Application Layer: Overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- video streaming and content distribution networks
- 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 (data + destination header)
- "name", e.g., cs.umass.edu - used by humans

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

Domain Name System (DNS):

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol:* hosts, DNS 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(real), alias names
- mail server aliasing
- load distribution
  - **replicated Web servers:** many IP addresses correspond to one name
  - DNS distributes millions of users across multiple servers by giving different IPs for the same domain name

Q: Why not centralize DNS?

- single point of failure
- traffic volume
- difficult to update and manage
- servers would be too far away from users

A: doesn't scale!

- Distributed e.g
  - Comcast DNS servers alone: 600B DNS queries/day
  - Akamai DNS servers alone: 2.2T DNS queries/day

Thinking about the DNS

humongous distributed database:

- ~ billion records, each simple

handles many *trillions* of queries/day:

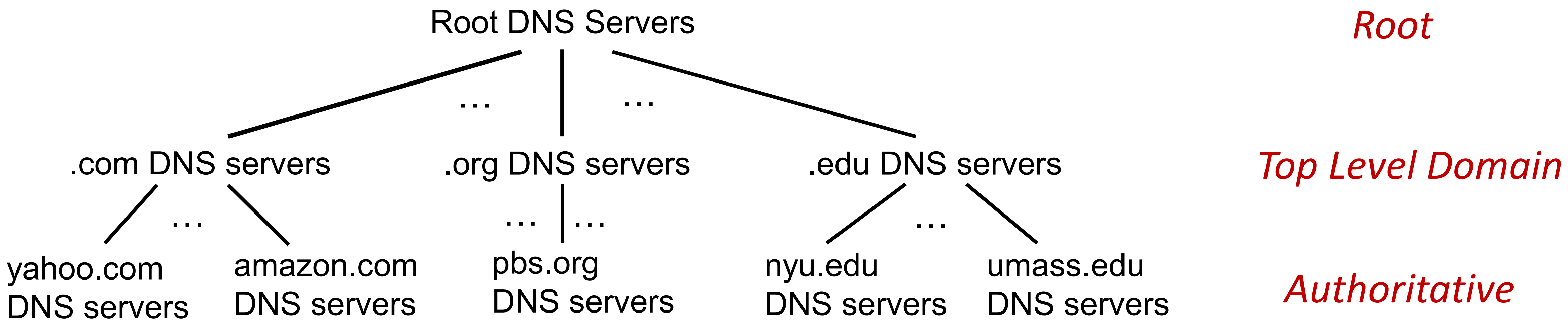
- *many* more reads than writes
- *performance matters:* almost every Internet transaction interacts with DNS - msecs count!

organizationally, physically decentralized:

- millions of different organizations responsible for their records

“bulletproof”: reliability, security

DNS: a distributed, hierarchical database

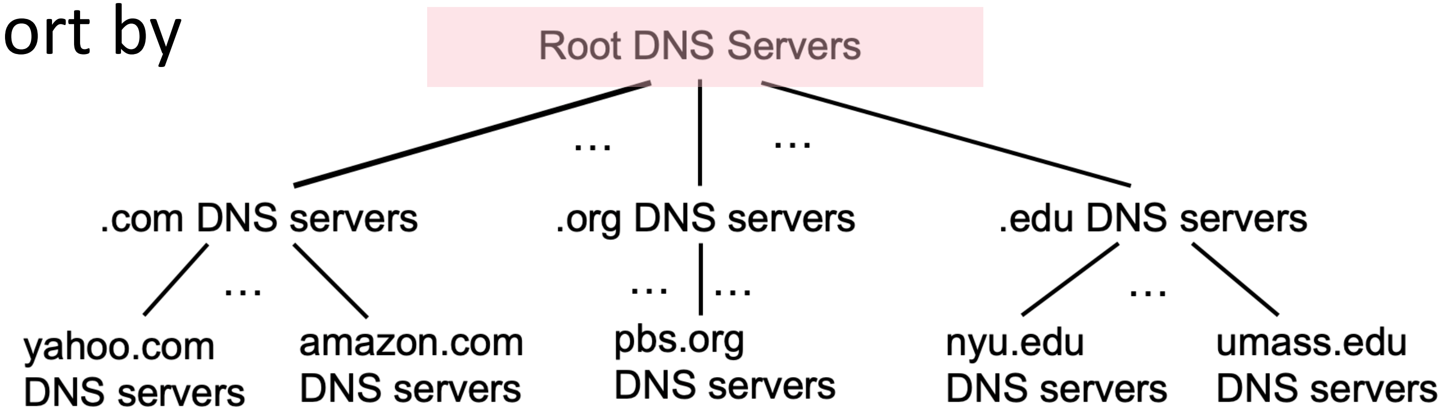


Client wants IP address for www.amazon.com; 1st approximation:

- 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

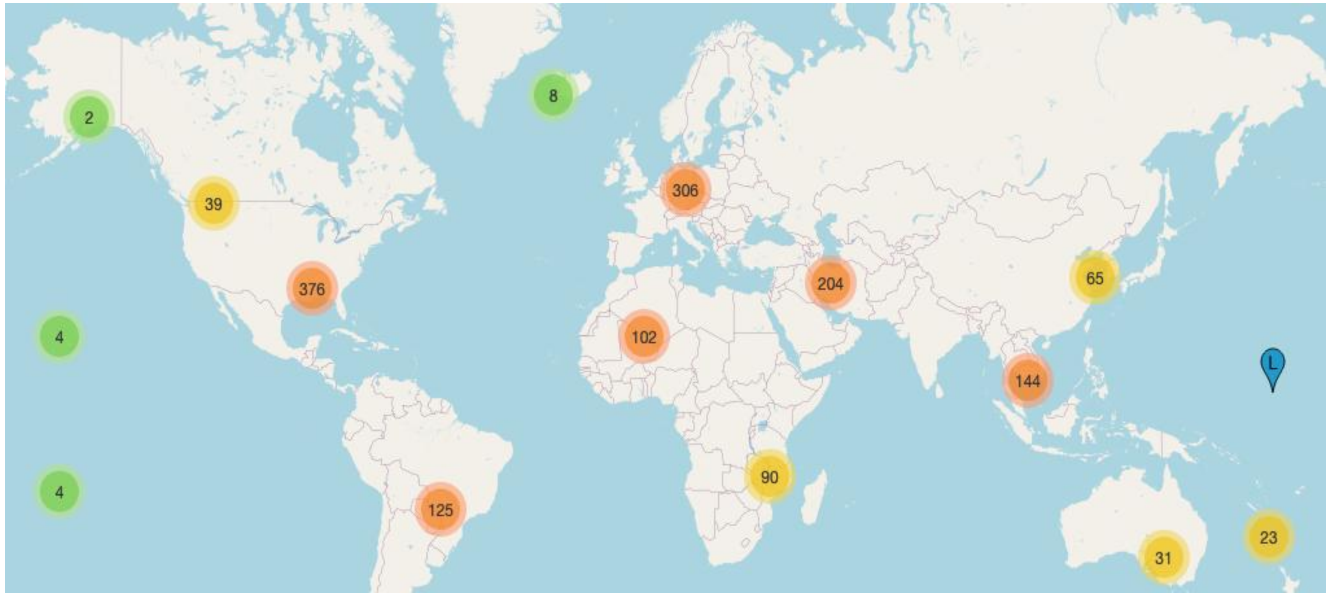
- official, contact-of-last-resort by name servers that can not resolve name



DNS: root name servers

- official, contact-of-last-resort by name servers that can not resolve name
- *incredibly important* Internet function
  - Internet couldn't function without it!
  - DNSSEC – provides security (authentication, message integrity)

- ICANN (Internet Corporation for Assigned Names and Numbers) manages root DNS domain

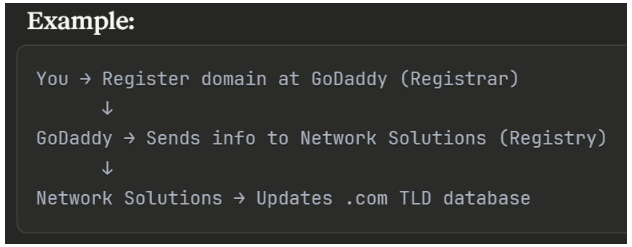


13 logical root name "servers" worldwide; each "server," replicated many times

Top-Level Domain, and authoritative servers

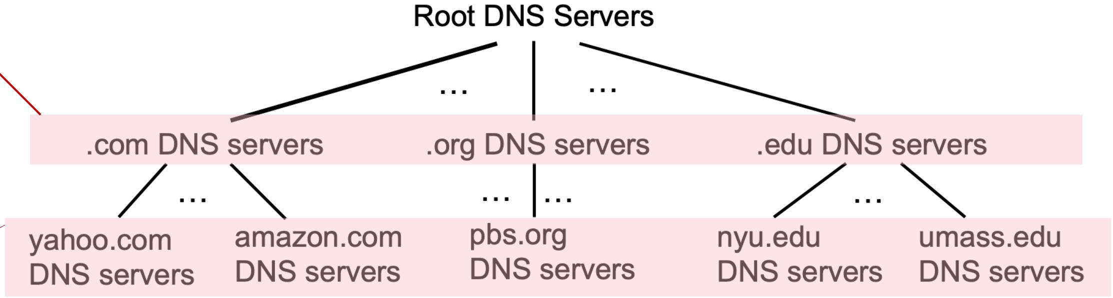
Top-Level Domain (TLD) servers:

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



Example:

```
You → Register domain at GoDaddy (Registrar)
      ↓
GoDaddy → Sends info to Network Solutions (Registry)
      ↓
Network Solutions → Updates .com TLD database
```



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 servers

- when host makes DNS query, it is sent to its *local* DNS server
  - Local DNS server returns reply, answering:
    - from its local cache of recent name-to-address translation pairs (possibly out of date!)
    - forwarding request into DNS hierarchy for resolution
  - each ISP has local DNS name server; to find yours:
    - MacOS: `% scutil --dns`
    - Windows: `> netsh interface ip show dns`

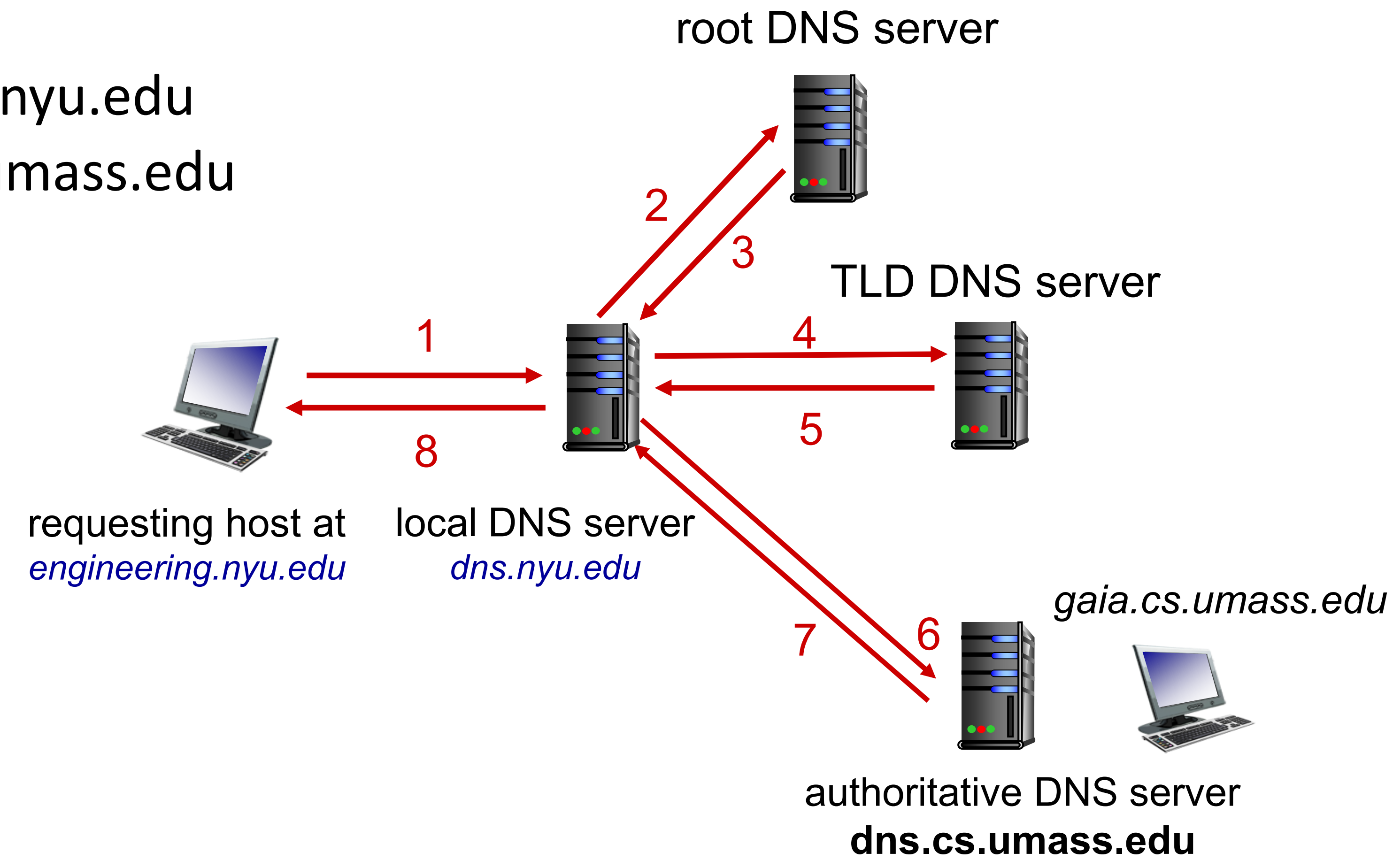
- local DNS server doesn't strictly belong to hierarchy

DNS name resolution: iterated query

Example: host at engineering.nyu.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"

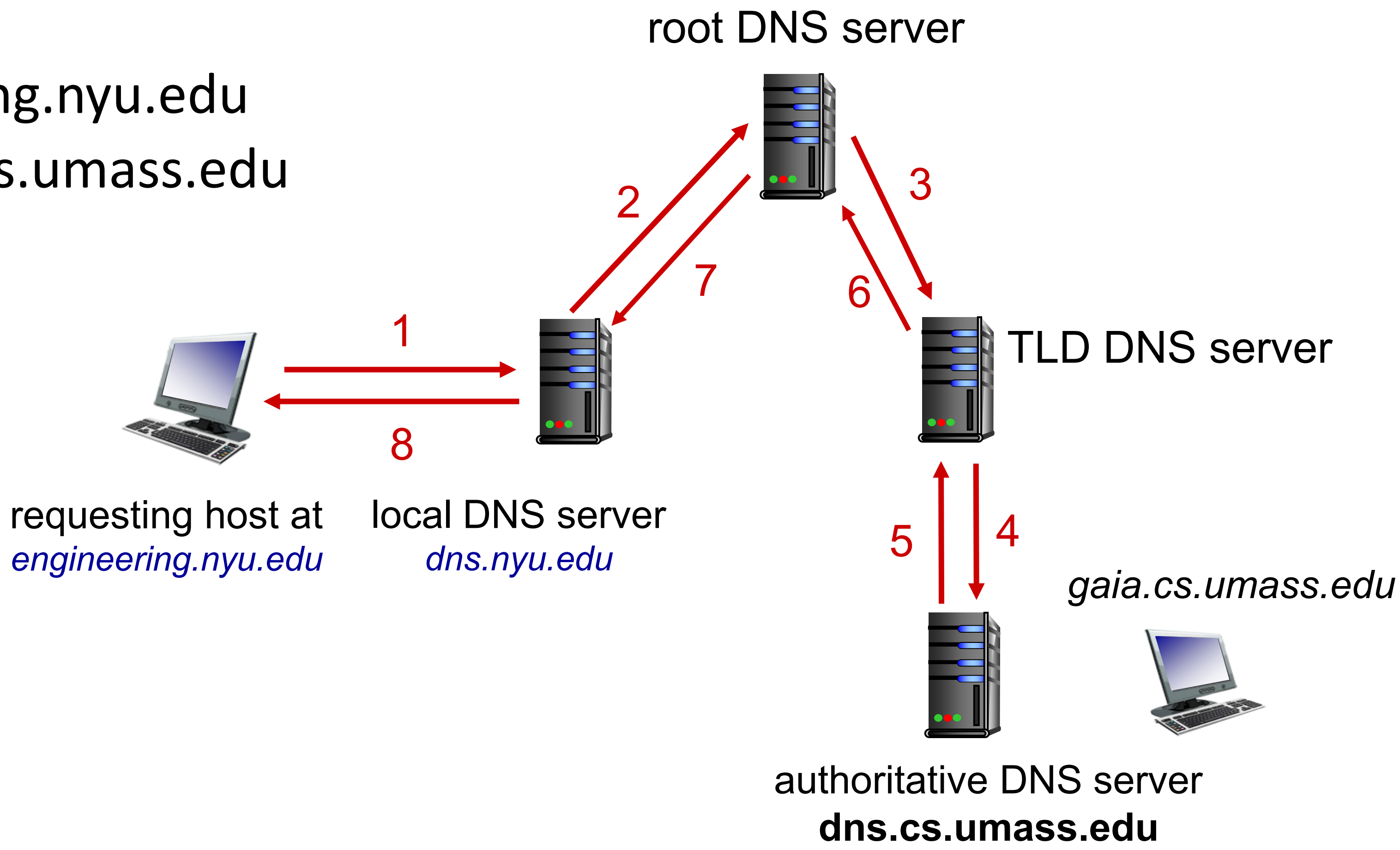


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

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



Caching DNS Information

- once (any) name server learns mapping, it *caches* mapping, and *immediately* returns a cached mapping in response to a query
  - caching improves response time
  - cache entries disappear after some time (TTL)
  - TLD servers typically cached in local name servers
- cached entries may be *out-of-date*
  - if named host changes IP address, may not be known Internet-wide until all TTLs expire!
  - DNS tries its best but cached data may be outdated

DNS records or RR (Resource Record)

DNS: distributed database storing resource records (RR)

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

type=A(Address record - IPv4 address)

- `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(Canonical NAME)

- `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(Mail eXchange)

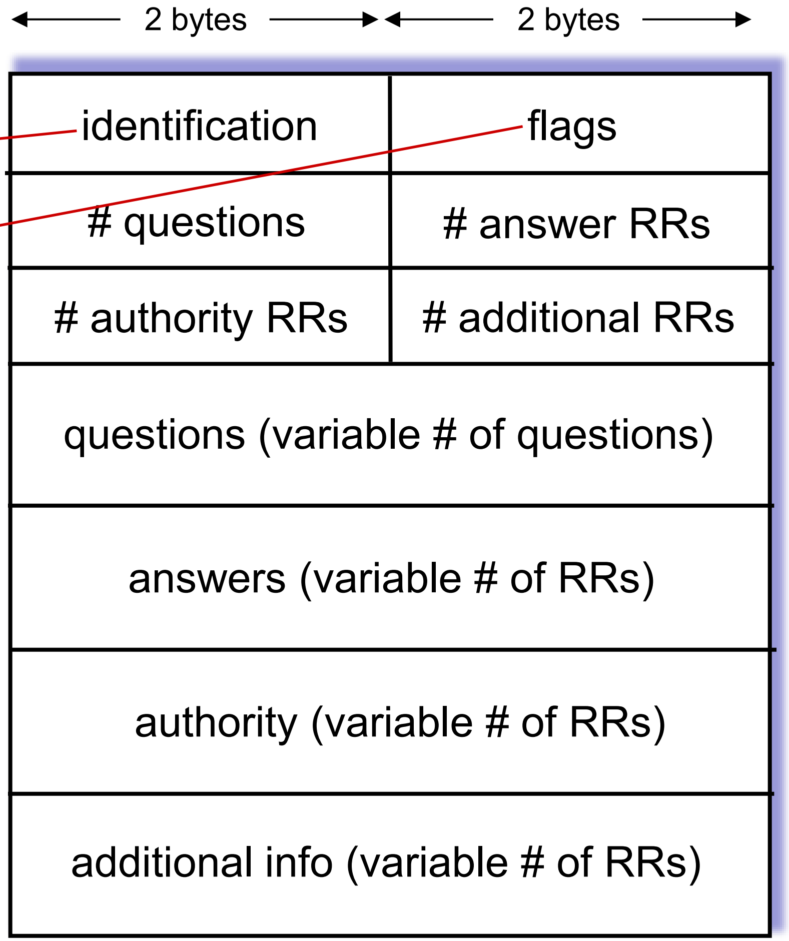
- `value` is name of SMTP mail server associated with `name`

DNS protocol messages

DNS *query* and *reply* messages, both have same *format:*

message header:

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



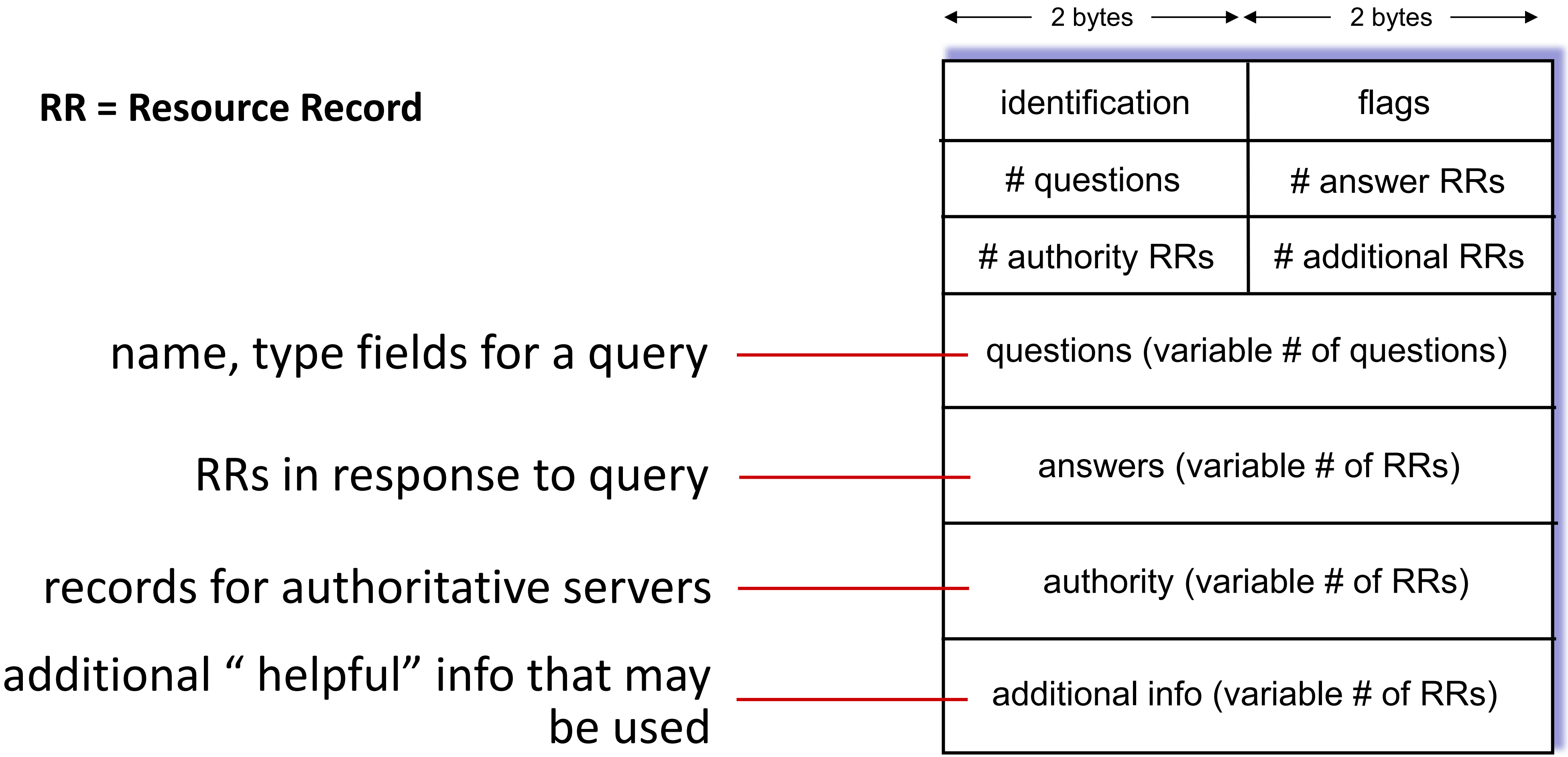
2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

RR = Resource Record (A, MX, NS, CNAME)

DNS protocol messages

DNS *query* and *reply* messages, both have same *format:*

RR = Resource Record



2 bytes

2 bytes

identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs

questions (variable # of questions)

answers (variable # of RRs)

authority (variable # of RRs)

additional info (variable # of RRs)

name, type fields for a query

RRs in response to query

records for authoritative servers

additional " helpful" info that may be used

Getting your info into the 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 NS, A RRs into .com TLD server:

    `(networkutopia.com, dns1.networkutopia.com, NS)`

    `(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server locally with IP address `212.212.212.1`
  - type A record for www.networkuptopia.com
  - type MX record for networkutopia.com

DNS security

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

Spoofing attacks

- intercept DNS queries, returning bogus replies
  - DNS cache poisoning
  - RFC 4033: DNSSEC authentication services

Attacks on DNS

DDoS bandwidth-flooding attack - Oct 21, 2002

- truck loads of ICMP ping messages to each of the 13 DNS root IP addresses
  - caused minimal damage

deluge of DNS queries to top-level-domain servers Oct 21, 2016

- botnet consisting of about one hundred thousand IoT devices such as
  - printers, IP cameras etc
- full day 'Amazon, Twitter, Netflix, Github and Spotify' were disturbed.

Interactive Exercises

DNS – BASICS

DNS - ITERATIVE VS RECURSIVE QUERY

Application layer: overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- video streaming and content distribution networks
- socket programming with UDP and TCP

Video Streaming and CDNs: context

- stream video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube, Amazon Prime: 80% of residential ISP traffic (2020)
- *challenge:* scale - how to reach ~1B users?
- *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*

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

frame *i+1*

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, 64Kbps – 12 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)

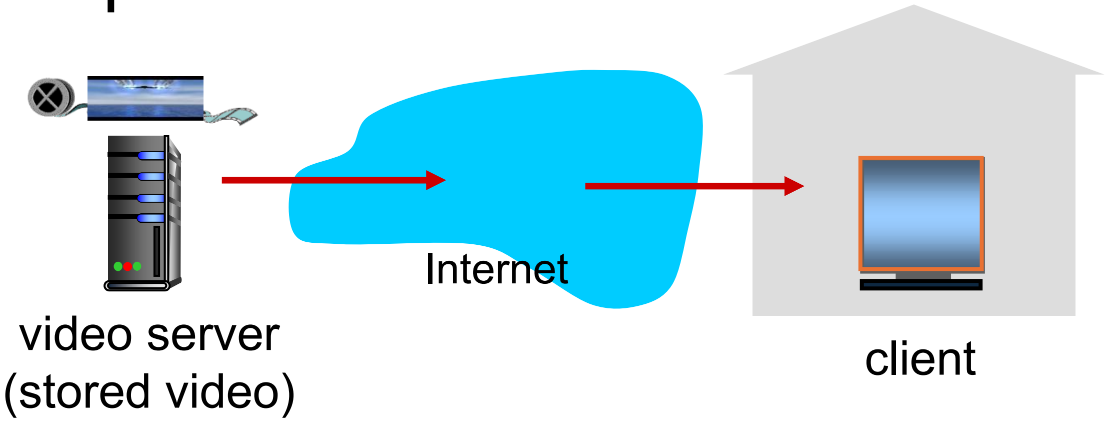
frame *i*

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

frame *i+1*

Streaming stored video

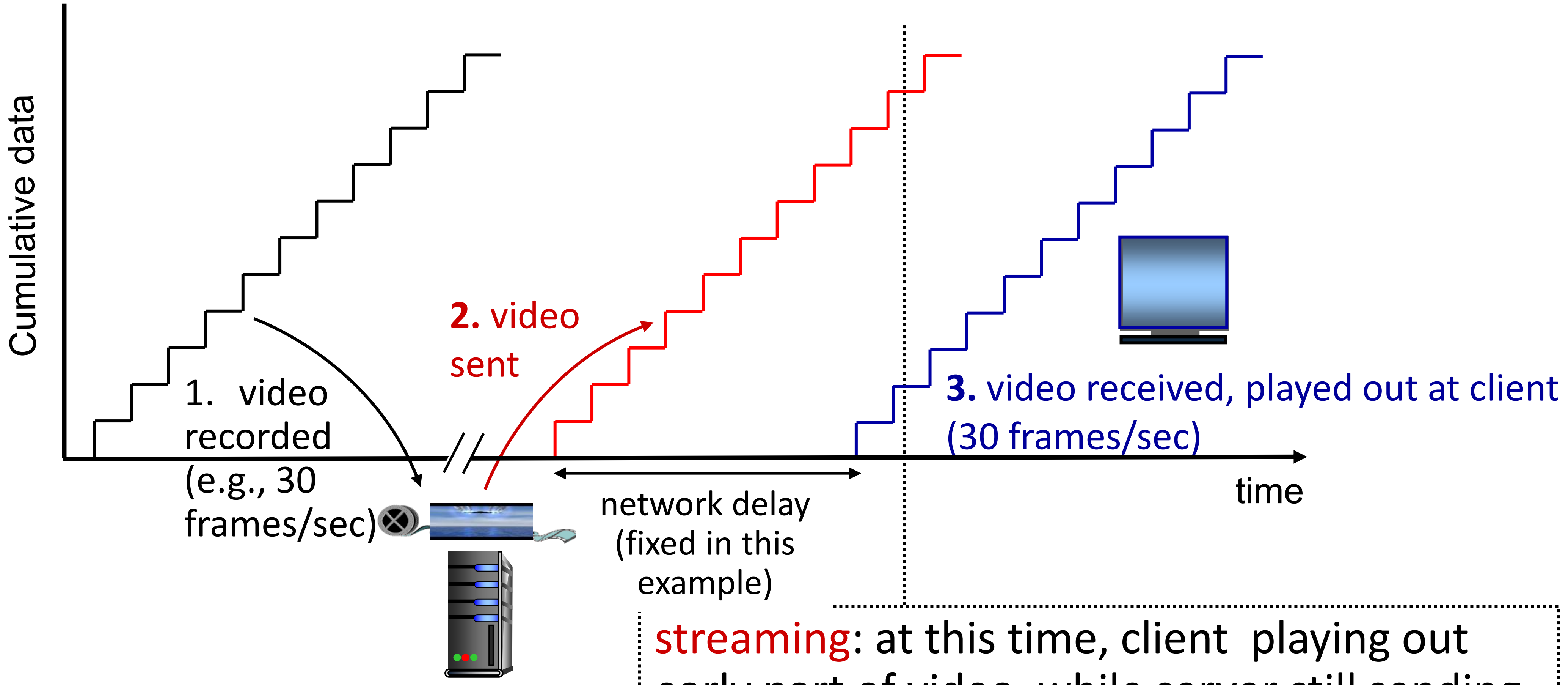
simple scenario:



Main challenges:

- server-to-client bandwidth will *vary* over time, with changing network congestion levels (in house, access network, network core, video server)
- packet loss, delay due to congestion will delay playout, or result in poor video quality

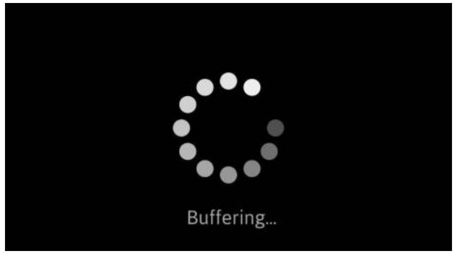
Streaming stored video


Cumulative data
1. video recorded (e.g., 30 frames/sec)
2. video sent
network delay (fixed in this example)
3. video received, played out at client (30 frames/sec)
time

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

Streaming stored video: Client-side challenges

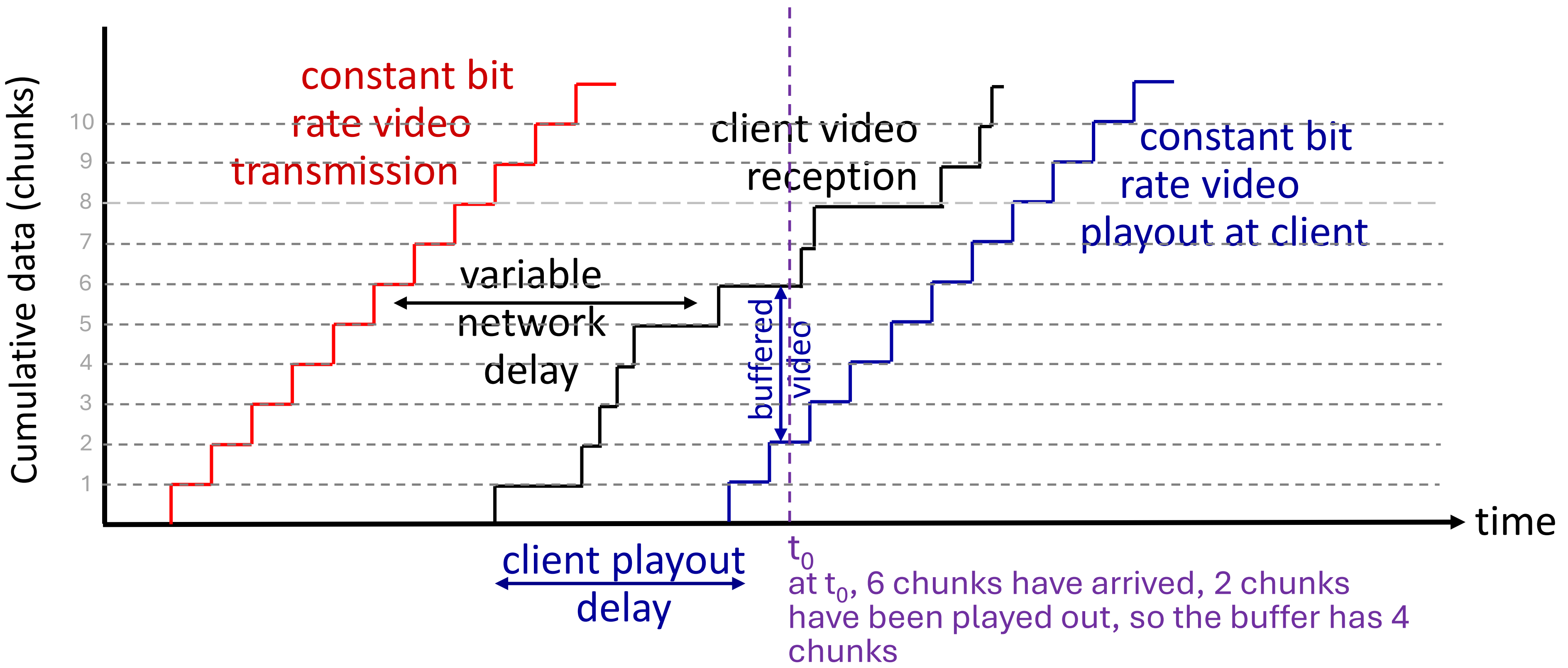
- continuous playout constraint: during client video playout, playout timing must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match continuous playout constraint



- other challenges:
  - client interactivity: pause, fast-forward, rewind, jump through video
  - video packets may be lost, retransmitted -> delay

Streaming stored video: playout buffering

solution to the jitter



at t_0, 6 chunks have arrived, 2 chunks have been played out, so the buffer has 4 chunks

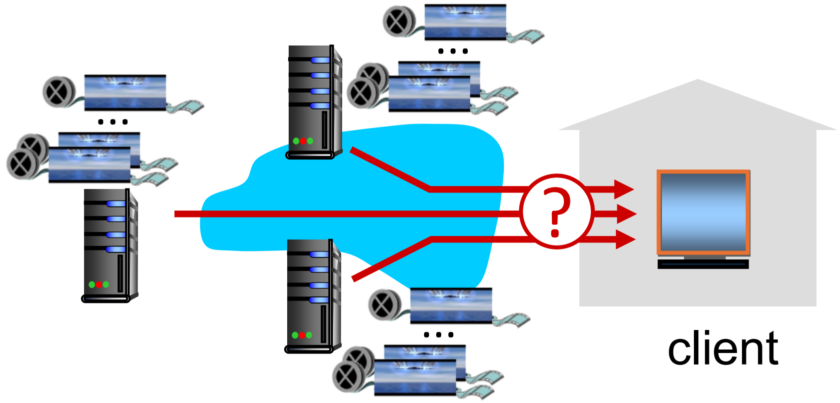
- *client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Streaming multimedia: DASH

*D*ynamic, *A*daptive *S*treaming over *H*TTP

server:

- divides video file into multiple chunks
- each chunk encoded at multiple different rates
- different rate encodings stored in different files
- files replicated in various CDN nodes
- *manifest file:* provides URLs for different chunks

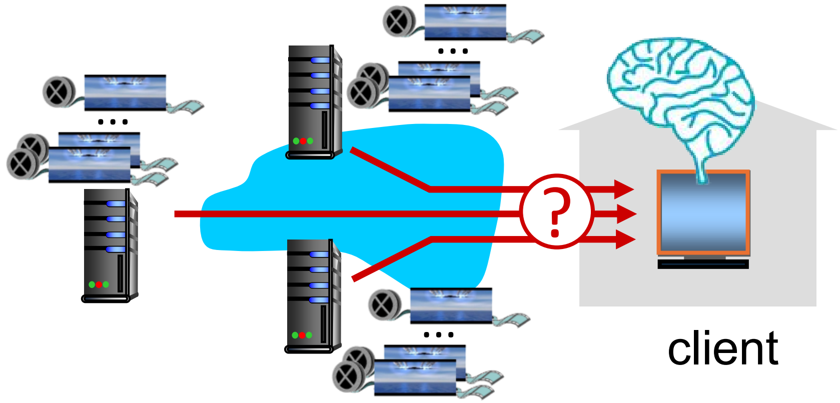


client:

- periodically estimates 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), and from different servers

Streaming multimedia: DASH

- *"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)



Streaming video = encoding + DASH + playout buffering

Content distribution networks (CDNs)

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

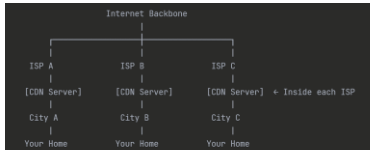
- *option 1:* single, large “mega-server”
  - single point of failure
  - point of network congestion
  - long (and possibly congested) path to distant clients

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

Content distribution networks (CDNs)

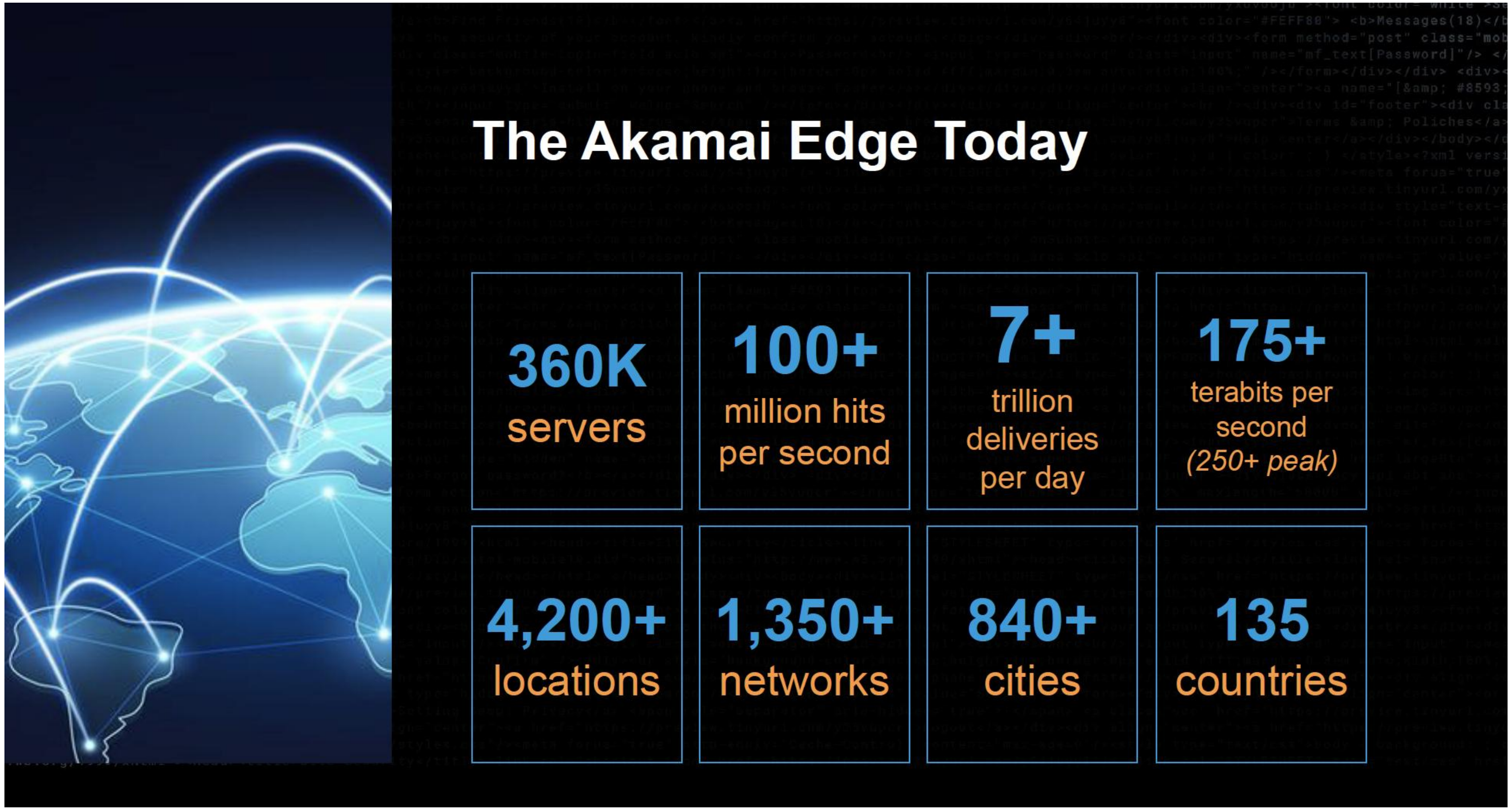
challenge: how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- *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
    - Akamai: 240,000 servers deployed in > 120 countries (2015)
  - *bring home:* smaller number (10's) of larger clusters in POPs near access nets
    - used by Limelight



```
              Internet Backbone
                      |
        ┌─────────────┼─────────────┐
        |             |             |
      ISP A         ISP B         ISP C
        |             |             |
   [CDN Server]  [CDN Server]  [CDN Server]  ← Inside each ISP
        |             |             |
      City A        City B        City C
        |             |             |
    Your Home     Your Home     Your Home
```

Akamai today:



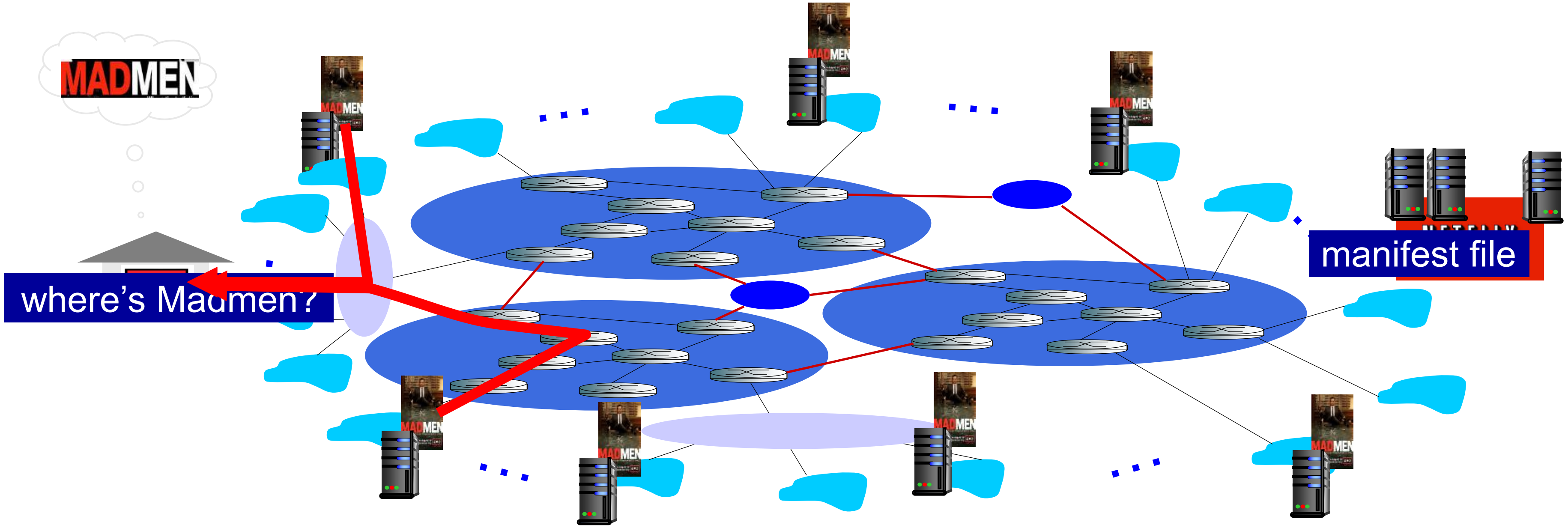
The Akamai Edge Today

360K servers	100+ million hits per second	7+ trillion deliveries per day	175+ terabits per second *(250+ peak)*
4,200+ locations	1,350+ networks	840+ cities	135 countries

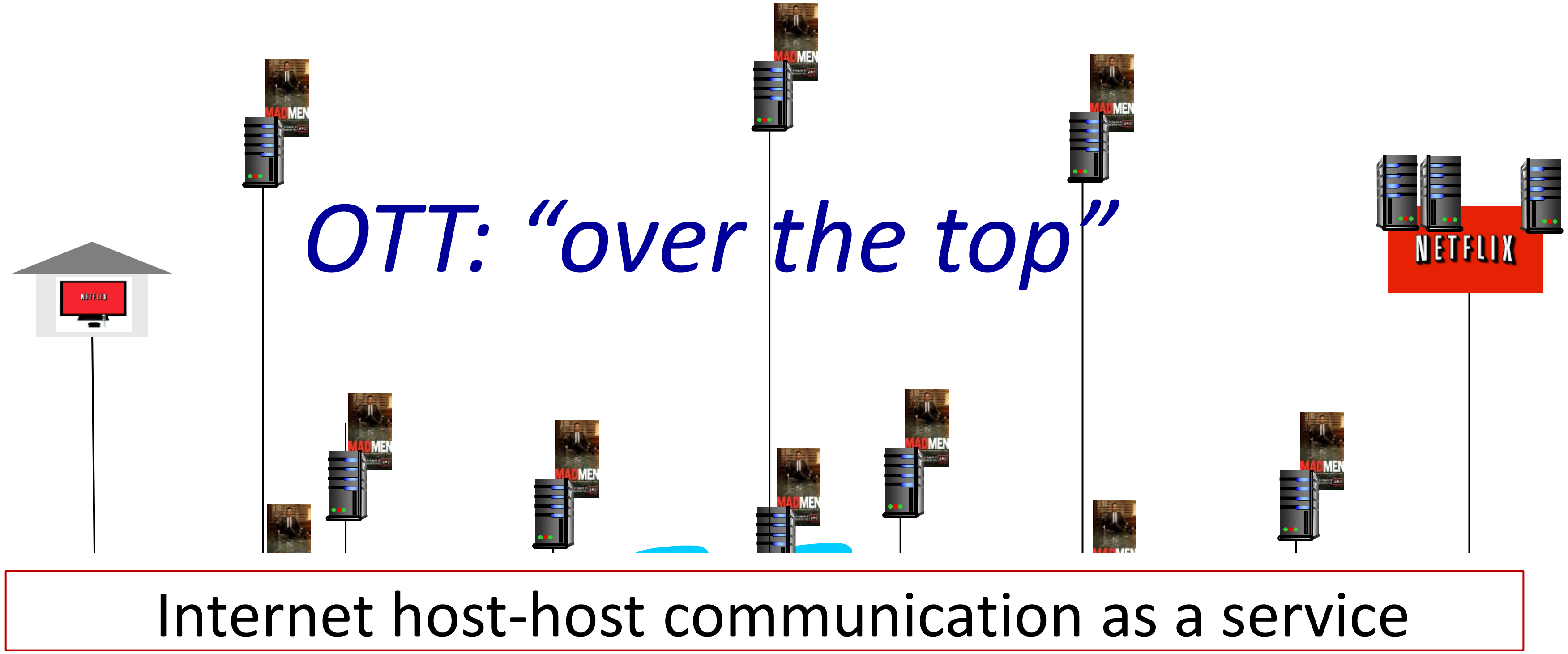
Source: https://networkingchannel.eu/living-on-the-edge-for-a-quarter-century-an-akamai-retrospective-downloads/

How does Netflix work?

- Netflix: stores copies of content (e.g., MADMEN) at its (worldwide) OpenConnect CDN nodes
- subscriber requests content, service provider returns manifest
  - using manifest, client retrieves content at highest supportable rate
  - may choose different rate or copy if network path congested



Content distribution networks (CDNs)



OTT: “over the top”

Internet host-host communication as a service

OTT challenges: coping with a congested Internet from the “edge”

- what content to place in which CDN node?
- from which CDN node to retrieve content? At which rate?