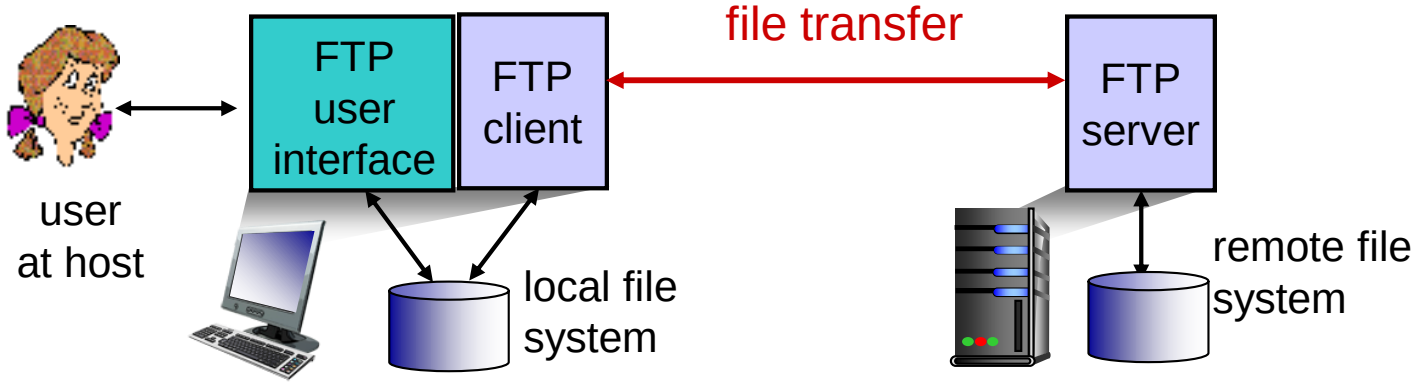
Chapter 2
Application Layer

Parte 2

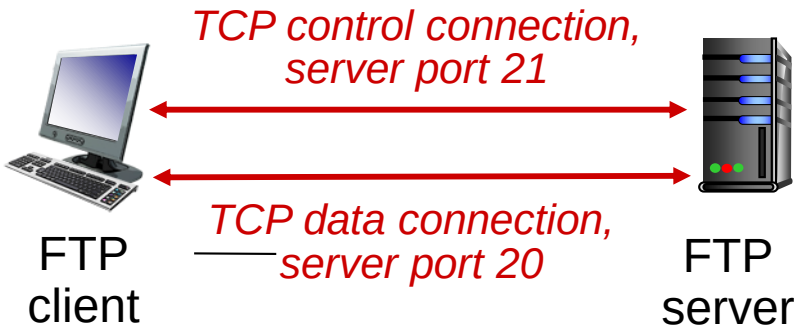
FTP: file transfer protocol



- transferimento file da/verso un host remoto
- modello client/server
  - *client:* parte che inzia la richiesta di trasferimento (da o verso l'host remoto)
  - *server:* remote host
- ftp: RFC 959
- ftp server: in attesa sulla porta 21

FTP: connessione di controllo, commessione dati

- Il client FTP contatta il server FTP server alla porta 21, usando TCP
- Il client invia le credenziali per l'autorizzazione mediante la connessione di controllo
- Il client può richiedere lista file (remoti), inviare comandi sulla connectione di controllo
- Quando il server riceve un comanda di trasferimento file, il server apre, verso il client, una seconda connessione TCP, la connessione dati
- Dopo il trasferimento del file, il server chiude la connessione dati



- Il client apre una connessione dati per ogni trasferimento file
- Sulla connessione di controllo, in alcune situazioni,. viene usato il meccanismo delle comunicazioni urgent *(out of band)*
- Il server FTP mantiene lo stato stato: directory corrente, l'autenticazione

FTP: commandi e risposte

alcuni comandi: (spediti come testo sulla connessione di controllo)

- **`USER`** ***`username`***
- **`PASS`** ***`password`***
- **`LIST`** per ottenere la lista dei file della directory corrente
- **`RETR filename`** per ottenere un il file
- **`STOR filename`** per memorizzare il file sull'host remoto

Codici di stato (simile ad HTTP)

- **`331 Username OK, password required`**
- **`125 data connection already open; transfer starting`**
- **`425 Can't open data connection`**
- **`452 Error writing file`**

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

DNS: domain name system

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., www.yahoo.com - used by humans

Q: how to map between IP address and name, and vice versa ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS: services, structure

DNS services

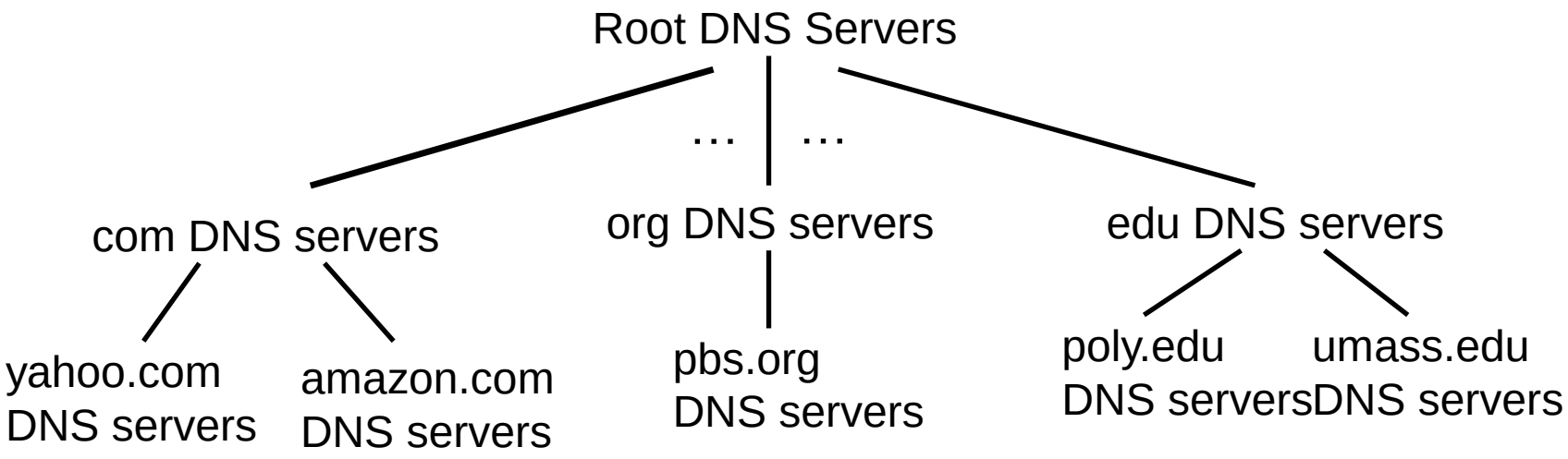
- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: doesn't scale!

DNS: a distributed, hierarchical database

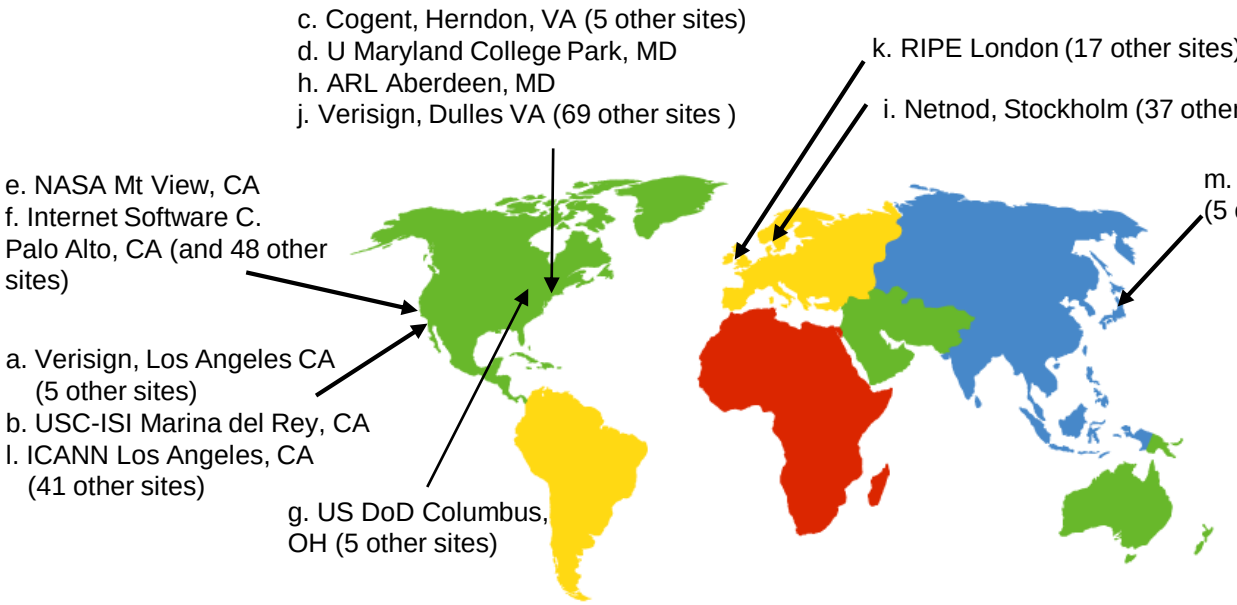


client wants IP for www.amazon.com; 1st approximation:

- client queries root server to find com DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

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
  - gets mapping
  - returns mapping to local name server



13 logical root name "servers" worldwide

•each "server" replicated many times

TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

authoritative DNS servers:

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

Local DNS name server

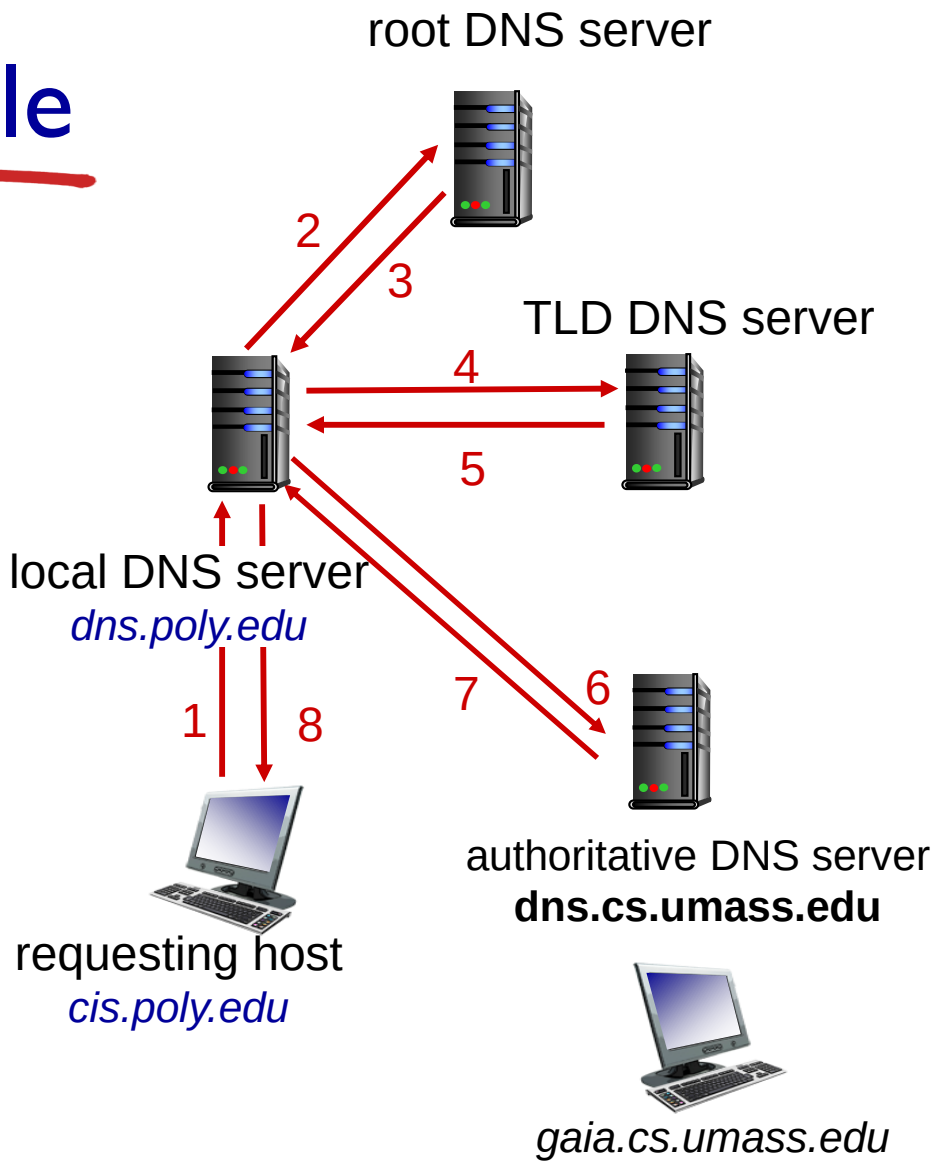
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:

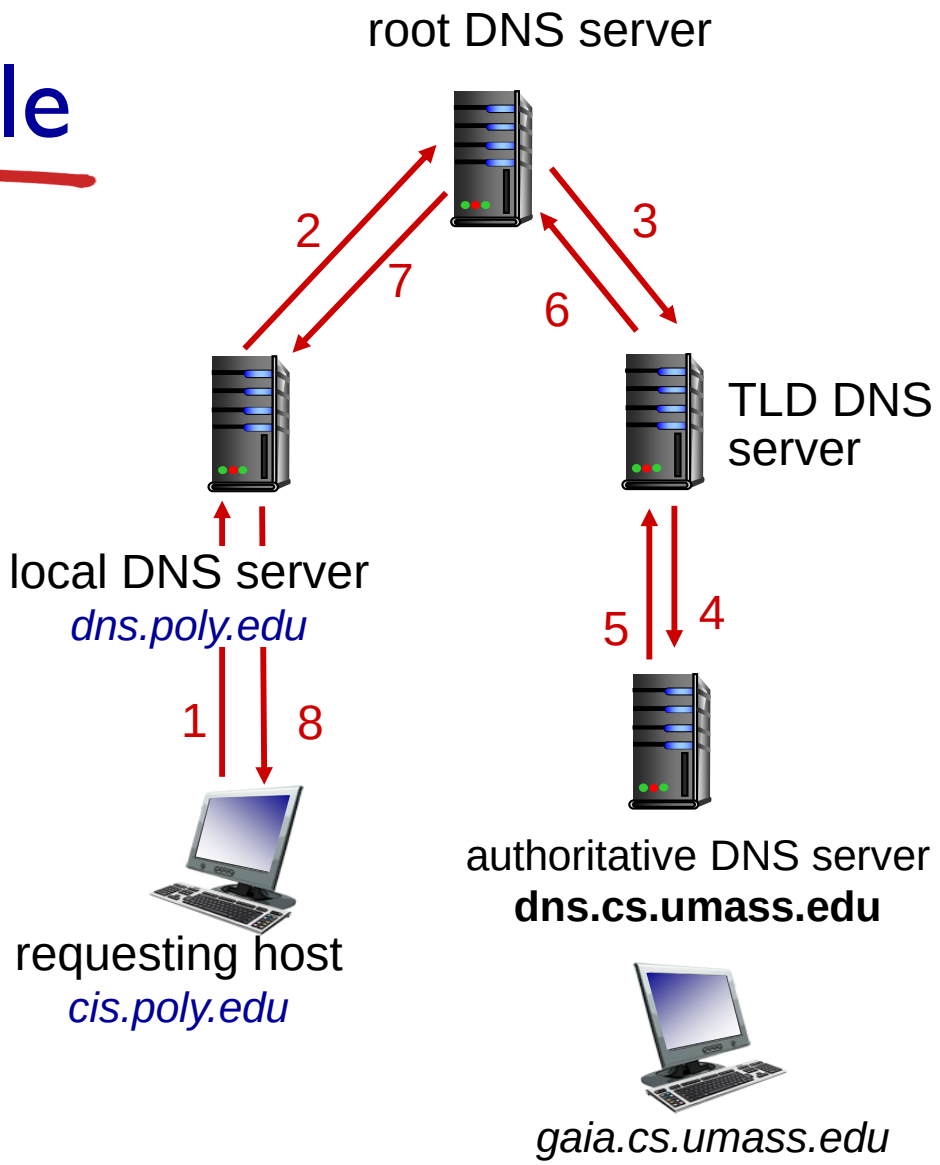
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS: caching, updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited
- cached entries may be *out-of-date* (best effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire
- update/notify mechanisms proposed IETF standard
  - RFC 2136

DNS records

DNS: distributed database 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 hostname of authoritative name server for this domain

type=CNAME

- **`name`** is alias name for some "canonical" (the real) name
- **`www.ibm.com`** is really **`servereast.backup2.ibm.com`**
- **`value`** is canonical name

type=MX

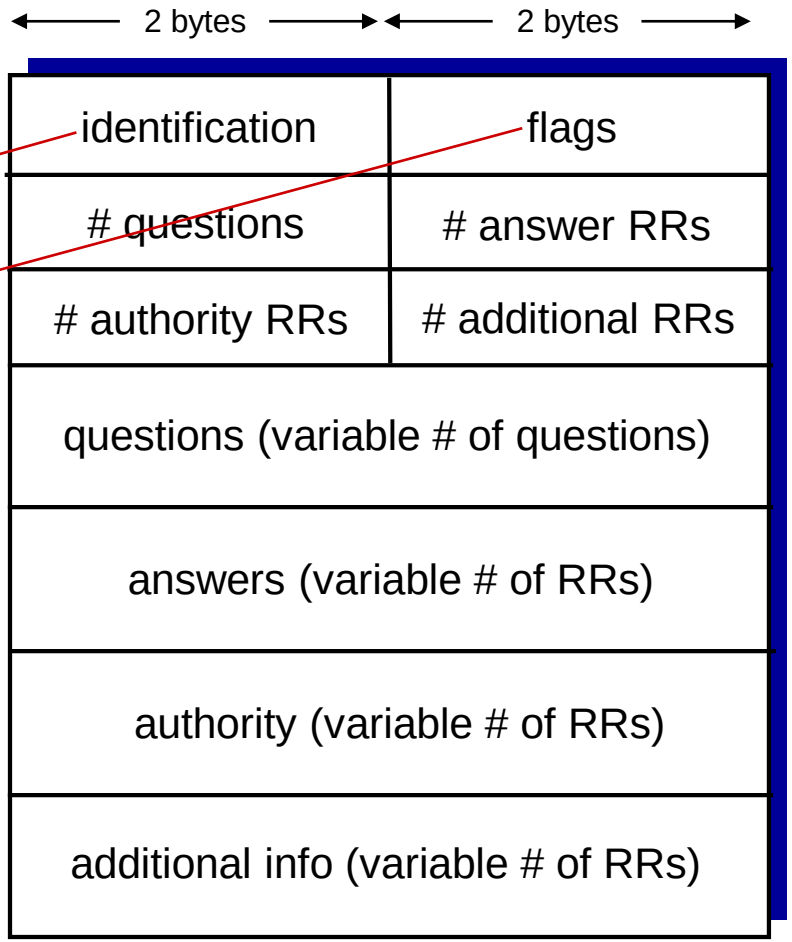
- **`value`** is name of mailserver associated with **`name`**

DNS protocol, messages

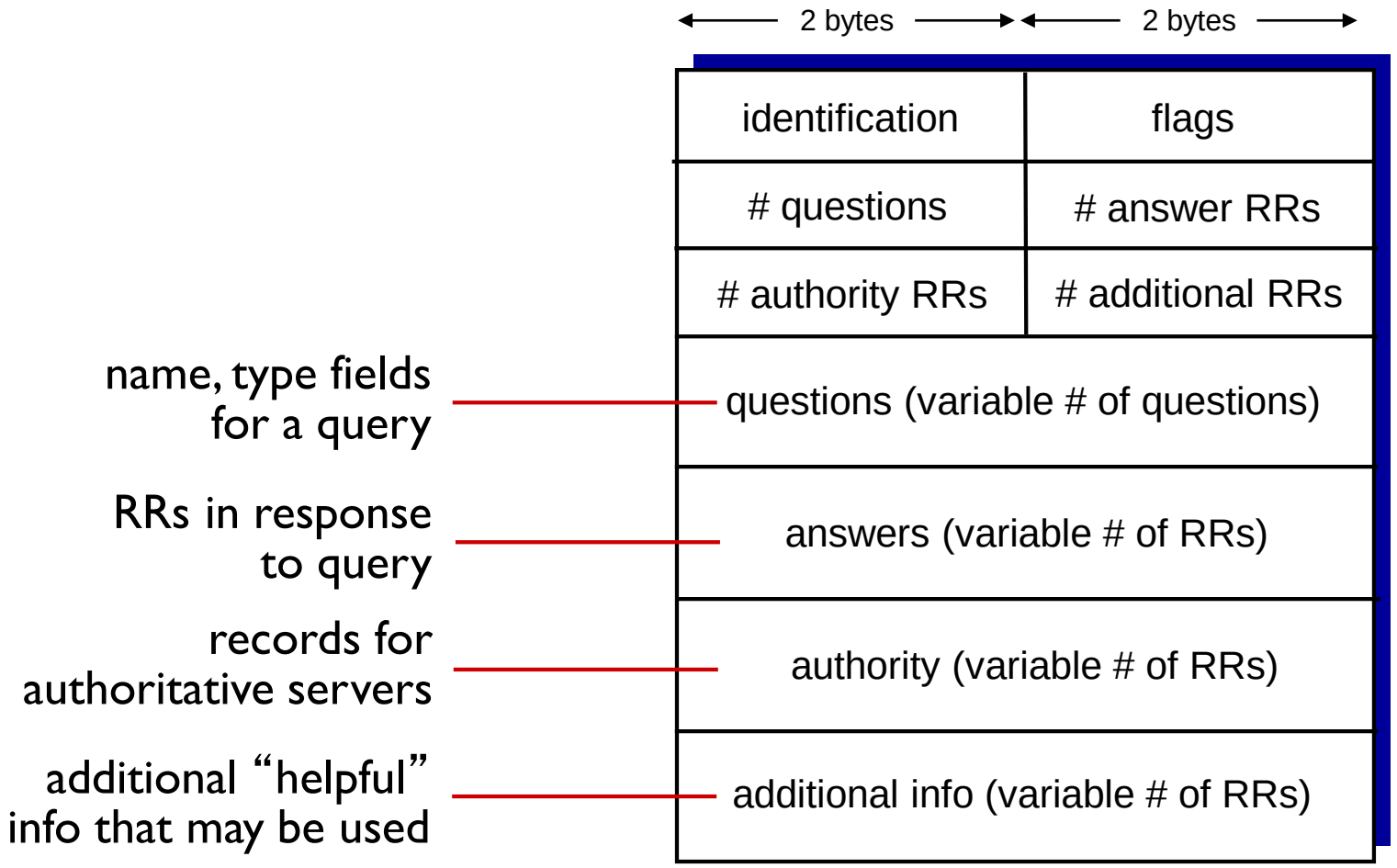
- *query* and *reply* messages, both with same *message format*

message header

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



DNS protocol, messages



Inserting records into DNS

- example: new startup "Network Utopia"
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into .com TLD server:
    `(networkutopia.com, dns1.networkutopia.com, NS)`
    `(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com

Attacking DNS

DDoS attacks

- bombard root servers with traffic
  - not successful to date
  - traffic filtering
  - local DNS servers cache IPs of TLD servers, allowing root server bypass
- bombard TLD servers
  - potentially more dangerous

redirect attacks

- man-in-middle
  - Intercept queries
- DNS poisoning
  - Send bogus relies to DNS server, which caches

exploit DNS for DDoS

- send queries with spoofed source address: target IP
- requires amplification

Mapping inverso

- Il mapping *da nome di dominio a resource record* è quello che più tipicamente viene richiesto
- In alcune situazioni sarebbe interessante poter effettuare la domanda inversa cioè sapere quali nomi possiedono un certo attributo con un ben determinato valore
- Dato che la risposta inversa non è affatto unica e che è molto complesso individuare il server che potrebbe dare la risposta, le *inverse query* **non sono implementate, anche se il formato del messaggio permette di esprimerle**
- Ma una inverse query è molto importante: quella da indirizzo IP a nome. Per questa inverse query è stata studiato un artificio che permette con una query diretta di avere la stessa informazione che avrei avuto con la query inversa
- Quindi è una particolare forma di query diretta …

Pointer query (I)

- Le pointer query chiedono un resource record particolare (PTR) ma soprattutto usano un nome molto particolare
- Nella **pointer query** la domanda deve contenere un indirizzo IP. Ma un indirizzo IP non ha la struttura di un nome di dominio, perché non comincia con un dominio giusto di top level!
- Allora si sono inventati un dominio DNS all'interno del quale sistemare gli indirizzi IP in forma di dotted quad, e ovviamente con i byte del netID "in alto" sull'albero!

Pointer query (II)

- A cosa può servire una pointer query?
  1. Ad una macchina diskless per sapere il proprio nome a partire dall'indirizzo IP ricevuto (ma BOOTP/DHCP lo possono fornire direttamente); non più usato per questo!
  2. I manager di rete assegnano nomi di dominio solo alle macchine di cui hanno fiducia: la presenza di un nome di dominio assegnato ad una macchina può quindi essere usata come semplice forma di verifica dell'affidabilità di una macchina. E questo è invece ancora molto usato

Pointer query (III)

- Come viene scritto un indirizzo IP in forma di nome di dominio? Si parte dalla dotted quad dell'indirizzo, che è già in forma a carattere

  `aaa.bbb.ccc.ddd`

- Il nome corrispondente ad un indirizzo IP è collocato nel dominio

  `in-addr.arpa.`

- e poi gli ottetti della dotted quad sono rovesciati perché gli ottetti più importanti sono quelli a sinistra e quindi devono essere posizionati in alto nell'albero

  `ddd.ccc.bbb.aaa.in-addr.arpa.`

Pointer query (IV)

- Il name server locale certamente non ha autorità né per `arpa` né per `in-addr.arpa`, e quindi deve contattare i root server
- Non si può demandare ad un unico server il compito del mapping inverso. Per ogni rete IP deve essere registrato l'indirizzo di un server DNS che è responsabile del mapping inverso se si vuole assegnare nomi di dominio a tali indirizzi

Resource Record (I)

- Vediamo finalmente quali resource record si possono trovare associati ad un nome...

Type	Meaning	Contents
A	Host Address	32-bit IP address
CNAME	Canonical Name	Canonical domain name for an alias
HINFO	CPU & OS	Name of CPU and operating system
MINFO	Mailbox info	Information about a mailbox or mail list
MX	Mail Exchanger	16-bit preference and name of host that acts as mail exchanger for the domain
NS	Name Server	Name of authoritative server for domain
PTR	Pointer	Domain name (like a symbolic link)
SOA	Start of Authority	Multiple fields that specify which parts of the naming hierarchy a server implements
TXT	Arbitrary text	Uninterpreted string of ASCII text

Resource Record (II)

- Il tipo di RR più comunemente usato è A (address) il cui valore è un indirizzo IP e quindi occupa 4 byte nella risposta.
- Quando una macchina ha più di una interfaccia oppure più indirizzi sono comunque mappati sullo stesso nome di dominio, viene ritornata una lista di indirizzi IP (non sempre nello stesso ordine.

Resource Record (III)

```
C:\>nslookup webmail.libero.it
Server:  ns.iunet.it
Address:  192.106.1.1
Risposta da un server non di fiducia:
Nome:    webmail.libero.it
Addresses: 193.70.192.64, 193.70.192.41,
           193.70.192.42, 193.70.192.61,
           193.70.192.62, 193.70.192.63
C:\>nslookup webmail.libero.it
Server:  ns.iunet.it
Address:  192.106.1.1
Risposta da un server non di fiducia:
Nome:    webmail.libero.it
Addresses: 193.70.192.61, 193.70.192.62,
           193.70.192.63, 193.70.192.64,
           193.70.192.41, 193.70.192.42
```

Resource Record (IV)

- Nel file di traccia, che trovate sul sito, potete vedere la traccia di una query su un altro DNS server, in altri momenti.
- Anche in questo caso due query successive riportano gli indirizzi IP in ordine diverso.
- Perché?

Resource Record (V)

- **CNAME:** Quando un nome di dominio ha il RR CNAME, vuol dire che il nome è in realtà un alias, cioè un nome alternativo. Il valore del RR CNAME è il nome di dominio a cui l'alias "punta"
- Gli alias sono usati **per dare il nome ad un servizio**, indipendentemente dal nome della macchina su sui è ospitato. Tipico è il nome di un webserver:

Resource Record (VI)

```
C:\>nslookup
Server predefinito:  ns.iunet.it
Address:  192.106.1.1
>  www.di.unito.it
Server:  ns.iunet.it
Address:  192.106.1.1
Risposta da un server non di fiducia:
Nome:    pianeta.di.unito.it
Address:  130.192.239.1
Aliases:  www.di.unito.it
>
```

- Questa è la ragione vera dell'esistenza dei CNAME, non il semplice desiderio di dare più nomi allo stesso indirizzo, che si può fare anche senza il CNAME
- **HINFO** e **MINFO** sono raramente usati

Resource Record (VII)

- **MX:** il valore di questo RR è un nome di dominio. Se un nome di dominio ha un RR MX, vuol dire che è usato come **nome di dominio di posta elettronica**: quello che segue il "@" negli indirizzi di posta!
- MX contiene il nome del server che gestisce le caselle postali di un dominio di posta. Ad es:….

```
> set q=mx
> di.unito.it
Server:  ns.iunet.it
Address:  192.106.1.1
Risposta da un server non di fiducia:
di.unito.it     MX preference = 10, mail
exchanger = pianeta.di.unito.it
di.unito.it     MX preference = 40, mail
exchanger = albert.unito.it
```

Resource Record (VIII)

- Ci sono due RR di tipo MX perché il dominio di posta `di.unito.it` è gestito dalla macchina `pianeta.di.unito.it`, ma in back-up anche dalla macchina `albert.unito.it`
- Chi vuole inviare posta a `sirovich@di.unito.it` deve contattare primariamente la macchina `pianeta.di.unito.it`, e dopo 10 tentativi falliti (preference 10) allora la macchina `albert.unito.it`.
- Il vostro client di mail consegna la posta da spedire al server del vostro ISP; è il server dell'ISP che cerca sul DNS l'MX record di `di.unito.it`, risolve il nome del server di posta e infine consegna la mail al server dell'utente di `di.unito.it`.

Resource Record (IX)

- **NS:** è un RR importantissimo perché informa nella risposta quale è il name server autorevole per la domanda. Quindi il client può contattare direttamente il server autorevole se non si fida o se ha dei problemi con la risposta non autorevole
- Lo troviamo spesso nella sezione AUTHORITY, ma può anche essere ricercato esplicitamente

```
> set q=ns
> di.unito.it
Server:  ns.iunet.it
Address:  192.106.1.1
Risposta da un server non di fiducia:di.unito.it
nameserver = albert.unito.it
di.unito.it     nameserver = amleto.di.unito.it
di.unito.it     nameserver = pianeta.di.unito.it
albert.unito.it internet address = 130.192.119.1
amleto.di.unito.it  internet address = 130.192.239.30
pianeta.di.unito.it internet address = 130.192.239.1
```

- Confrontate le risposte che ottenete quando domandate un NS record ad un server autorevole per il RR…

Resource Record (X)

- **PTR:** è il RR che contiene il nome della macchina che ha l'indirizzo IP specificato nel nome di dominio della domanda. Es:

```
> set q=ptr
>  130.192.239.1
Server:  ns.iunet.it
Address:  192.106.1.1
Risposta da un server non di fiducia:
1.239.192.130.in-addr.arpa name = pianeta.di.unito.it
239.192.130.in-addr.arpa nameserver =
  pianeta.di.unito.it
239.192.130.in-addr.arpa nameserver = itaca.di.unito.it
pianeta.di.unito.it internet address = 130.192.239.1
itaca.di.unito.it   internet address = 130.192.239.182
```

- Il DNS vi risponde con i server autorevoli per dare la risposta. Se rifacciamo la domanda a pianeta …

Resource Record (XI)

- **SOA:** ritorna le informazioni riguardo la parte della gerarchia dei nomi a cui un nome di dominio appartiene. Es:

```
> set q=soa
> di.unito.it
Server:  ns.iunet.it
Address:  192.106.1.1
Risposta da un server di fiducia:
di.unito.it primary name server =
   pianeta.di.unito.it
   responsible mail addr =
   root.pianeta.di.unito.it
  serial  = 2001050901
  refresh = 86400 (1 day)
  retry   = 3600 (1 hour)
  expire  = 302400 (3 days 12 hours)
  default TTL = 43200 (12 hours)
di.unito.it     nameserver = albert.unito.it
di.unito.it     nameserver = amleto.di.unito.it
di.unito.it     nameserver = pianeta.di.unito.it
amleto.di.unito.it internet address = 130.192.239.30
pianeta.di.unito.it internet address = 130.192.239.1
```

Resource Record (XII)

- Il server risponde che la risposta è autorevole perché ha appena contattato uno dei server autorevoli per il dominio di.unito.it..
- Se domandiamo il SOA di pianeta.di.unito.it., che non è il nome di dominio di una SOA …
- Lo stesso server non da risposta…ma da le informazioni richieste nella sezione Authorities, non fornendo neanche l'indirizzo IP di pianeta nelle Additional Informations!!!

Server primari e secondari (I)

- Se il server autorevole per un dominio è down, nessuno può ottenere informazioni: occorre ***evitare single points of failure***
- Quando un dominio è molto usato perché contiene nomi molto usati, il server DNS autorevole diventa molto carico e può essere un collo di bottiglia: occorre ***dividere il carico fra numerosi server DNS***
- Occorre risolvere ambedue i problemi!
- **Server secondari:** sono server DNS che prendono le informazioni su una intera *Zona di Autorità* per cui sono server secondari da un server primario
- Una operazione di protocollo detta **zone transfer** permette il trasferimento di tutte le informazioni su cui il server primario è autorevole. Naturalmente non può essere usato UDP!

Server primari e secondari (II)

- Quanti server secondari? Dove?
- Per avere affidabilità devono essere su reti IP diverse e naturalmente non dipendere da risorse comuni, ad es. l'alimentazione. Quindi tendenzialmente "lontani" uno dall'altro
- Come si fa a dividere il traffico fra i vari server? Come fanno i client a sapere che esistono?
- I client locali sono configurati ad avere conoscenza dei server locali, primari e secondari e in configurazione sono distribuiti in round-robin. Un server alternativo viene contattato solo se quelli precedenti nella lista non rispondono
- I server remoti come fanno a sapere che esistono server alternativi? Perché sono descritti nel DNS stesso con il RR di tipo NS.

Server primari e secondari (III)

```
> set q=ns
> di.unito.it
Server:  ns.iunet.it
Address:  192.106.1.1
Risposta da un server non di fiducia:
di.unito.it     nameserver = albert.unito.it
di.unito.it     nameserver = amleto.di.unito.it
di.unito.it     nameserver = pianeta.di.unito.it
albert.unito.it internet address = 130.192.119.1
amleto.di.unito.it      internet address =
  130.192.239.30
pianeta.di.unito.it     internet address =
  130.192.239.1
>
```

Server primari e secondari (IV)

- Notate le informazioni addizionali che permettono di contattare direttamente i server autorevoli
- Questi RR permettono anche di acquisire le informazioni autorevoli se non ci si accontenta di quelle in cache. Ma la ricerca è a carico del client
- Si **deve** consultare le tracce di uso del DNS che ci sono sul server della didattica per capire come fa un client a venire a conoscere i DNS autorevoli di una SOA!

UDP o TCP?

- DNS può essere usato sulla porta 53 di UDP e di TCP; quindi può usare ambedue i trasporti
- Il resolver in genere usa UDP, ma se la risposta indica che c'è stato un troncamento, il resolver riprova la domanda questa volta usando TCP
- Zone transfer vengono richieste solo usando TCP perché le riposte sicuramente eccedono il limite specificato nell'uso di UDP
- Quando il resolver e i server usano UDP, dato che lavorano su rete geografica devono prevedere la perdita di pacchetti, ritrasmettendo dopo timeout e abbandonando dopo un certo numero di ritrasmissioni. Ma non hanno la sofisticazione di TCP per adattarsi a ritardi variabili e a congestioni della rete!

Come ottenere autorità per un sottodominio (I)

- Per avere autorità su un dominio di secondo livello, occorre sottostare ad alcune condizioni:
  1. Essere disposti a gestire un server DNS che soddisfa gli standard Internet, o farselo gestire da qualcuno che è capace. Se lo volete gestire voi, dovete "metterlo in piedi" e poi l'autorità che vi concede il dominio farà una verifica compatibilità e correttezza prima di darvi l'autorizzazione
  2. Si deve dimostrare nell'uso che il server conosce l'indirizzo di almeno un server di root
  3. Si deve dimostrare che se si hanno sottodomini gestiti da altri server, il server parent conosce l'indirizzo dei server dei sottodomini
  4. Bisogna avere almeno un server secondario, ragionevolmente separato dal server primario (in pratica questo vuol dire spesso appoggiarsi ad un ISP)

Come ottenere autorità per un sottodominio (II)

- I server DNS non sono affatto semplici da sviluppare. Devono:
  1. essere capaci di gestire più domini, anche indipendenti
  2. essere capaci di gestire numerose operazioni in parallelo, per avere la necessaria performance, specialmente in occasione di richieste ricorsive, che hanno lunga durata perché richiedono l'interazione con altri server
  3. supportare la replica via zone transfer
  4. gestire in modo efficiente cache anche molto grandi

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

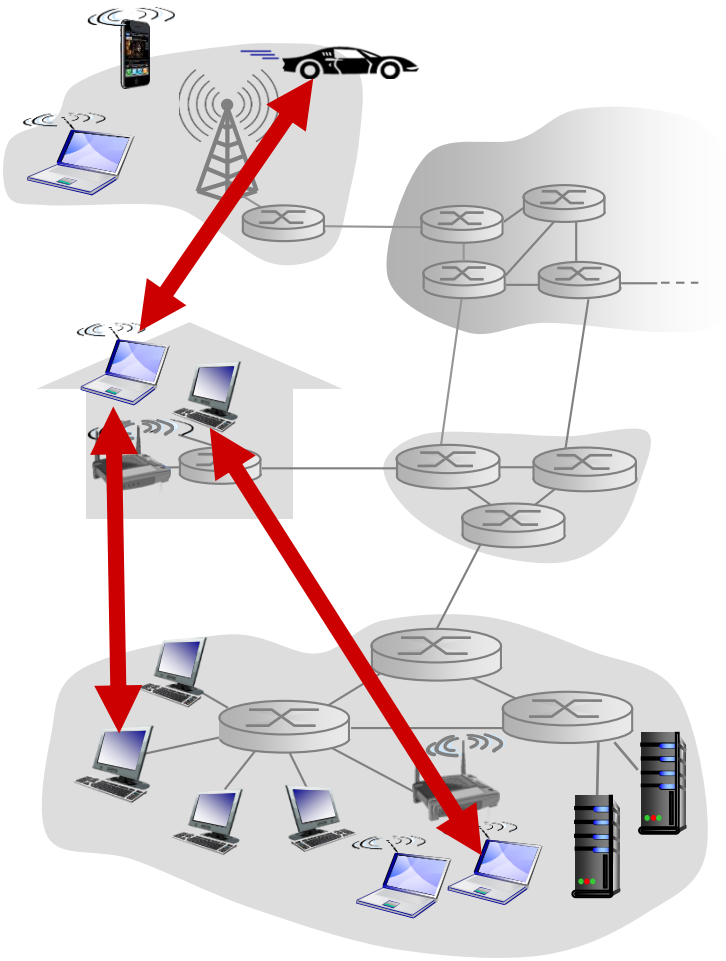
2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses



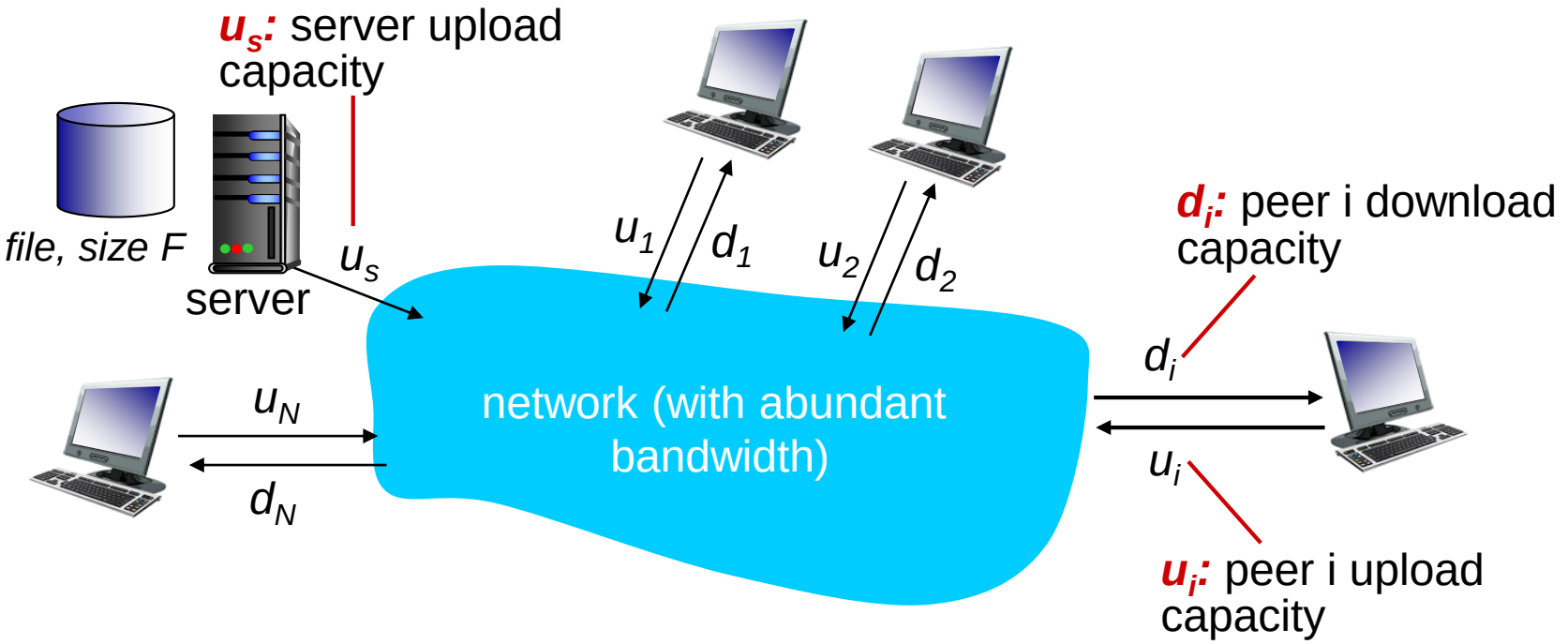
examples:

- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)

File distribution: client-server vs P2P

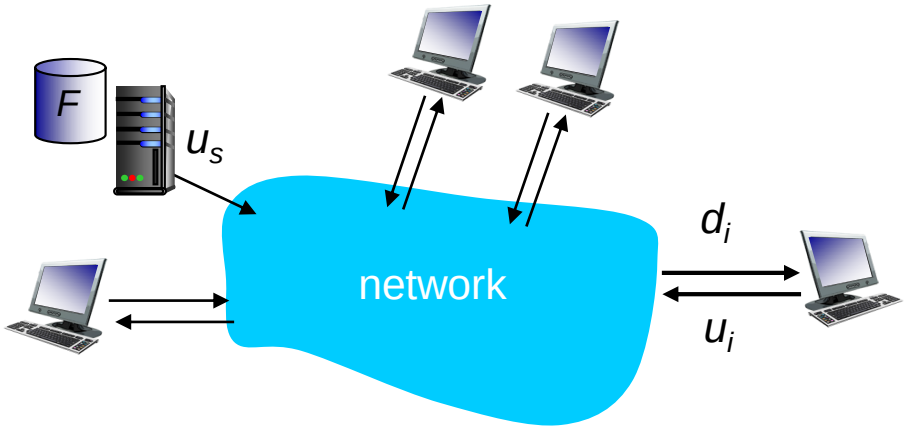
Question: how much time to distribute file (size *F*) from one server to *N* peers?

- peer upload/download capacity is limited resource



File distribution time: client-server

- *server transmission:* must sequentially send (upload) *N* file copies:
  - time to send one copy: F/u_s
  - time to send *N* copies: NF/u_s
- *client:* each client must download file copy
  - d_{min} = min client download rate
  - min client download time: F/d_{min}



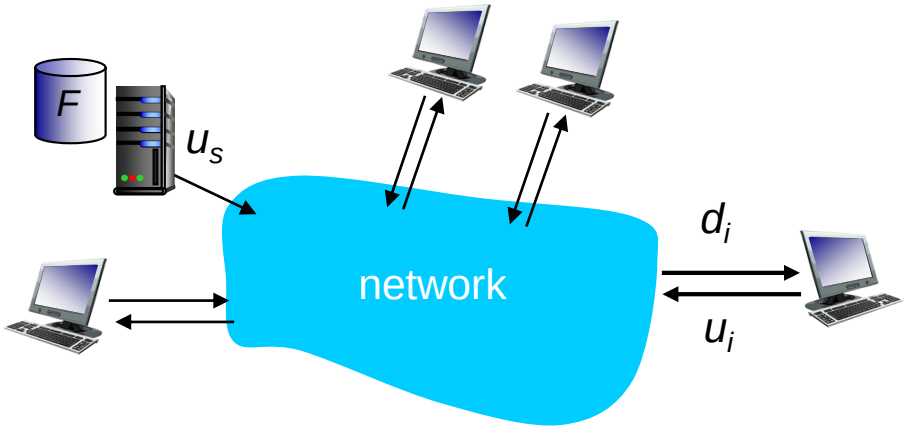
time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$$

increases linearly in N

File distribution time: P2P

- *server transmission:* must upload at least one copy
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - min client download time: F/d_{min}
- *clients:* as aggregate must download NF bits
  - max upload rate (limiting max download rate) is $u_s + \Sigma u_i$



time to distribute F to N clients using P2P approach

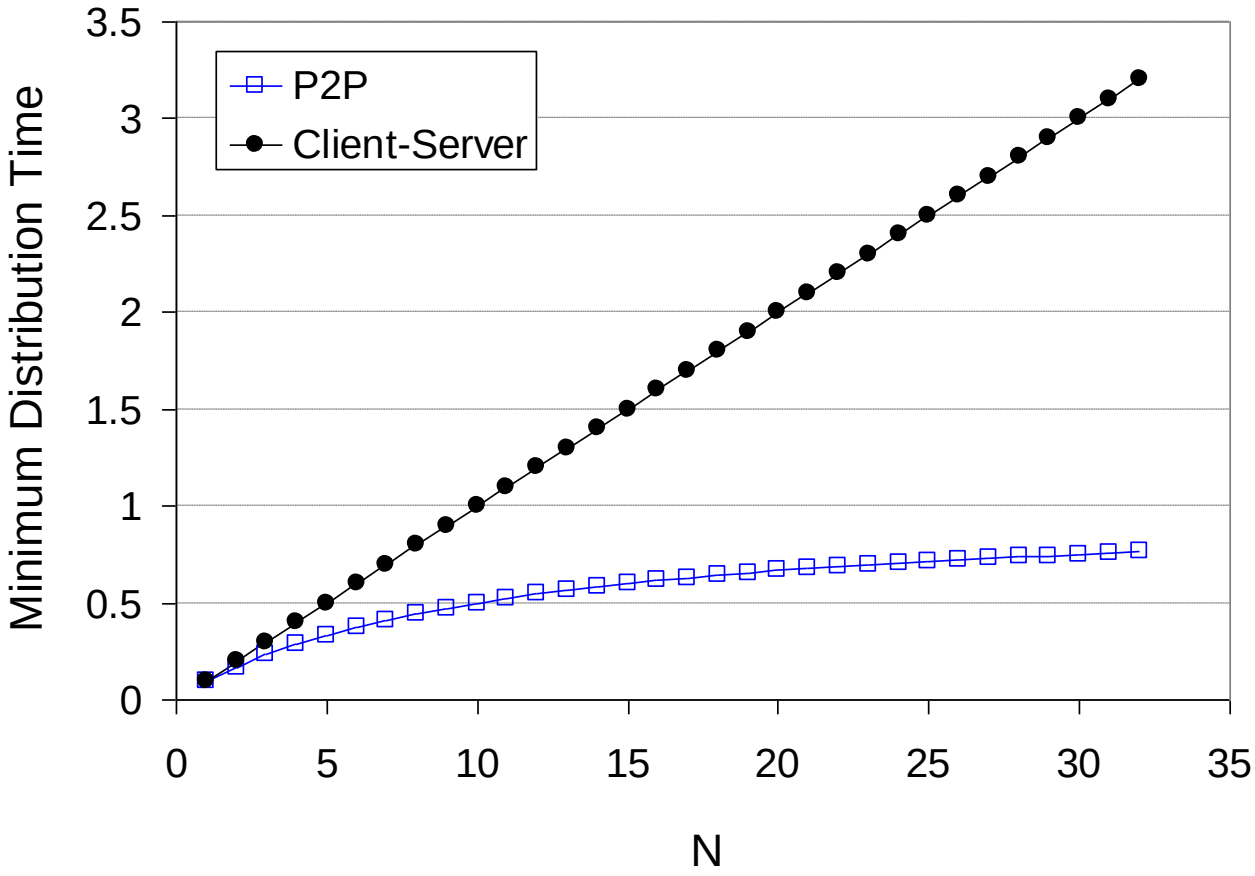
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in N …

… but so does this, as each peer brings service capacity

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$

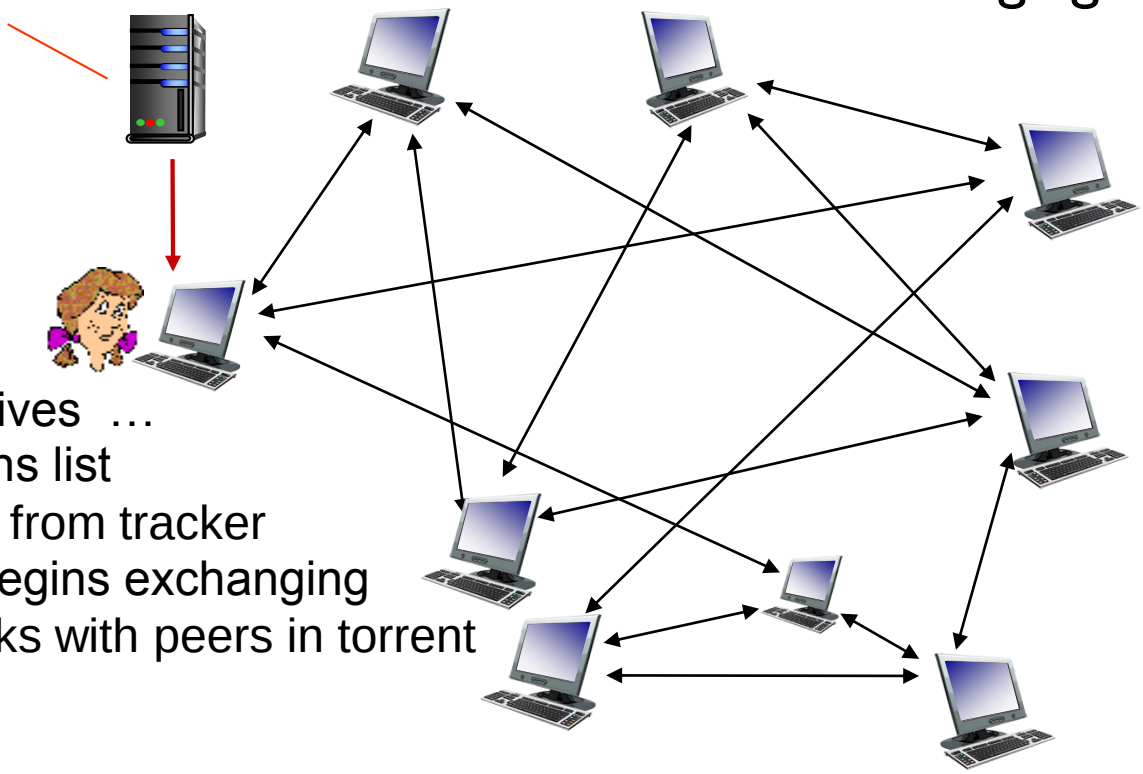


P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

tracker: tracks peers participating in torrent

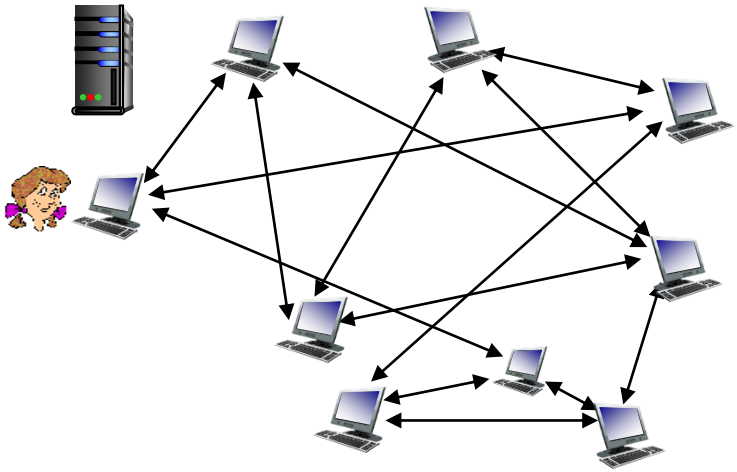
torrent: group of peers exchanging chunks of a file



Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

P2P file distribution: BitTorrent

- peer joining torrent:
  - has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")



- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- *churn:* peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

BitTorrent: requesting, sending file chunks

requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

sending chunks: tit-for-tat

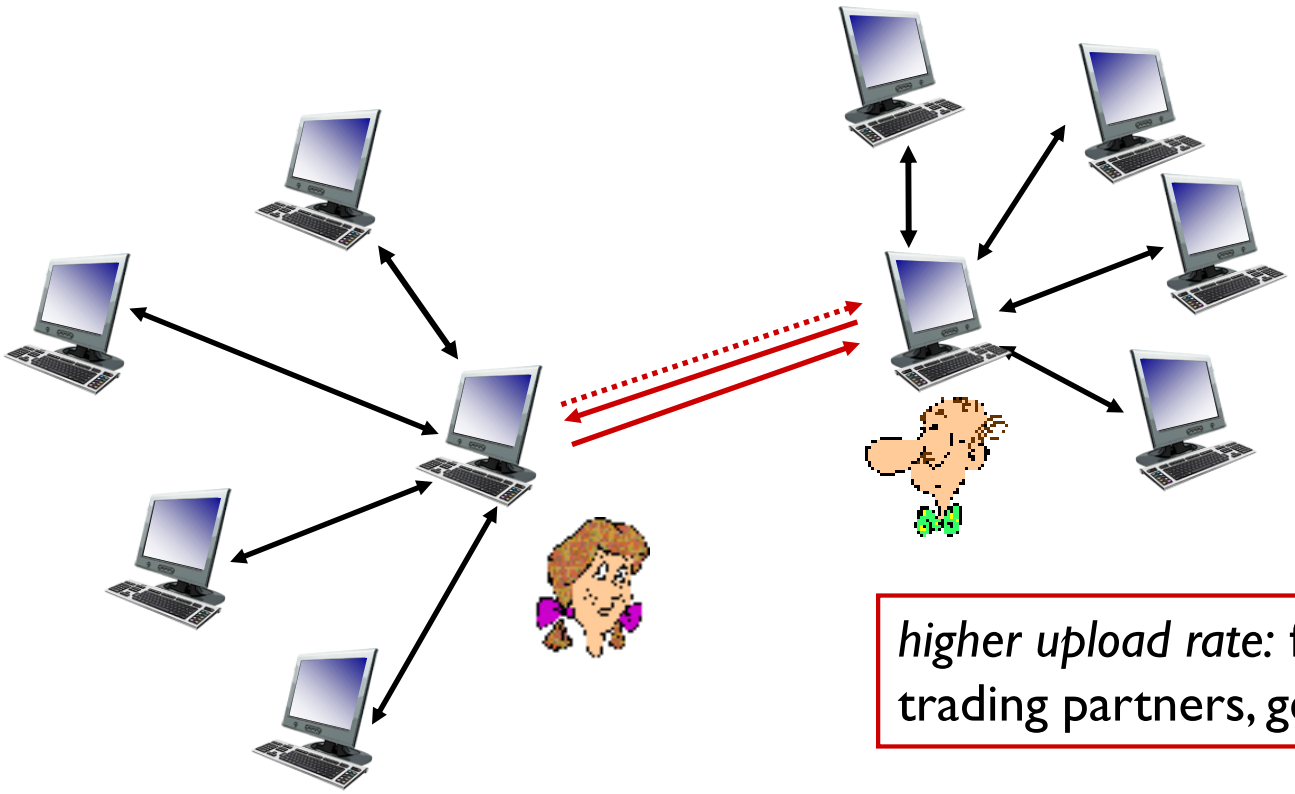
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster !

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users
- challenge: scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- *solution:* distributed, application-level infrastructure

Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple) and number of repeated values (*N)

frame *i*

frame *i+1*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Multimedia: video

- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values (*N)

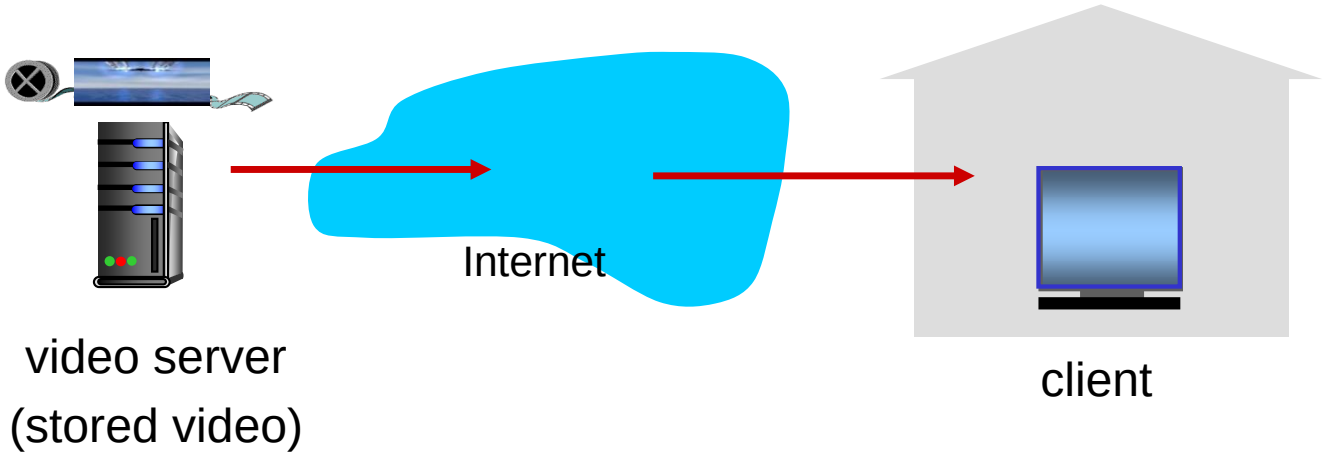
frame *i*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

frame *i+1*

Streaming stored video:

simple scenario:



Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

- *DASH: Dynamic, Adaptive Streaming over HTTP*
- *"intelligence"* at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur)
  - *what encoding rate* to request (higher quality when more bandwidth available)
  - *where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- *option 1:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

….quite simply: this solution *doesn't scale*

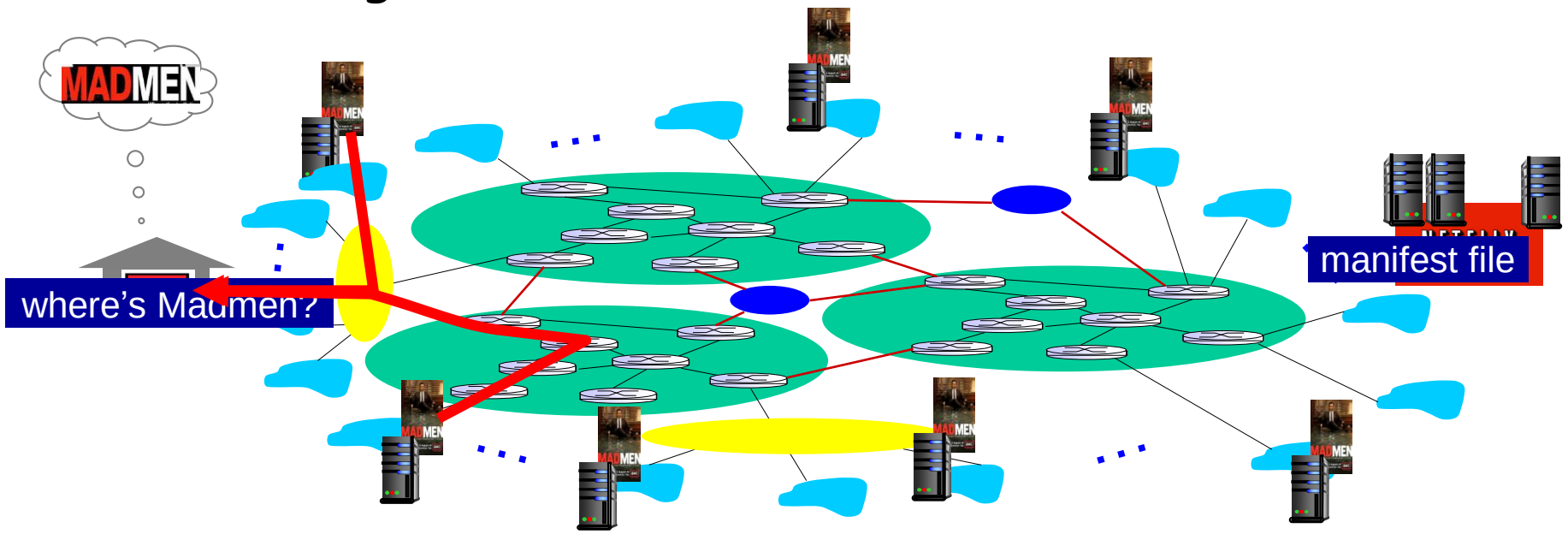
Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- *option 2:* store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - used by Akamai, 1700 locations
  - *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

Content Distribution Networks (CDNs)

- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - may choose different copy if network path congested



Content Distribution Networks (CDNs)

OTT challenges: coping with a congested Internet

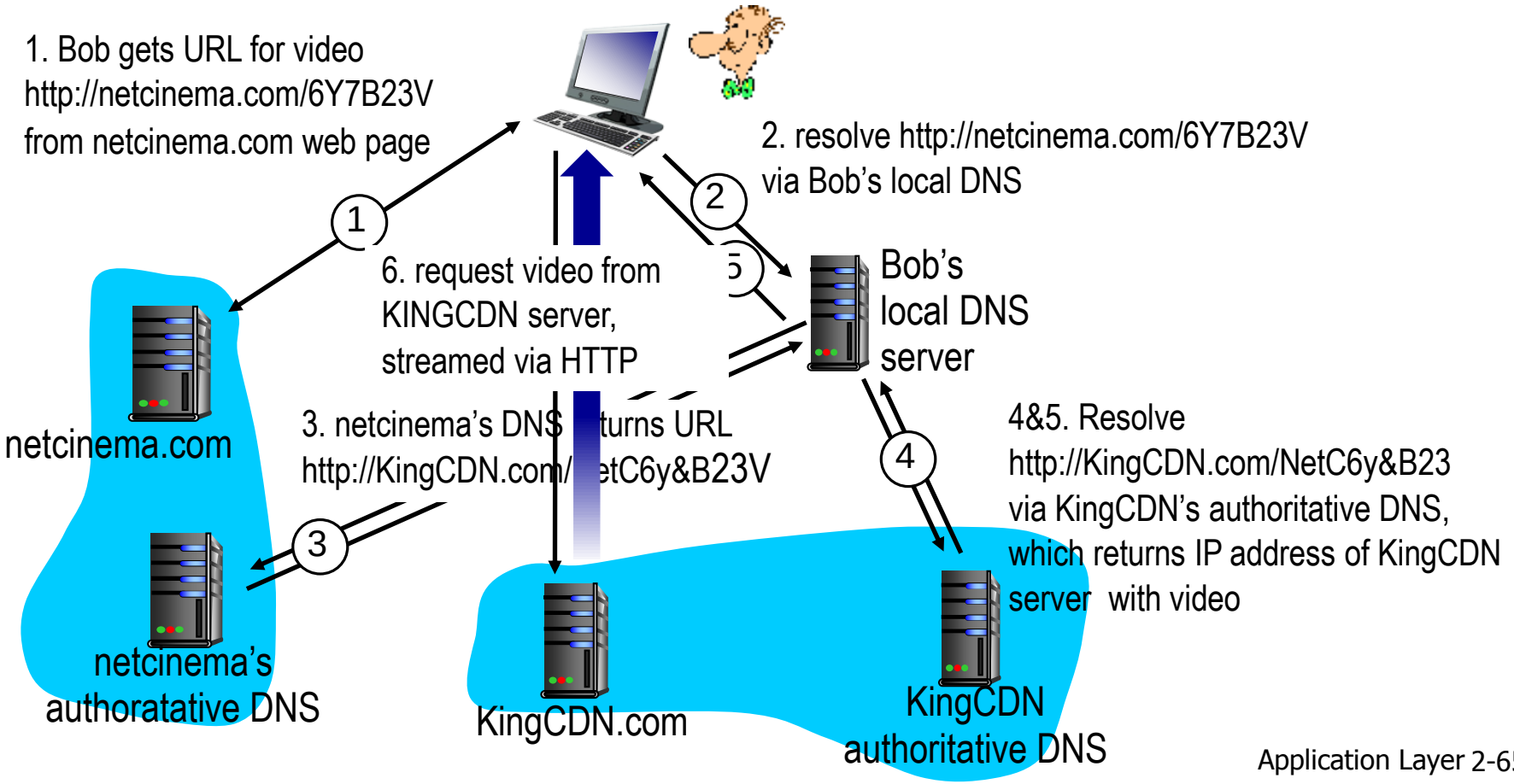
- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

more .. in chapter 7

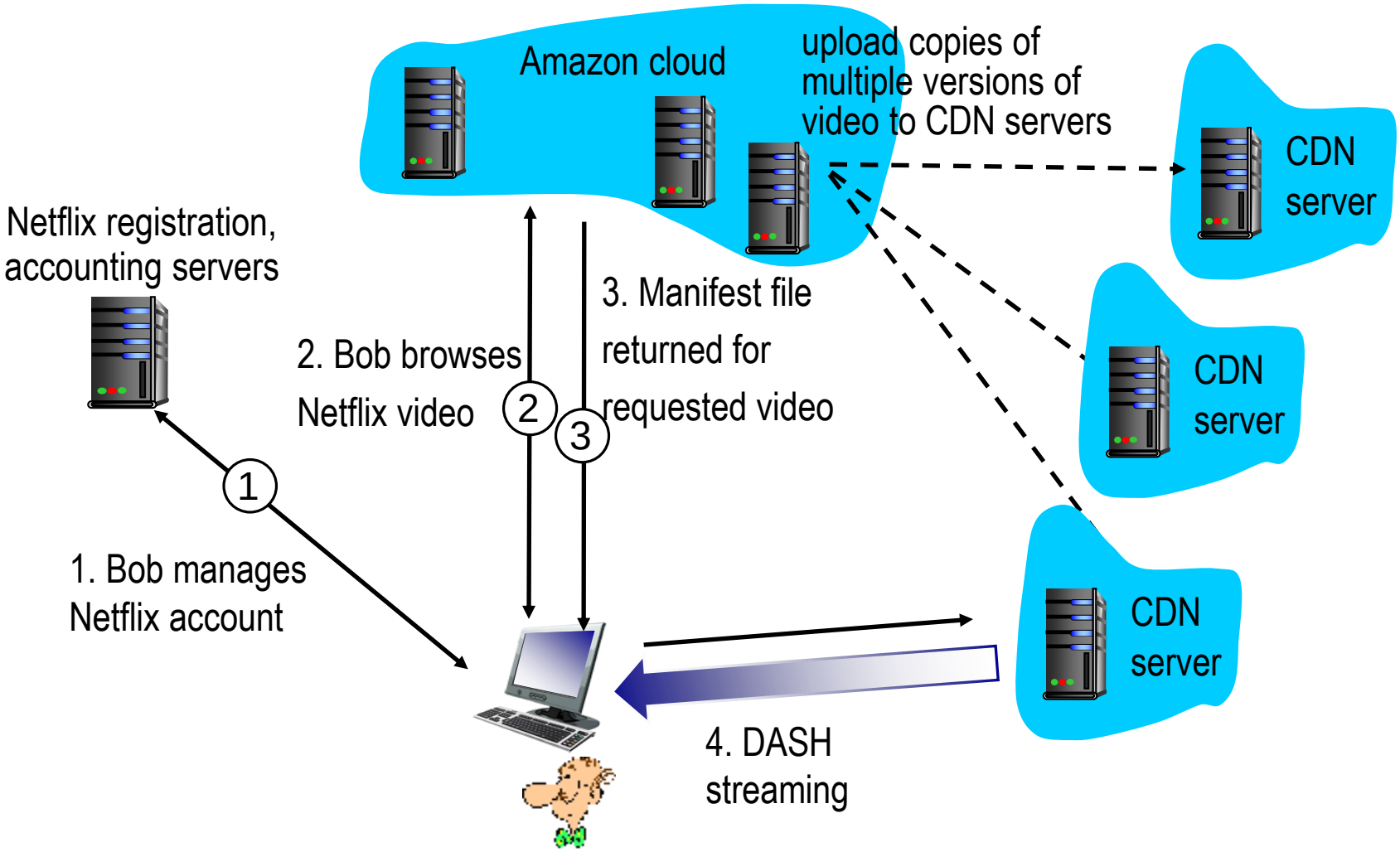
CDN content access: a closer look

Bob (client) requests video http://netcinema.com/6Y7B

- video stored in CDN at http://KingCDN.com/NetC6y&B23V



Case study: Netflix



Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP