Internet Exchange Points (IXPs)

Introduction to Internet Exchange Points

- A bit of history
- What are they?
- Why use them?
- Design Considerations

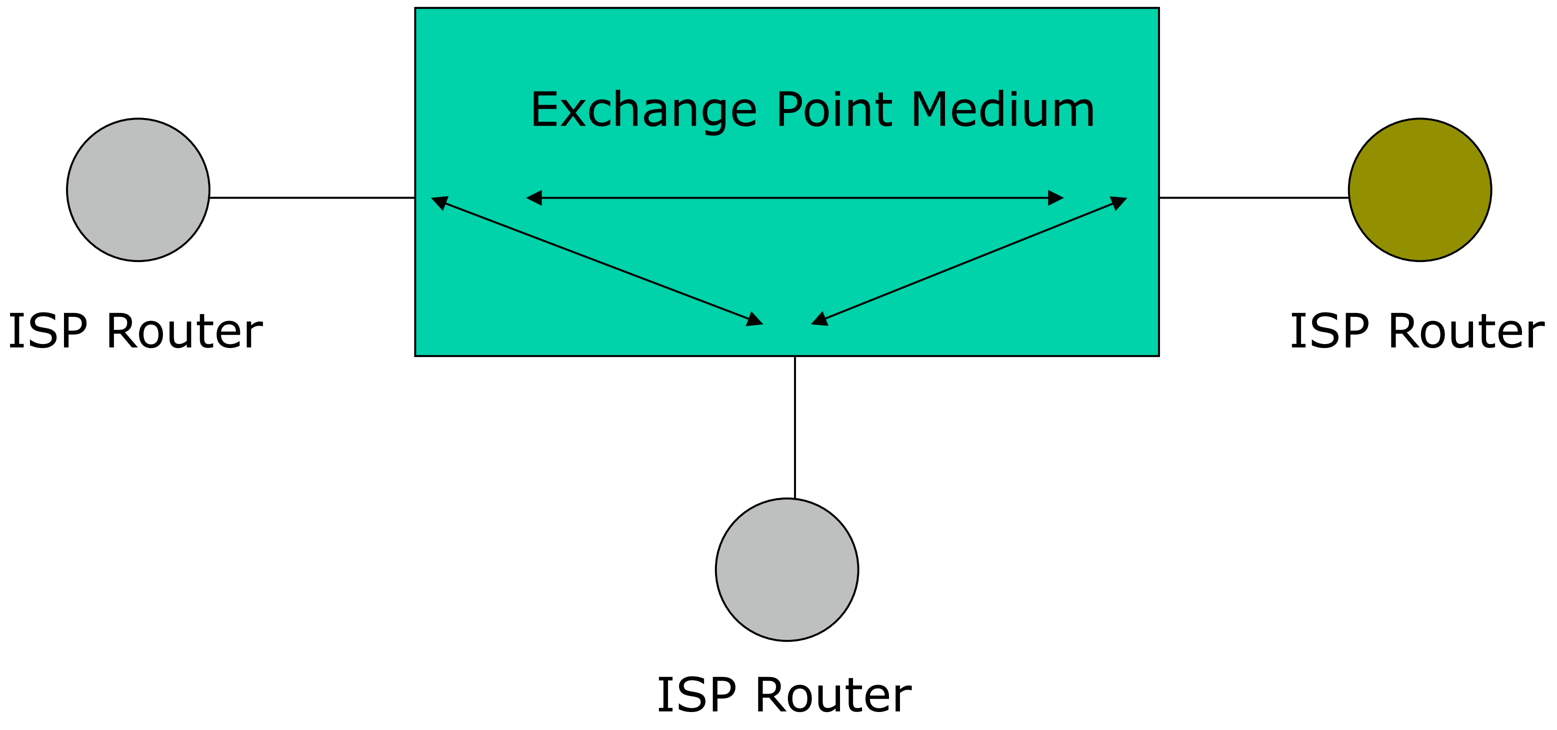
A Bit of History…

- End of NSFnet – one major backbone
- Move towards commercial Internet
  - Private companies selling their bandwidth
- Need for coordination of routing exchange between providers
  - Traffic from ISP A needs to get to ISP B
- Routing Arbiter project created to facilitate this

What is an Exchange Point

- Network Access Points (NAPs) established at end of NSFnet
  - The original “exchange points”
- Major providers connect their networks and exchange traffic
- High-speed network or ethernet switch
- Simple concept – any place where providers come together to exchange traffic

Conceptual Diagram of an IXP



Internet Exchange Point
Why peer?

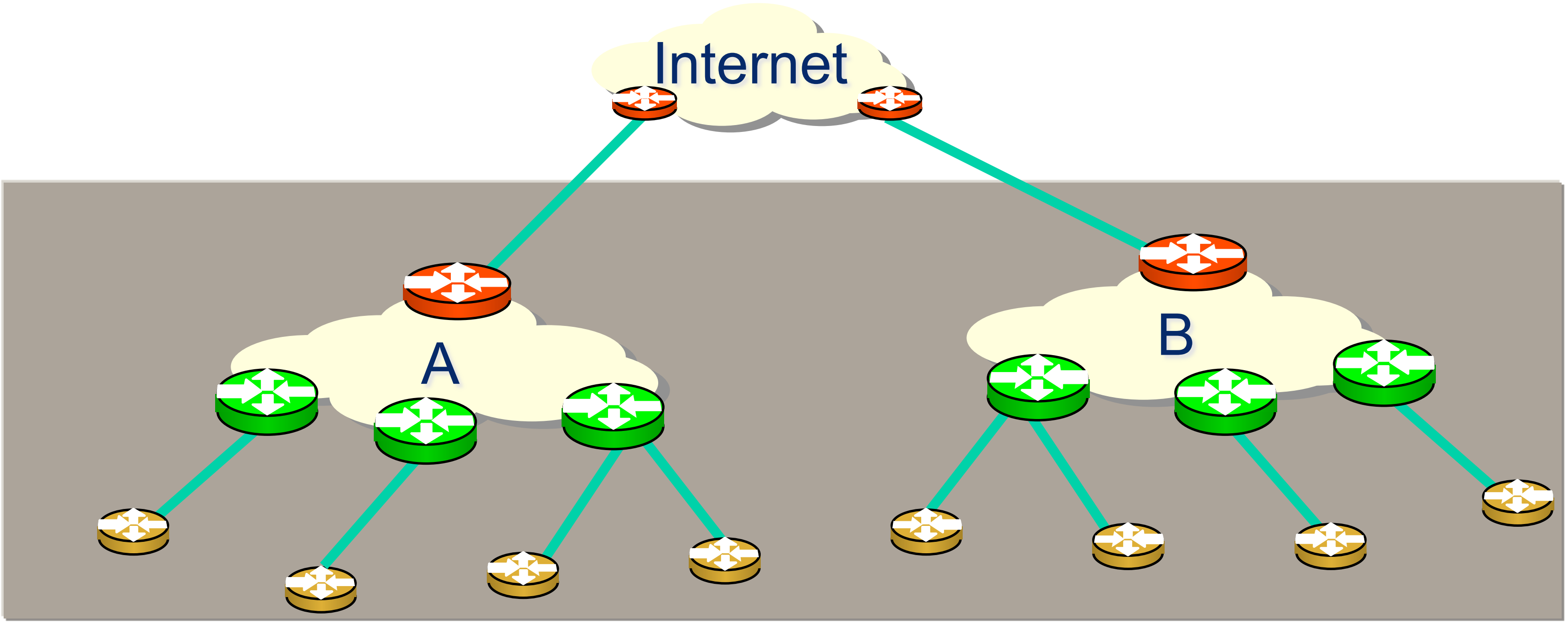
- Consider a region with one ISP
  - They provide internet connectivity to their customers
  - They have one or two international connections
- Internet grows, another ISP sets up in competition
  - They provide internet connectivity to their customers
  - They have one or two international connections
- How does traffic from customer of one ISP get to customer of the other ISP?
  - Via the international connections

Internet Exchange Point
Why peer?

- Yes, International Connections…
  - If satellite, RTT is around 550ms per hop
  - So local traffic takes over 1s round trip
- International bandwidth…
  - Costs order of magnitude or two more than domestic bandwidth
  - Becomes congested with local traffic
- Wastes money, harms performance

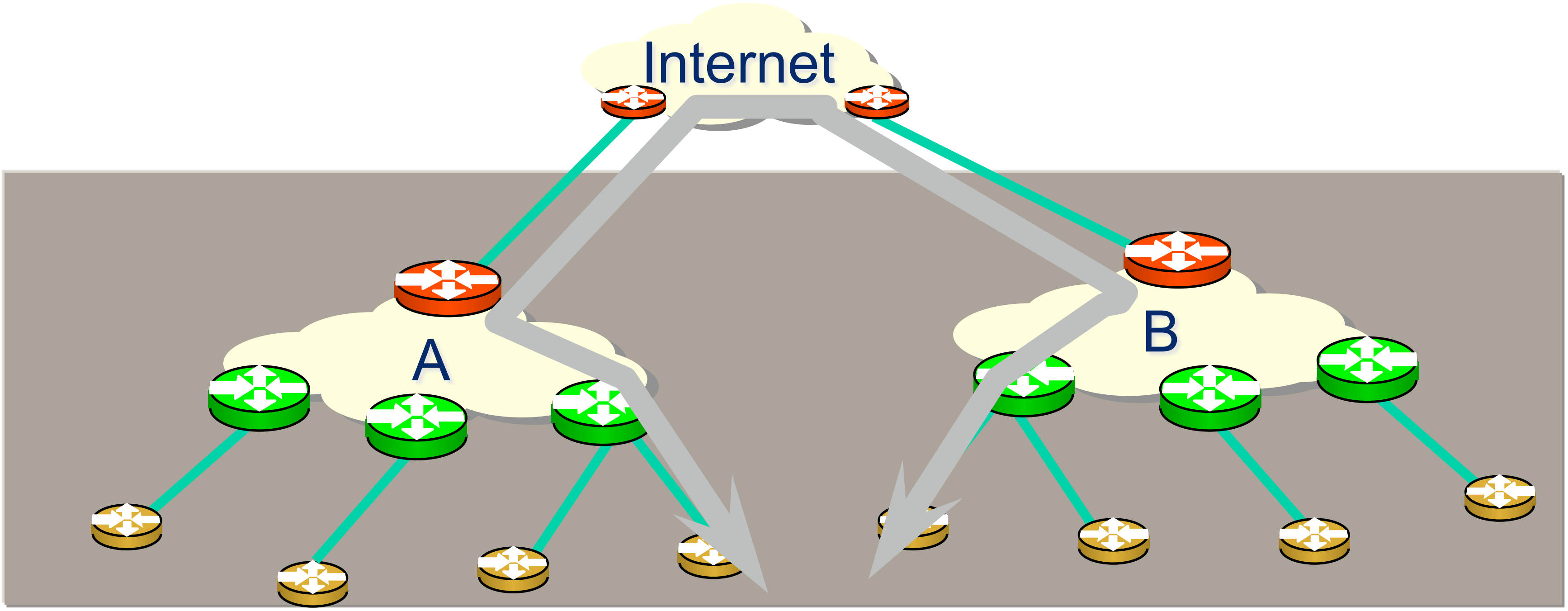
Internet Exchange Point
Why peer?

- Multiple service providers
- Each with Internet connectivity



Why IXPs?

- Is not cost effective
- Backhaul issue causes cost to both parties

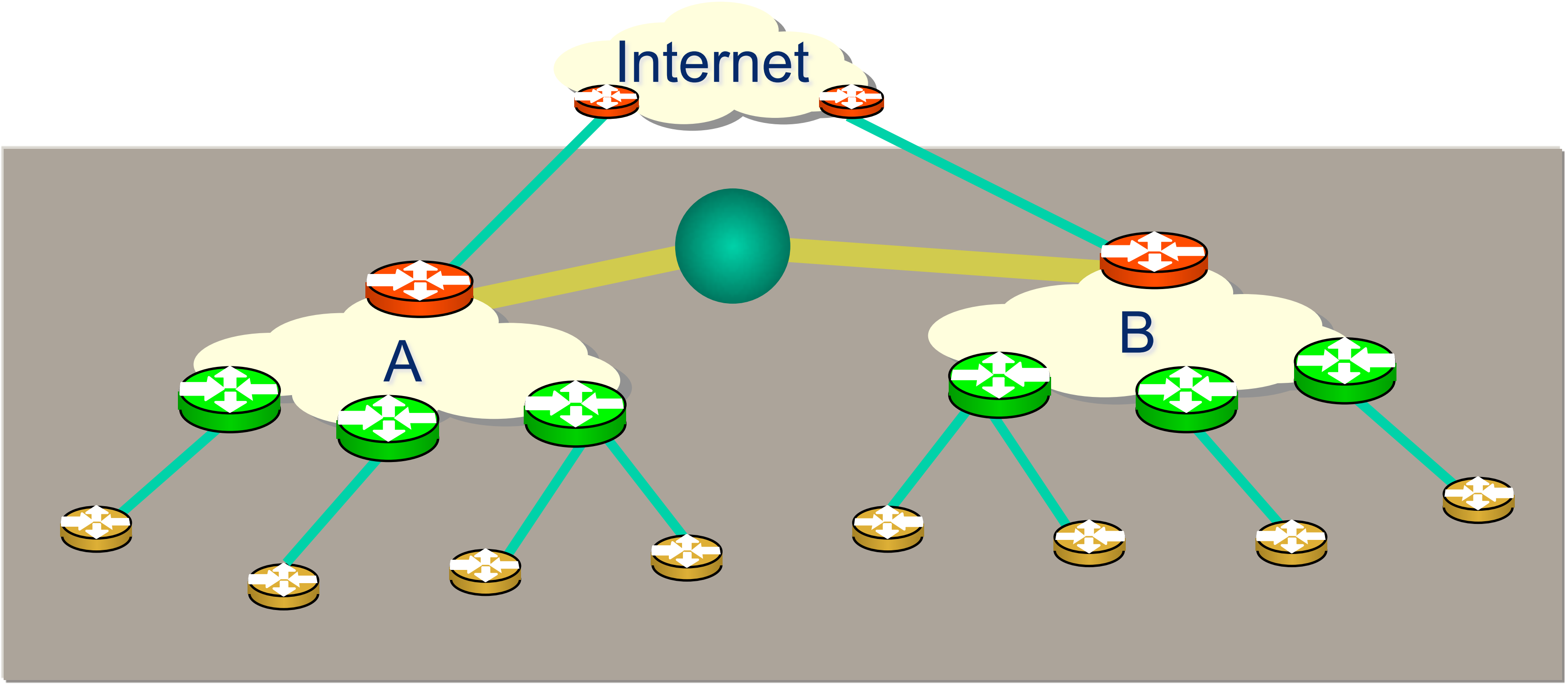


Internet Exchange Point
Why peer?

- Solution:
  - Two competing ISPs peer with each other
- Result:
  - Both save money
  - Local traffic stays local
  - Better network performance, better QoS,…
  - More international bandwidth for expensive international traffic
  - Everyone is happy

Why IXPs?

- Domestic Interconnection

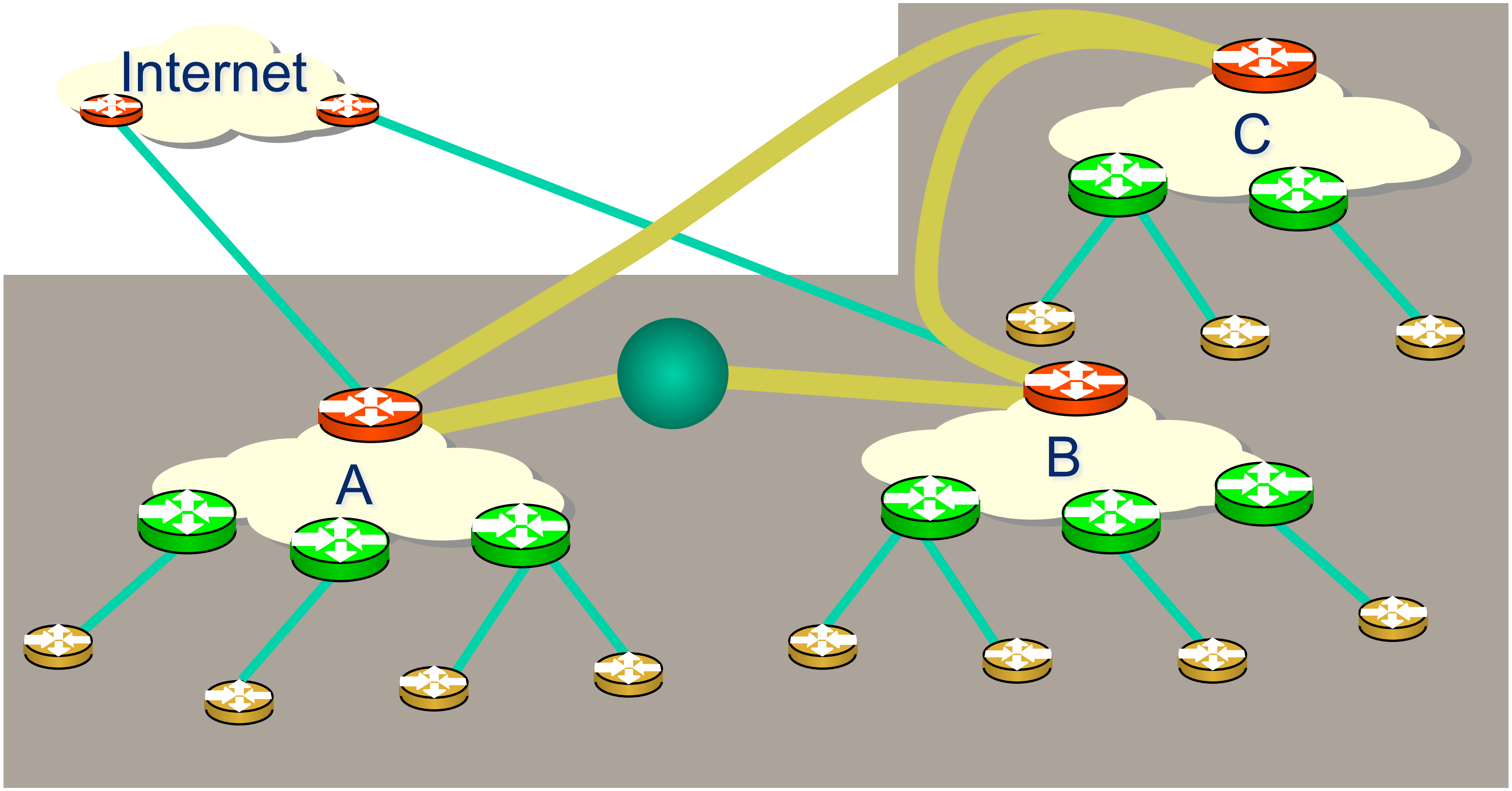


Internet Exchange Point
Why peer?

- A third ISP enters the equation
  - Becomes a significant player in the region
  - Local and international traffic goes over their international connections
- They agree to peer with the two other ISPs
  - To save money
  - To keep local traffic local
  - To improve network performance, QoS,…

Why IXPs?

- A third ISP enters the equation



Internet Exchange Point
Why peer?

- Peering means that the three ISPs have to buy circuits between each other
  - Works for three ISPs, but adding a fourth or a fifth means this does not scale
- Solution:
  - Internet Exchange Point

Internet Exchange Point

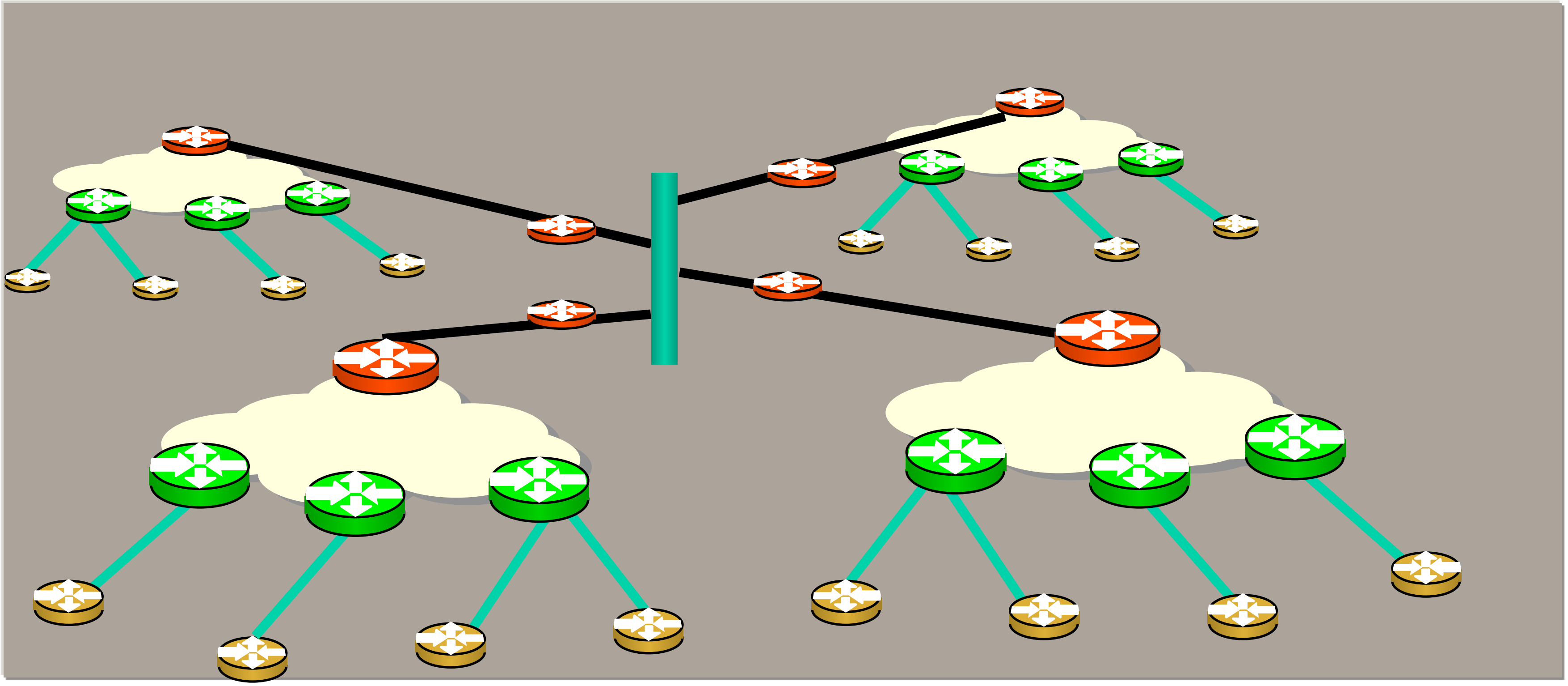
- Every participant has to buy just one whole circuit
  - From their premises to the IXP
- Rather than N-1 half circuits to connect to the N-1 other ISPs
  - 5 ISPs have to buy 4 half circuits = 2 whole circuits → already twice the cost of the IXP connection

Internet Exchange Point

- Solution
  - Every ISP participates in the IXP
  - Cost is minimal – one local circuit covers all domestic traffic
  - International circuits are used for just international traffic – and backing up domestic links in case the IXP fails
- Result:
  - Local traffic stays local
  - QoS considerations for local traffic is not an issue
  - RTTs are typically sub 10ms
  - Customers enjoy the Internet experience
  - Local Internet economy grows rapidly

Internet Exchange Point

- Ethernet switch in the middle



Why use an IXP?

- PEERING
  - Shared medium vs. point-to-point
  - Shared
    - can exchange traffic with multiple peers at one location via one interface
  - Point-to-Point
    - for high volumes of traffic

Why use an IXP?

- KEEP LOCAL TRAFFIC LOCAL!!!
  - ISPs within a region peer with each other at the local exchange
  - No need to have traffic go overseas only to come back
  - Much reduced latency and increased performance

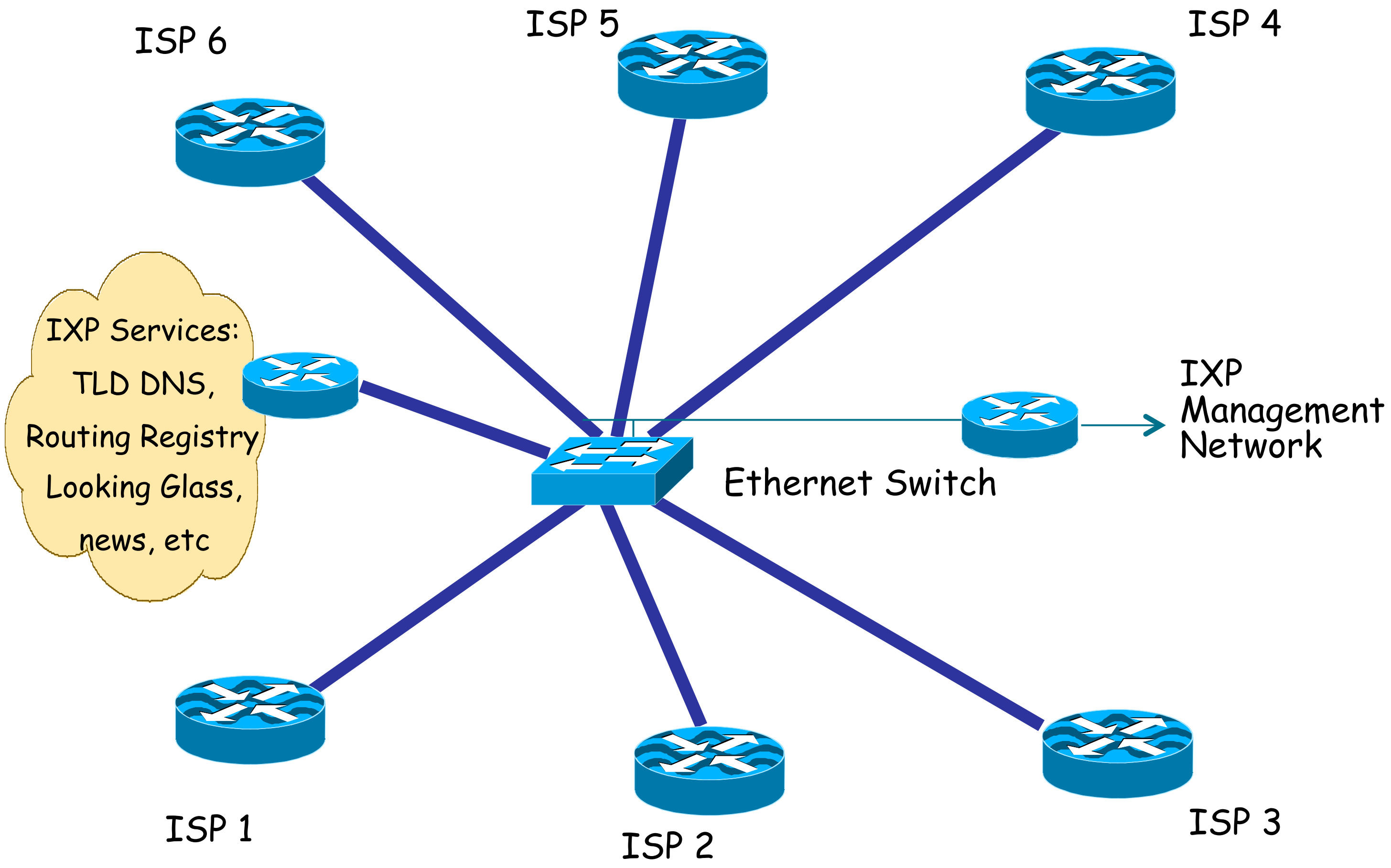
Why use an IXP?

- SAVES MONEY!!!
  - Traffic going overseas means transit charges paid to your upstream ISP
  - Money stays in local economy
    - Used to provide better local infrastructure and services for customers
  - Customers pay less for Internet access
    - Therefore more customers sign up
    - ISP has more customers, better business

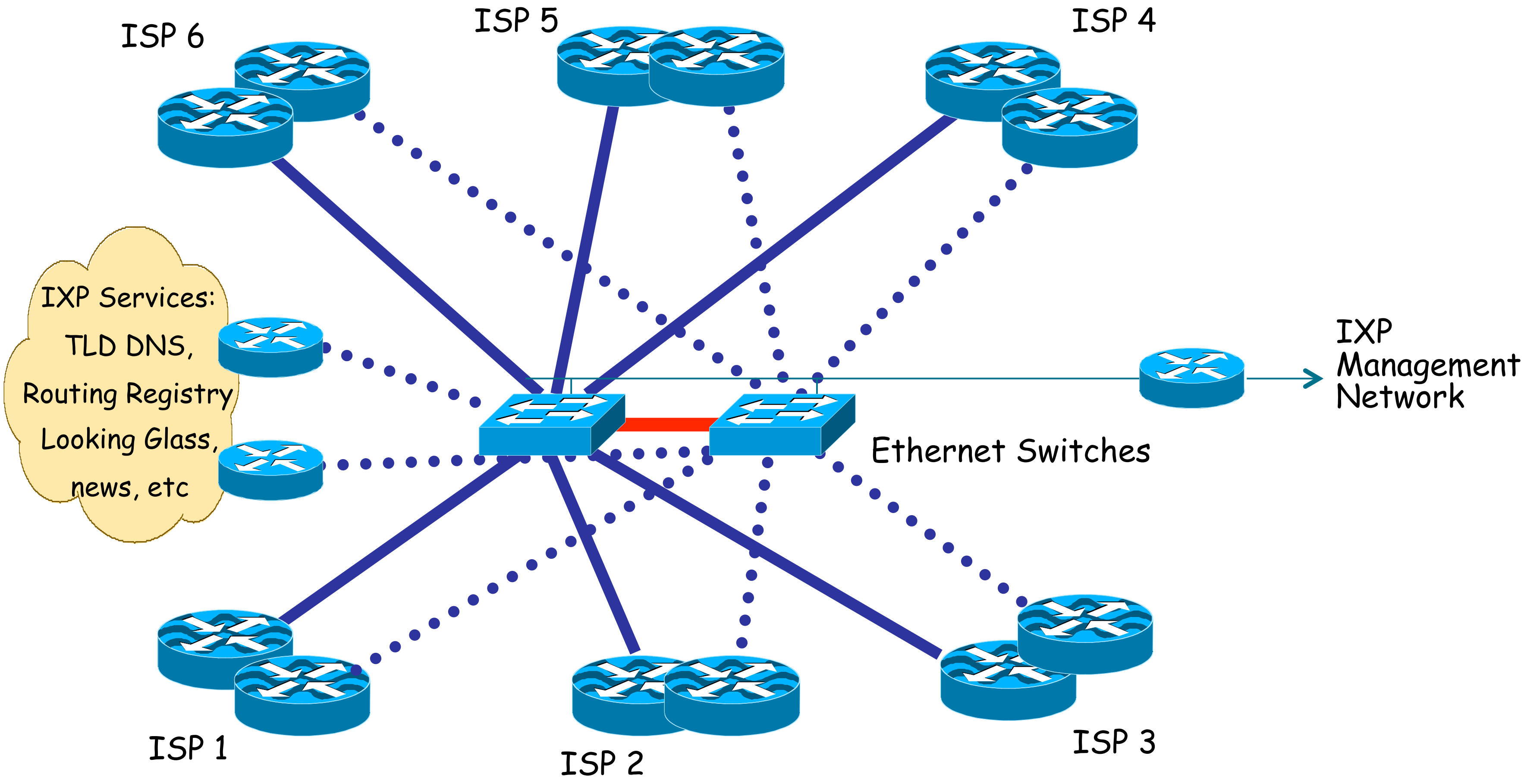
Why use an IXP?

- VASTLY IMPROVES PERFORMANCE!!!
  - Network RTTs between organisations in the local economy is measured in milliseconds, not seconds
  - Packet loss becomes virtually non-existent
  - Customers use the Internet for more products, services, and activities

Exchange Point Design



Exchange Point Design



Peering at an IXP

- Each participant needs to run BGP
  - They need their own AS number
- Each participant configures external BGP with the other participants in the IXP
  - Peering with all participants

    or
  - Peering with a subset of participants

IP Address Space

- Some IXPs use private addresses for the IXP LAN
  - Public address space means the IXP network can be leaked to the Internet, which could be undesirable
  - Filtering RFC1918 address space by ISPs is Best Practice; this avoids leakage
- Some IXPs use public addresses for the IXP LAN
  - Address space is available from the RIRs for IXPs
  - IXP terms of participation usually forbid carrying the IXP LAN addressing in the ISP backbone

Exchange Point examples

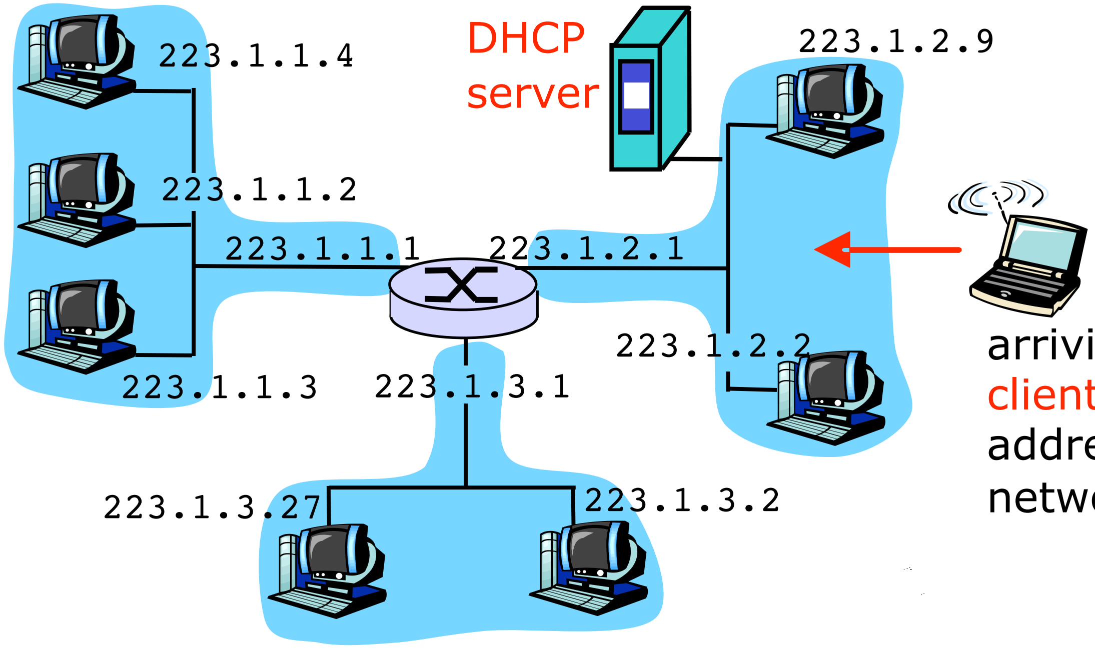
- LINX in London, UK
  - Ethernet switches
- AMS-IX in Amsterdam, NL
  - Ethernet switches
- SIX in Seattle, US
  - Ethernet switches
- JPNAP in Tokyo, Japan
  - Ethernet switches

DHCP

DHCP: Dynamic Host Configuration Protocol

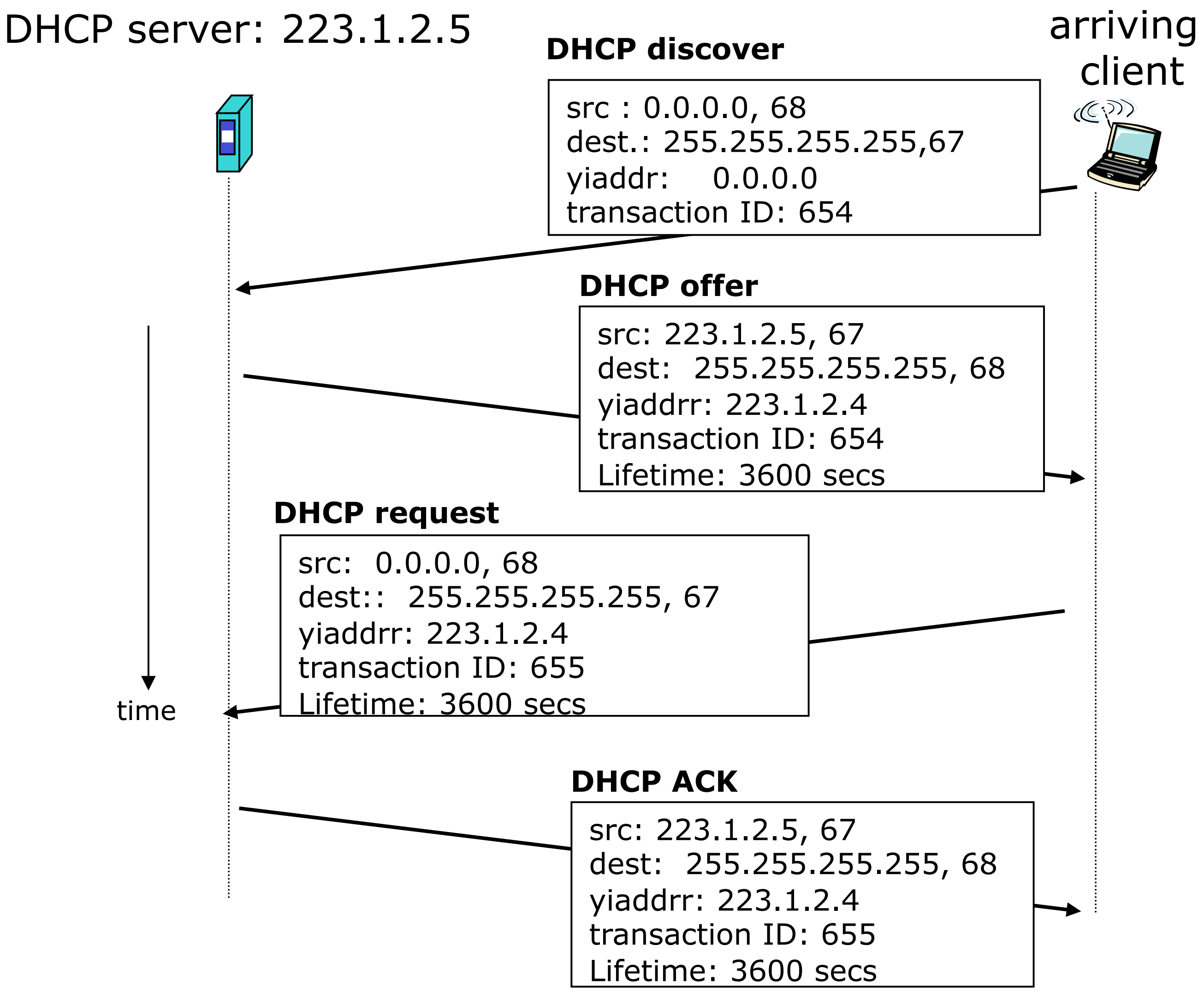
- Goal: allow host to dynamically obtain its IP address from network server when it joins network
  - Can renew its lease on address in use
  - Allows reuse of addresses (only hold address while connected an “on”
  - Support for mobile users who want to join network (more shortly)
- DHCP overview:
  - host broadcasts “DHCP discover” msg
  - DHCP server responds with “DHCP offer” msg
  - host requests IP address: “DHCP request” msg
  - DHCP server sends address: “DHCP ack” msg

DHCP client-server scenario



arriving DHCP client needs address in this network

DHCP client-server scenario



DHCP server: 223.1.2.5

arriving
client

DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
Lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

Domain Name System

(DNS)

Computers use IP addresses. Why do we need names?

- Names are easier for people to remember

- Computers may be moved between networks, in which case their IP address will change.

The old solution: HOSTS.TXT

- A centrally-maintained file, distributed to all hosts on the Internet

```
SPARKY                    128.4.13.9
UCB-MAILGATE              4.98.133.7
FTPHOST                   200.10.194.33
... etc
```

- This feature still exists:
  - /etc/hosts (UNIX)
  - c:\windows\hosts

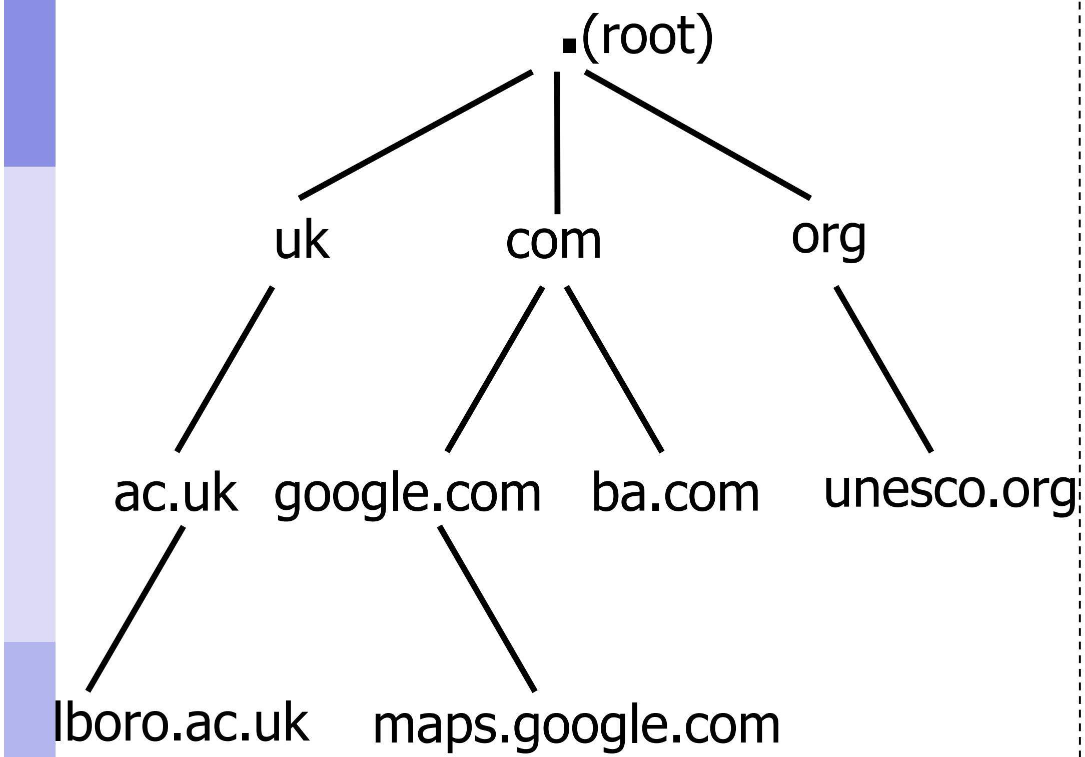
hosts.txt does not scale

- ✗ Huge file (traffic and load)
- ✗ Name collisions (name uniqueness)
- ✗ Consistency
- ✗ Always out of date
- ✗ Single point of Administration
- ✗ Did not scale well

The Domain Name System was born

- DNS is a distributed database for holding name to IP address (and other) information
- Distributed:
  - Shares the Administration
  - Shares the Load
- Robustness and improved performance achieved through
  - replication
  - and caching
- Employs a client-server architecture
- A critical piece of the Internet's infrastructure

DNS is Hierarchical



.(root)

uk

com

org

ac.uk

google.com

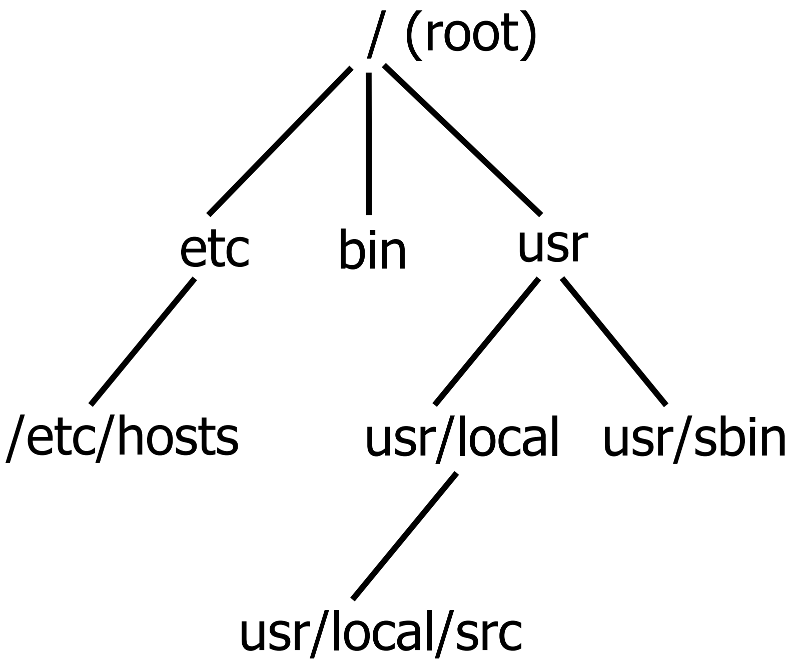
ba.com

unesco.org

lboro.ac.uk

maps.google.com

DNS Database



/ (root)

etc

bin

usr

/etc/hosts

usr/local

usr/sbin

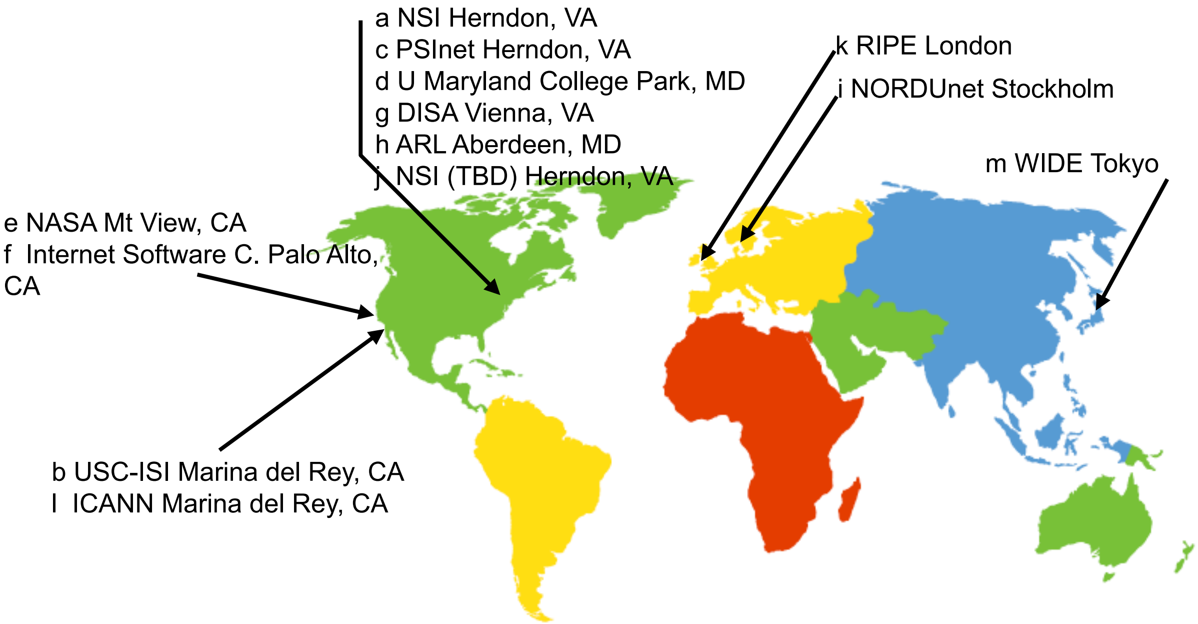
usr/local/src

Unix Filesystem

Forms a tree structure

DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - returns mapping to top-level name server



a NSI Herndon, VA
c PSInet Herndon, VA
d U Maryland College Park, MD
g DISA Vienna, VA
h ARL Aberdeen, MD
j NSI (TBD) Herndon, VA

k RIPE London

i NORDUnet Stockholm

m WIDE Tokyo

e NASA Mt View, CA
f Internet Software C. Palo Alto, CA

b USC-ISI Marina del Rey, CA
l ICANN Marina del Rey, CA

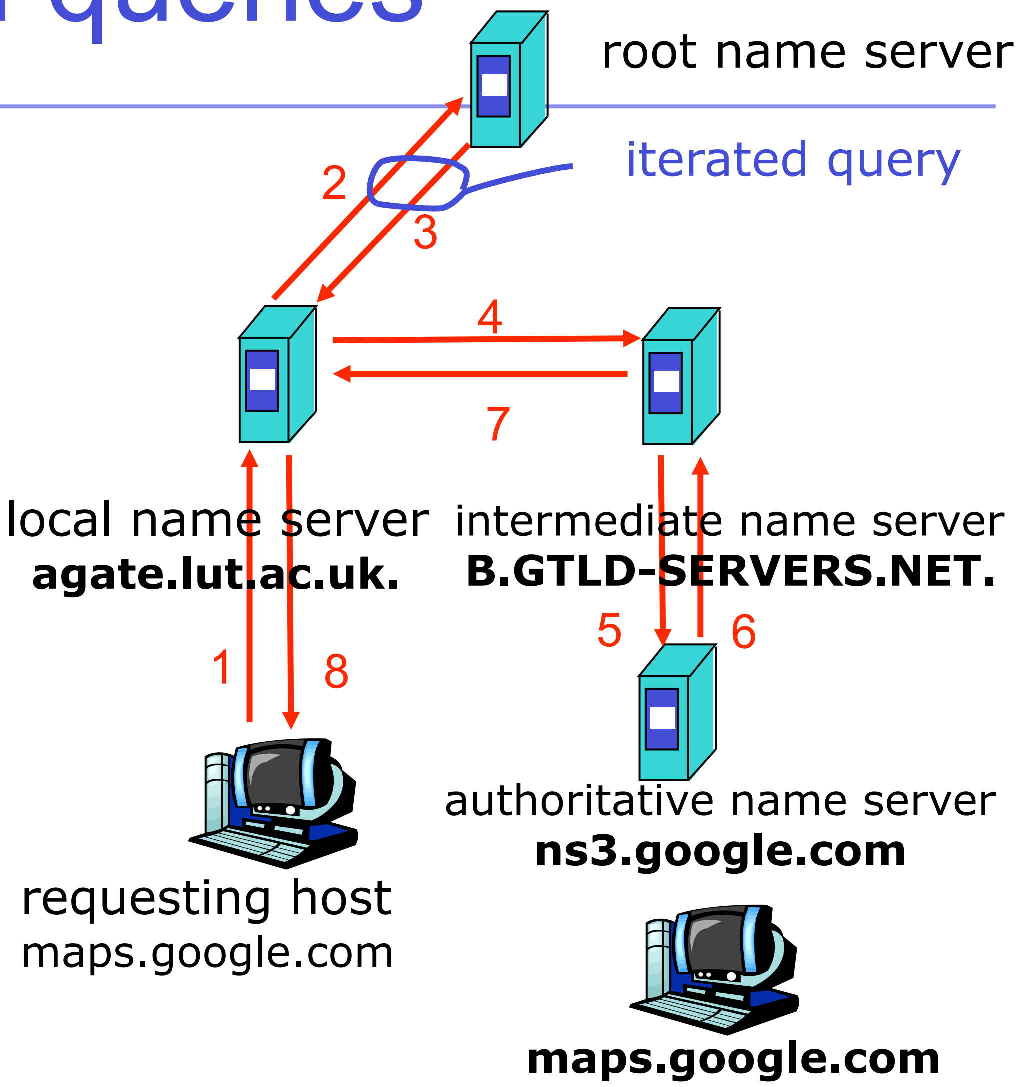
DNS: iterated queries

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?

iterated query:

- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”



DNS is Hierarchical (contd.)

- Globally unique names
- Administered in zones (parts of the tree)
- You can give away ("delegate") control of part of the tree underneath you
- Example:
  - google.com on one set of nameservers
  - maps.google.com on a different set
  - www.maps.google.com on another set

Domain Names are (almost) unlimited

- Max 255 characters total length
- Max 63 characters in each part
  - RFC 1034, RFC 1035
- If a domain name is being used as a host name, you should abide by some restrictions
  - RFC 952 (old!)
  - a-z 0-9 and minus (-) only
  - No underscores (_)

Using the DNS

- A Domain Name (like www.lut.ac.uk) is the KEY to look up information
- The result is one or more RESOURCE RECORDS (RRs)
- There are different RRs for different types of information
- You can ask for the specific type you want, or ask for "any" RRs associated with the domain name

Commonly seen Resource Records (RRs)

- A (address): map hostname to IPv4 address
- AAAA (quad A): map a hostname to IPv6 address
- PTR (pointer): map IP address to hostname
- MX (mail exchanger): where to deliver mail for *user@domain*
- CNAME (canonical name): map alternative hostname to real hostname
- TXT (text): any descriptive text
- NS (name server), SOA (start of authority): used for delegation and management of the DNS itself

A Simple Example

- Query: **www.lut.ac.uk.**
- Query type: **A**
- Result:

```
www.lut.ac.uk.    22725  IN   A  158.125.1.208
```

- **In this case a single RR is found, but in general, multiple RRs may be returned.**
  - (IN is the "class" for INTERNET use of the DNS)

Possible results from a Query

- POSITIVE
  - one or more RRs found
- NEGATIVE
  - definitely no RRs match the query
- SERVER FAIL
  - cannot find the answer
- REFUSED
  - not allowed to query the server

How do you use an IP address as the key for a DNS query

- Convert the IP address to dotted-quad
- Reverse the four parts
- Add ".in-addr.arpa." to the end; special domain reserved for this purpose

`e.g. to find name for 158.125.1.208`

`Domain name: 208.1.125.158.in-addr.arpa.`

`Query Type: PTR`

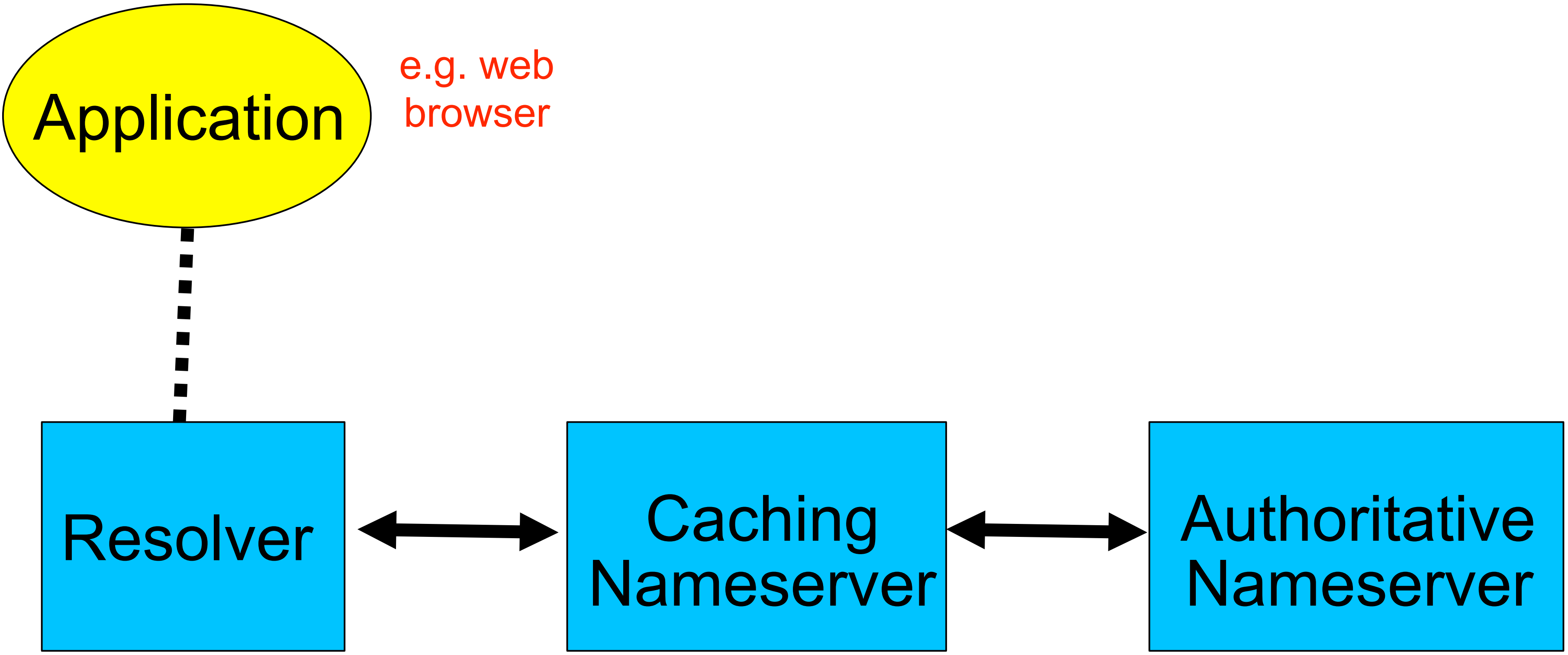
`Result: www.lut.ac.uk.`

Known as a "reverse DNS lookup" (because we are looking up the name for an IP address, rather than the IP address for a name)

DNS is a Client-Server application

- (Of course - it runs across a network)
- Requests and responses are normally sent in UDP packets, port 53
- Occasionally uses TCP, port 53
  - for very large requests (larger than 512-bytes) e.g. zone transfer from master to slave or an IPv6 AAAA (quad A) record.

There are three roles involved in DNS



Three roles in DNS

- RESOLVER
  - Takes request from application, formats it into UDP packet, sends to cache
- CACHING NAMESERVER
  - Returns the answer if already known
  - Otherwise searches for an authoritative server which has the information
  - Caches the result for future queries
  - Also known as RECURSIVE nameserver
- AUTHORITATIVE NAMESERVER
  - Contains the actual information put into the DNS by the domain owner

Three roles in DNS

- The SAME protocol is used for resolver <-> cache and cache <-> auth NS communication
- It is possible to configure a single name server as both caching and authoritative
- But it still performs only one role for each incoming query
- Common but NOT RECOMMENDED to configure in this way (we will see why later).

ROLE 1: THE RESOLVER

- A piece of software which formats a DNS request into a UDP packet, sends it to a cache, and decodes the answer

- Usually a shared library (e.g. libresolv.so under Unix) because so many applications need it

- EVERY host needs a resolver - e.g. every Windows workstation has one

How does the resolver find a caching nameserver?

- It has to be explicitly configured (statically, or via DHCP etc)
- Must be configured with the IP ADDRESS of a cache (why not name?)
- Good idea to configure more than one cache, in case the first one fails

How do you choose which cache(s) to configure?

- Must have PERMISSION to use it
  - e.g. cache at your ISP, or your own
- Prefer a nearby cache
  - Minimises round-trip time and packet loss
  - Can reduce traffic on your external link, since often the cache can answer without contacting other servers
- Prefer a reliable cache
  - Perhaps your own?

Resolver can be configured with default domain(s)

- If "foo.bar" fails, then retry query as "foo.bar.mydomain.com"
- Can save typing but adds confusion
- May generate extra unnecessary traffic
- Usually best avoided

Example: Unix resolver configuration

/etc/resolv.conf

```
domain lboro.ac.uk
nameserver 158.125.1.100
nameserver 131.231.16.7
```

That's all you need to configure a resolver

Testing DNS with "dig"

- "dig" is a program which just makes DNS queries and displays the results
- Better than "nslookup", "host" because it shows the raw information in full

```
dig lboro.ac.uk.
  -- defaults to query type "A"
dig lboro.ac.uk. mx
  -- specified query type
dig @8.8.8.8 lboro.ac.uk. mx
  -- send to particular cache (overrides
     /etc/resolv.conf)
```

The trailing dot

```
# dig lboro.ac.uk.
```

- Prevents any default domain being appended
- Get into the habit of using it always when testing DNS
  - only on domain names, not IP addresses or e-mail addresses

```
dig lboro.ac.uk.
```

```
; <<>> DiG 9.6.0-APPLE-P2 <<>> lboro.ac.uk.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26566
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 7

;; QUESTION SECTION:
;lboro.ac.uk.                               IN         A

;; ANSWER SECTION:
lboro.ac.uk.                     86400      IN         A          158.125.1.208

;; AUTHORITY SECTION:
lboro.ac.uk.                     86400      IN         NS         cgate.lut.ac.uk.
lboro.ac.uk.                     86400      IN         NS         agate.lut.ac.uk.
lboro.ac.uk.                     86400      IN         NS         bgate.lut.ac.uk.
lboro.ac.uk.                     86400      IN         NS         ns3.ja.net.

;; ADDITIONAL SECTION:
agate.lut.ac.uk.                 86400      IN         A          158.125.1.100
bgate.lut.ac.uk.                 86400      IN         A          131.231.16.7
bgate.lut.ac.uk.                 86400      IN         AAAA       2001:630:301:4605::b53
cgate.lut.ac.uk.                 86400      IN         A          131.231.16.16
cgate.lut.ac.uk.                 86400      IN         AAAA       2001:630:301:2905::c53
ns3.ja.net.                      21325      IN         A          193.63.106.103
ns3.ja.net.                      39250      IN         AAAA       2001:630:0:46::67

;; Query time: 30 msec
;; SERVER: 158.125.1.100#53(158.125.1.100)
;; WHEN: Sun Dec  6 15:51:52 2009
;; MSG SIZE  rcvd: 281
```

Understanding output from dig

- STATUS
  - NOERROR: 0 or more RRs returned
  - NXDOMAIN: non-existent domain
  - SERVFAIL: cache could not locate answer
  - REFUSED: query not available on cache server
- FLAGS
  - AA: Authoritative answer (not from cache)
  - You can ignore the others
    - QR: Query/Response (1 = Response)
    - RD: Recursion Desired
    - RA: Recursion Available
- ANSWER: number of RRs in answer

Understanding output from dig

- Answer section (RRs requested)
  - Each record has a Time To Live (TTL)
  - Says how long the cache will keep it
- Authority section
  - Which nameservers are authoritative for this domain
- Additional section
  - More RRs (typically IP addresses for the authoritative nameservers)
- Total query time
- Check which server gave the response!
  - If you make a typing error, the query may go to a default server

DNS records

DNS: distributed db storing resource records (RR)

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

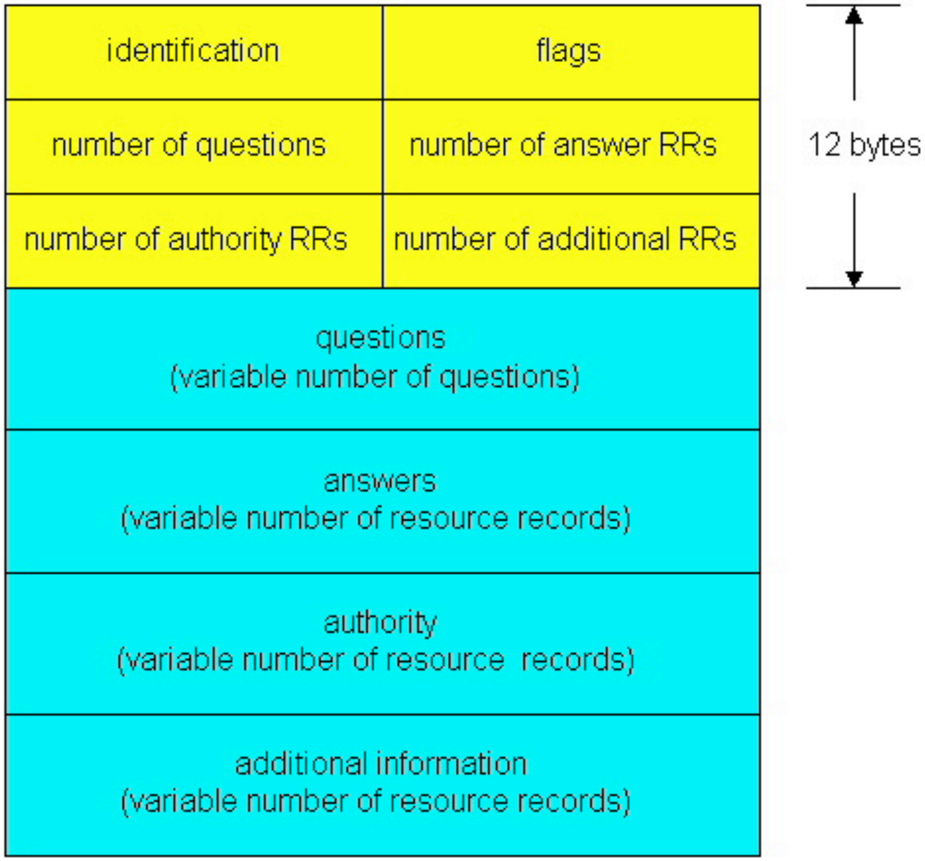
- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is IP address of authoritative name server for this domain
- Type=CNAME
  - **name** is alias name for some "cannonical" (the real) name
  - `www.lboro.ac.uk` is really
  - `www.lut.ac.uk.`
  - **value** is cannonical name
- Type=MX
  - **value** is name of mailserver associated with **name**

DNS protocol, messages

DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



identification | flags

number of questions | number of answer RRs

number of authority RRs | number of additional RRs

12 bytes

questions (variable number of questions)

answers (variable number of resource records)

authority (variable number of resource records)

additional information (variable number of resource records)

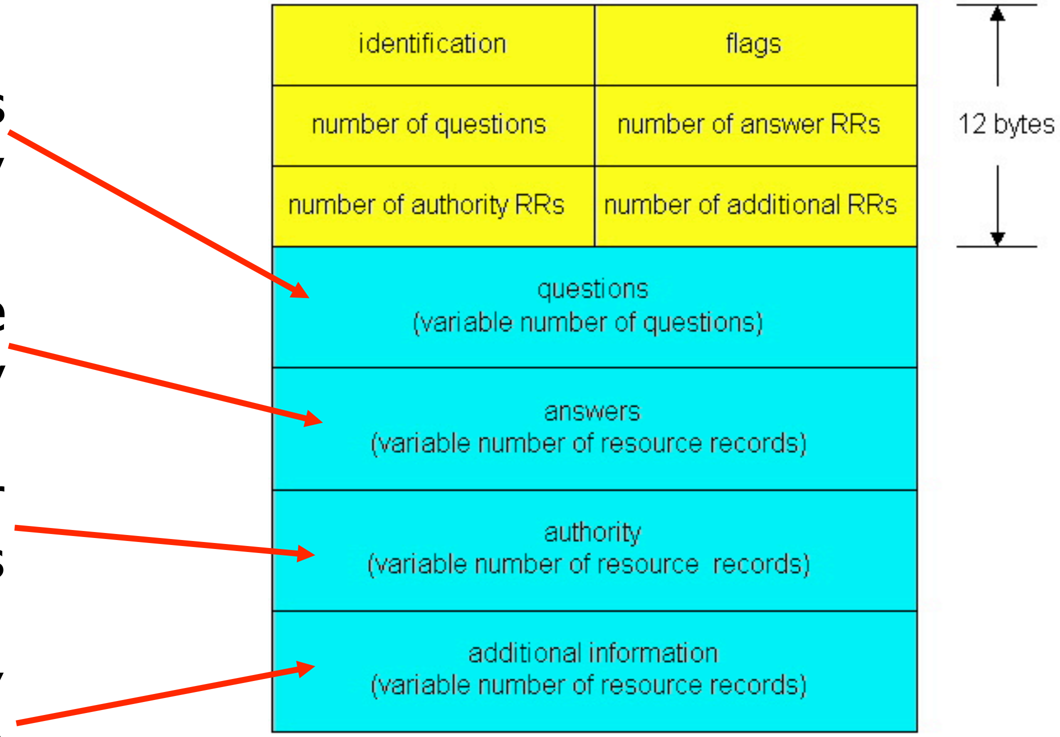
DNS protocol, messages

Name, type fields
for a query

RRs in reponse
to query

records for
authoritative servers

additional "helpful"
info that may be used



identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

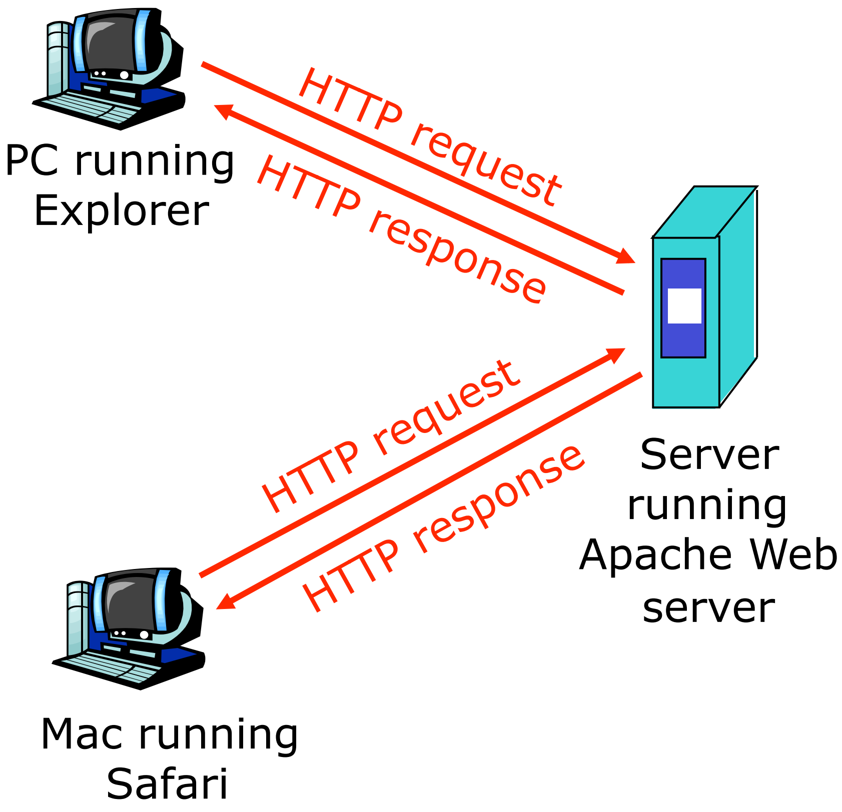
HTTP

(Hypertext Transfer Protocol)

HTTP overview

HTTP: 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 1.0: RFC 1945
- HTTP 1.1: RFC 2068



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

aside

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 connections

Nonpersistent HTTP

- At most one object is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

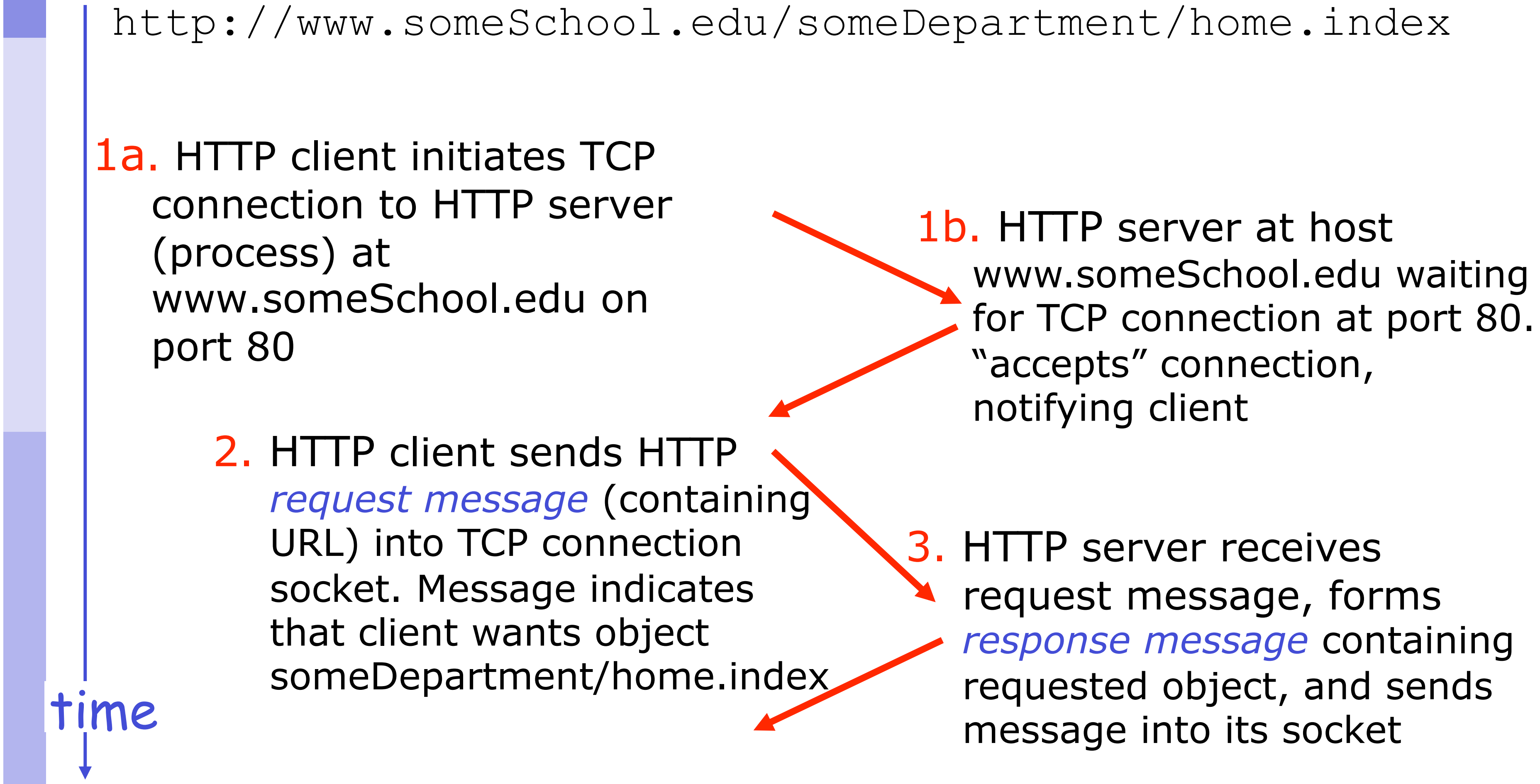
- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

Nonpersistent HTTP

Suppose user enters URL:

(contains text,
references to 10
jpeg images)

`http://www.someSchool.edu/someDepartment/home.index`



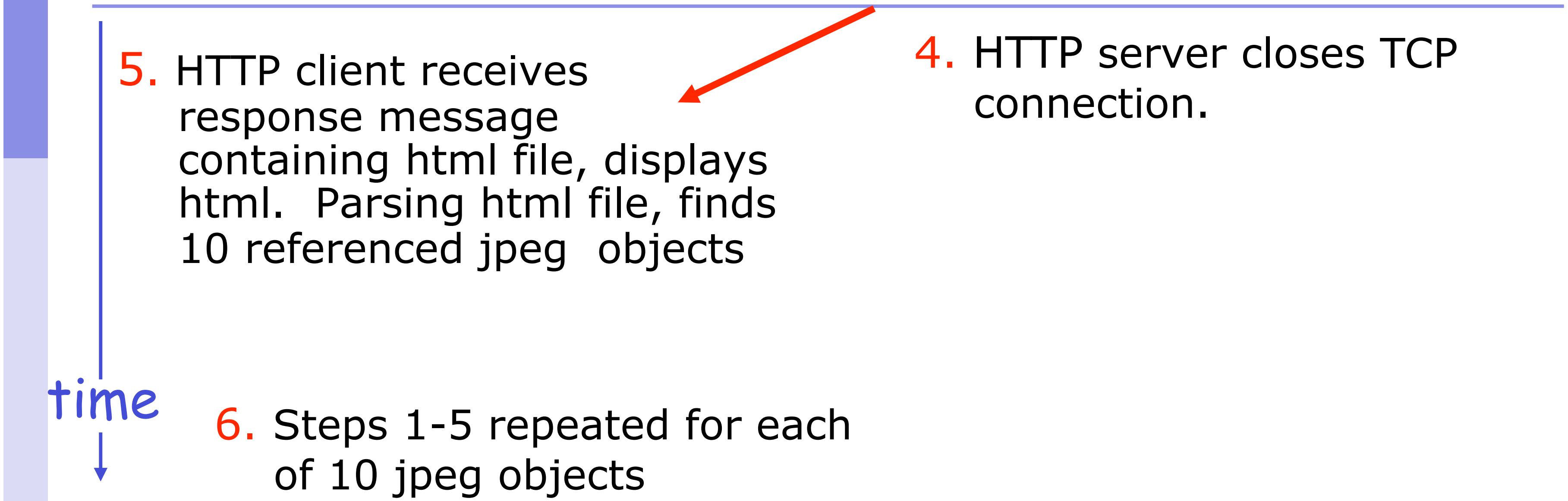
1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

Nonpersistent HTTP (cont.)



4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

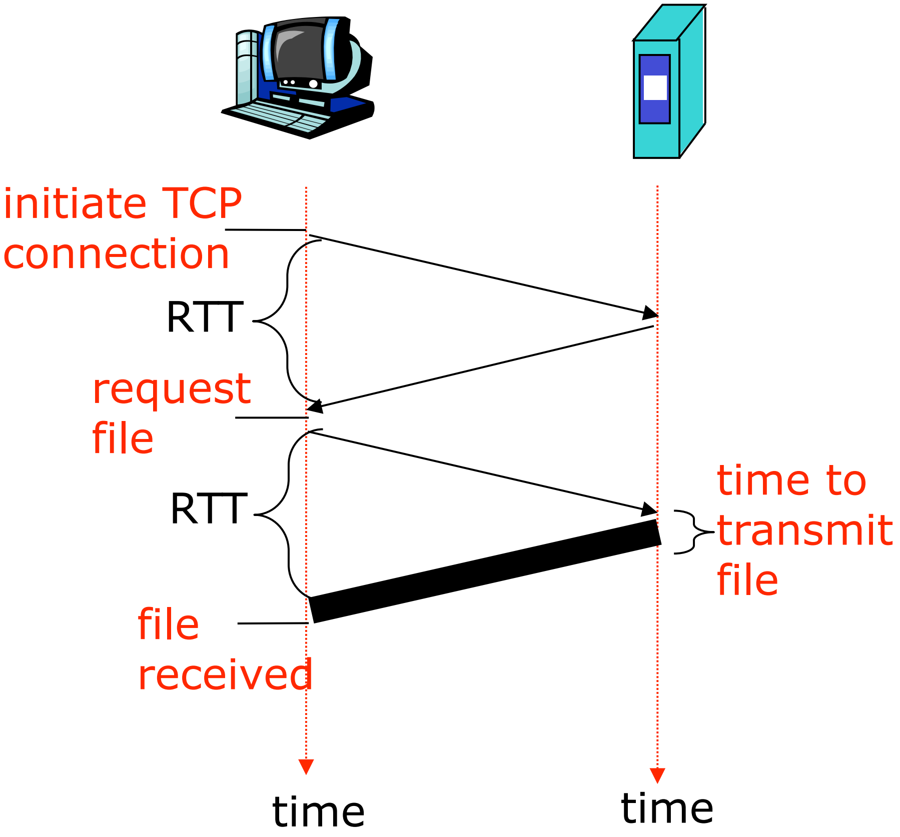
Response time modeling

Definition of RTT: time to send a small packet to travel from client to server and back.

Response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = 2RTT+transmit time



Persistent HTTP

Nonpersistent HTTP issues:

- requires 2 RTTs per object
- OS must work and allocate host resources for each TCP connection
- but browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server are sent over connection

Persistent without pipelining:

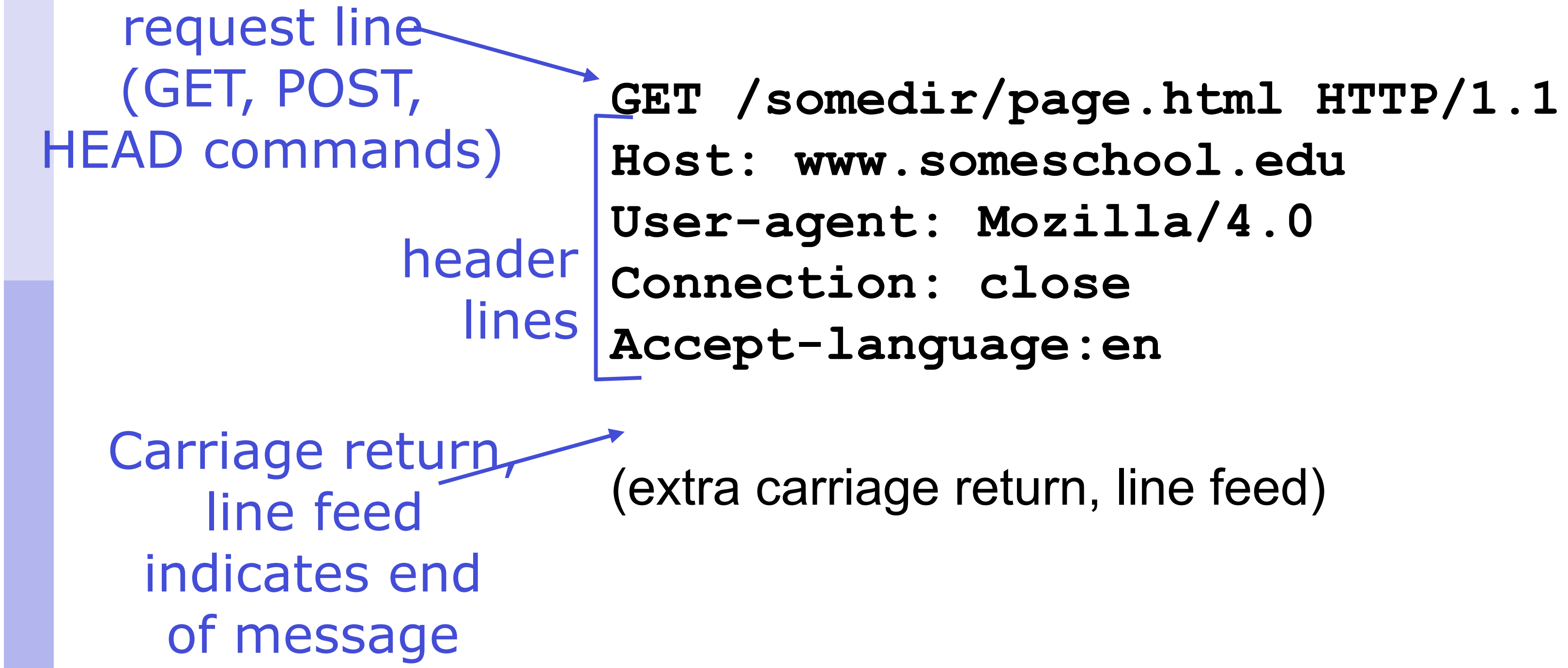
- client issues new request only when previous response has been received
- one RTT for each referenced object

Persistent with pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP request message

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



request line (GET, POST, HEAD commands)

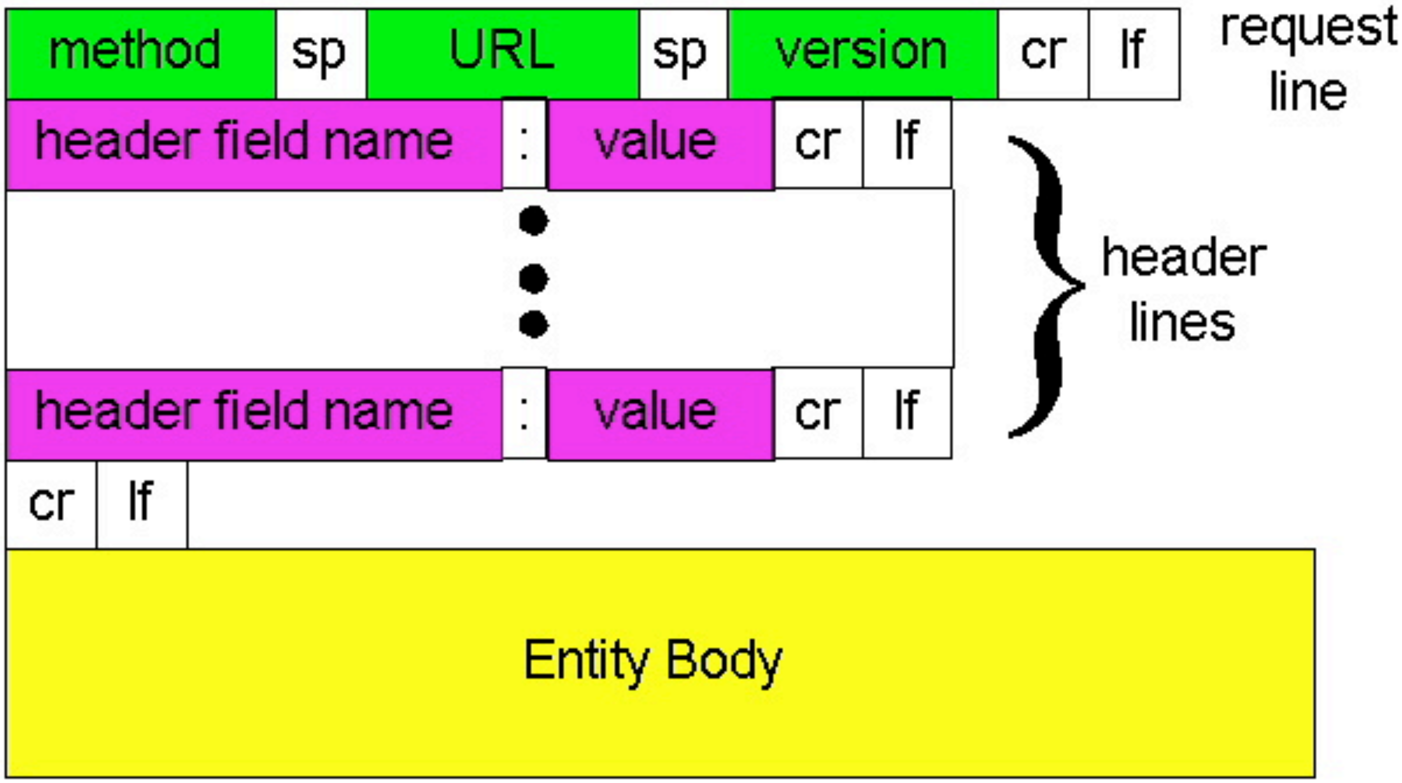
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:en
```

header lines

Carriage return, line feed indicates end of message

(extra carriage return, line feed)

HTTP request message: general format



method sp URL sp version cr lf

request line

header field name : value cr lf

•
•
•

header field name : value cr lf

header lines

cr lf

Entity Body

Uploading form input

Post method:

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

URL method:

- Uses GET method
- Input is uploaded in URL field of request line:

```
www.somesite.com/animalsearch?monkeys&banana
```

Method types

HTTP/1.0

- GET
- POST
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1

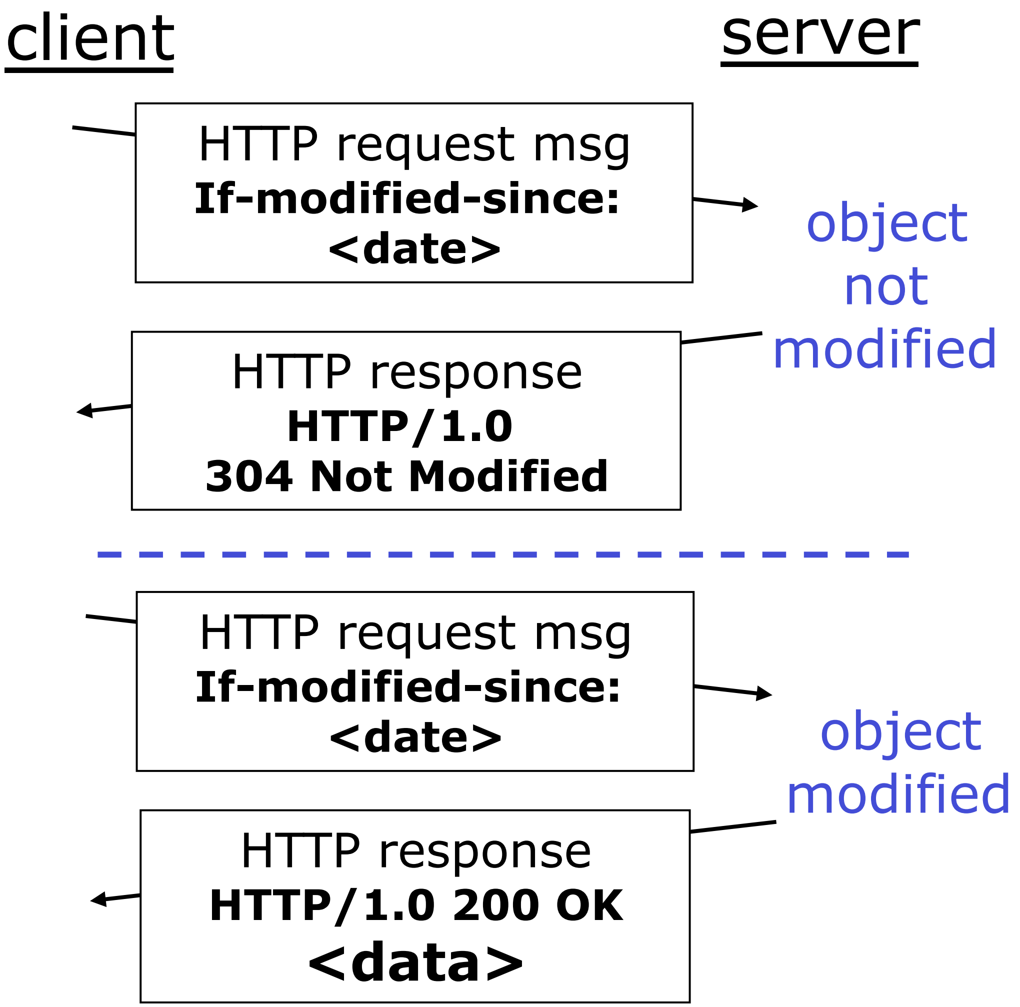
- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

Conditional GET: client-side caching

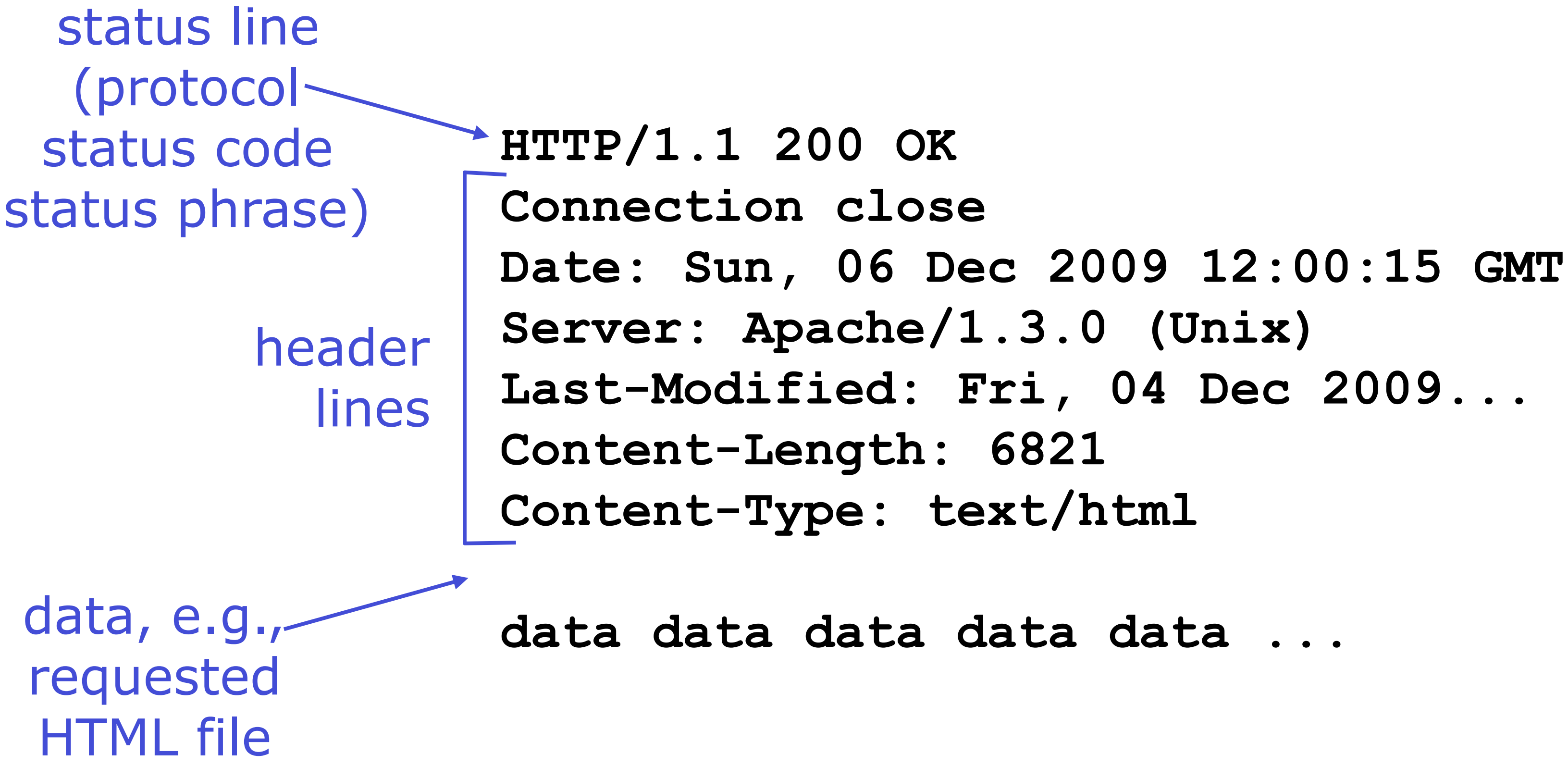
- Goal: don’t send object if client has up-to-date cached version
- client: specify date of cached copy in HTTP request

  `If-modified-since: <date>`
- server: response contains no object if cached copy is up-to-date:

  `HTTP/1.0 304 Not Modified`



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: Sun, 06 Dec 2009 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Fri, 04 Dec 2009...
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`

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet www.lboro.ac.uk 80
```

Opens TCP connection to port 80
(default HTTP server port).
Anything typed in sent
to port 80 at www.lut.ac.uk.

2. Type in a GET HTTP request:

```
GET /index.html HTTP/1.0
```

By typing this in (hit carriage
return twice), you send
this minimal (but complete)
GET request to HTTP server

3. Look at response message sent by HTTP server!

Cookies: keeping “state”

Many major Web sites use cookies

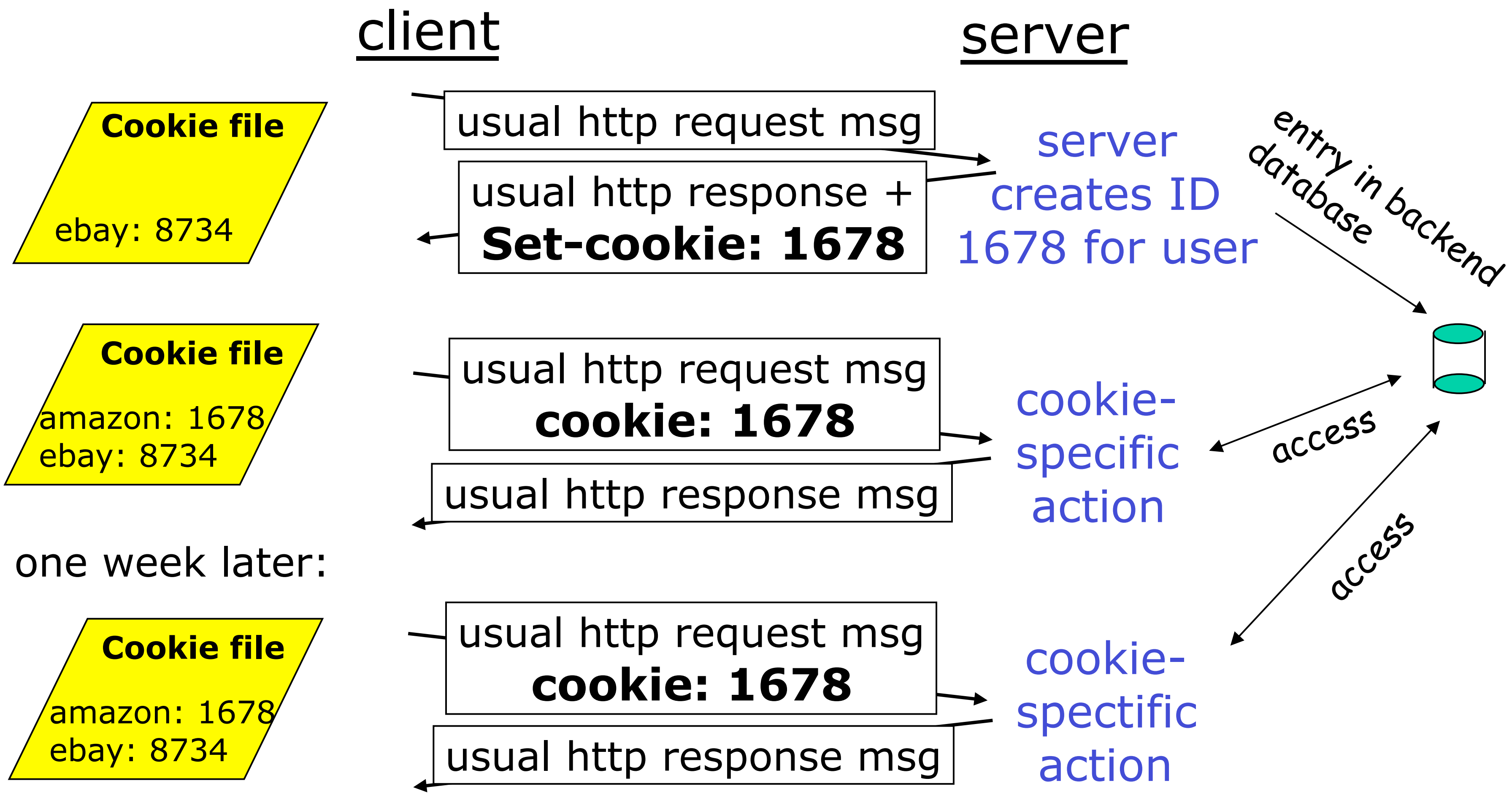
Four components:

1) cookie header line in the HTTP response message
2) cookie header line in HTTP request message
3) cookie file kept on user’s host and managed by user’s browser
4) back-end database at Web site

Example:

- Susan access Internet always from same PC
- She visits a specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates a unique ID and creates an entry in backend database for ID

Cookies: keeping “state” (cont.)



Cookies (continued)

What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

aside

Cookies and privacy:

- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites
- search engines use redirection & cookies to learn yet more
- advertising companies obtain info across sites

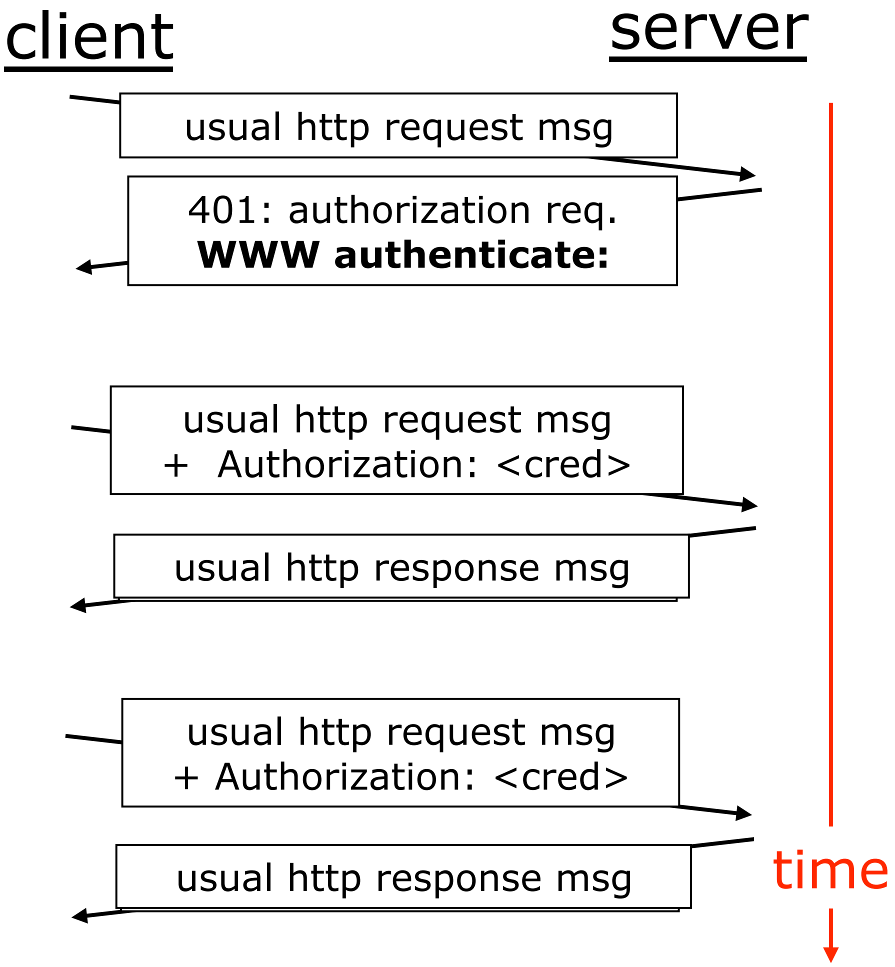
User-server interaction: authorization

Authorization: control access to server content

- authorization credentials: typically name, password
- stateless: client must present authorization in *each* request
  - authorization: header line in each request
  - if no authorization: header, server refuses access, sends

    WWW authenticate:

    header line in response



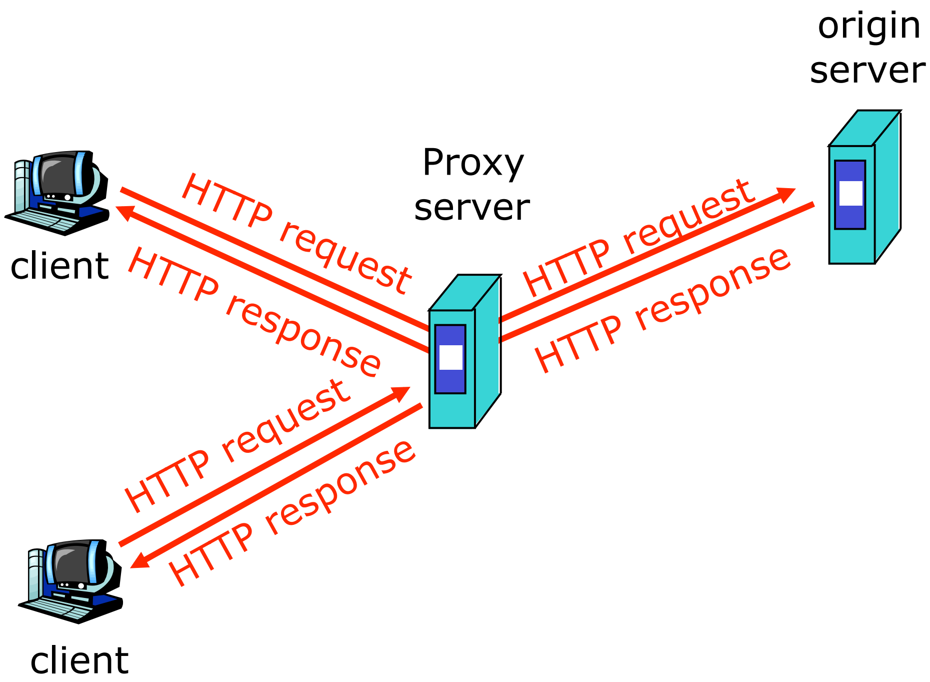
CDNs

(Content Distribution Networks)

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



More about Web caching

- Cache acts as both client and server
- Cache can do up-to-date check using `If-modified-since` HTTP header
  - Issue: should cache take risk and deliver cached object without checking?
  - Heuristics are used.
- Typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

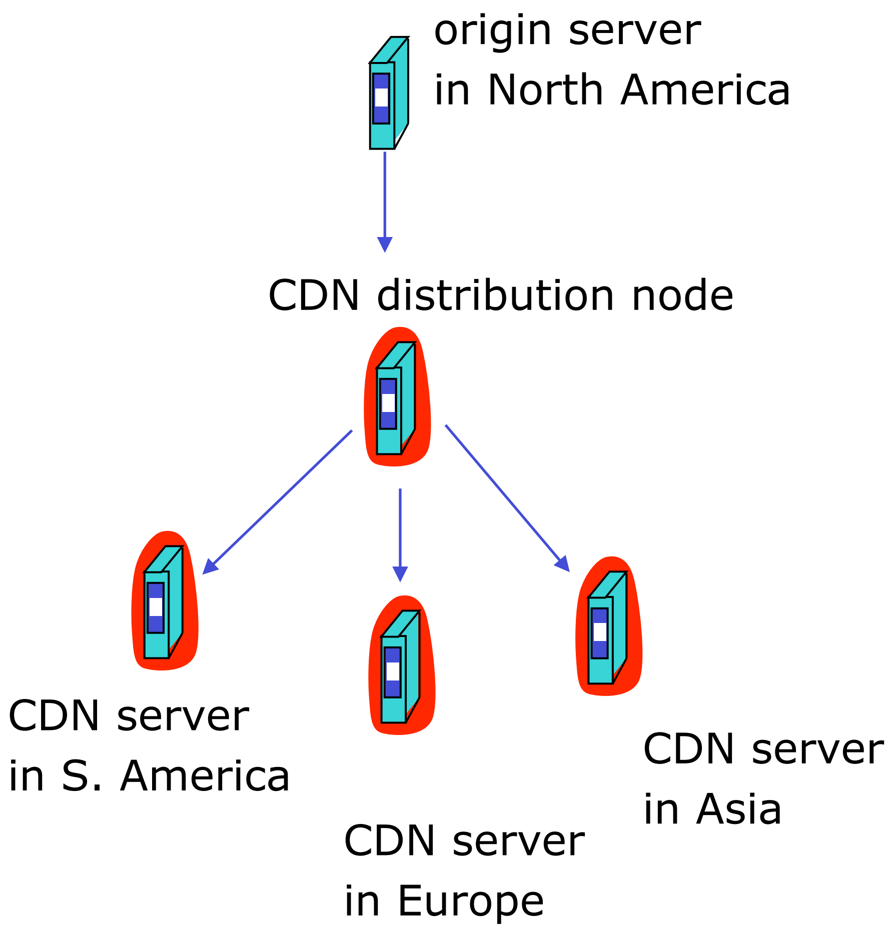
- Reduce response time for client request.
- Reduce traffic on an institution's access link.
- Internet dense with caches enables "poor" content providers to effectively deliver content

Content distribution networks (CDNs)

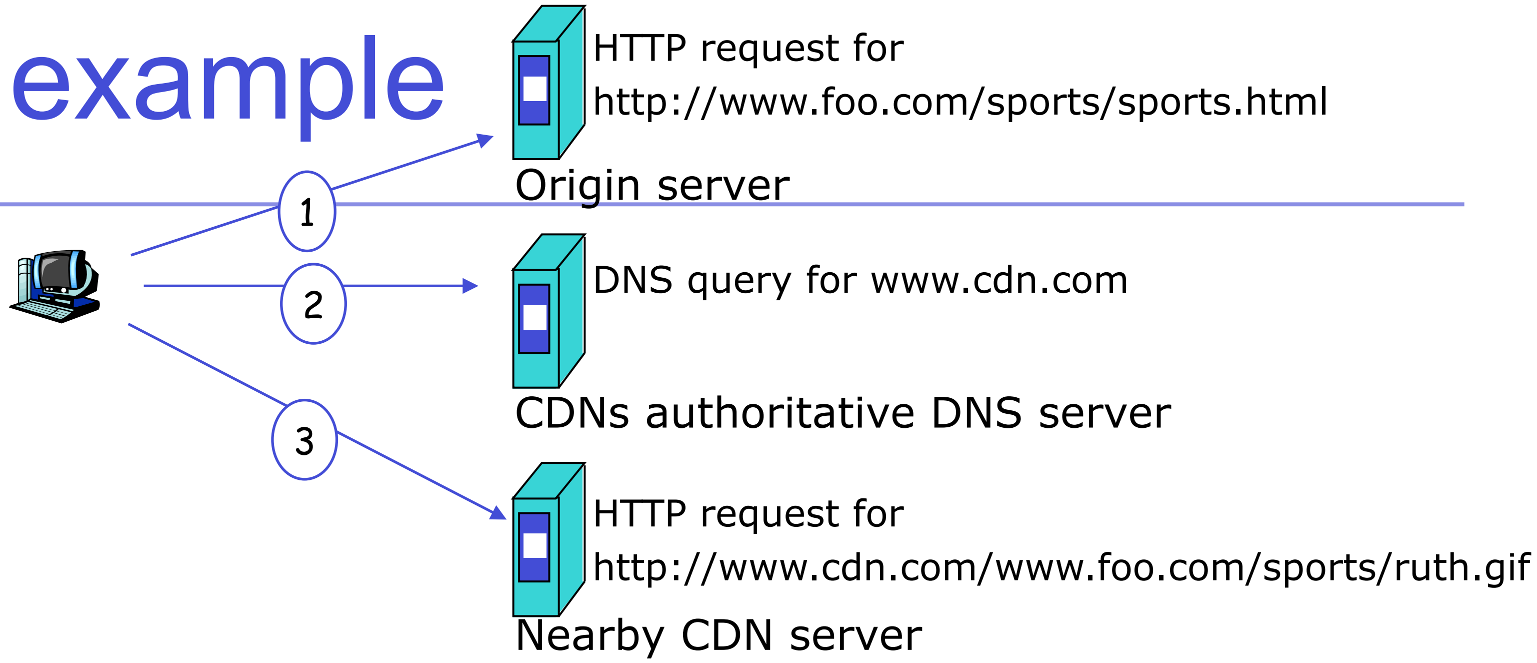
- The content providers are the CDN customers.

Content replication

- CDN company installs hundreds of CDN servers throughout Internet
  - in lower-tier ISPs, close to users
- CDN replicates its customers' content in CDN servers. When provider updates content, CDN updates servers



CDN example



1 HTTP request for http://www.foo.com/sports/sports.html

Origin server

2 DNS query for www.cdn.com

CDNs authoritative DNS server

3 HTTP request for http://www.cdn.com/www.foo.com/sports/ruth.gif

Nearby CDN server

origin server

- www.foo.com
- distributes HTML
- Replaces:

  `http://www.foo.com/sports.ruth.gif`

  with

  `http://www.cdn.com/www.foo.com/sports/ruth.gif`

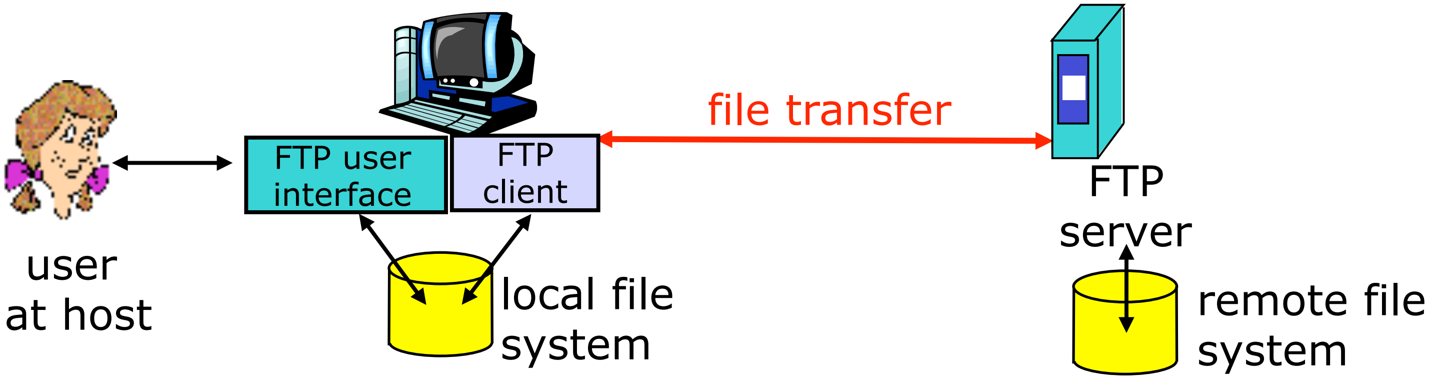
CDN company

- cdn.com
- distributes gif files
- uses its authoritative DNS server to route redirect requests

FTP

(File Transfer Protocol)

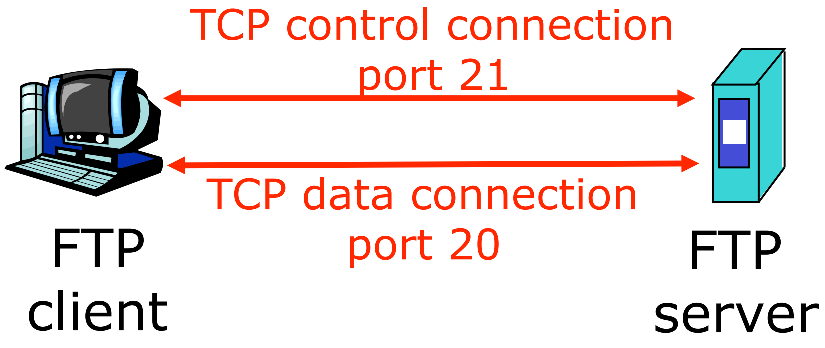
FTP: the file transfer protocol



- transfer file to/from remote host
- client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- ftp: RFC 959
- ftp server: port 21

FTP: separate control, data connections

- FTP client contacts FTP server at port 21, specifying TCP as transport protocol
- Client obtains authorization over control connection
- Client browses remote directory by sending commands over control connection.
- When server receives a command for a file transfer, the server opens a TCP data connection to client
- After transferring one file, server closes connection.



- Server opens a second TCP data connection to transfer another file.
- Control connection: "out of band"
- FTP server maintains "state": current directory, earlier authentication

FTP commands, responses

Sample commands:

- sent as ASCII text over control channel
- **USER *username***
- **PASS *password***
- **LIST** return list of file in current directory
- **RETR filename** retrieves (gets) file
- **STOR filename** stores (puts) file onto remote host

Sample return codes

- status code and phrase (as in HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can’t open data connection**
- **452 Error writing file**

E-Mail

(SMTP, POP, IMAP)

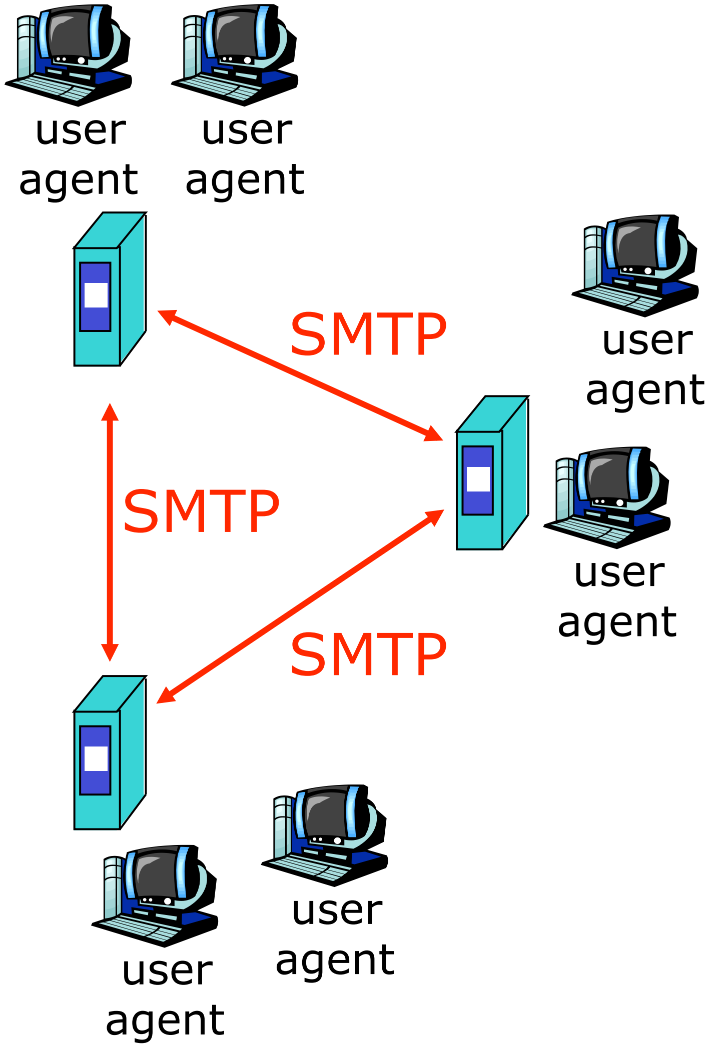
Electronic Mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

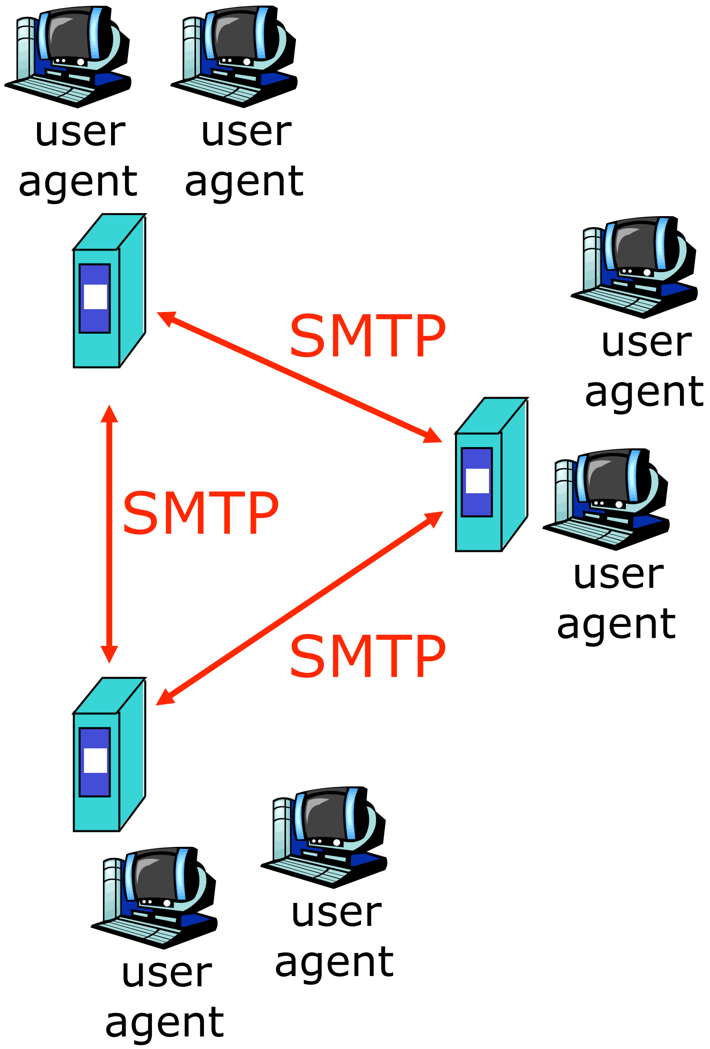
- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, mutt, Apple Mail
- outgoing, incoming messages stored on server



Electronic Mail

Mail Servers

- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages
- SMTP protocol between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server



Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

Scenario: Alice sends message to Bob

1) Alice composes a message "to" `bob@someschool.edu`

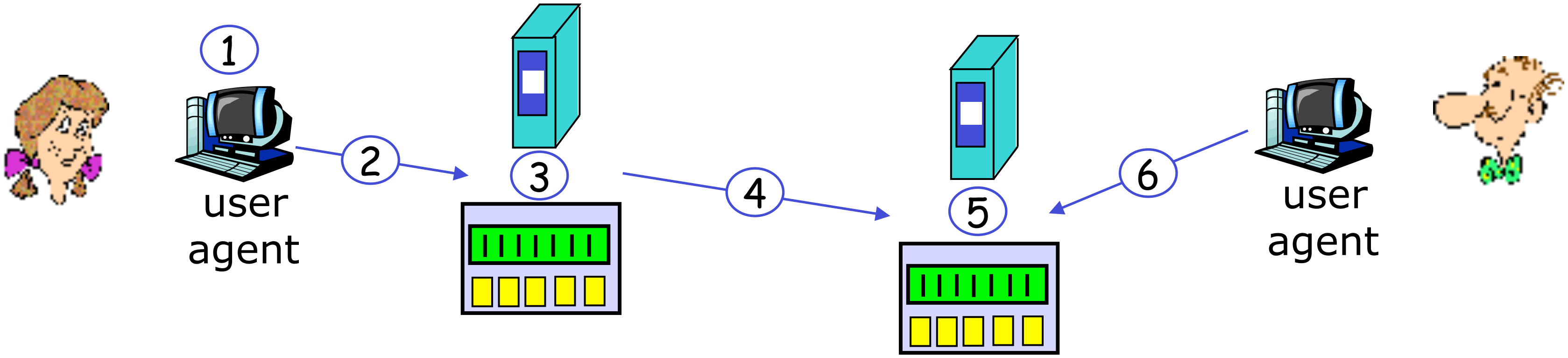
2) Alice's user agent sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message



Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C:   How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Try SMTP interaction for yourself:

- **`telnet servername 25`**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

SMTP: final words

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII

Comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

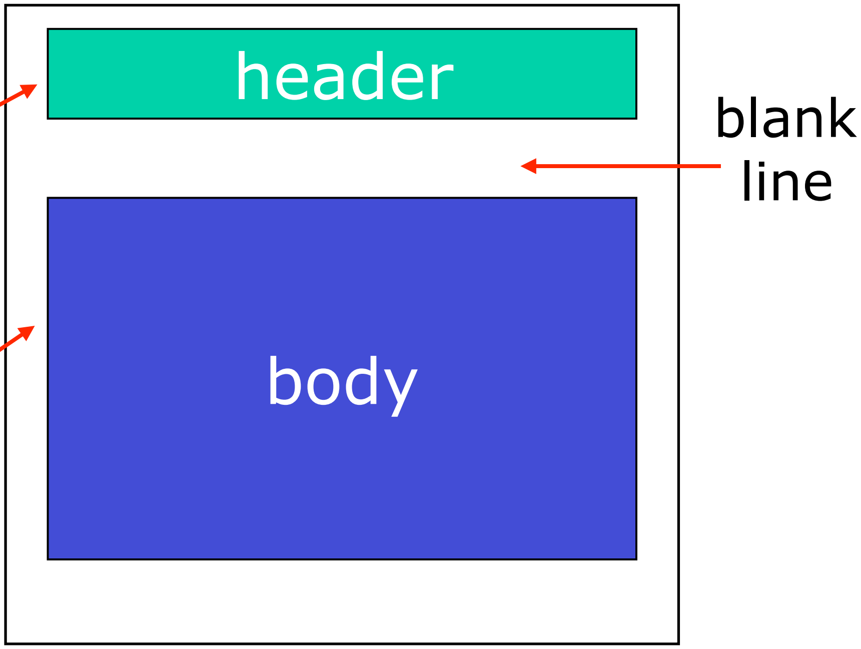
Mail message format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

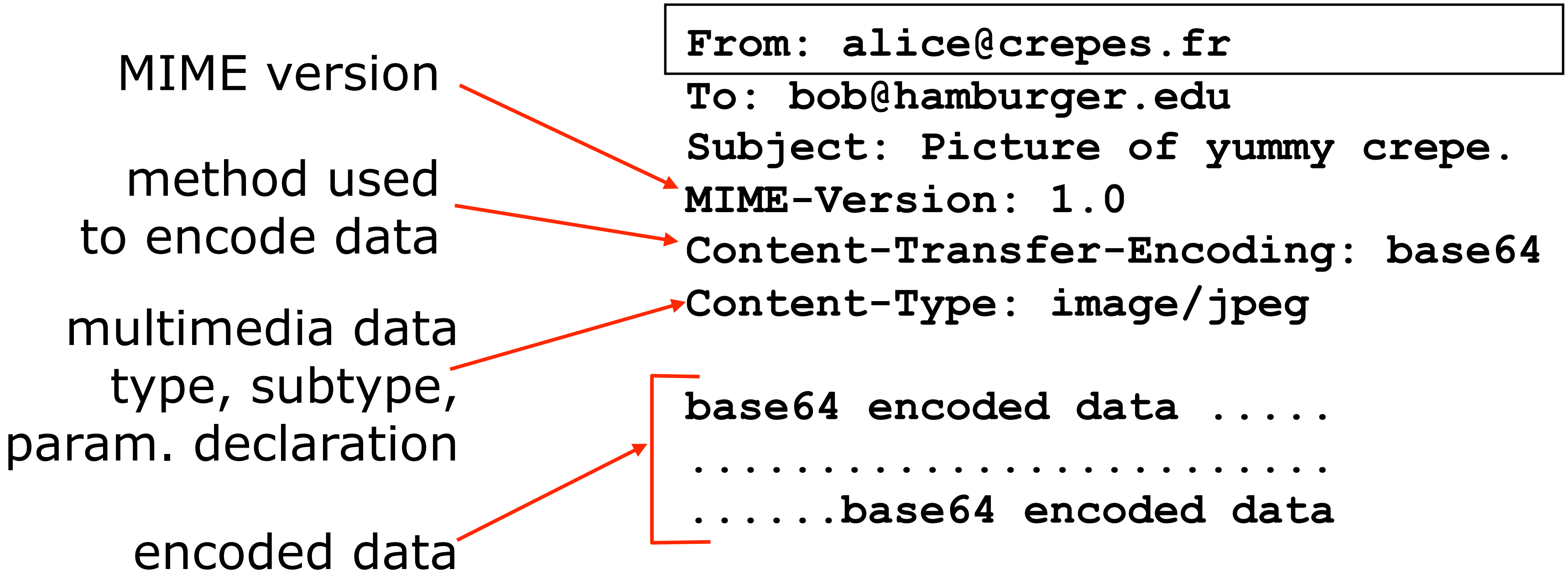
- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP commands!
- body
  - the "message", ASCII characters only



Message format: multimedia extensions

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



MIME version

method used to encode data

multimedia data type, subtype, param. declaration

encoded data

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

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

MIME types

`Content-Type: type/subtype; parameters`

Text

- example subtypes: **`plain`**, **`html`**

Image

- example subtypes: **`jpeg`**, **`gif`**

Audio

- example subtypes: **`basic`** (8-bit mu-law encoded), **`32kadpcm`** (32 kbps coding)

Video

- example subtypes: **`mpeg`**, **`quicktime`**

Application

- other data that must be processed by reader before "viewable"
- example subtypes: **`msword`**, **`octet-stream`**

Multipart Type

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=StartOfNextPart

--StartOfNextPart
Dear Bob, Please find a picture of a crepe.
--StartOfNextPart
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.........................
......base64 encoded data
--StartOfNextPart
Do you want the recipe?
```

POP3

- Short for ***Post Office Protocol***, a protocol used to retrieve e-mail from a mail server.
- Still most e-mail applications use the POP protocol, although some can use the newer IMAP (Internet Message Access Protocol).
- There are two versions of POP. The first, called POP2, became a standard in the mid-80's and requires SMTP to send messages. The newer version, POP3, can be used with or without SMTP. POP3 uses TCP/IP port 110.

POP3 protocol

authorization phase

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

transaction phase, client:

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

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

```
 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
```

POP3 (more) and IMAP

More about POP3

- Previous example uses “download and delete” mode.
- Bob cannot re-read e-mail if he changes client
- “Download-and-keep”: copies of messages on different clients
- POP3 is stateless across sessions

IMAP

- Keep all messages in one place: the server
- Allows user to organize messages in folders
- IMAP keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name

IMAP

- IMAP is an **Internet Message Access Protocol**. It is a method of accessing electronic mail messages that are kept on a possibly shared mail server. In other words, it permits a "client" email program to access remote message stores as if they were local. For example, email stored on an IMAP server can be manipulated from a desktop computer at home, a workstation at the office, and a notebook computer while travelling, without the need to transfer messages or files back and forth between these computers.
- IMAP uses TCP/IP port 143.

POP3 vs IMAP

- With IMAP, all your mail stays on the server in multiple folders, some of which you have created. This enables you to connect to any computer and see all your mail and mail folders. In general, IMAP is great if you have a dedicated connection to the Internet or you like to check your mail from various locations.
- With POP3 you only have one folder, the Inbox folder. When you open your mailbox, new mail is moved from the host server and saved on your computer. If you want to be able to see your old mail messages, you have to go back to the computer where you last opened your mail.
- With POP3 "leave mail on server" only your email messages are on the server, but with IMAP your email folders are also on the server.

Test POP3 and IMAP …..

```
# telnet localhost 143
Connected to staff-mail.lboro.ac.uk.
Escape character is '^]'.
* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE
THREAD=ORDEREDSUBJECT
THREAD=REFERENCES SORT QUOTA IDLE ACL ACL2=UNION STARTTLS]
Courier-IMAP ready.
Copyright 1998-2005 Double Precision, Inc.  See COPYING for distribution
information.
a login username password
a OK LOGIN Ok.
a examine inbox
* FLAGS (\Answered \Flagged \Deleted \Seen \Recent)
* OK [PERMANENTFLAGS ()] No permanent flags permitted
* 26 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 989061119] Ok
* OK [READ-ONLY] Ok
a logout
* BYE Courier-IMAP server shutting down
a OK LOGOUT completed
Connection closed by foreign host.
```

Exim

- Exim is an open source mail transfer agent (MTA), which is a program responsible for receiving, routing, and delivering e-mail messages (this type of program is sometimes referred to as an Internet mailer, or a mail server program). MTAs receive e-mail messages and recipient addresses from local users and remote hosts, perform alias creation and forwarding functions, and deliver the messages to their destinations. Exim was developed at the University of Cambridge for the use of Unix systems connected over the Internet. The software can be installed in place of sendmail, the most common MTA for UNIX and Linux systems. In comparison to sendmail, Exim is said to feature more straightforward configuration and task management.

Network Address Translation

(NAT)

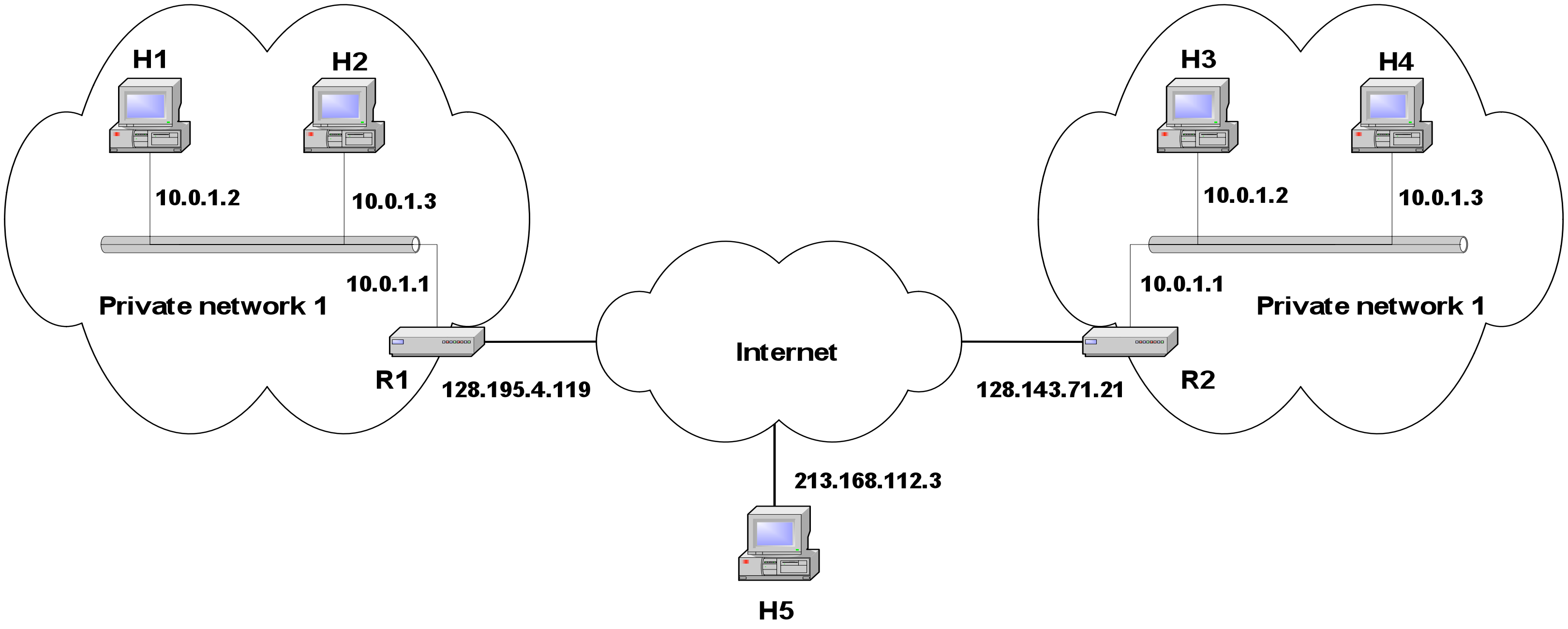
NAT: Network Address Translation

- Motivation: local network uses just one IP address as far as outside word is concerned:
  - no need to be allocated range of addresses from ISP: just one IP address is used for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).

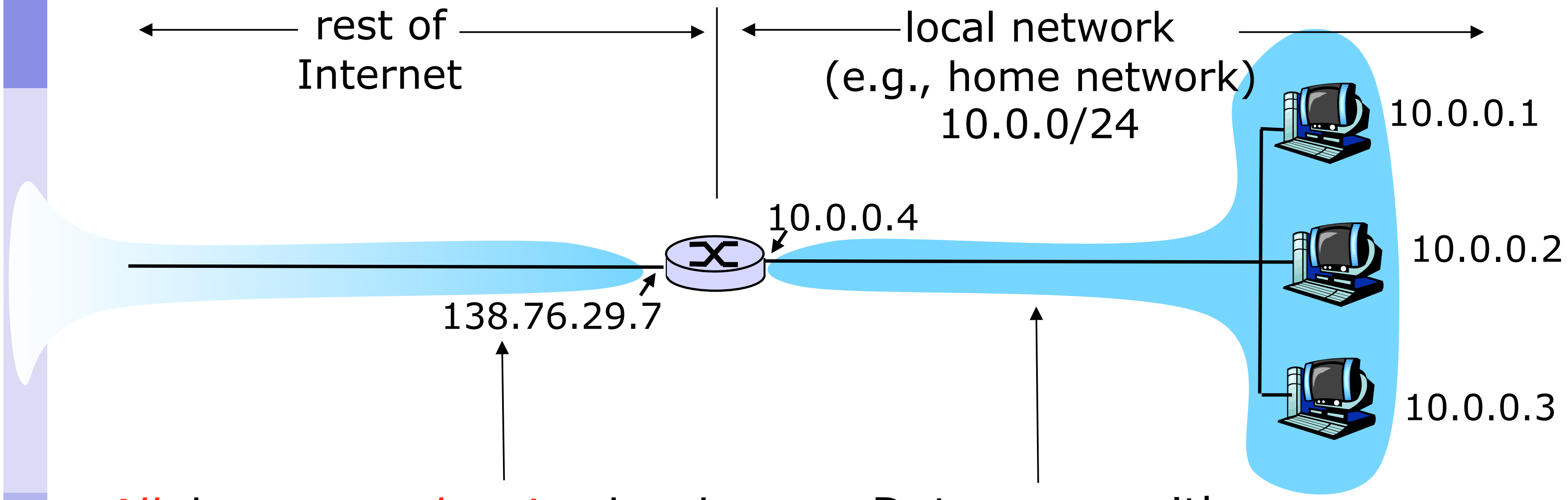
Private Network

- *Private IP* network is an IP network that is not directly connected to the Internet
- IP addresses in a private network can be assigned arbitrarily.
  - Not registered and not guaranteed to be globally unique
- Generally, private networks use addresses from the following experimental address ranges (*non-routable addresses*):
  - 10.0.0.0 – 10.255.255.255
  - 172.16.0.0 – 172.31.255.255
  - 192.168.0.0 – 192.168.255.255

Private Addresses



NAT: Network Address Translation



All datagrams *leaving* local network have same single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: Network Address Translation

Implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr.

- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

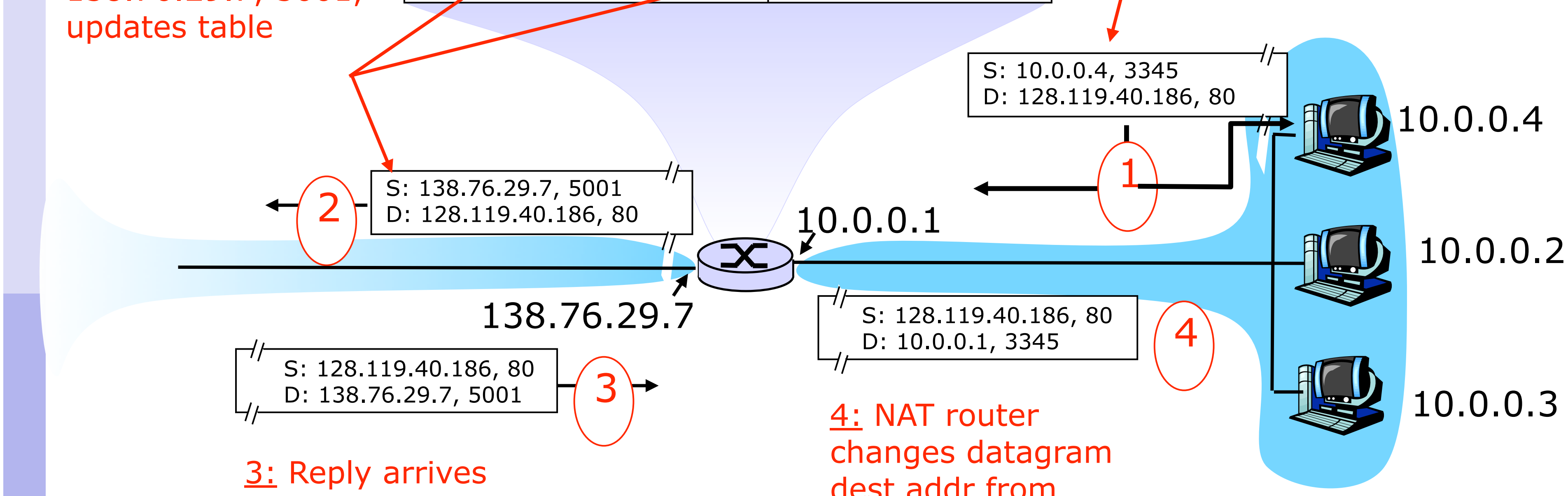
NAT: Network Address Translation

NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.4, 3345
……	……

2: NAT gw changes datagram source addr from 10.0.0.4, 3345 to 138.76.29.7, 5001, updates table

1: host 10.0.0.4 sends datagram to 128.119.40, 80



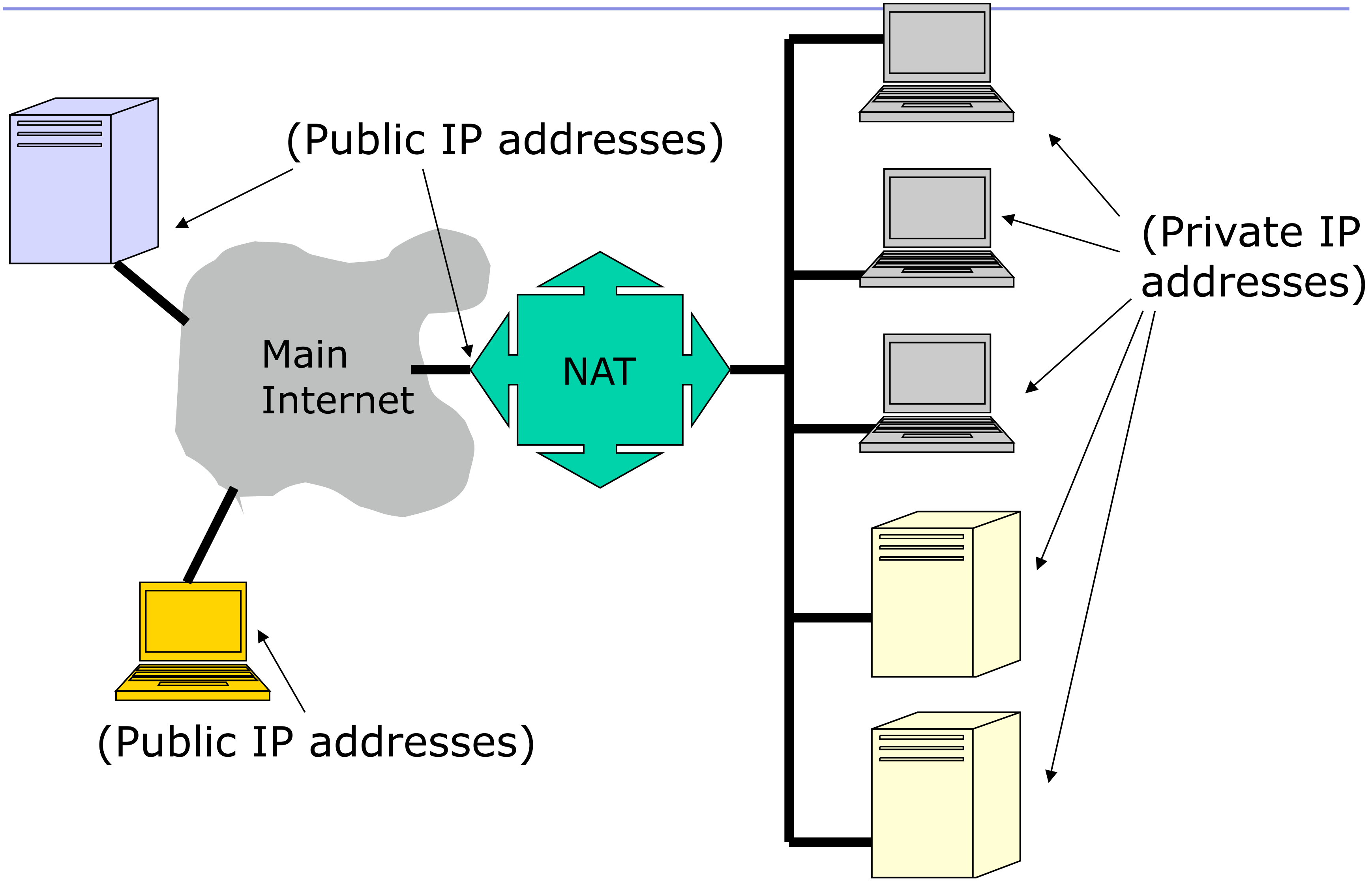
3: Reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.4, 3345

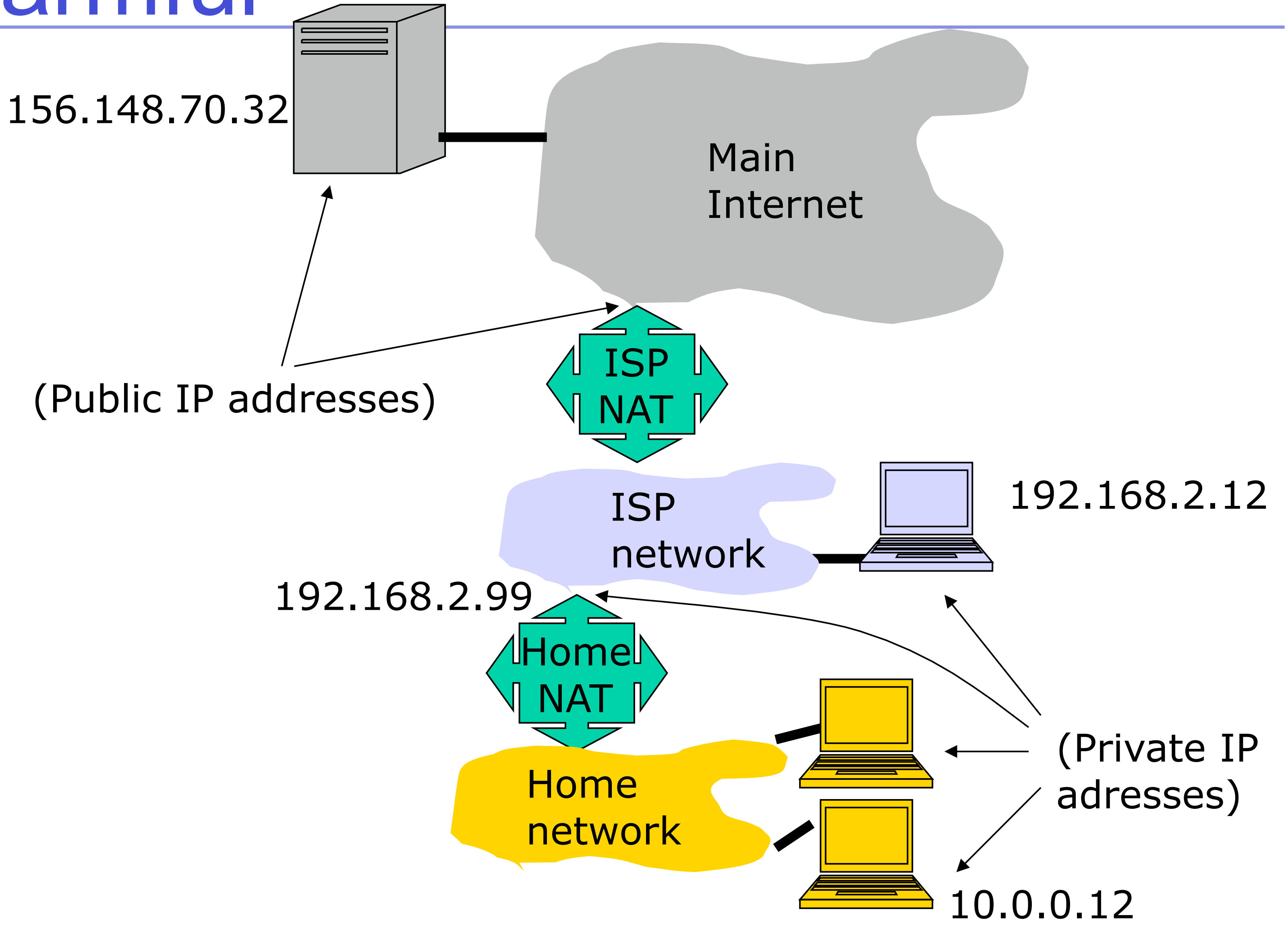
NAT: Network Address Translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, eg, P2P applications
  - address shortage should instead be solved by IPv6

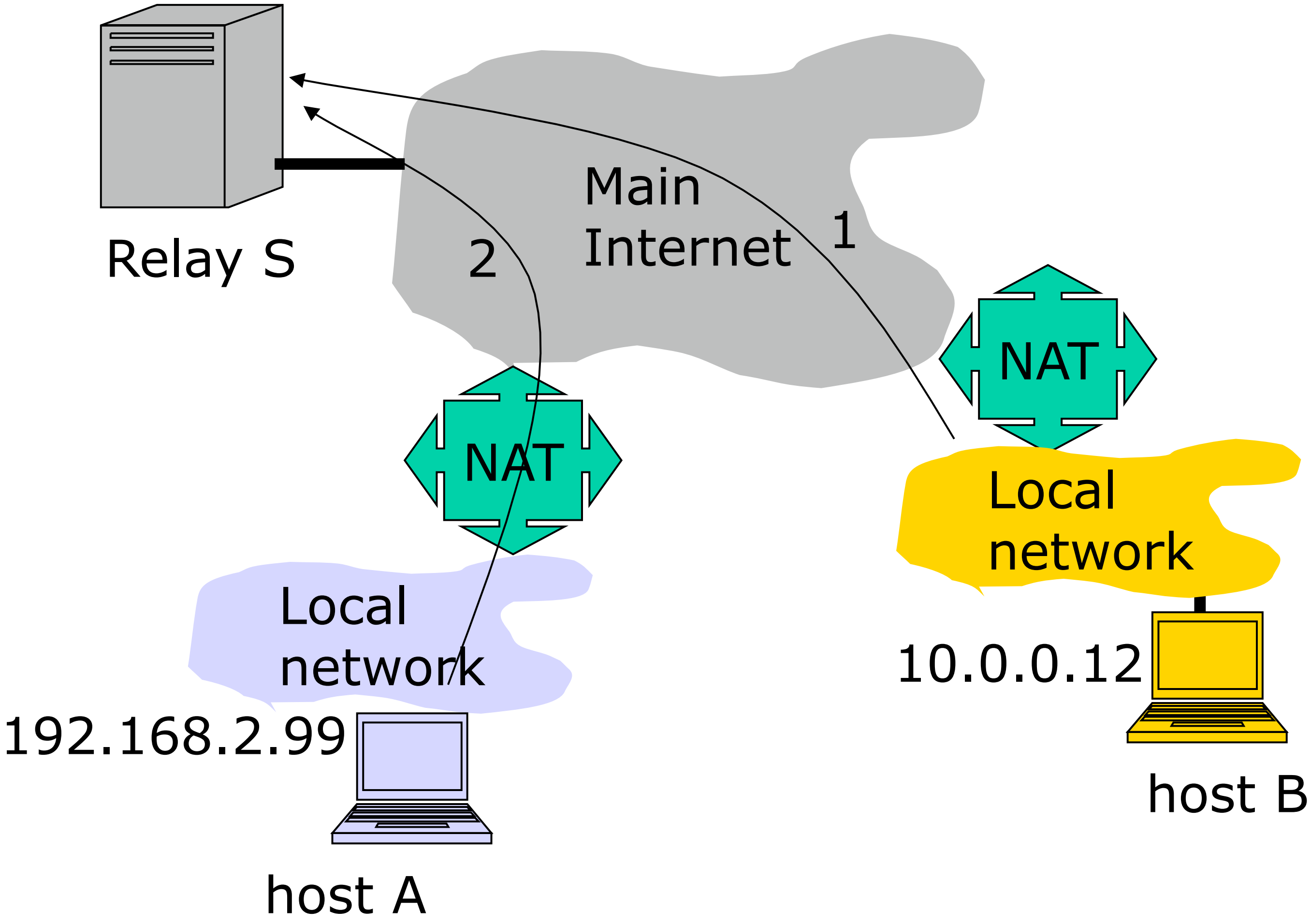
Simple NAT


(Public IP addresses)
Main Internet
NAT
(Private IP addresses)
(Public IP addresses)

Provider NATs? Considered Harmful



NAT traversal: relay



TURN protocol

- Protocol for UDP/TCP relaying behind NAT
- Data is bounced to a public TURN server
- No hole punching
- TURN works even behind symmetric NAT

Hole punching

- Technique to allow traffic from/to a host behind a firewall/NAT without collaboration of the NAT itself
- UDP: simple☺
- TCP:
  - Berkeley sockets allows TCP socket to initiate an outgoing or listen for an incoming connections
    but not both
  - Solution: bind multiple sockets to same local endpoint

STUN (RFC 3489)

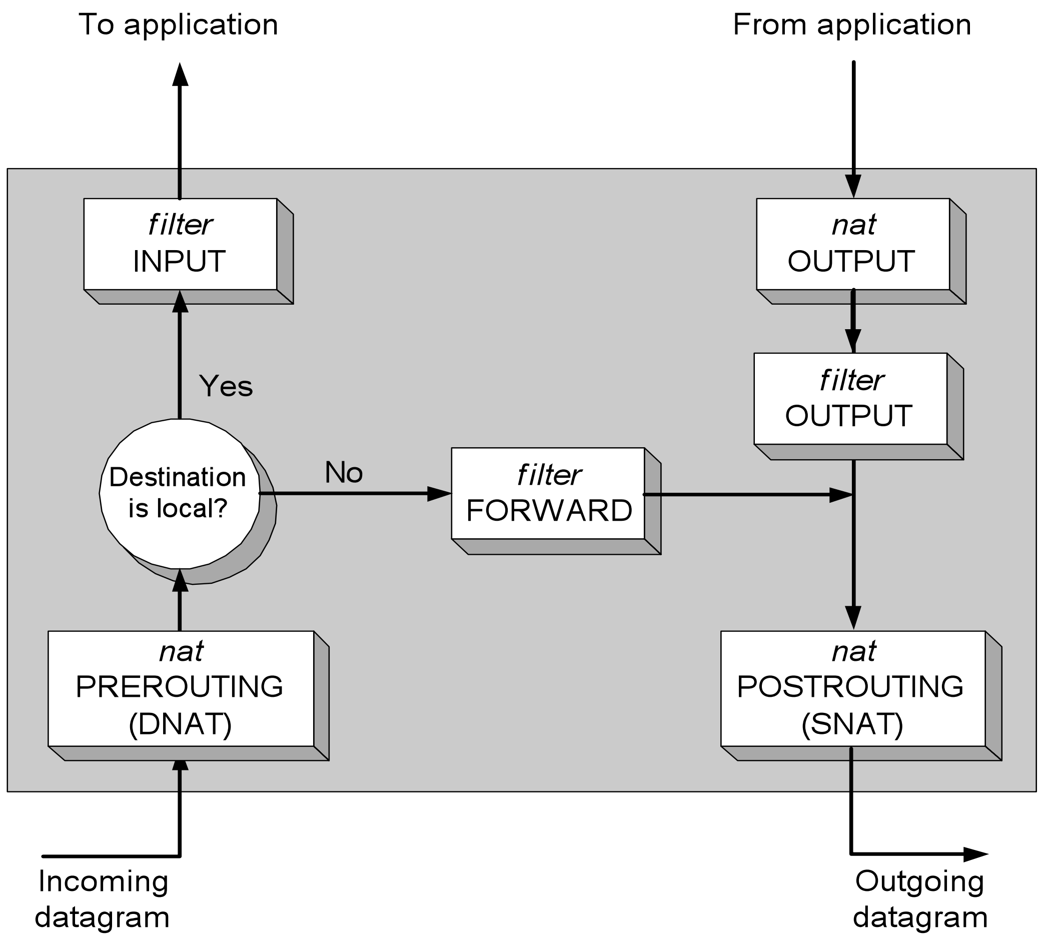
- Defines operations and message formats to understand type of NAT
- Discovers presence and type of NAT and firewalls between them and Internet
- Allows applications to determine their public NAT IP address

STUNT

- Simple Traversal of UDP Through NATs and TCP too (STUNT)
- Extends STUN to include TCP functionality

Configuring NAT in Linux

- Linux uses the Netfilter/iptable package to add filtering rules to the IP module



Configuring NAT with iptable

- **First example:**

```
iptables -t nat -A POSTROUTING -s 10.0.1.2
         -j SNAT --to-source 128.143.71.21
```

- **Pooling of IP addresses:**

```
iptables -t nat -A POSTROUTING -s 10.0.1.0/24
         -j SNAT --to-source 128.128.71.0–128.143.71.30
```

- **ISP migration:**

```
iptables -t nat -R POSTROUTING -s 10.0.1.0/24
         -j SNAT --to-source 128.195.4.0–128.195.4.254
```

- **IP masquerading:**

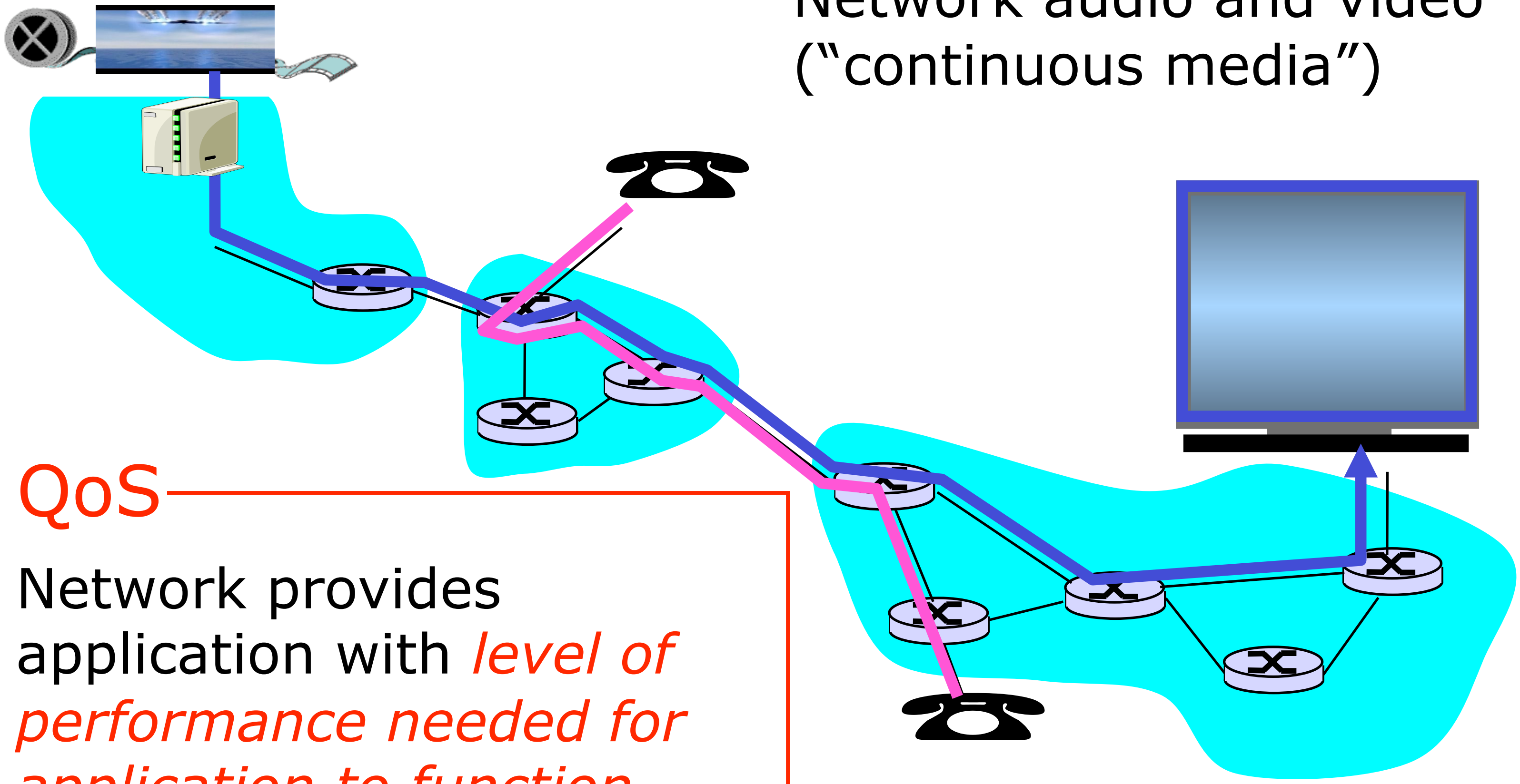
```
iptables -t nat -A POSTROUTING -s 10.0.1.0/24
         -o eth1 -j MASQUERADE
```

- **Load balancing:**

```
iptables -t nat -A PREROUTING -i eth1 -j DNAT --to-
destination 10.0.1.2-10.0.1.4
```

Multimedia Networking

Multimedia, Quality of Service (QoS): What is it?



Multimedia applications: Network audio and video ("continuous media")

QoS

Network provides application with *level of performance needed for application to function.*

Goals

Principles

- Classify multimedia applications
- Identify the network services the apps need
- Making the best of best effort service

Protocols and architectures

- Specific protocols for best-effort
- Architectures for QoS

MM networking applications

Classes of MM applications:

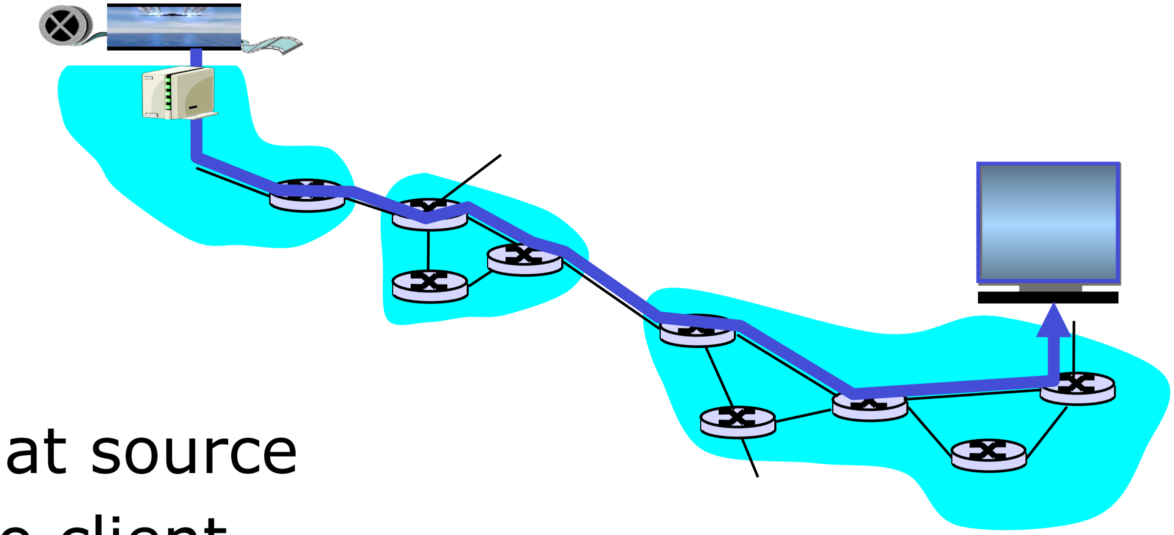
1) Streaming stored audio and video
2) Streaming live audio and video
3) Real-time interactive audio and video

Jitter is the variability of packet delays within the same packet stream

Fundamental characteristics:

- Typically **delay sensitive**
  - End-to-end delay
  - Delay jitter
- But **loss tolerant**: infrequent losses cause minor glitches
- Antithesis of data, which are loss intolerant but delay tolerant.

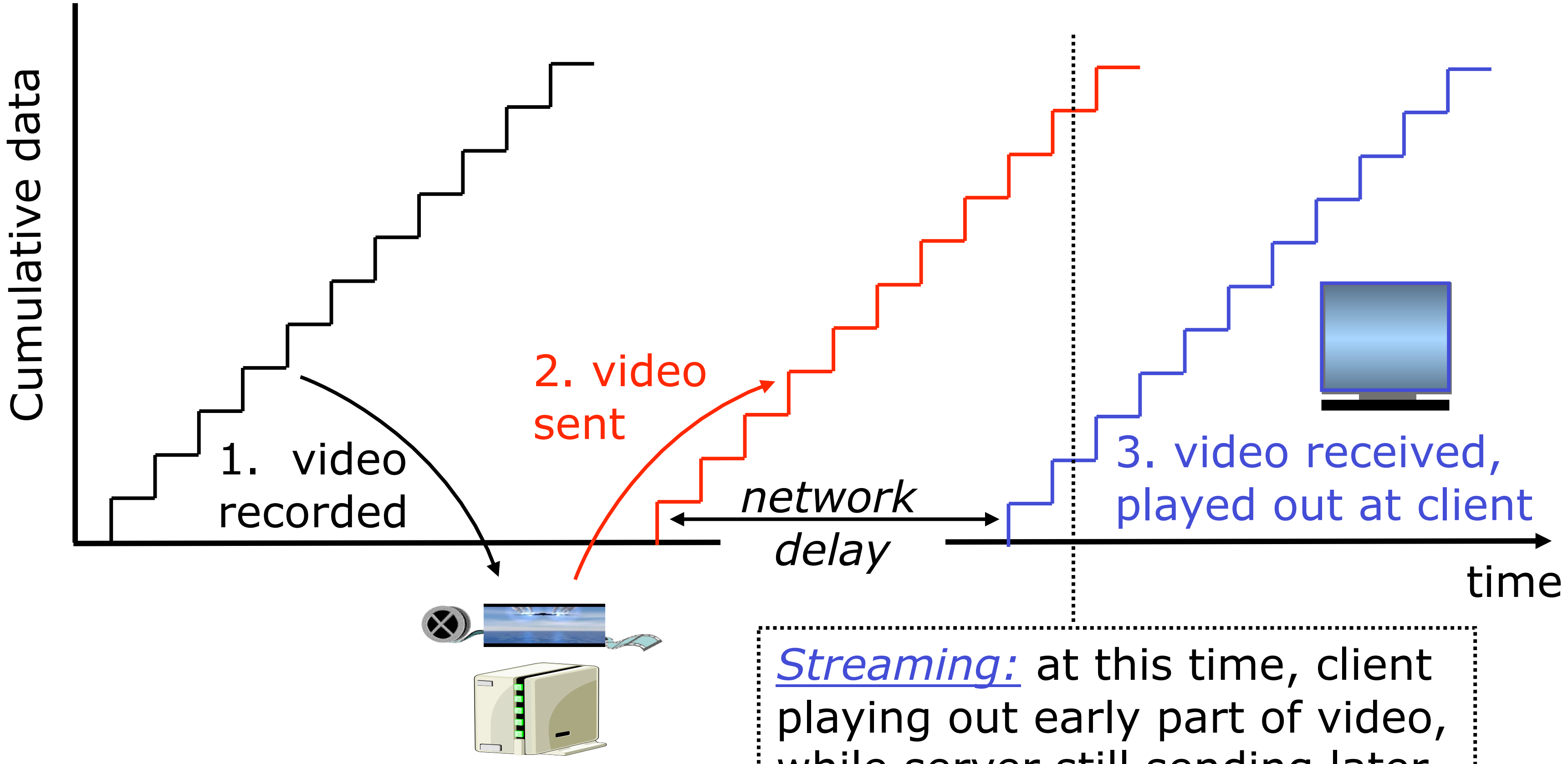
Streaming stored multimedia



- Streaming:
- Media stored at source
- Transmitted to client

- Streaming: client playout begins *before* all data has arrived
- Timing constraint for still-to-be transmitted data: in time for playout

Streaming stored multimedia: What is it?



Streaming: at this time, client playing out early part of video, while server still sending later part of video

Streaming live multimedia

Examples:

- Internet radio talk show
- Live sporting event

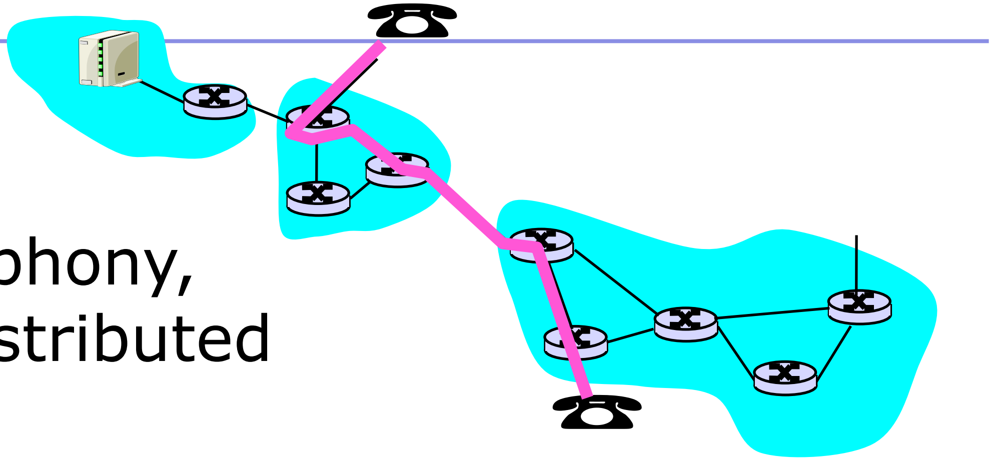
Streaming

- Playback buffer
- Playback can lag tens of seconds after transmission
- Still have timing constraint

Interactivity

- Fast forward impossible
- Rewind, pause possible!

Interactive, real-time multimedia



- Applications: IP telephony, video conference, distributed interactive worlds
  - End-end delay requirements:
    - Audio: < 150 msec good, < 400 msec OK
      - Includes application-level (packetization) and network delays
      - Higher delays noticeable, impair interactivity
  - Session initialization
    - How does callee advertise its IP address, port number, encoding algorithms?

A few words about audio compression

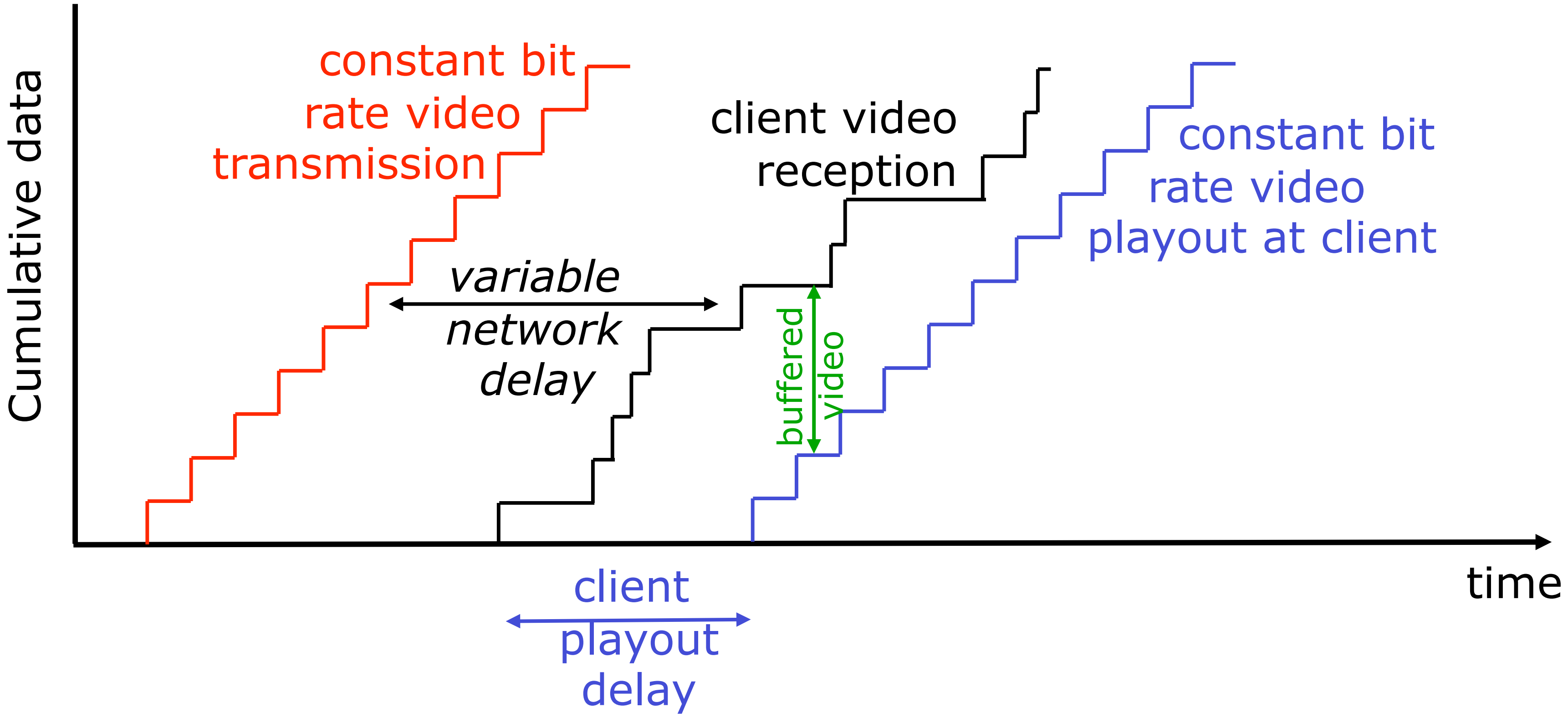
- Analog signal sampled at constant rate
  - Telephone: 8,000 samples/sec
  - CD music: 44,100 samples/sec
- Each sample quantized, i.e., rounded
  - E.g., 2^8=256 possible quantized values
- Each quantized value represented by bits
  - 8 bits for 256 values

- Example: 8,000 samples/sec, 256 quantized values --> 64,000 bps
- Receiver converts it back to analog signal:
  - Some quality reduction

Example rates

- CD: 1.411 Mbps
- MP3: 96, 128, 160 kbps
- Internet telephony: 5.3 - 13 kbps

Streaming multimedia: Client buffering



- Client-side buffering, playout delay compensate for network-added delay, delay jitter

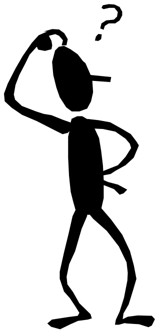
Packet loss and delay

- Network loss: IP datagram lost due to network congestion (router buffer overflow)
- Delay loss: IP datagram arrives too late for playout at receiver
  - Delays: processing, queuing in network; end-system (sender, receiver) delays
  - Typical maximum tolerable delay: 400 ms
- Loss tolerance: depending on voice encoding, losses concealed, packet loss rates between 1% and 10% can be tolerated.

Multimedia over today’s Internet

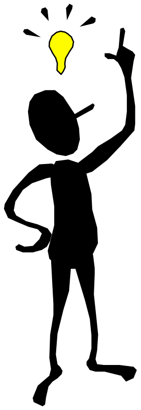
TCP/UDP/IP: “best-effort service” no “QoS”

- *No* guarantees on delay, loss



? ? ? ? ? ?

But you said multimedia apps requires ?
QoS and level of performance to be
? ? effective! ? ?



Today’s Internet multimedia applications use application-level techniques to mitigate (as best possible) effects of delay, loss

How should the Internet evolve to better support multimedia?

Integrated services philosophy:

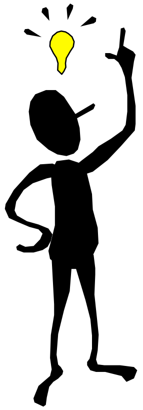
- Fundamental changes in Internet so that apps can reserve end-to-end resources including bandwidth
- Requires new, complex software in hosts & routers

Laissez-faire:

- No major changes
- More bandwidth when needed
- Content distribution, application-layer multicast
  - Application layer

Differentiated services philosophy:

- Fewer changes to Internet infrastructure, yet provide 1st and 2nd class service



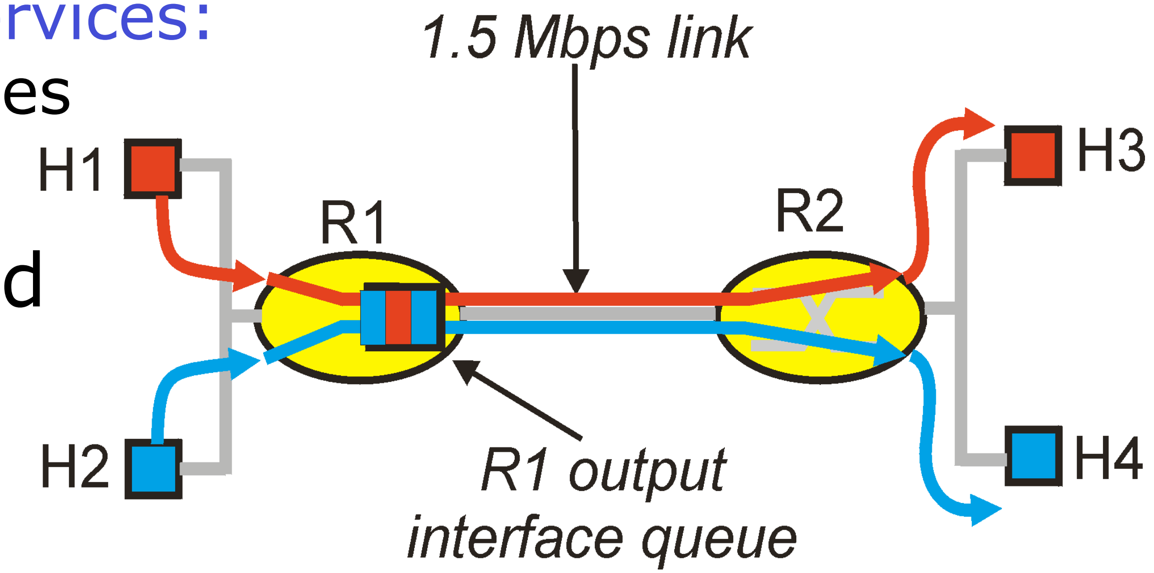
Improving QOS in IP Networks

Thus far: “making the best of best effort”

Future: next generation Internet with QoS guarantees

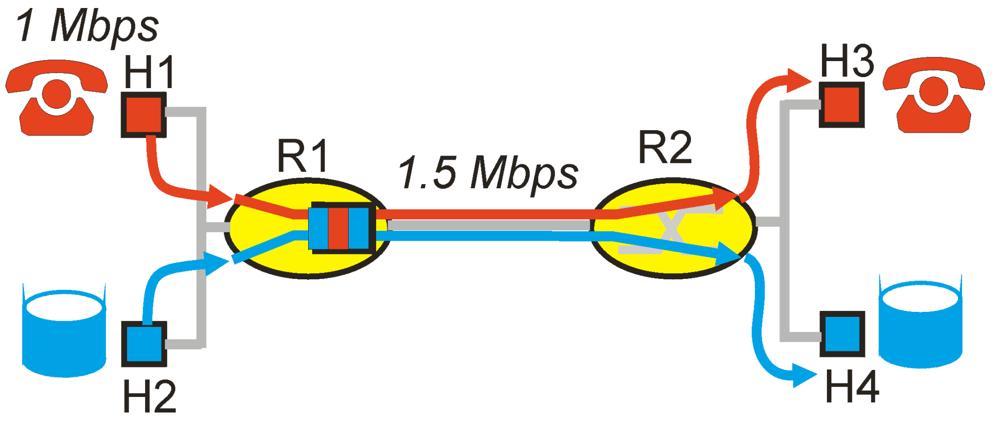
- RSVP: signaling for resource reservations
- Differentiated Services: differential guarantees
- Integrated Services: firm guarantees

- simple model for sharing and congestion studies:



Principles for QOS Guarantees

- Example: 1Mbps IP phones and FTP/p2p share 1.5 Mbps link.
  - bursts of FTP can congest router, cause audio loss
  - want to give priority to audio over FTP

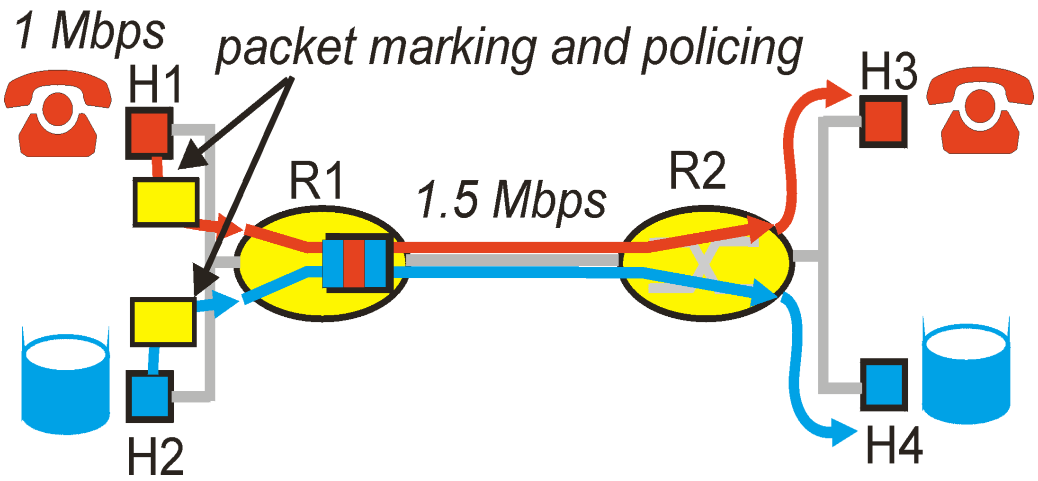


Principle 1

packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly

Principles for QOS Guarantees

- what if applications misbehave (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:
  - similar to ATM UNI (User Network Interface)

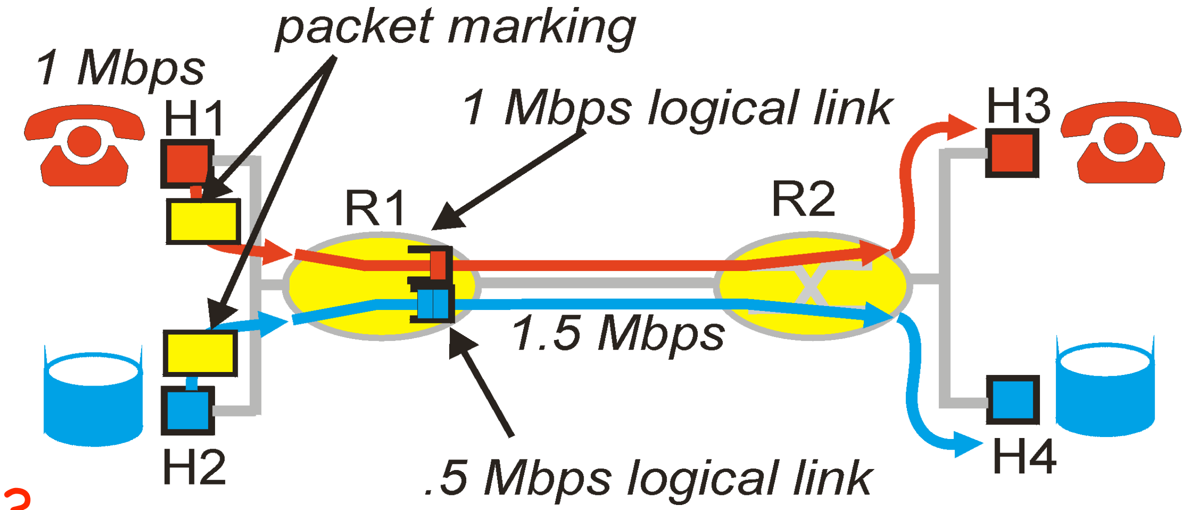


Principle 2

provide protection (*isolation*) for one class from others

Principles for QOS Guarantees

- Allocating fixed (non-sharable) bandwidth to flow: inefficient use of bandwidth if flows doesn’t use its allocation

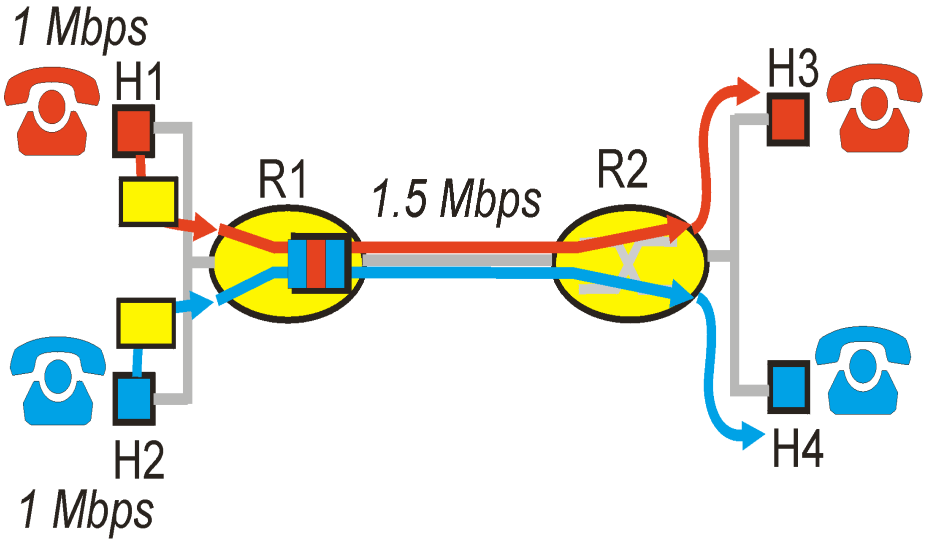


Principle 3

While providing isolation, it is desirable to use resources as efficiently as possible

Principles for QOS Guarantees

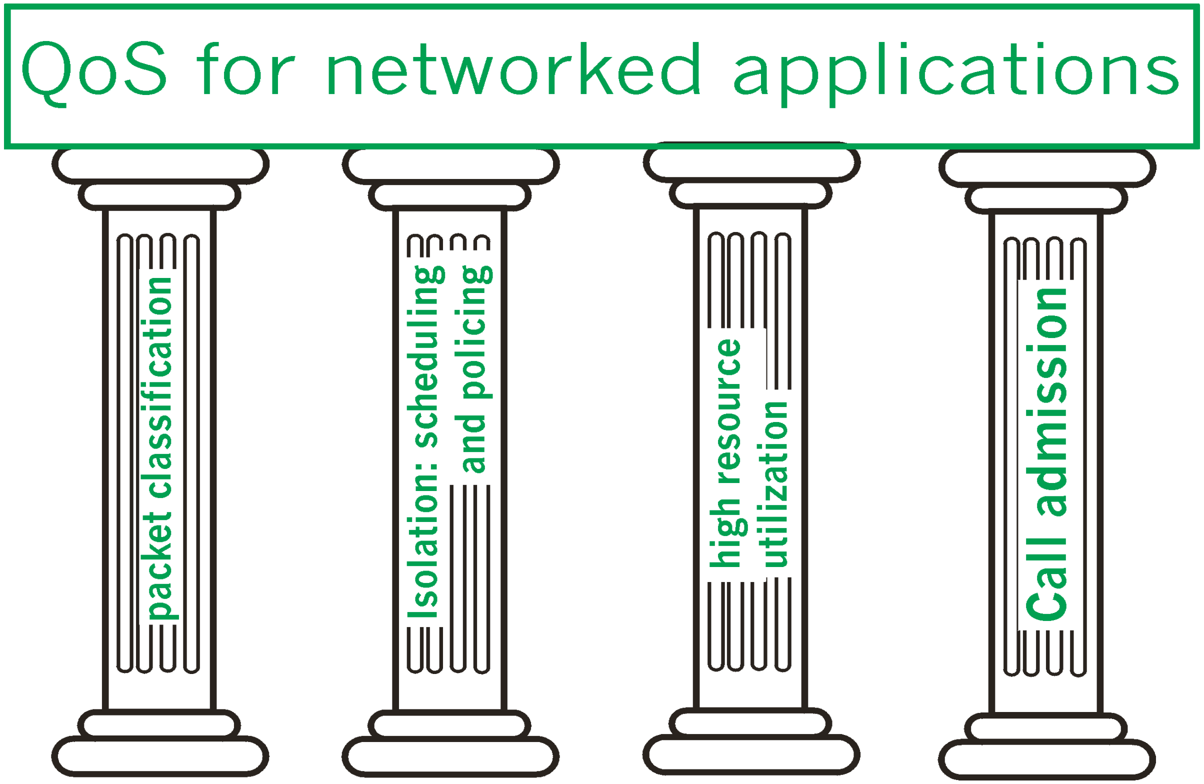
- *Basic fact of life:* can not support traffic demands beyond link capacity



Principle 4

Call Admission: flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs

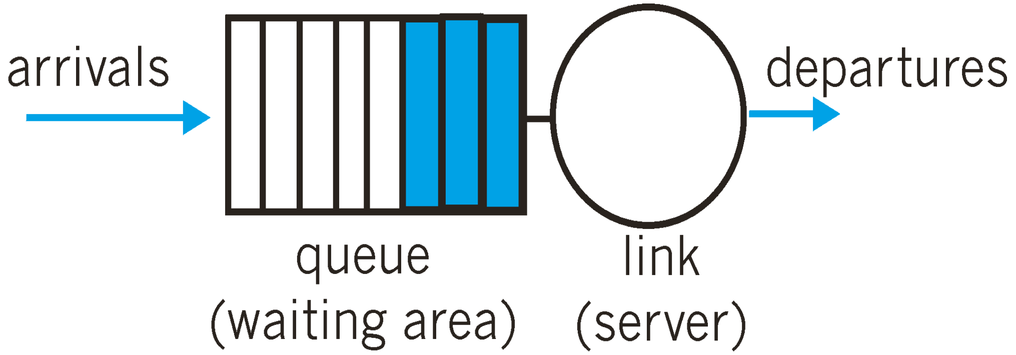
Summary of QoS Principles



QoS for networked applications

Scheduling And Policing Mechanisms

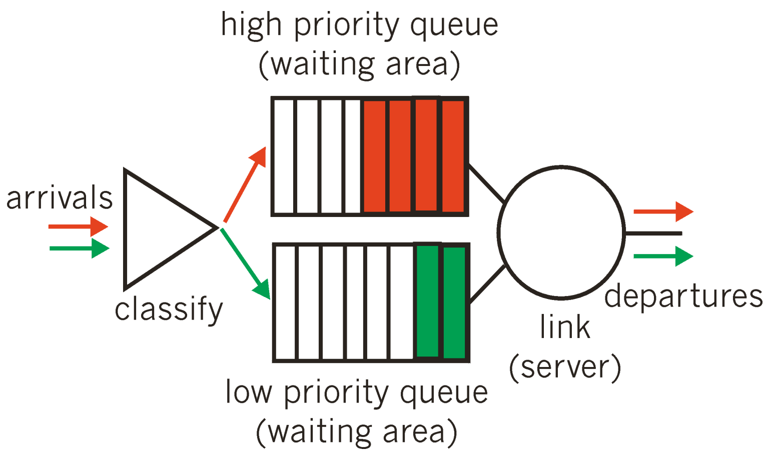
- scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - discard policy: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - priority: drop/remove on priority basis
    - random: drop/remove randomly

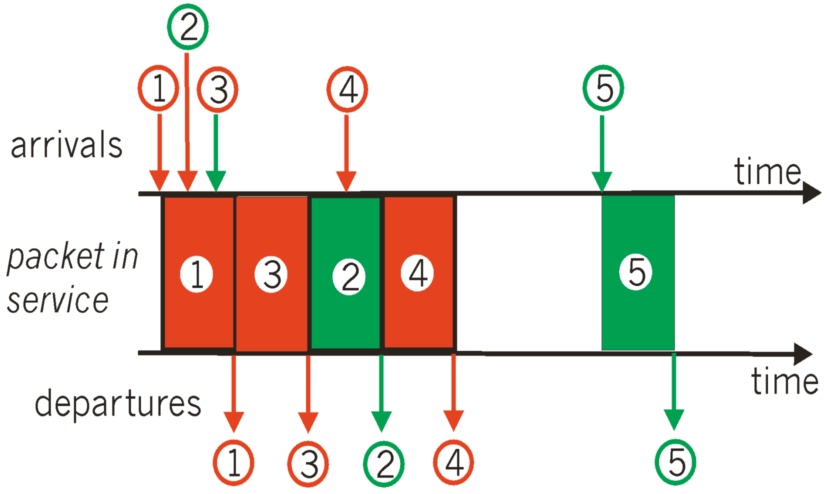


Scheduling Policies: more

Priority scheduling: transmit highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..

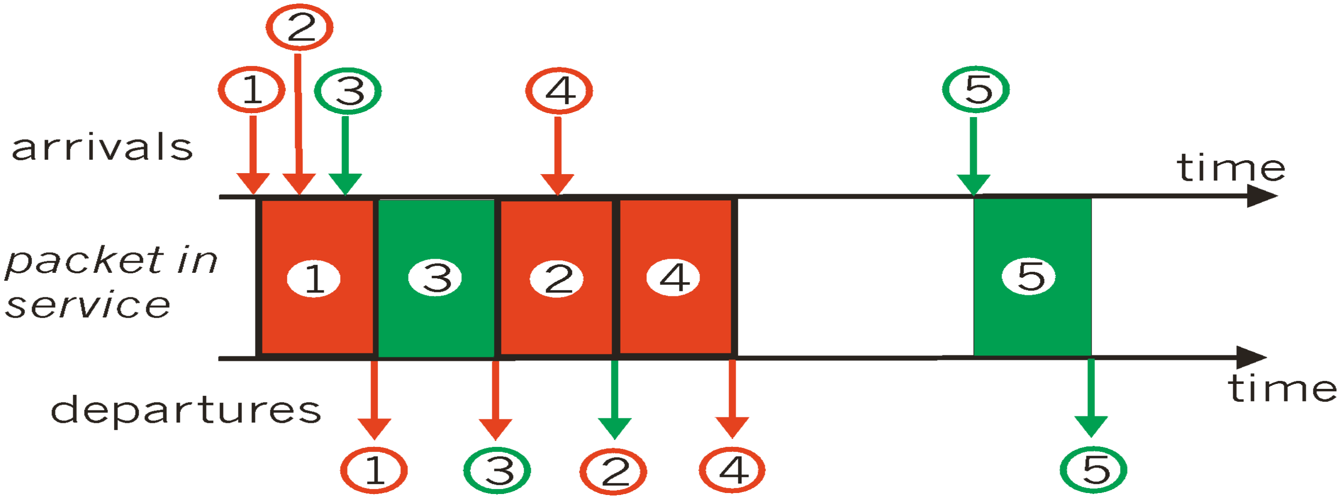




Scheduling Policies: still more

round robin scheduling:

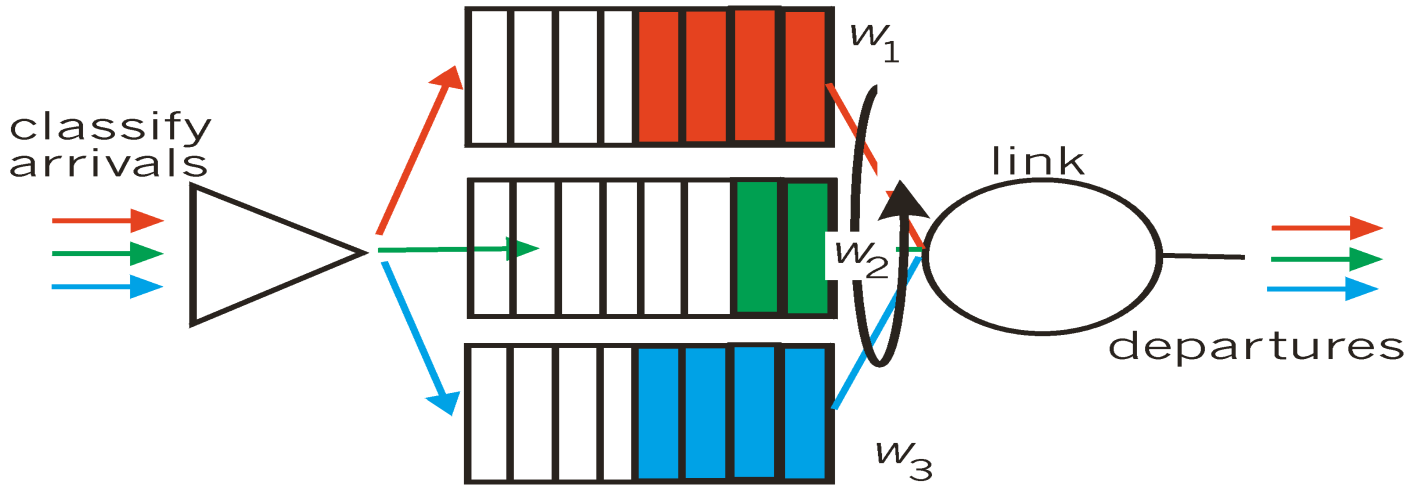
- multiple classes
- cyclically scan class queues, serving one from each class (if available)



Scheduling Policies: still more

Weighted Fair Queuing:

- generalized Round Robin
- each class gets weighted amount of service in each cycle



IETF Differentiated Services

Diffserv approach:

- simple functions in network core, relatively complex functions at edge routers (or hosts)
- Do’t define define service classes, provide functional components to build service classes

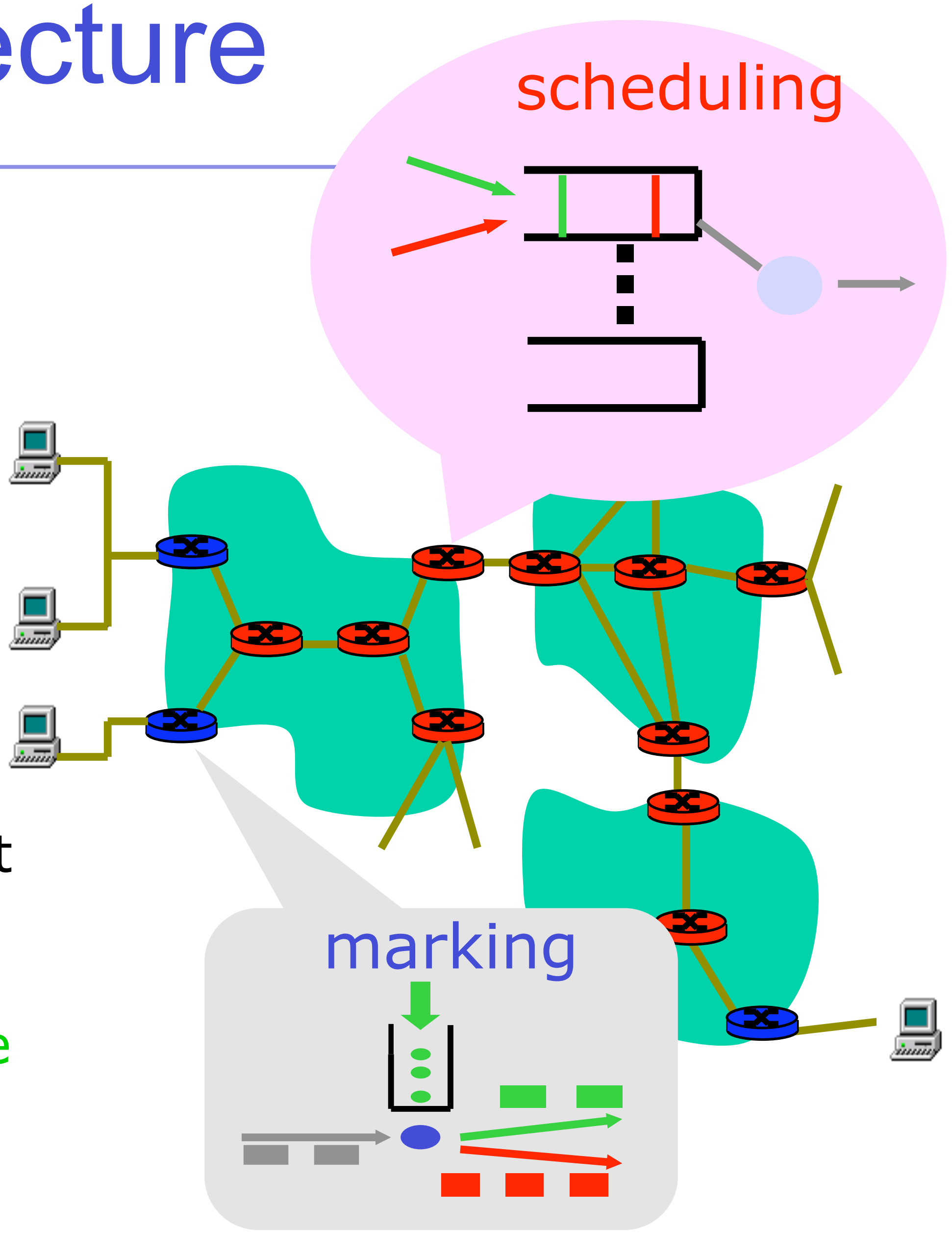
Diffserv Architecture

Edge router:

- per-flow traffic management
- marks packets as in-profile and out-profile

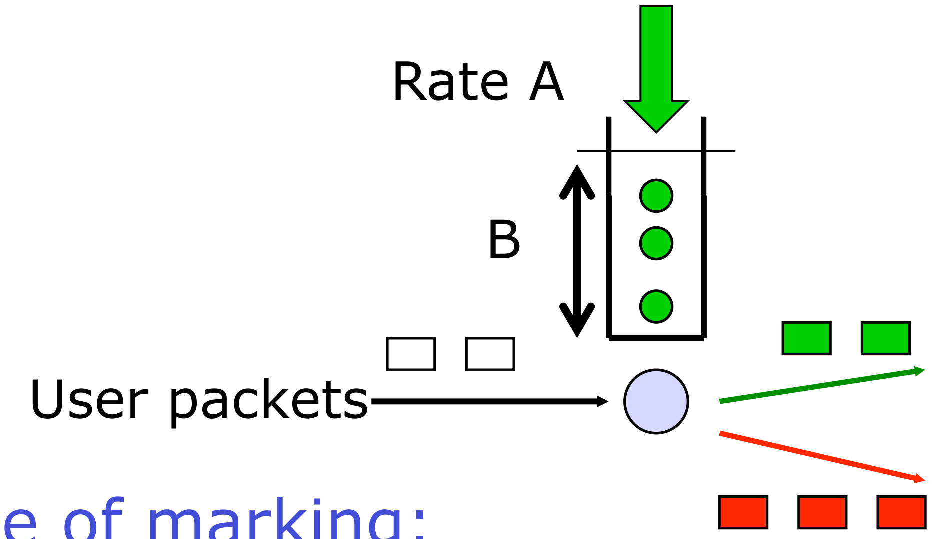
Core router:

- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets
- Assured Forwarding



Edge router pckt marking

- profile: pre-negotiated rate A, bucket size B
- packet marking at edge based on per-flow profile

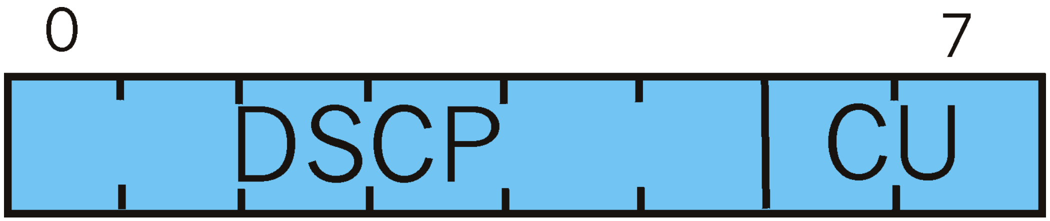


Possible usage of marking:

- class-based marking: packets of different classes marked differently
- intra-class marking: conforming portion of flow marked differently than non-conforming one

Classification and Conditioning

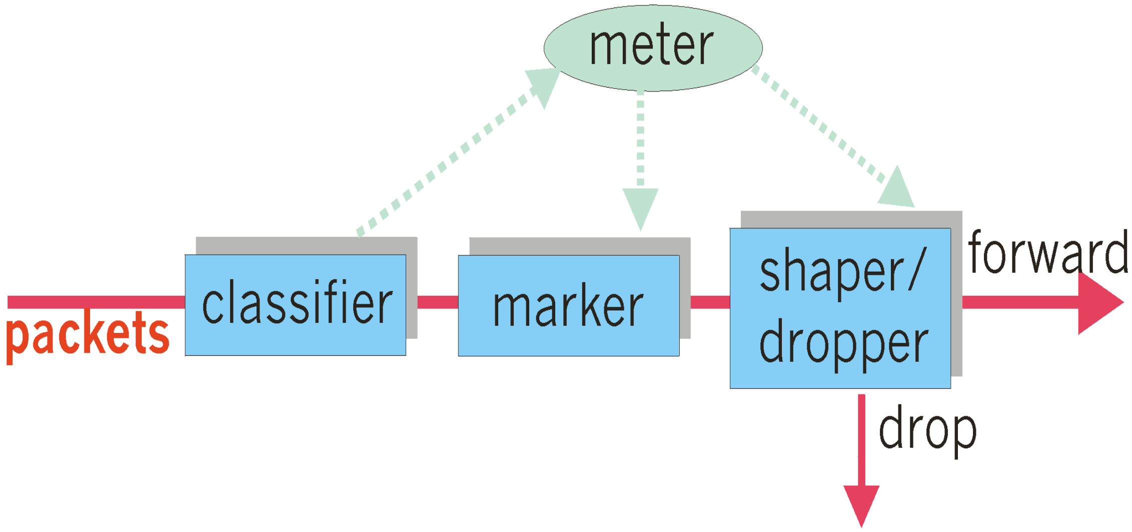
- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits are currently unused



Classification and Conditioning

may be desirable to limit traffic injection rate of some class:

- user declares traffic profile (e.g., rate, burst size)
- traffic metered, shaped if non-conforming



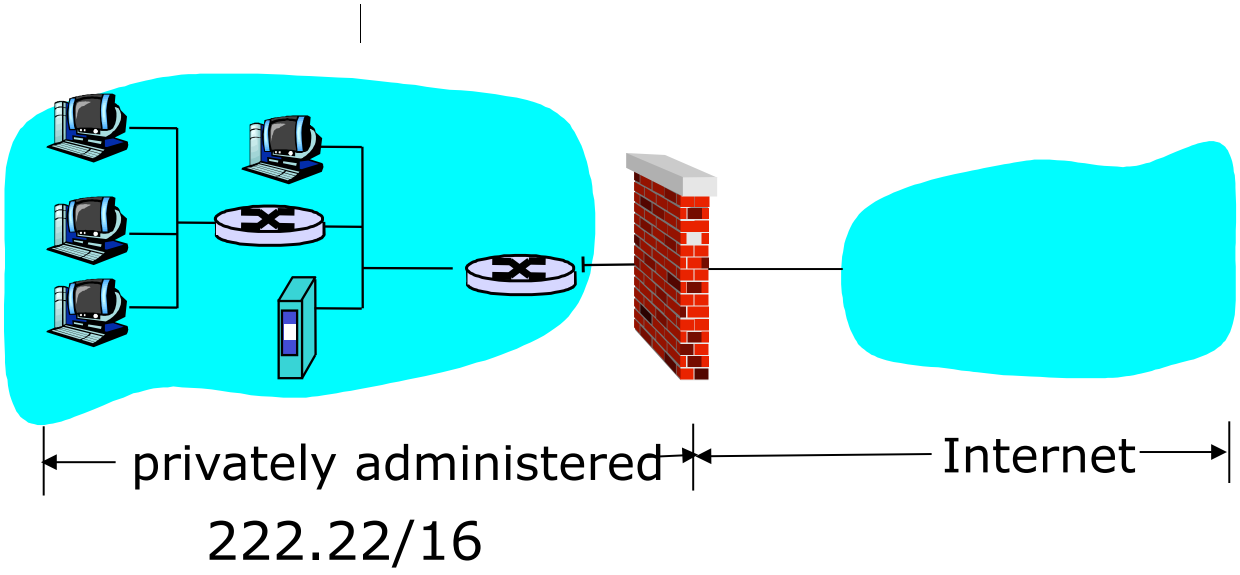
Firewalls

Firewalls

By conventional definition, a firewall is a partition made of fireproof material designed to prevent the spread of fire from one part of a building to another.

firewall

isolates organization’s internal net from larger Internet, allowing some packets to pass, blocking others.



Firewall goals

- All traffic from outside to inside and vice-versa passes through the firewall.
- Only authorized traffic, as defined by local security policy, will be allowed to pass.
- The firewall itself is immune to penetration.

Firewalls: taxonomy

1. Traditional packet filters
   - filters often combined with router, creating a firewall
2. Stateful filters
3. Application gateways

Major firewall vendors:
Checkpoint
Cisco PIX

Traditional packet filters

Analyzes each datagram going through it; makes drop decision based on:

- source IP address
- destination IP address
- source port
- destination port
- TCP flag bits
  - SYN bit set: datagram for connection initiation
  - ACK bit set: part of established connection
- TCP or UDP or ICMP
  - Firewalls often configured to block all UDP
- direction
  - Is the datagram leaving or entering the internal network?
- router interface
  - decisions can be different for different interfaces

Filtering Rules - Examples

Policy	Firewall Setting
No outside Web access.	Drop all outgoing packets to any IP address, port 80
Outside connections to public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a Smuft DoS attack.	Drop all ICMP packets going to a "broadcast" address (eg 130.207.255.255).
Prevent your network from being tracerouted	Drop all incoming ICMP

Access control lists

Apply rules from top to bottom:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Access control lists

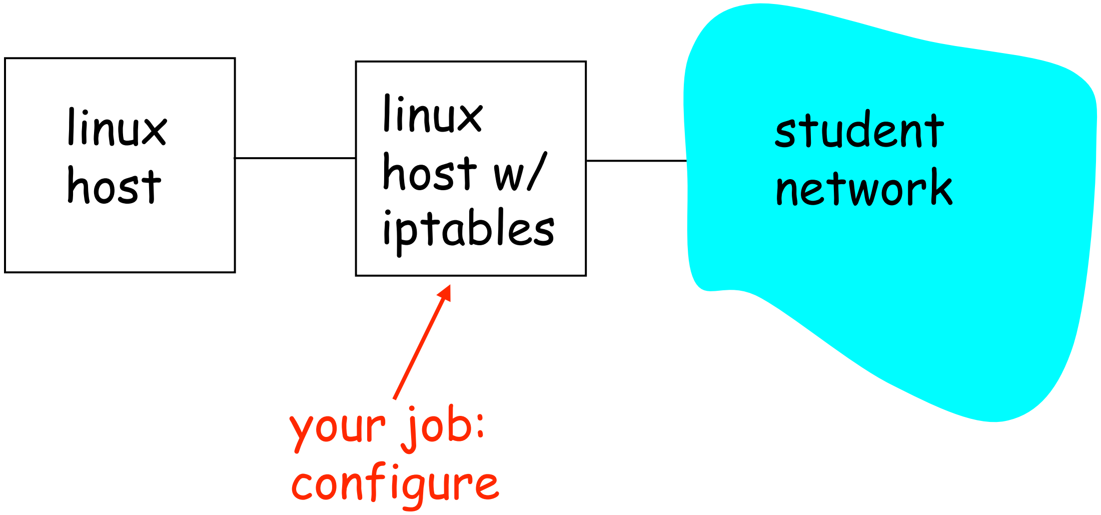
- Each router/firewall interface can have its own ACL
- Most firewall vendors provide both command-line and graphical configuration interface

Advantages and disadvantages of traditional packet filters

- Advantages
  - One screening router can protect entire network
  - Can be efficient if filtering rules are kept simple
  - Widely available. Almost any router, even Linux boxes
- Disadvantages
  - Can be penetrated
  - Cannot enforce some policies. For example, permit certain users.
  - Rules can get complicated and difficult to test

Firewall Lab: iptables

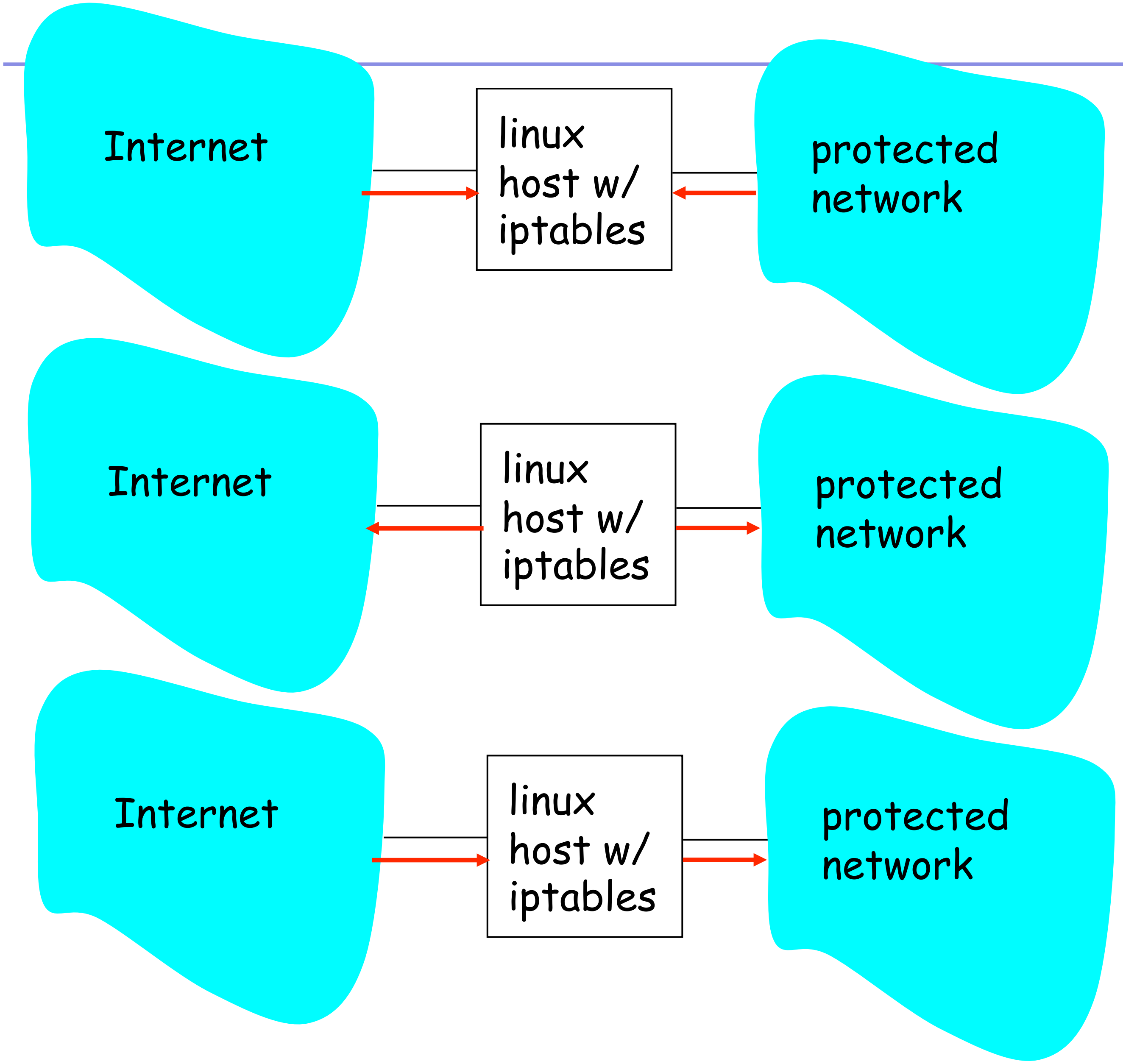
- Converts linux box into a packet filter.
- Part of most linux distributions today.



Firewall lab: iptables

- iptables
  - Provides firewall capability to a linux host
  - Comes installed with linux Fedora
  - With other versions of Linux, may need to install RPM

Chain types



Internet

linux host w/ iptables

protected network

INPUT chain

Internet

linux host w/ iptables

protected network

OUTPUT chain

Internet

linux host w/ iptables

protected network

FORWARD chain

iptables: Example command

```
iptables -A INPUT -i eth0 -s 232.16.4.0/24 -j ACCEPT
```

- Sets a rule
  - Accepts packets that enter from interface eth0 and have source address in 232.16.4/24
- Kernel applies the rules in order.
  - The first rule that matches packet determines the action for that packet
- Append: -A
  - Adds rule to bottom of list of existing rules

iptables: More examples

```
iptables -L
```

- list current rules

```
iptables -F
```

- flush all rules

```
iptables - D INPUT 2
```

- deletes 2nd rule in INPUT chain

```
iptables -I INPUT 1 -p tcp -tcp-flags SYN -s
  232.16.4.0/24 -d 0/0:22 -j ACCEPT
```

- -I INPUT 1, put rule at top
- Accept TCP SYNs to port 22 (ssh) from 232.16.4.0/24

iptables Options

```
-p protocol type (tcp, udp, icmp)
-s source IP address & port number
-d dest IP address & port number
-i interface name (lo, ppp0, eth0)
-j target (ACCEPT, DENY)
-l log this packet
--sport source port
--dport dest port
--icmp-type
```

Firewall Lab: Part A

- Rules for outgoing traffic:
    - Your local machine should be able to communicate with the student network without any restrictions.
- Rules for incoming traffic:
    - All incoming connection requests should be rejected, with the following exception:
        - Your machine should respond to Ping from network 10.0.0/24
        - Your machine should accept all incoming SSH, HTTP, FTP requests from Network 10.0/16
        - Your machine should accept all incoming telnet connections from the machine 10.0.0.1 and 10.0.0.110.
        - All multicast traffic should be allowed
        - OSPF traffic should be allowed

Firewall Lab: Part B

- Rules for outgoing traffic from internal node:
  - Outgoing SSH, and ICMP traffic should be allowed
  - All multicast traffic should be allowed
  - OSPF traffic should be allowed
  - All other traffic should be blocked
- Rules for incoming traffic to protected server:
  - All incoming SSH, http, SMTP, Ping, and anonymous ftp should be permitted
  - All multicast traffic should be allowed
  - OSPF traffic should be allowed
  - All other incoming traffic should be blocked

Stateful Filters

- In previous example, any packet with ACK=1 and source port 80 gets in.
  - Attacker could, for example, attempt a malformed packet attack by sending ACK=1 segments
- Stateful filter: Adds more intelligence to the filter decision-making process
  - Stateful = remember past packets
  - Memory implemented in a very dynamic state table

Stateful filters: example

- Log each TCP connection initiated through firewall: SYN segment
- Timeout entries which see no activity for, say, 60 seconds

source address	dest address	source port	dest port
222.22.1.7	37.96.87.123	12699	80
222.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

If rule table indicates that stateful table must be checked: check to see if there is already a connection in stateful table

Stateful filters can also remember outgoing UDP segments

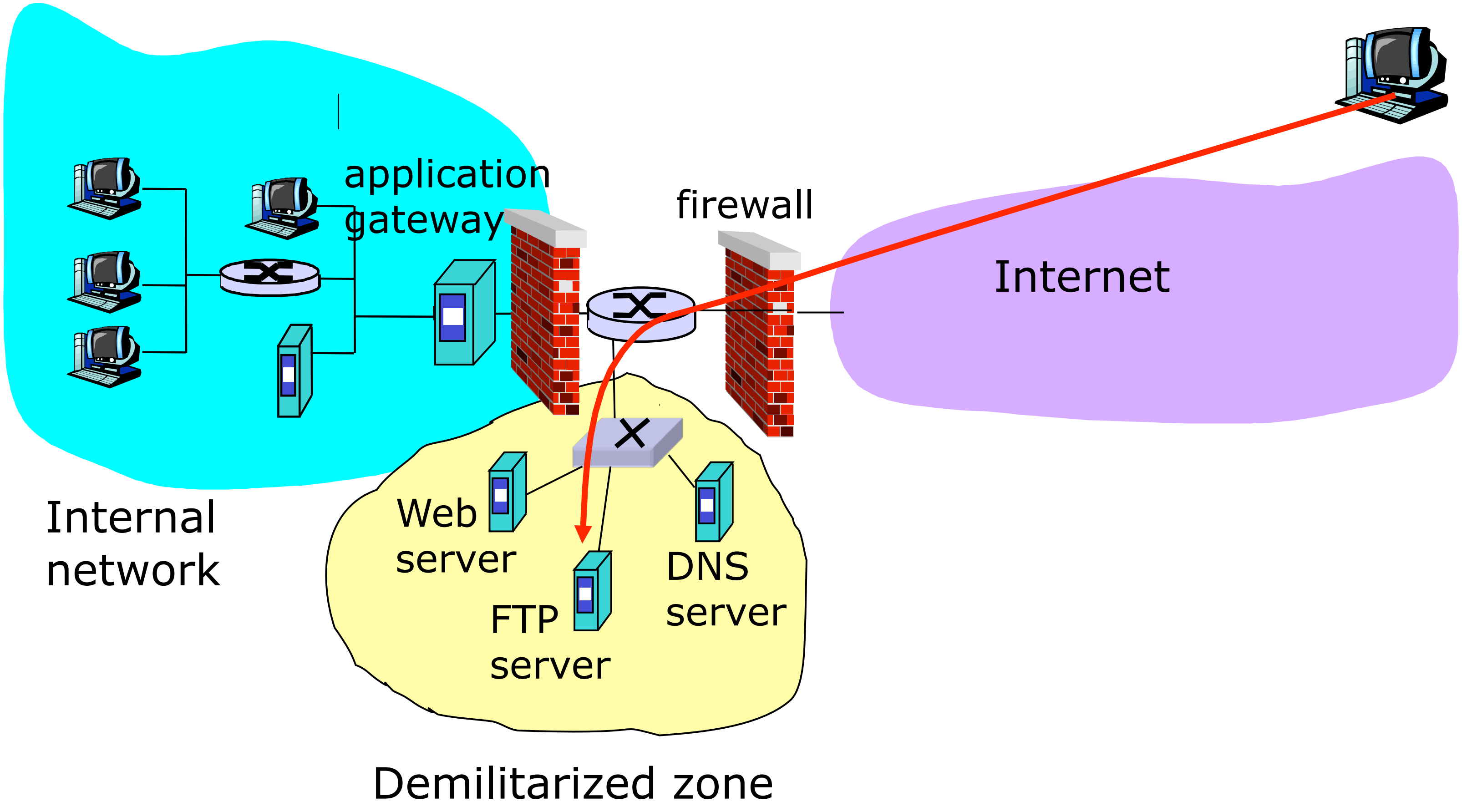
Stateful example

1) Packet arrives from outside: SA=37.96.87.123, SP=80, DA=222.22.1.7, DP=12699, SYN=0, ACK=1
2) Check filter table ➜ check stateful table

action	source address	dest address	proto	source port	dest port	flag bit	check conxion
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	×
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	×
deny	all	all	all	all	all	all	

3) Connection is listed in connection table ➜ let packet through

Demarcation Zone (DMZ)


application
gateway
firewall
Internet
Internal
network
Web
server
FTP
server
DNS
server
Demilitarized zone