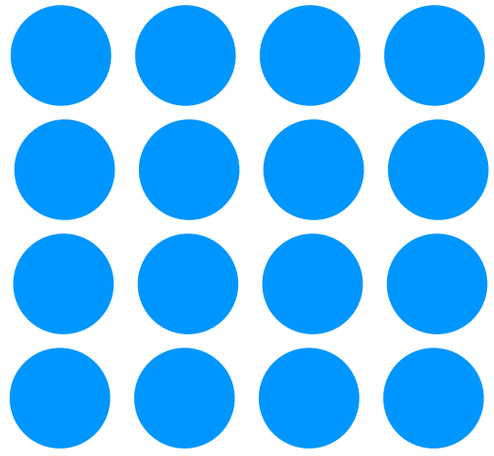
Lecture 3:

The Application Layer

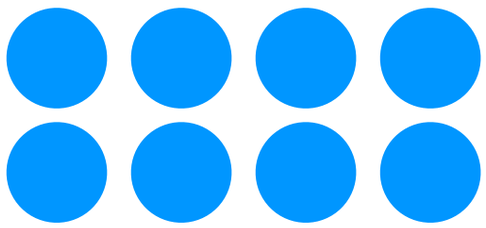
Katerina Argyraki, EPFL

- Tesla Model 3 controller
- your washing machine
- heart pacemaker



Google servers

- end-system
  - laptop
  - smartphone



World of warcraft servers

```
while (...) {
    message = ...;
    send ( message, ... );
}
```

processes

```
while (...) {
    message = receive ( ... );
}
```

Alice

Bob

IP address

port number

process name: 128.156.17.23, 80

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - what kind of delivery is needed?

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - What kind of delivery is needed?

a process that is always running

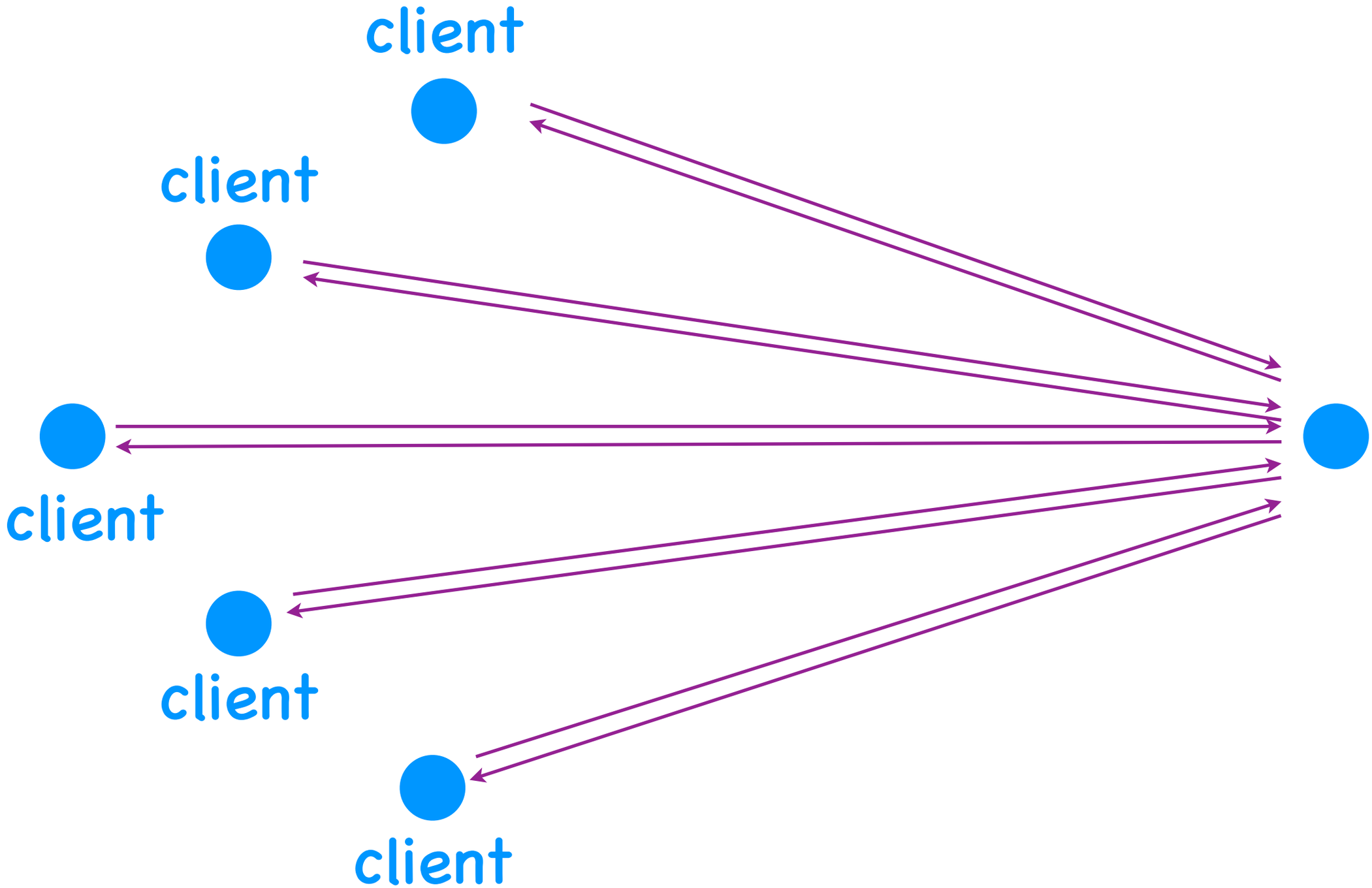
reachable at a fixed,
known process name

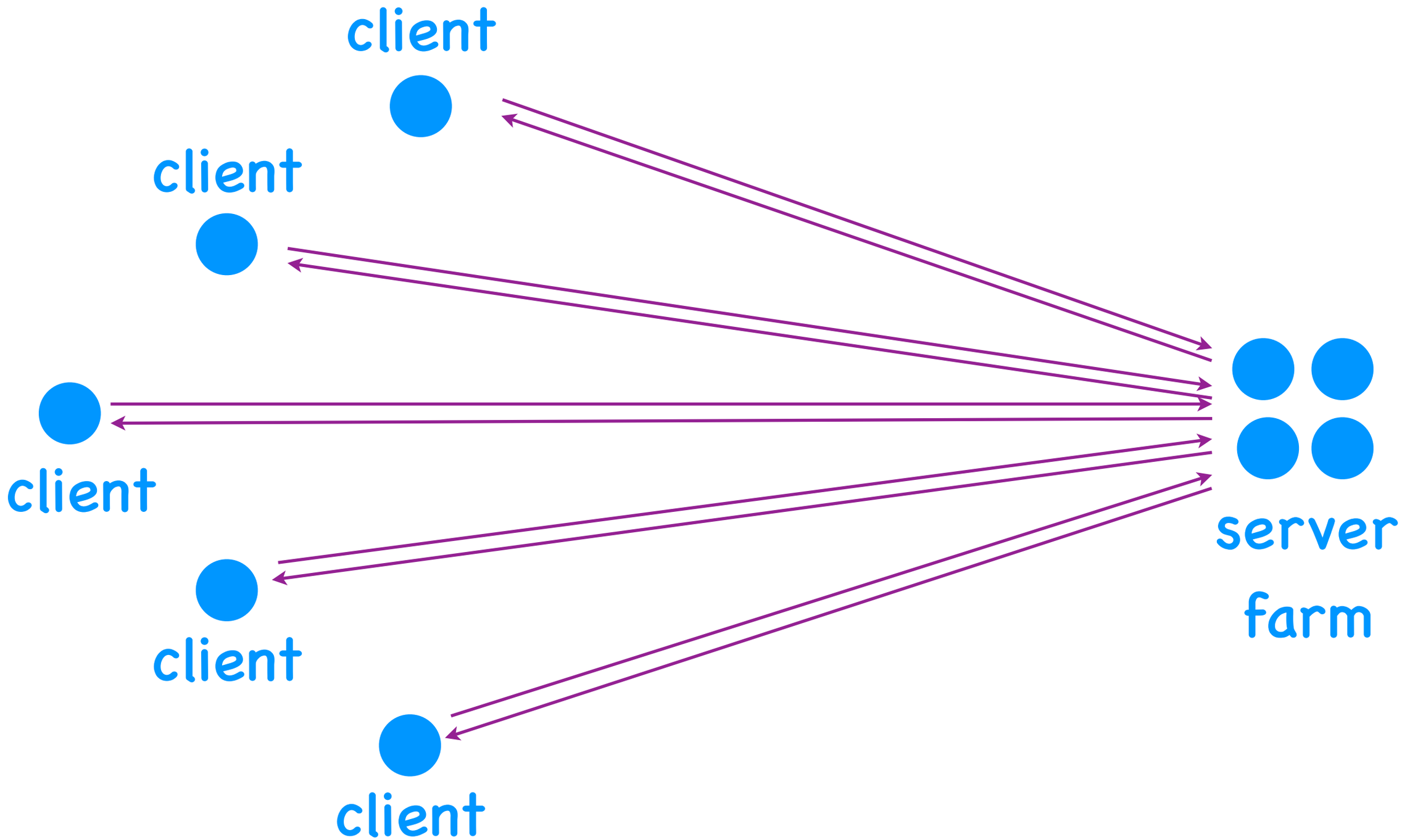
answers service requests

client

server

a process that generates
service requests


client
client
client
client
client
client


client
client
client
client
client
server
farm

Client-server architecture

- Clear separation of roles
  - a client generates service requests
  - a server answers (or denies) the requests
- Server runs on dedicated infrastructure
  - could be one computer
  - or an entire data-center

peer

peer

a process that may both
generate and answer requests

Peer-to-peer architecture

- A peer may act as both server and client
  - generates service requests
  - answer (or deny) requests
- Runs on personally owned end-system
  - PC, laptop, smartphone
  - no dedicated infrastructure

Client-server or peer-to-peer?

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport service
  - what delivery guarantees are needed?

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - what kind of delivery is needed?

Reliable data delivery

- Deliver message to the destination process or signal failure
  - detect and recover from packet loss or corruption
  - loss-sensitive applications, e.g., web, file transfer, email, …

Guaranteed performance

- Minimum throughput
  - throughput-sensitive applications,
    e.g., video-conferencing
- Maximum end-to-end packet delay
  - delay-sensitive applications,
    e.g., emergency services, voice, gaming, …

Guaranteed security

- Confidentiality
  - message is revealed only to the destination
- Authenticity
  - message indeed came from claimed source
- Data integrity
  - message is not changed along the way

Internet transport-layer protocols

- TCP: Transmission Control Protocol
  - reliable, in-order data delivery,
    flow control, congestion control
- UDP: User Datagram Protocol
  - detection of packet corruption
- No protocol offering guaranteed performance

application web file transfer email

SSL encryption, decryption, authentication, ...

transport TCP UDP

network

link

physical

TCP code at the destination keeps state on the source

TCP code at the source keeps state on the destination

Connection = memory

- TCP is "connection-oriented" or "stateful"
  = maintains state on all the local/remote process pairs that use TCP

- UDP is "connection-less" or "stateless"
  = does not maintain state on remote processes

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - what type of delivery is needed?

Example 1: the web

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport service
  - what delivery guarantees are needed?

a process that is always running

reachable at a fixed,
known process name

answers web requests

web client = web browser

web server

a process that generates
web requests

Web page

- Base file + referenced files
  - base file specifies structure and potentially content
  - referenced files can be images, video, scripts, …
- Each file has its own URL
  - URL = address for Internet resources
  - e.g., http://www.epfl.ch/index.html

Design an application =

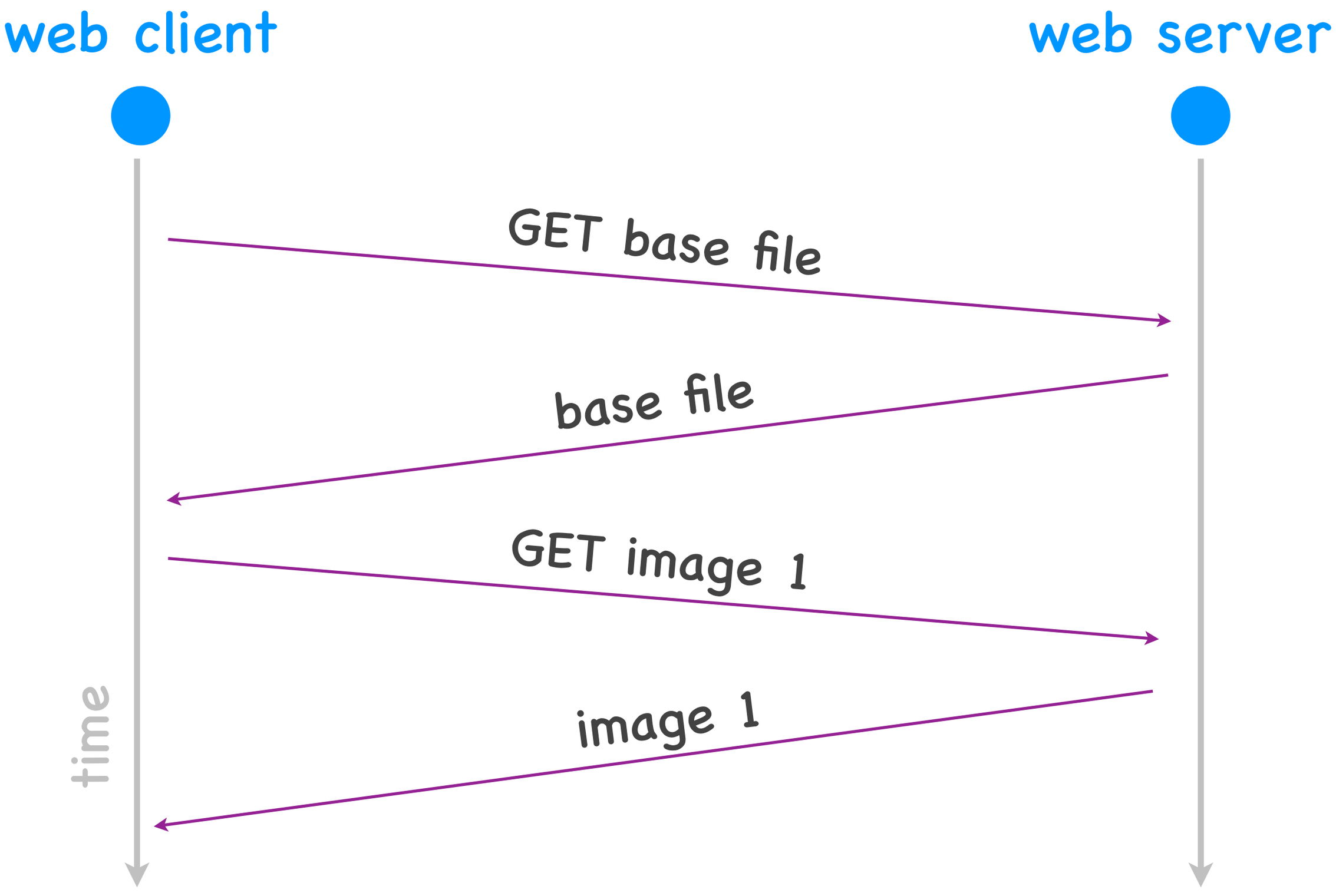
- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - what type of delivery is needed?

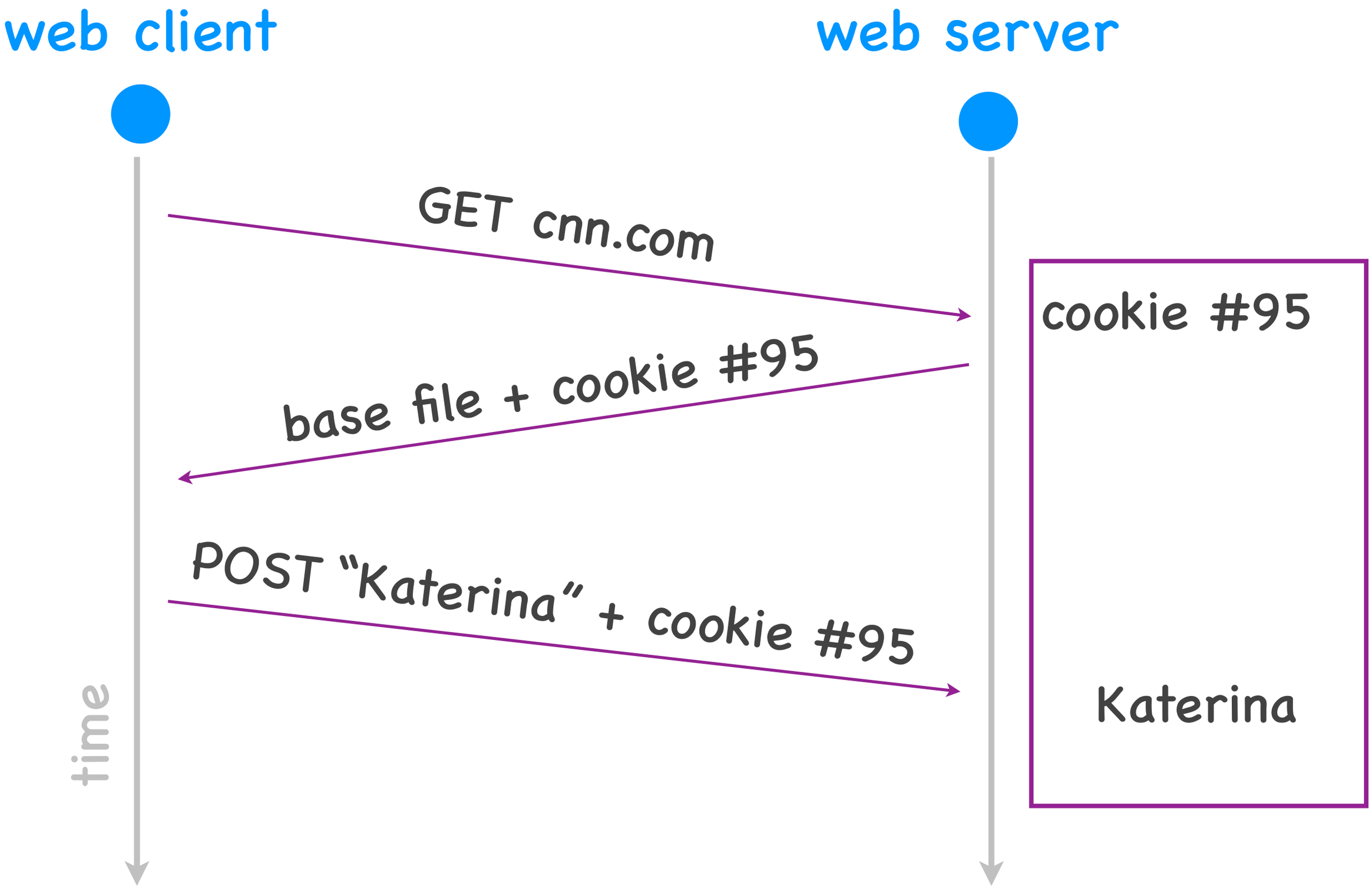
HTTP request types

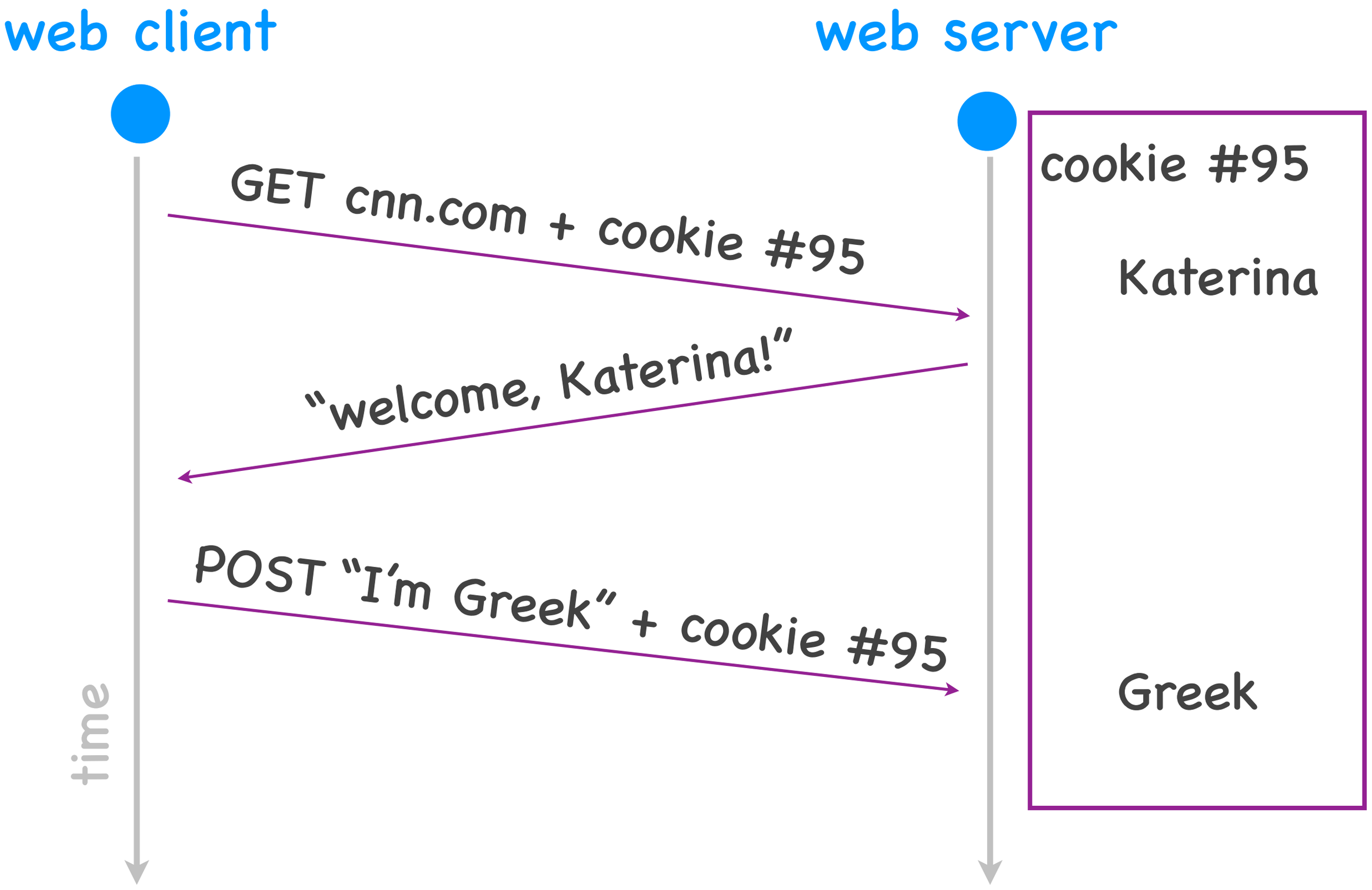
- GET: client requests to download a file
- POST: client provides information
- HEAD: client requests file metadata
- PUT: client requests to upload a file
- …

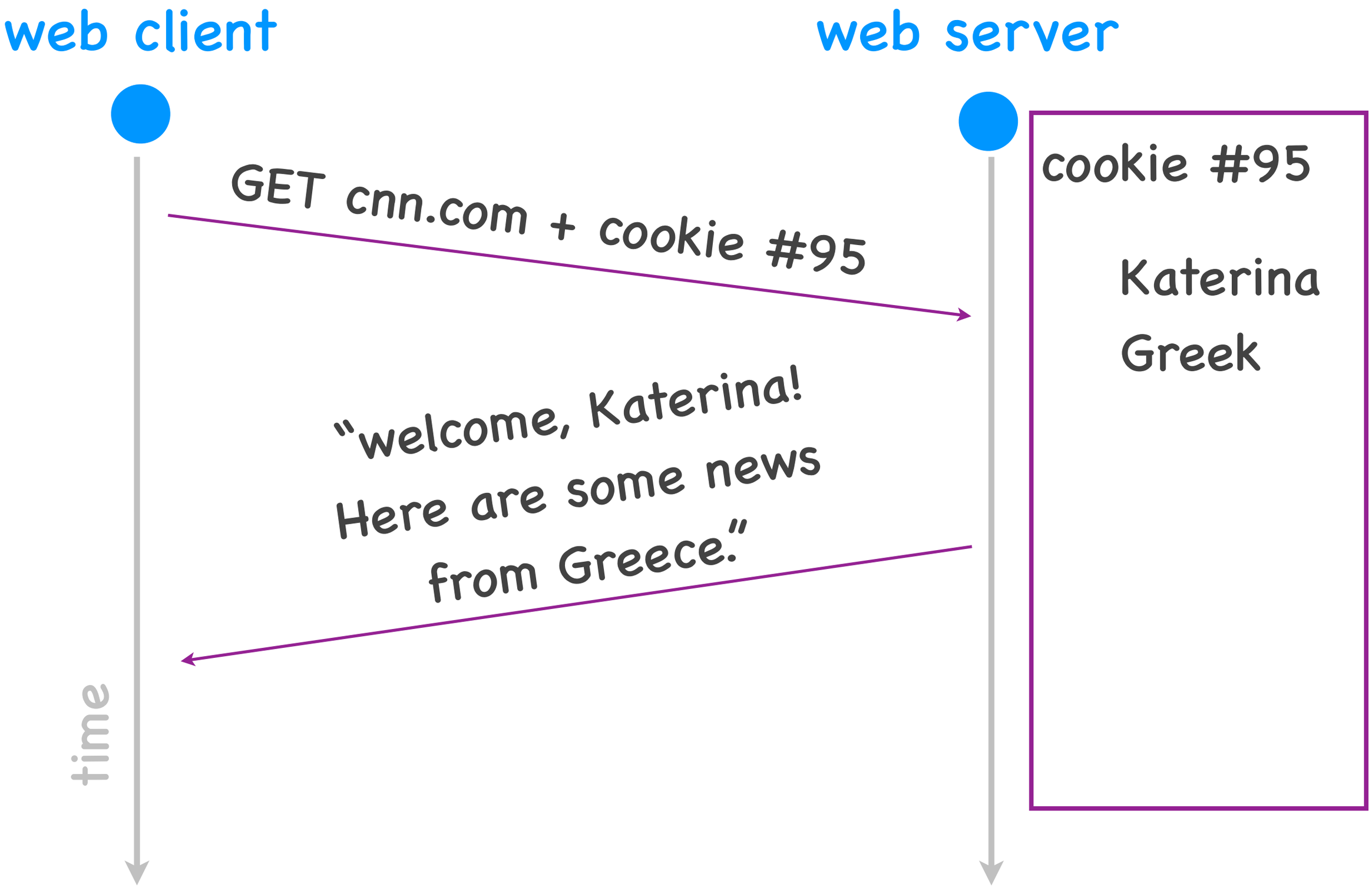
HTTP response types

- OK
- Not found
- Moved permanently
- Bad request
- …


web client
web server
GET base file
base file
GET image 1
image 1
time


web client
web server
GET cnn.com
base file + cookie #95
POST "Katerina" + cookie #95
time
cookie #95
Katerina


web client
web server
GET cnn.com + cookie #95
"welcome, Katerina!"
POST "I'm Greek" + cookie #95
time
cookie #95
Katerina
Greek


web client
web server
GET cnn.com + cookie #95
"welcome, Katerina!
Here are some news
from Greece."
time
cookie #95
Katerina
Greek

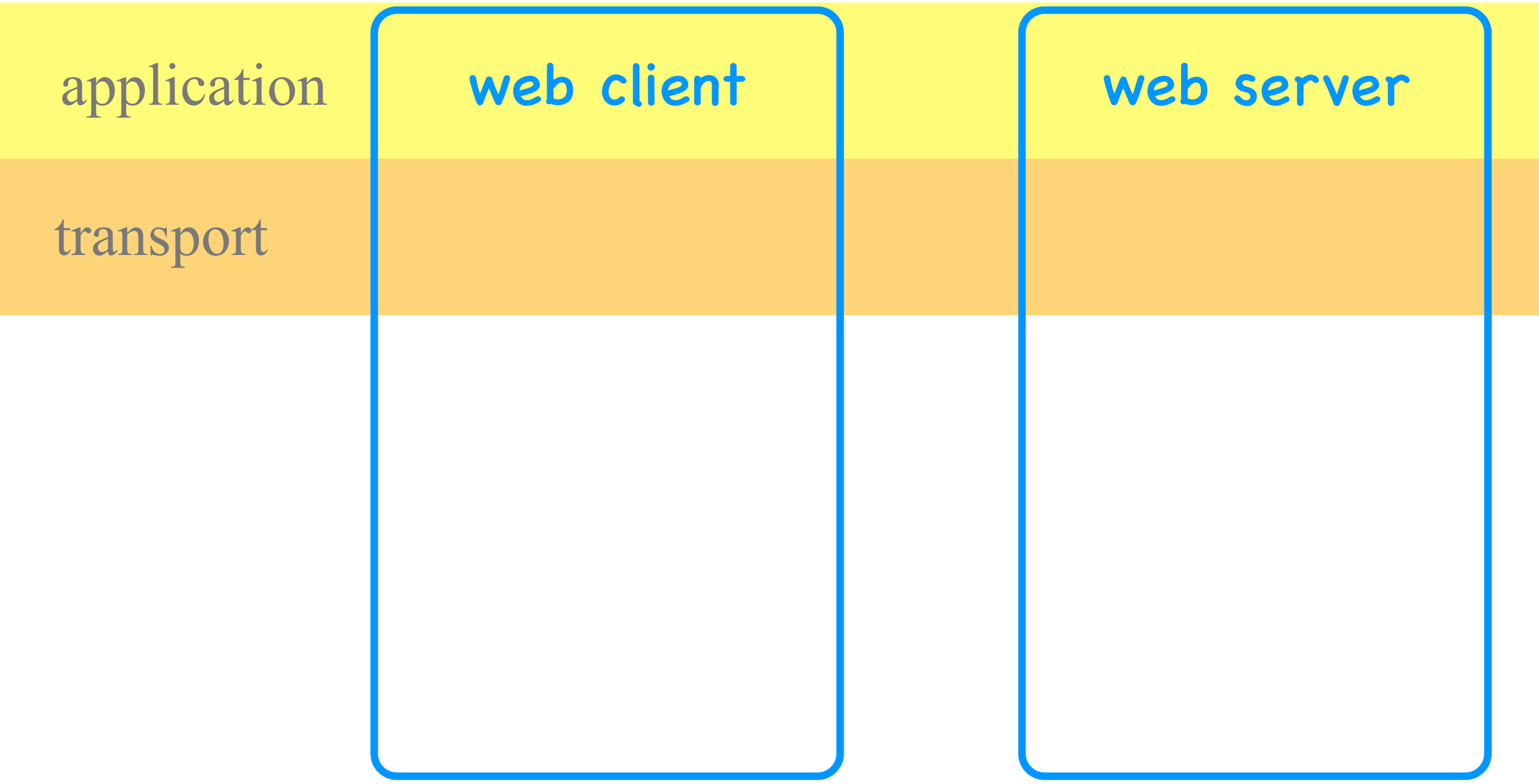
Cookies

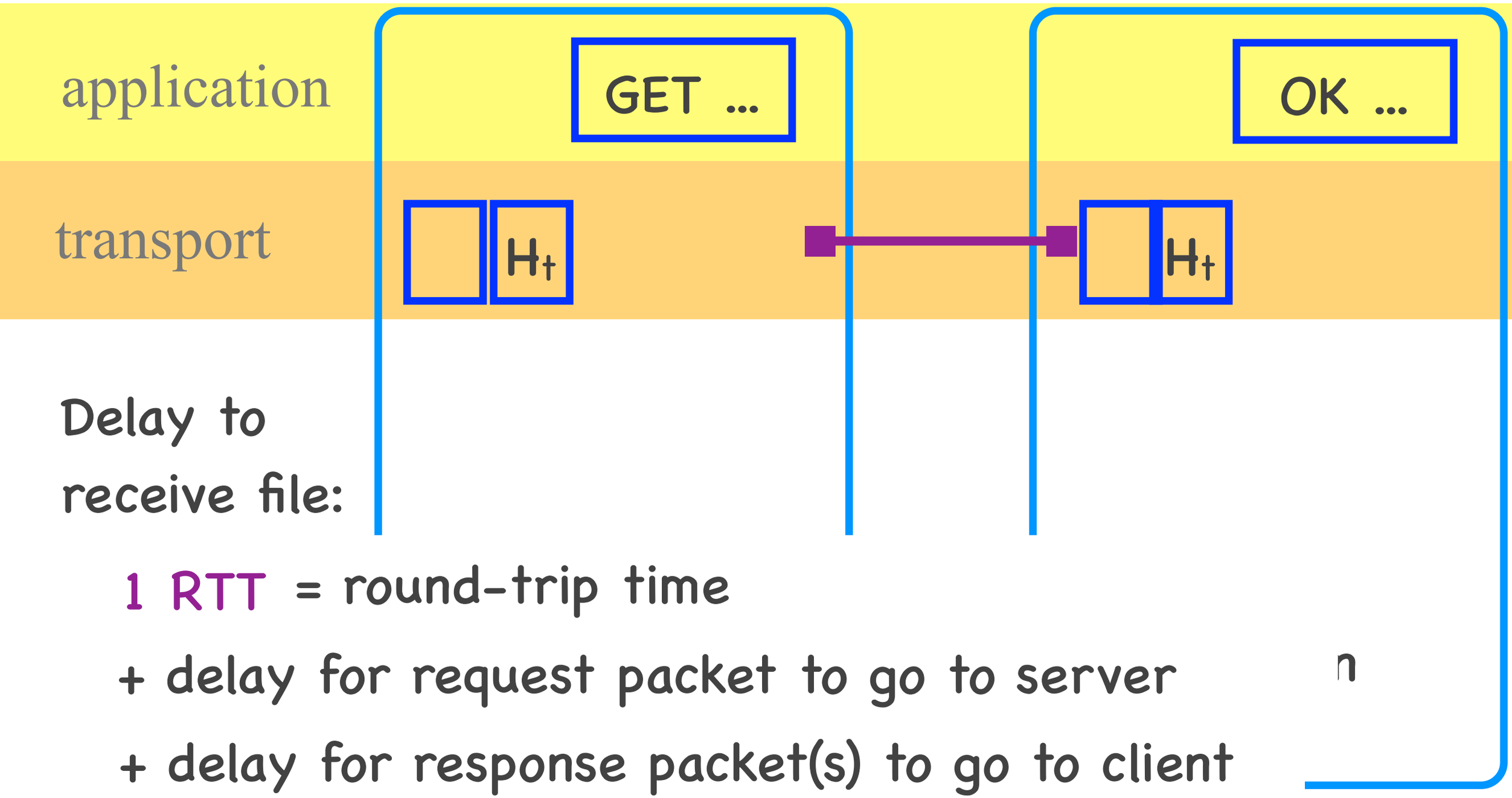
- Cookie = state created by the web server, stored by the web client and potentially the web server
- It links subsequent HTTP requests to the same web client

Anything wrong with cookies?

Design an application =

- Design the architecture
  - which process does what?
- Design the communication protocol
  - what sequences of messages can be exchanged?
- Choose the transport-layer technology
  - what kind of delivery is needed?


application
transport
web client
web server



Delay to receive file:

1 RTT = round-trip time

+ delay for request packet to go to server

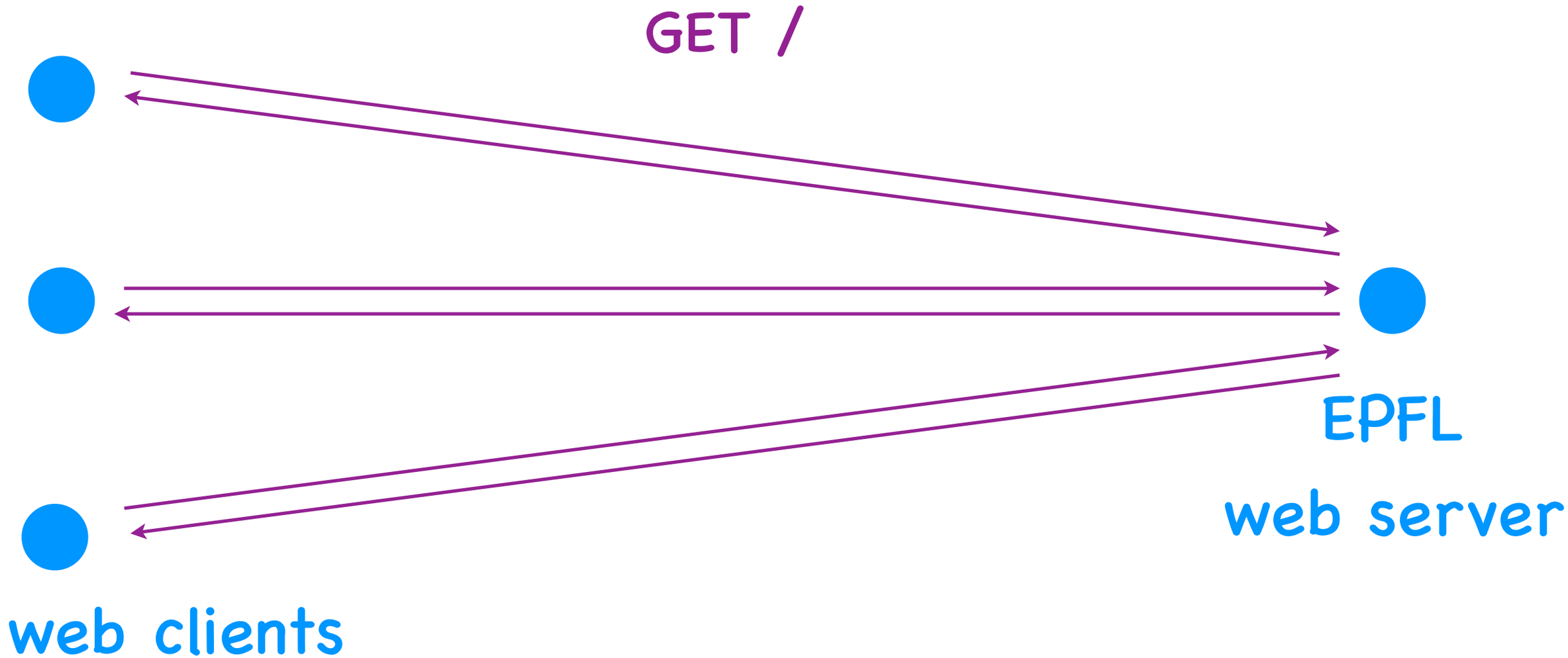
+ delay for response packet(s) to go to client

n

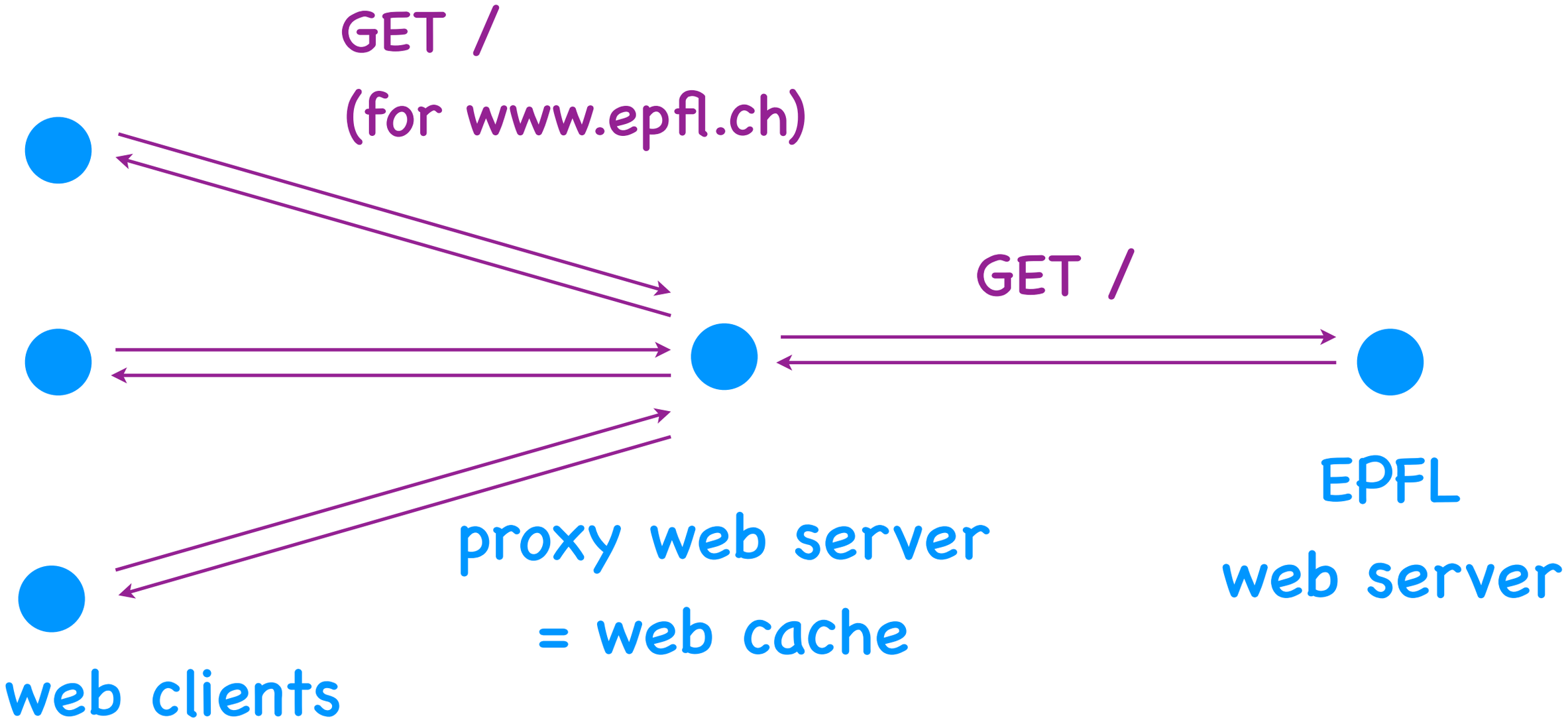
Typical ways to use TCP

- Persistent TCP connections
  - reuse the same TCP connection for multiple HTTP requests and responses
- Parallel TCP connections
  - exchange multiple HTTP requests and responses in parallel

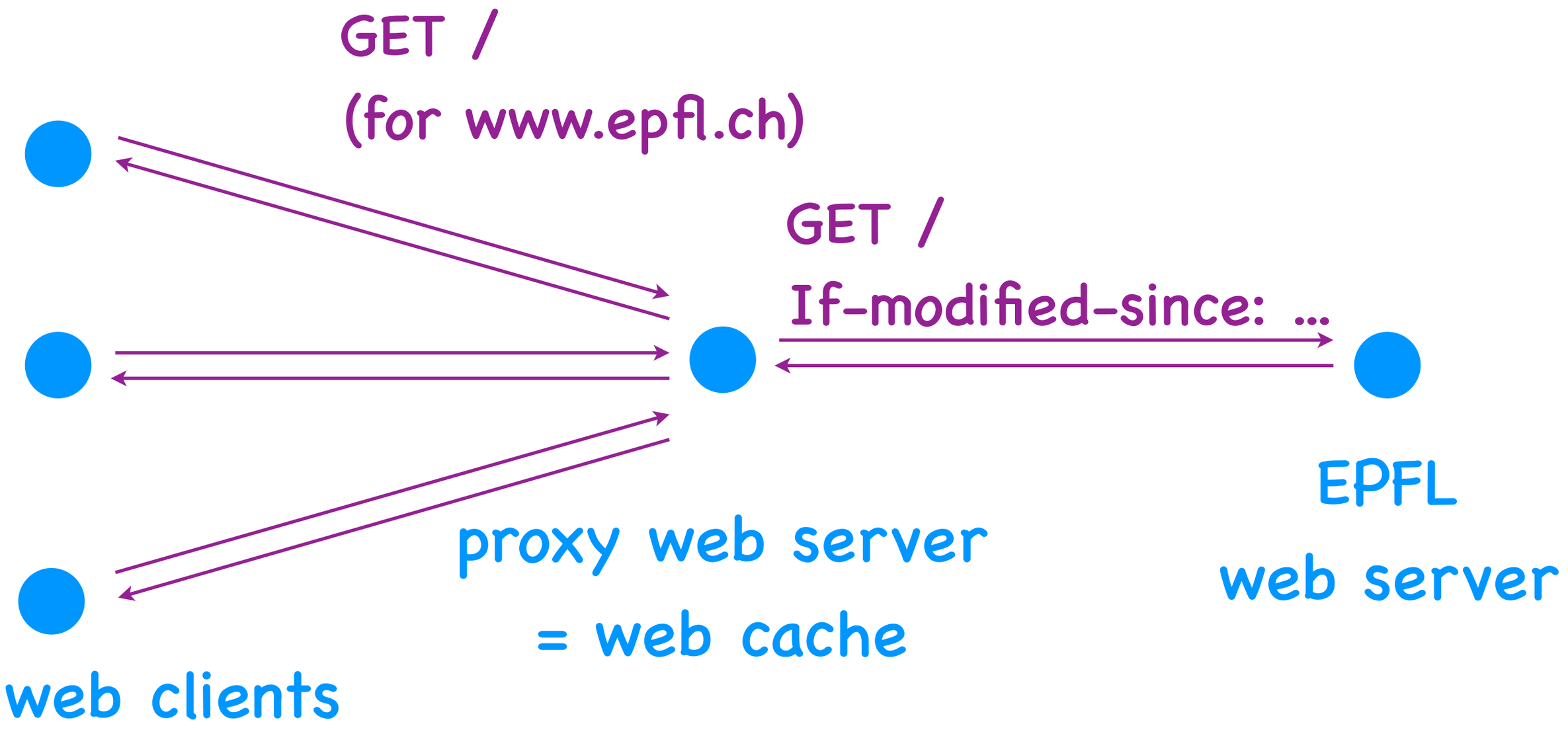
Silicon Valley



Silicon Valley



Silicon Valley



Web caching

- Proxy web server or web cache
  - caches copies of other web-server files
  - acts as a web server to nearby web clients
- Reduces delay experienced by web clients
- Relies on conditional GET to ensure data freshness

Caching

- Universal technique for improving performance
- Challenge: stale data
  - option #1: dynamic check for staleness
  - may introduce significant delay