
- Type=MX
  - **value** is hostname of mail server associated with **name**
- Type=SRV
  - general extension for services
- Type=PTR
  - a pointer to another name

# Discussion

---

- Can DNS handle multiple values for the same (name, service)?

# Try DNS: Examples

---

- dig <name> <type>
  - Try xm.edu.cn / others and various types
  
- dig <domain> txt to retrieve spf

<http://www.zytrax.com/books/dns/ch9/spf.html>

# Observations

---

- MX can return multiple servers
- DNS may rotate the servers in answer
- Address can also return multiple addresses
- SPF is encoded as the txt type