Computer Networks X_400487

Lecture 10

Chapter 7: The Application Layer

Lecturer: Jesse Donkervliet

with slides from Lin Wang

VRIJE UNIVERSITEIT AMSTERDAM

Additive increase multiplicative decrease in TCP

AIMD used to prevent network congestion. Converges to fair and efficient bandwidth allocation

TCP implements this using its ***congestion window***

Congestion window is tracked on the sender.
Specifies how many segments can be transmitted.

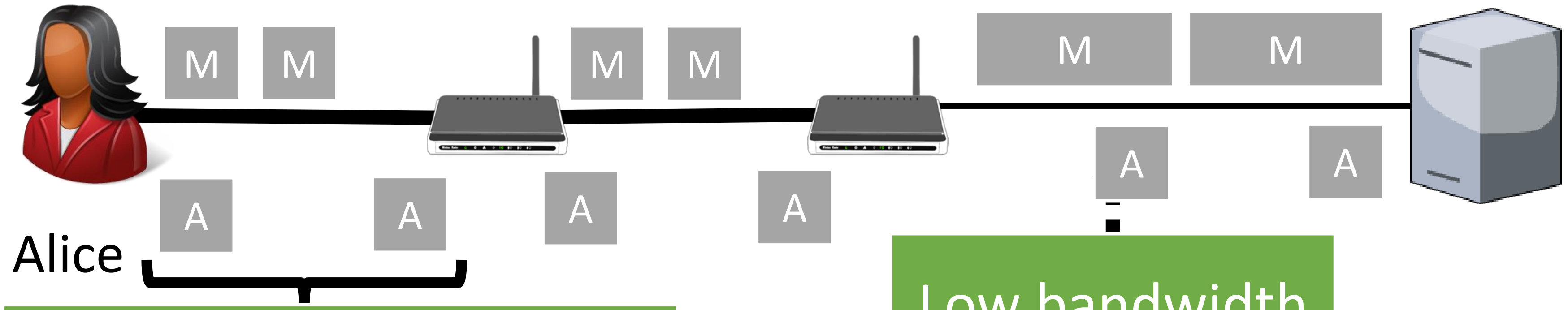
Not the same as the ‘window size’ field in the TCP segment header!

Q: How does TCP combine the two windows?

AIMD in TCP
What value to start with?

We want ***fast convergence***, but sending a large burst can occupy low-bandwidth links for a long time.

Increase congestion window whenever acknowledgements arrive

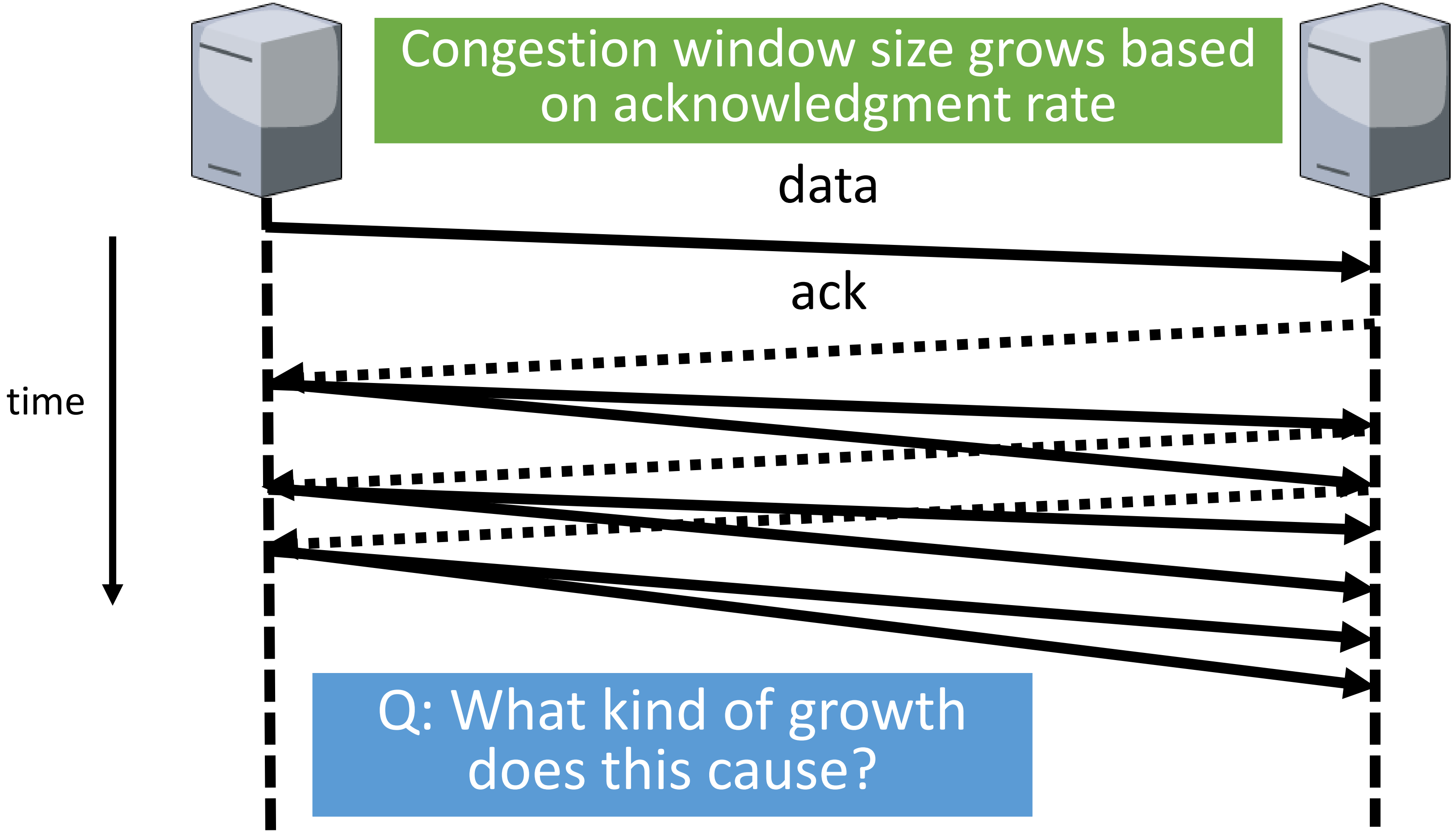


Acknowledgement rate tells us data rate of slowest link

Low bandwidth

AIMD in TCP ‘slow’ start

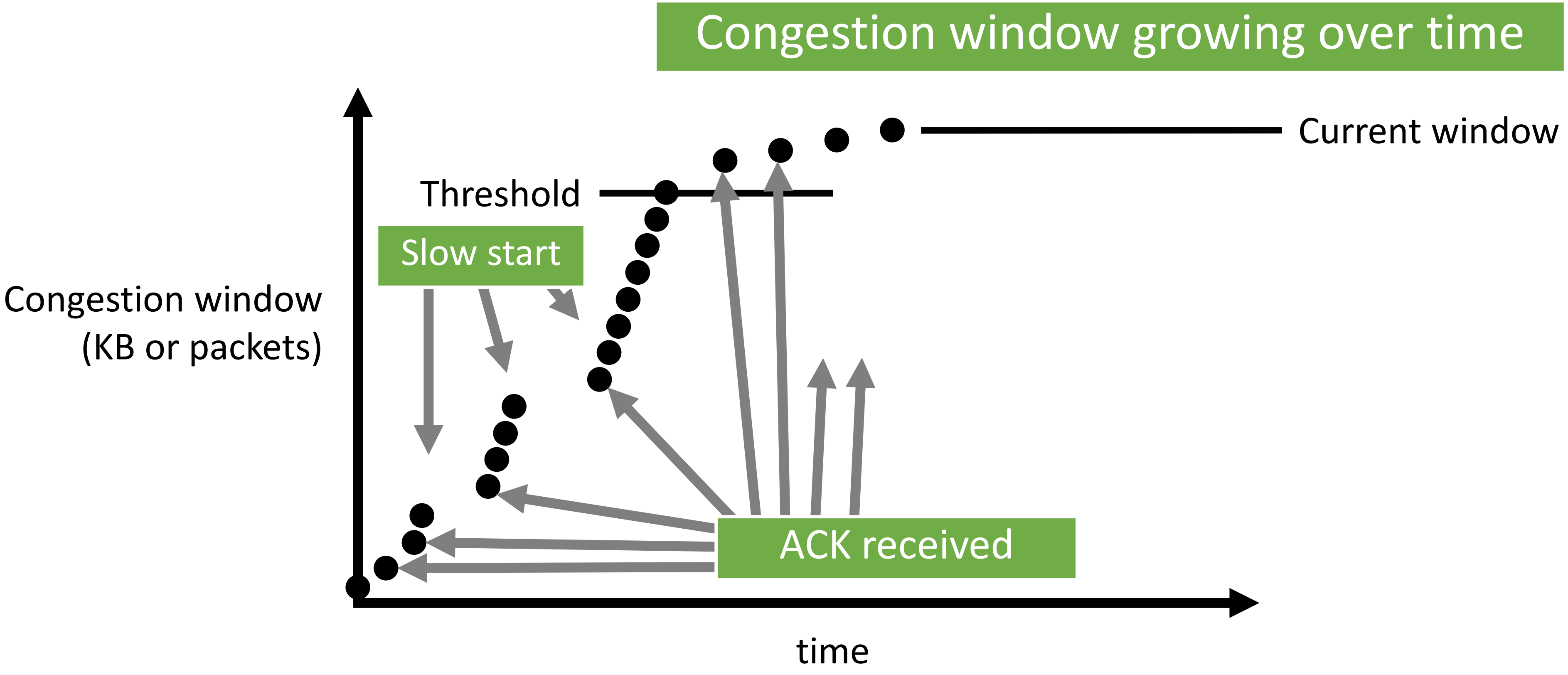
Previous algorithm used congestion window = flow control window. Slow start is slower in comparison



Q: What kind of growth does this cause?

TCP 'slow' start

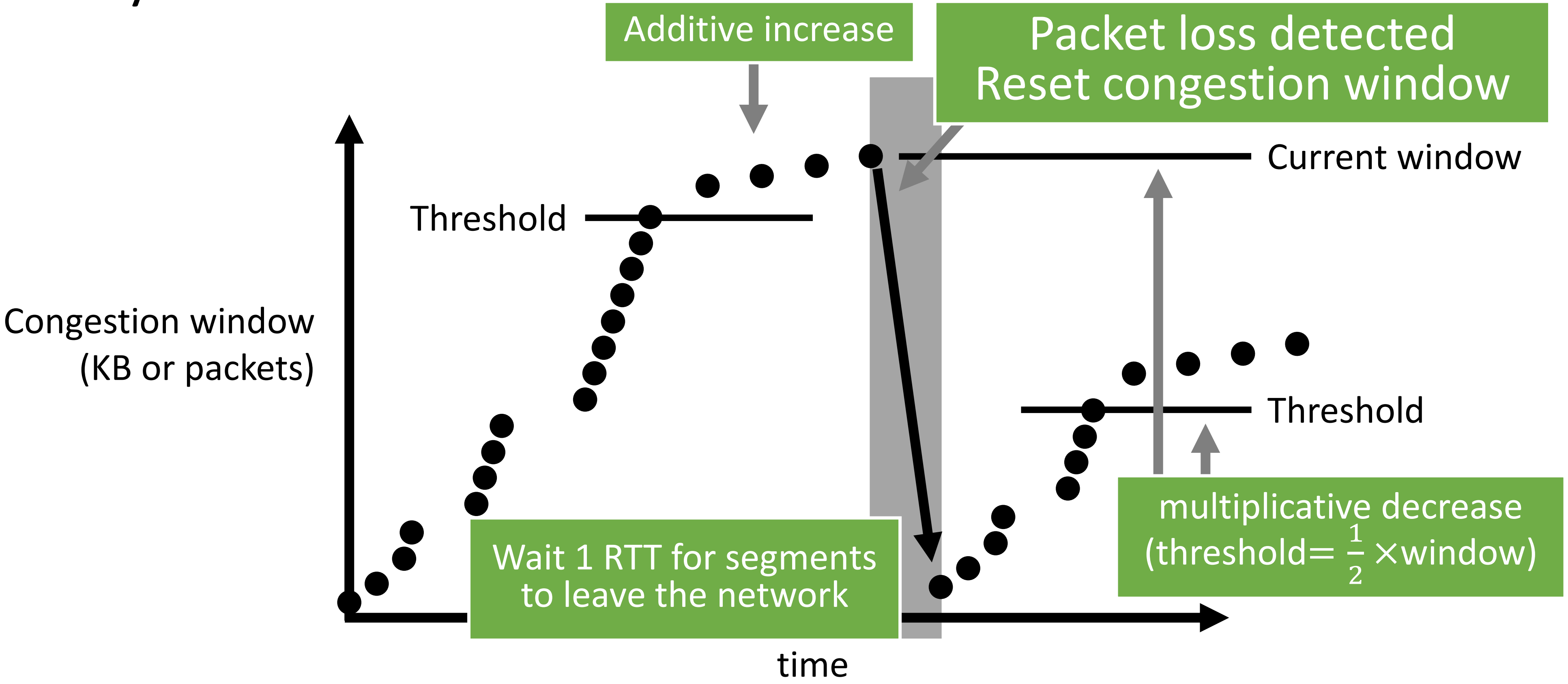
Arbitrary threshold switches from 'slow' start to ***additive increase***.



TCP Tahoe

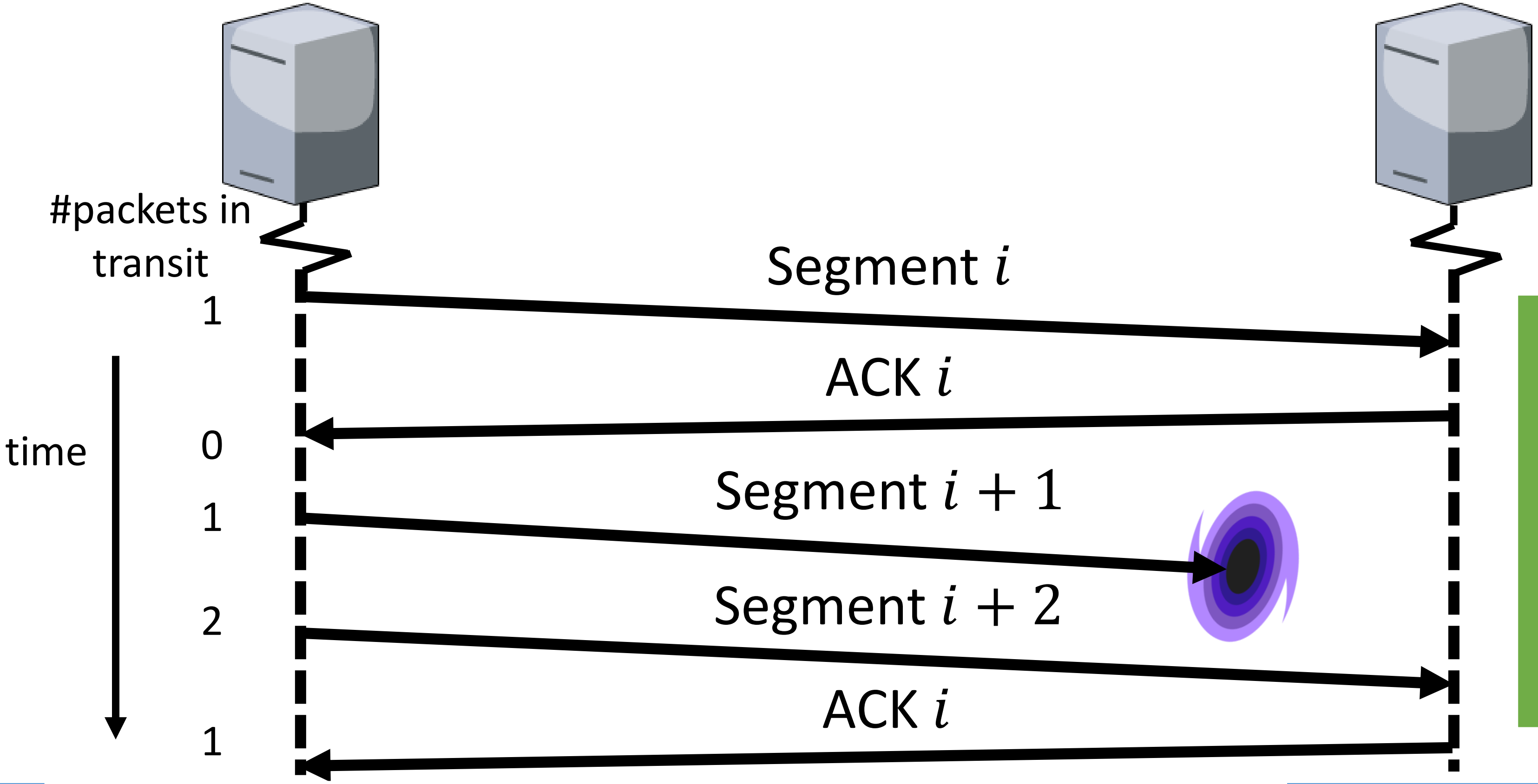
Q: Can you think of another way to detect packet loss?

Arbitrary threshold switches from 'slow' start to ***additive increase***.



Performance improvement
Fast ***retransmission***

Detects lost packets before ack timer runs out



TCP fast retransmission detects (read: assumes) lost segment after 3 duplicate acks. Triggers retransmission and *multiplicative decrease*

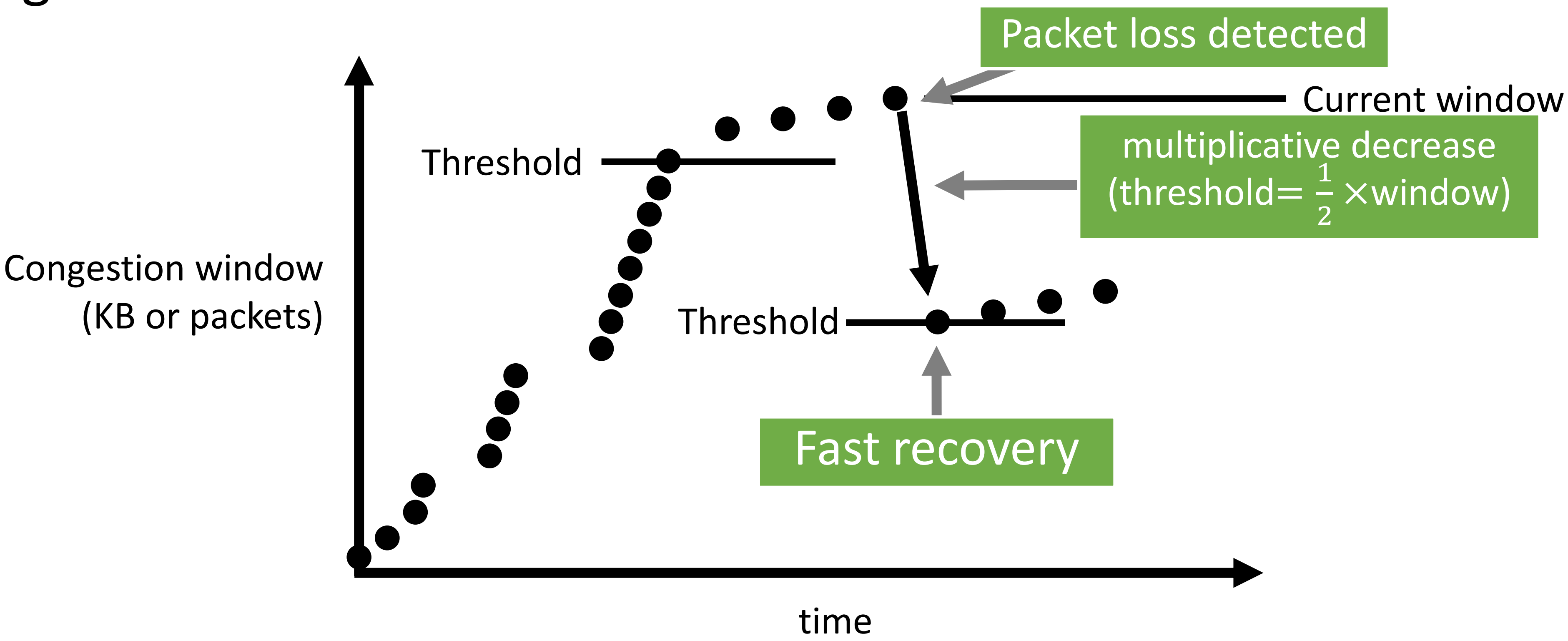
We can count the number of packets in the network!

TCP Reno
(= TCP Tahoe + fast *recovery*)

Calculates the number of segments in the network by counting the number of duplicate acknowledgements (home study: see previous slide)

Threshold reduced using ***multiplicative decrease***

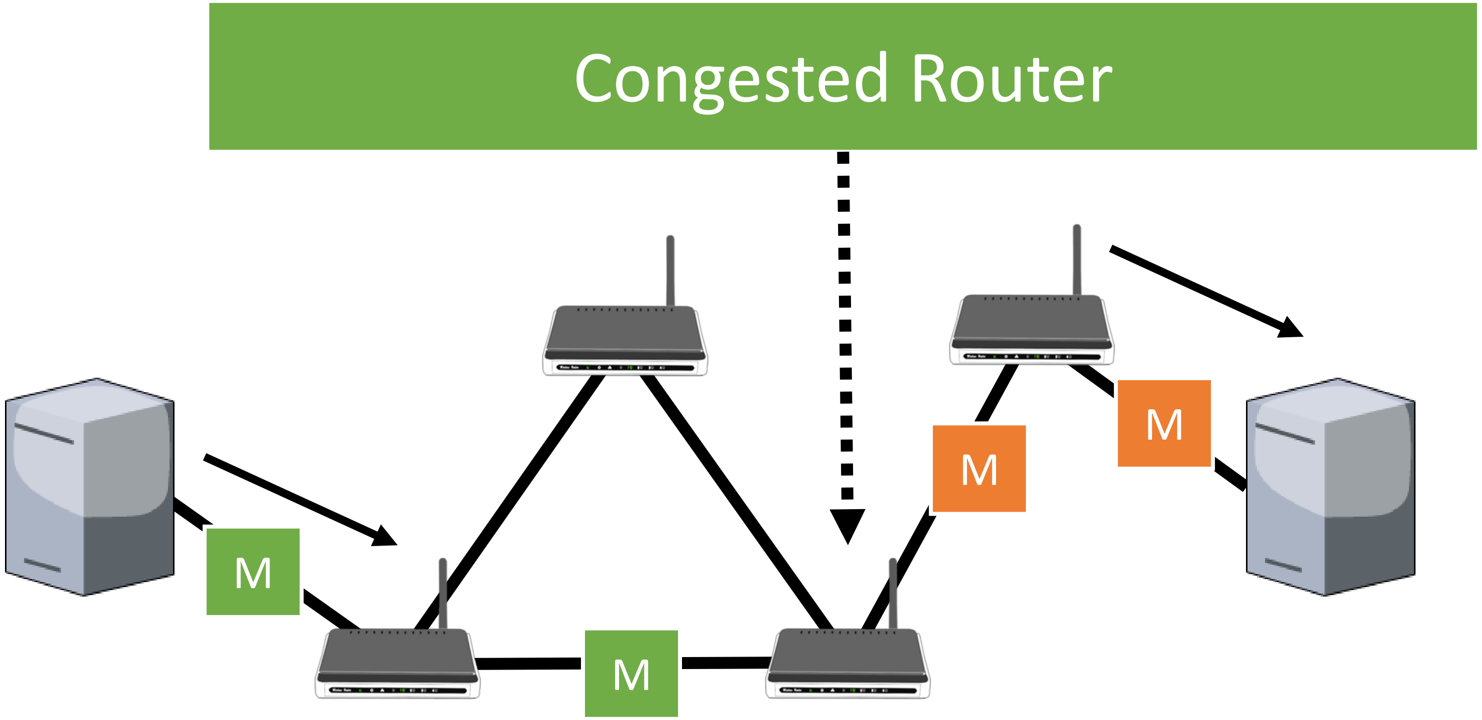
Congestion window set to new threshold value



What about
Explicit Congestion Notification?

M = regular IP packet with TCP segment

M = Explicit Congestion Notification (ECN) set in IP header

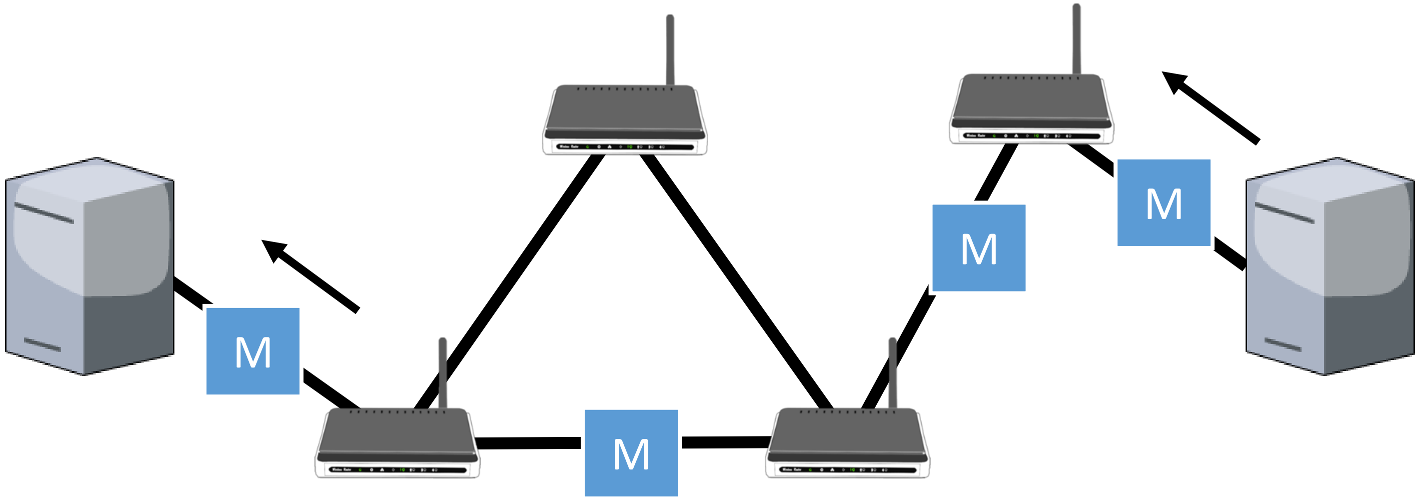


What about Explicit Congestion Notification?

M = regular IP packet with TCP segment

M = Explicit Congestion Notification (ECN) set in IP header

M = ECN-Echo (ECE) set in TCP header



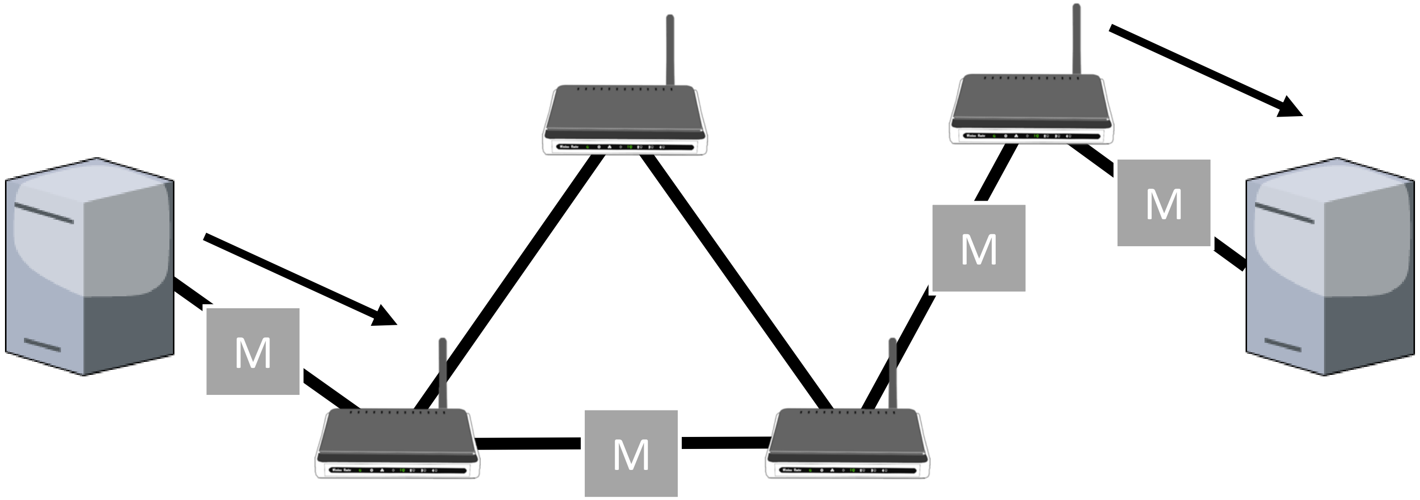
What about
Explicit Congestion Notification?

M = regular IP packet with TCP segment

M = Explicit Congestion Notification (ECN) set in **IP header**

M = ECN-Echo (ECE) set in **TCP header**

M = Congestion Window Reduced (CWR) set in **TCP header**



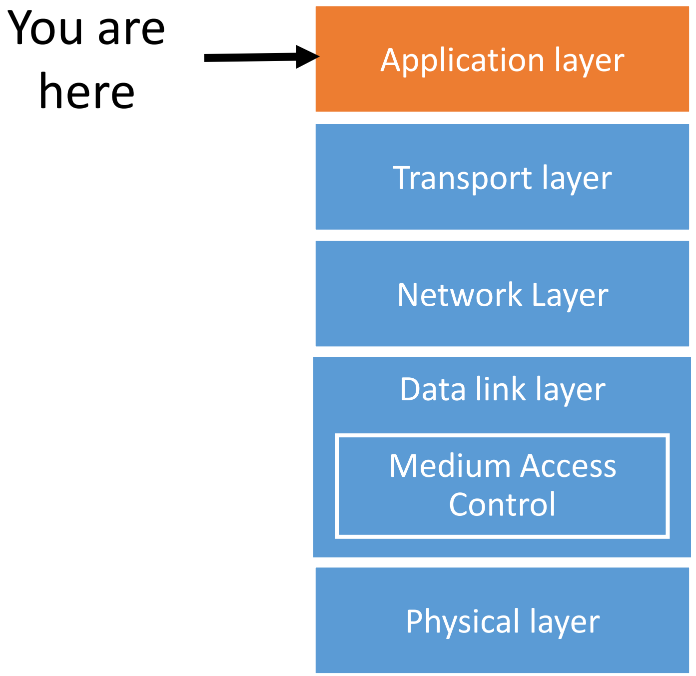
Milestone reached!

Creating large-scale distributed systems is difficult!

We can now ***start*** building applications and systems that communicate over a network!

Advanced courses unlocked:

1. Advanced Network Programming
2. Advanced Computer Networks
3. Distributed systems
   (also requires Computer Organization and Operating Systems)



Now we can finally build applications and no longer worry about networking!

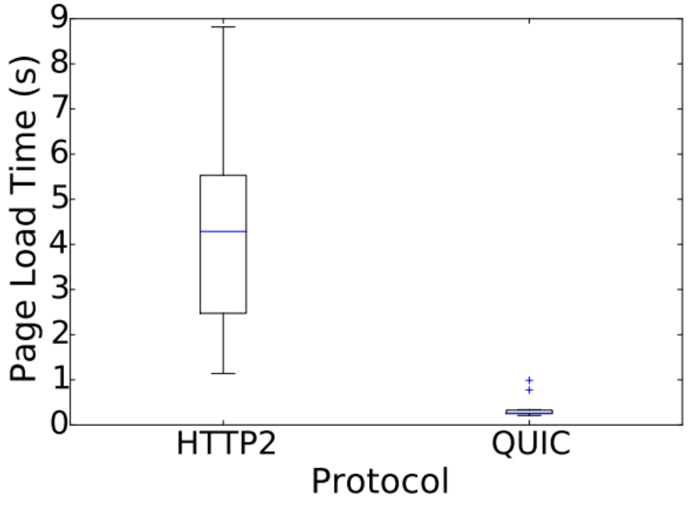
Or so we thought!

SSH IMAP FTP

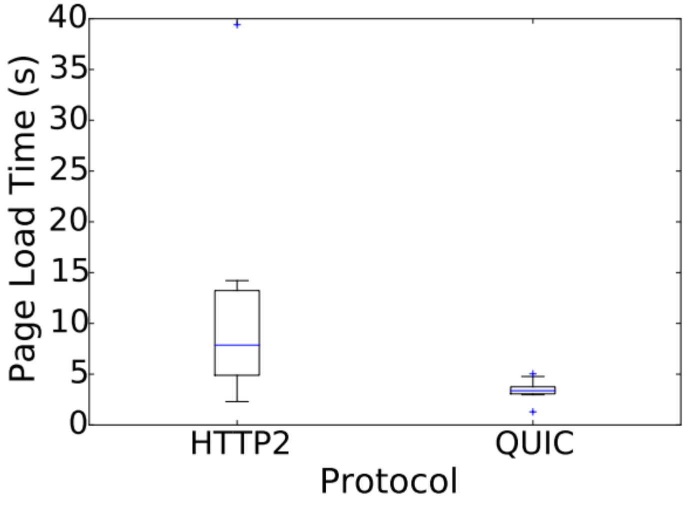
...

RPC MQTT SMTP

XMPP HTTP QUIC



(a) High Loss, Low Delay

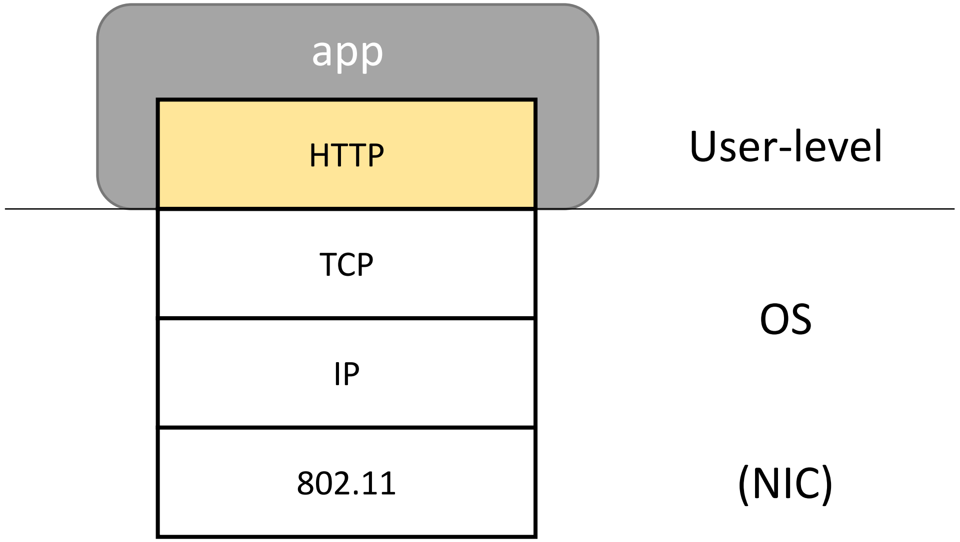


(b) High Loss, High Delay

Where The Application Layer Sits

Application layer protocols are often part of an "app"

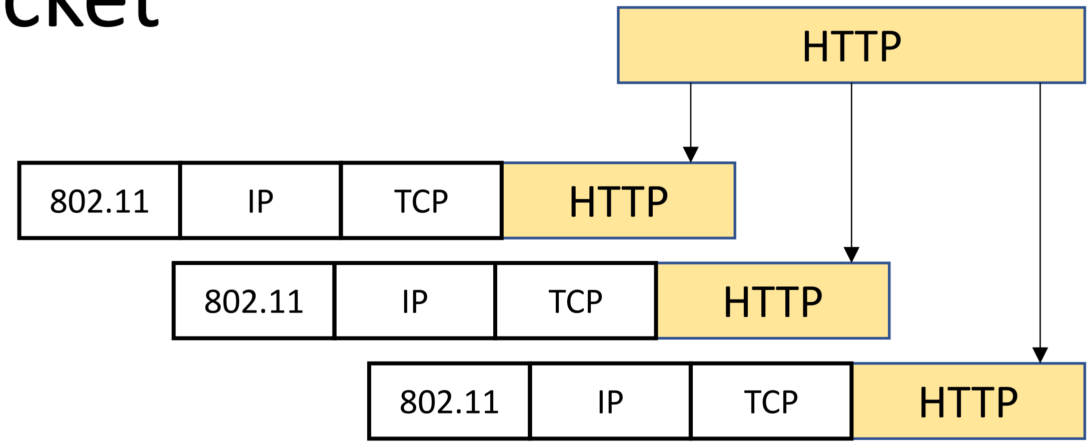
But they don't need a GUI, e.g., DNS



Application Layer Messages

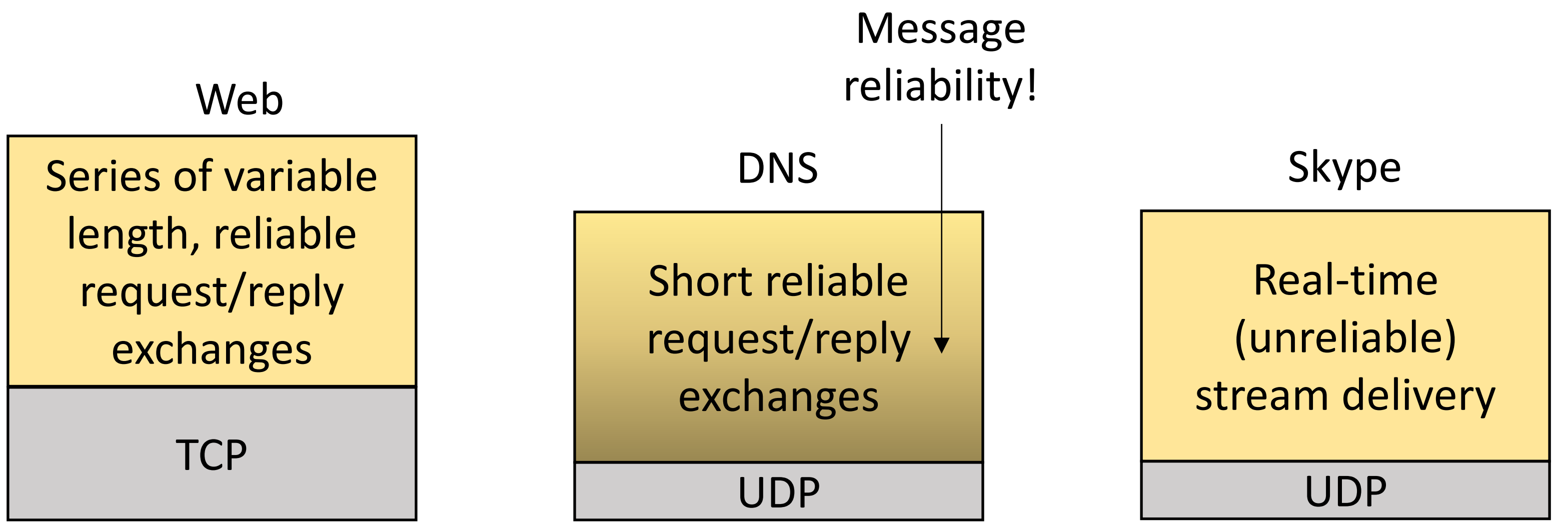
Application layer messages are often split over multiple packets

Or may be aggregated in a packet



Application Communication Needs

Vary widely with app; must build on Transport services



OSI Session/Presentation Layers

Two relevant concepts…

Consider part of the applications, not strictly layered!

Layer	Function
Application	Provides functions needed by users
Presentation	Converts different representations
Session	Manages task dialogs
Transport	Provides end-to-end delivery
Network	Sends packets over multiple links
Data Link	Sends frames of information
Physical	Sends bits as signals

Session Concept

A session is a series of related network interactions in support of an application task

- Often informal, not explicit

Examples:

- Web page fetches multiple resources
- Zoom call involves audio, video, chat

Presentation Concept

Apps need to identify the type of content, and encode it for transfer

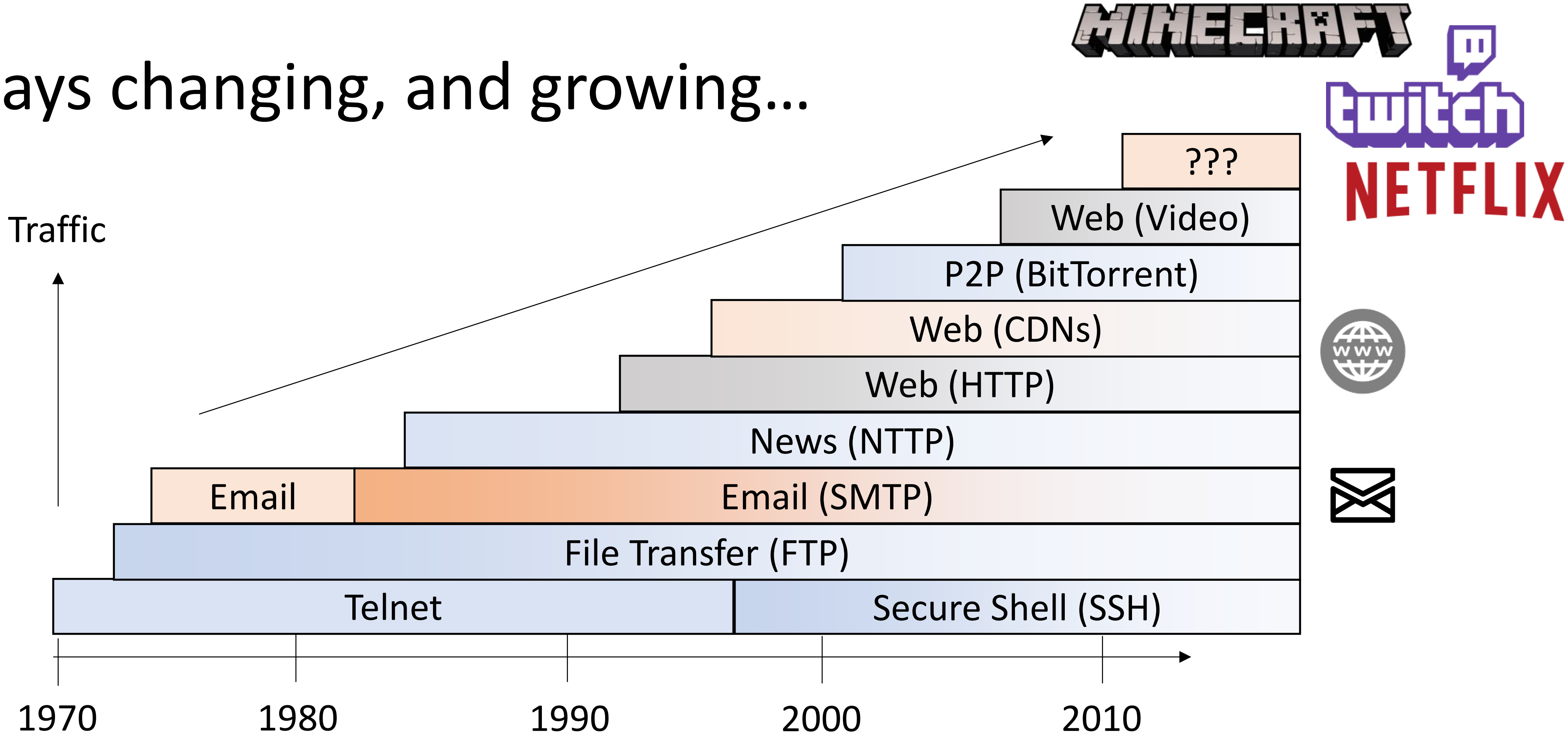
- These are Presentation functions

Examples:

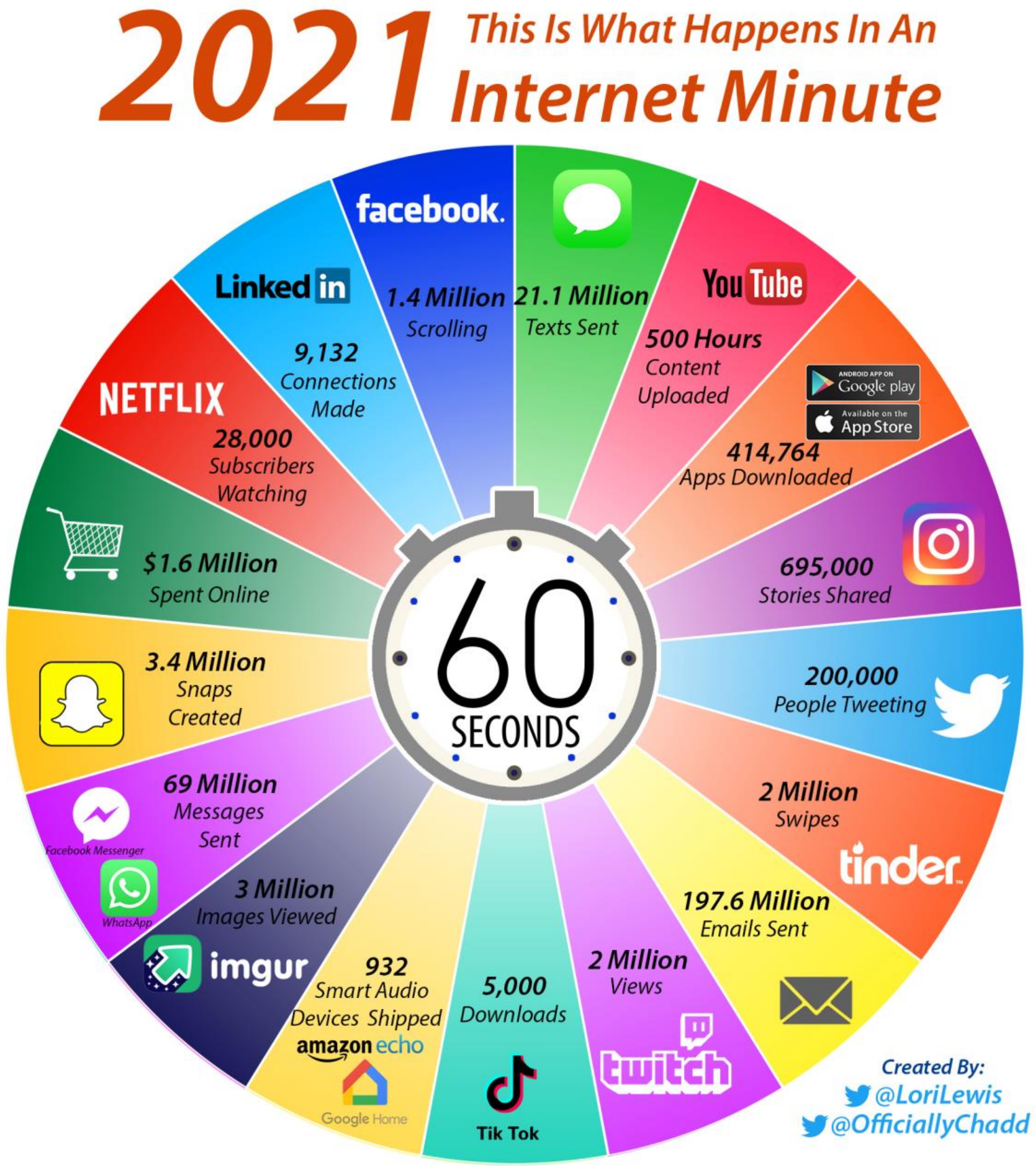
- Media (MIME) types, e.g., image/jpeg, identify the type of content
- Transfer encodings, e.g., gzip, identify the encoding of the content
- Application headers are often simple and readable versus packed for efficiency

Evolution of Internet Applications

Always changing, and growing…



1. 28,000 people watching Netflix
2. 500 hours of content uploaded to YouTube
3. 2 million Twitch views
4. 3.4 million Snaps created



Application Layer Topics

1. **Domain Name System (DNS)**
2. Email
3. Web (HTTP, Web caching/proxy)
4. Multimedia applications

Domain Name System

Port 53

Domain Name System

An application used by the network itself!

Machines on the internet are identified by their ***IP address***

These addresses are difficult for humans to remember!

Q: Can you think of another disadvantage?

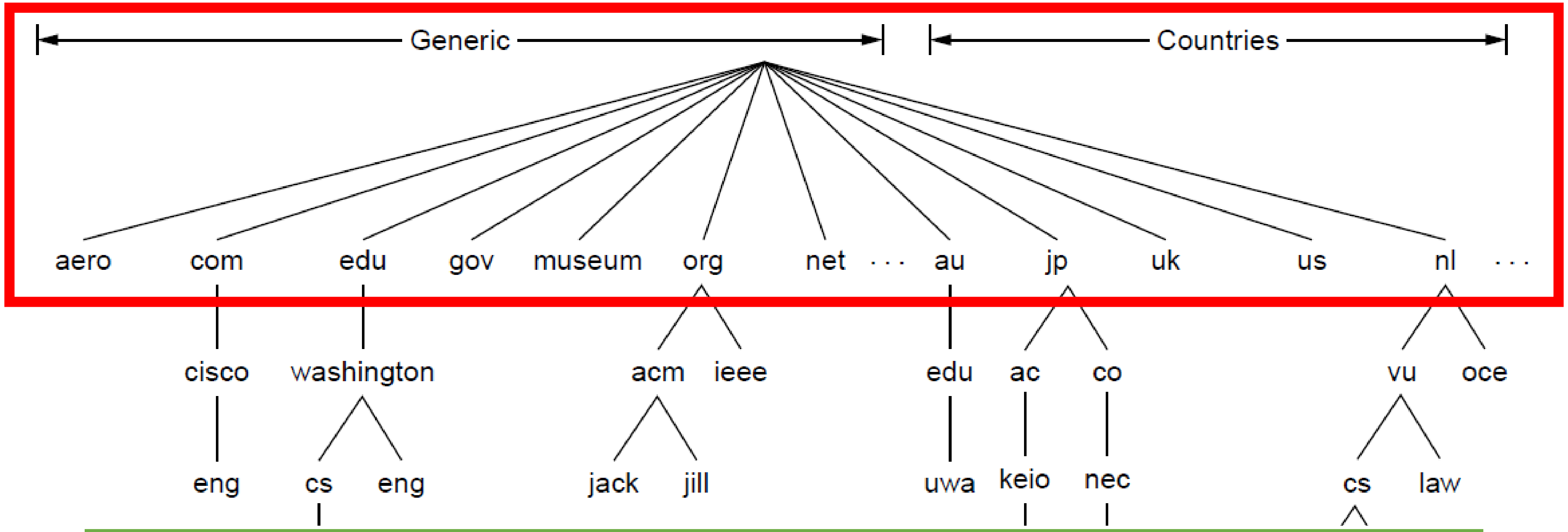
http://4.31.198.44/rfc/rfc1035.txt

j.j.r.donkervliet@131.180.77.82

DNS translates ***human readable names*** to IP addresses

DNS name space

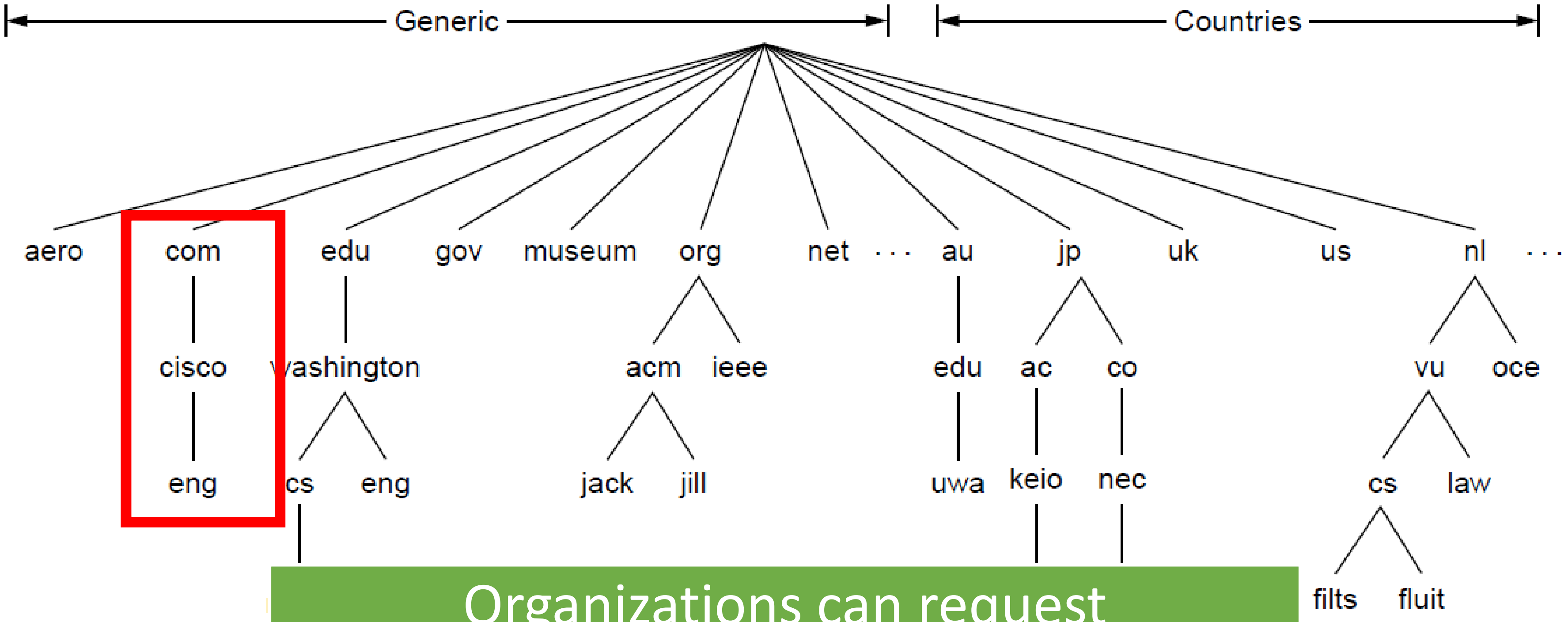
Hierarchical structure.



Top level domains controlled by Internet Corporation for Assigned Names and Numbers (ICANN).

DNS name space

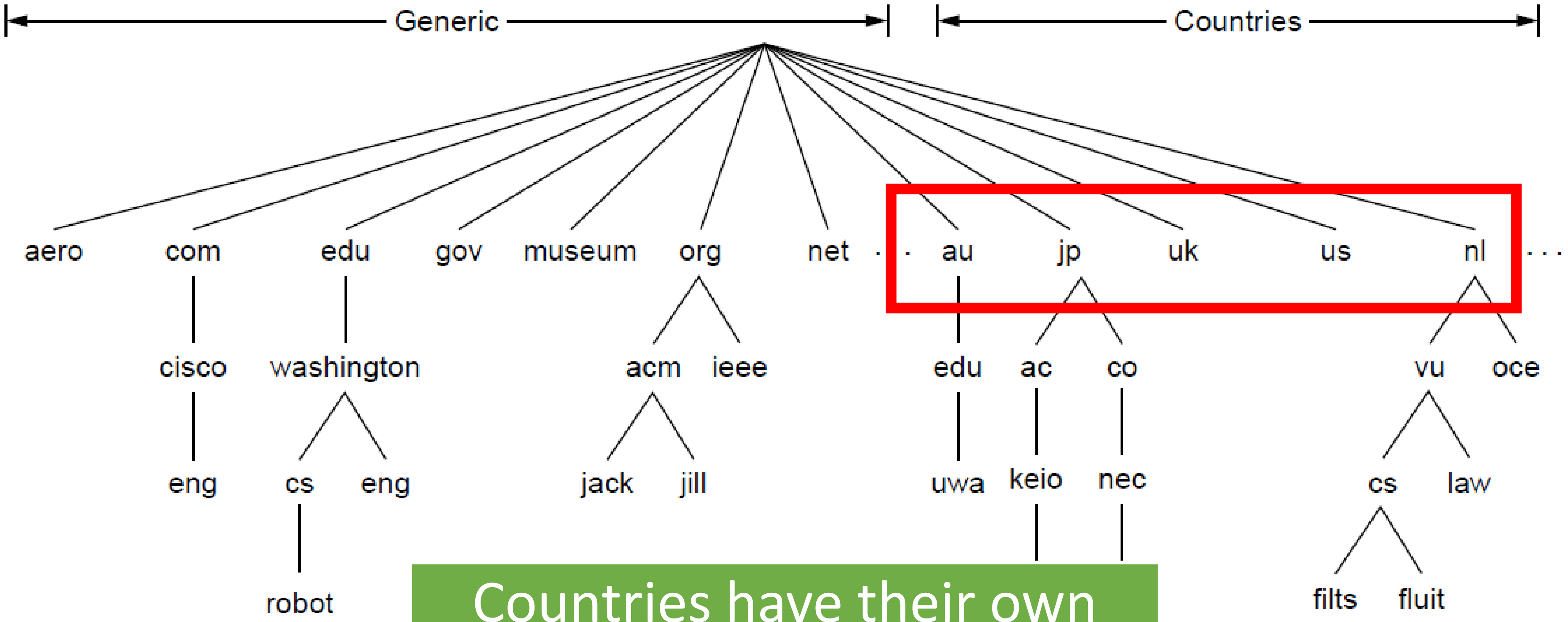
Hierarchical structure.



Organizations can request
second-level domains from ***registrars***.

DNS name space

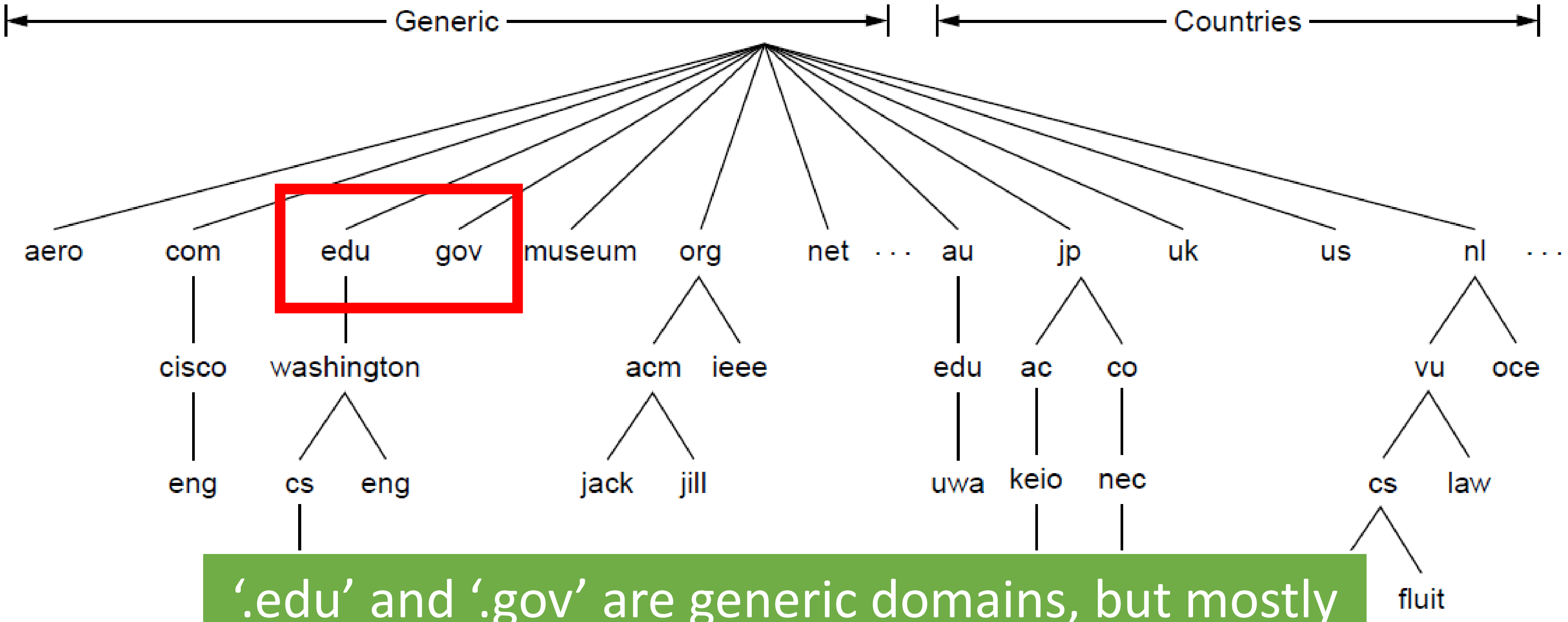
Hierarchical structure.



Countries have their own second-level domains.

DNS name space

Hierarchical structure.



‘.edu’ and ‘.gov’ are generic domains, but mostly used by organizations in the United States.

DNS name space

If you control a domain, you can specify arbitrary subdomains.

United Kingdom uses
ac.uk. for academic use and
co.uk. for commercial use.

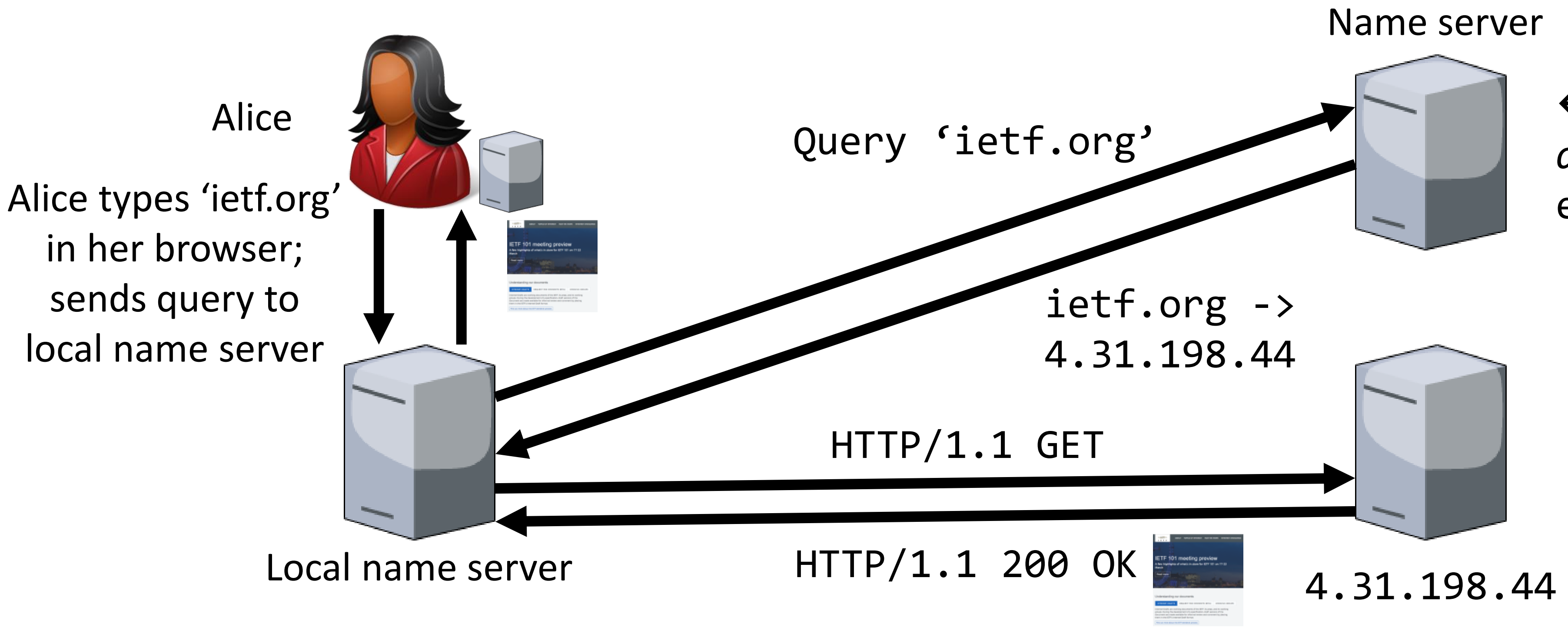
The Netherlands puts everything directly under ***.nl.***.

Name servers

DNS server assignment:	Automatic (DHCP)
Link speed (Receive/Transmit):	100/100 (Mbps)
IPv6 address:	2a02:a446:1d89:1:582d:e6d0:d591:784a
Link-local IPv6 address:	fe80::6eec:4d0f:f0c7:6d35%16
IPv6 DNS servers:	2a02:a47f:e000::53 (Unencrypted) 2a02:a47f:e000::54 (Unencrypted)
IPv4 address:	192.168.2.70
IPv4 DNS servers:	192.168.2.254 (Unencrypted)

Q: How does Alice’s machine know where to find the name server?

To translate a domain name to an IP address, you ask a ***name server***.



Q: How does the name server know the location of ietf.org?

Location of name servers

Hosts learn about the location of name servers via ***DHCP***

The ***operating system*** keeps track of name servers and dynamically selects which one to use

Linux

```
cat /etc/resolv.conf
```

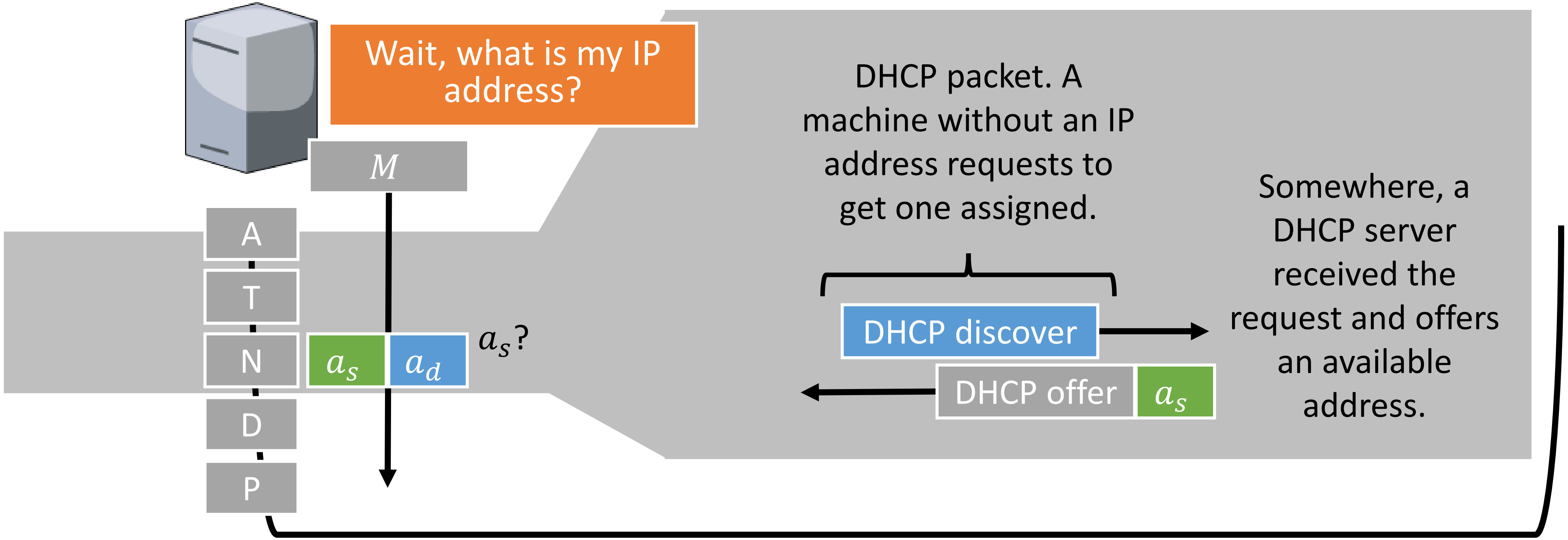
Windows

```
ipconfig /all
```

Dynamic Host Configuration Protocol (DHCP)

MAC addresses are built into NICs. But network addresses are not.

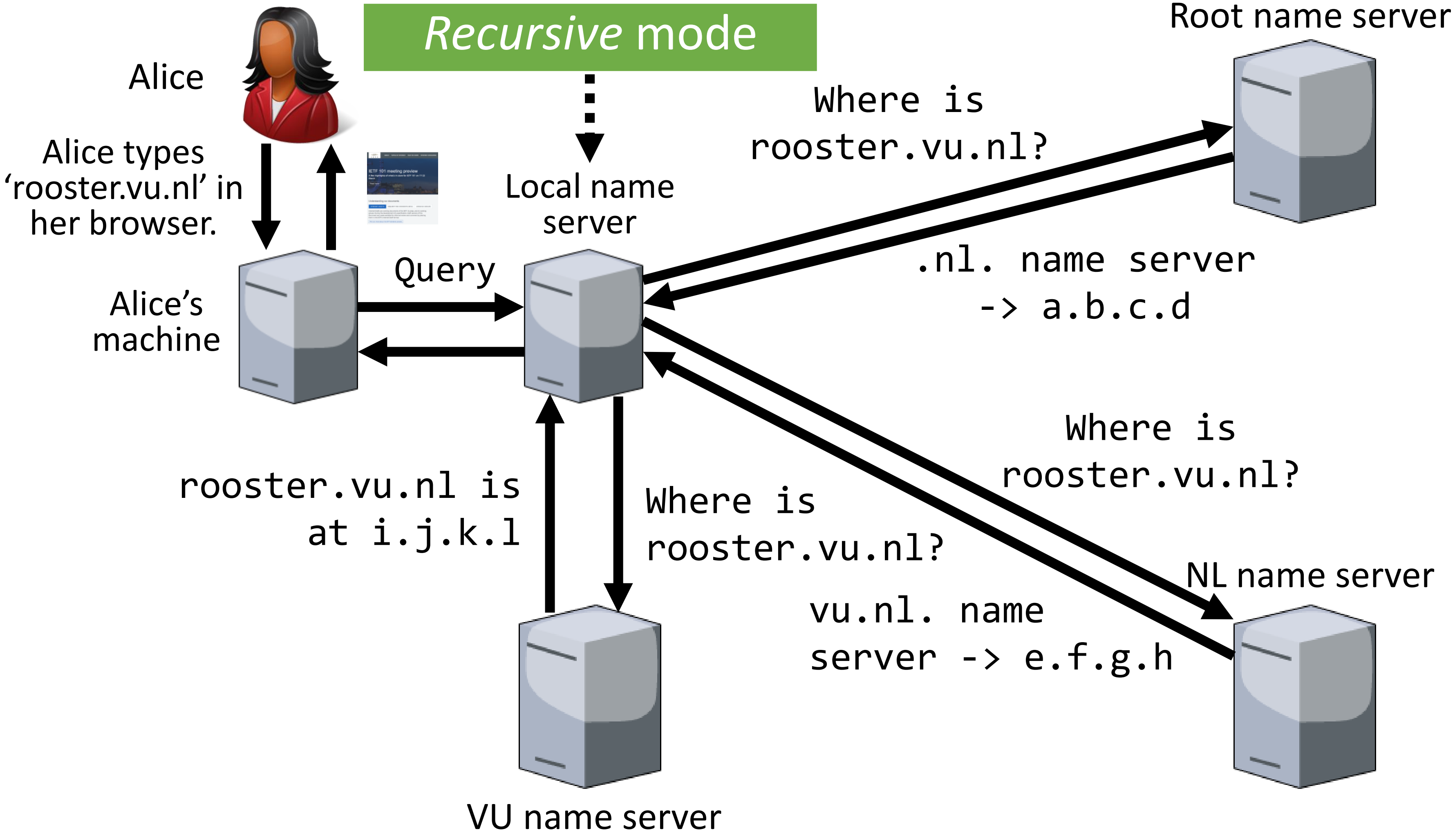
Used to configure other settings such as: ***DNS name servers***, addresses of default gateway, time servers, etc.



Q: How to send DHCP offer back to machine without an address?

Other name servers are in *iterative* mode

Recursive and iterative queries



```
(base) jesse@Jesses-XPS-2022:~$ dig canvas.vu.nl

; <<>> DiG 9.16.1-Ubuntu <<>> canvas.vu.nl
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41657
;; flags: qr rd ad; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;canvas.vu.nl.                  IN      A

;; ANSWER SECTION:
canvas.vu.nl.           0       IN      CNAME   vu-vanity.instructure.com.
vu-vanity.instructure.com. 0    IN      CNAME   canvas-dub-prod-c84-1303699784.eu-west-1.elb.amazonaws.com.
canvas-dub-prod-c84-1303699784.eu-west-1.elb.amazonaws.com. 0 IN A 52.17.144.218
canvas-dub-prod-c84-1303699784.eu-west-1.elb.amazonaws.com. 0 IN A 54.216.29.136
canvas-dub-prod-c84-1303699784.eu-west-1.elb.amazonaws.com. 0 IN A 54.77.55.232

;; Query time: 9 msec
;; SERVER: 172.31.224.1#53(172.31.224.1)
;; WHEN: Mon May 15 14:47:46 CEST 2023
;; MSG SIZE  rcvd: 284
```

DNS Resource Record (RR) Types

Name servers reply with ***domain resource records***.
A record can contain:

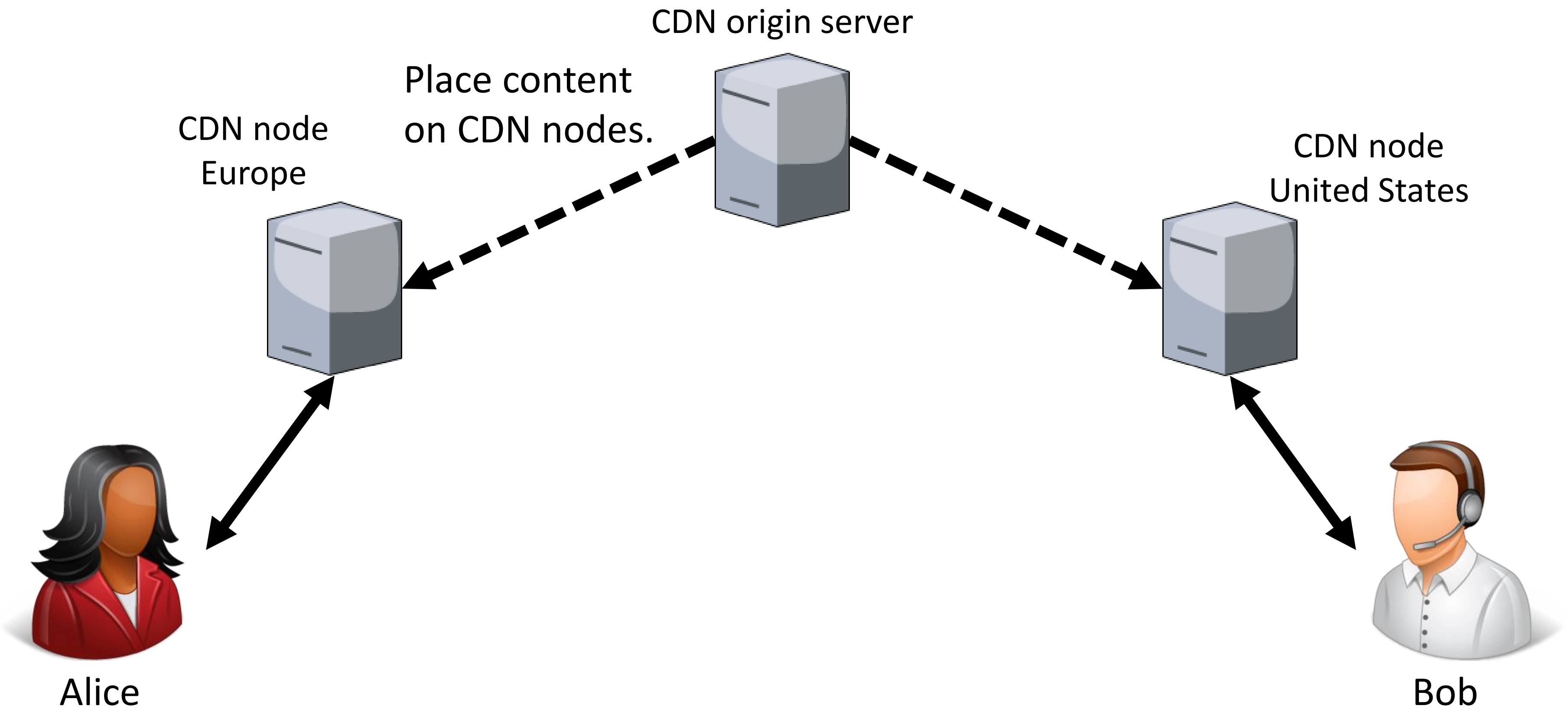
1. **IPv4 address** (record type A)
2. **IPv6 address** (record type AAAA)
3. Domain that accepts **email** (record type MX)
4. **Name server** for this domain (record type NS)
5. **Alias** to Canonical Name (record type CNAME)
6. …

Content Delivery Networks

Content delivery networks

Q: How to make sure users do not all contact the same node?

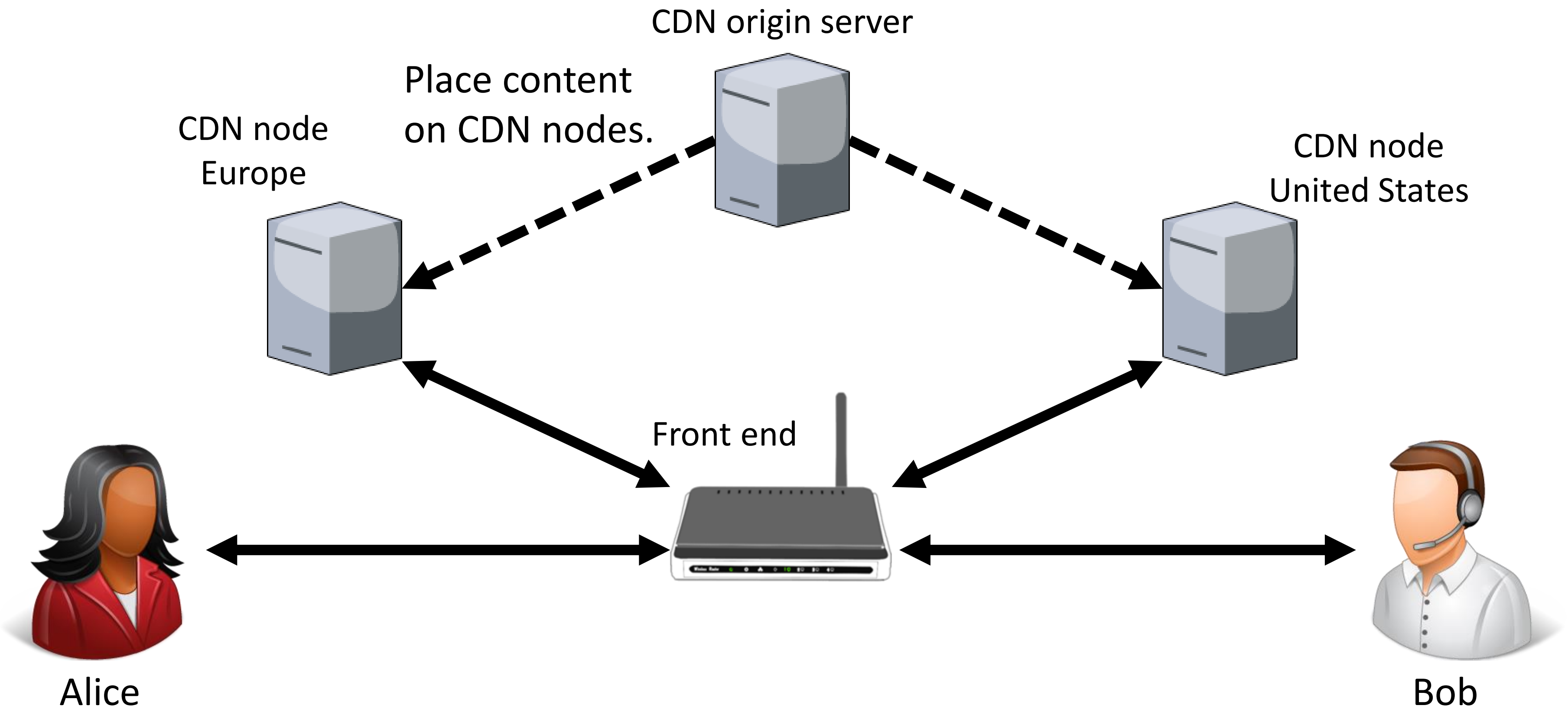
A type of ***caching*** to increase system scalability.



Content delivery networks

Front end forwards requests and distributes load

A type of ***caching*** to increase system scalability.

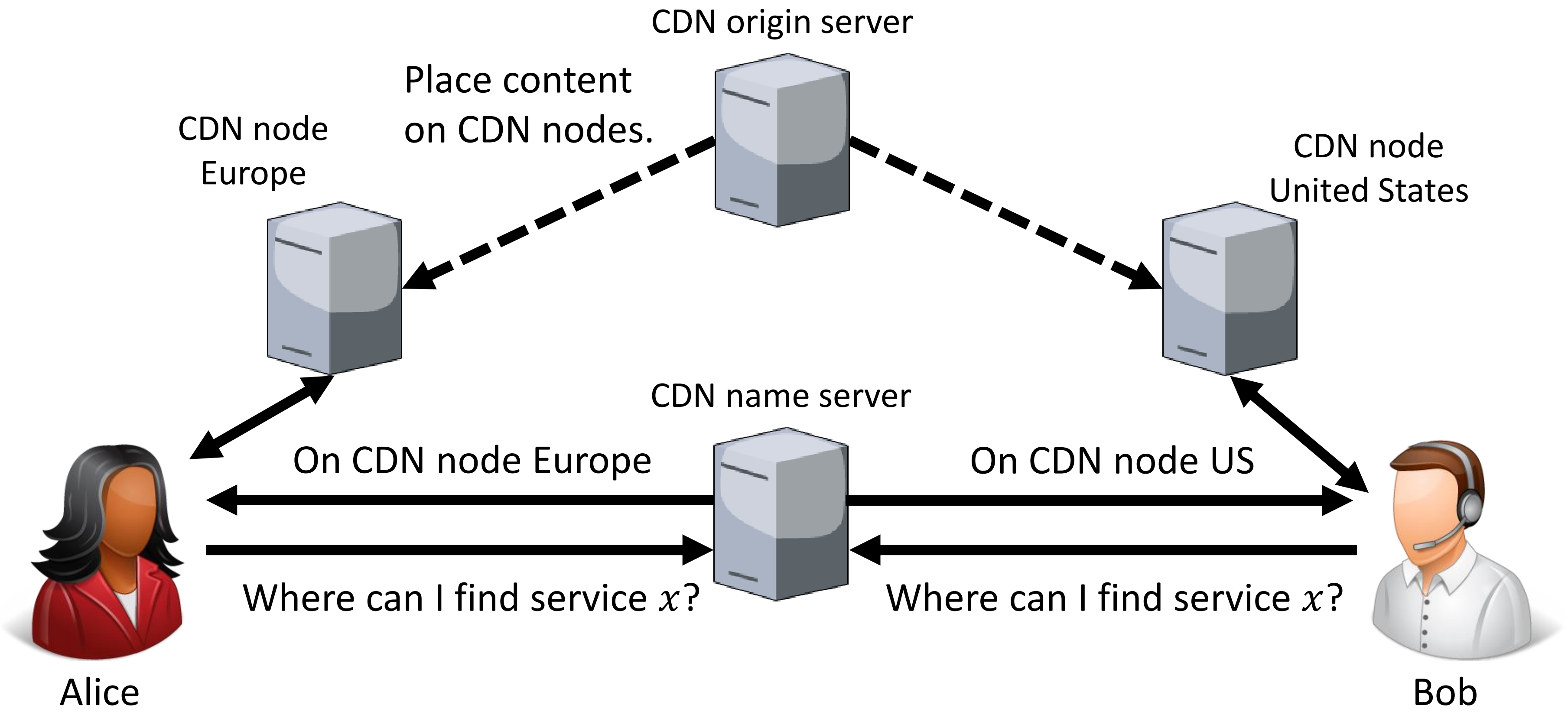


Content delivery networks

Powered by DNS!

DNS can be used for load balancing!

A type of ***caching*** to increase system scalability.



Content delivery networks

Powered by DNS!

```
$ dig @192.5.6.30 ibm.com

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;ibm.com.                        IN      A

;; AUTHORITY SECTION:
ibm.com.         172800  IN      NS      usw2.akam.net.
ibm.com.         172800  IN      NS      usc2.akam.net.
ibm.com.         172800  IN      NS      eur2.akam.net.
ibm.com.         172800  IN      NS      ns1-99.akam.net.
ibm.com.         172800  IN      NS      ns1-206.akam.net.
ibm.com.         172800  IN      NS      asia3.akam.net.
ibm.com.         172800  IN      NS      usc3.akam.net.
ibm.com.         172800  IN      NS      eur5.akam.net.
```

Content delivery networks

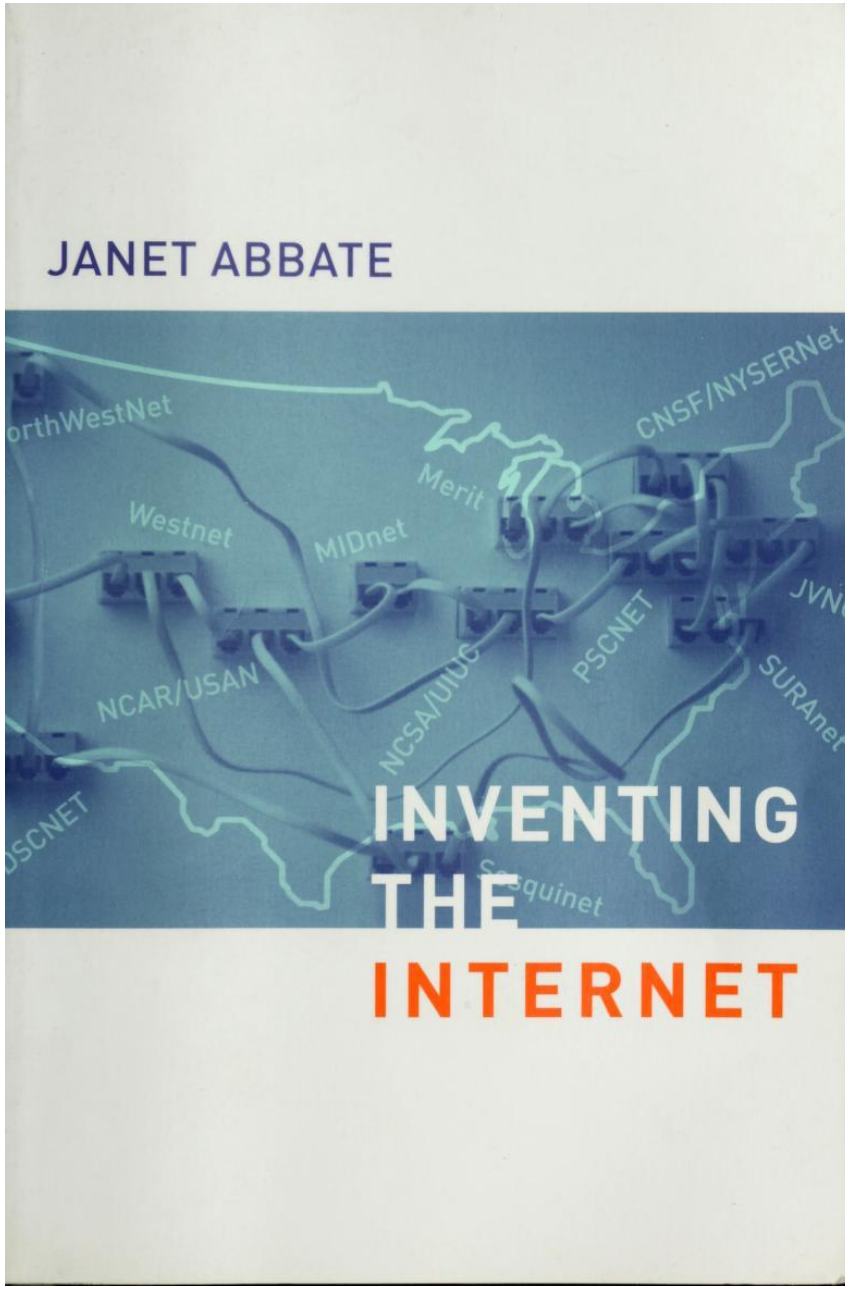
Powered by DNS!

```
$ dig @192.5.6.30 ibm.com

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;
;; QUESTION SECTION:
;ibm.com.

;; AUTHORITY SECTION:
ibm.com.        172800  IN      NS      usw2.akam.net.
ibm.com.        172800  IN      NS      usc2.akam.net.
ibm.com.        172800  IN      NS      eur2.akam.net.
ibm.com.        172800  IN      NS      ns1-99.akam.net.
ibm.com.        172800  IN      NS      ns1-206.akam.net.
ibm.com.        172800  IN      NS      asia3.akam.net.
ibm.com.        172800  IN      NS      usc3.akam.net.
ibm.com.        172800  IN      NS      eur5.akam.net.
```

‘akam’ means ‘Akamai’, a CDN company.

Application Layer Topics

1. Domain Name System (DNS)
2. **Email**
3. Web (HTTP, Web caching/proxy)
4. Multimedia applications



Email

Not too long ago, email was all we had! Now, more options are available.

Almost ***200 million*** emails sent every minute!

- 9 out of 10 emails are spam!

You can send and receive email on ***your own*** domain.

Or you can use a (free) email service provided by a company or organization:

1. http://www.visualcapitalist.com/happens-internet-minute-2017/
2. *McAfee Threat Reports: First Quarter 2010*, McAfee Inc., 2010.

Metcalfe’s Law

The value of a network is proportional to the square of the number of users.

(I.e., value is proportional to the number of possible connections.)

As networks get larger, there is more value in joining them, making them larger, …

We don’t know what will be tomorrow’s network applications, but we know that these networks will continue growing.

Mobile, Internet of Things (IoT), …

Email
Message formats

Envelope is used to get message to correct recipient.

Messages contain:

1. An envelope
2. A header
3. A body

From: Alice
To: Bob
Encryption: None

From: Alice
To: Bob
Subject: How does email work?

Hi Bob,

I really want to know how email works. Do you know of any CS courses I could follow to learn more about it?

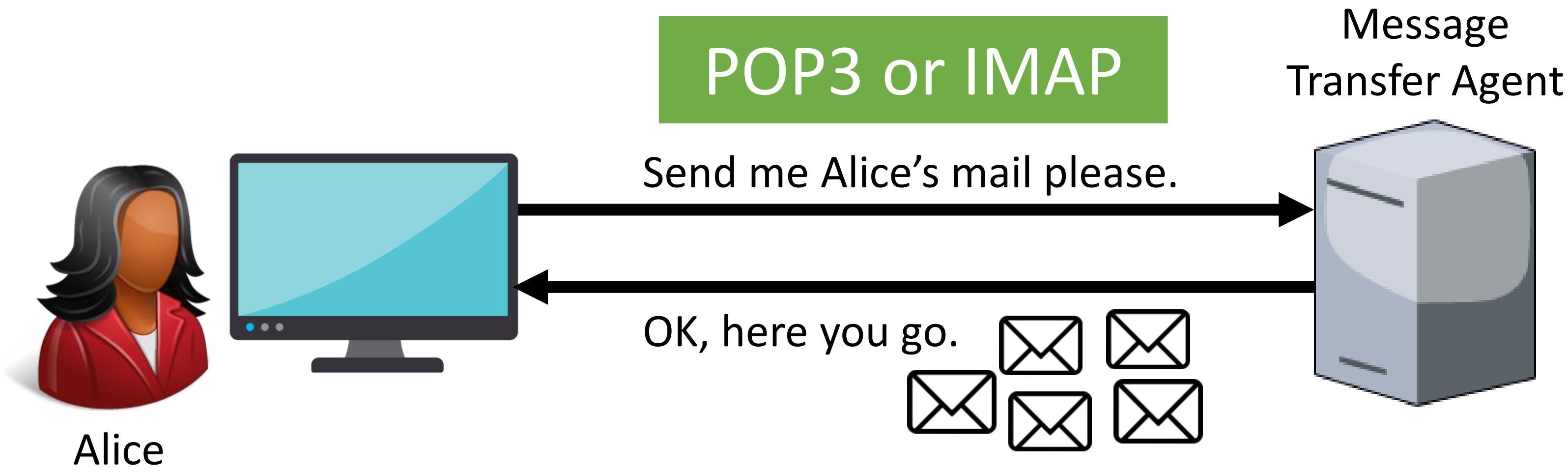
Other helpful headers:

Message-Id, In-Reply-To, Reply-To, …

Email
How does it work?

Email uses multiple protocols:

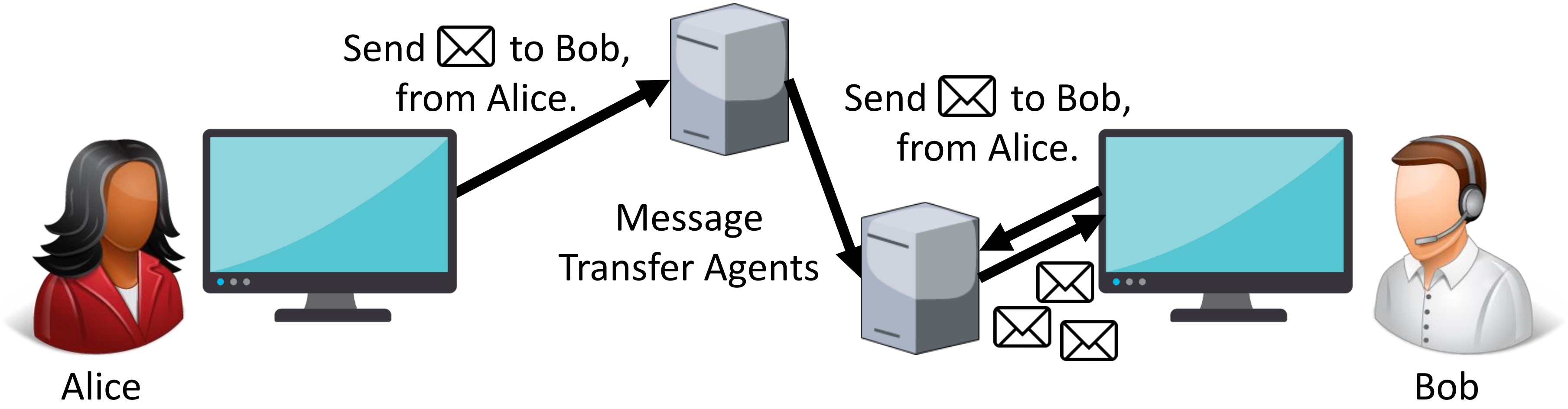
1. Users use ***POP3*** or ***IMAP*** to interact with their ***mailbox***.



Email
How does it work?

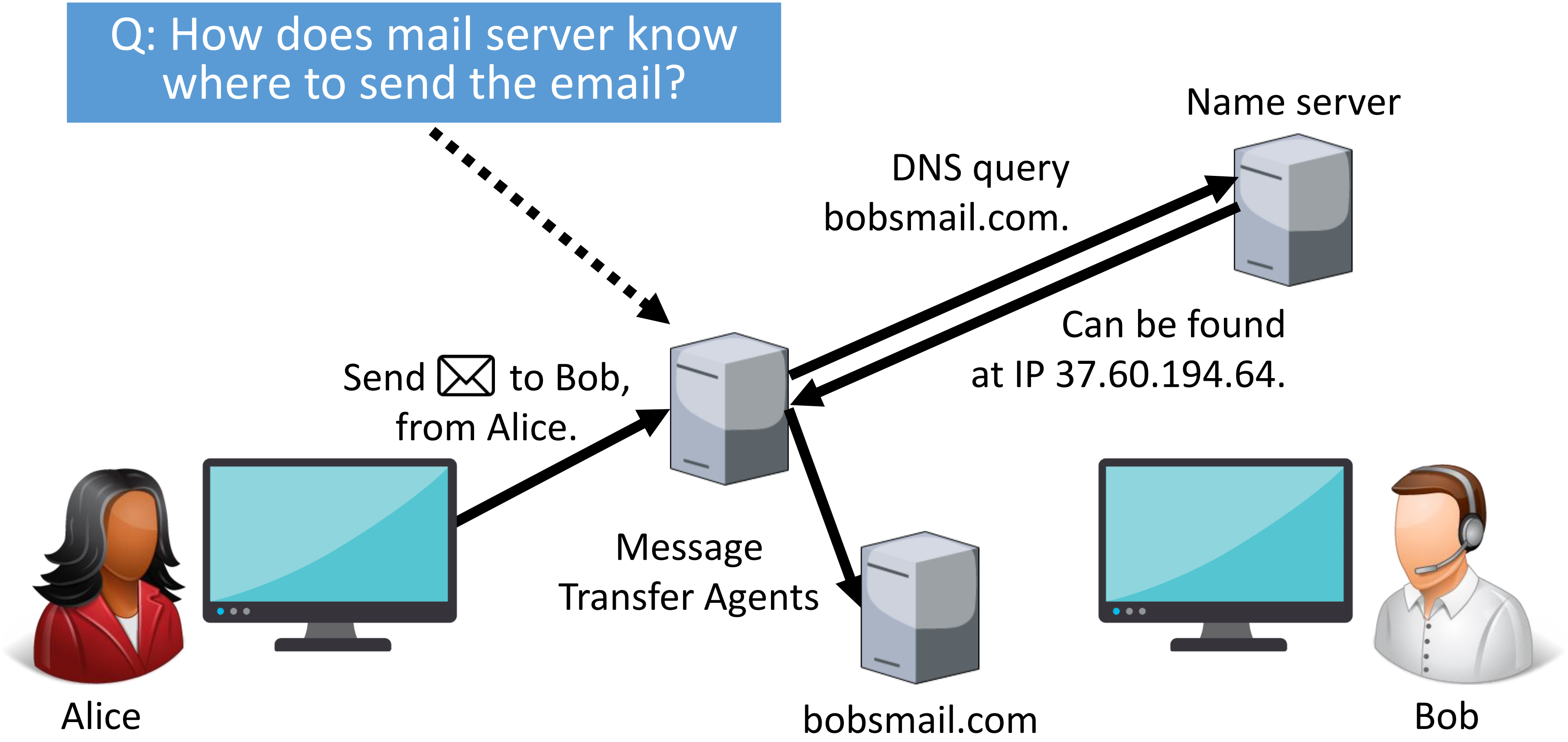
Email uses multiple protocols:

1. Users use ***POP3*** or ***IMAP*** to interact with their ***mailbox***.
2. Users and ***Message Transfer Agents*** use ***SMTP*** to send email from a source to a destination.



Email
How does it work?

Powered by DNS!

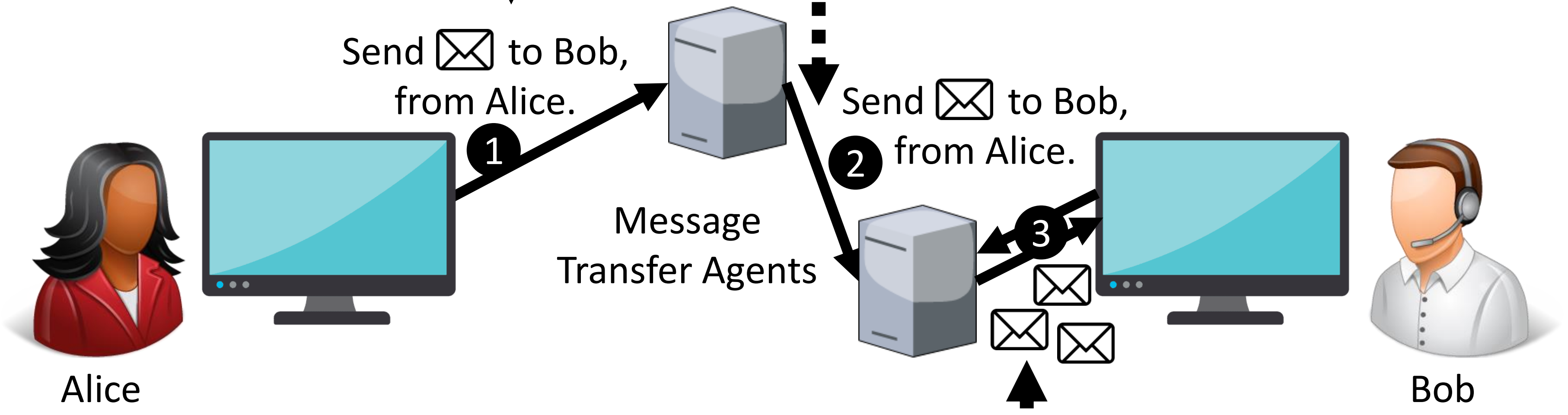


Email
How does it work?

Q: Example of a proprietary protocol used for ***final delivery***?

Mail submission uses SMTP + Extensions (e.g. AUTH).

Message transfer between mail servers uses SMTP.



Final delivery uses IMAP/POP3 or a propriety protocol.

Internet Message Access Protocol (IMAP)

Q: Does **`gmail.com`** use POP3 or IMAP?

RFC 9000

Ports: 143, 993

Sends commands to ***mail server*** to manipulate mailboxes

Common commands:

1. LOGIN. Log into server
2. FETCH. Fetch messages from a folder
3. CREATE/DELETE. Create or delete a folder
4. EXPUNGE. Remove messages marked for deletion

Uses mostly plain text!

Security through TLS (not covered in the course)

Replaced POP3 protocol

Simple Mail Transfer Protocol (SMTP)

RFC 5321

Ports: 25, 587

SMTP uses ASCII

You can use TELNET to talk to a mail server!

```
       S: 220 ee.uwa.edu.au SMTP service ready
C: HELO abcd.com
       S: 250 cs.washington.edu says hello to ee.uwa.edu.au
C: MAIL FROM: <alice@cs.washington.edu>
       S: 250 sender ok
C: RCPT TO: <bob@ee.uwa.edu.au>
       S: 250 recipient ok
C: DATA
       S: 354 Send mail; end with "." on a line by itself
C: …
```

FROM field ***not*** checked!

Basic SMTP does not support binary data!

Basic SMTP does not include authentication!

Many extensions exist to address these issues.

Multipurpose Internet Mail Extensions (MIME)

Developed for email, now used more broadly

Adds headers to email:

```
MIME-Version
Content-Description
Content-Id
Content-Transfer-Encoding
Content-Type
```

If MIME-Version in header
 check Content-Type
Else
 plain text

MIME Content-Type

1. Text:	text/plain,	text/html
2. Images:	image/jpeg,	image/gif
3. Video:	video/mp4,	video/mpeg
4. Multipart:	multipart/mixed,	multipart/alternative

Used to create messaged with multiple data types
(e.g., an email with attachment).

Basic SMTP does not support binary data!

Multipurpose Internet Mail Extensions (MIME)

Developed for email, now used more broadly

Adds headers to email:

```
MIME-Version
Content-Description
Content-Id
Content-Transfer-Encoding
Content-Type
```

If MIME-Version in header
  check Content-Type
Else
  plain text

Modern SMTP protocol supports binary data

Sending binary data via ASCII-only SMTP

When MIME was introduced, servers were not expecting non-ASCII data.

Q: How to send binary via a server that can only handle ASCII?

1010010101010010101 Encoder BARhucehboc20Au

data

Base64 encoding converts binary data into ASCII

Base64 encoding

Q: How large is the overhead of base64 encoding?

Used to convert binary data to and from ASCII.

Alphabet: `[A-Za-z0-9+/]`

6 bits are translated into 1 character.

ASCII to ASCII example

	C	a	t
ASCII	01000011	01100001	01110100

↓ Base64 encoding

	Q	2	F	0
ASCII	010000	110110	000101	110100

Base64 encoding

binary to ASCII example

0x0A0D7F

Binary	0x0A		0x0D	0x7F
	00001010		00001101	01111111
	000010	100000	110101	111111
ASCII	C	g	1	/

000000: A
000001: B
000010: C
…
011001: Z
011010: a
011011: b
…
110011: z
110100: 0
110101: 1
…
111101: 9
111110: +
111111: /

Base64 encoding

Only 4 bits left!

Missing bits set to 0

Padded 2 bits

Binary	0x0A		0x0D	
	00001010		00001101	
	000010	100000	110100	
ASCII	C	g	0	=

000000: A
000001: B
000010: C
…
011001: Z
011010: a
011011: b
…
110011: z
110100: 0
110101: 1
…
111101: 9
111110: +
111111: /

Base64 encoding

Binary

0x0A

00001010

Missing bits set to 0

000010 100000

Padded 4 bits

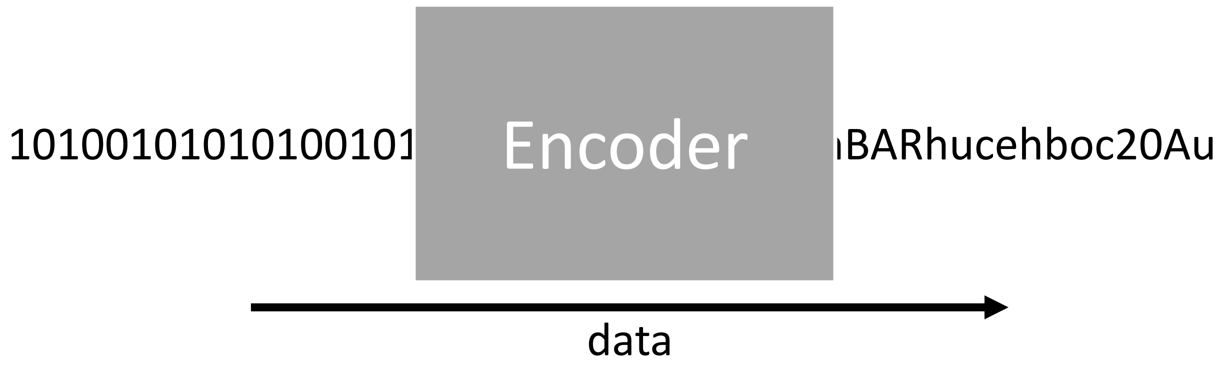
ASCII

C g = =

000000: A
000001: B
000010: C
…
011001: Z
011010: a
011011: b
…
110011: z
110100: 0
110101: 1
…
111101: 9
111110: +
111111: /

Base64 encoding to send arbitrary data types

1. Text: text/plain, text/html
2. Images: image/jpeg, image/gif
3. Video: video/mp4, video/mpeg
4. ***Multipart***: multipart/mixed, multipart/alternative



The Web provides a common interface to our digital society

Application Layer Topics

1. Domain Name System (DNS)
2. Email
3. **Web (HTTP, QUIC, WebSocket)**
4. Multimedia applications

HTTP in The World Wide Web

Hypertext

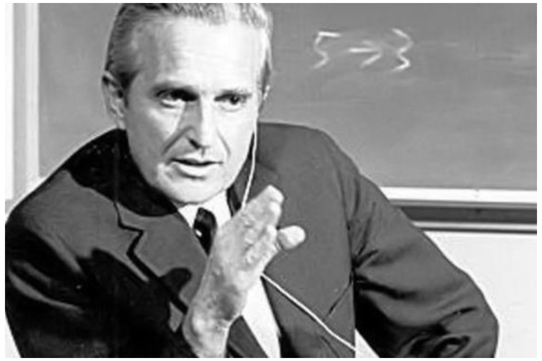
Vannevar Bush described the Memex, a device for storing data *associatively*

The idea existed before digital computers and digital media (e.g., libraries)

Vannevar Bush

Hypertext invented by Ted Nelson and Douglas Engelbart

Ted Nelson

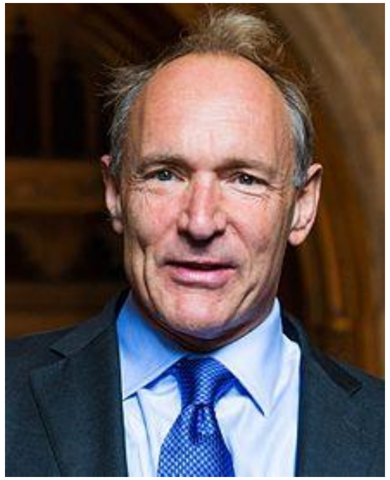


Douglas Engelbart

The Web
TCP+DNS+Hypertext

Tim Berners-Lee, a computer engineer at CERN, started the modern Web by combining TCP, DNS, and hypertext in 1989

He now directs the World Wide Web Consortium (W3C)



Tim Berners-Lee

Page last updated at 10:52 GMT, Wednesday, 14 October 2009 11:52 UK

E-mail this to a friend

Printable version

Berners-Lee 'sorry' for slashes

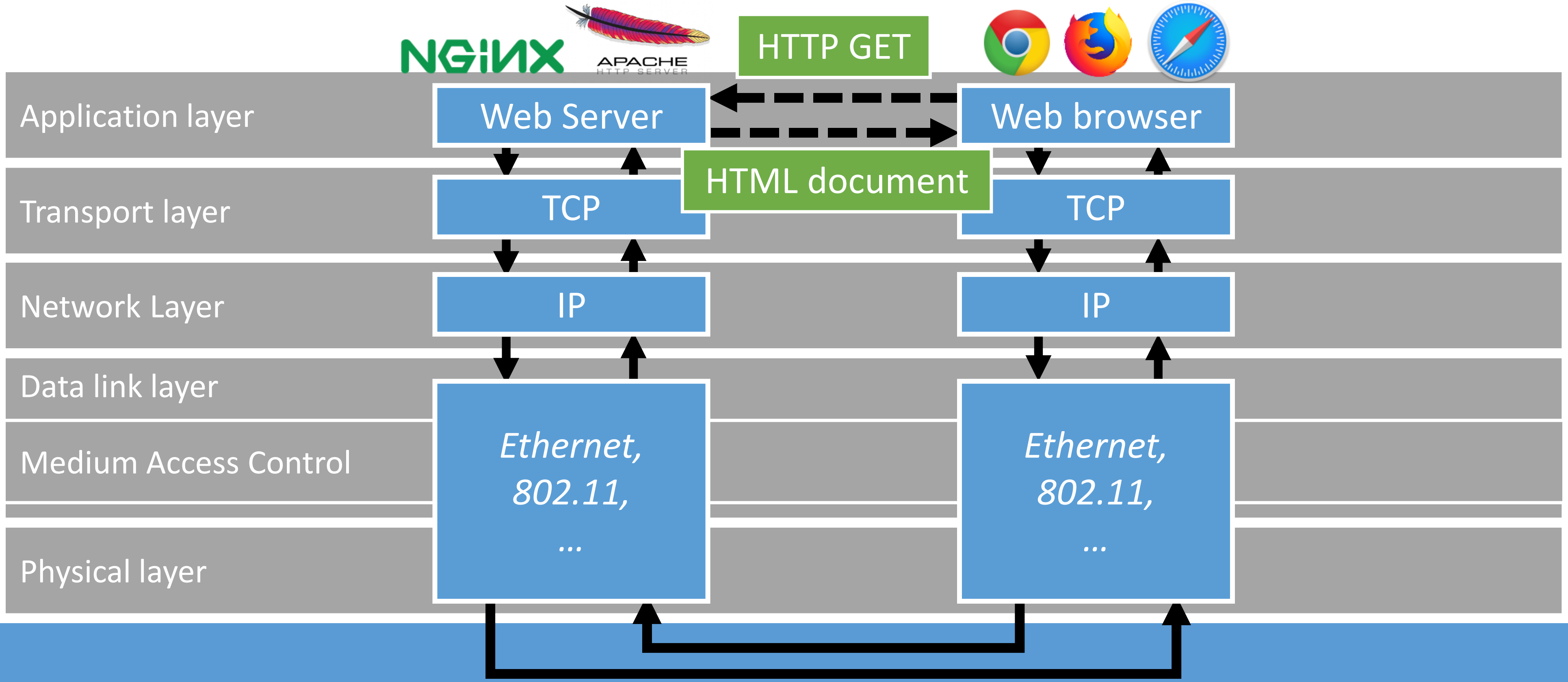
The forward slashes at the beginning of internet addresses have long annoyed net users and now the man behind them has apologised for using them.

Sir Tim Berners-Lee, the creator of the World Wide Web, has confessed that the // in a web address were actually "unnecessary".

Tim Berners-Lee started the web to help scientists communicate

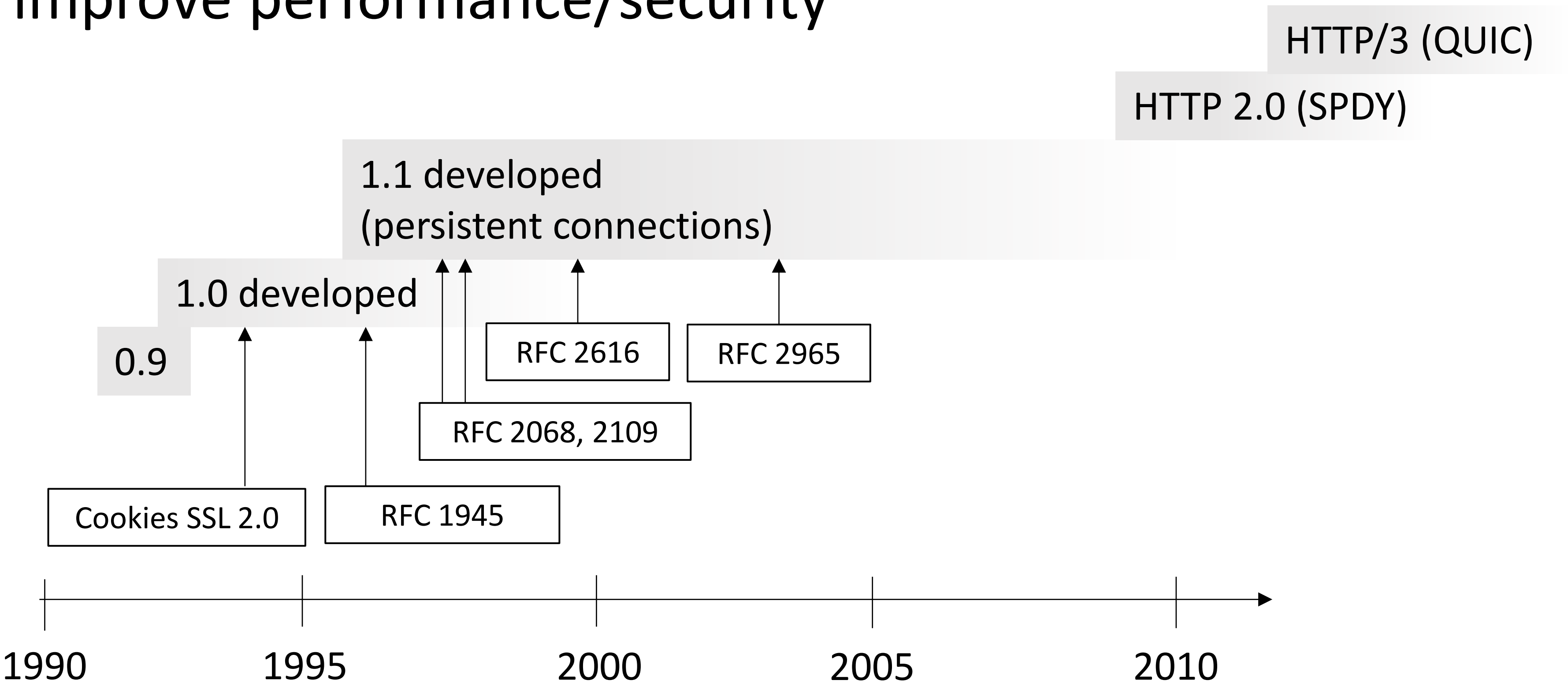
HTTP Request/Response

HTML documents hosted by servers.
Clients sends request for document from server.



Evolution of HTTP

Optimizations are gradually incorporated to improve performance/security



HTTP Protocol

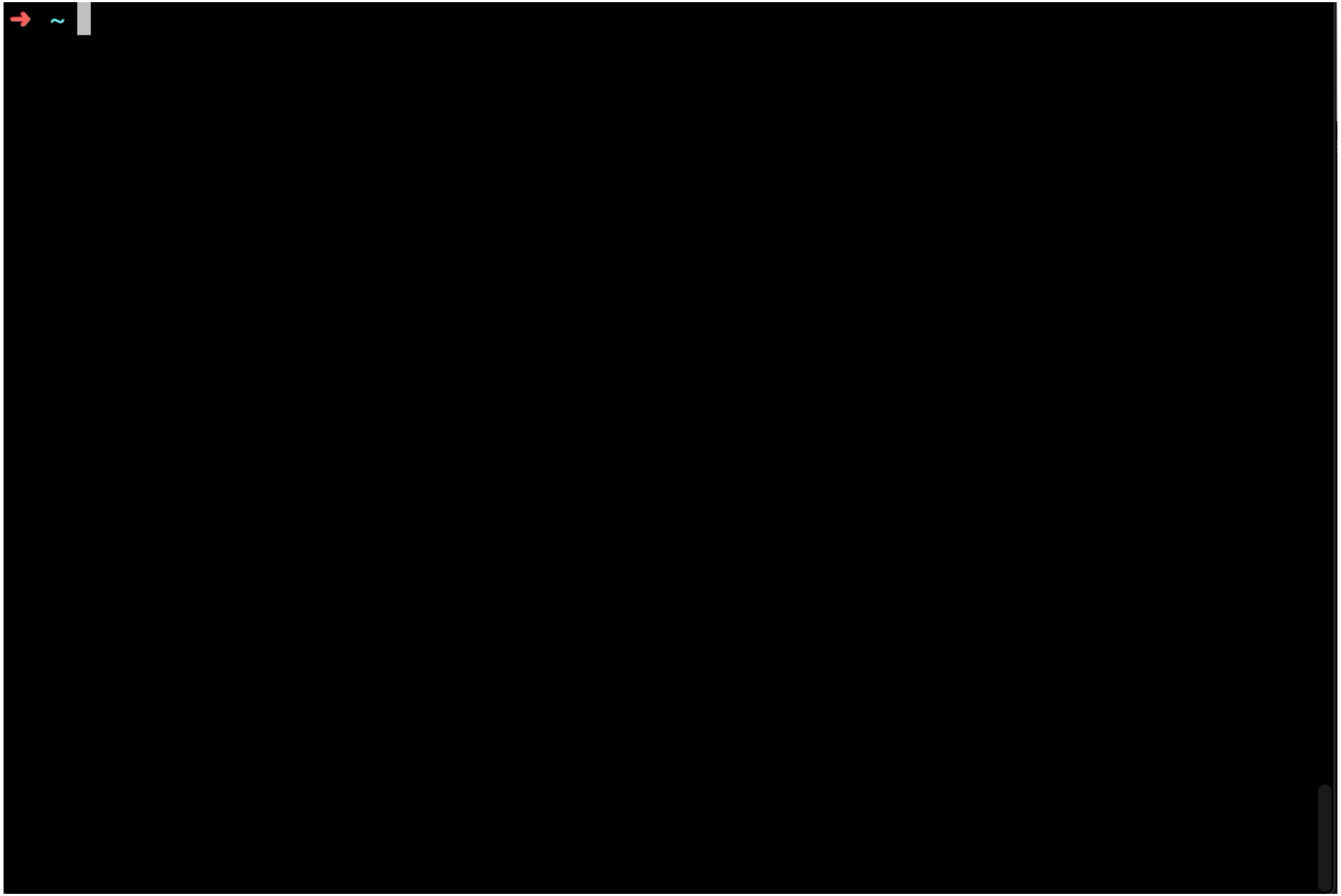
Similar to chat application from the lab!

Originally a simple text-based protocol
Many options added over time

Try it yourself:

```
$ telnet en.wikipedia.org 80
GET wiki/HTML HTTP/1.0
```

HTTP Request via TELNET



HTTP Request Methods

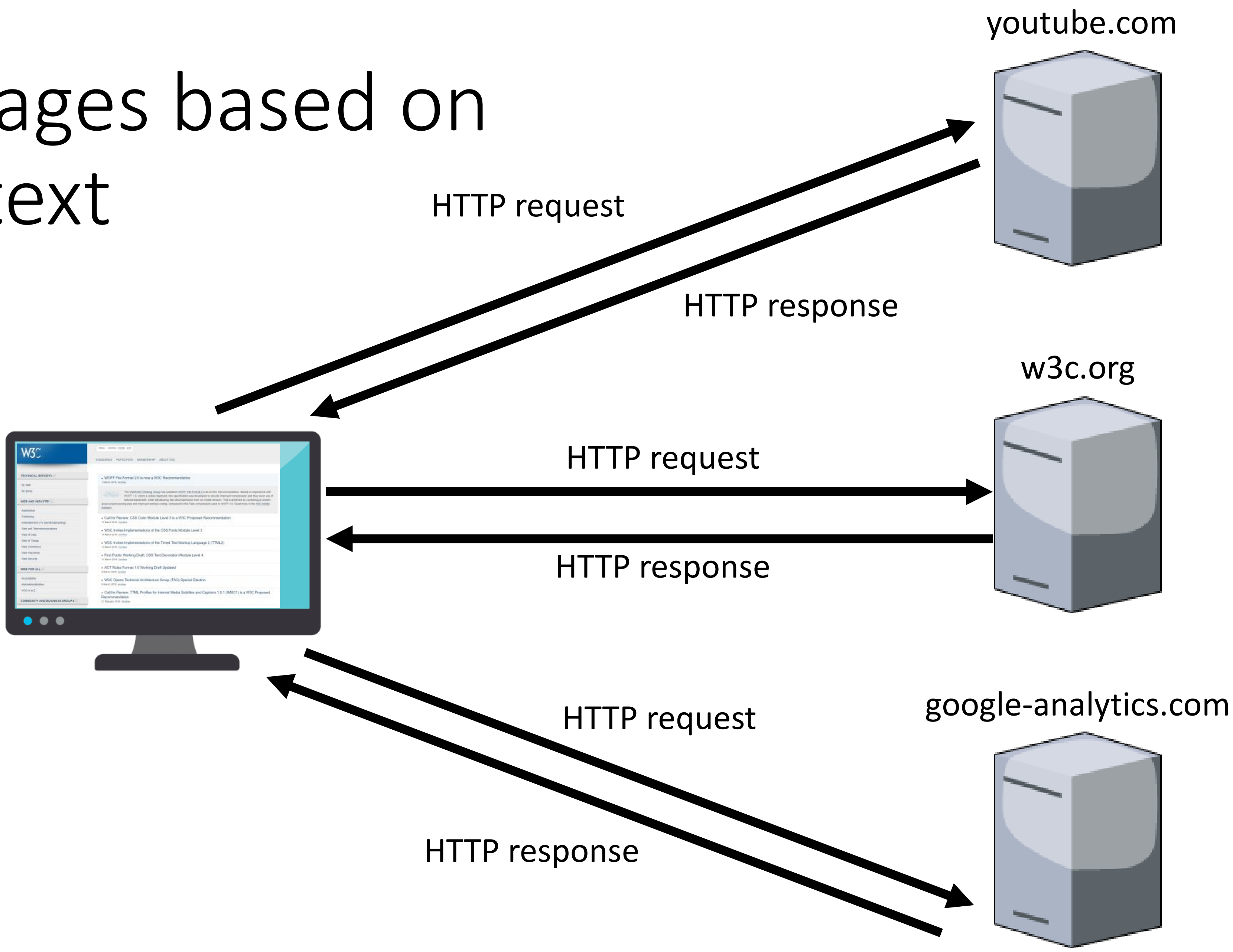
Methods: GET, POST, PUT, HEAD, …

```
$ curl -v -L --http1.1 https://vu.nl -o /dev/null
…
> GET / HTTP/1.1
> Host: vu.nl
> User-Agent: curl/7.64.1
> Accept: */*
>
…
```

https://**www.w3.org**/**TR/2010/WD-html5-20100624/**

Specifies the **protocol**, the **domain name**, and a **path**.

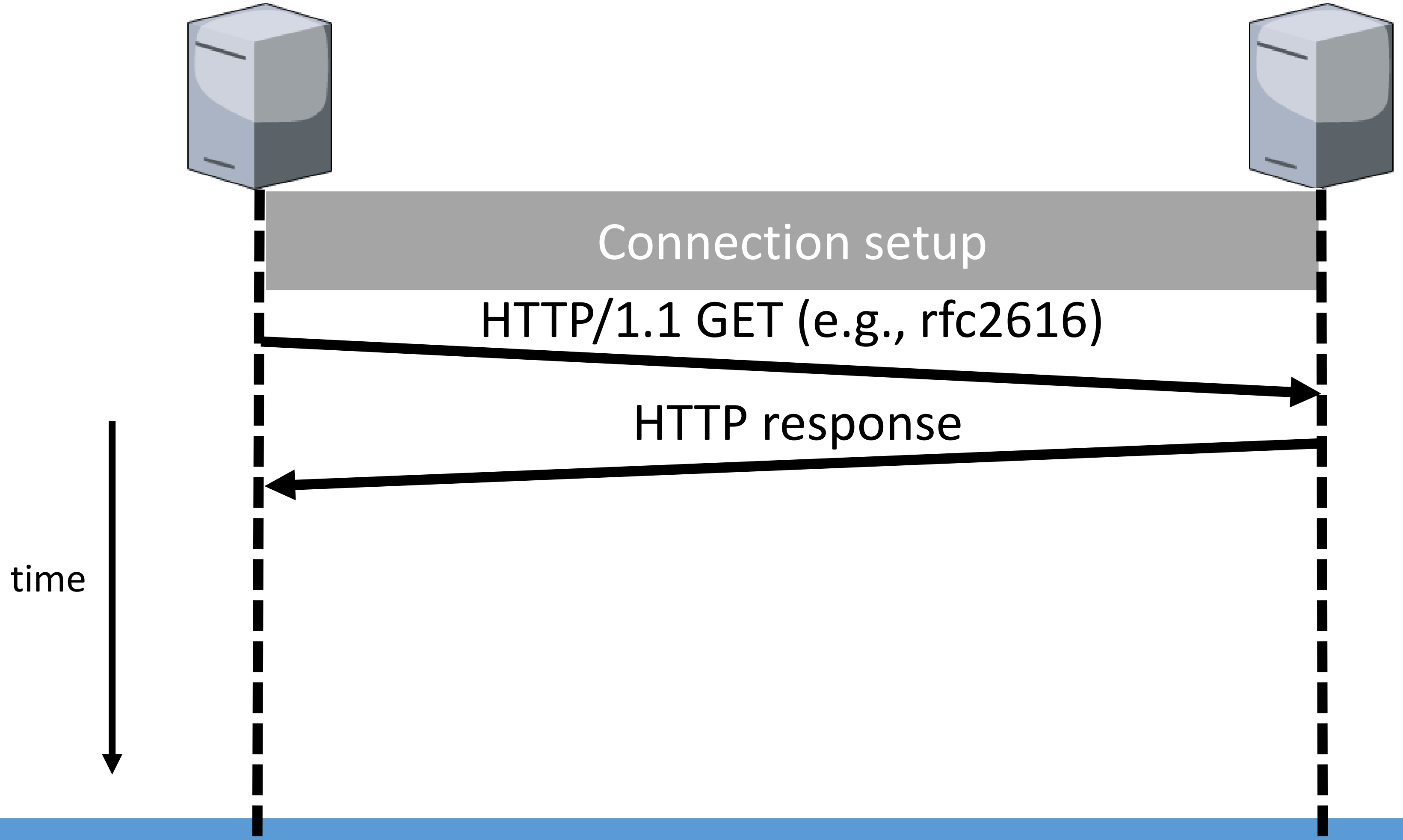
Web pages based on Hypertext



Web and HTTP Performance

The Web and HTTP continues to evolve, with servers sending ***more*** and ***larger*** responses

Single document

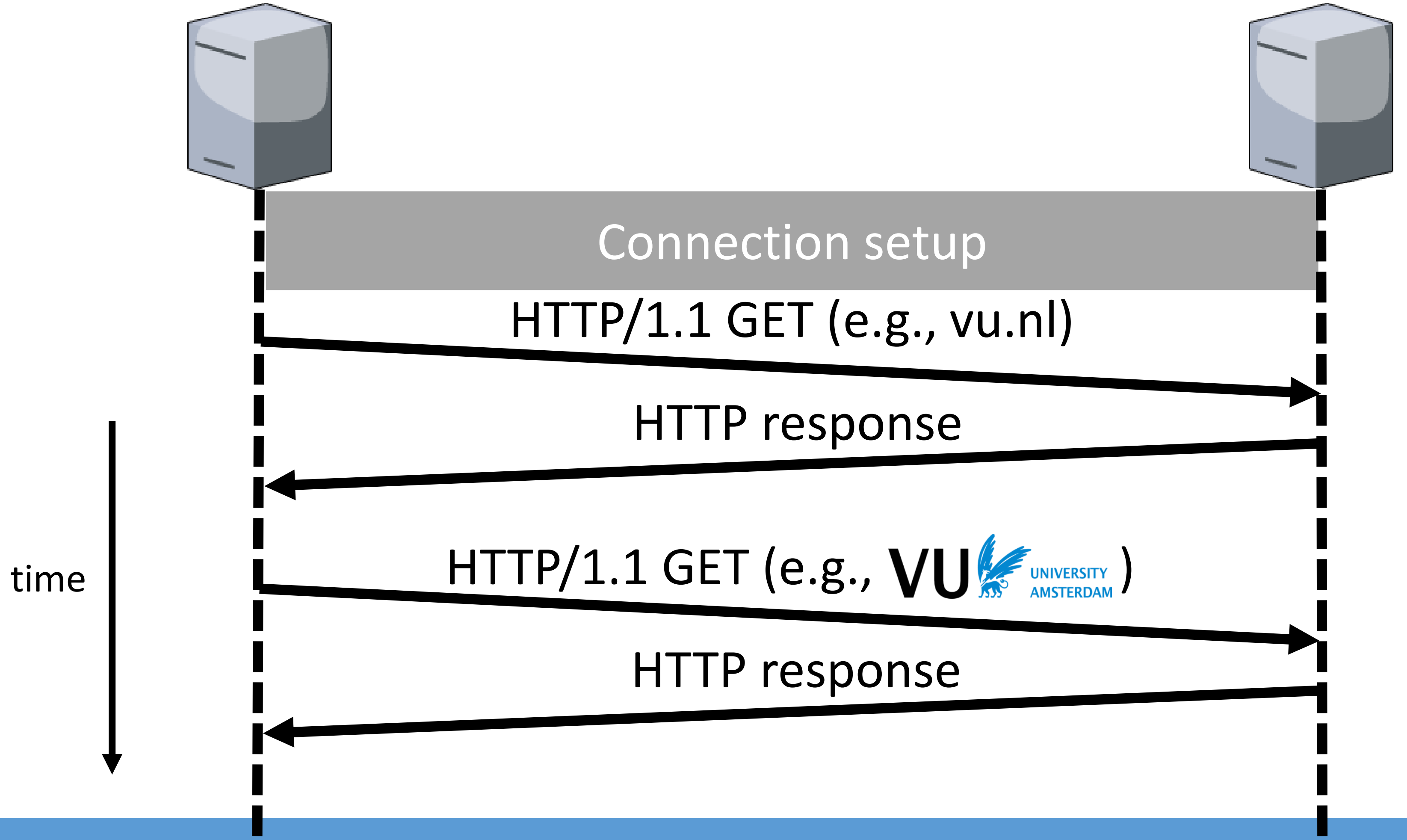


Single document Example

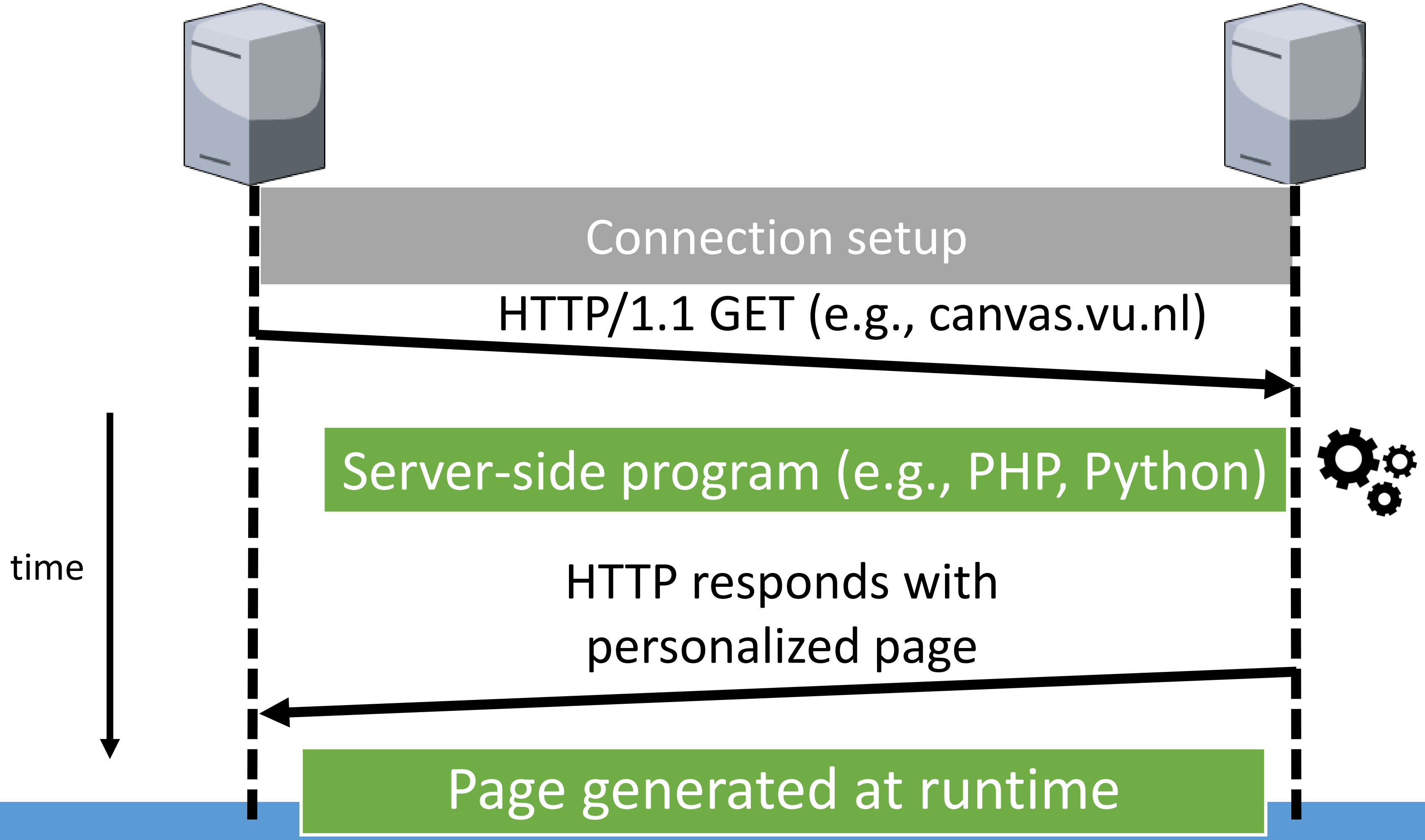
https://www.w3.org/Protocols/rfc2616/rfc2616.html

Name	Domain	Type	Transfer Size	Time

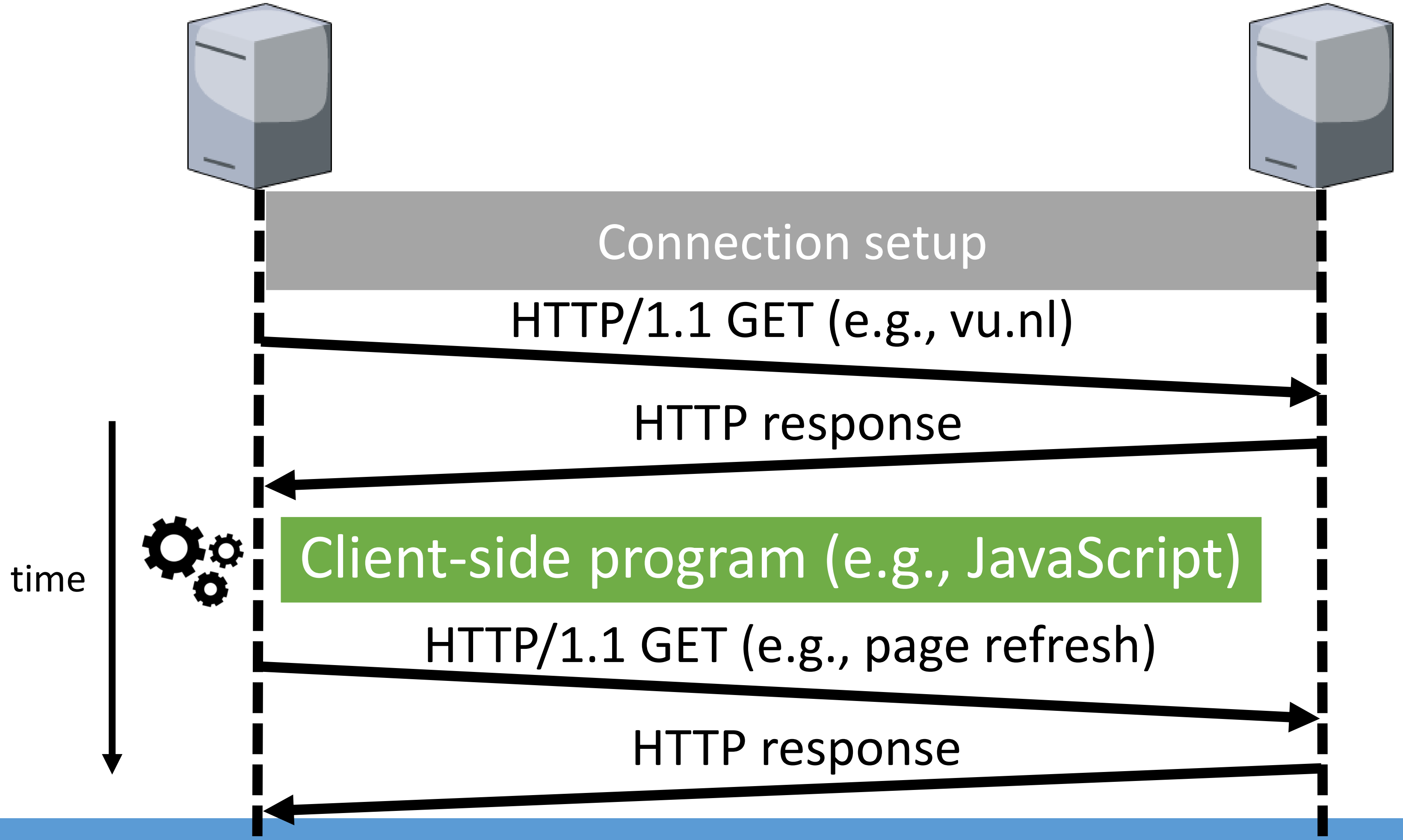
External resources



Server-side programs



Client-side programs

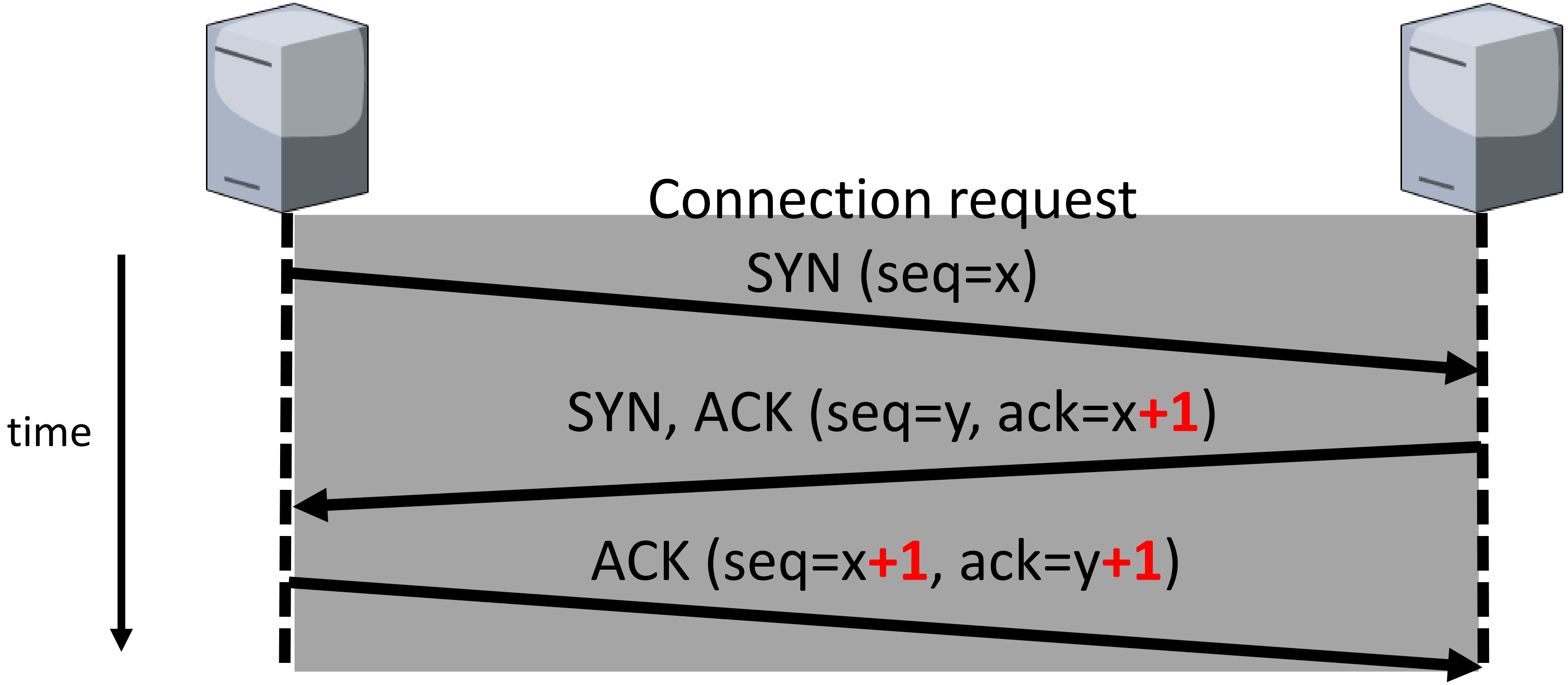


Modern webpages
Many requests

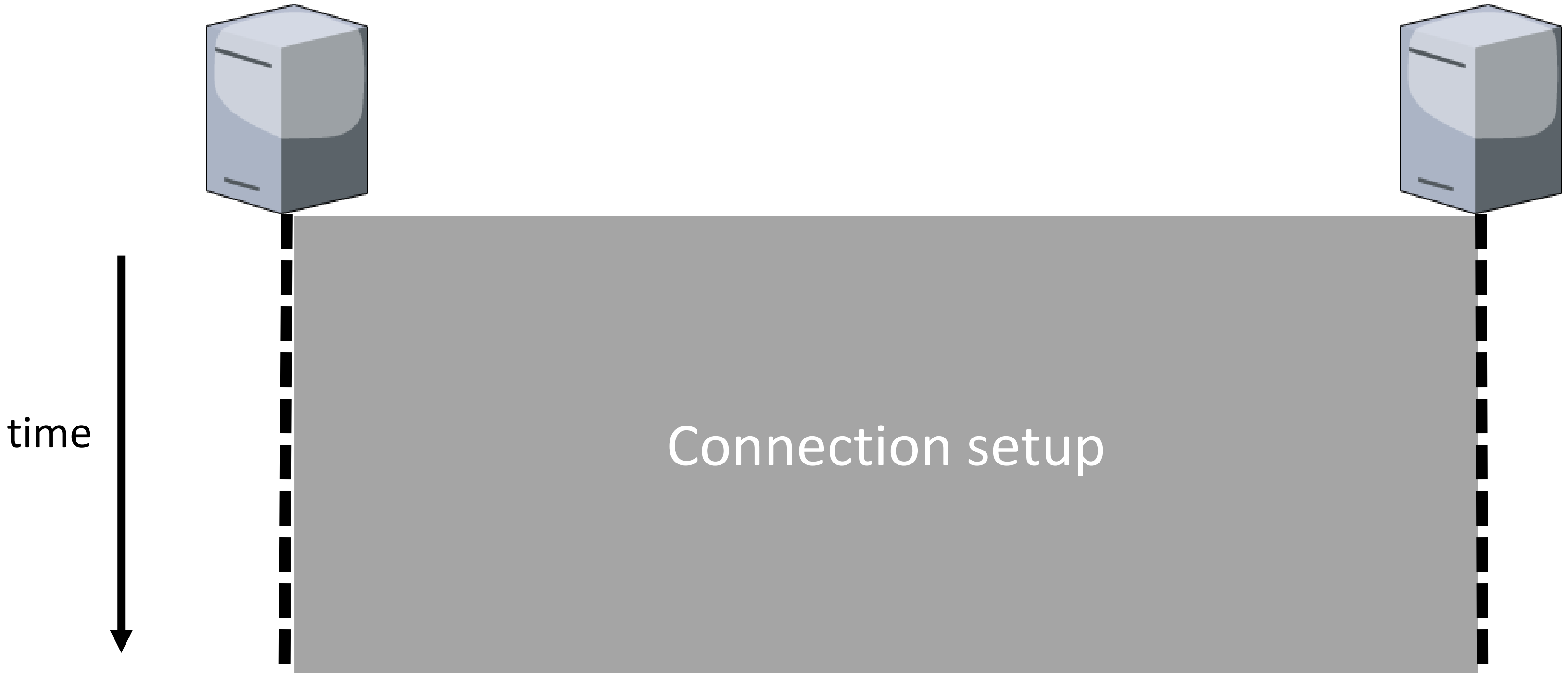
https://canvas.vu.nl/

Name	Domain	Type	Transfer Size

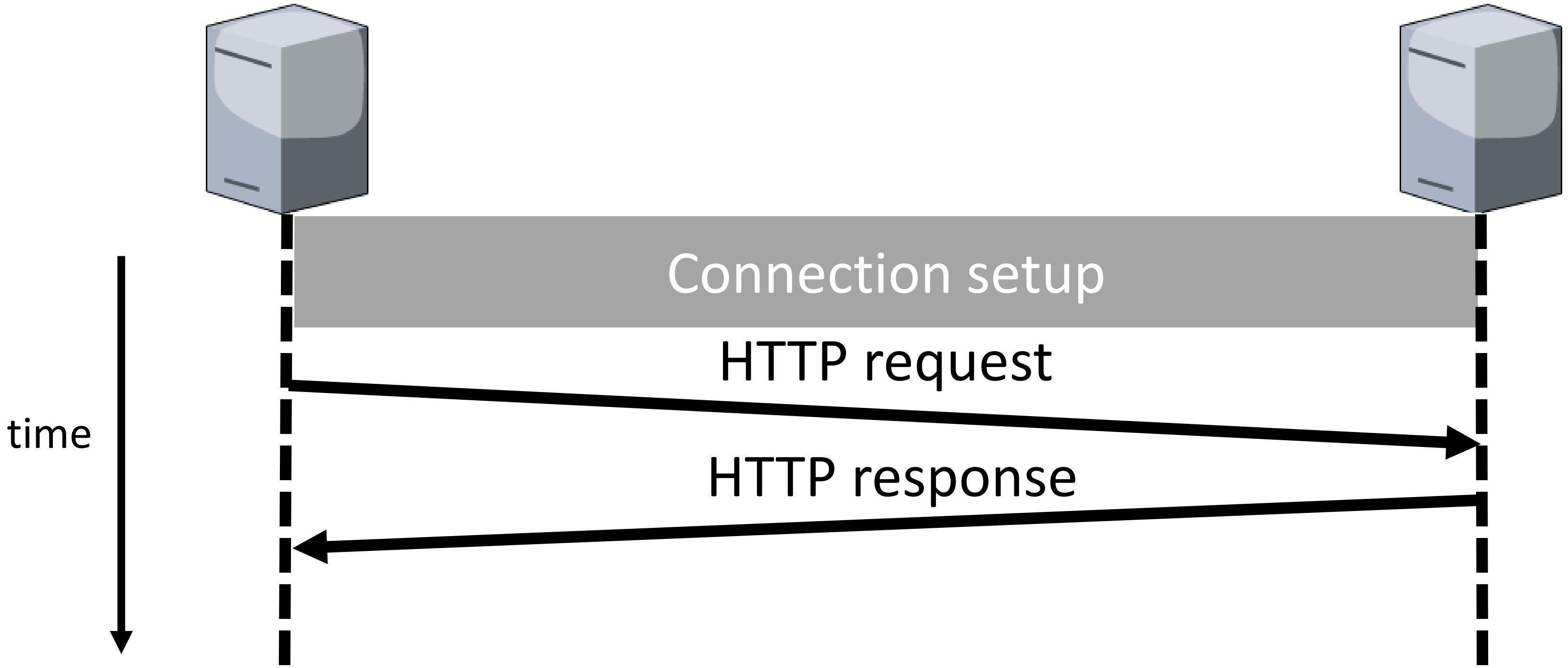
Recap
TCP Connection setup



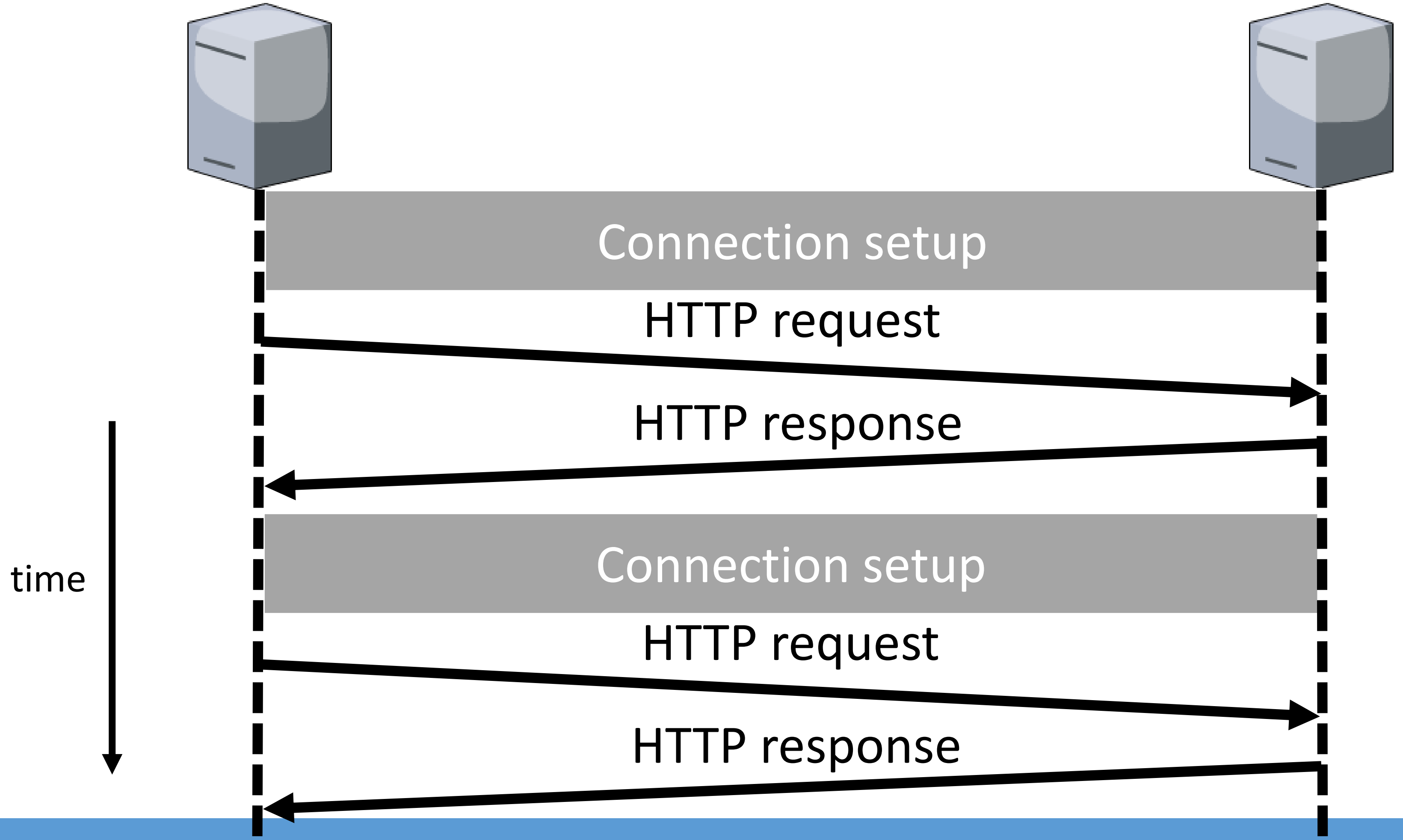
Recap
TCP Connection setup



HTTP
Sequential requests



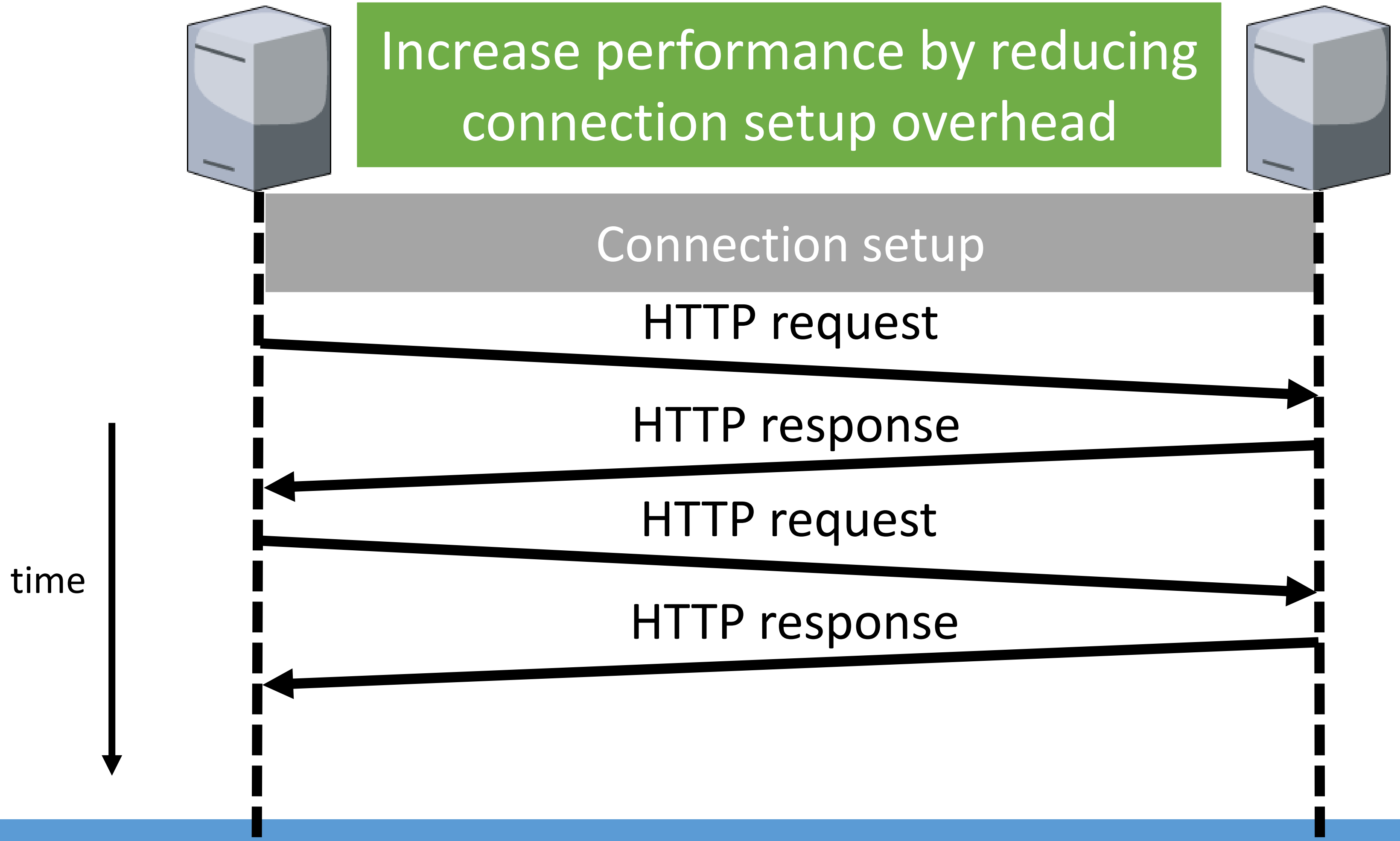
HTTP
Sequential requests



HTTP
Persistent connection

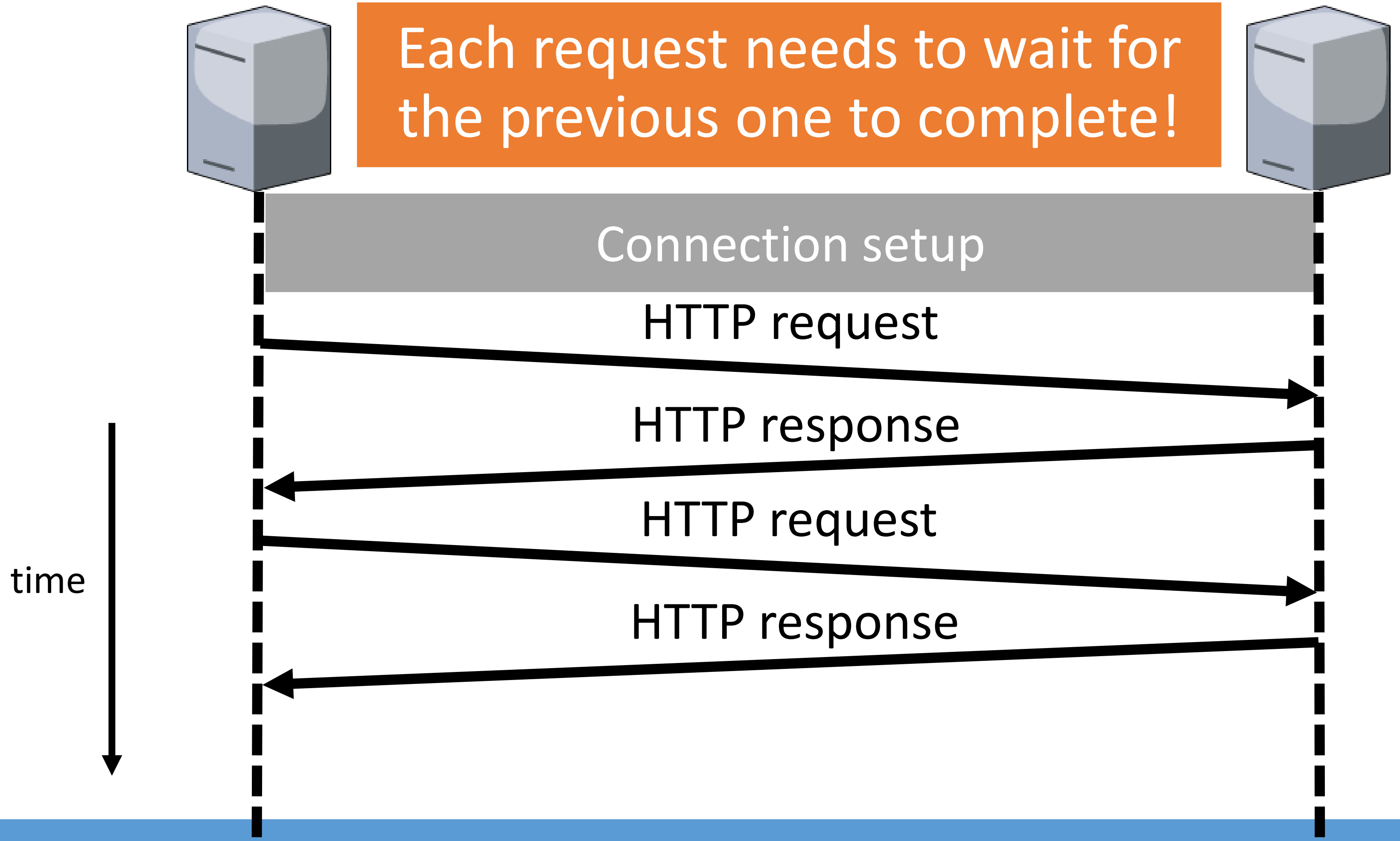
Persistent connections allow browsers to issue multiple requests over the same TCP connection

Increase performance by reducing connection setup overhead



HTTP Performance Problem
Head of Line Blocking (HOL)

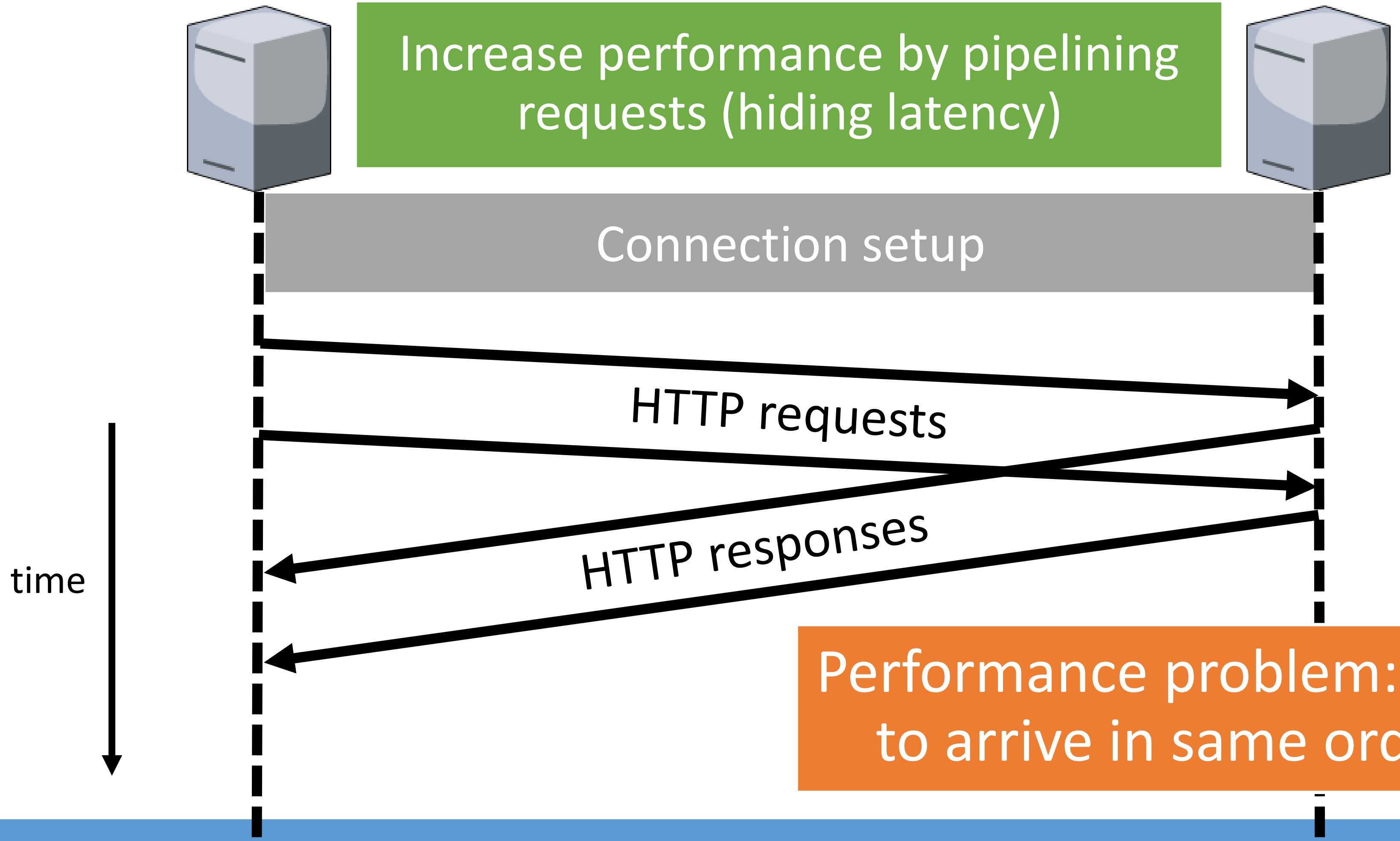
Each request needs to wait for the previous one to complete!



Reduces Head of Line Blocking!

HTTP1.1 Pipelined requests

Increase performance by pipelining requests (hiding latency)



Performance problem: Responses need to arrive in same order as requests

HTTP/2

1. Binary instead of plaintext.

WIRESHARK

Easier for machines to parse

More difficult for humans to read

Q: Why would it be easier for machines?

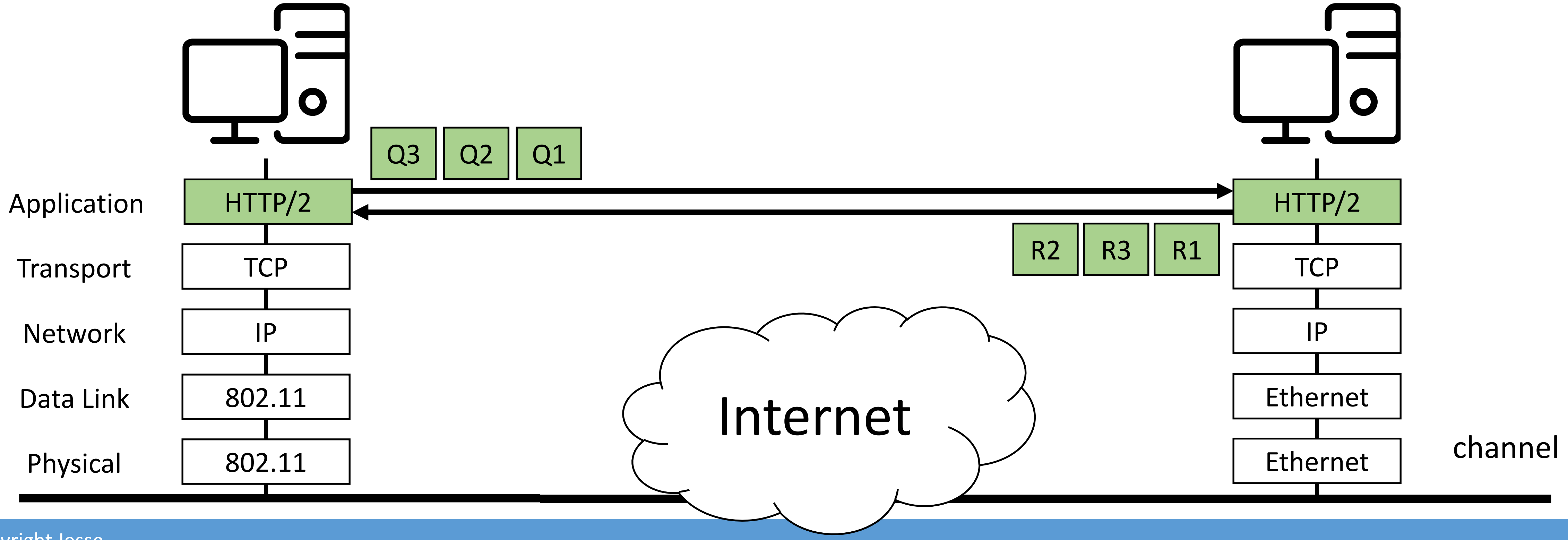
2. Multiplexed streams over a single TCP connection.

Supports out-of-order responses!

3. Server push allows the server to send resources before the client asks for it explicitly.

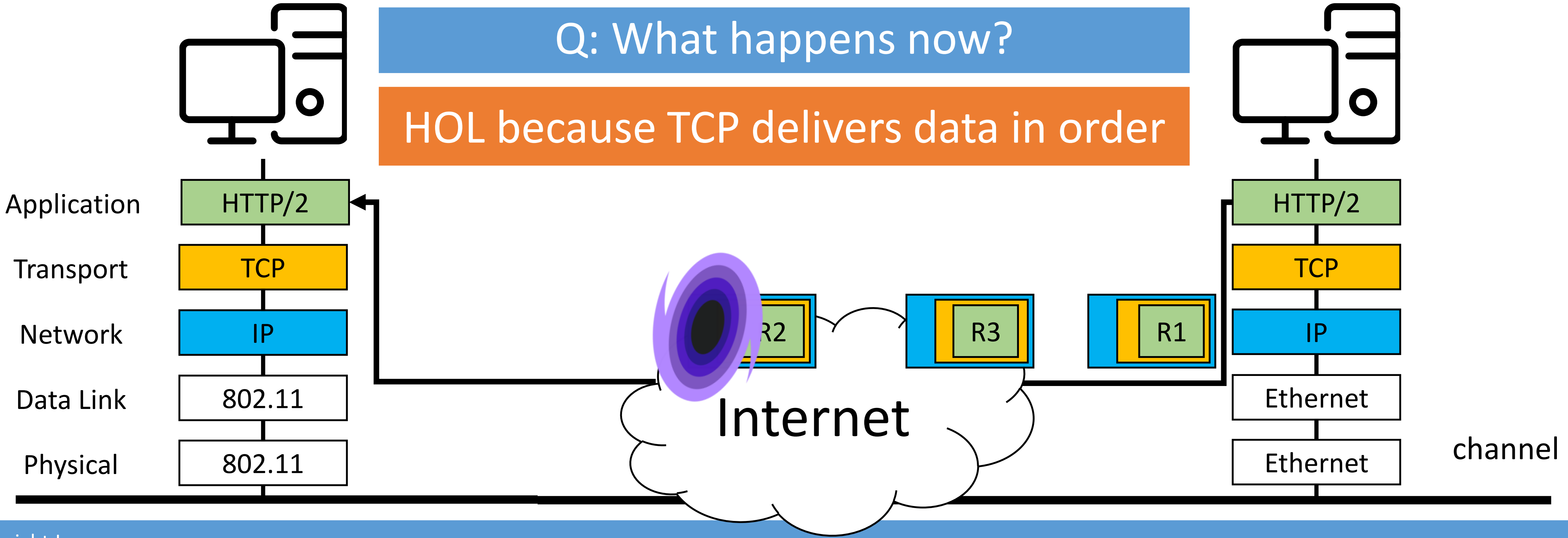
Head-of-Line Blocking in HTTP/2

Despite ***pipelining*** (HTTP1.1) and ***out-of-order responses*** (HTTP/2), HTTP/2 performance still suffers from a type of Head of Line blocking



Head-of-Line Blocking in HTTP/2

Despite ***pipelining*** (HTTP1.1) and ***out-of-order responses*** (HTTP/2), HTTP/2 performance still suffers from a type of Head of Line blocking



QUIC orders data per *stream*

RFC 9051

HTTP/3 (HTTP + *QUIC*)

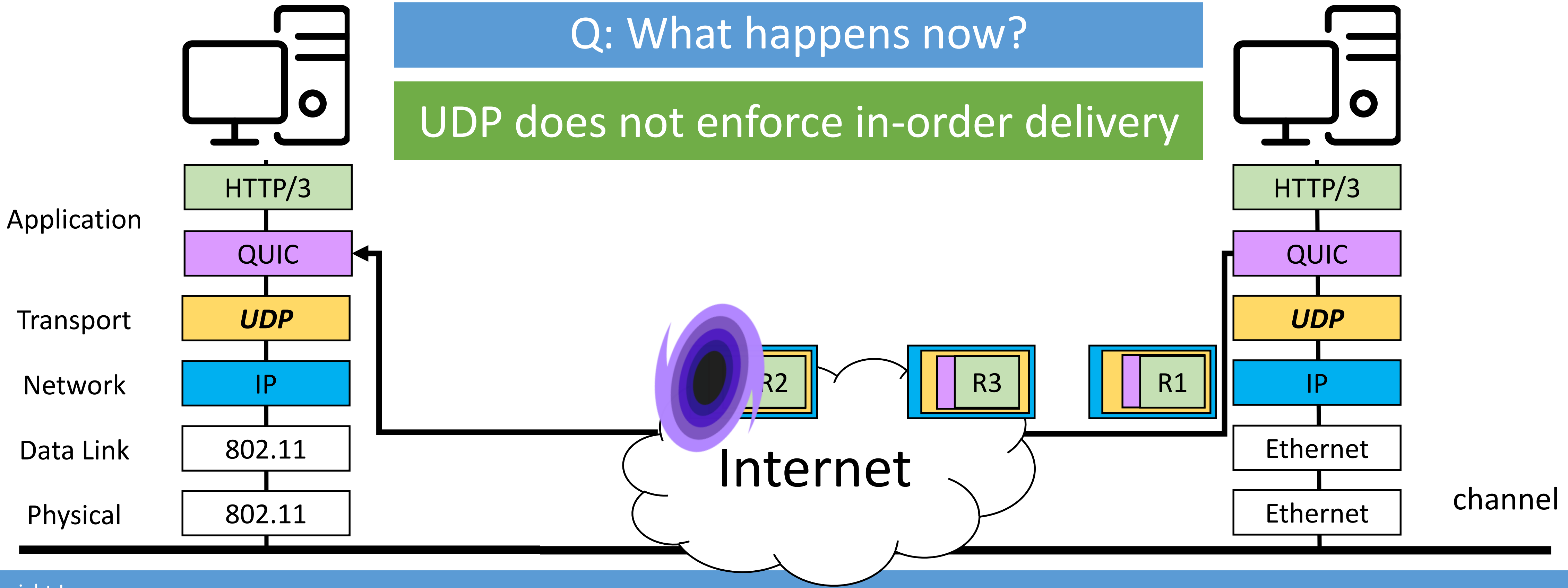
Each HTTP request can use a separate stream; within a stream, data is delivered in order; across streams no such guarantee is made

HTTP/3 uses the ***QUIC*** protocol

QUIC performs multiplexing, uses UDP

Q: What happens now?

UDP does not enforce in-order delivery



RFC 6455

WebSockets

Application layer protocol

Q: Can the application layer contain protocols?

A socket-like interface on the application layer.

Full-duplex connection between server and client.

Q: Can you think of a use-case?

Increasingly complex ‘apps’ on the Web that need to send data continuously.

Examples:

1. `irc-ws.chat.twitch.tv`

2. `ws.todoist.com`

WebSockets

RFC 6455

Application layer protocol

A socket-like interface on the application layer.

Full-duplex connection between server and client.

Q: Can you think of a use-case?

Increasingly complex ‘apps’ on the Web that need to send data continuously.

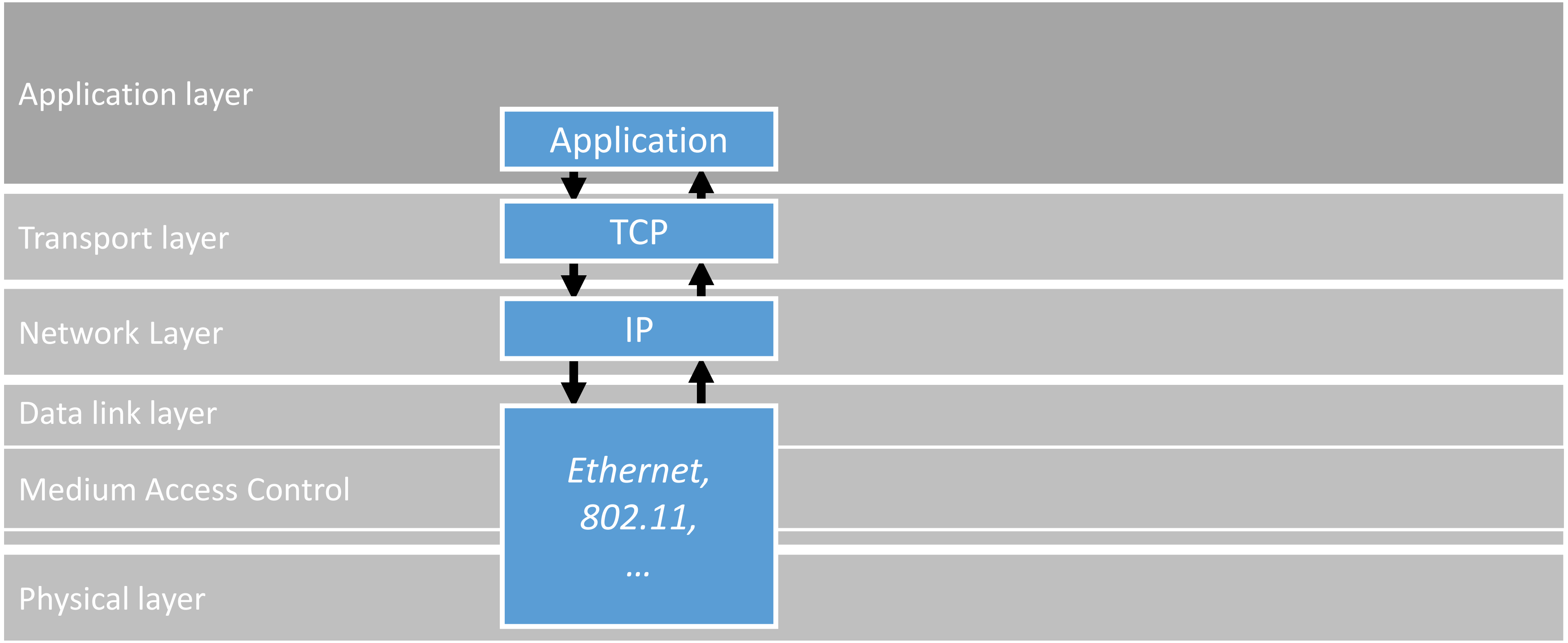
Examples:

1. `irc-ws.chat.twitch.tv`

2. `ws.todoist.com`

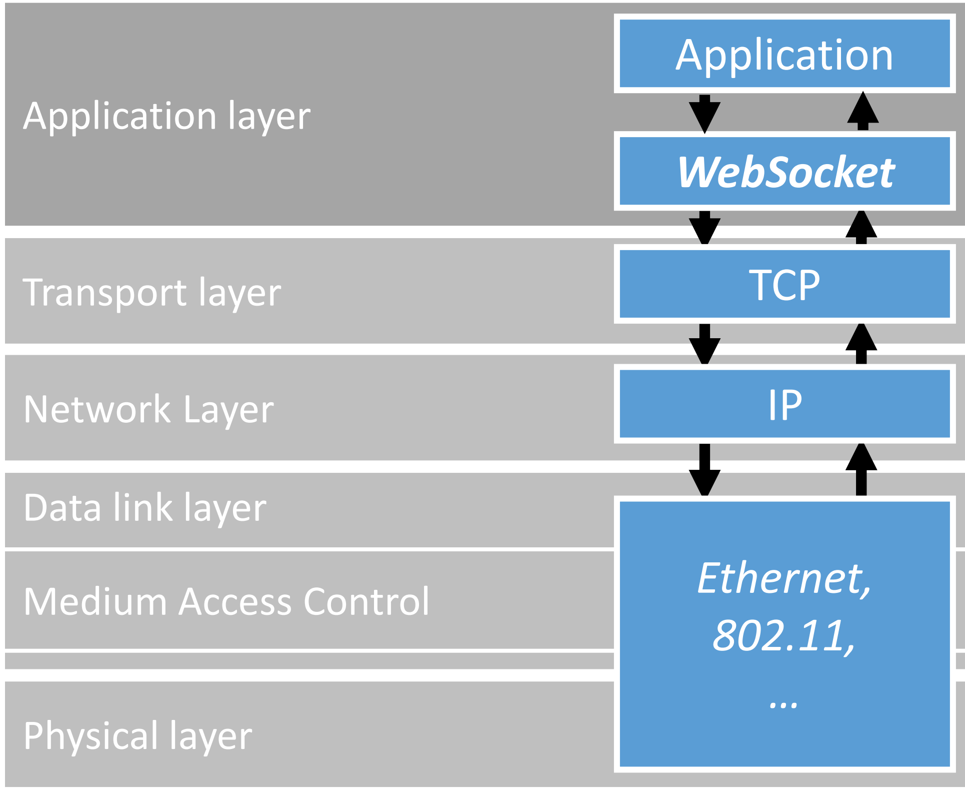
‘ws’ stands for WebSocket

Stacking
Application layer protocols



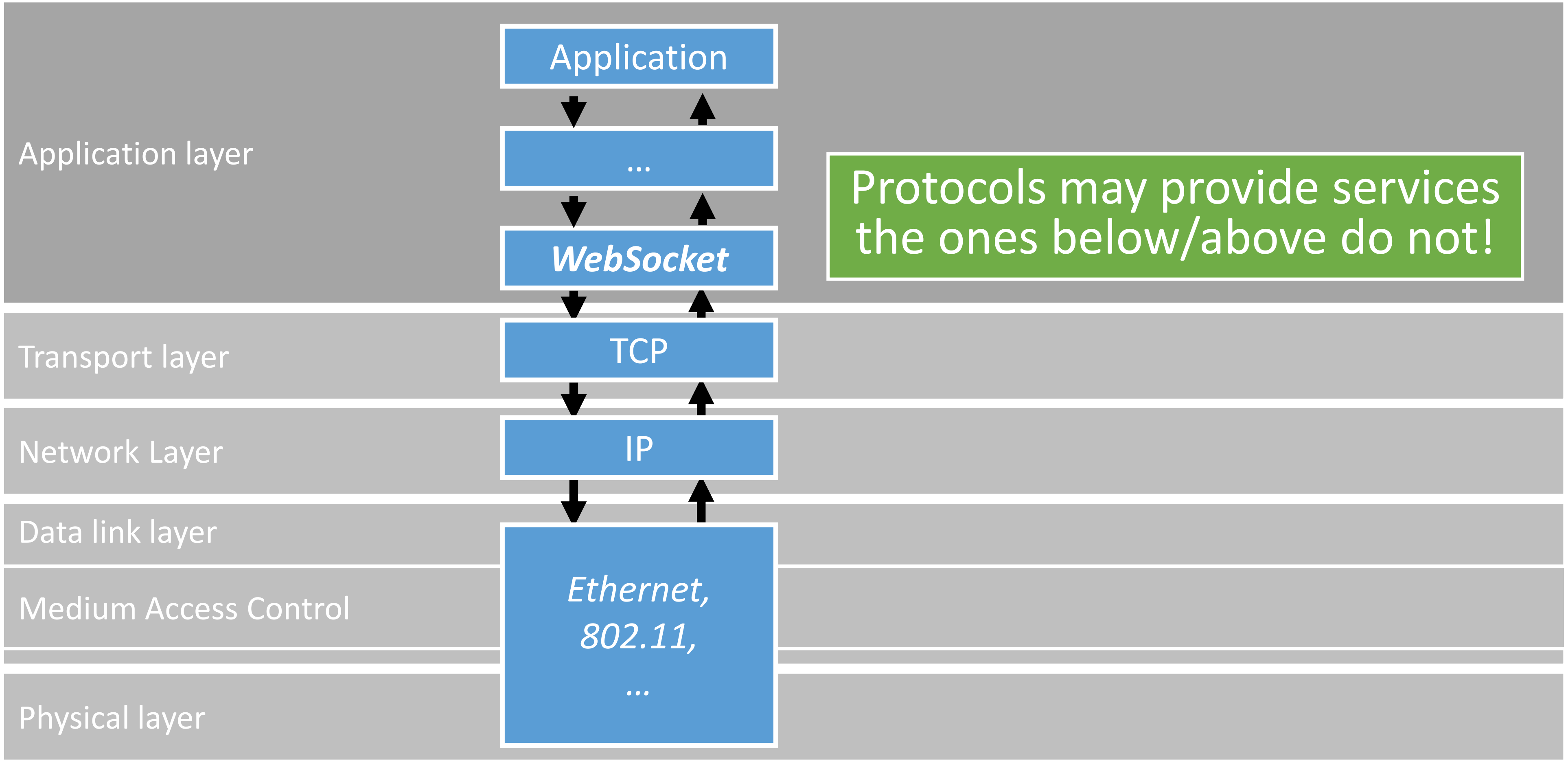
Stacking
Application layer protocols

Application layer can continue stacking protocols



Protocols may provide services the ones below/above do not!

Stacking
Application layer protocols



Starting a WebSocket over HTTP

```
GET /chat HTTP/1.1
Host: example.com:80
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

Client requests to switch to WebSocket protocol

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept:
s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```

Reply from server if it accepts

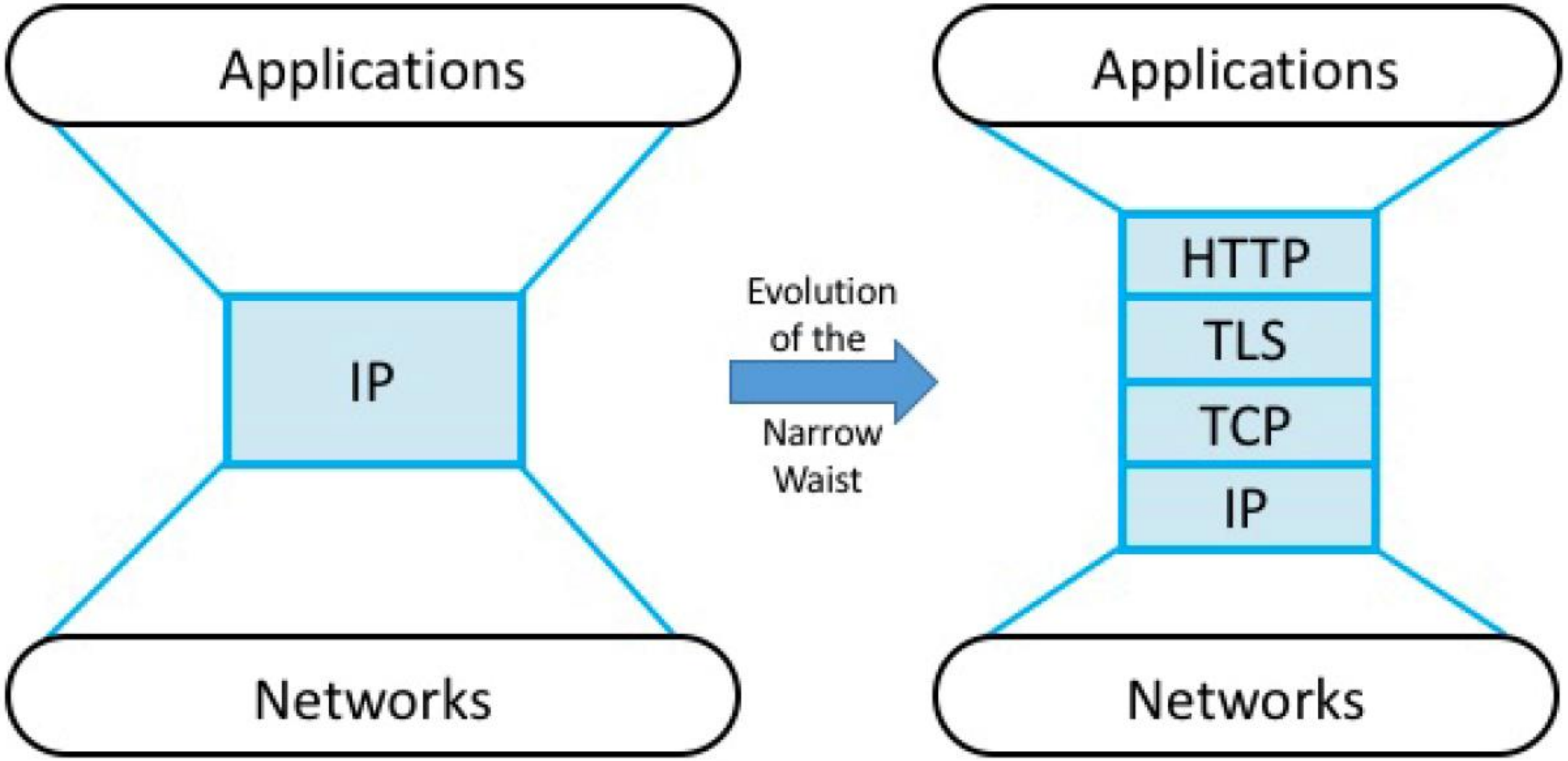
WebSocket frame format

```
Frame format:

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-------+-+-------------+-------------------------------+
     |F|R|R|R| opcode|M| Payload len |    Extended payload length    |
     |I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
     |N|V|V|V|       |S|             |   (if payload len==126/127)   |
     | |1|2|3|       |K|             |                               |
     +-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
     |     Extended payload length continued, if payload len == 127  |
     + - - - - - - - - - - - - - - - +-------------------------------+
     |                               |Masking-key, if MASK set to 1  |
     +-------------------------------+-------------------------------+
     | Masking-key (continued)       |          Payload Data         |
     +-------------------------------- - - - - - - - - - - - - - - - +
     :                     Payload Data continued ...                :
     + - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
     |                     Payload Data continued ...                |
     +---------------------------------------------------------------+
```

E.g., REST APIs

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page

HTTP is the new "narrow waist"



Q: Advantages over using TCP directly?

Answers include:

Provides set of methods

Provides security

Provides *naming*

Source: Peterson, Larry L., and Bruce S. Davie. *Computer networks: a systems approach*. Elsevier, 2007.

Application Layer Topics

1. Domain Name System (DNS)
2. Email
3. Web (HTTP, QUIC, WebSocket)
4. **Multimedia applications**

Video dominates

Video constitutes around 70 percent of all global mobile network traffic in 2022

- 28,000 people watching Netflix
- 500 hours of content uploaded to YouTube
- 2 million Twitch views
- 3.4 million Snaps created

Source: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/traffic-by-application

Streaming Video Requires Compression

1024 height x 2048 width = 2M pixels

1 pixel = 1 byte

30 frames per second → 60 MB/s = 480 Mbps

Without compression, only possible over wired fibre-optic channels

Compression reduced bandwidth requirement by an order of magnitude

Internet connection speed recommendations

To watch TV shows and movies on Netflix, we recommended having a stable internet connection with a download speed shown below in megabits per second (Mbps).

Video quality	Resolution	Recommended speed
High definition (HD)	720p	3 Mbps or higher
Full high definition (FHD)	1080p	5 Mbps or higher
Ultra high definition (UHD)	4K	$\boxed{\text{15 Mbps or higher}}$

Source: https://help.netflix.com/en/node/306

Large compression rates $> \times 10$.

Digital audio compression

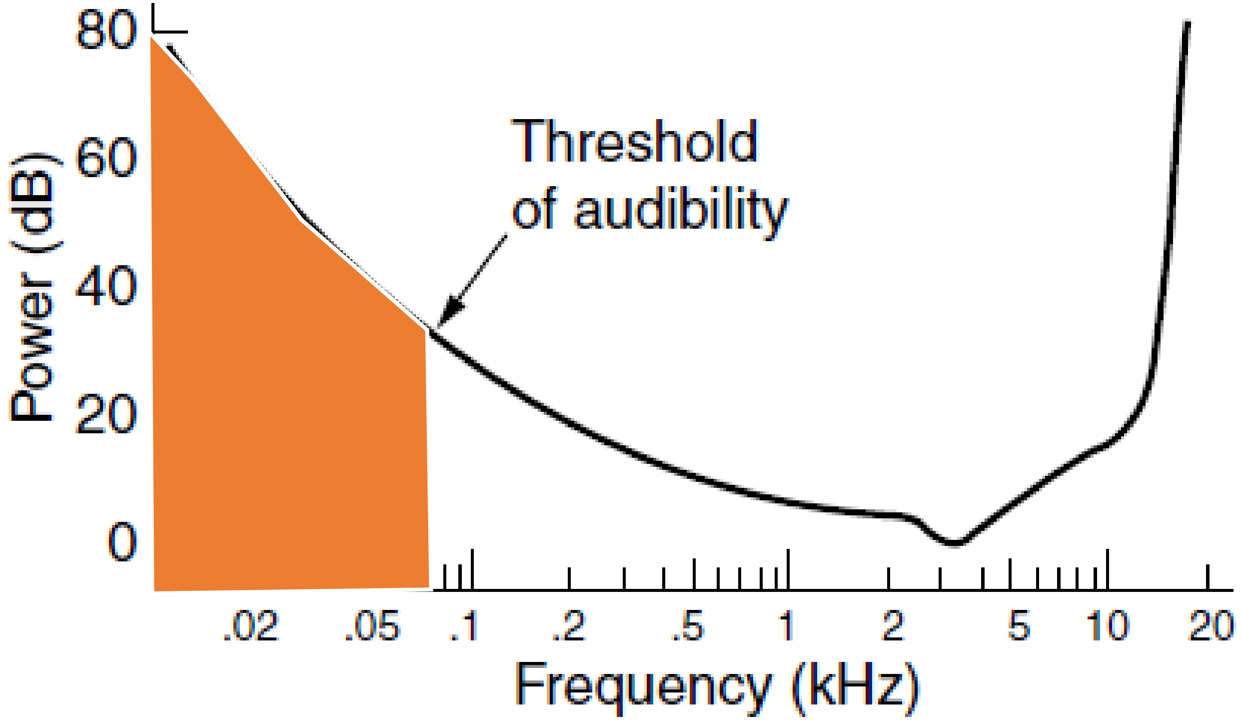
Audio typically compressed before sending.

Lossy compression achieves higher compression rates than ***lossless compression***, but ***loses data.***

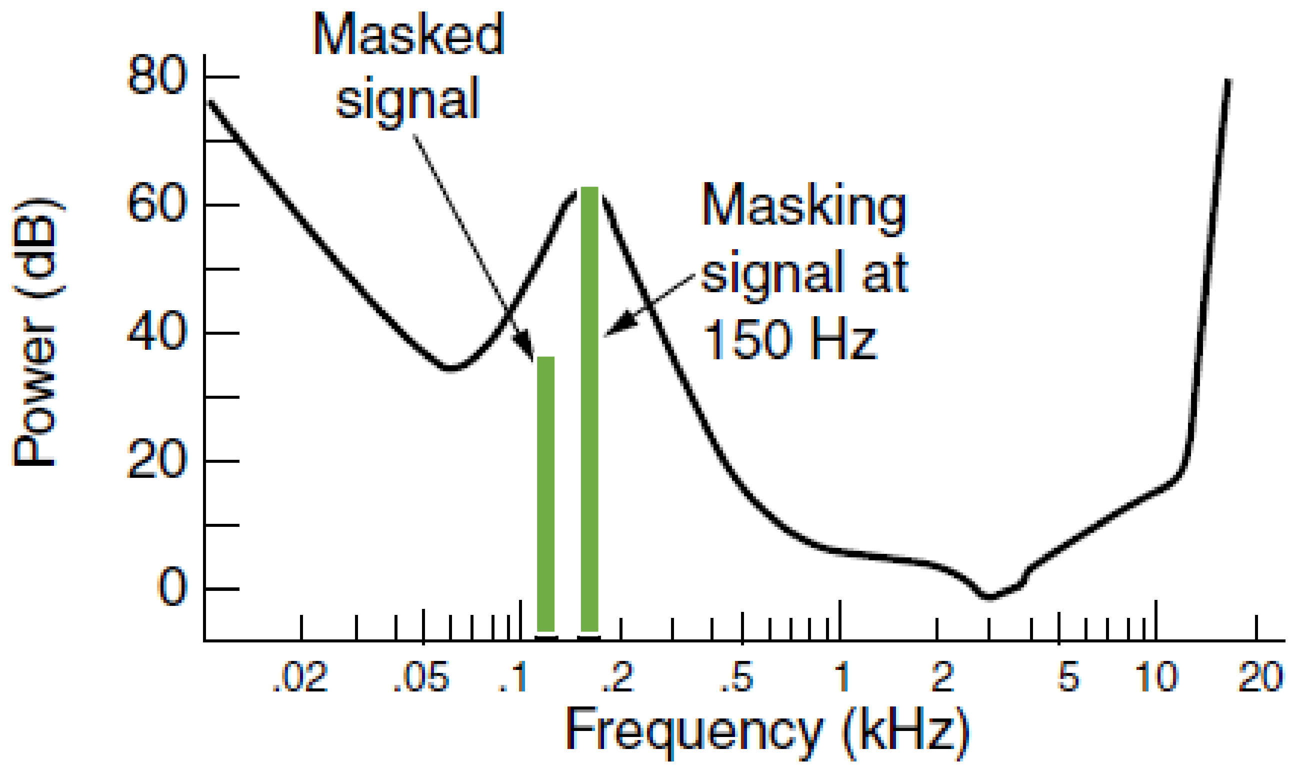
Q: Why is lossy compression acceptable?

Lossy encoders based on how humans perceive sound.

Human hearing frequency range



Human hearing masked signals



Digital video
JPEG compression

Changes RGB to YC_bC_r.

Y is luminance.

C_bC_r are chrominances.

Q: Why change to this format?

Eyes are ***less*** sensitive to chrominance than to luminance.

JPEG reduces size of Cb and Cr.
Total compression rate $\times$ 2.

Large compression rates >× 50.

Digital video

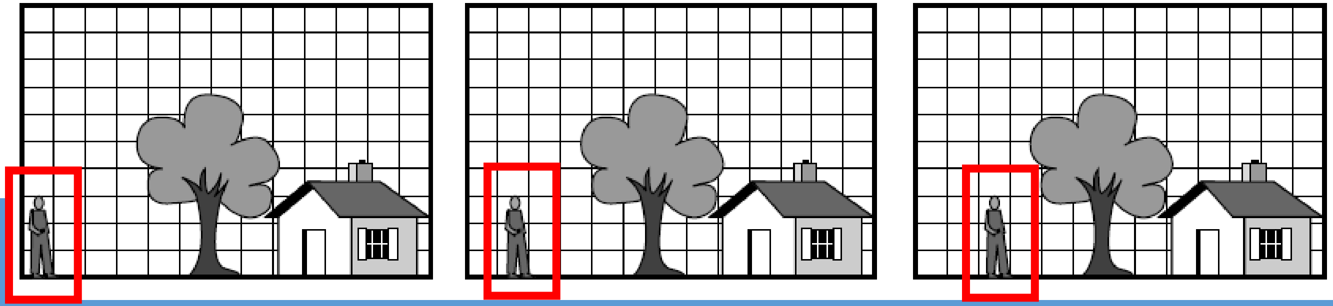
Q: What is the use of ***bidirectional*** frames?

MPEG compresses over a sequence of frames, further using motion tracking to remove temporal redundancy

1. I (Intra-coded) frames are self-contained
2. P (Predictive) Looks for comparable ***macro blocks*** in previous frames.

How long to search is up to the implementation.

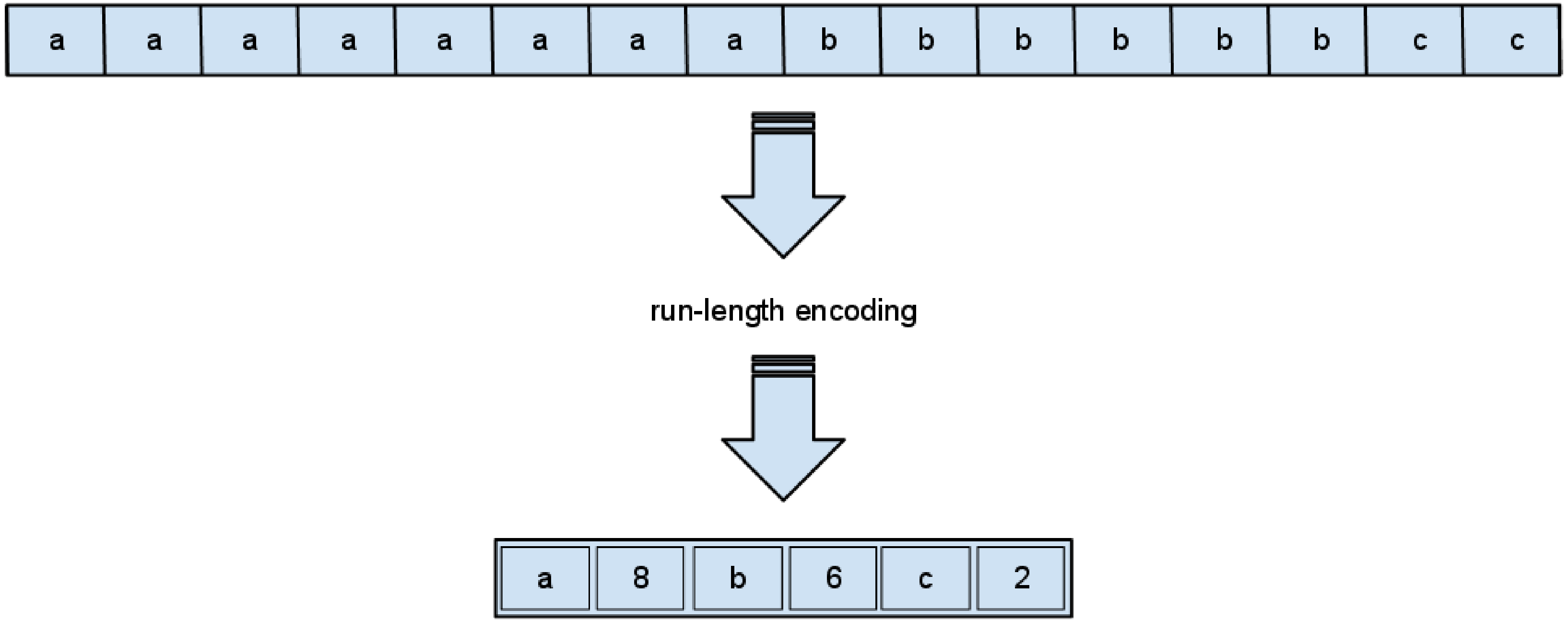
3. B (Bidirectional) frames may base prediction on previous frames and ***future*** frames.



Run-Length Encoding

Part of JPEG Compression

A lossless compression technique.



Huffman Encoding

Prefix code: no code word is prefix of other code word

Q: Why is this useful?

String

“application layer”

ASCII

61 70 70 6c 69 63 61 74 69 6f 6e 6c 61 79 65 72
(128 bits)

Huffman Encoding

11 101 101 100 0111 0110 11 0101 0111 0100 0011 100 11 0010 0001 0000
(54 bits)

Less than half the original size!

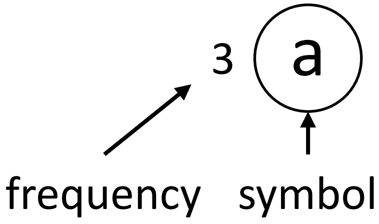
$$\frac{54}{128} < 0.42$$

11 101 101 100 0111 0110 11 0101 0111 0100 0011 100 11 0010 0001 0000

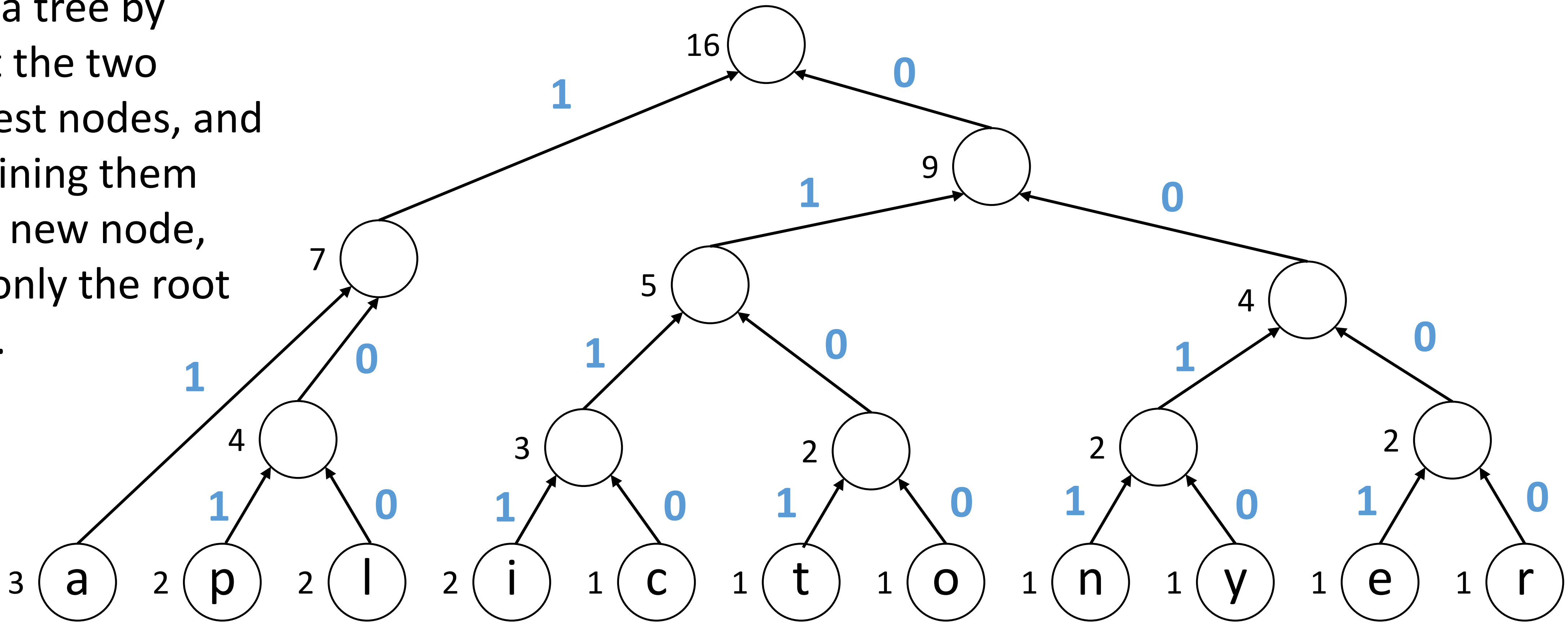
(54 bits) “application layer”

Huffman Encoding

Part of JPEG Compression



Form a tree by select the two smallest nodes, and combining them into a new node, until only the root is left.

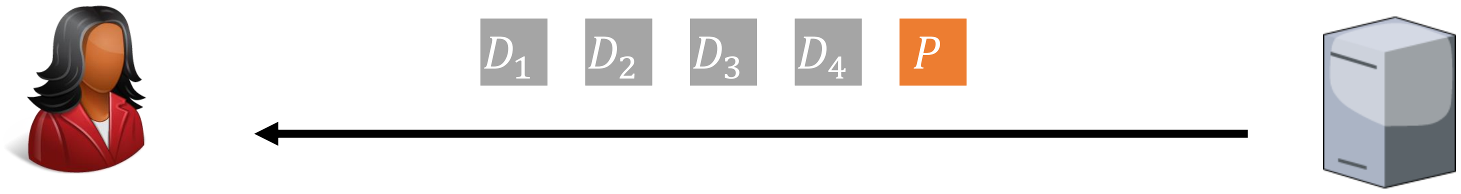


Networking Challenges for Multimedia Applications

Challenge 1
Streaming stored media

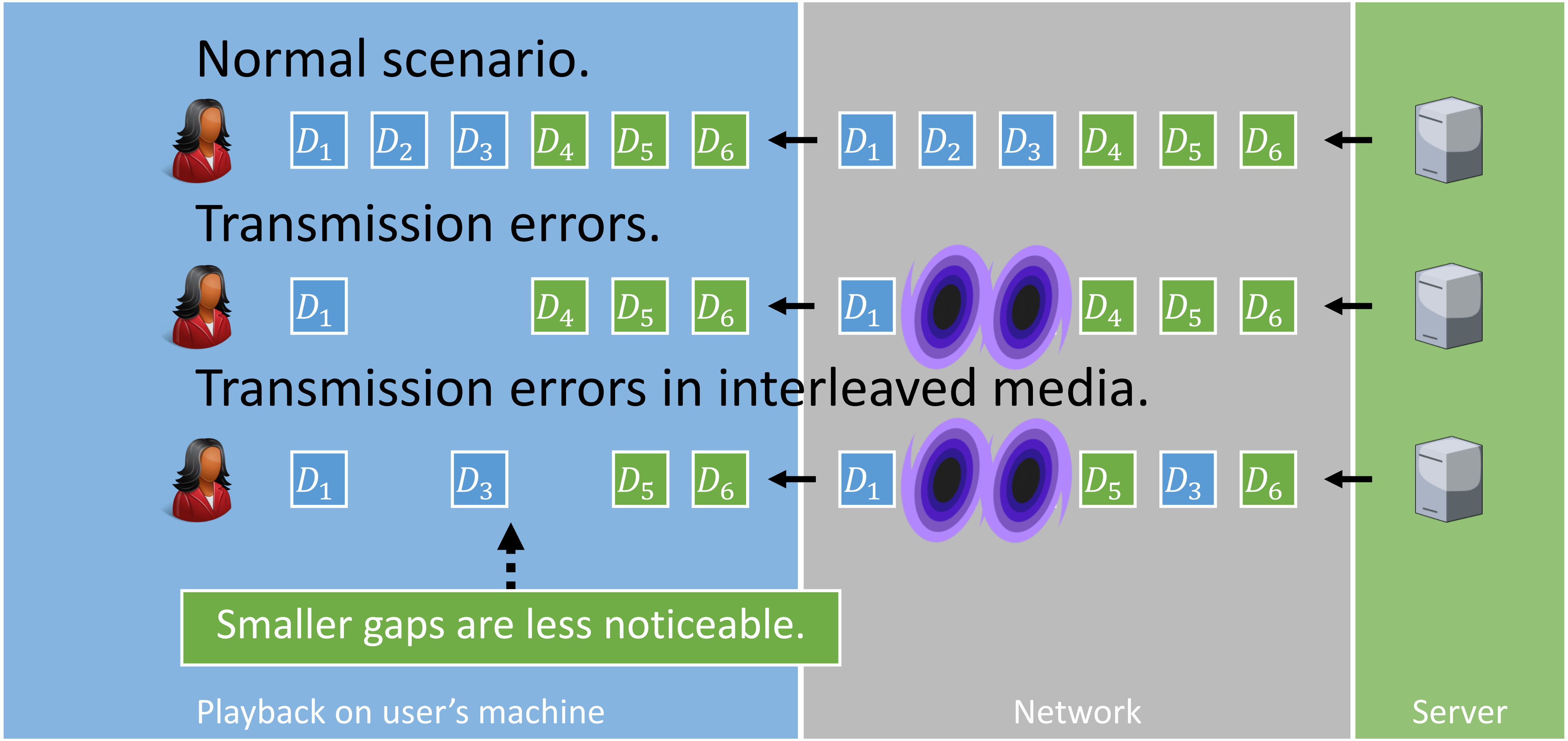
How to handle ***transmission errors***?

1. Use reliable transport (e.g., TCP).
   - Increases jitter significantly.
2. Use ***forward error correction***
   (error correction in the application layer).
   - Increases jitter, decoding complexity, and overhead.

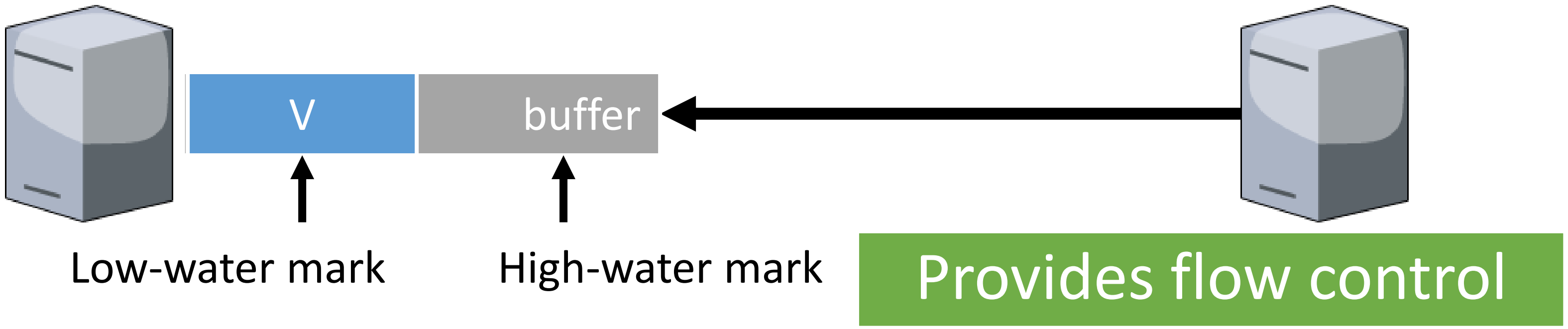


3. Interleave media
   - Slightly increases jitter and decoding complexity.

Masking errors by interleaving media



Challenge 1
Streaming stored media



Low-water mark prevents ***stalls*** in playback.

High-water mark gives client time to prevent ***running out of buffer space***.

Challenge 2
Streaming live media

Streaming live media is similar to the stored case plus:

1. Can't stream faster than ***live rate*** to get ahead
   - Usually need larger buffer to absorb jitter
2. Often have many users viewing at the same time
   - UDP with multicast greatly improves efficiency. It is rarely available, so ***many TCP connections are used***.

Challenge 3
Streaming interactive media

Real-time conferencing has two or more connected live media streams, e.g., voice over IP, Skype video call

Requires low jitter **and** low latency.

1. Benefits from network support (Quality of Service).
2. Large bandwidth (no congestion).

Difficult to provide across long distances/multiple networks

Take-Home Message

- Many responsibilies and pseudo layers hidden in Application Layer
  - From OSI: Presentation, Session. Others: WebSocket, RTP, etc.
- Important behind-the-Scenes applications exist (e.g., DNS)
- Traditional “killer apps” for the Internet:
  - Email
  - The Web
- HTTP is the new “narrow waist”
  - Improved over time (HTTP/2 [SPDY], HTTP/3 [QUIC])
- Today’s Internet is increasingly used for multimedia applications
  - Provide new challenges (high bandwidth, low latency, low jitter)

Quiz

Check canvas or scan the code →

Rules of Engagement:

- Pen and paper allowed
- Do the quiz by yourself
- Closed-book (no external sources of information)
- No calculators

<u>Quiz ends at 17:19!</u>