

CS3640

---

# Application Layer (3): The Web & HTTP

**Prof. Supreeth Shastri**

*Computer Science*

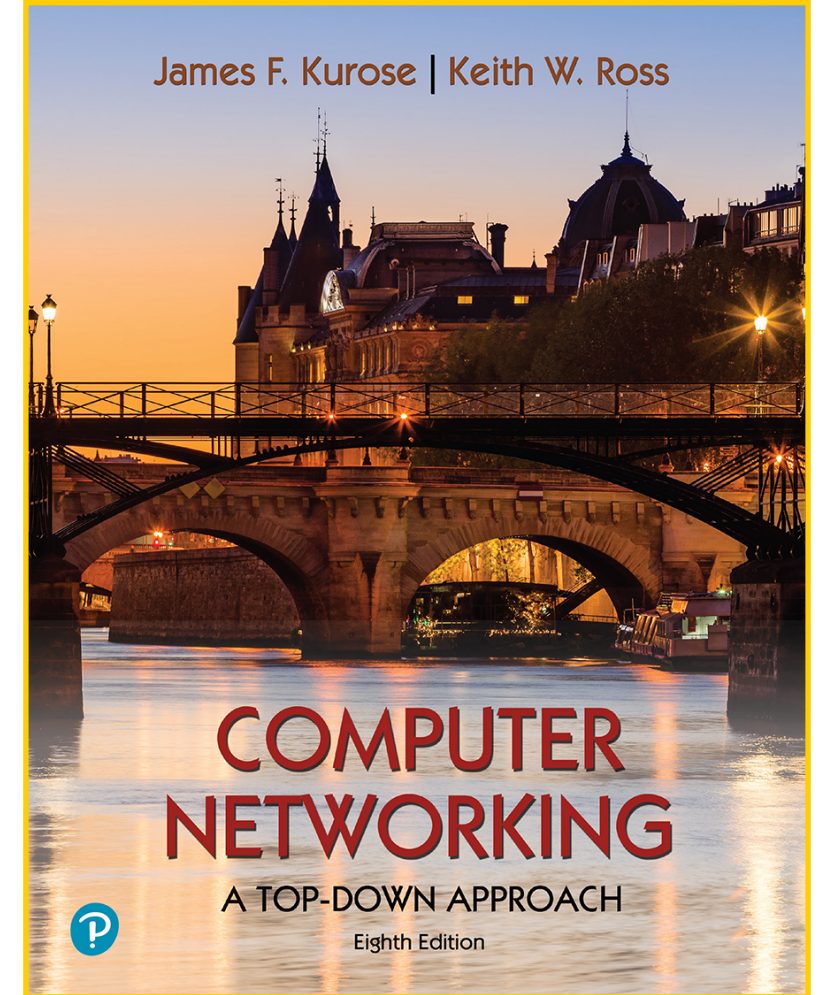
*The University of Iowa*

# Lecture goals

---

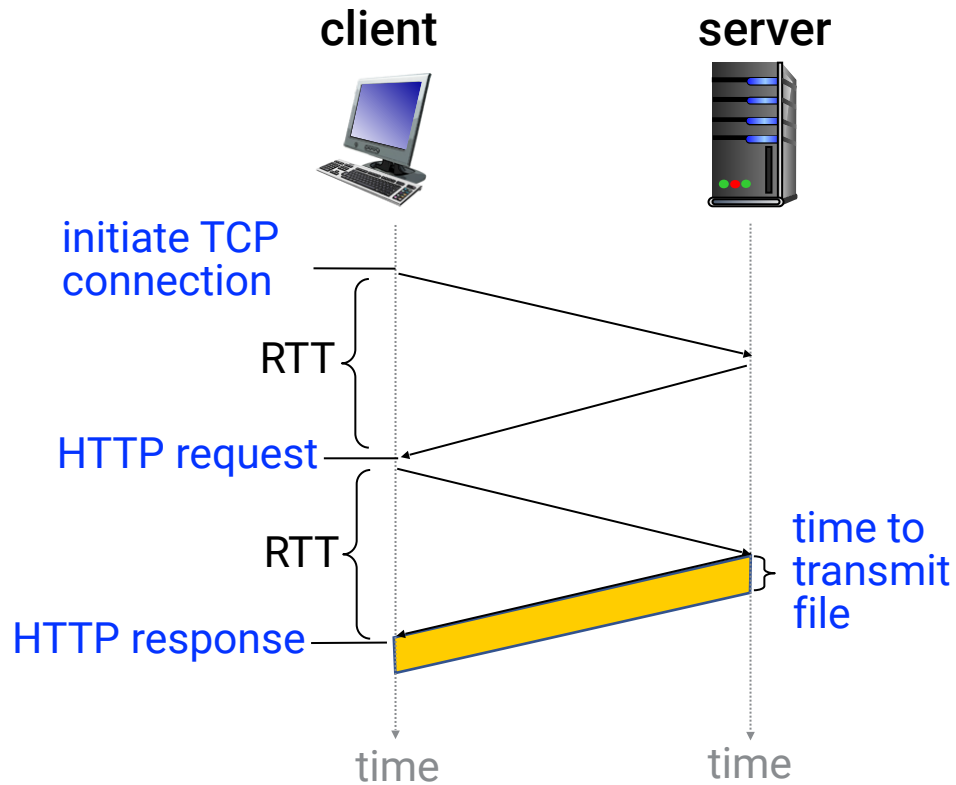
*Deep dive into the design and operation of the world wide web*

- *HTTP*
- *Web cookies*
- *Four Optimizations*



Chapter 2.2

# Recap: HTTP



## Protocol specs

- Human readable ASCII format
- Client-server model
- Uses TCP for reliable transmission

## HTTP messages

- Two messages: request and response
- **Structure:** request/status line, then a set of headers, and finally, a body
- Five modes of request, and five categories of responses

## Connection persistence

- HTTP/1.0 takes 2 RTTs per object
- HTTP/1.1 allows keeping the TCP connection open, reducing the response time to 1 RTT (on average)

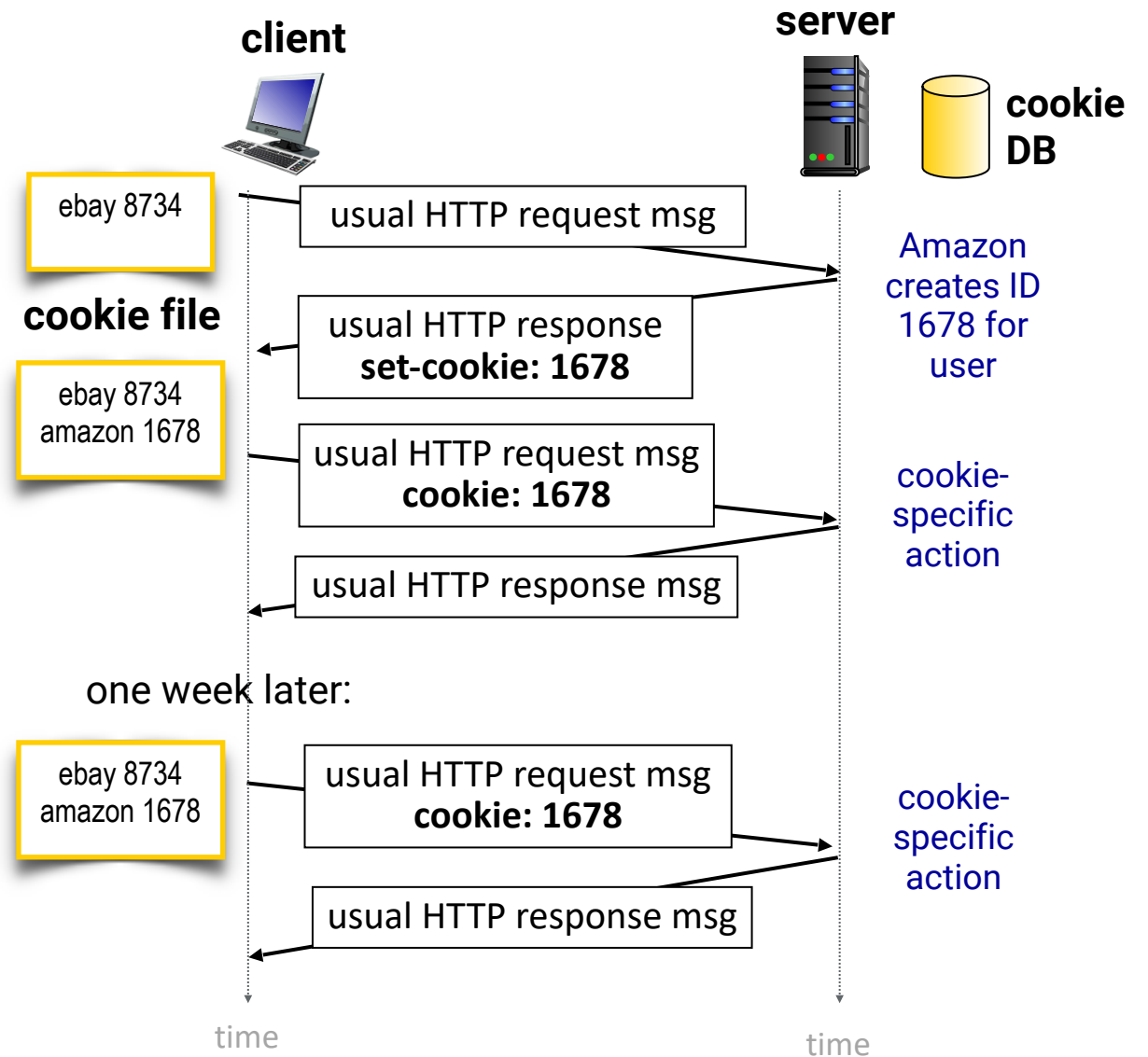
# Recap: HTTP Cookies

## HTTP is stateless

- no notion of multiple HTTP msg completing a “transaction”
- yet, many use cases of the web required maintaining the state

## Solution: web cookies

- RFC 6265 proposes a standard mechanism for state management
- **key components:** cookie headers, client cache file, server cache db
- **side effects:** third-party tracking, need for privacy regulations



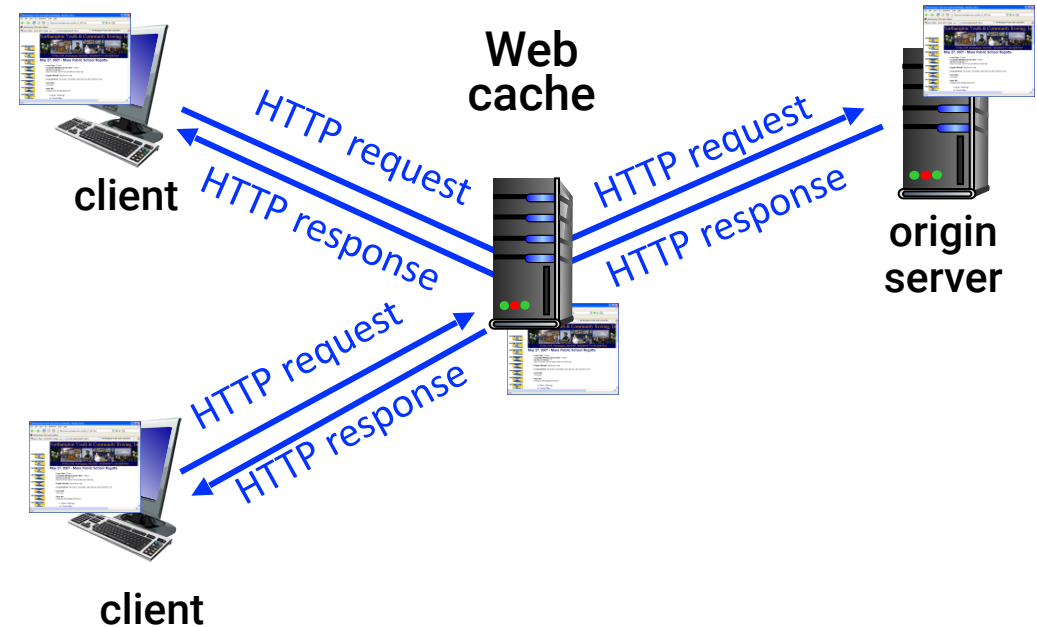
1

# Web Caches

# Web caches

**Goal:** satisfy client requests without involving origin server

- web users configure their browser to point to a (local) **web cache/proxy**
- browser sends all HTTP requests to the cache (instead of the web server)
- if object in cache: cache simply returns the object. Otherwise, it requests object from origin server, caches received object, then returns object to client





## Cache mechanics

- web cache acts both as server (*to the requesting client*) and as client (*to the origin server*)
- origin server informs the cache about a given object's ability to be cached (via response header)

```
Cache-Control: max-age=<seconds>
```

```
Cache-Control: no-cache
```

## Why cache?

- reduces response time for client requests (*since cache is closer to client*)
- reduces traffic on an institution's access link
- allows resource-starved content providers and administrators to deliver content more effectively

**Internet is dense with web caches**

*Do web caches violate end-to-end principle?*

# A web access scenario

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- size of web objects: 100Kb
- users make 15 request/sec (avg) for web objects
- avg data rate to institutional router = 1.5 Mbps

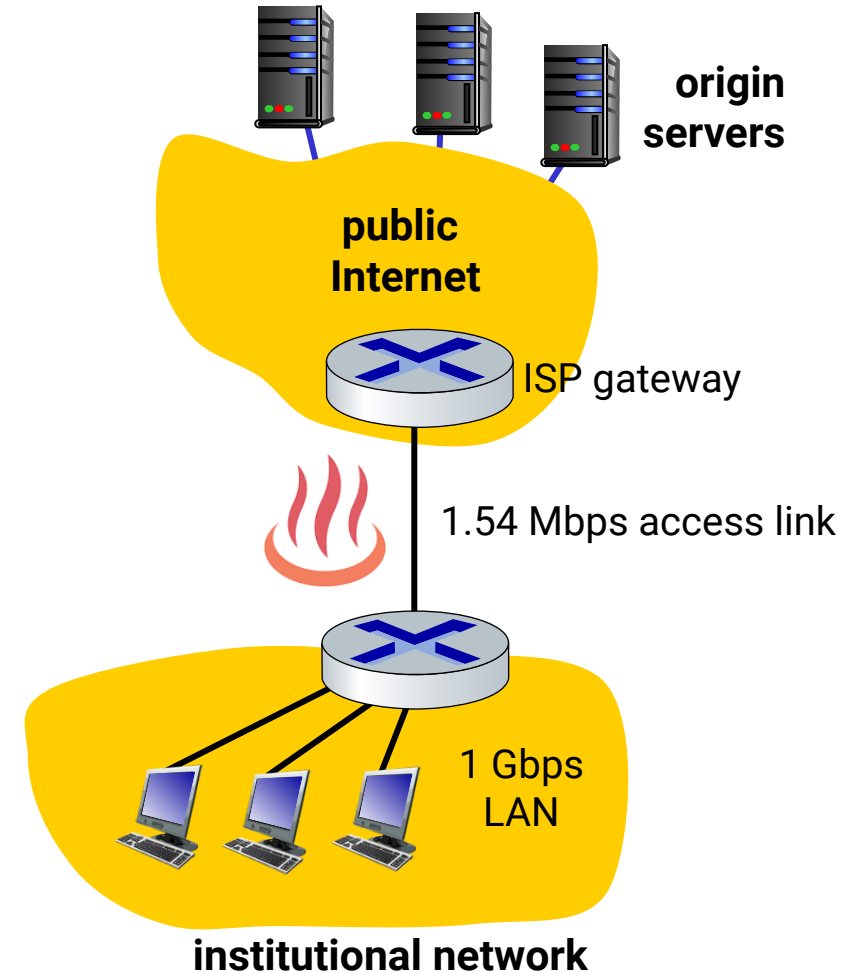
## End to end delay

access link utilization =  $1.5\text{Mbps} / 1.54\text{Mbps} = 97\%$

end-to-end delay

= Internet delay + access link delay + LAN delay

= 2 sec + several minutes +  $\mu\text{sec}$





## Option-1: buy a faster access link

- access link rate: ~~1.54 Mbps~~ **154 Mbps**
- RTT from institutional router to server: 2 sec
- size of web objects: 100Kb
- users make 15 request/sec (avg) for web objects
- avg data rate to institutional router = 1.5 Mbps

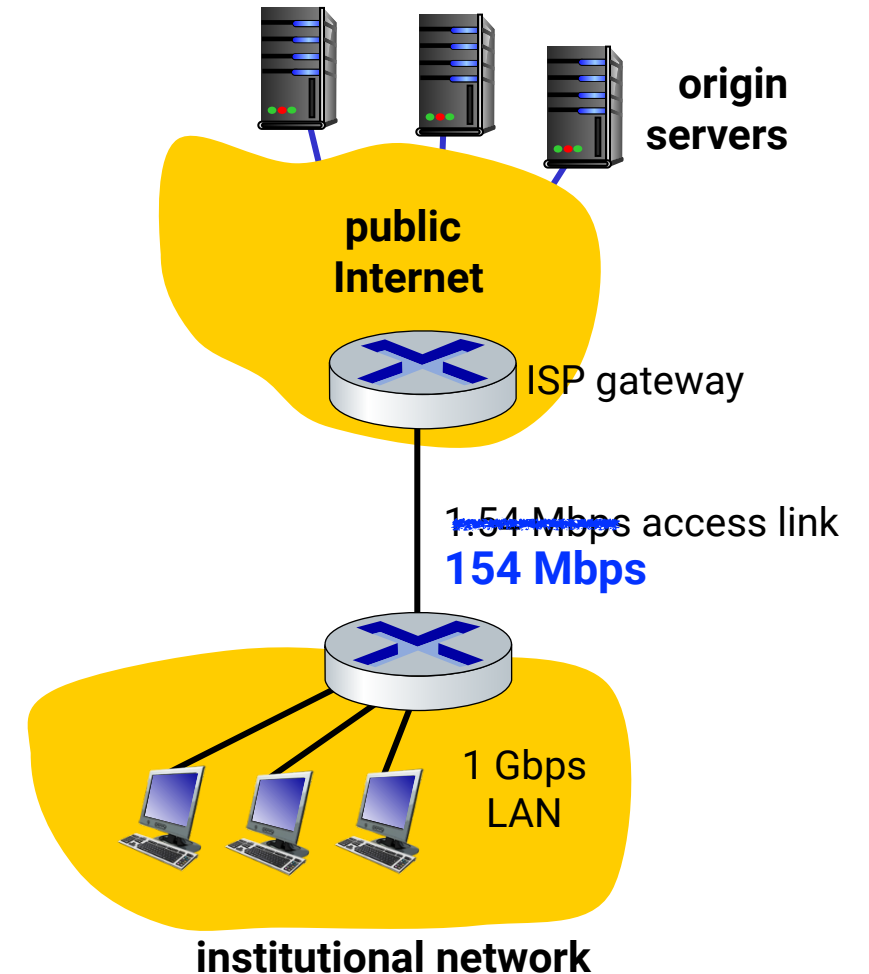
## End to end delay

access link utilization =  $1.5\text{Mbps} / 154\text{Mbps} = 0.97\%$

end-to-end delay

= Internet delay + access link delay + LAN delay

= 2 sec + ~~several minutes~~ +  $\mu\text{sec}$   $\approx$  **2 sec**  
 **$\mu\text{sec}$**



Faster access links are **expensive!**

## Option-2: install a web cache

### Cache hit rate

- fraction of requests fulfilled locally by the cache
- its value tends to be 20-70% in practice
- *suppose our cache has a hit rate of 40%*

### For requests satisfied at origin server

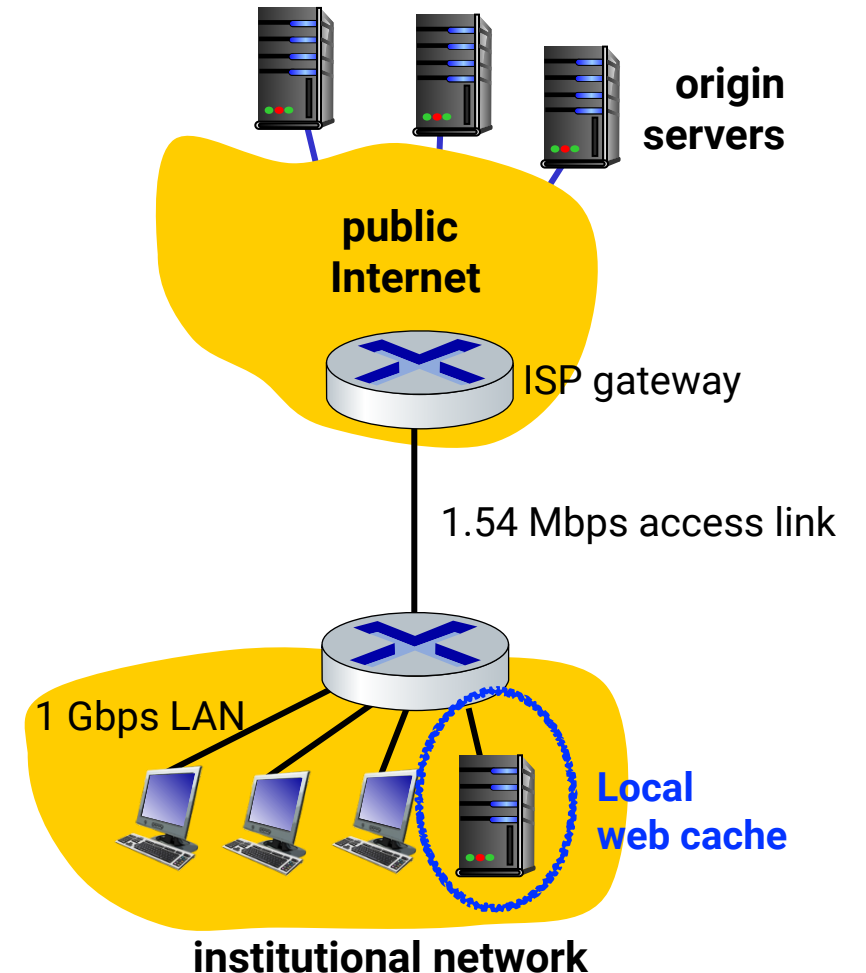
access link utilization =  $(1.5\text{Mbps} * 60\%) / 1.54\text{Mbps} = 58\%$   
this results in low ( $\mu\text{sec}$ ) queueing delay at access link

### For requests satisfied at cache

delay = LAN delay =  $\sim \mu\text{sec}$

### Average end-end delay

$$\begin{aligned} &= 60\% * (\text{delay to origin servers}) + 40\% * (\text{delay to cache}) \\ &= 60\% * (2 \text{ sec}) + 40\% * (\sim \mu\text{sec}) = \sim \mathbf{1.2\text{sec}} \end{aligned}$$



Cache achieves **lower delay at cheaper cost!**

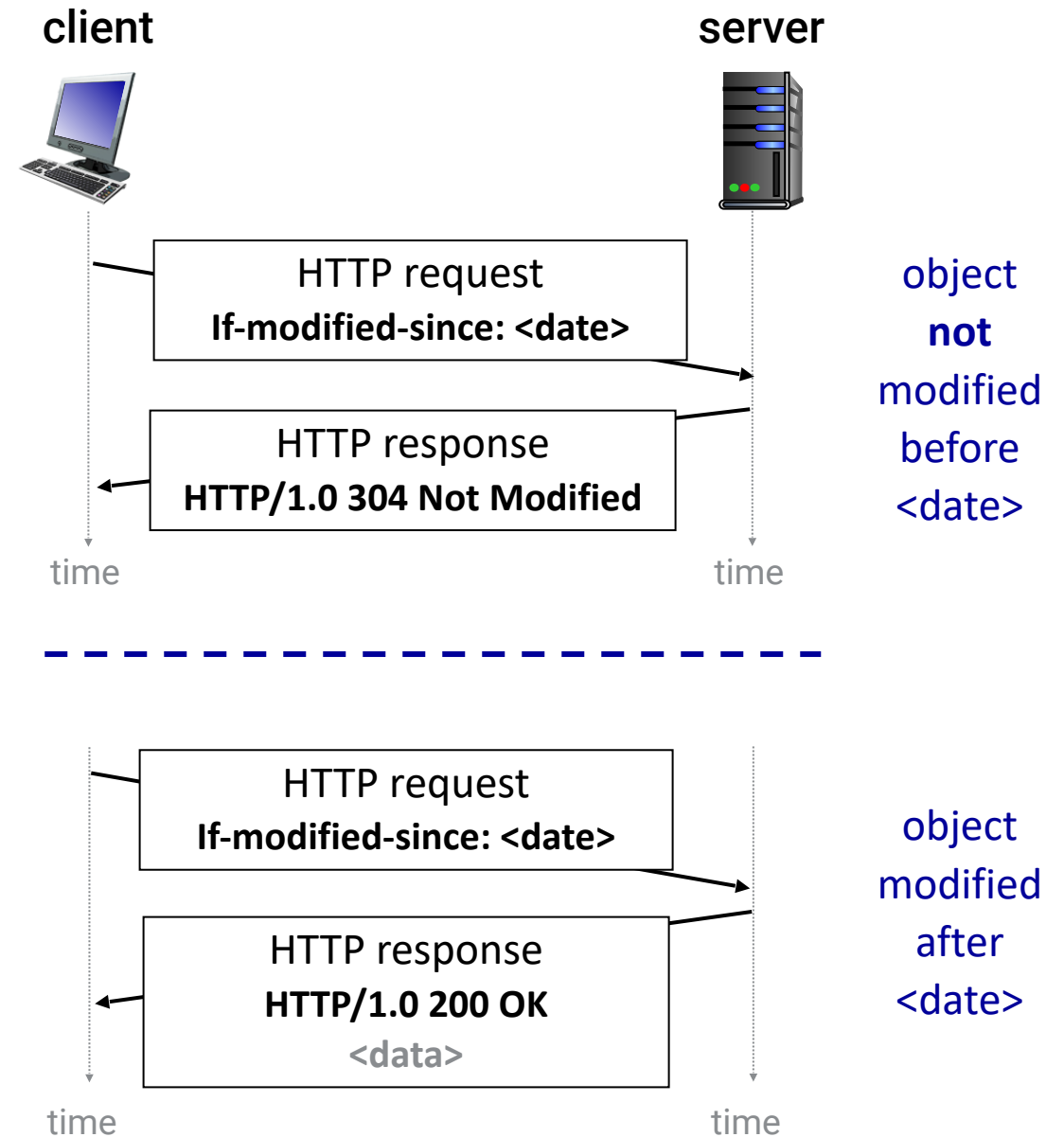
2

## **Conditional GET**

# Conditional GET

**Goal:** *don't send object if the client already has up-to-date version*

- no object transmission delay and no use of network resources
- **client** will specify date of cached copy using **if-modified-since** header in HTTP request
- **server** will respond with status code of **304 not modified** with an empty body if the cached copy is up to date



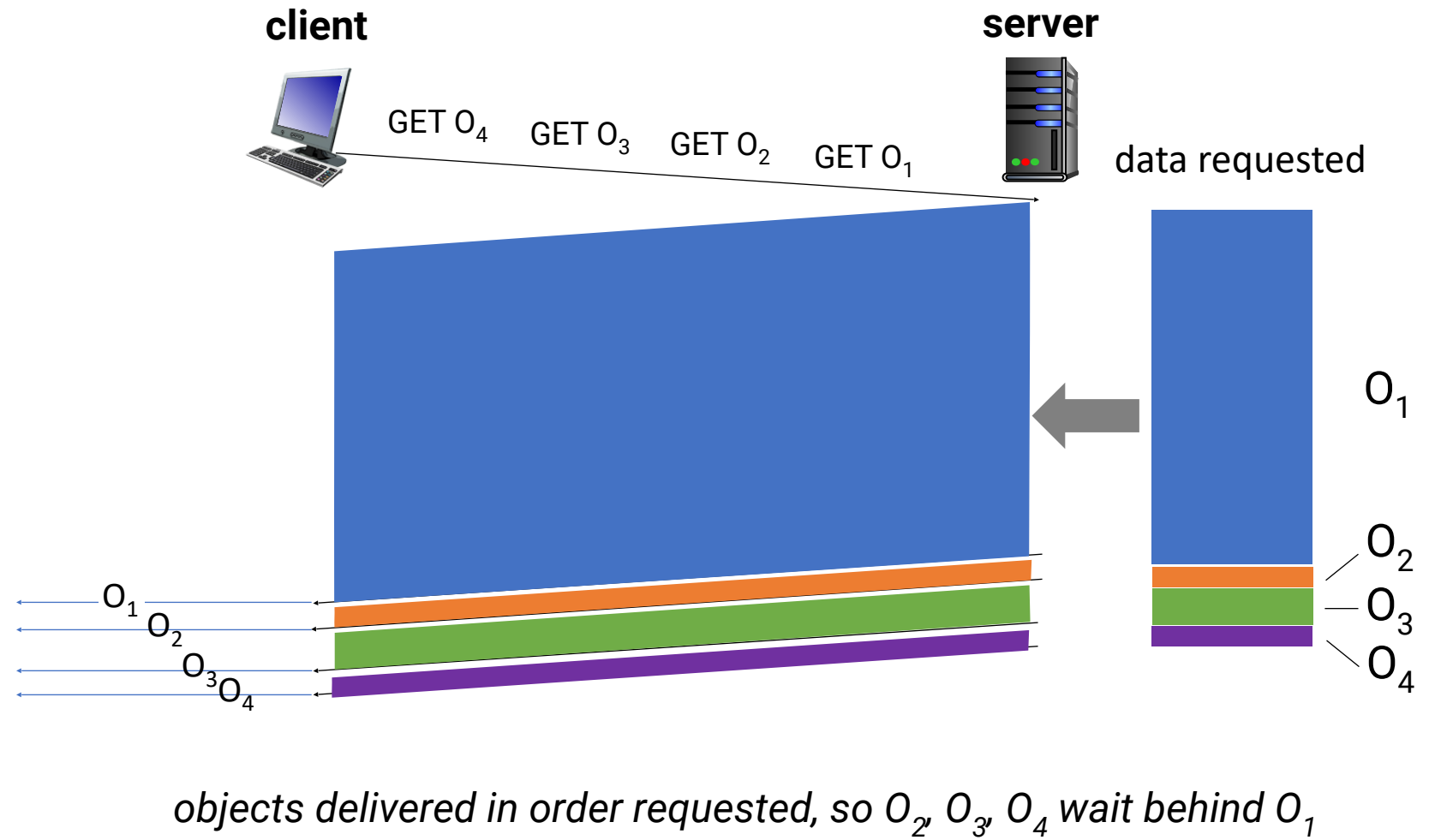
3

## HOL Blocking

# HTTP/1.1: HOL blocking

- HTTP/1.1 introduced multiple, pipelined GETs over a single TCP connection
- HTTP server responds *in-order* (FCFS: first-come-first-serve) to GET requests
- FCFS could result in small objects having to wait for transmission behind large object(s), known as **head of line (HOL) blocking**

*Example: a client requests 1 large object and 3 smaller objects*



# HTTP/2

*Key goal: decrease the delay in multi-object HTTP requests*

- transmission order of objects is based on client-specified priority (not necessarily FCFS)
- divide objects into frames, and then schedule frames to mitigate HOL blocking
- push unrequested objects to client
- methods, status codes, most header fields unchanged from HTTP/1.1

Internet Engineering Task Force (IETF)  
Request for Comments: 7540  
Category: Standards Track  
ISSN: 2070-1721

M. Belshe  
BitGo  
R. Peon  
Google, Inc  
M. Thomson, Ed.  
Mozilla  
May 2015

## Hypertext Transfer Protocol Version 2 (HTTP/2)

### Abstract

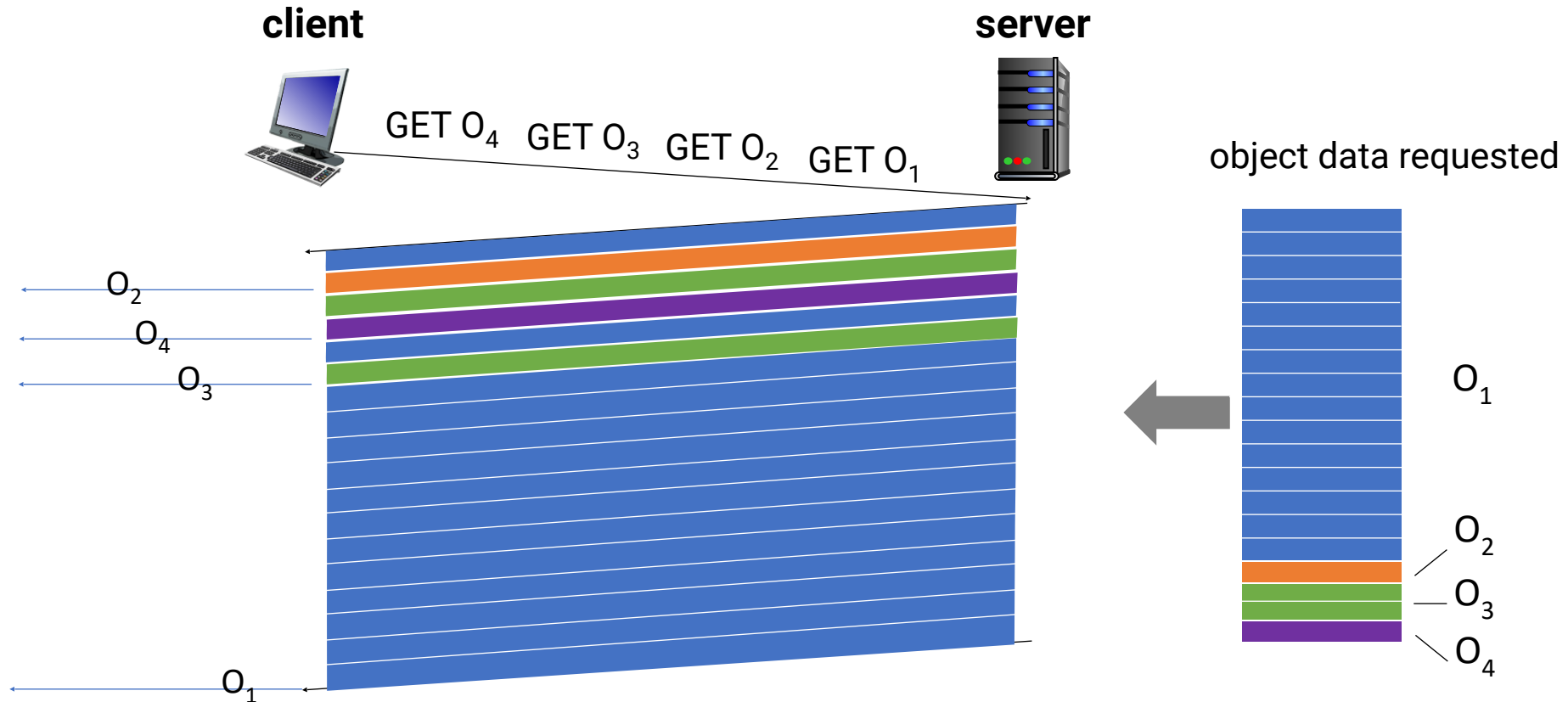
This specification describes an optimized expression of the semantics of the Hypertext Transfer Protocol (HTTP), referred to as HTTP version 2 (HTTP/2). HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing header field compression and allowing multiple concurrent exchanges on the same connection. It also introduces unsolicited push of representations from servers to clients.

This specification is an alternative to, but does not obsolete, the HTTP/1.1 message syntax. HTTP's existing semantics remain unchanged.



# HTTP/2: mitigating HOL blocking

*Divide objects into frames, and interleave frame transmission*



*O<sub>2</sub>, O<sub>3</sub>, O<sub>4</sub> delivered quickly, O<sub>1</sub> slightly delayed*

4

Moving away from **TCP**

# HTTP/2 → HTTP/3

## HTTP/2 shortcomings

*all HTTP traffic flows over a single TCP connection*

- recovery from a packet loss stalls all subsequent object transmissions
- thus, browsers are still compelled to open multiple parallel TCP connections to reduce stalling, and improve throughput

offers no built-in security

Internet Engineering Task Force (IETF)  
Request for Comments: 9114  
Category: Standards Track  
ISSN: 2070-1721

M. Bishop, Ed.  
Akamai  
June 2022

### HTTP/3

#### Abstract

The QUIC transport protocol has several features that are desirable in a transport for HTTP, such as stream multiplexing, per-stream flow control, and low-latency connection establishment. This document describes a mapping of HTTP semantics over QUIC. This document also identifies HTTP/2 features that are subsumed by QUIC and describes how HTTP/2 extensions can be ported to HTTP/3.

## HTTP/3 (released a short while ago)

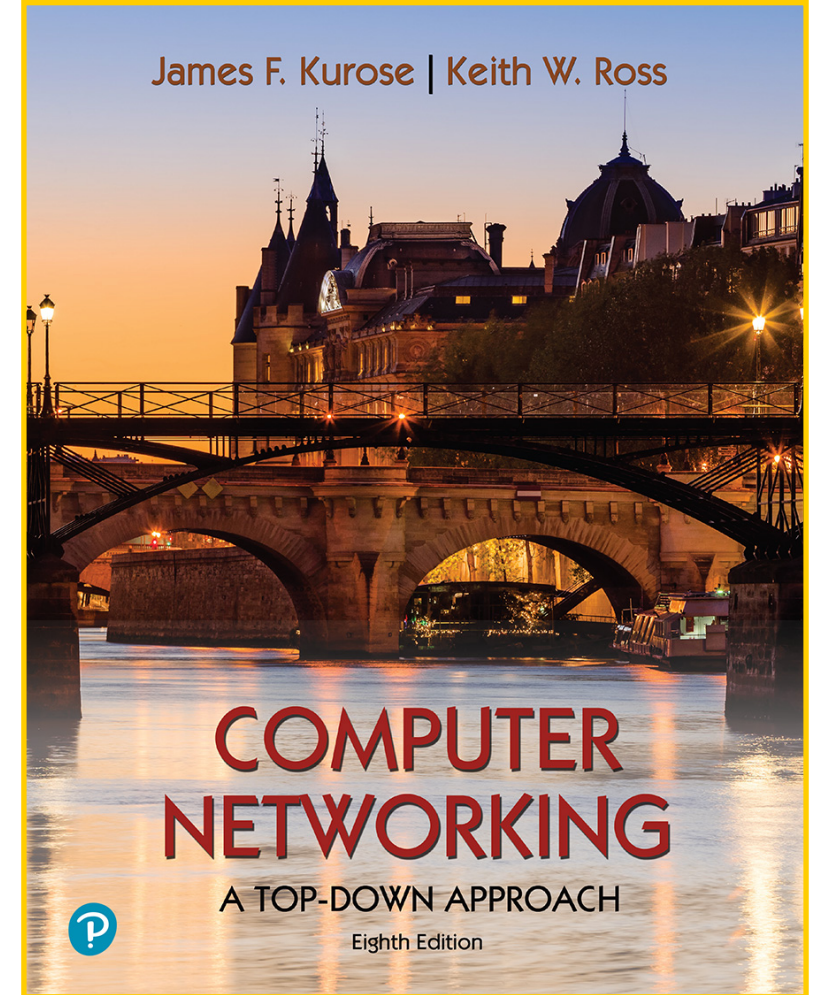
- replaces TCP with QUIC
- allows per object error- and congestion-control
- offers security natively

# Next lecture

---

*Understand the protocols and mechanics of the electronic mail*

- *Email infrastructure*
- *SMTP*
- *IMAP*



Chapter 2.3

# **Spot Quiz (ICON)**