NTNU – Trondheim
Norwegian University of
Science and Technology

Application Layer – Web and Email

TTM4200 – Computer Networks

2[1]min binary

- X [K M G T]i → 2^n
  n→10, 20, 30, 40
- X [KMGT] → 10^n
  n→3, 6, 9,12

- b, bit → bit
- B, byte → byte (8 bit)

Prefixes for decimal and binary multiples

Decimal				Binary				
Value		SI		Value		IEC		JEDEC
1000	10^3	k	kilo	1024	2^{10}	Ki	kibi	K kilo
1000^2	10^6	M	mega	1024^2	2^{20}	Mi	mebi	M mega
1000^3	10^9	G	giga	1024^3	2^{30}	Gi	gibi	G giga
1000^4	10^{12}	T	tera	1024^4	2^{40}	Ti	tebi	T tera
1000^5	10^{15}	P	peta	1024^5	2^{50}	Pi	pebi	–
1000^6	10^{18}	E	exa	1024^6	2^{60}	Ei	exbi	–
1000^7	10^{21}	Z	zetta	1024^7	2^{70}	Zi	zebi	–
1000^8	10^{24}	Y	yotta	1024^8	2^{80}	Yi	yobi	–
1000^9	10^{27}	R	ronna			–		
1000^{10}	10^{30}	Q	quetta			–		

https://en.wikipedia.org/wiki/Binary_prefix V·T·E

individual Readiness Assurance Test

- Find the right answer **alone.**
- Choose the answer that fits **best.**
- **One** handwritten A4 page as helping material.

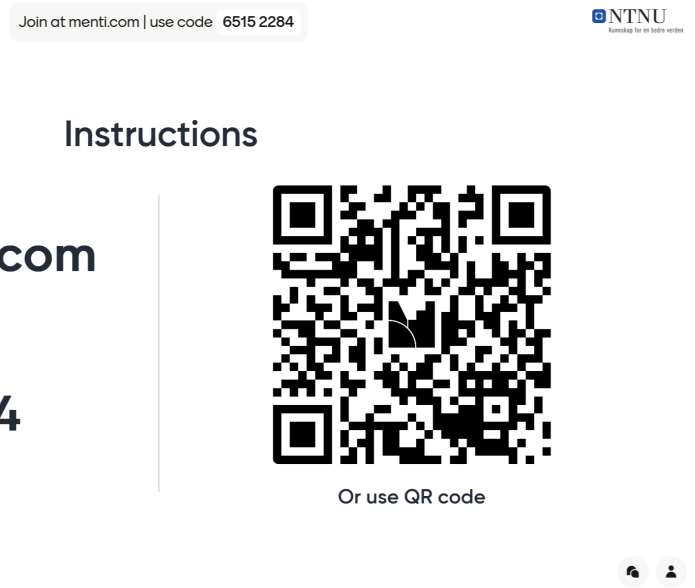
team Readiness Assurance Test

- Find the right answer **in teams (breakout rooms).**
- Choose the answer that fits **best.**
- **One** handwritten A4 page as helping material.

- **Discuss!**

Time for a break

we restart at 11:20



Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

HTTP request message

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

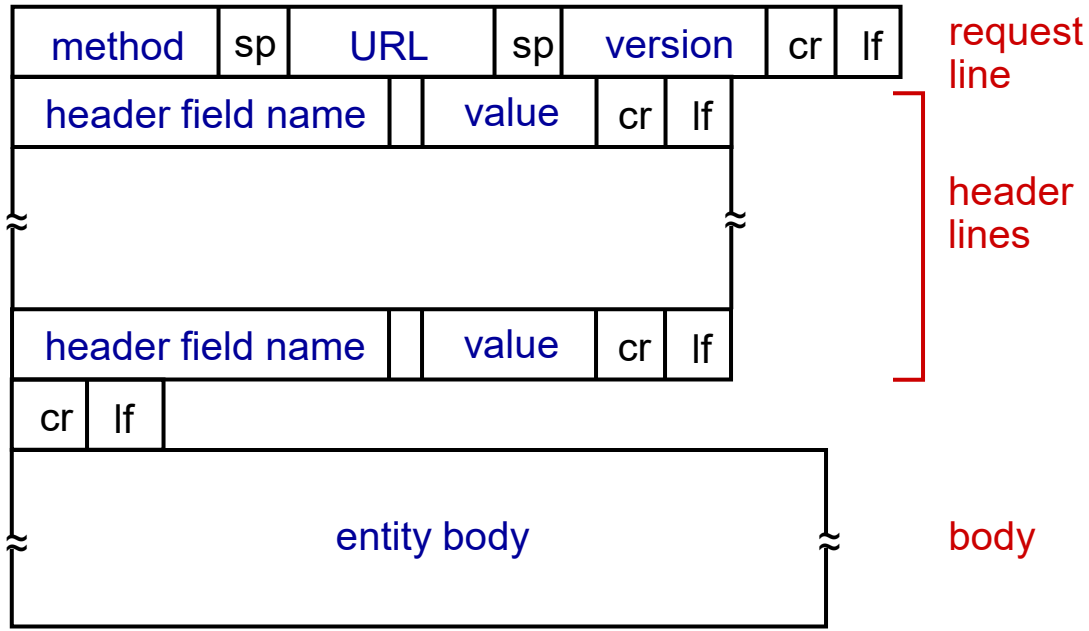
carriage return character
line-feed character

request line (GET,
POST,
HEAD commands)

carriage return, line
feed at start of line
indicates end of header
lines

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

HTTP request message: general format



Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

Other HTTP request messages

POST method:

- web page often includes form input
- user input sent from client to server in entity body of HTTP POST request message

GET method (for sending data to server):

- include user data in URL field of HTTP GET request message (following a '?'):

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

HEAD method:

- requests headers (only) that would be returned *if* specified URL were requested with an HTTP GET method.

PUT method:

- uploads new file (object) to server
- completely replaces file that exists at specified URL with content in entity body of POST HTTP request message

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

HTTP response message

status line (protocol status code status phrase) → `HTTP/1.1 200 OK`

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

HTTP response status codes

- status code appears in 1st line in server-to-client response message.
- some 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 (in Location: field)

400 Bad Request
- request msg not understood by server

404 Not Found
- requested document not found on this server

505 HTTP Version Not Supported

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

Trying out HTTP (client side) for yourself

1. netcat to your favorite Web server:

   % nc -c -v ntnu.no 80
   (ncat –C –v ntnu.no 80

   - opens TCP connection to port 80 (default HTTP server port) at ntnu.no.
   - anything typed in will be sent to port 80 at ntnu.no

2. type in a GET HTTP request:

```
GET / HTTP/1.1
Host: www.ntnu.no
```

   - 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!

   (or use Wireshark to look at captured HTTP request/response)

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

SMTP: observations

comparison with HTTP:

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

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

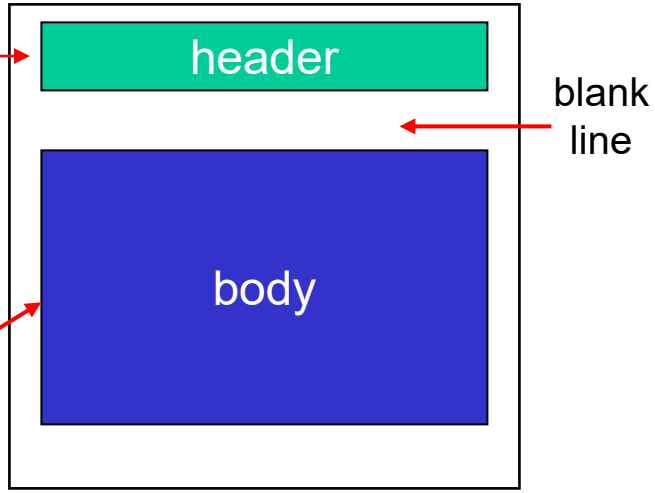
Mail message format

SMTP: protocol for exchanging e-mail messages, defined in RFC 5321 (like RFC 7231 defines HTTP)

RFC 2822 defines *syntax* for e-mail message itself (like HTML defines syntax for web documents)

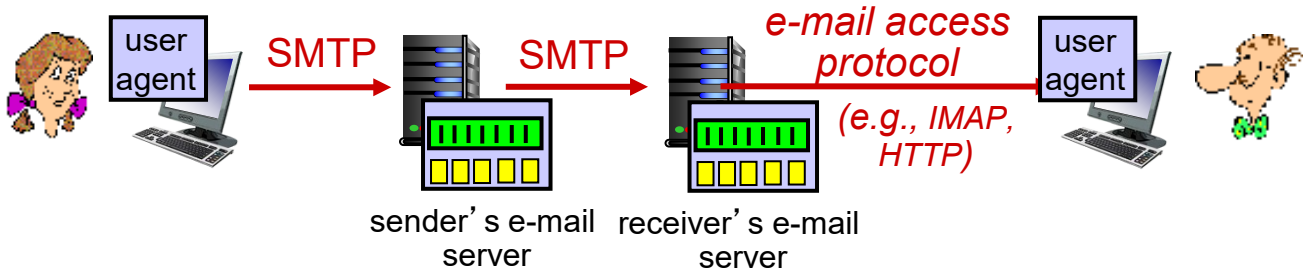
- header lines, e.g.,
  - To:
  - From:
  - Subject:

  these lines, within the body of the email message area different from SMTP MAIL FROM:, RCPT TO: commands!
- Body: the "message" , ASCII characters only



Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

Retrieving email: mail access protocols



- SMTP: delivery/storage of e-mail messages to receiver's server
- mail access protocol: retrieval from server
  - IMAP: Internet Mail Access Protocol [RFC 3501]: messages stored on server, IMAP provides retrieval, deletion, folders of stored messages on server
- HTTP: gmail, Hotmail, Yahoo!Mail, etc. provides web-based interface on top of SMTP (to send), IMAP (or POP) to retrieve e-mail messages

openssl

ansatt / stud

```
openssl s_client -quiet -starttls smtp -connect smtp.XXXXX.ntnu.no:587 (-crlf)

EHLO localhost
AUTH
MAIL FROM: thomas.zinner@ntnu.no
RCPT TO: thomas.zinner@ntnu.no
DATA
FROM: thomas.zinner@ntnu.no
TO: thomas.zinner@ntnu.no
SUBJECT: Testing terminal emails!
TTM4200 is great!?
Best regards...

.

QUIT
```

```
AUTH PLAIN echo -ne
“\0username@ntnu.no\0password” | base64
or AUTH LOGIN
Linux/MAC: echo -n username@ntnu.no | base64
Windows Powershell:
[Convert]::ToBase64String([System.Text.Encoding]:
:UTF8.GetBytes(“username@ntnu.no"))
```

```
$ openssl s_client -quiet -starttls smtp -connect smtp.ansatt.ntnu.no:587
depth=2 C = US, ST = New Jersey, L = Jersey City, O = The USERTRUST Network, CN = USERTrust RSA Certification (…)
depth=0 C = NO, ST = Tr\C3\B8ndelag, O = Norges Teknisk-Naturvitenskapelige Universitet NTNU, CN = smtp.stud.ntnu.no
verify return:1
250 CHUNKING
HELO LOCALHOST
250-mailgw401.it.ntnu.no
AUTH LOGIN
334 VXNlcm5hbWU6
YOUR USERNAME in BASE64
334 UGFzc3dvcmQ6
YOUR PASSWORD in BASE64
235 2.7.0 Authentication successful
MAIL FROM thomas.zinner@ntnu.no
501 5.5.4 Syntax: MAIL FROM:<address>
MAIL FROM: thomas.zinner@ntnu.no
250 2.1.0 Ok
RCPT TO: thomas.zinner@ntnu.no
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: thomas.zinner@ntnu.no
To: thomas.zinner@ntnu.no
Subject: TTM4200 fun!?

This is the body.
.
250 2.0.0 Ok: queued as 64937640030
quit
221 2.0.0 Bye
```

Enough for today

we continue Thursday at 10:15

Join at menti.com | use code 6515 2284

NTNU
Kunnskap for en bedre verden

Instructions

Go to

www.menti.com

Enter the code

6515 2284

Or use QR code

Welcome!

Join at menti.com | use code **8979 0516**

The code allows your audience to join the presentation and expires in 2 days.

Instructions

Go to

www.menti.com

Enter the code

8979 0516

Or use QR code

Questions and Comments

- When will we get the dates for the oral exam?

→ Will discuss this during the first reference group meeting and then make a survey available

- Will there be a language requirement for the oral exam

→ We can discuss the rules for the oral exam

- Do we write a report on todays lab? Or only later on lab3 and 4?

→ Announcement Trond

- On canvas it says that preperation material will be posted in "Preperation Materials and Readings", but only the week 3 rat preperations were published here. Will preperation material be posted there?

→ Added now week 4 and 5; can do that parallel, but will also keep the webpage updated

RATs

Question 2: Assuming similar network conditions, the time for retrieving a single webpage without associated objects is:

The same for persistent and non-persistent HTTP connections because we only retrieve 1 file. (This is the best answer.)

64.0%

The same for persistent and non-persistent HTTP connections if the propagation delay is the same.

20.0%

Higher if we use a persistent HTTP connection.

4.0%

Lower if we use a persistent HTTP connection.

12.0%

RATs

Question 9: Suppose \textit{Client A} wants to retrieve a large file of $\textrm{6\,MiB}$ from \textit{Server B} using HTTP. The path from \textit{Client A} to \textit{Server B} has three consecutive links of rates $R_1=\textrm{1\,Mbit/s}$, $R_2=\textrm{2\,Mbit/s}$, $R_3=800\textrm{\,kbit/s}$ (i.e. A <-- 1Mbit/s --> R <-- 2Mbit/s --> R <-- 800kbit/s --> B). Assume there's no other traffic in the network, \textbf{how long does it take} to complete the file transfer?

$2\textrm{RTT} + \textrm{slightly more than } 1\textrm{\,min}$ (This is the best answer.)
44.0%

$2\textrm{RTT} + \textrm{exactly } 1\textrm{\,min}$
30.0%

$\textrm{RTT} + \textrm{exactly } 1\textrm{\,min}$
6.0%

$\textrm{RTT} + \textrm{slightly more than } 7.5\textrm{\,s}$
20.0%

Question 9: Suppose *Client A* wants to retrieve a large file of 6 MiB from *Server B* using HTTP. The path from *Client A* to *Server B* has three consecutive links of rates $R_1 = 1\,\text{Mbit/s}$, $R_2 = 2\,\text{Mbit/s}$, $R_3 = 800\,\text{kbit/s}$ (i.e. A <- 1Mbit/s -> R <- 2Mbit/s -> R <- 800kbit/s -> B). Assume there's no other traffic in the network, **how long does it take** to complete the file transfer?

A 2RTT + slightly more than 1 min
B 2RTT + exactly 1 min
C RTT + exactly 1 min
D RTT + slightly more than 7.5 s

RATs

Question 10: A small departmental website serves one HTML page plus 18 additional objects (images/CSS), all from the same host over HTTP/1.1. Current server settings: Persistent connections are disabled (non-persistent HTTP). Static files have no Cache-Control. Clients are not configured to use any proxy (forward cache). RTT to the server is about 100 ms, with negligible loss. Which change will most reduce the total completion time of the initial page load for typical users under these constraints?

A Enable HTTP persistent connections (keep-alive) so the browser can reuse one TCP connection for multiple object requests.

B Add Cache-Control (e.g., max-age) and validators (ETag/Last-Modified) to all static objects.

C Stand up a forward web cache (proxy) near the server, but keep client browsers unchanged.

D Enable cookie-based sessions so the server can maintain state across object requests.

Problem 10 (*pg. 200 & 201)*

Consider a 30-meter link, over which a sender can transmit at a rate of $300\ \text{bps}$ in both directions.

Suppose that packets containing data are $100\ \text{kbits}$ long, and packets containing only control (e.g., ACK or handshaking) are $200\ \text{bits}$ long.

Assume that N parallel connections each get $1/N$ of the link's bandwidth.

Now, consider the HTTP protocol and suppose that each downloaded object is $100\ \text{kbits}$ long, and that the initial downloaded object contains 10 referenced objects from the same sender.

a) Would **parallel downloads** via parallel instances of **non-persistent HTTP** make sense in this case?

b) Now consider **persistent HTTP** (not parallel). Do you expect significant gains over the non-persistent case? Justify and explain your answer.

Notes: Ignore headers; Ignore TCP tear down; Speed of light: $300 \cdot 10^6\ \text{m/s}$;

Solving Problem 10 *(pg. 200 & 201)*

General observations:

- each downloaded object can be completely put into one data packet.
- d_p denotes the one-way propagation delay between the client and the server
- d_t denotes the one-way transmission delay between the client and the server.

a) Parallel downloads would allow 10 connections (up to 10 because we can only download 10 objects, could be fewer) to share the $300\ \text{bits/s}$ bandwidth, giving each just $30\ \text{bits/s}$. Thus, the total time needed to receive all objects is given by:

$$d = (\text{TCP connection} + \text{requesting initial file} + \text{transmitting initial file}) + (\text{TCP connection} + \text{requesting ref. object} + \text{transmitting ref. object})$$

(10 in parallel)

$$\left(\frac{200}{300} + d_p + \frac{200}{300} + d_p + \frac{200}{300} + d_p + \frac{100000}{300} + d_p\right) + \left(\frac{200}{30} + d_p + \frac{200}{30} + d_p + \frac{200}{30} + d_p + \frac{100000}{30} + d_p\right) = 3688.66(6) + 8 \cdot d_p \text{ seconds}$$

$d_p = \frac{30}{300 \cdot 10^6} = 0.1\ \mu sec$ ← negligible!

Now consider a persistent HTTP connection. The total time needed is given by:

$$d = (\text{TCP connection} + \text{requesting initial file} + \text{transmitting initial file}) + (\text{requesting ref. object} + \text{transmitting ref. object})$$

$$\left(\frac{200}{300} + d_p + \frac{200}{300} + d_p + \frac{200}{300} + d_p + \frac{100000}{300} + d_p\right) + 10 \cdot \left(\frac{200}{300} + d_p + \frac{100000}{300} + d_p\right) = 3675.33(3) + 24 \cdot d_p \text{ seconds}$$

b) We see that persistent HTTP is not significantly faster than the non-persistent case with parallel download because the “control data” overhead is minimal and so is the propagation delay.

Problem 11 (*pg. 201)*

Consider the scenario introduced in the previous problem. Now, suppose that the link is shared by Alice with Bob. Alice does not use parallel instances of non-persistent HTTP while Bob uses non-persistent HTTP with five parallel downloads each.

a) Does Alice have any advantage over Bob? Why or why not?

b) If Alice opens five parallel instances of non-persistent HTTP, then would her parallel connections be beneficial? Why or why not?

Solving Problem 11 (*pg. 201)*

a) No, because Bob has more connections, he can get a larger share of the link bandwidth.

b) Yes, otherwise, she will get less bandwidth.

Problem X

Assume you request a webpage consisting of one document and five images. The document size is $1\ \text{kbyte}$, all images have the same size $50\ \text{kbytes}$, the download rate is $1\ \text{Mbps}$, and the RTT is $100\ \text{ms}$. How long does it take to obtain the whole webpage under the following conditions? (assume no DNS name query is needed and the impact of the request line and the headers in the HTTP messages is negligible).

a) Nonpersistent HTTP with serial connections.

b) Nonpersistent HTTP with two parallel connections.

c) Nonpersistent HTTP with six parallel connections.

d) Persistent HTTP with one connection.

Note: $1\ \text{kbyte} \neq 1\text{kiB}$

Solving Problem X

a) $2 \cdot 100\ \text{ms} + 8 \cdot 10^3 \text{bits}/1 \cdot 10^6\ \text{bits/s} + 5 \cdot (2 \cdot 100\ \text{ms} + 8 \cdot 50 \cdot 10^3 \text{bits}/1 \cdot 10^6\ \text{bits/s}) = 3.208\ \text{s}$

b) $2 \cdot 100\ \text{ms} + 8 \cdot 10^3 \text{bits}/1 \cdot 10^6\ \text{bits/s} + 2 \cdot (2 \cdot 100\ \text{ms} + 4 \cdot 10^5\ \text{bits}/5 \cdot 10^5\ \text{bits/s}) + (2 \cdot 100\ \text{ms} + 4 \cdot 10^5\ \text{bits}/1 \cdot 10^6\ \text{bits/s}) = 2.808\ \text{s}$

c) $2 \cdot 100\ \text{ms} + 8 \cdot 10^3 \text{bits}/1 \cdot 10^6\ \text{bits/s} + 2 \cdot 100\ \text{ms} + 4 \cdot 10^5\ \text{bits}/2 \cdot 10^5\ \text{bits/s} = 2.408\ \text{s}$

d) $2 \cdot 100\ \text{ms} + 8 \cdot 10^3 \text{bits}/1 \cdot 10^6\ \text{bits/s} + 5 \cdot (4 \cdot 10^5 \text{bits}/1 \cdot 10^6\ \text{bits/s}) = 2.208\ \text{s}$

Readings for next week

Book pages

- 152-164, 166 (Sec. 2.4)
- 166-173 (2.5)

Key concepts

- DNS
  - Aliasing
  - Load distribution
  - Records (A, NS, CNAME and MX)
  - Distributed vs. Centralised
  - Root, TLD & authoritative servers
  - Recursive vs. iterative queries
  - DNS Caching
- P2P vs. Client-server architectures
- Scalability
- BitTorrent
  - Rarest first
  - Unchoked and optimistically unchoked

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

Application Layer: Overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

DNS: Domain Name System

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "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"

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

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

Q: Why not centralize DNS?

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

A: doesn't scale!

- Comcast DNS servers alone: 600B DNS queries/day
- Akamai DNS servers alone: 2.2T DNS queries/day

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

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!

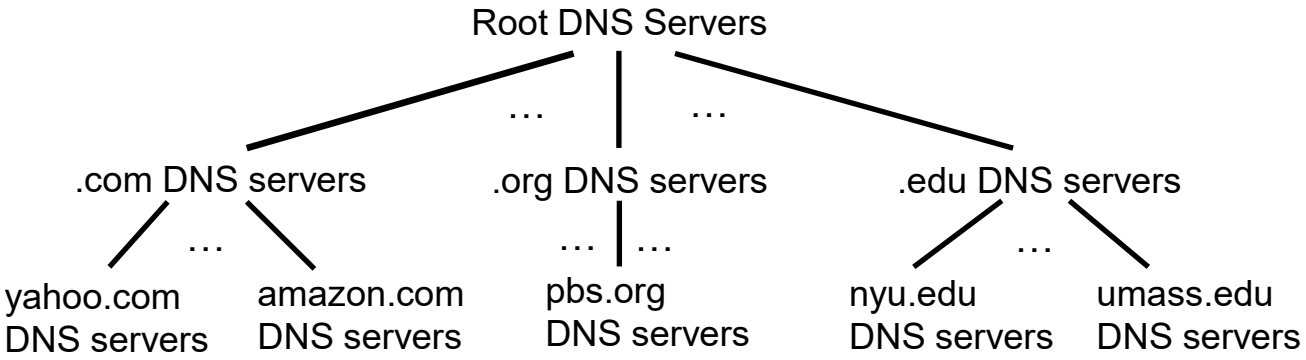
organizationally, physically decentralized:

- millions of different organizations responsible for their records

"bulletproof": reliability, security

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

DNS: a distributed, hierarchical database



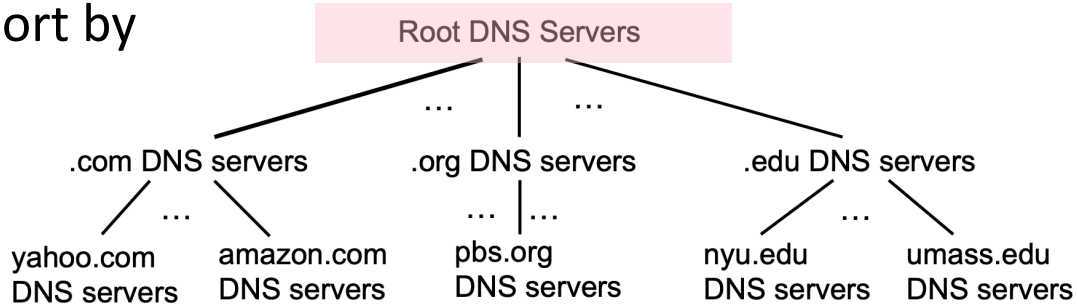
Client wants IP address for www.amazon.com; 1st approximation:

- 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

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

DNS: root name servers

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

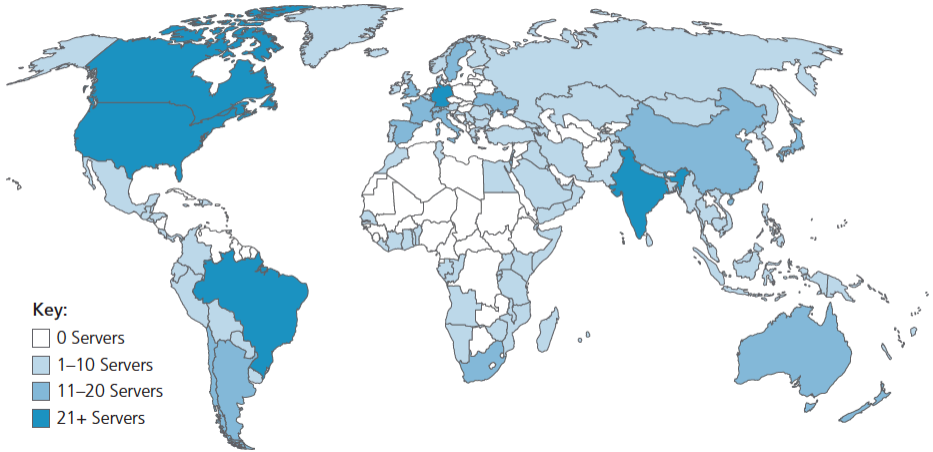


Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

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 (~200 servers in US)

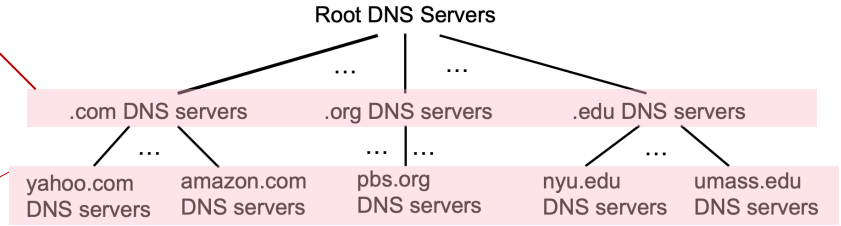


Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

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.: .no, .cn, .uk, .fr, .ca, .jp
- Network Solutions: authoritative registry for .com, .net TLD
- Educause: .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

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

Application Layer: Overview

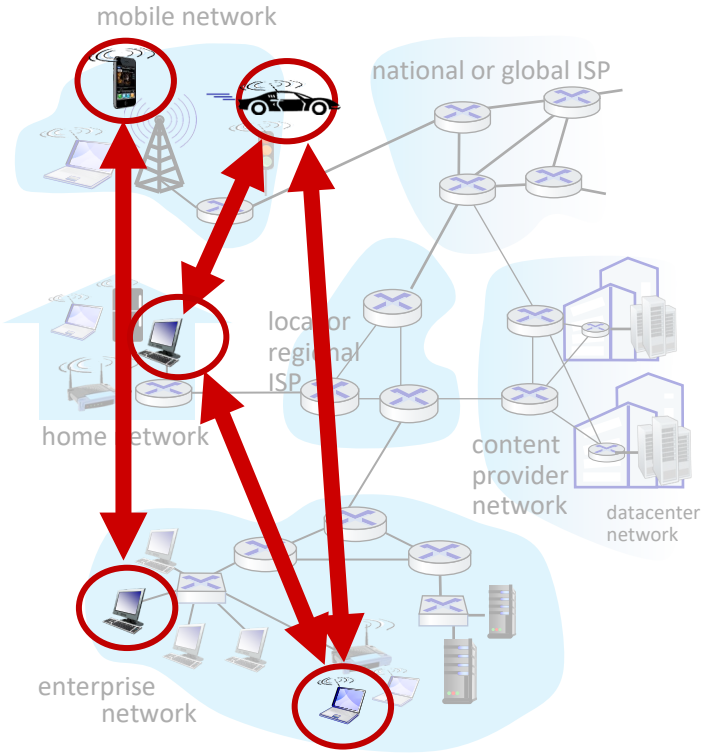
- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS

- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

Peer-to-peer (P2P) architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, and new service demands
- peers are intermittently connected and change IP addresses
  - complex management
- examples: P2P file sharing (BitTorrent), streaming (KanKan), VoIP (Skype)

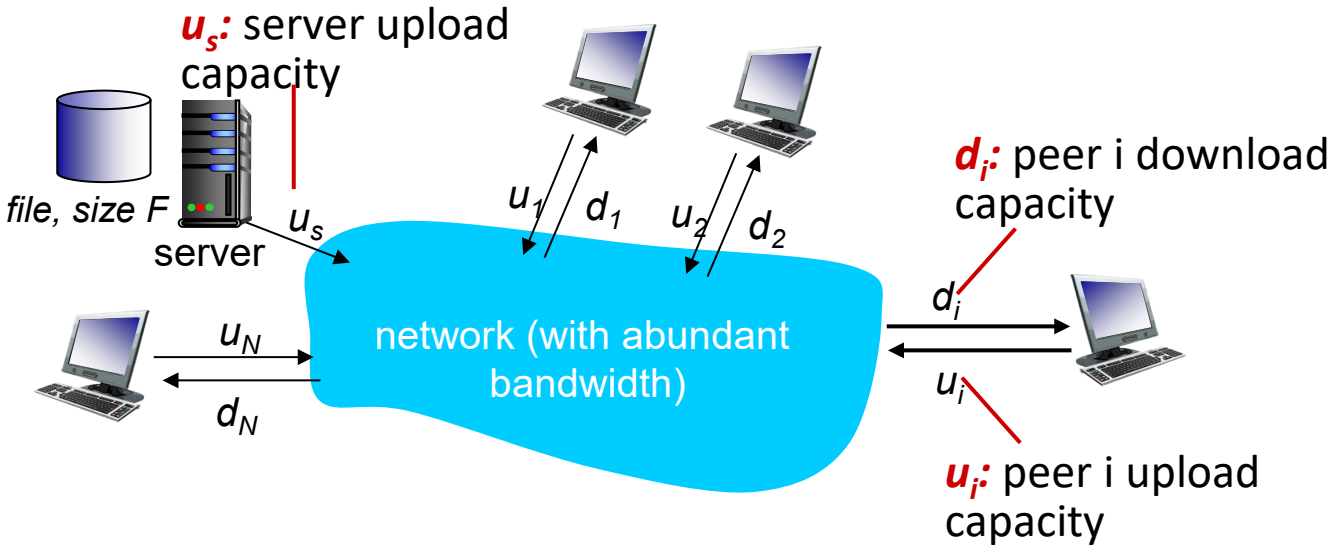


Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

File distribution: client-server vs P2P

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

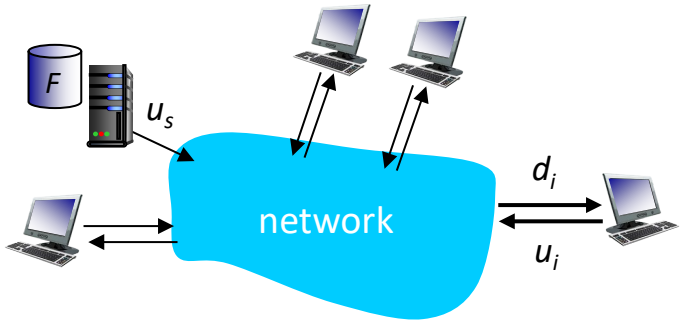
- peer upload/download capacity is limited resource



Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

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

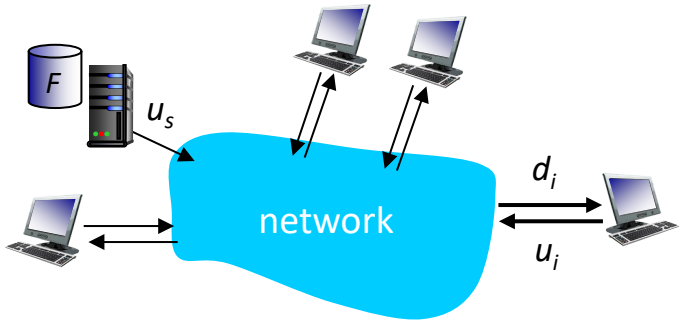
$$D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$$

increases linearly in N

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

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

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

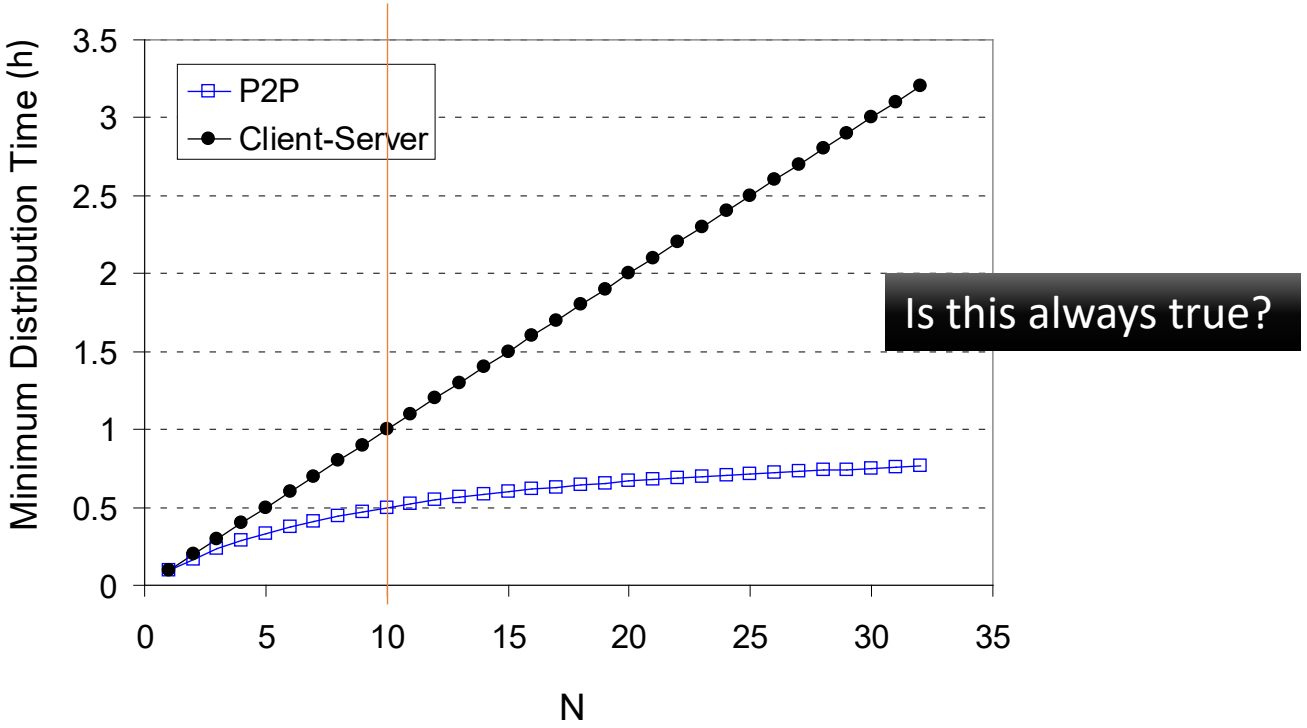
increases linearly in N …

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

Adapted from "*Computer Networking: A Top-Down Approach 8th edition*"
© Copyright 1996-2020 J.F Kurose and K.W. Ross, All Rights Reserved

Client-server vs. P2P: example

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



Readings for next week

Book pages

- 152-164, 166 (Sec. 2.4)
- 166-173 (2.5)

Key concepts

- DNS
  - Aliasing
  - Load distribution
  - Records (A, NS, CNAME and MX)
  - Distributed vs. Centralised
  - Root, TLD & authoritative servers
  - Recursive vs. iterative queries
  - DNS Caching
- P2P vs. Client-server architectures
- Scalability
- BitTorrent
  - Rarest first
  - Unchoked and optimistically unchoked

Short survey

Join at menti.com | use code 4784 6984

NTNU
Kunnskap for en bedre verden

Instructions

Go to

www.menti.com

Enter the code

4784 6984

Or use QR code