רשתות מחשבים ואינטרנט 2

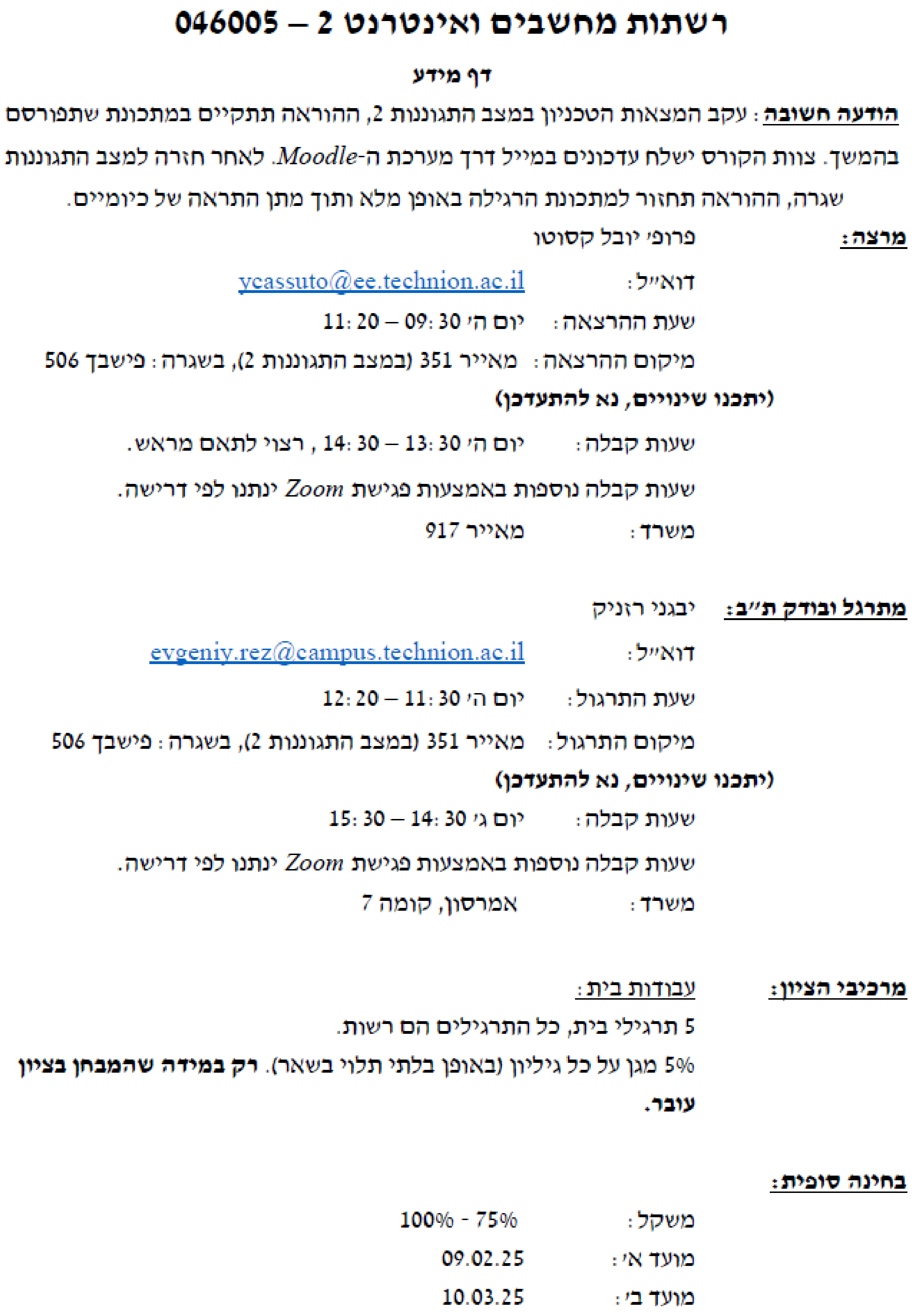
(046005)

חורף – 2024/25

Course Objectives (vs. Networks 1)

1. Know more about the **Internet**
   1. Applications (new: distributed and peer-to-peer)
   2. Protocols (new: details of TCP, new: routing protocols)

2. Know more **Networking tools**
   1. Queuing theory (advanced: queuing networks)
   2. Routing algorithms (new)
   3. Network optimization algorithms (new)



רשתות מחשבים ואינטרנט 2 – 046005

דף מידע

<u>הודעה חשובה</u>: עקב המצאות הטכניון במצב התגוננות 2, ההוראה תתקיים במתכונת שתפורסם בהמשך. צוות הקורס ישלח עדכונים במייל דרך מערכת ה-*Moodle*. לאחר חזרה למצב התגוננות שגרה, ההוראה תחזור למתכונת הרגילה באופן מלא ותוך מתן התראה של כיומיים.

<u>מרצה:</u> פרופ׳ יובל קסוטו

דוא״ל: ycassuto@ee.technion.ac.il

שעת ההרצאה: יום ה׳ 09: 30 – 11: 20

מיקום ההרצאה: מאייר 351 (במצב התגוננות 2), בשגרה: פישבך 506

(יתכנו שינויים, נא להתעדכן)

שעות קבלה: יום ה׳ 13: 30 – 14: 30 , רצוי לתאם מראש.

שעות קבלה נוספות באמצעות פגישת *Zoom* ינתנו לפי דרישה.

משרד: מאייר 917

<u>מתרגל ובודק ת״ב:</u> יבגני רזניק

דוא״ל: evgeniy.rez@campus.technion.ac.il

שעת התרגול: יום ה׳ 11: 30 – 12: 20

מיקום התרגול: מאייר 351 (במצב התגוננות 2), בשגרה: פישבך 506

(יתכנו שינויים, נא להתעדכן)

שעות קבלה: יום ג׳ 14: 30 – 15: 30

שעות קבלה נוספות באמצעות פגישת *Zoom* ינתנו לפי דרישה.

משרד: אמרסון, קומה 7

<u>מרכיבי הציון:</u> <u>עבודות בית:</u>

5 תרגילי בית, כל התרגילים הם רשות.

5% מגן על כל גיליון (באופן בלתי תלוי בשאר). **רק במידה שהמבחן בציון עובר.**

<u>בחינה סופית:</u>

משקל: 75% - 100%

מועד א׳: 09.02.25

מועד ב׳: 10.03.25

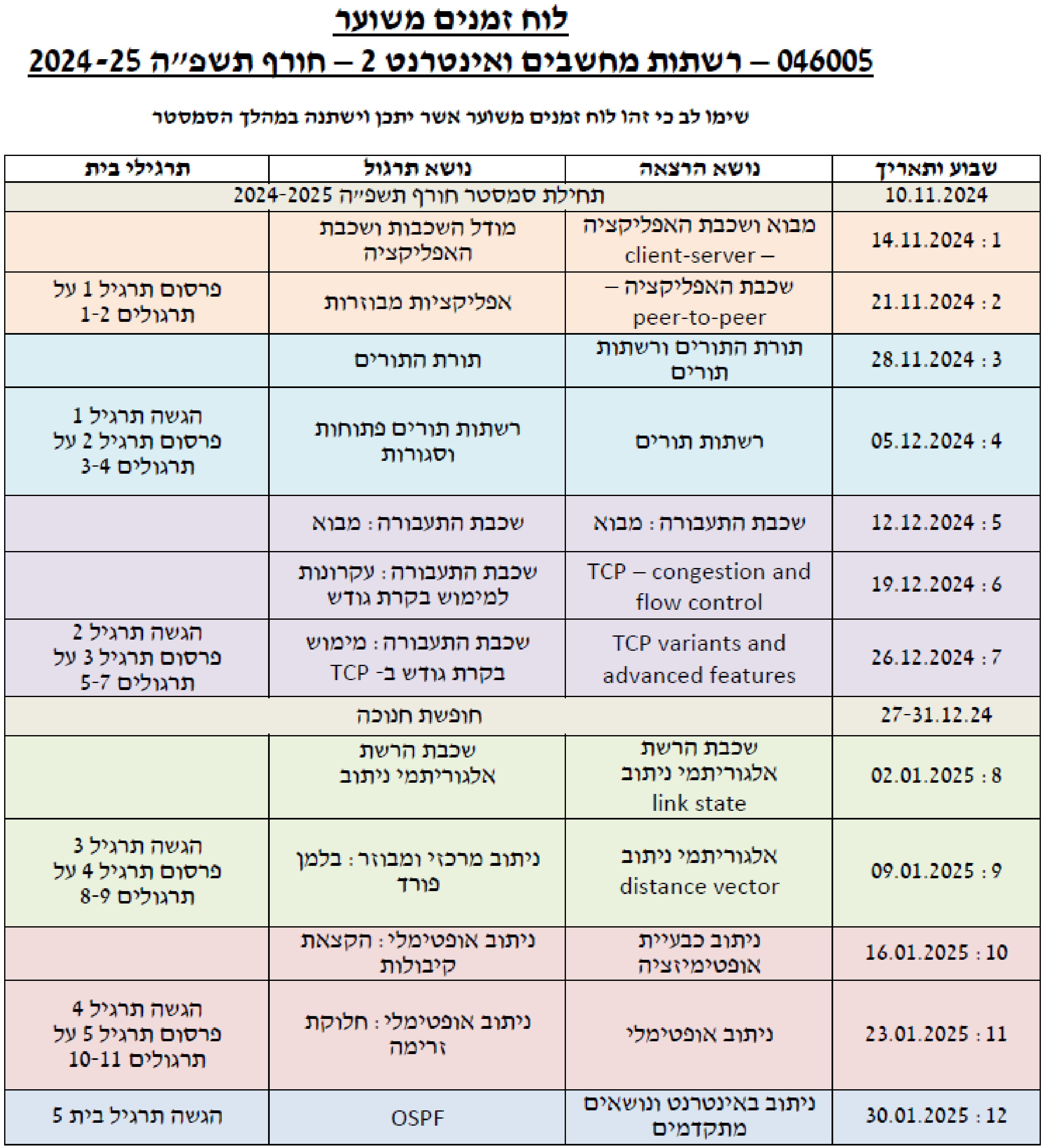
אתר הקורס ב-moodle

- ❐ תוכן מתווסף במהלך הסמסטר
  - ❖ הודעות תימסרנה דרך רשימת תפוצת Moodle בדואר אלקטרוני
    - באחריות הסטודנטים לעקוב אחר ההודעות בדואר האלקטרוני ובאתר הקורס.
  - ❖ הקלטות, בחינות ותוכן מסמסטרים קודמים לנוחותכם בלבד – אין "אחריות"
  - ❖ משוב סטודנטים יכול לעזור מאד!

Course structure: Top Down

- Networks 1
  - Layer 2 – Data link and MAC
  - Layer 3 – Network
  - Layer 4 – Transport
  - (Layer 5 – Application)

- Networks 2
  - Layer 5 – Application
  - Layer 4 – Transport
  - Layer 3 – Network
  - Top down approach!

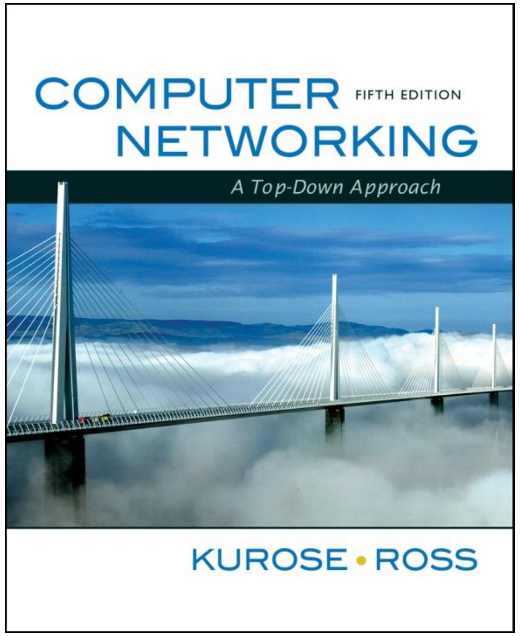


לוח זמנים משוער

046005 – רשתות מחשבים ואינטרנט 2 – חורף תשפ״ה 2024-25

שימו לב כי זהו לוח זמנים משוער אשר יתכן וישתנה במהלך הסמסטר

שבוע ותאריך	נושא הרצאה	נושא תרגול	תרגילי בית
10.11.2024	תחילת סמסטר חורף תשפ״ה 2024-2025		
1 : 14.11.2024	מבוא ושכבת האפליקציה – client-server	מודל השכבות ושכבת האפליקציה	
2 : 21.11.2024	שכבת האפליקציה – peer-to-peer	אפליקציות מבוזרות	פרסום תרגיל 1 על תרגולים 1-2
3 : 28.11.2024	תורת התורים ורשתות תורים	תורת התורים	
4 : 05.12.2024	רשתות תורים	רשתות תורים פתוחות וסגורות	הגשה תרגיל 1 פרסום תרגיל 2 על תרגולים 3-4
5 : 12.12.2024	שכבת התעבורה : מבוא	שכבת התעבורה : מבוא	
6 : 19.12.2024	TCP – congestion and flow control	שכבת התעבורה : עקרונות למימוש בקרת גודש	
7 : 26.12.2024	TCP variants and advanced features	שכבת התעבורה : מימוש בקרת גודש ב- TCP	הגשה תרגיל 2 פרסום תרגיל 3 על תרגולים 5-7
27-31.12.24	חופשת חנוכה		
8 : 02.01.2025	שכבת הרשת אלגוריתמי ניתוב link state	שכבת הרשת אלגוריתמי ניתוב	
9 : 09.01.2025	אלגוריתמי ניתוב distance vector	ניתוב מרכזי ומבוזר : בלמן פורד	הגשה תרגיל 3 פרסום תרגיל 4 על תרגולים 8-9
10 : 16.01.2025	ניתוב כבעיית אופטימיזציה	ניתוב אופטימלי : הקצאת קיבולות	
11 : 23.01.2025	ניתוב אופטימלי	ניתוב אופטימלי : חלוקת זרימה	הגשה תרגיל 4 פרסום תרגיל 5 על תרגולים 10-11
12 : 30.01.2025	ניתוב באינטרנט ונושאים מתקדמים	OSPF	הגשה תרגיל בית 5



A note on the use of these ppt slides:

We’re making these slides freely available to all (faculty, students, readers). They’re in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

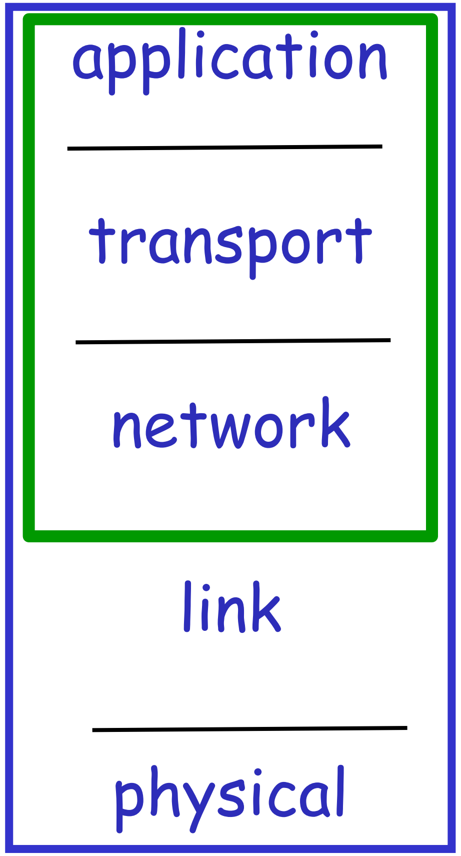
- ❑ If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we’d like people to use our book!)
- ❑ If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach ,
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

Review: Internet protocol stack

5. Application: supporting network applications
   - FTP, SMTP, HTTP
4. Transport: process-process data transfer
   - TCP, UDP
3. Network: routing of datagrams from source to destination
   - IP, routing protocols
2. Link: data transfer between neighboring network elements
   - PPP, Ethernet
1. Physical: bits “on the wire”



application

transport

network

link

physical

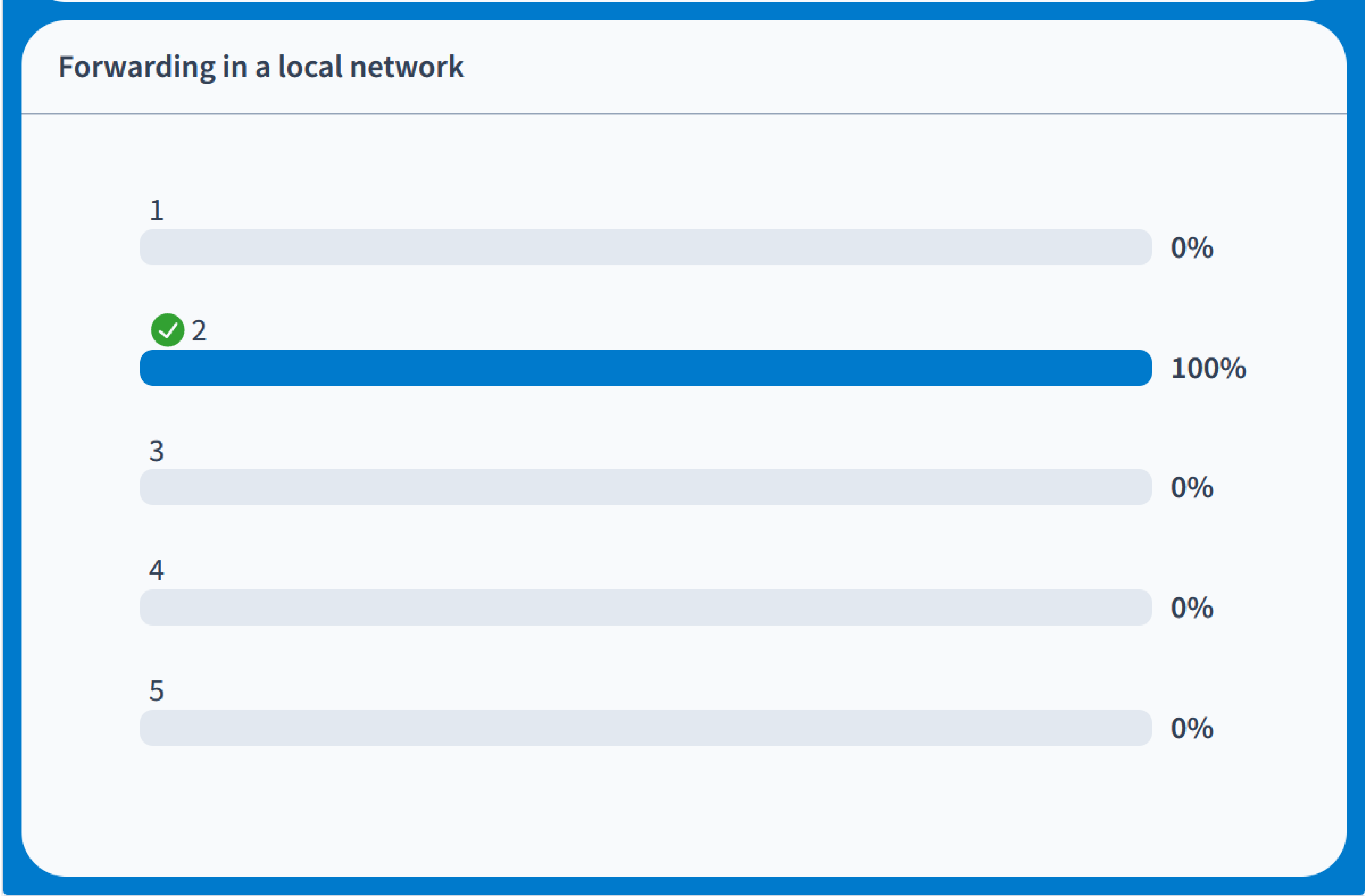
Quiz: in which layer we can find the service?

1. Physical
2. Link
3. Network
4. Transport
5. Application

Forwarding in a local network

1. Physical
2. Link
3. Network
4. Transport
5. Application

?



Forwarding in a local network

1 0%

✓ 2 100%

3 0%

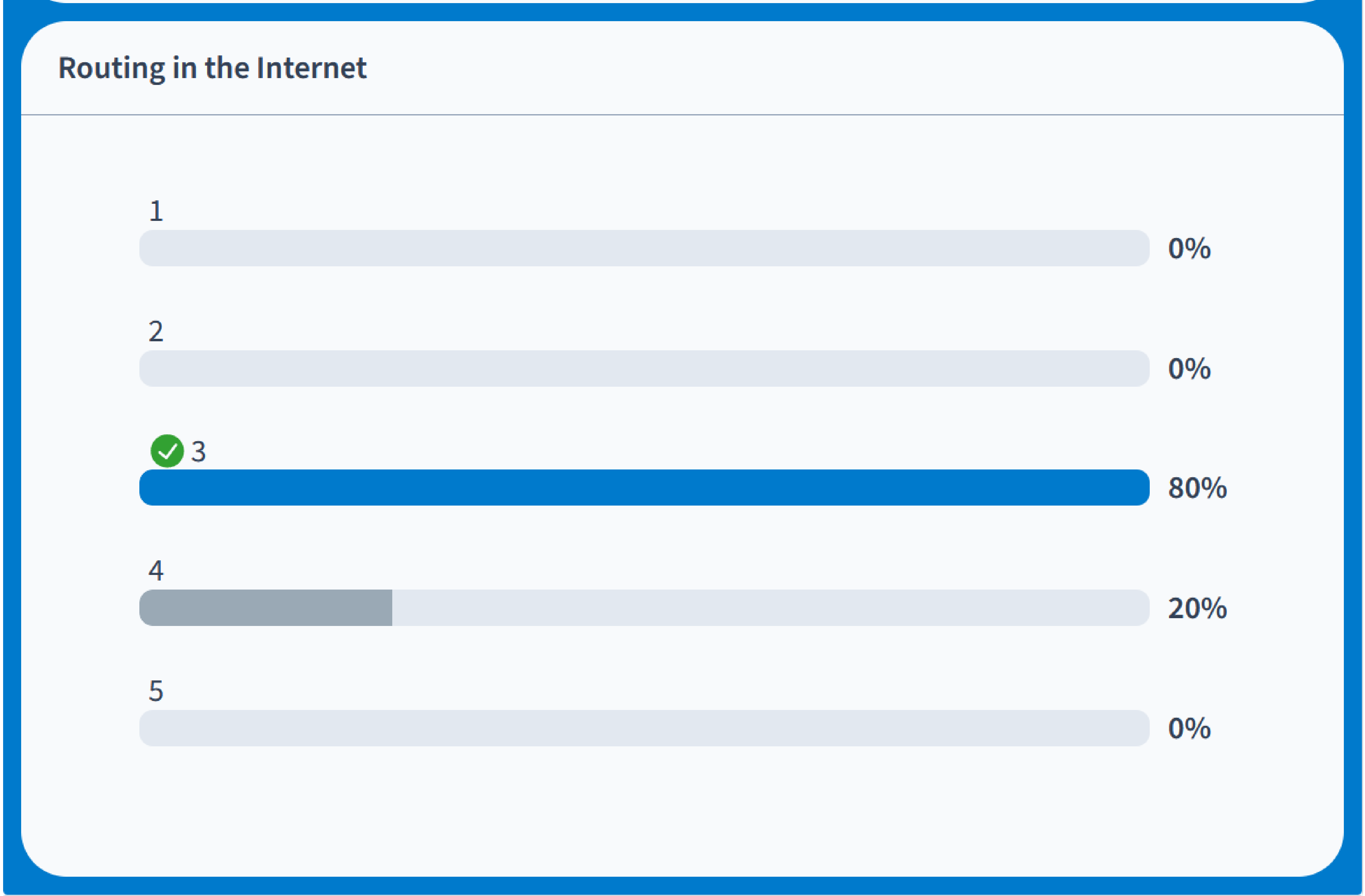
4 0%

5 0%

Routing in the Internet

1. Physical
2. Link
3. Network
4. Transport
5. Application

?



Routing in the Internet

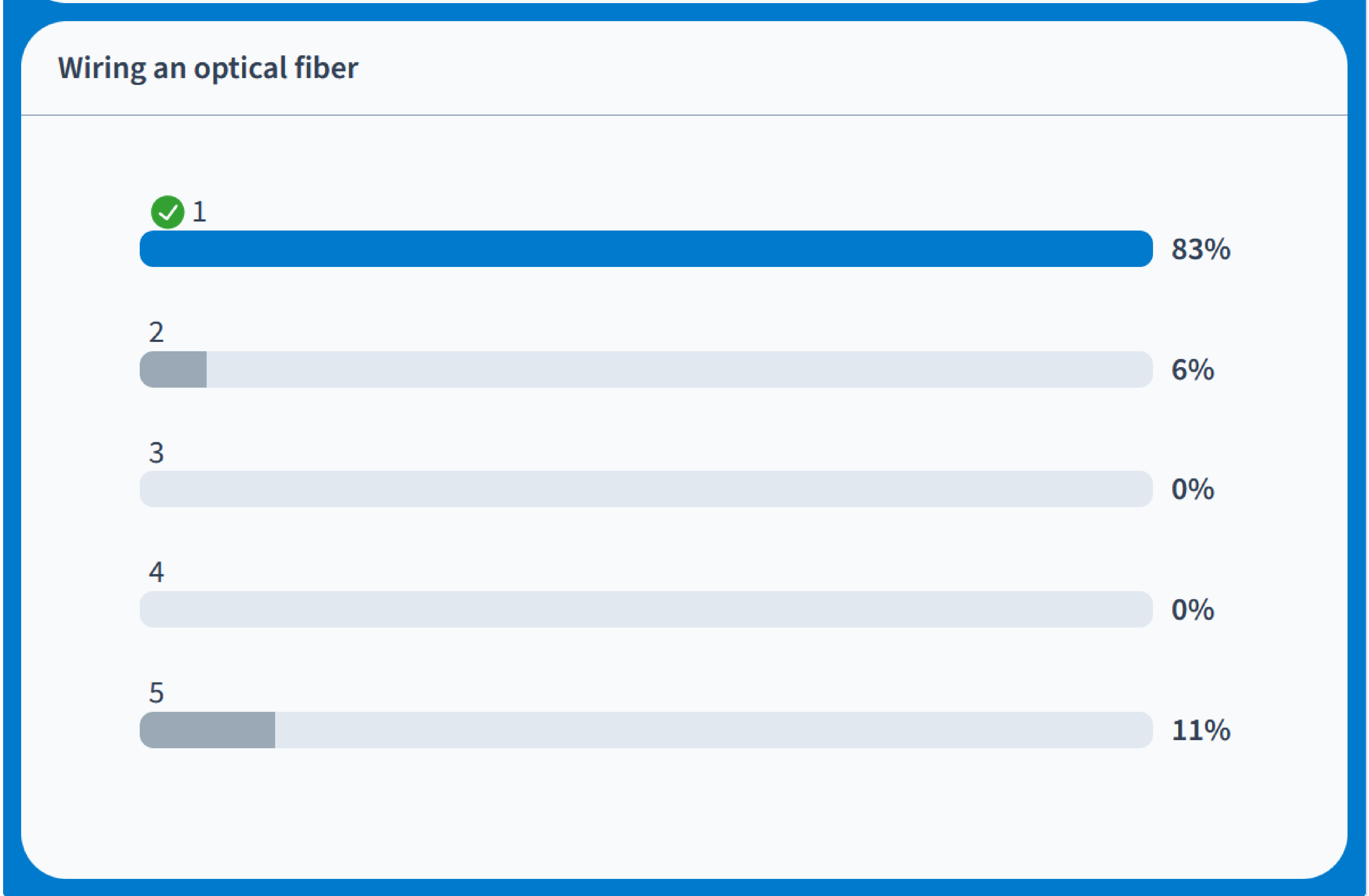
1	0%
2	0%
✓ 3	80%
4	20%
5	0%

Wiring an optical fiber

1. Physical
2. Link
3. Network
4. Transport
5. Application

?

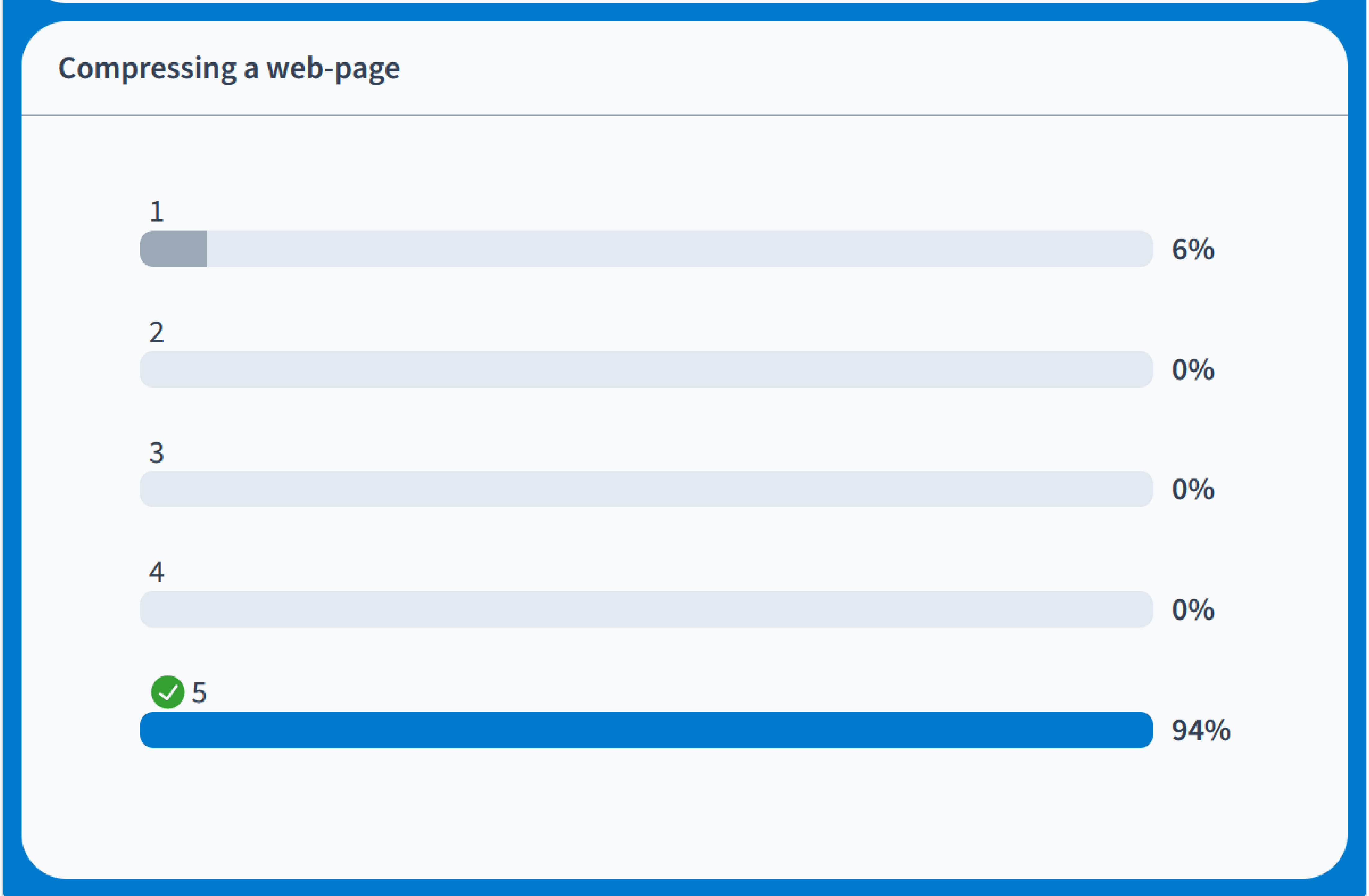
Wiring an optical fiber



Compressing a web page

1. Physical
2. Link
3. Network
4. Transport
5. Application

?



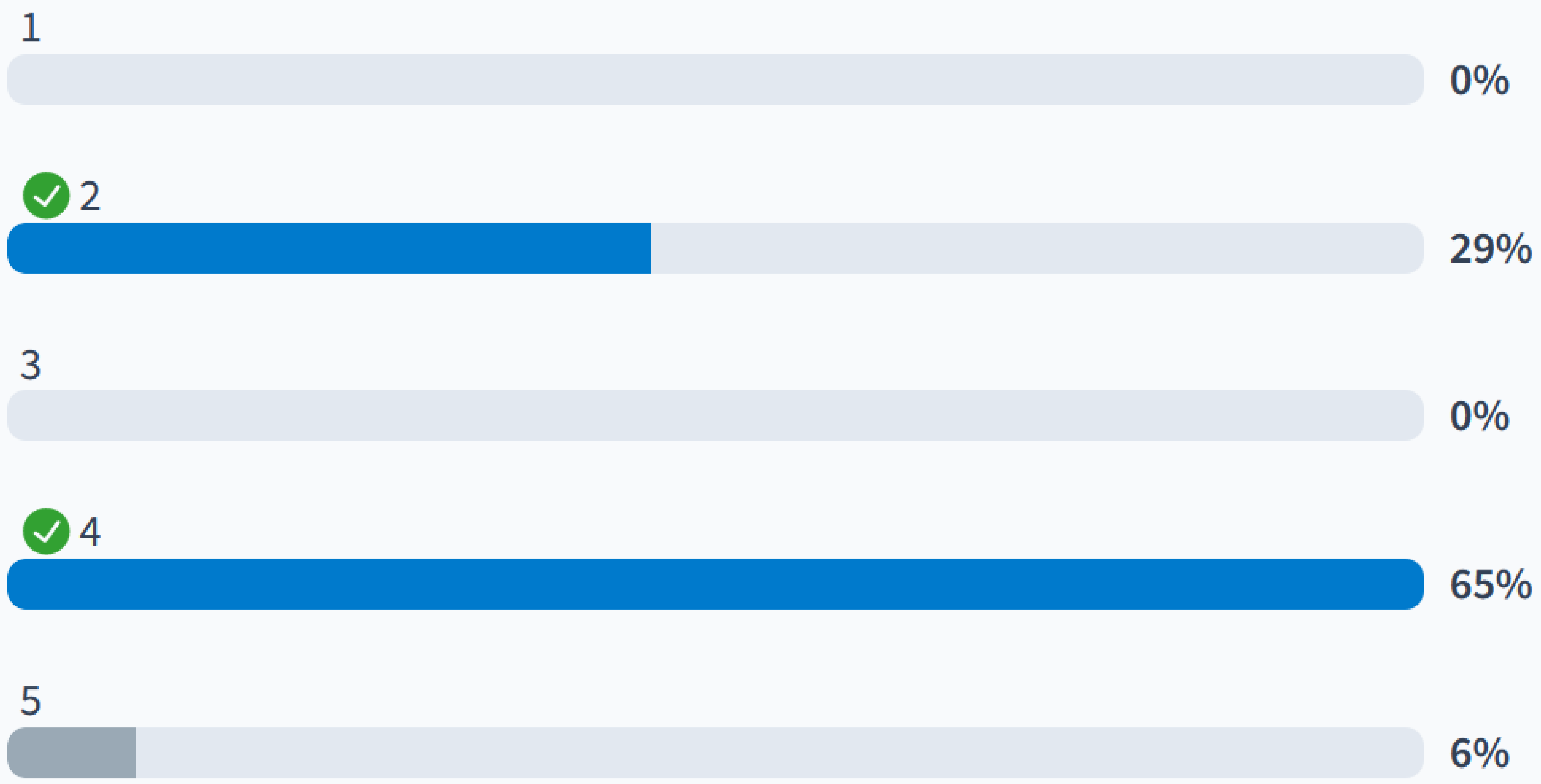
Compressing a web-page

Managing point-to-point flow

1. Physical
2. Link
3. Network
4. Transport
5. Application

?

Managing a point-to-point flow



Some network applications

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips
- social networks

- voice over IP
- real-time video conferencing (Zoom)
- grid computing
  - Combination of computer resources from multiple administrative domains to reach a common goal
- cloud computing
  - Delivery of computing as a service rather than a product

✓ Applications are programs that
  - ✓ Run on (different) end systems
  - ✓ Communicate over network

✓ No need to write software for network-core devices

✓ Use operating system services at edges

Application architectures

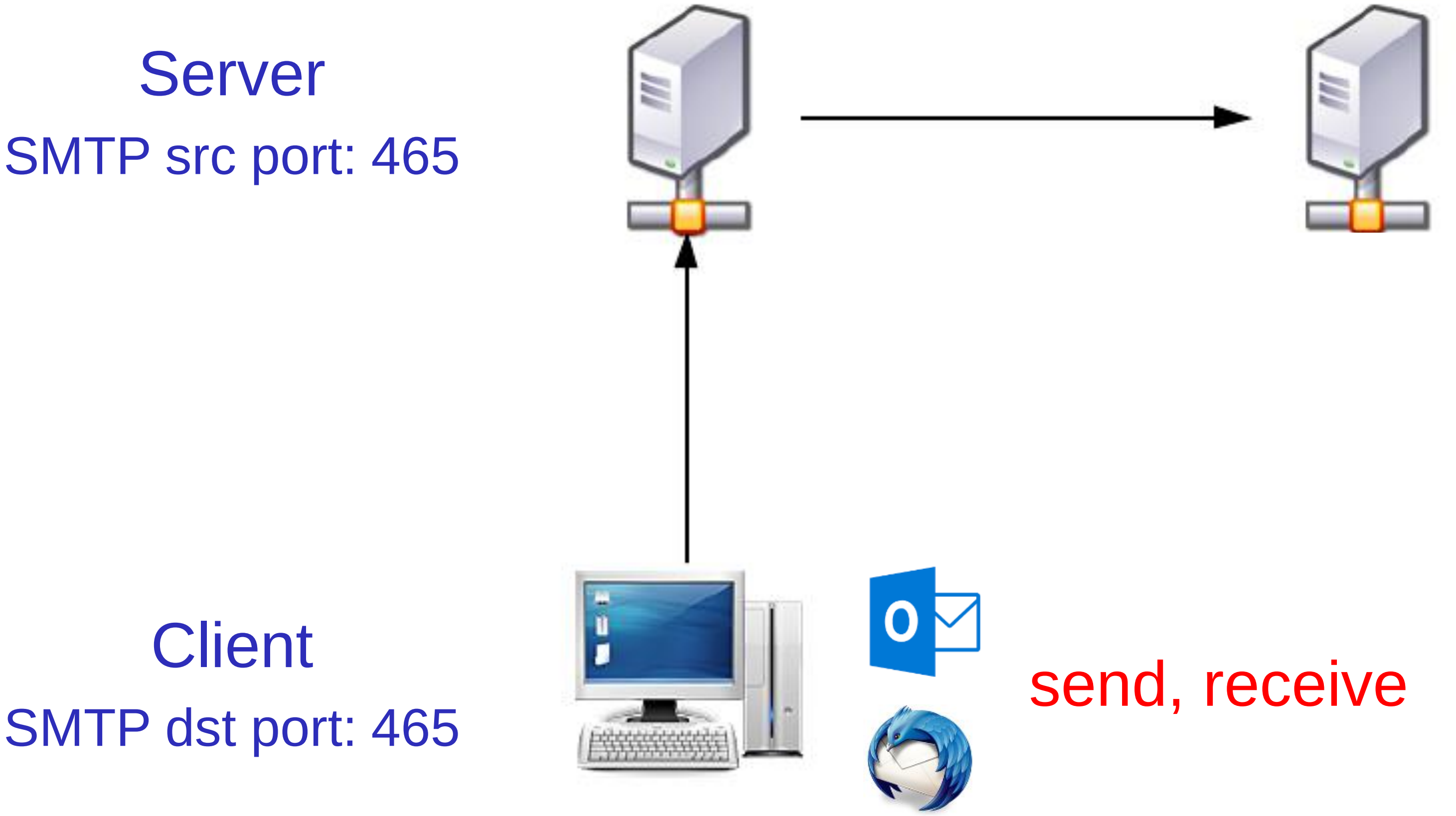
- Client-server ← Today
  - Including data centers / cloud computing
- Peer-to-peer (P2P)
- Hybrid of client-server and P2P

Client-Server Protocols

	Connection (layer 4)	Information (layer 5)
Client	Initiate	Request
Server	Listen, Accept	Respond

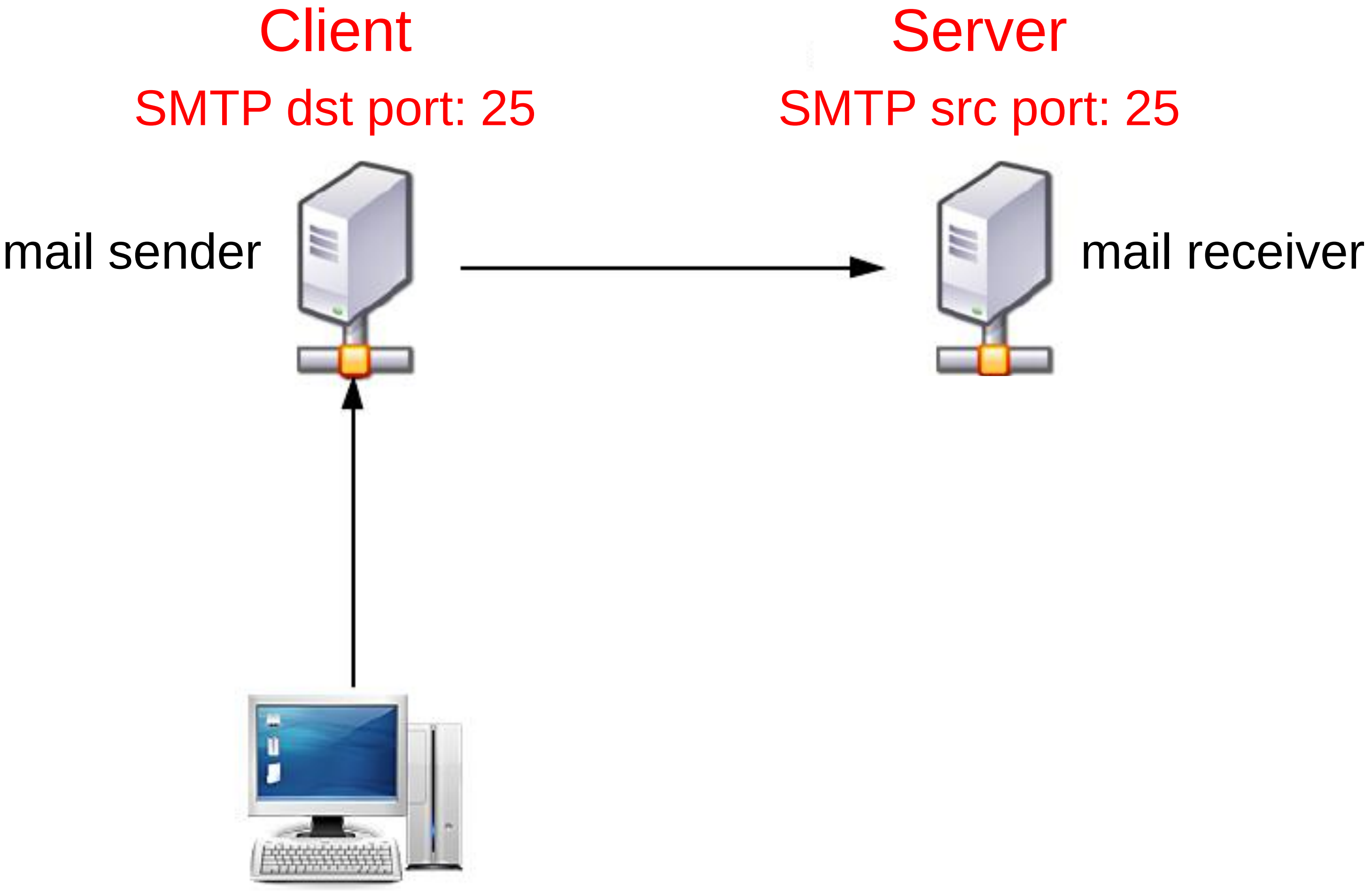
Client or server?

Example: email (SMTP)



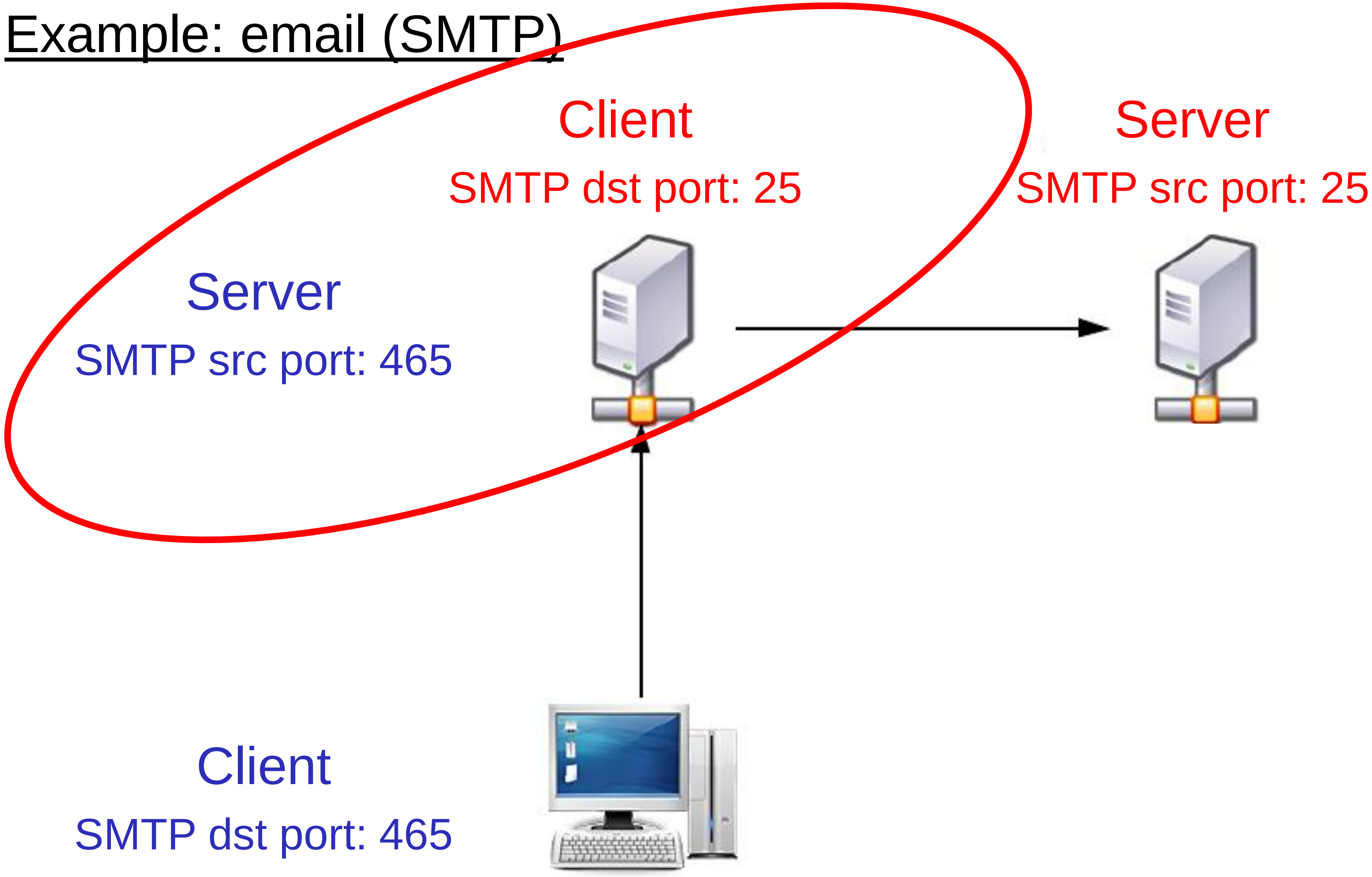
Client or server: relative, not absolute

Example: email (SMTP)



Client or server: relative, not absolute

Example: email (SMTP)



Client

SMTP dst port: 25

Server

SMTP src port: 25

Server

SMTP src port: 465

Client

SMTP dst port: 465

The most useful client-server protocol?

HTTP

HTTPS = HTTP + TLS (Transport Layer Security)

web, video streaming (youtube), social networking (facebook), in both browser and mobile apps

Browser traffic

?%

App traffic

?%

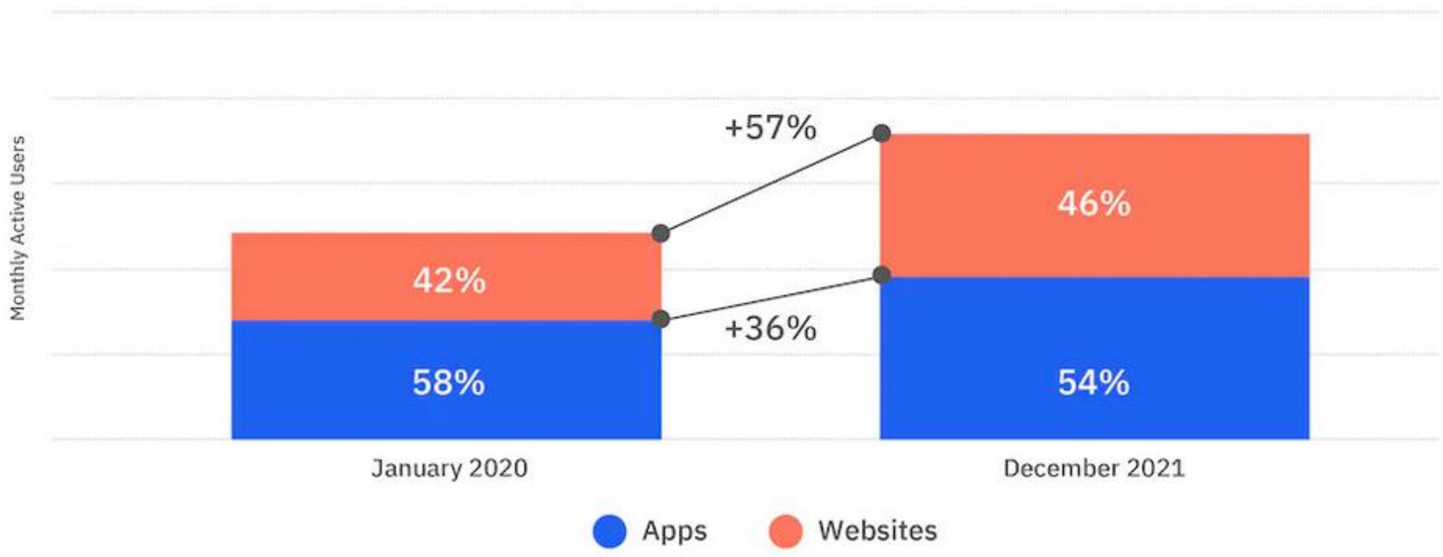
The most useful client-server protocol?

HTTP

HTTPS = HTTP + TLS (Transport Layer Security)

web, video streaming (youtube), social networking (facebook), in both browser and mobile apps

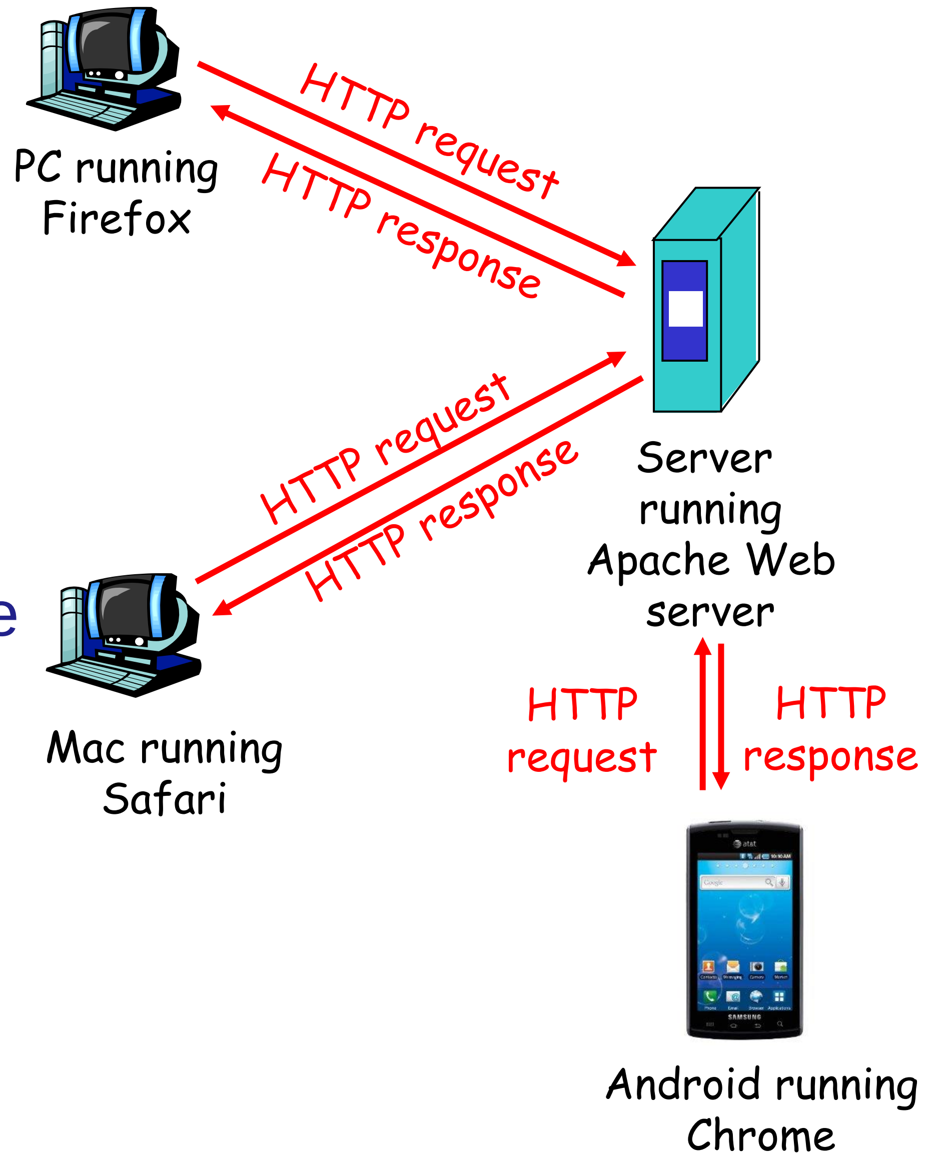
Percentage growth in monthly active users of websites and mobile apps, comparing January 2020 to December 2021.



Amplitude Labs | 2022 App vs. Website Trend Report

HTTP & Web

- ❒ HTTP: hypertext transfer protocol
  - ❖ Web's application layer protocol
- ❒ Client/Server model
  - ❖ Client: browser/app that requests, receives, "displays" web objects
  - ❖ Server: web server sends objects in response to requests
- ❒ HTML: hypertext markup language
  - ❖ Base file describes content as tree of objects
    - Typically, >100 of objects
  - ❖ Objects are text, HTML files, script code, java-script, audio, images, …
  - ❖ Each object addressable by URL
    - **www.host.name/path/to/object/such/as/a.gif**



HTTP over TCP

- Request-response on top of TCP (application-layer protocol messages)
  - Client initiates TCP connection to server, port 80, server accepts
  - HTTP messages exchanged between client and server
  - TCP connection closed
- “Stateless”
  - Server maintains no information about past client requests 😉
  - Protocols that maintain “state” are complex!
    - Past history (state) must be maintained
    - If server/client crashes, their views of “state” may be inconsistent, must be reconciled
  - HTTP is stateless does not imply application is stateless

HTTP Messages (from RFC)

Request

- Request line: method + URL + version
  - GET/HEAD/POST/PUT/DELETE/CONNECT http://request/URL/ HTTP/version CRLF
- Headers: optional + custom
  - Header: value CRLF
  - Connection: keep-alive/close
  - Transfer-Encoding: chunked
  - Host: www.tx.ac.il (why needed?)
- Empty line: CRLF
- Message body (optional)
  - Txt, XML, json, …

Response

- Status: code + reason
  - HTTP/version CODE reason CRLF
  - 1xx info, 2xx success, 3xx redirect, 4xx client err, 5xx server err
- Headers: optional + custom
  - Date: Tue, 15 Mar 2016 07:10:30 GMT
  - Content-Length: 70 (why needed?)
  - Content-Type: text/html
- Empty line: CRLF
- Body
  - HTML

A brief history of HTTP

- 1996: HTTP 1.0
  - Non-persistent
- 1999: HTTP 1.1
  - Persistent
  - Pipelining
- 2015: HTTP 2
  - Streams
- Future: HTTP 3 (over QUIC transport protocol)

Initial HTTP (version 1.0)

- Non-persistent HTTP
  - Client initiates TCP connection to server, port 80, server accepts
  - One HTTP request-response
  - TCP connection closed by server

repeated per requested object

Non-persistent HTTP Example

Suppose user enters URL www.technion.ac.il/path/page.html

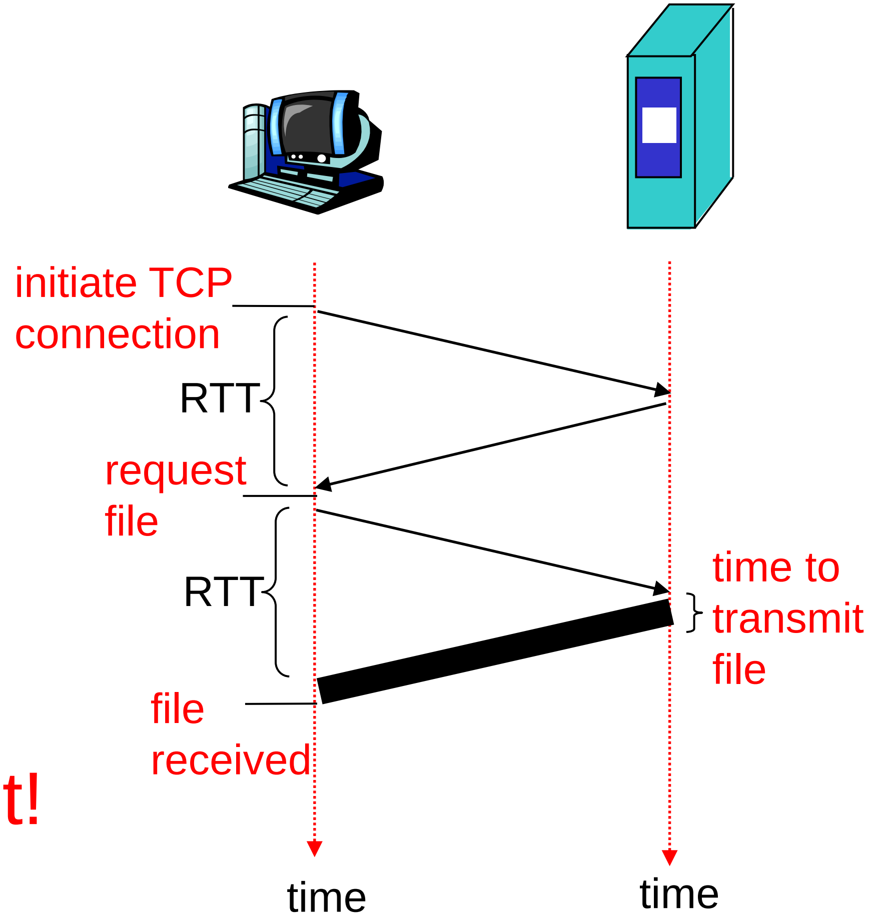
browser **server**

1. TCP connect to www.technion.ac.il:80
2. listen on port 80, accept
3. establish TCP connection, send request (**GET** /path/page.html)
4. respond with main HTML page (**possibly multiple segments**)
5. close TCP connection
6. receive HTML, parse, display, find referenced objects
7. repeat for each object

Time

Non-Persistent HTTP: Response time

- Definition of RTT: round trip time
  - Time for a small packet to travel from client to server and back.
- Response time:
  - One RTT to initiate TCP connection
  - One RTT for HTTP request and first few bytes of HTTP response to return
  - File transmission time
- Total = 2RTT+transmit time – per object!



Disadvantages of non-persistent

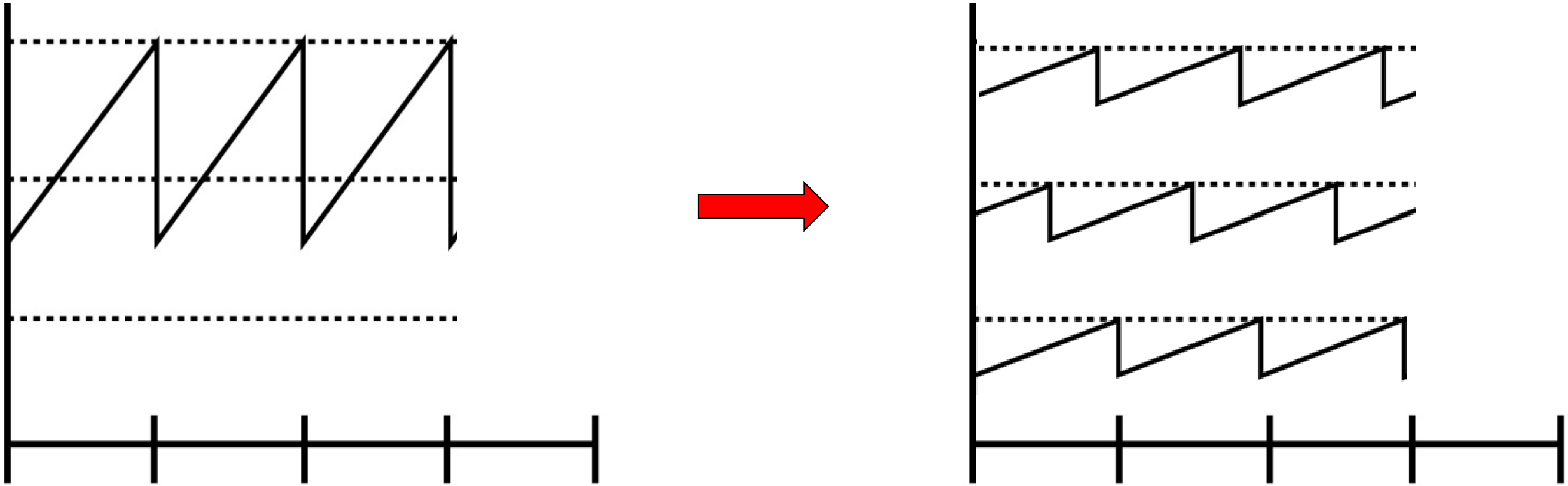
1. Long response time
2. Clients mitigation: use parallel connections

   Why is this bad?

Non-persistent → parallel TCP connections

2a: Overhead: manage multiple buffers, protocol states (OS resources)

2b: Skewed congestion control



single connection

3 parallel connections

3/N bandwidth instead of 1/N

Disadvantages of non-persistent

1. Long response time

2a. Parallel connections: resource overhead

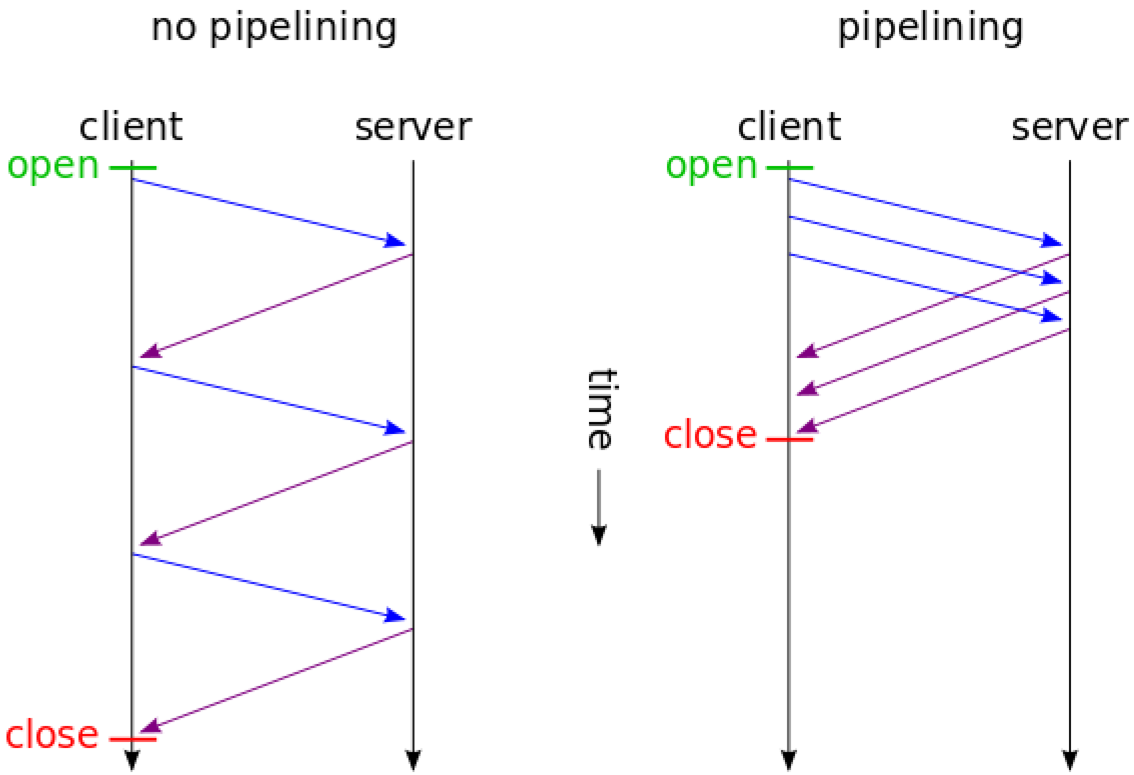
2b. Parallel connections: skewed congestion control

HTTP Connection (revisit)

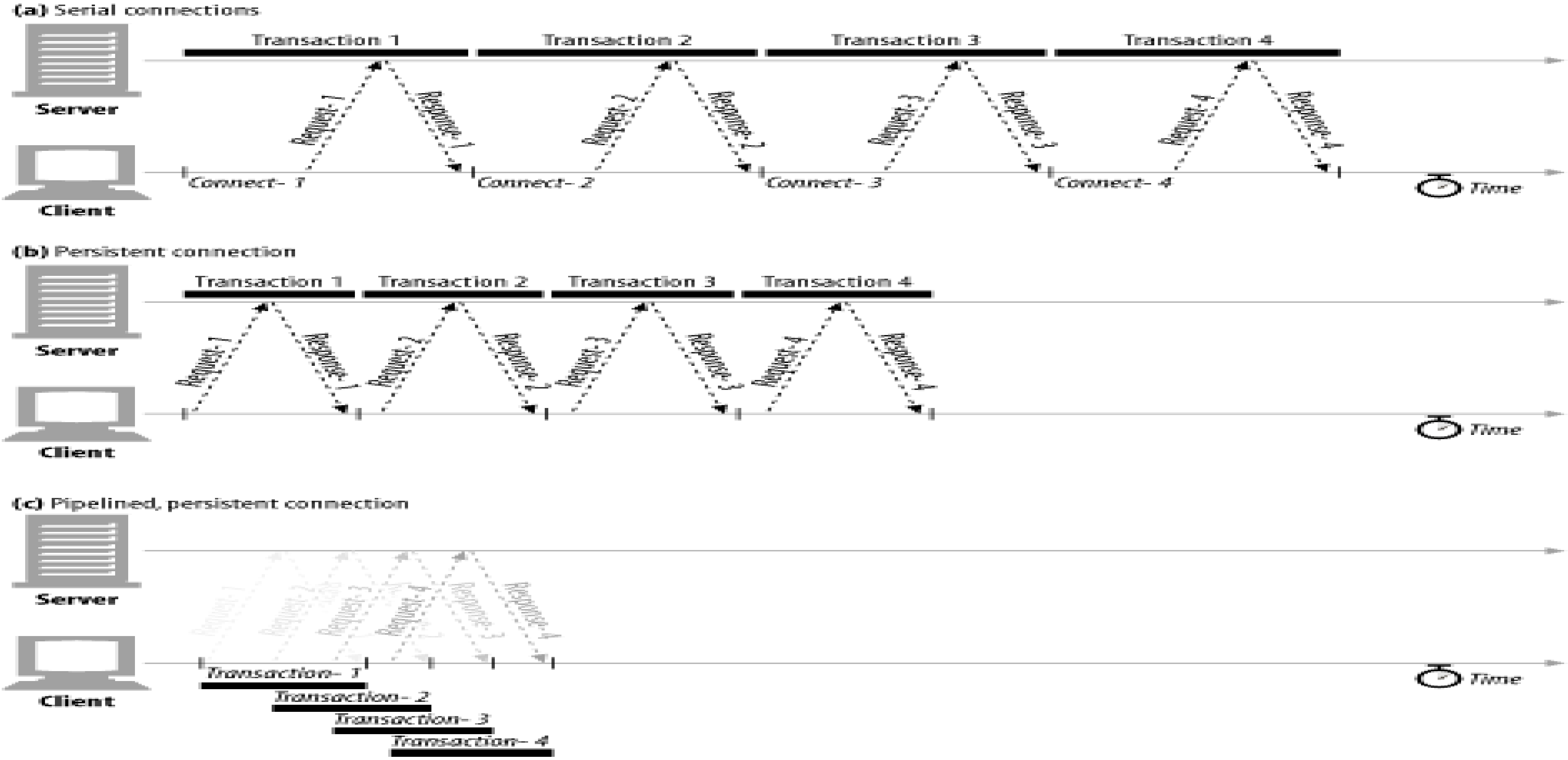
- Request-response on top of TCP (application-layer protocol messages)
  - Client initiates TCP connection to server, port 80, server accepts
  - HTTP messages exchanged between client and server
  - TCP connection closed
- Non-persistent HTTP (version 1.0 default)
  - Close connection after one request (single object fetched)
  - Simple, no idle open connections, more overhead (2 RTTs) per object
  - Typically used with parallel connections

Improved HTTP (version 1.1)

- Persistent connection by default
  - Add “connection: close” to notify non-persistent connection
- Pipelined connections
  - Send multiple requests on connection
    - Responses in FIFO order only
  - Need to handle failures



Pipelined HTTP (https://www.safaribooksonline.com/library/view/http-the-definitive/1565925092/ch04s06.html)



Pipelined HTTP Advantages

1. Overlap request and response propagation times
2. Overlap **request-processing times** and communication time

Static WEB → **Dynamic WEB** **("WEB 2.0")**

Read (static) content object from storage

Generate content object from server data and business logic (user customization)

Issues with Pipelined HTTP

1. Head-of-Line blocking due to FIFO → **Solution:** Client optimization
   - Slow requests block those after them
2. Handling failures
   - What if the connection is closed after pipelining N requests?

Solution (from RFC 2616):

Clients SHOULD NOT pipeline requests using **non-idempotent** methods or **non-idempotent sequences** of methods (see section 9.1.2).

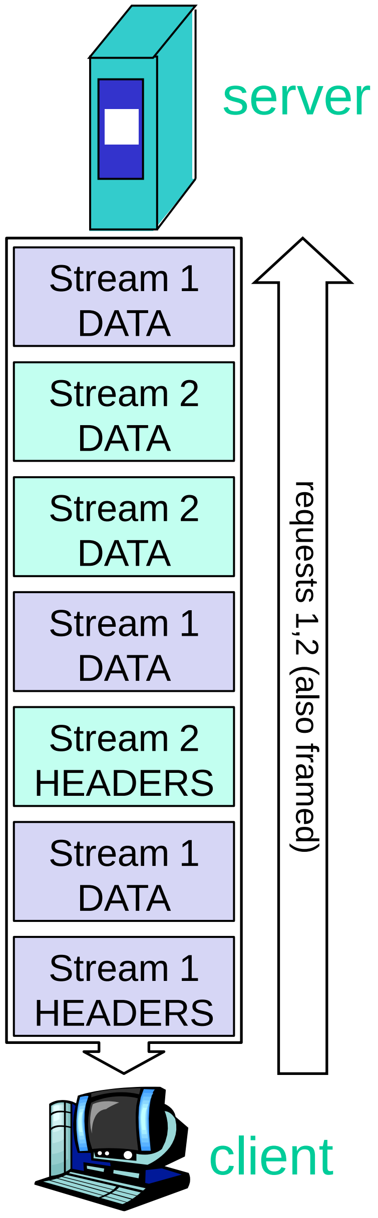
Otherwise, a premature termination of the transport connection could lead to indeterminate results.

HTTP 1.0/1.1 Summary

- Request-response on top of TCP (application-layer protocol messages)
  - Client initiates TCP connection to server, port 80, server accepts
  - HTTP messages exchanged between client and server
  - TCP connection closed
- Non-persistent HTTP (version 1.0 default)
  - Close connection after one request (single object fetched)
  - Simple, no idle open connections, more overhead (2 RTTs) per object
  - Typically used with parallel connections
- Persistent HTTP (version 1.1 default)
  - Multiple request-response messages over same TCP connection
  - Less overhead (1 RTT) per request (from 2nd object) , **longer TCP**, fewer connections
  - Can use a pipeline (queue of requests), can still use parallel connections

HTTP 2.0

- (proposed by Google)
- Goal:
  single connection, pipelining, but no head-of-line blocking
- Idea:
  multiplex HTTP sub-streams on a single TCP connection
  - Send data stream in interleaved frames
    - Per-frame 8B binary header (len, type, flags), per-frame 31b id
  - HEADER frames
    - Establish stream, id, define priority, compressed (diff)
  - DATA frames
    - Pieces of data that can be multiplexed, <16KB
- Better TCP throughput, fewer TCP connections, less data
  - Many small files? No problem.
- More
  - Stream and connection flow control
  - Server push
  - Book: https://hpbn.co/ High Performance Browser Networking / ILYA GRIGORIK



HTTP in Practice

- Persistent, but some limitations
  - Many objects are from other hosts
  - Server may close connections
- We don’t always talk with the server
  - Redirections
  - Gateways / caches
  - CDNs (Content Distribution Networks)
- Complex content
  - Browser processing may delay network
  - Dependencies between objects
  - Active content

HTTP Redirect

Application-layer redirect

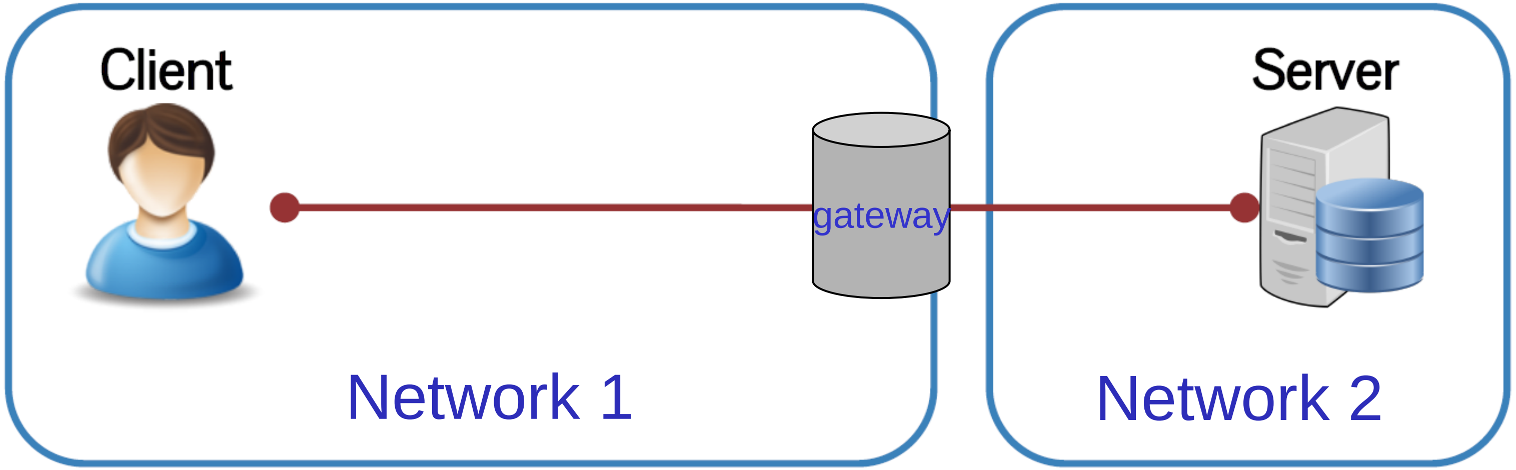
Request

- Request line: method + URL
  - GET/HEAD/POST/PUT/DELETE/CONNECT http://request/URL/ HTTP/version CRLF
- Headers: optional + custom
  - Header: value CRLF
  - Connection: keep-alive/close
  - Transfer-Encoding: chunked
  - Host: www.tx.ac.il
- Empty line: CRLF
- Message body (optional)
  - Txt, XML, json, …

Response

- Status: code + reason
  - HTTP/version CODE reason CRLF
  - 1xx info, 2xx success, **3xx redirect**, 4xx client err, 5xx server err
- Headers: optional + custom
  - Date: Tue, 15 Mar 2016 07:10:30 GMT
  - Content-Length: 70
  - Content-Type: text/html
- Empty line: CRLF
- Body
  - HTML

HTTP in real networks: gateways/caches



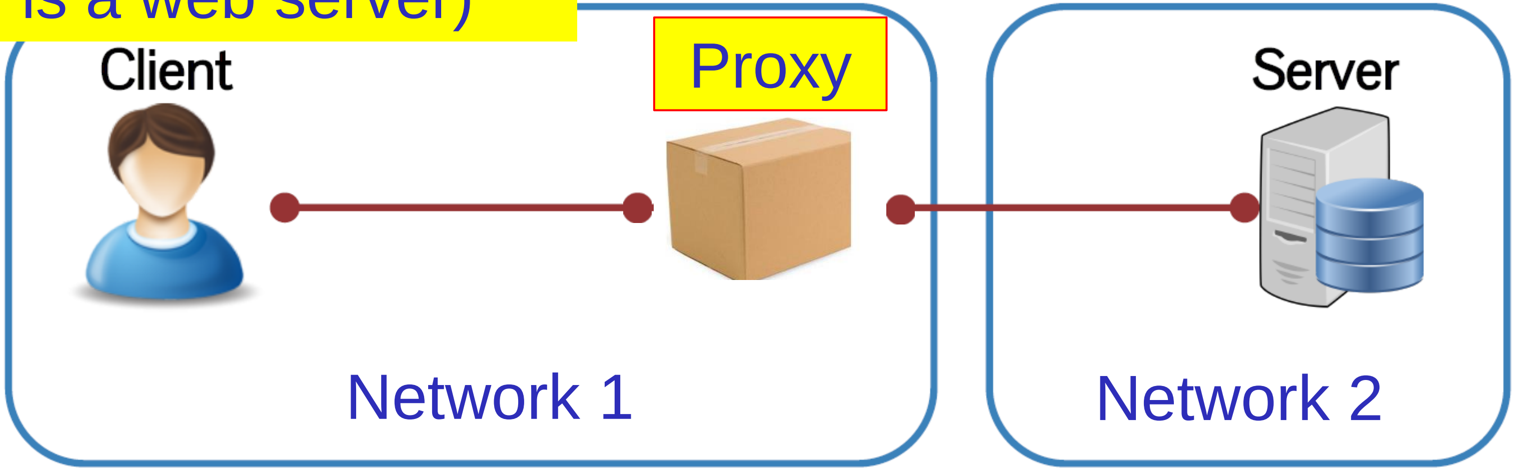
1. ISP network (DSL, Cable)
2. Cellular network
3. Satellite network

The wide Internet

HTTP by Proxy

Transparent to client (thinks Proxy is a web server)

Transparent to web server (thinks Proxy is a client)



0

1) Client sends http requests to Proxy server

2) Proxy forwards requests to web server

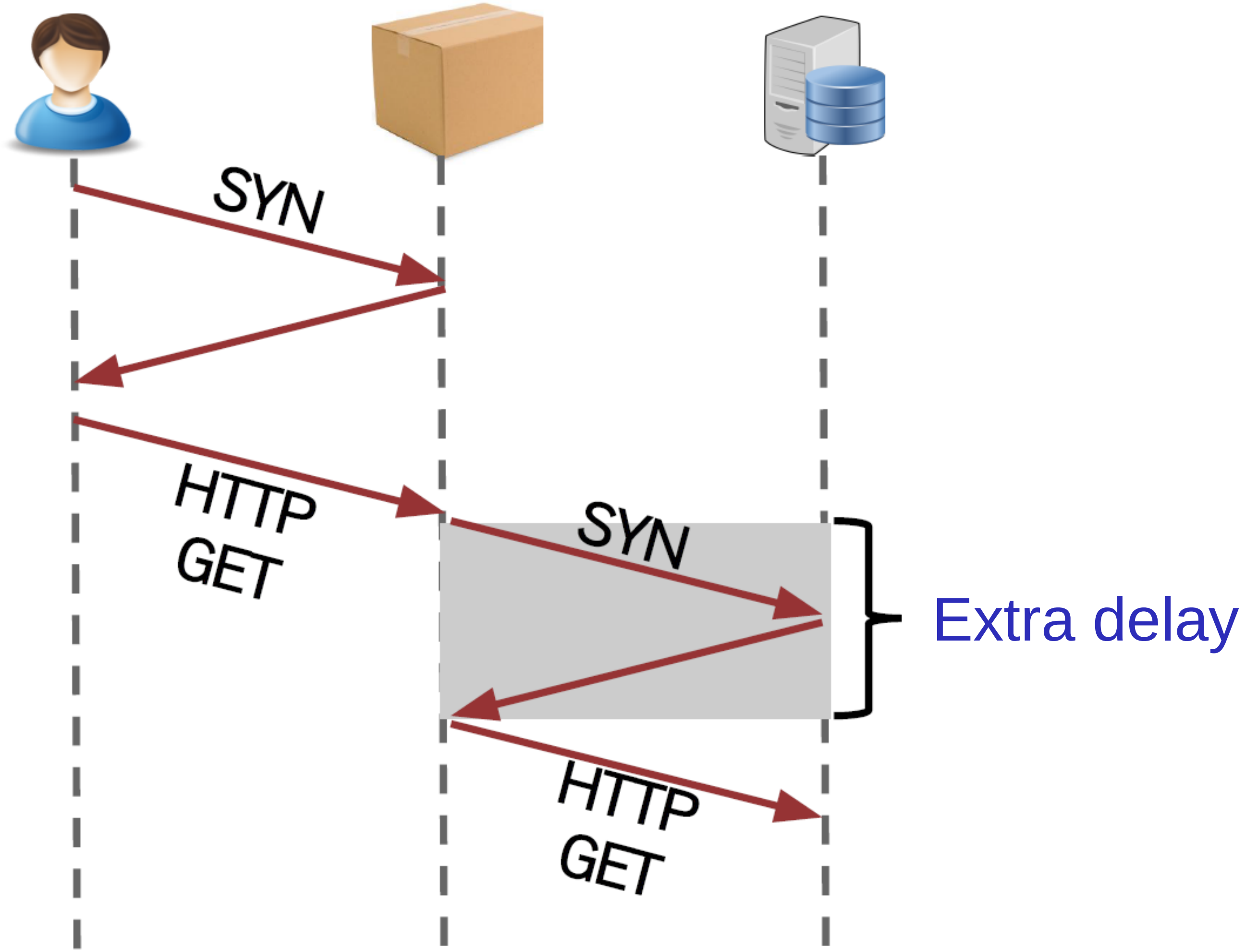
4) Proxy server sends responses to client

3) Web server sends response to Proxy

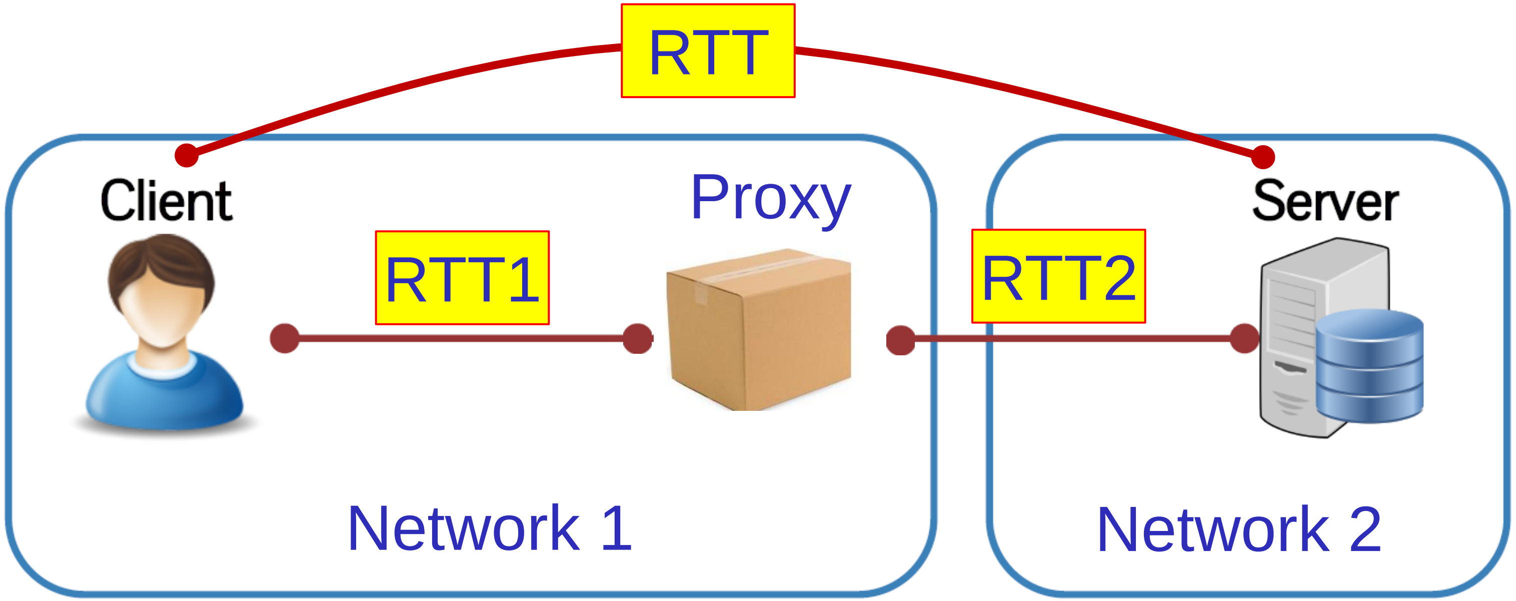
Why Proxy?

- Many reasons
  - Help the client
  - Protect the client
  - “Protect” (censor) the client
  - Save resources

Snapshot of http with Proxy



Proxy performance advantage

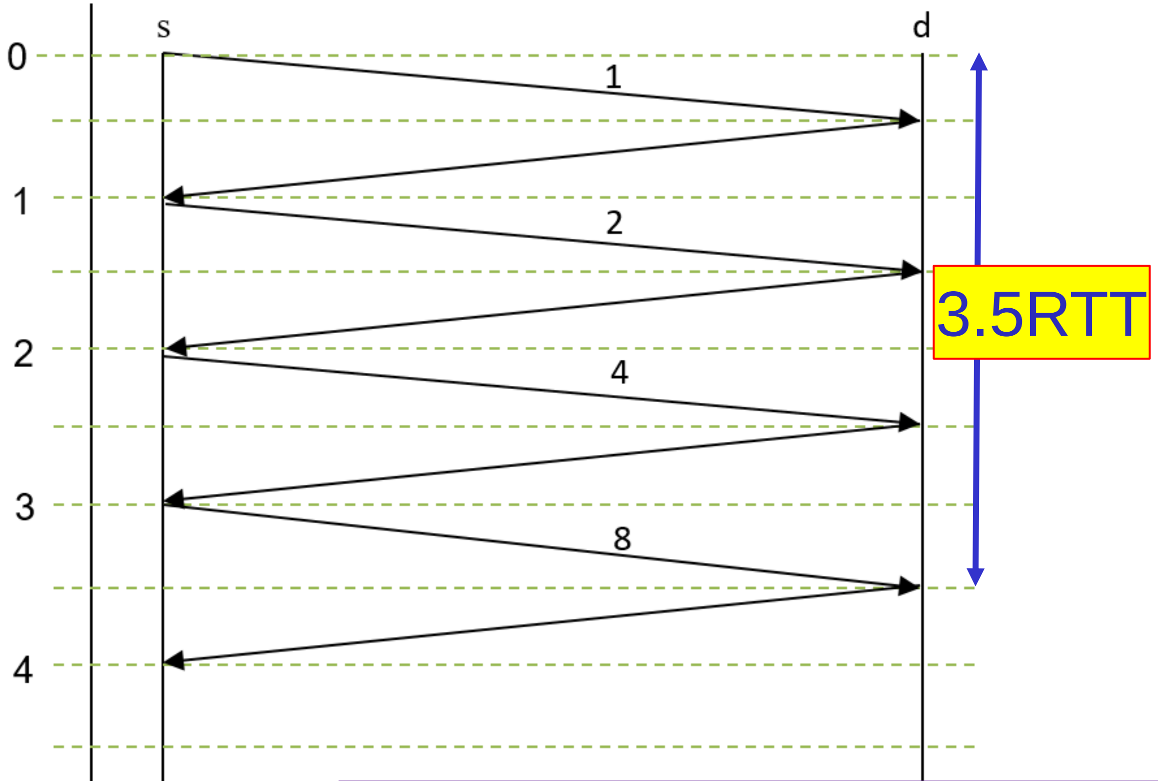


0

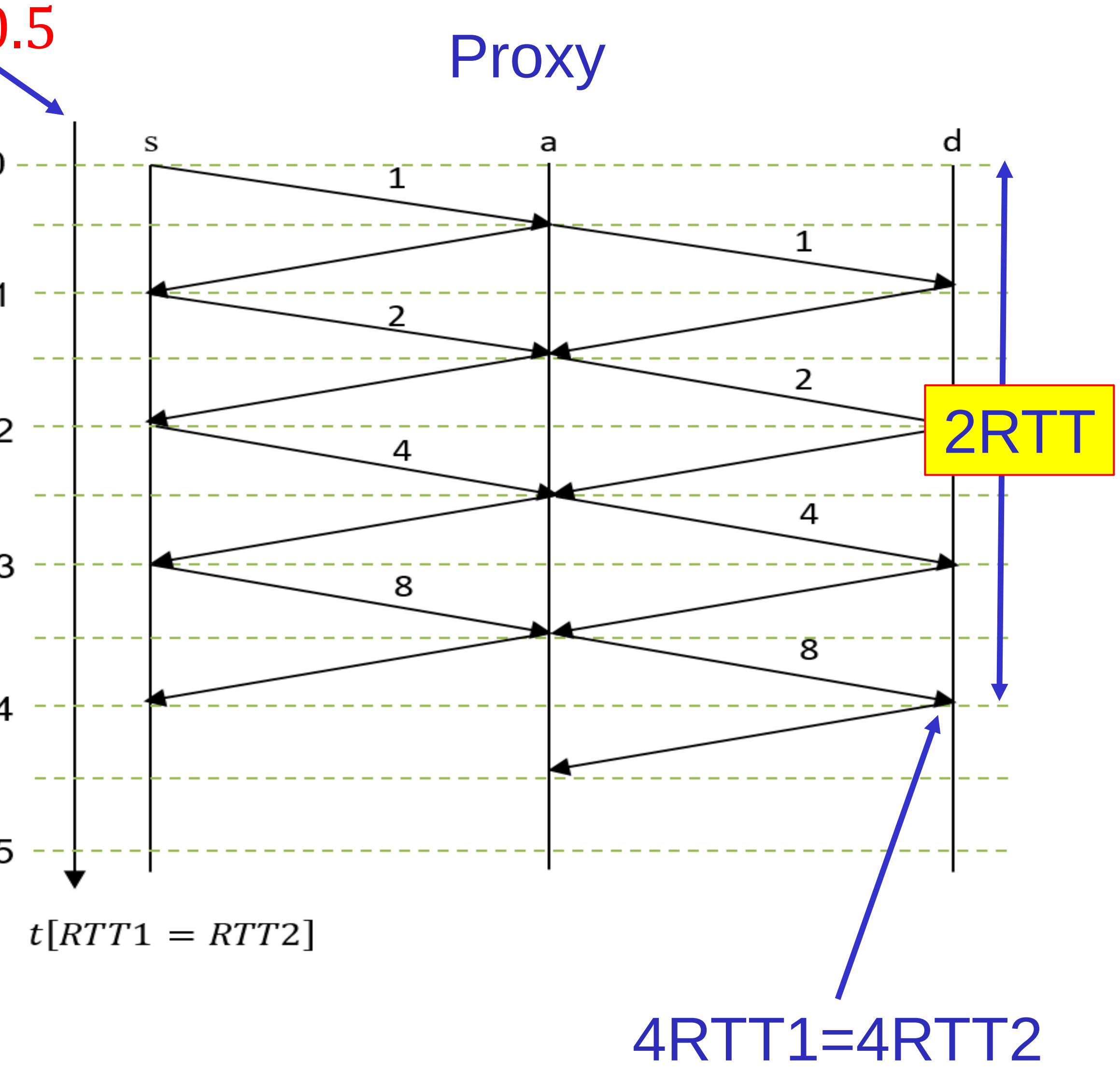
Transport (RTT1) → Transport (RTT2) vs. Transport (RTT1+RTT2) ?

Proxy transport advantage

No Proxy



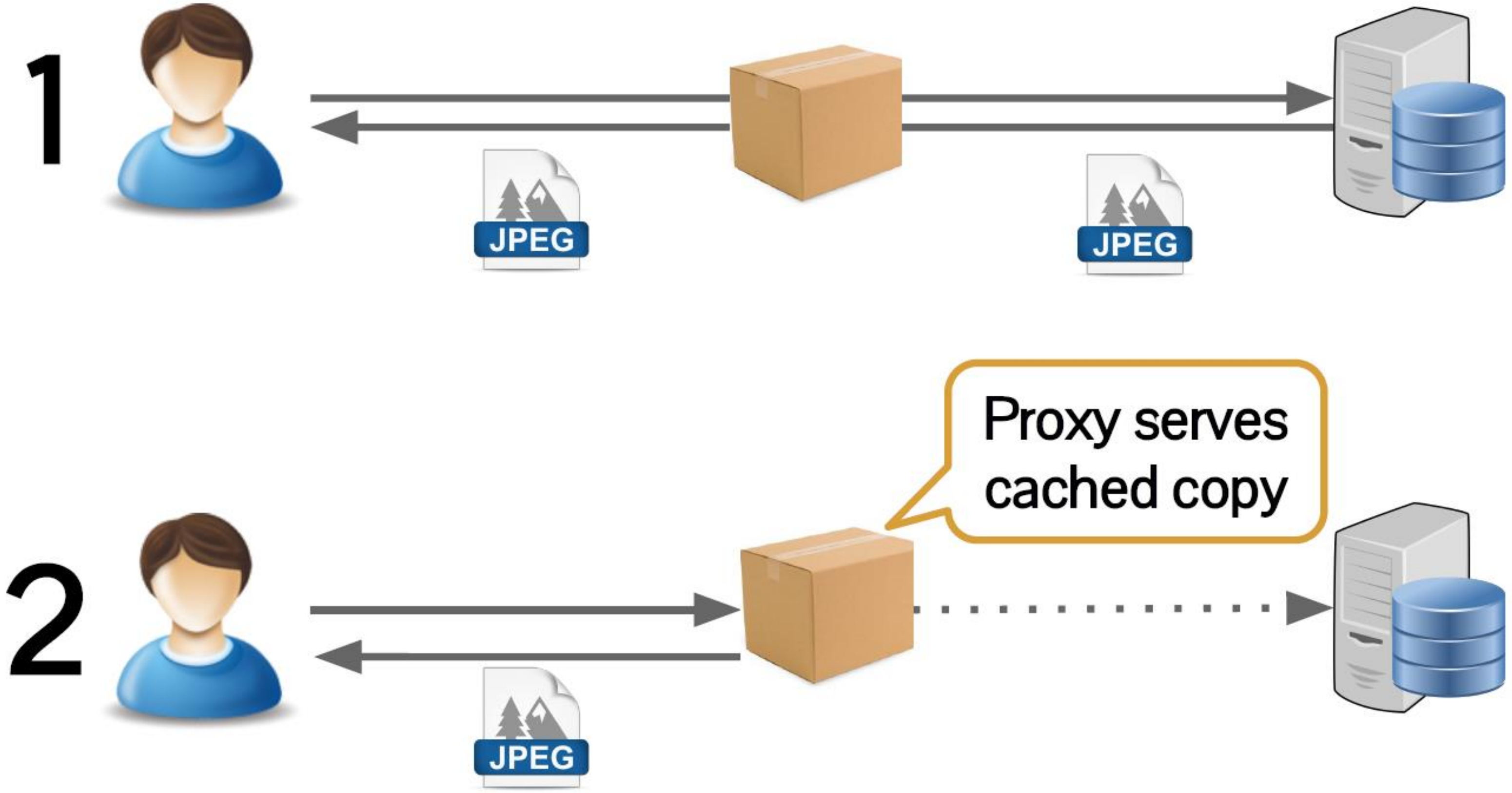
Proxy



Assumptions:

- RTT1=RTT2
- ti negligible (compared to RTT)

HTTP Caching

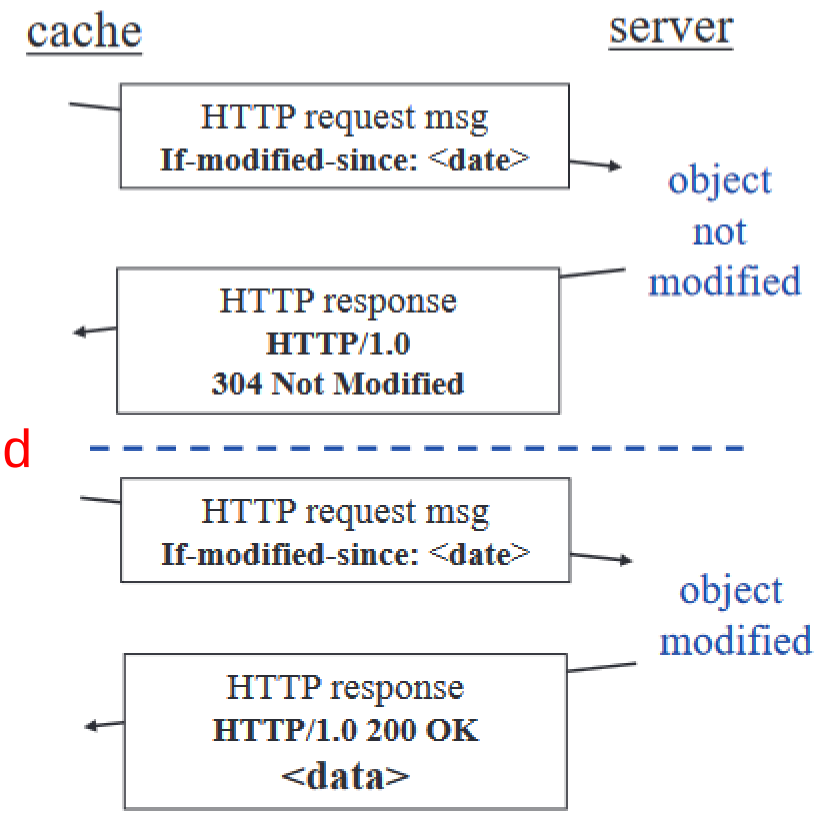


Challenges?

HTTP Caching

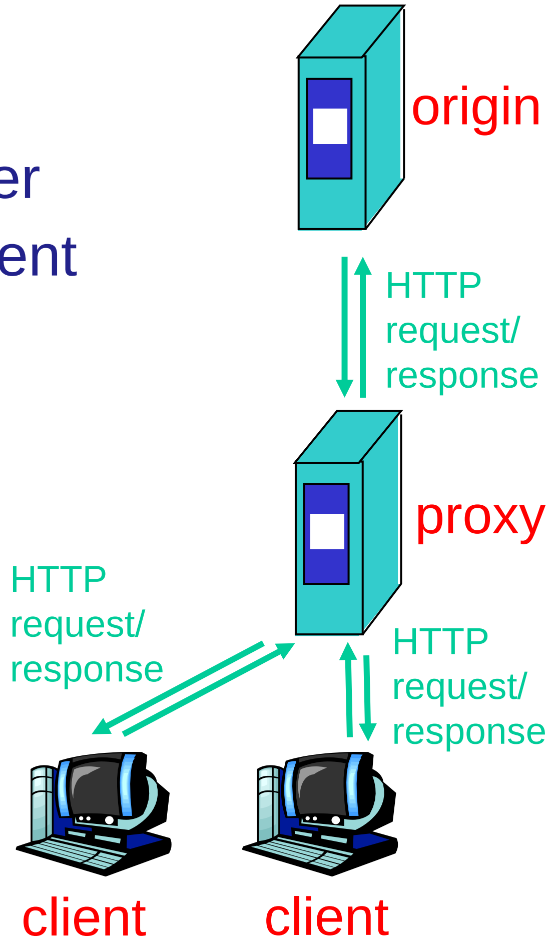
- Staleness
  - Expiration time, last modified time
  - Conditional GET: ask server if stale
    - **Client**: If-Modified-Since: date-time, **server**: 304 Not Modified

- Uncacheable objects?
  - Dynamic data (quickly becomes stale)
  - User-specific data, cookies, encryption
  - Analytic/Tracking (deliberately make data unique)



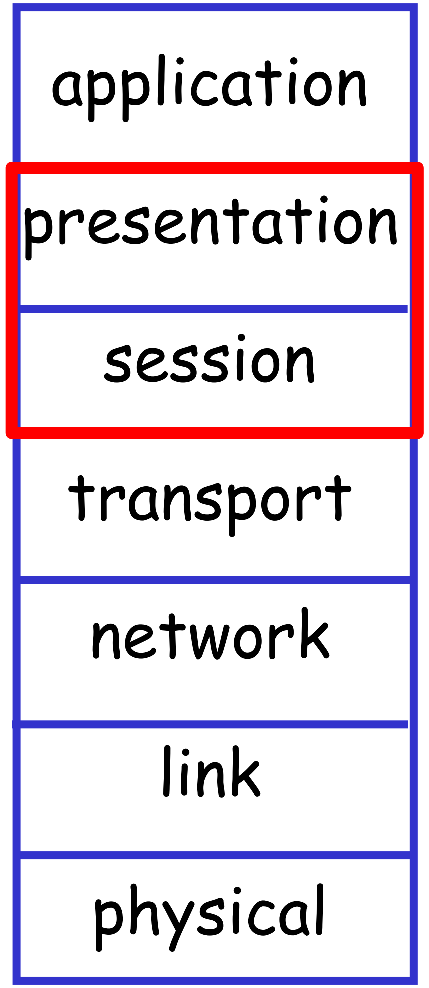
HTTP Proxy Summary

- Broker between client and server
  - Accepts request from client, forwards (a new request) to server
  - Gets responses from server, forwards (a new response) to client
  - 2 separate TCP connections, 2 separate HTTP connections
    - Proxy acts as HTTP server
- Why?
  - Performance
    - Lower latency → more efficient TCP
    - cache: fetch common content only once
      - less connections on server, less traffic, better latency
    - offload: perform common server tasks (e.g., encryption)
  - Privacy – hide client from server
  - Filtering – hide content from client

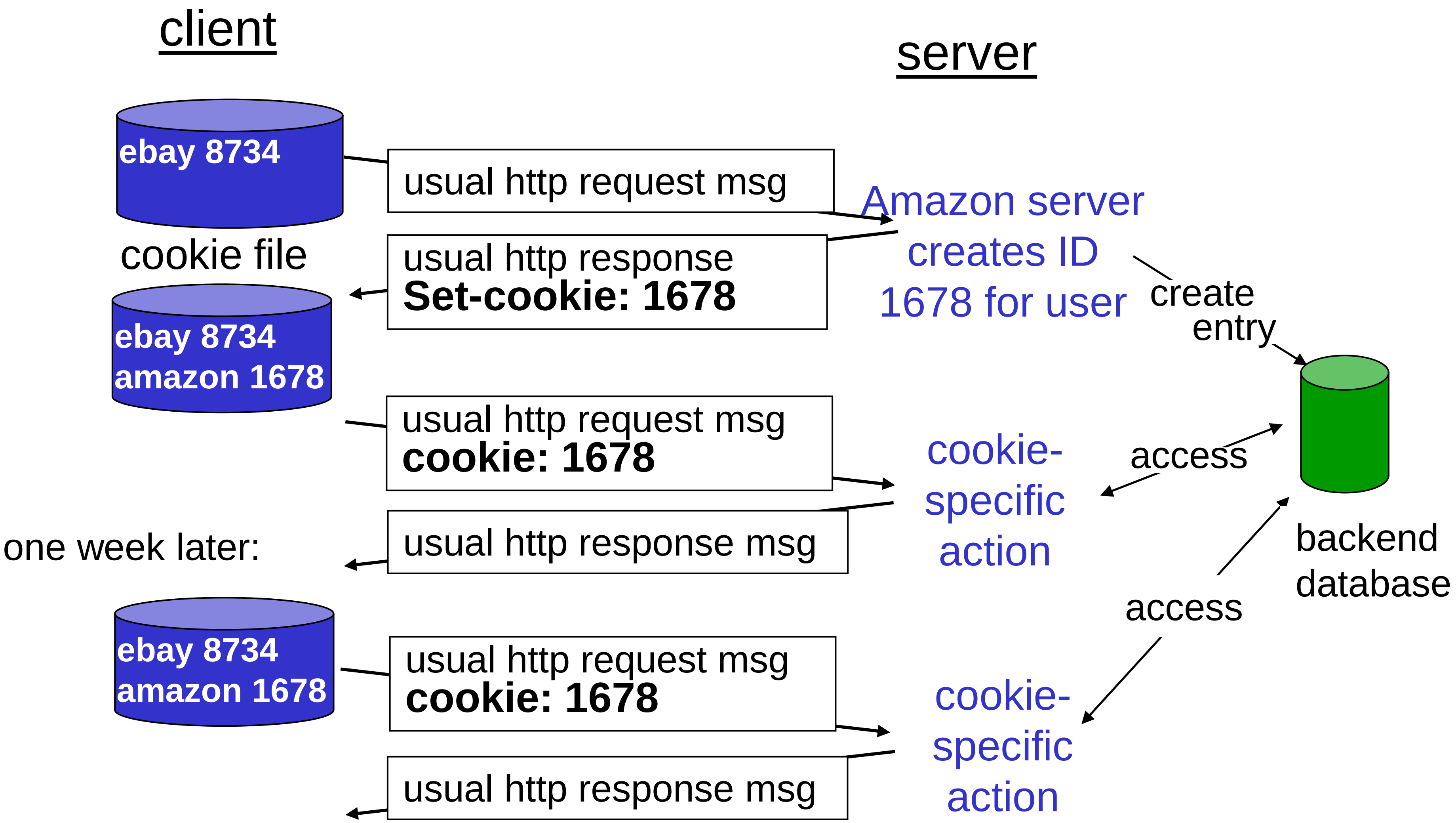


The missing layers

- Presentation: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- Session: synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - These services, if needed, must be implemented in application



Cookies: keeping "state"



client

server

ebay 8734

cookie file

usual http request msg

Amazon server creates ID 1678 for user

create entry

usual http response
Set-cookie: 1678

ebay 8734
amazon 1678

usual http request msg
cookie: 1678

cookie-specific action

access

backend database

usual http response msg

one week later:

access

ebay 8734
amazon 1678

usual http request msg
cookie: 1678

cookie-specific action

usual http response msg

Implementing user-server sessions: cookies

```
<img src="http://bank.example.com/withdraw?account=bob&amount=1000000&for=mallory">
```

possible solution: “same-origin policy”

- ❒ Mechanism
  - ❖ Cookie generated by web site on first time interaction
    - Unique id
    - Entry in back-end database at web site
    - Cookie header line of HTTP response message
  - ❖ Cookie stored by client
    - Cookie file kept on user’s host, managed by user’s browser
  - ❖ Cookie sent back to server on next interaction
    - Cookie header line in HTTP request message
- ❒ Motivation
  - ❖ Long-term state (e.g., membership, authorization)
  - ❖ Short-term state (e.g., shopping carts, email session)
  - ❖ User tracking (e.g., recommendations, advertisements)
    - Privacy issues, track user on multiple sites