Building Fast Web Applications

Thierry Sans

Users respond to speed

"Amazon found every 100ms of latency cost them 1% in sales"

"Google found an extra .5 seconds in search page generation time dropped traffic by 20%"

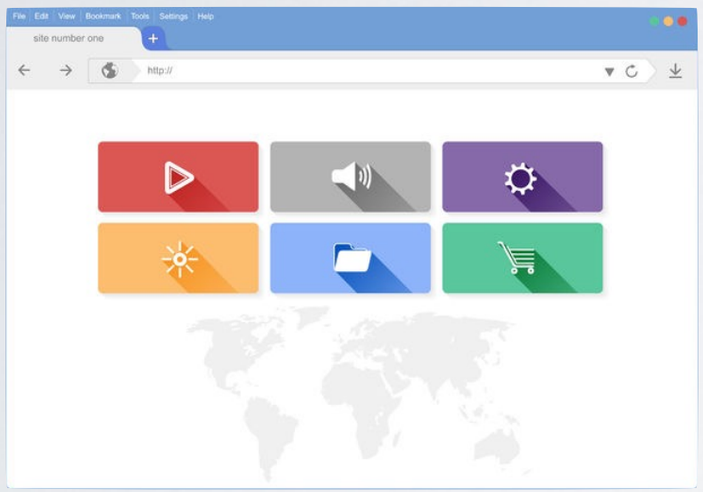
"A broker could lose $4 million in revenues per millisecond if their electronic trading platform is 5 milliseconds behind the competition"

http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/

Frontend packing

The problem

Browser



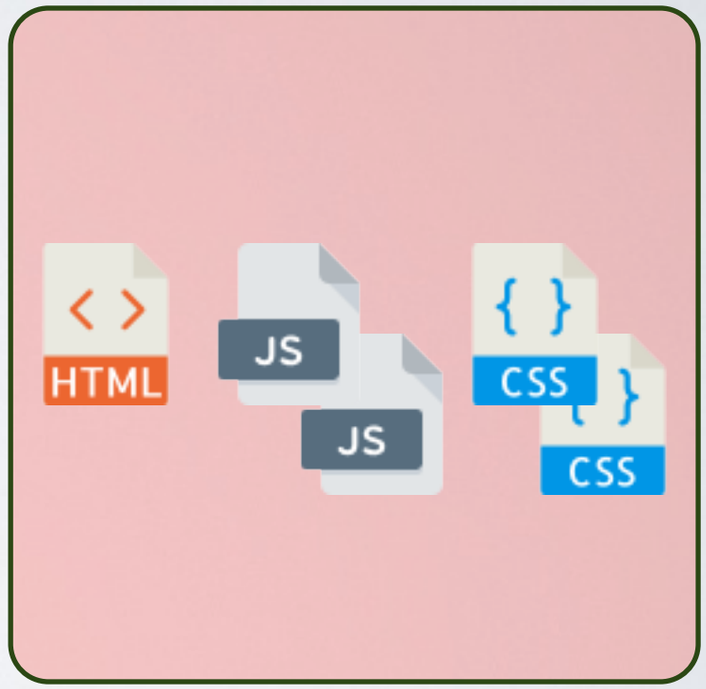
GET /

GET /js/lib.js

GET /js/index.js

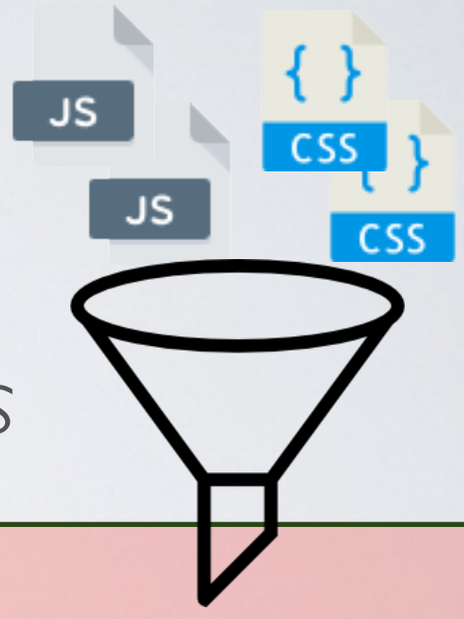
GET /style/index.css

GET /style/generic.css



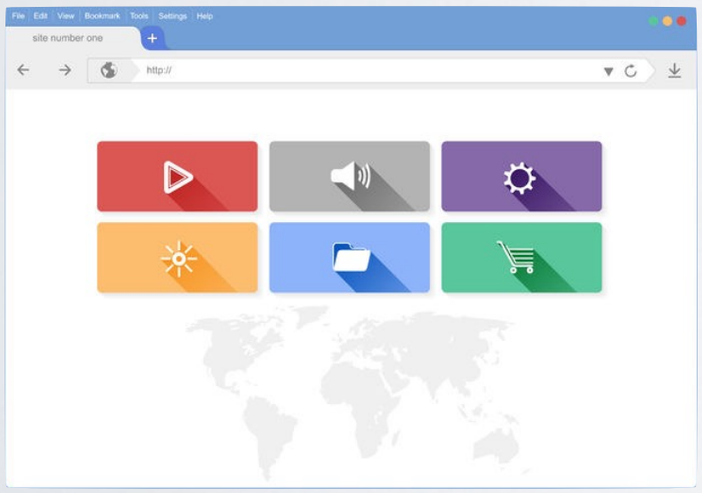
Frontend Server

The solution - using a frontend packer



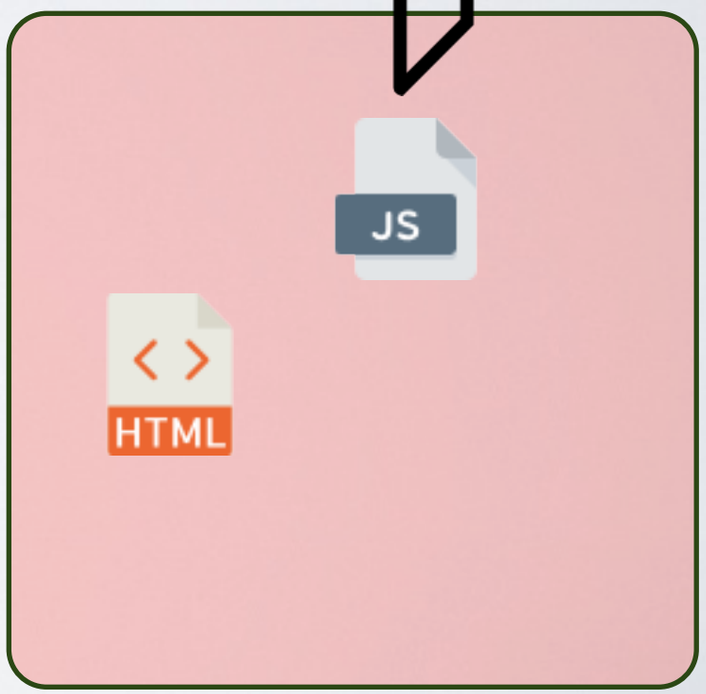
e.g. *webpack.js*

Browser




```
GET /
```

```
GET /js/bundle.js
```



Frontend Server

HTTP/2

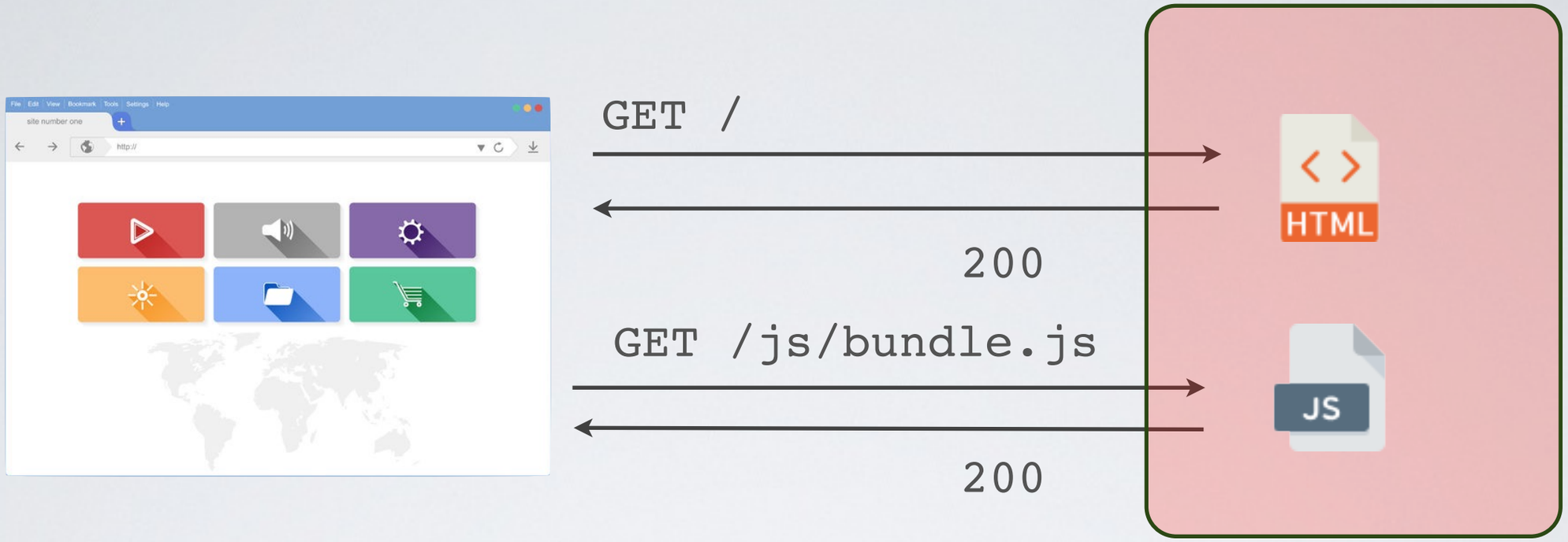
HTTP/2

HTTP/2 enables multiplexing

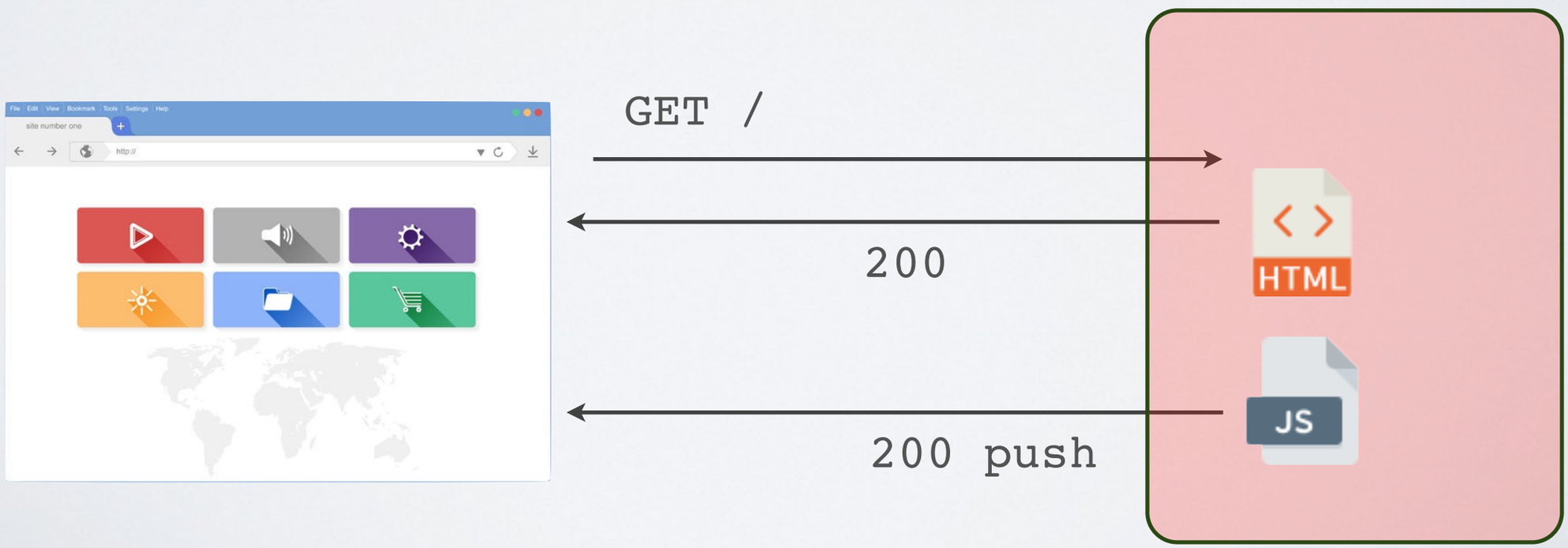
➡ send multiple HTTP responses for a given request (a.ka `push`)

- Proposed by Google (called SPDY)
- Adopted as an standard in 2015 (RFC 7540)
- HTTP/2 is compatible with HTTP/1 (same protocol)

HTTP 1.1



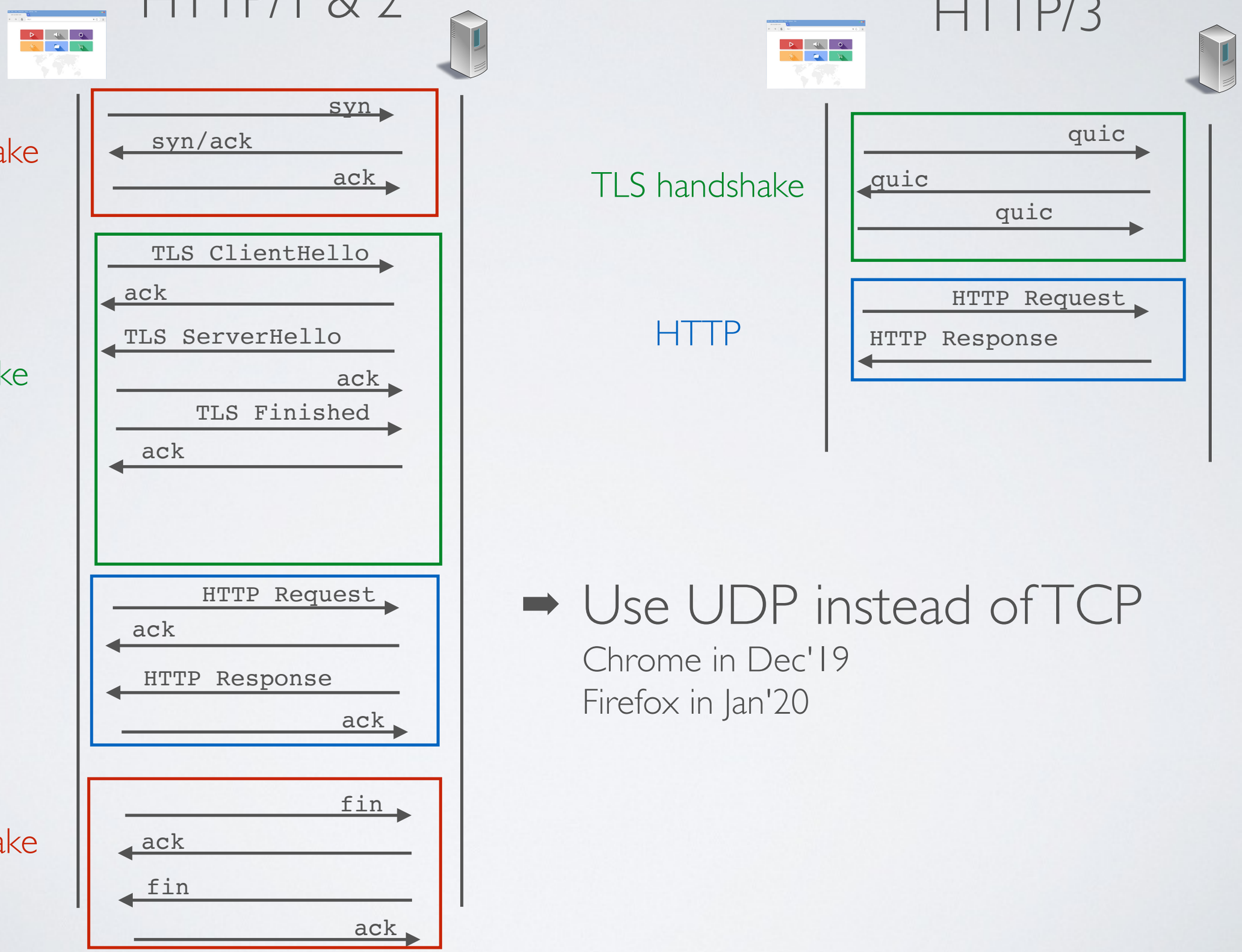
HTTP 2.0



HTTP/3

(work in progress)

HTTP/3 (standard draft)



➡ Use UDP instead of TCP

Chrome in Dec'19

Firefox in Jan'20

Long Polling

Short Polling vs Long Polling

Short Polling

- The frontend request an update from the backend every few seconds
- The backend replies right away regardless if there is an update or not

◉ Many request/responses are wasted

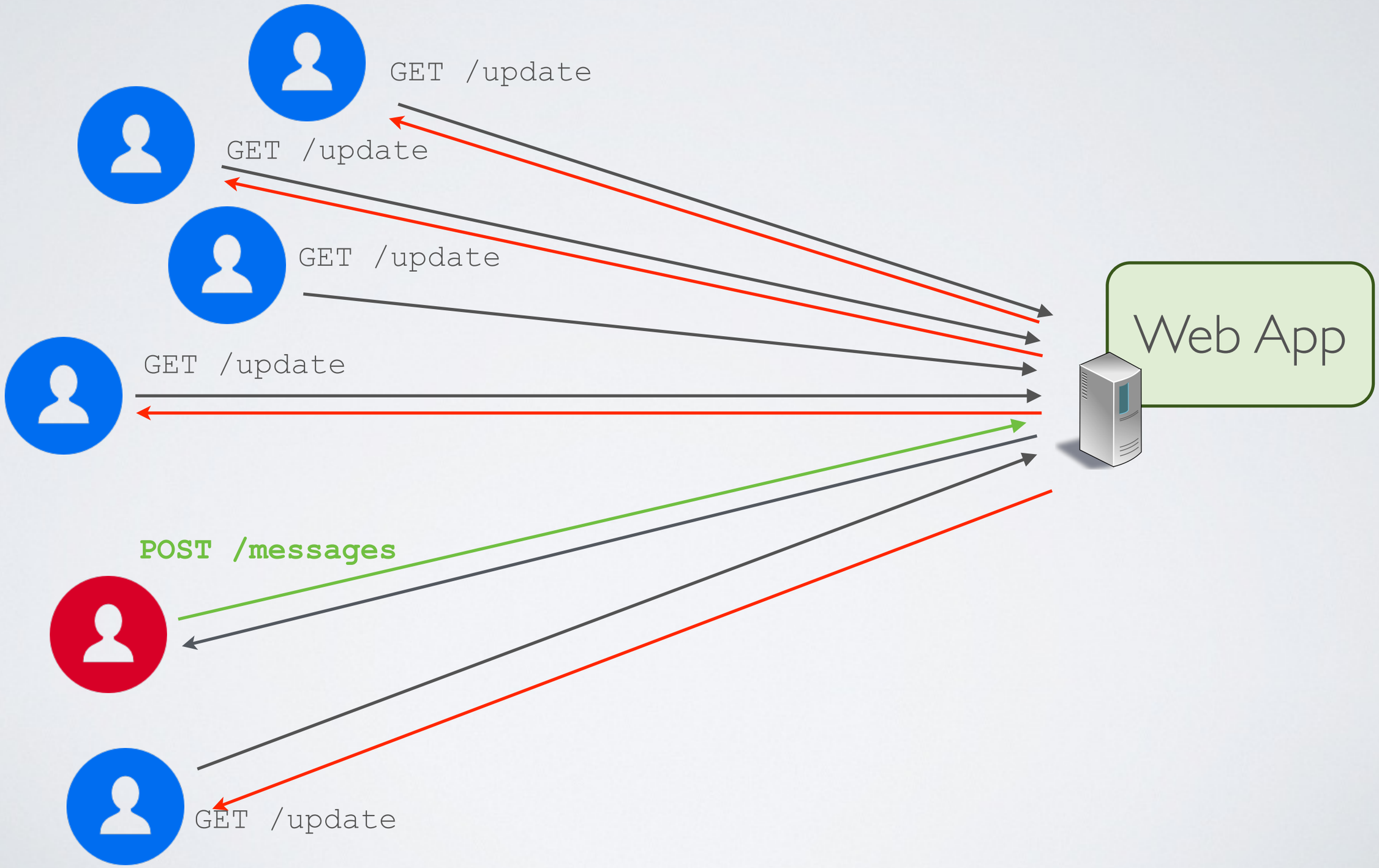
Long Polling

- The frontend request an update from the backend and wait for the response
- The backend replies to the update request only when there is an update

✓ No request/response wasted

✓ Updates are processed as soon as they arrived

Long Polling



Web Sockets

The idea

➡ Full-duplex client-server communication

- Similar to low-level POSIX sockets
- Does not rely on HTTP at all (except for initialization)

Web RTC

Real-time communication for the web

The idea

➡ Full-duplex communication between clients (browsers)

- P2P communications, perfect for sending text, video, audio without going through the server ...

- ... except for initialization and signalling that go through the server (usually using Web Sockets)

PWA

Progressive Web Applications

The idea

➡ A web application that can be installed on your system

- Relies on browser local storage to store the frontend (and checks for update with the server)

- Relies on Web-Workers for caching and communication