Computer Networks X_400487

Lecture 11

Chapter 7: The Application Layer—Part 2

Lecturer: Jesse Donkervliet

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

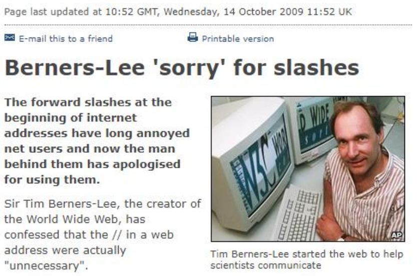
Douglas Engelbart

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.

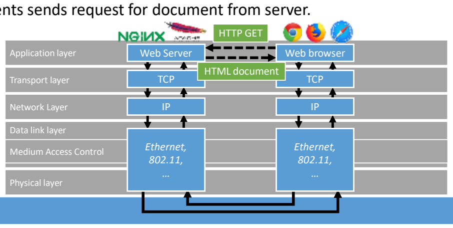
Sir Tim Berners-Lee, the creator of the World Wide Web, has confessed that the // in a web address were actually "unnecessary".

Tim Berners-Lee started the web to help scientists communicate

http://news.bbc.co.uk/2/hi/technology/8306631.stm

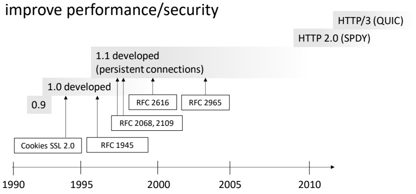
HTTP Request/Response

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
```

https://tools.ietf.org/html/rfc2616

HTTP Request via TELNET

```
➜ ~
```

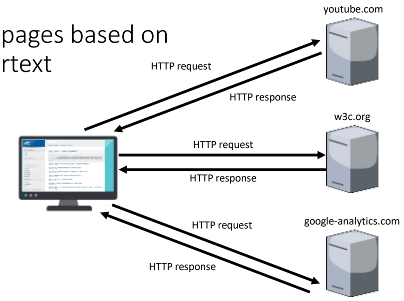
HTTP Request Methods

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.

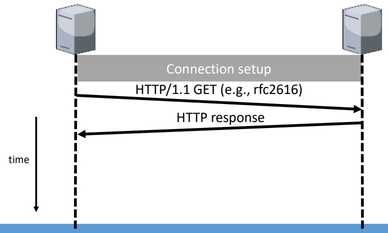
Web pages based on Hypertext



Web and HTTP Performance

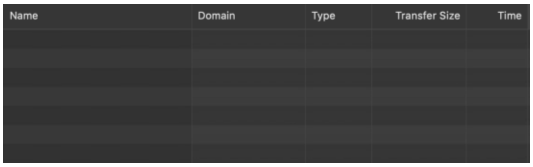
The Web and HTTP continues to evolve, with servers sending ***more*** and ***larger*** responses

Single document



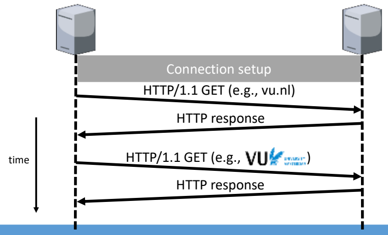
Single document Example

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

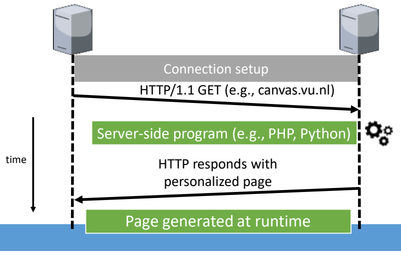


Name	Domain	Type	Transfer Size	Time

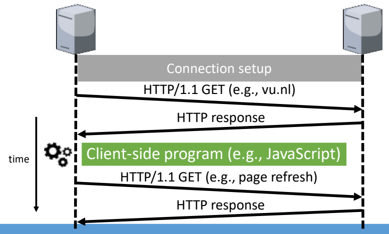
External resources



Server-side programs

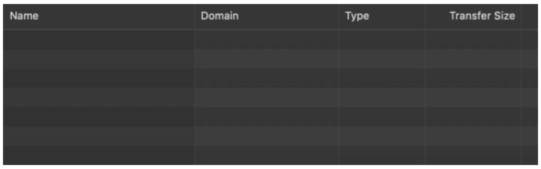

Connection setup
HTTP/1.1 GET (e.g., canvas.vu.nl)
Server-side program (e.g., PHP, Python)
time
HTTP responds with personalized page
Page generated at runtime

Client-side programs


Connection setup
HTTP/1.1 GET (e.g., vu.nl)
HTTP response
time
Client-side program (e.g., JavaScript)
HTTP/1.1 GET (e.g., page refresh)
HTTP response

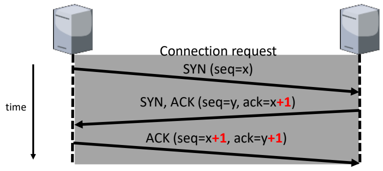
Modern webpages
Many requests

https://canvas.vu.nl/

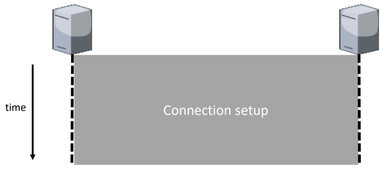


Name	Domain	Type	Transfer Size

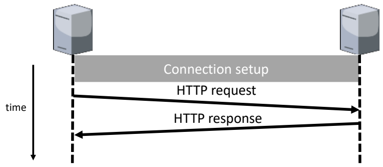
Recap
TCP Connection setup



Recap
TCP Connection setup

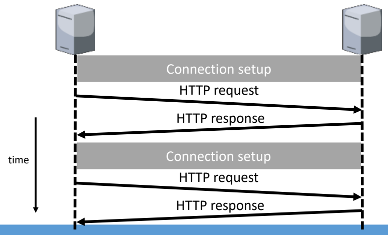


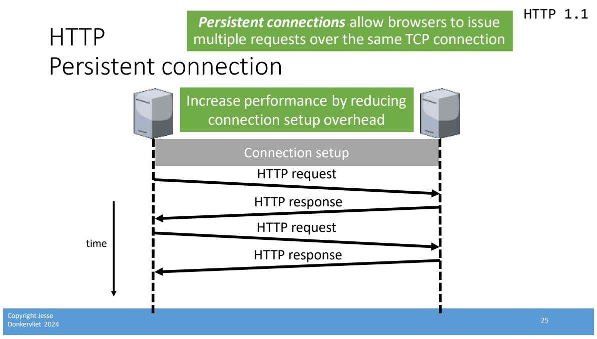
HTTP
Sequential requests



HTTP 1.0

HTTP
Sequential requests



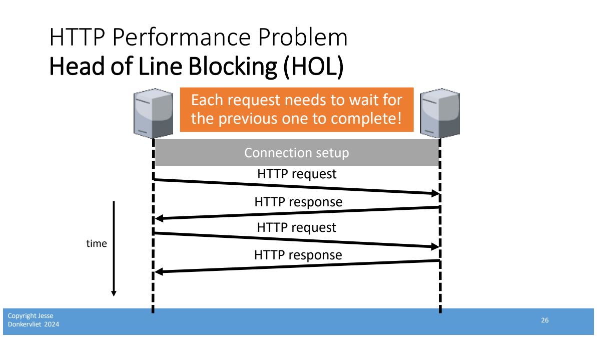


HTTP 1.1

HTTP
Persistent connection

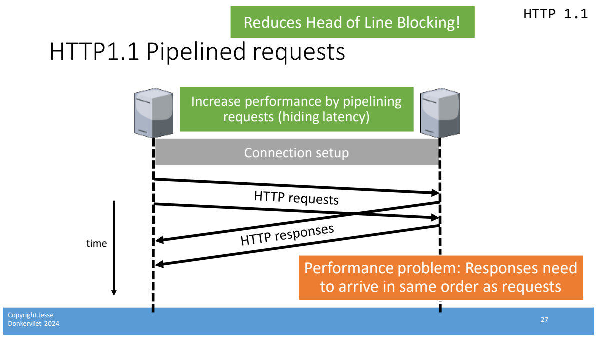
Persistent connections allow browsers to issue multiple requests over the same TCP connection

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!



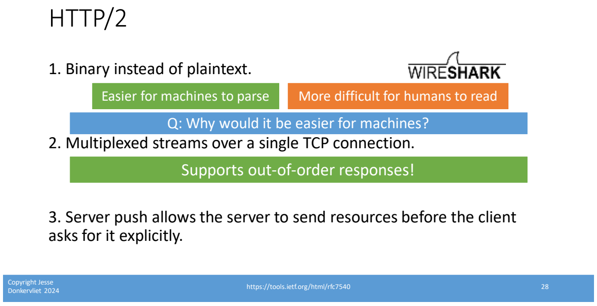
HTTP 1.1

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.

Easier for machines to parse

More difficult for humans to read

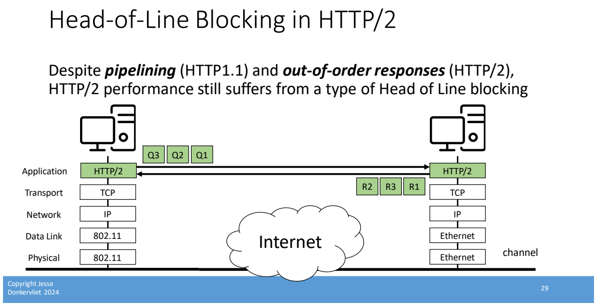
Q: Why would it be easier for machines?

2. Multiplexed streams over a single TCP connection.

Supports out-of-order responses!

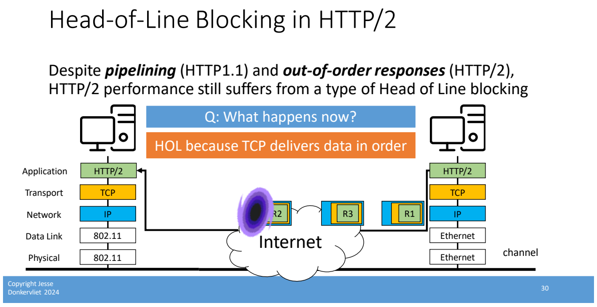
3. Server push allows the server to send resources before the client asks for it explicitly.

https://tools.ietf.org/html/rfc7540



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

Q: What happens now?

HOL because TCP delivers data in order

QUIC orders data per *stream*

RFC 9051

HTTP/3 (HTTP + *QUIC*)

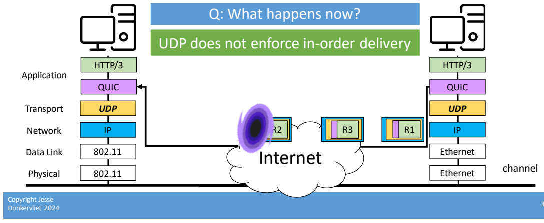
Each HTTP request can use a separate stream; within a stream, data is delivered in order; across streams no such guarantee is made

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`

irc-ws.chat.twitch.tv other 1.10 MB

2. `ws.todoist.com`

RFC 6455

WebSockets

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:

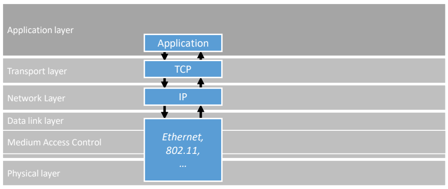
1. `irc-ws.chat.twitch.tv`

irc-ws.chat.twitch.tv other 1.10 MB

2. `ws.todoist.com`

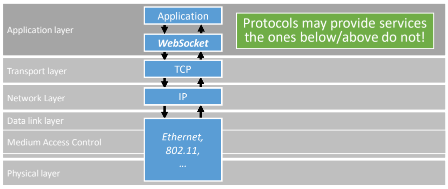
‘ws’ stands for WebSocket

Stacking
Application layer protocols



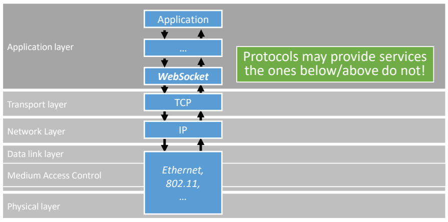
Stacking
Application layer protocols

Application layer can continue stacking protocols



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

Stacking
Application layer protocols



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

Starting a WebSocket over HTTP

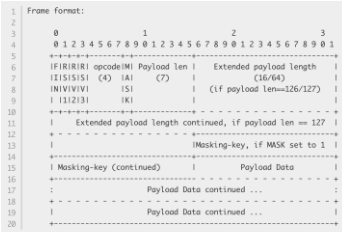
```
GET /chat HTTP/1.1
Host: example.com:80
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

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

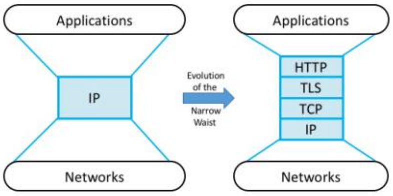


```
Frame format:

      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 ...                |
     +---------------------------------------------------------------+
```

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers

HTTP is the new "narrow waist"



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

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.

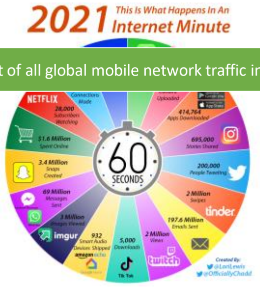
Application Layer Topics

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

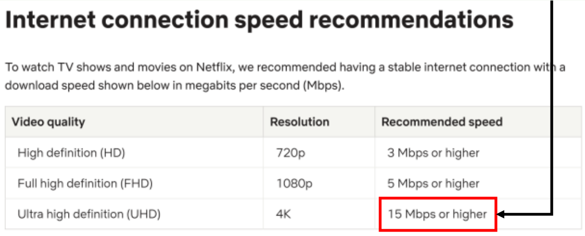
1024 height x 2048 width = 2M pixels

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	15 Mbps or higher

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

Large compression rates >× 10.

Digital audio compression

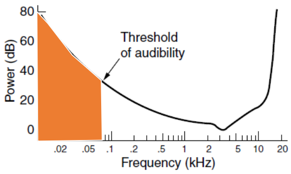
Audio typically compressed before sending.

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

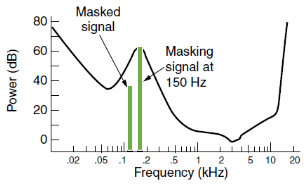
Q: Why is lossy compression acceptable?

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 × 2.

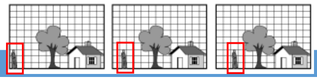
Large compression rates >× 50.

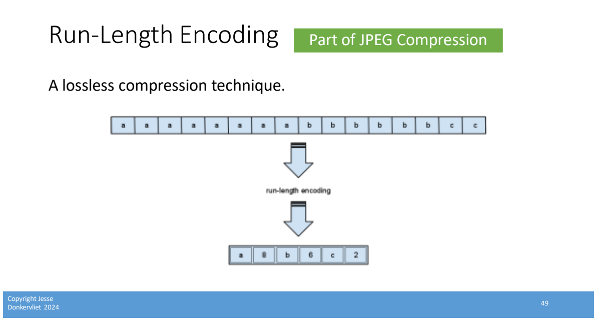
Digital video

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.

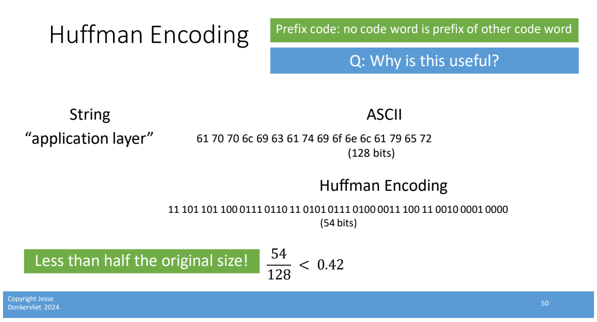




Run-Length Encoding

Part of JPEG Compression

A lossless compression technique.



Huffman Encoding

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

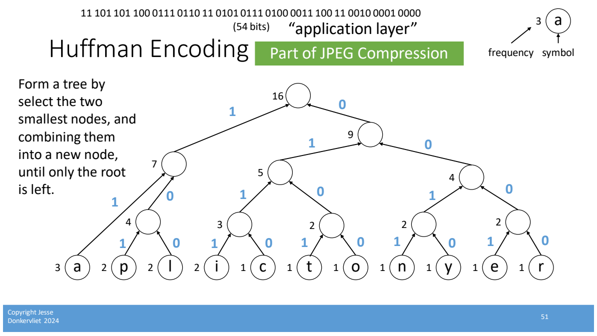
Q: Why is this useful?

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

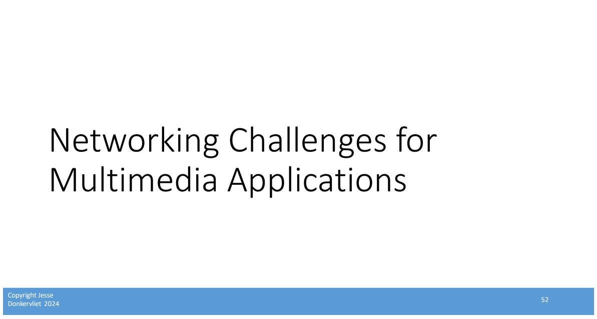


11 101 101 100 0111 0110 11 0101 0111 0100 0011 100 11 0010 0001 0000
(54 bits) “application layer”

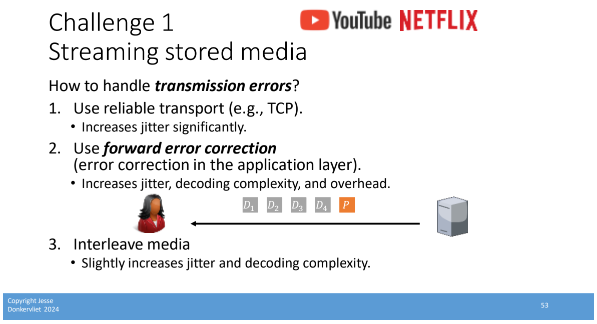
Huffman Encoding

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.



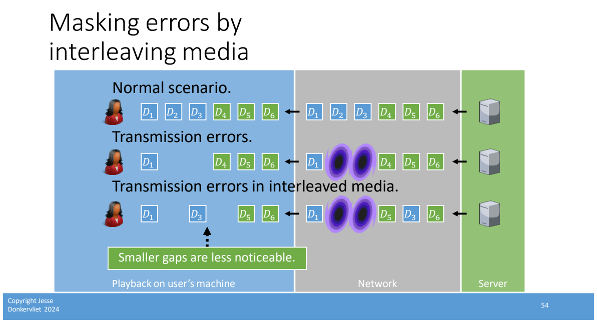
Networking Challenges for Multimedia Applications



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

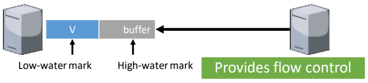
Normal scenario.

Transmission errors.

Transmission errors in interleaved media.

Smaller gaps are less noticeable.

Challenge 1
Streaming stored media



Provides flow control

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)