
Warstwa aplikacji

część 1: HTTP

Sieci komputerowe

Wykład 8


Marcin Bieńkowski


Trochę statystyk: całkowity ruch (2022)

CATEGORY TRAFFIC SHARE		
TOTAL TRAFFIC		
	Category	Total Volume
1	Video	53.72%
2	Social	12.69%
3	Web	9.86%
4	Gaming	5.67%
5	Messaging	5.35%
6	Marketplace	4.54%
7	File Sharing	3.74%
8	Cloud	2.73%
9	VPN	1.39%
10	Audio	0.31%

Obrazek ze raportu <https://www.sandvine.com/phenomena>

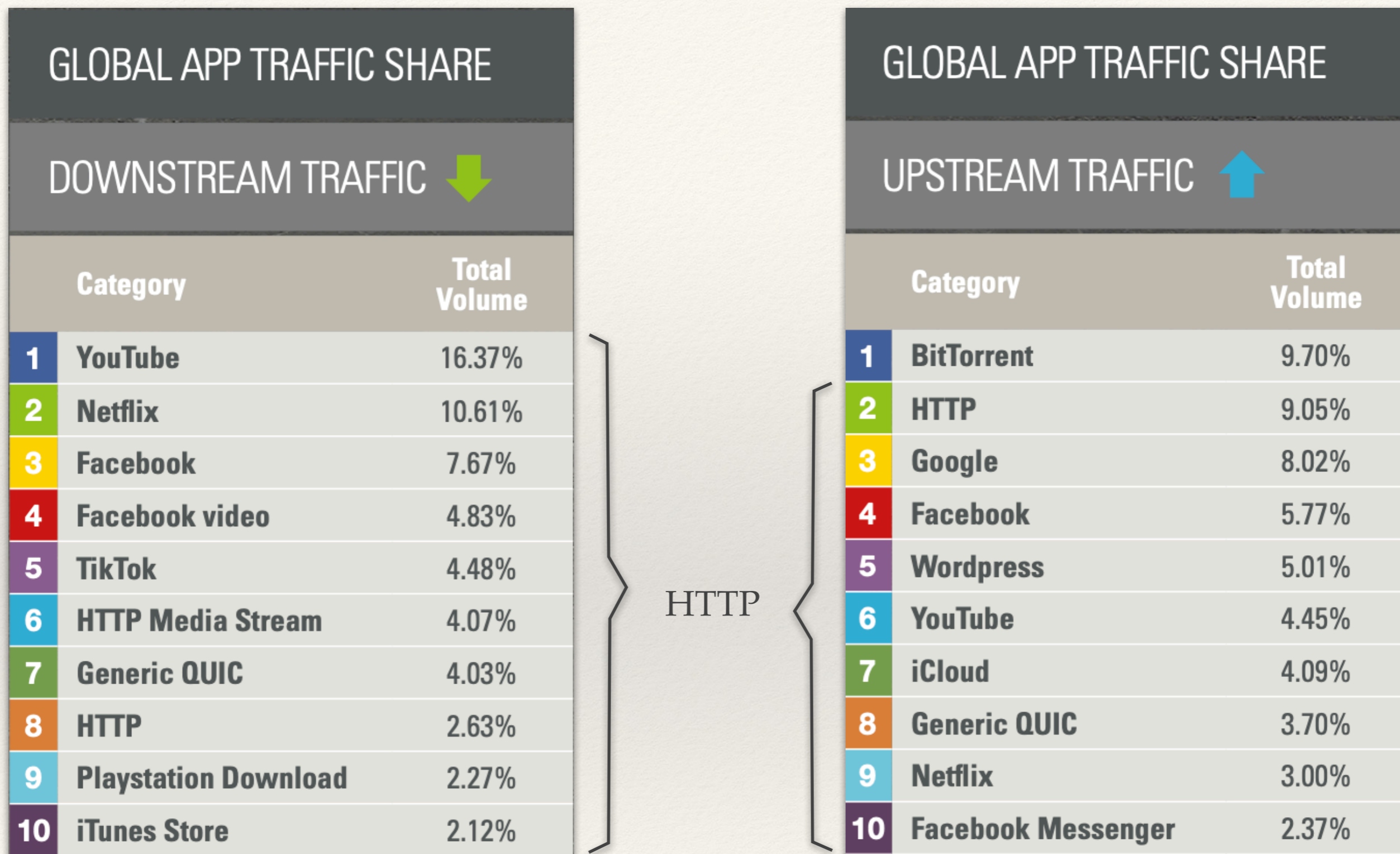
Trochę statystyk: aplikacje (2022)

GLOBAL APP TRAFFIC SHARE		
DOWNSTREAM TRAFFIC 		
	Category	Total Volume
1	YouTube	16.37%
2	Netflix	10.61%
3	Facebook	7.67%
4	Facebook video	4.83%
5	TikTok	4.48%
6	HTTP Media Stream	4.07%
7	Generic QUIC	4.03%
8	HTTP	2.63%
9	Playstation Download	2.27%
10	iTunes Store	2.12%

GLOBAL APP TRAFFIC SHARE		
UPSTREAM TRAFFIC 		
	Category	Total Volume
1	BitTorrent	9.70%
2	HTTP	9.05%
3	Google	8.02%
4	Facebook	5.77%
5	Wordpress	5.01%
6	YouTube	4.45%
7	iCloud	4.09%
8	Generic QUIC	3.70%
9	Netflix	3.00%
10	Facebook Messenger	2.37%

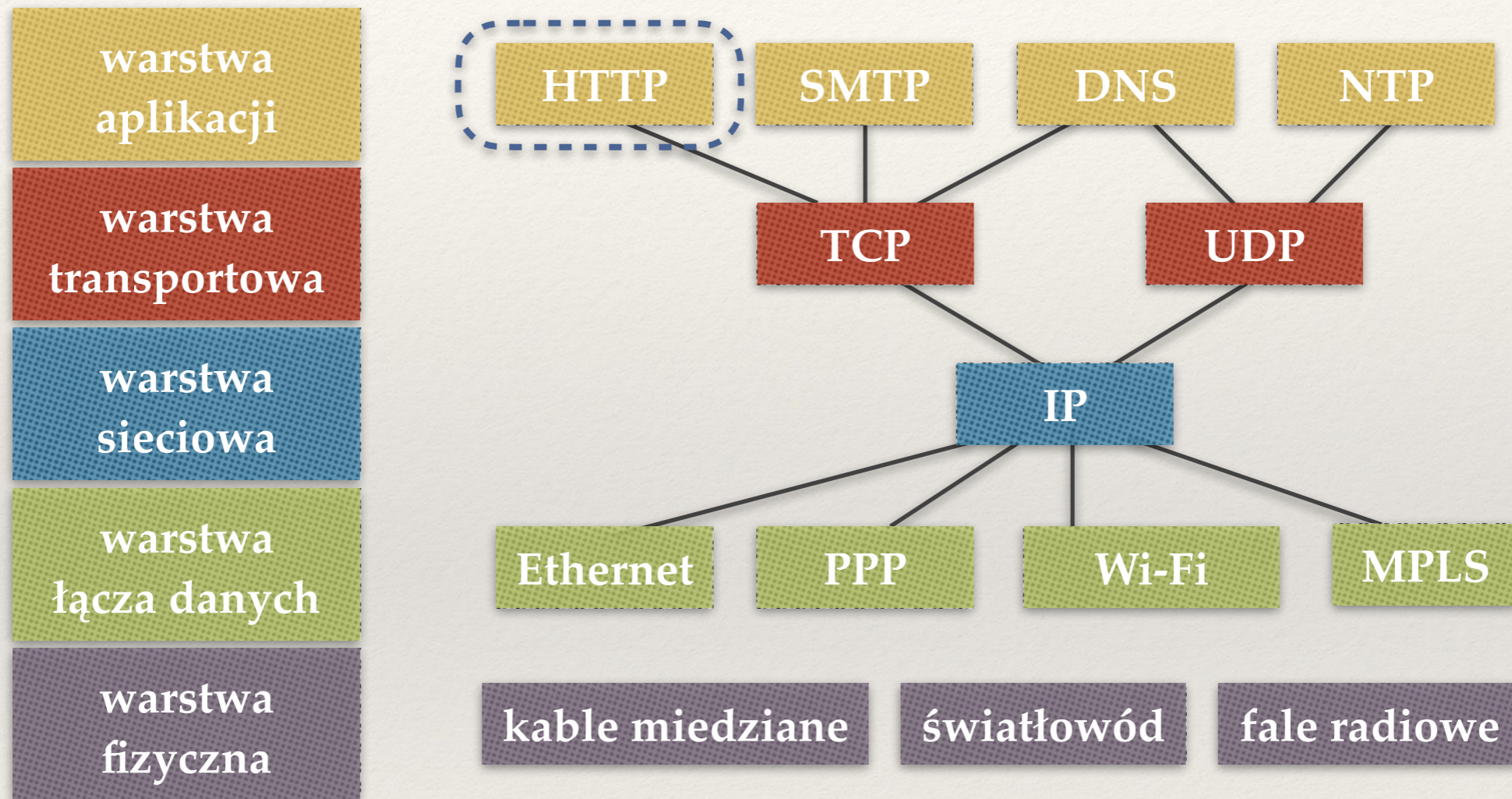
Obrazek ze raportu <https://www.sandvine.com/phenomena>

Trochę statystyk: aplikacje (2022)



Obrazek ze raportu <https://www.sandvine.com/phenomena>

HTTP (Hypertext Transfer Protocol)



HTTP/1.1

HTTP

- ❖ Zaprojektowany do przesyłania hipertekstu (tekst z odnośnikami).
- ❖ Obecnie: również do przesyłania przesyłania olbrzymich danych, streamingu video (Youtube, Netflix), ...
- ❖ Korzysta z protokołu TCP, portu 80 (szyfrowana wersja: port 443).

URL (Uniform Resource Locator)

Identyfikuje dany zasób.

- ❖ Składa się z 2 części rozdzielonych dwukropkiem:
 - ♦ schemat: (`http`, `https`, `ftp`, `mailto`, ...)
 - ♦ część zależna od rodzaju zasobu.

- ❖ Przykłady:
 - ♦ `http://www.ii.uni.wroc.pl/index.html`
 - ♦ `https://pl.wikipedia.org/wiki/URL`
 - ♦ `mailto:jan.kowalski@serwer.com`

URL dla schematu http lub https

❖ Po dwukropku:

♦ //

♦ nazwa serwera WWW

♦ opcjonalnie :port

♦ /

♦ identyfikator zasobu wewnątrz serwera:

♦ niekoniecznie ścieżka do pliku,

♦ / w identyfikatorze wskazuje na hierarchię.

❖ Przykład: `https://en.wikipedia.org:443/wiki/HTTP/3.`

Pobieranie strony WWW krok po kroku (1)

- ❖ Przeglądarka WWW dostaje URL.
- ❖ URL jest rozbijany na części.
- ❖ Nawiązuje połączenie TCP z portem 80 serwera WWW.
- ❖ Wysyła żądanie HTTP:

GET /wiki/HTTP/3 **HTTP/1.1**

Host: en.wikipedia.org

Accept: text/html;q=0.9,application/xml;q=0.8

Accept-Language: en-US,en;q=0.8,pl;q=0.6,de;q=0.4

User-Agent: Mozilla/5.0 ... Chrome/49.0.2623.112

Referer: https://www.google.com/

Pobieranie strony WWW krok po kroku (2)

- ❖ Serwer analizuje żądanie, pobiera z dysku odpowiedni plik.
- ❖ Serwer sprawdza typ MIME (*Multipurpose Internet Mail Extensions*) pliku (heurystyki na podstawie tego jak plik wygląda, rozszerzenia itp.). Przykłady:
 - ♦ `text/plain`
 - ♦ `text/html`
 - ♦ `image/jpeg`
 - ♦ `video/mpeg`
 - ♦ `application/msword` — dokument `.doc` (x)
 - ♦ `application/pdf` — dokument PDF
 - ♦ `application/octet-stream` — ciąg bajtów bez interpretacji

Pobieranie strony WWW krok po kroku (3)

- ❖ Serwer wysyła odpowiedź:

HTTP/1.1 **200 OK**

Server: Apache/2.4.38 ... OpenSSL/0.9.8k

Last-Modified: Wed, 29 Apr 2020 21:58:30 GMT

Content-Length: 5387

Content-Type: text/html

← typ MIME
wysyłanego pliku

PLIK (w tym przypadku dokument HTML)

- ❖ Serwer zamyka połączenie TCP (lub czeka na następne polecenie).
- ❖ Przeglądarka wykonuje akcję w zależności od typu MIME (wyświetla, używa wtyczki, używa zewnętrznej aplikacji, ...).

Typy odpowiedzi HTTP

- ❖ 1xx: informacyjne.
- ❖ 2xx: sukces (200 = OK).
- ❖ 3xx: przekierowania do strony X (po otrzymaniu przeglądarka wykona zapytanie HTTP o stronę X).
- ❖ 4xx: błąd po stronie klienta (błędne żądanie, brak autoryzacji, zabroniony dostęp, 404 = Not Found).
- ❖ 5xx: błąd po stronie serwera (500 = Internal Server Error).

Hipertekst

- ❖ Wiele standardów: HTML, XHTML, XML, ...
- ❖ Dokument zawiera:
 - ◆ odnośniki do innych dokumentów
 - ◆ oraz odnośniki do elementów osadzonych w dokumencie, m.in.:
 - ◆ obrazki i filmy,
 - ◆ skrypty w javascript,
 - ◆ arkusze stylów CSS (definiują wygląd, HTML określa tylko strukturę),
 - ◆ czcionki.
 - ◆ Elementy osadzone są pobierane przez kolejne żądania HTTP i wyświetlane przez przeglądarkę.

Bezstanowość

- ❖ **Protokół HTTP jest bezstanowy**
 - ♦ Serwer nie pamięta poprzednich żądań klienta.
 - ♦ Zapytanie klienta musi zawierać wszystkie informacje potrzebne serwerowi.

Bezstanowość

- ❖ **Protokół HTTP jest bezstanowy**
 - ♦ Serwer nie pamięta poprzednich żądań klienta.
 - ♦ Zapytanie klienta musi zawierać wszystkie informacje potrzebne serwerowi.

- ❖ **To jak działają:**
 - ♦ koszyk w sklepie internetowym?
 - ♦ ustawienie języka czy innych preferencji?

Aplikacje po stronie serwera HTTP

Takie zapytanie można zrobić np. formularzem w którym użytkownik wpisze „3”.



Aplikacje po stronie serwera HTTP

Takie zapytanie można zrobić np. formularzem w którym użytkownik wpisze „3”.

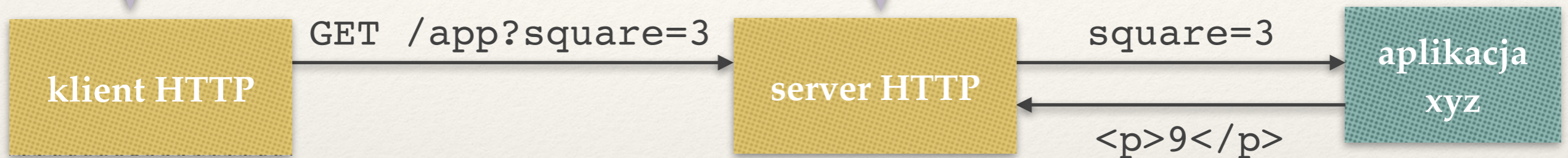
Skonfigurowany, żeby rzeczy zaczynające się od /app przekazywać do aplikacji xyz.



Aplikacje po stronie serwera HTTP

Takie zapytanie można zrobić np. formularzem w którym użytkownik wpisze „3”.

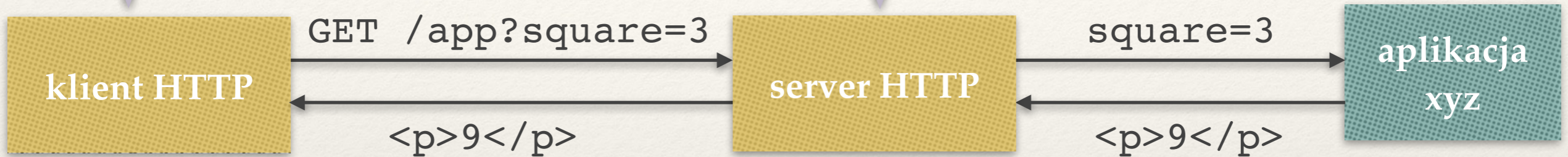
Skonfigurowany, żeby rzeczy zaczynające się od /app przekazywać do aplikacji xyz.



Aplikacje po stronie serwera HTTP

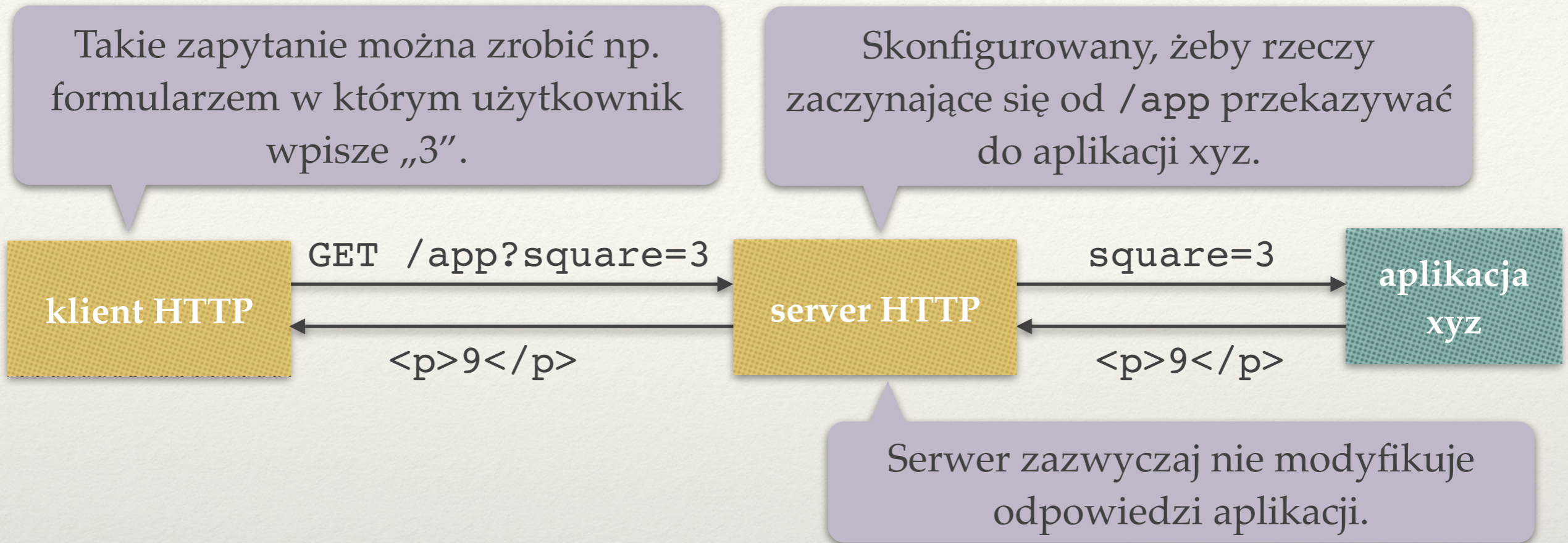
Takie zapytanie można zrobić np. formularzem w którym użytkownik wpisze „3”.

Skonfigurowany, żeby rzeczy zaczynające się od /app przekazywać do aplikacji xyz.



Serwer zazwyczaj nie modyfikuje odpowiedzi aplikacji.

Aplikacje po stronie serwera HTTP



Komunikacja między serwerem a aplikacją:

- ❖ Opcja 1: każdorazowe uruchamianie xyz (standard CGI, mało wydajne).
- ❖ Opcja 2: aplikacja jest modułem serwera (w PHP, Pythonie, ...).
- ❖ Opcja 3: aplikacja po prostu rozumie HTTP.
 - ♦ Serwer HTTP jest tylko pośrednikiem + serwuje statyczne strony.

Cookie



Cookie



Cookie



Zapamiętuje informacje
o użytkowniku
pod identyfikatorem sesji ba...3953.

Cookie



W przypadku komunikacji z tym samym serwerem wysyła wszystkie ustawione uprzednio Cookie.

Zapamiętuje informacje o użytkowniku pod identyfikatorem sesji ba...3953.

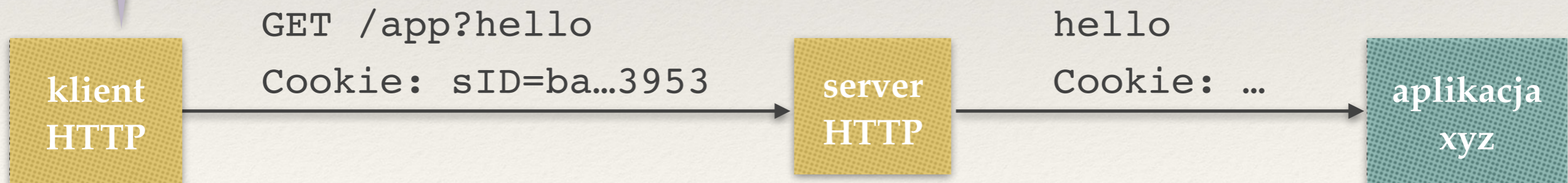


Cookie



W przypadku komunikacji z tym samym serwerem wysyła wszystkie ustawione uprzednio Cookie.

Zapamiętuje informacje o użytkowniku pod identyfikatorem sesji ba...3953.



Cookie



W przypadku komunikacji z tym samym serwerem wysyła wszystkie ustawione uprzednio Cookie.

Zapamiętuje informacje o użytkowniku pod identyfikatorem sesji ba...3953.



Formularze

❖ Wysyłanie metodą GET.

- ♦ Przeglądarka pobiera stronę `http://domena/program?par1=val1&par2=val2`.
- ♦ Serwer WWW uruchamia program i przekazuje mu parametry, program generuje odpowiedź (np. HTML).
- ♦ Problem: nie powinno się tak przekazywać haseł. (Dlaczego?)
- ♦ Problem: ograniczenie na rozmiar przekazywanych danych.

❖ Wysyłanie metodą POST.

- ♦ Przeglądarka wysyła żądanie POST o stronę `http://domena/program`.
- ♦ W treści żądania (nie w nagłówku) znajduje się `par1=val1&par2=val2`.
- ♦ Można w ten sposób wysyłać też pliki do serwera.

JavaScript

- ❖ Część dynamiki strony WWW odbywa się wyłącznie po stronie klienta, bez komunikacji z serwerem.
- ❖ **Javascript:** język zintegrowany z HTML, mający łatwy dostęp do struktury dokumentu i mogący ją modyfikować.
 - ◆ Wzbogacany przez różne biblioteki (React, Angular, ...).
 - ◆ Może też sam wysyłać zapytania HTTP do serwera.

Poprawianie wydajności

Połączenia trwałe (1)

❖ Do HTTP 1.0.

- ♦ Każda para żądanie-odpowiedź w osobnym połączeniu TCP.
- ♦ Nawiazywanie połączenia TCP → duży narzut czasowy.
- ♦ Kończenie połączenia TCP → narzut czasowy + dużo połączeń w stanie `TIME_WAIT`.

❖ Od HTTP 1.1.

- ♦ Wiele żądań i odpowiedzi w jednym połączeniu TCP.
- ♦ Połączenie domyślnie otwarte.
- ♦ Zamknięcie połączenia po odpowiedzi na żądanie, w którym umieścimy wiersz `Connection: close`.

Połączenia trwałe (2)

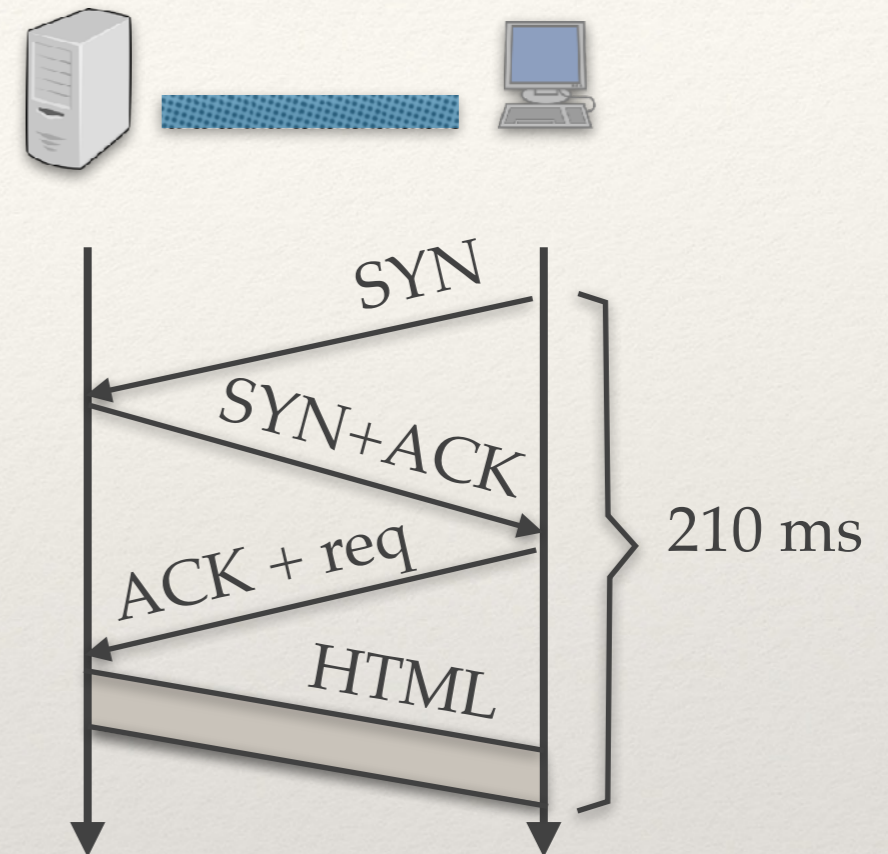
Przykład:

- ❖ Pobieranie strony HTML + 10 obrazków.
- ❖ Każdy obiekt mieści się w jednym segmencie TCP.
- ❖ Czas propagacji: 50 ms.
- ❖ Czas nadawania (pełnego) segmentu z danymi: 10 ms.
- ❖ Czas nadawania segmentu kontrolnego TCP lub segmentu z zapytaniem HTTP: 0 ms.

Połączenia trwałe (3)

HTTP/1.0 (bez połączeń trwałych).

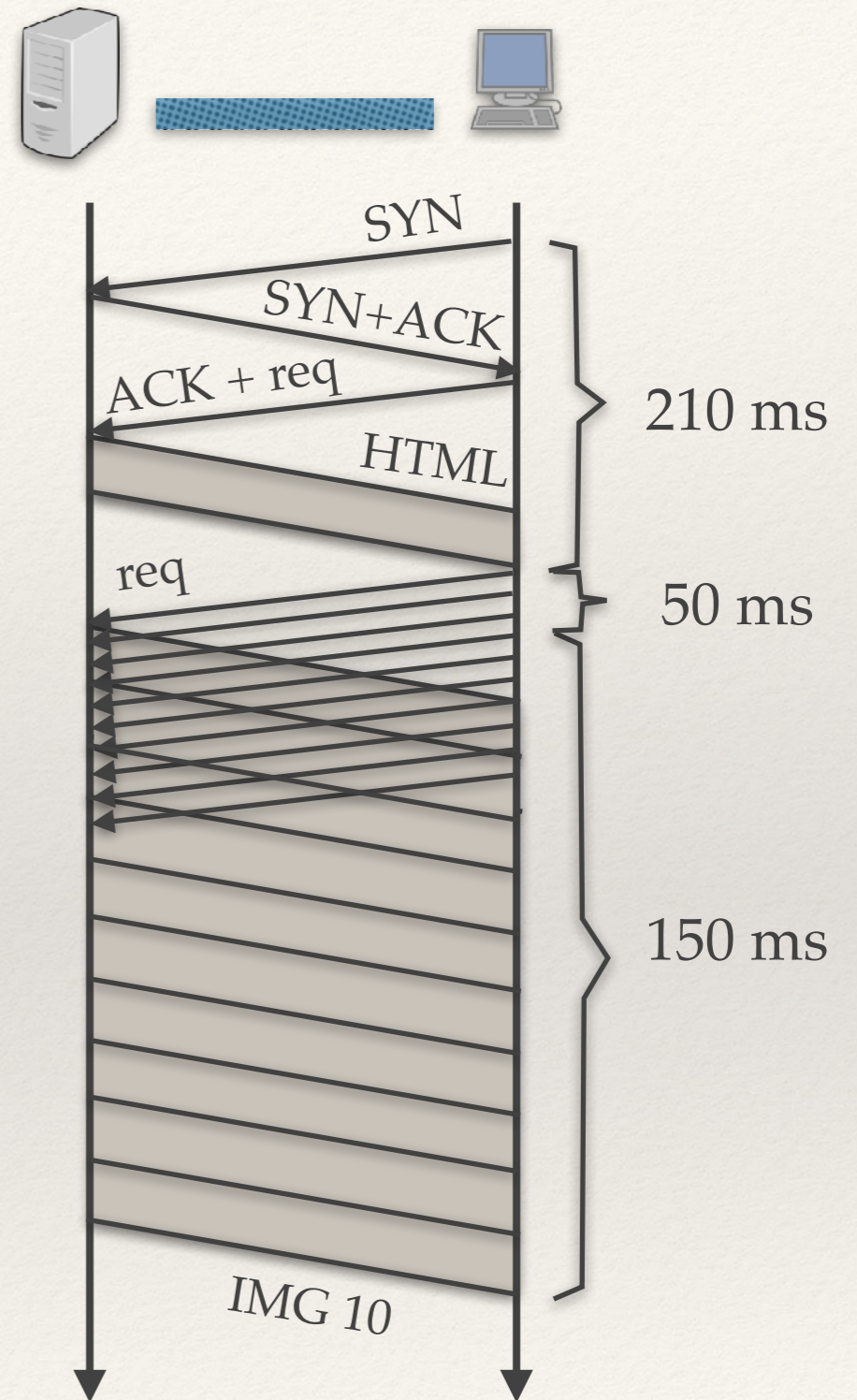
- ❖ Otrzymanie strony HTML: 210 ms.
- ❖ Pobieranie każdego z obrazków: kolejne: 210 ms.
- ❖ Usprawnienie: dwa równoległe połączenia do serwera
→ pobieranie 10 obrazków trwa $210 \text{ ms} \cdot (10/2) = 1050 \text{ ms}$.
- ❖ Całkowity czas: $210 + 1050 = 1260 \text{ ms}$.



Połączenia trwałe (4)

HTTP/1.1 (połączenia trwałe).

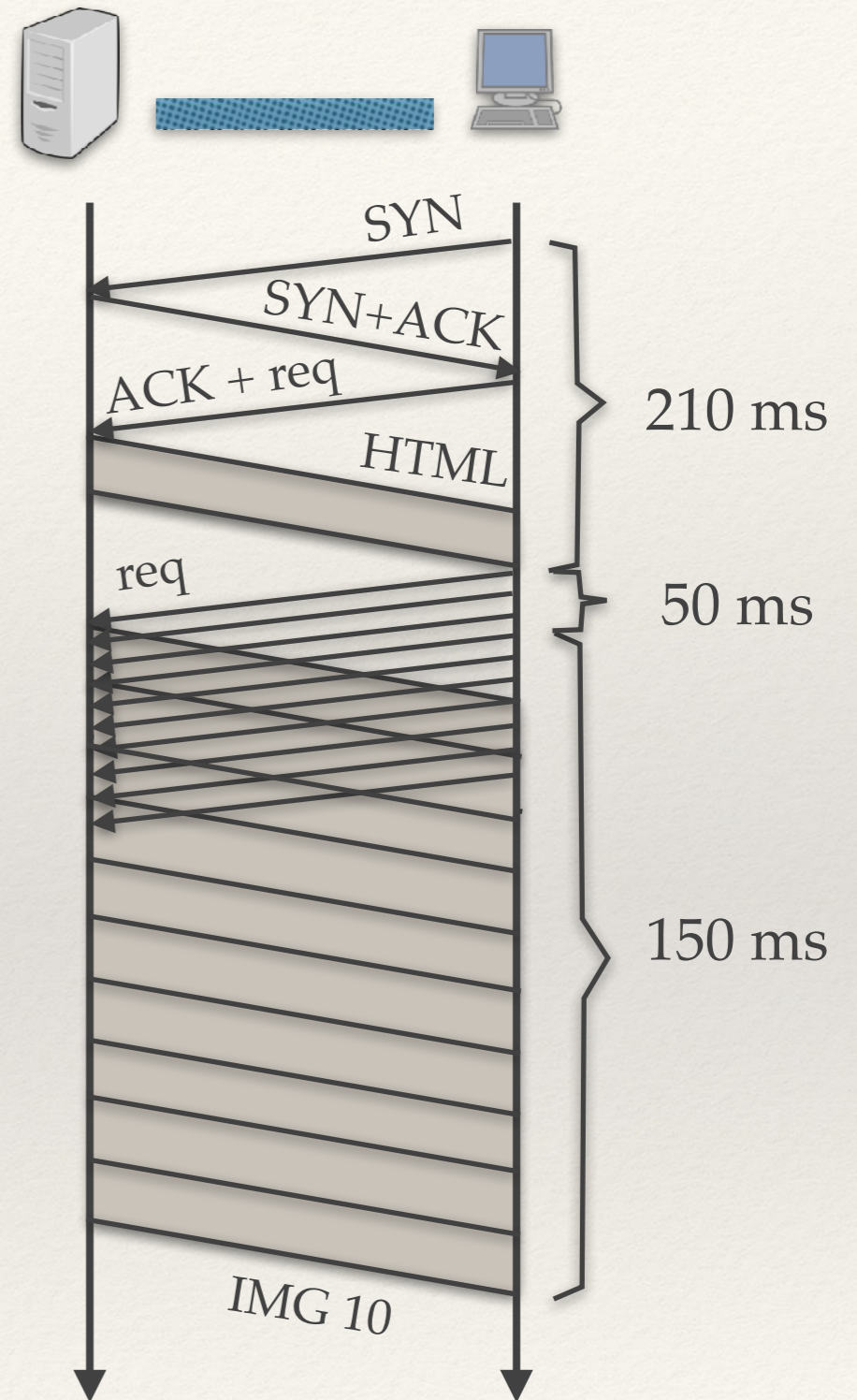
- ❖ Otrzymanie strony HTML: 210 ms.
- ❖ Zapytanie o obrazek nr 1: 50 ms.
- ❖ Wysyłanie obrazków:
 $50 + 10 \cdot 10 = 150$ ms.
- ❖ Całkowity czas: $210 + 200 = 410$ ms.



Połączenia trwałe (4)

HTTP/1.1 (połączenia trwałe).

- ❖ Otrzymanie strony HTML: 210 ms.
- ❖ Zapytanie o obrazek nr 1: 50 ms.
- ❖ Wysyłanie obrazków:
 $50 + 10 \cdot 10 = 150$ ms.
- ❖ Całkowity czas: $210 + 200 = 410$ ms.
- ❖ Niepotrzebne dodatkowe połączenia TCP.
- ❖ Jedno połączenie: okno TCP szybciej rośnie (kontrola przeciążenia).



Zapytanie warunkowe GET

- ❖ W nagłówku podajemy:

If-Modified-Since: Wed, 20 Apr 2021 23:27:04 GMT

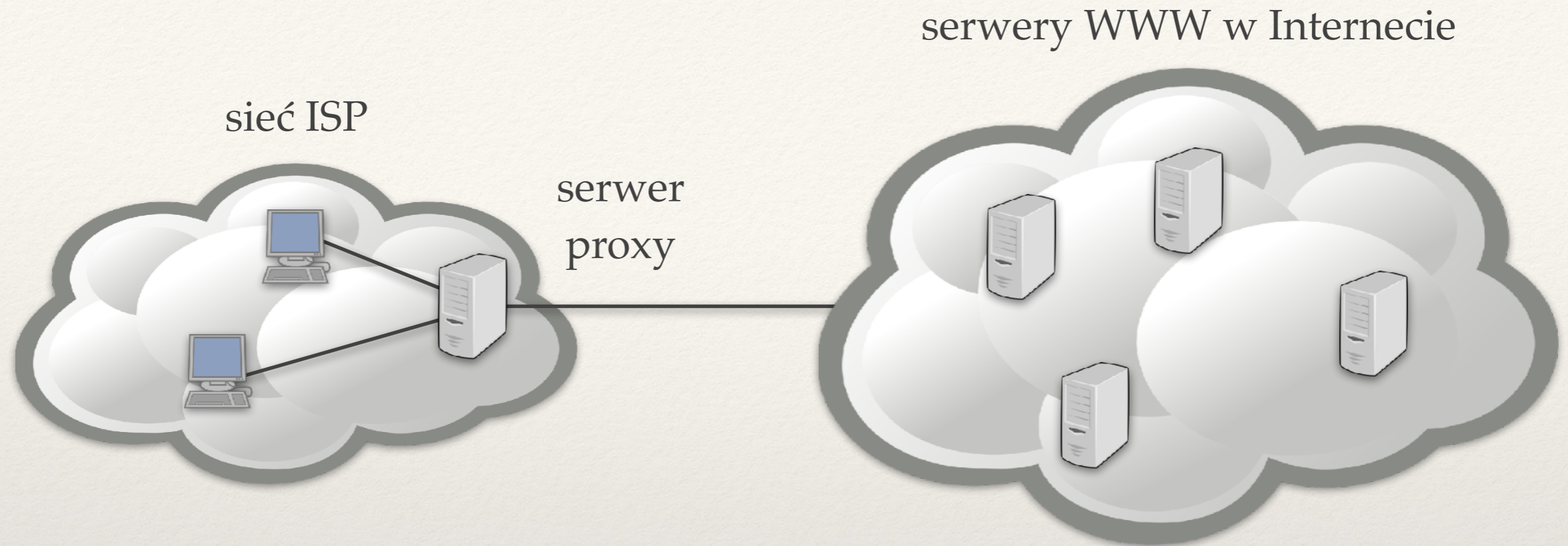
- ❖ Możliwe odpowiedzi:

- ◆ 200 OK
- ◆ 304 Not Modified

Pamięć podręczna w przeglądarce WWW

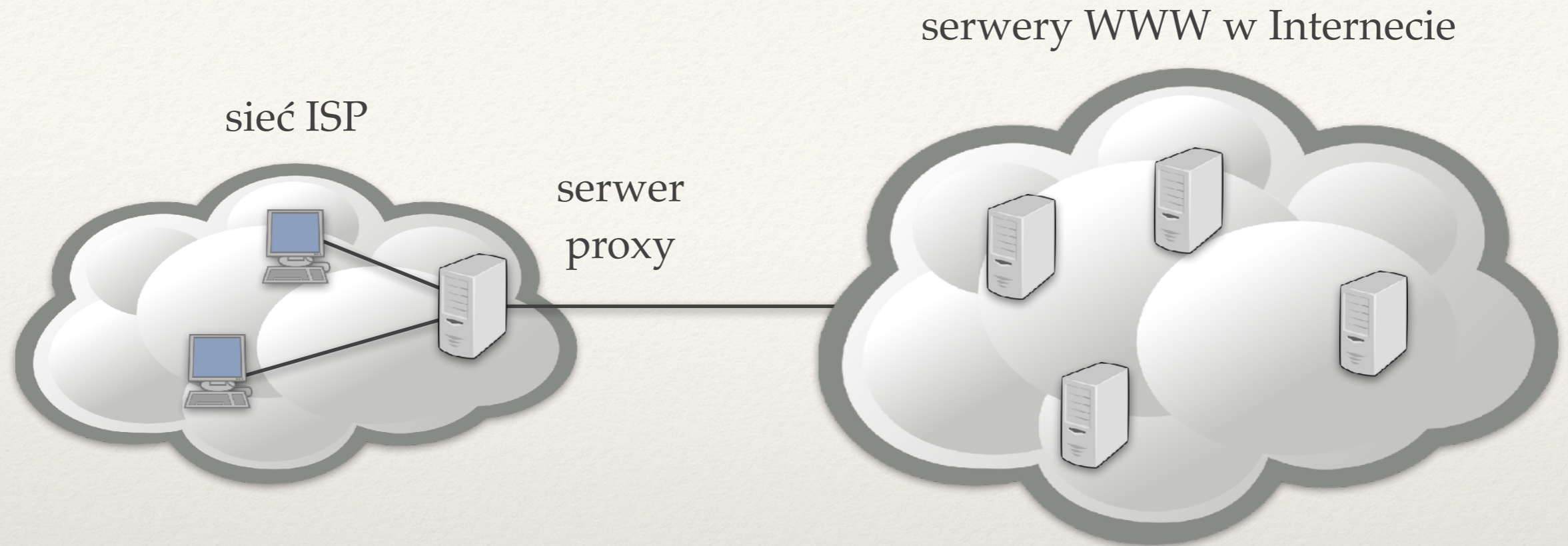
- ❖ Serwer może umieszczać w nagłówku odpowiedzi pola:
 - ◆ `Expires`: (do kiedy można trzymać dokument w pamięci podręcznej) → można całkowicie pominąć żądanie strony.
 - ◆ `Cache-Control: no-cache` (nigdy nie trzymaj w pamięci podręcznej)

Serwer proxy (1)



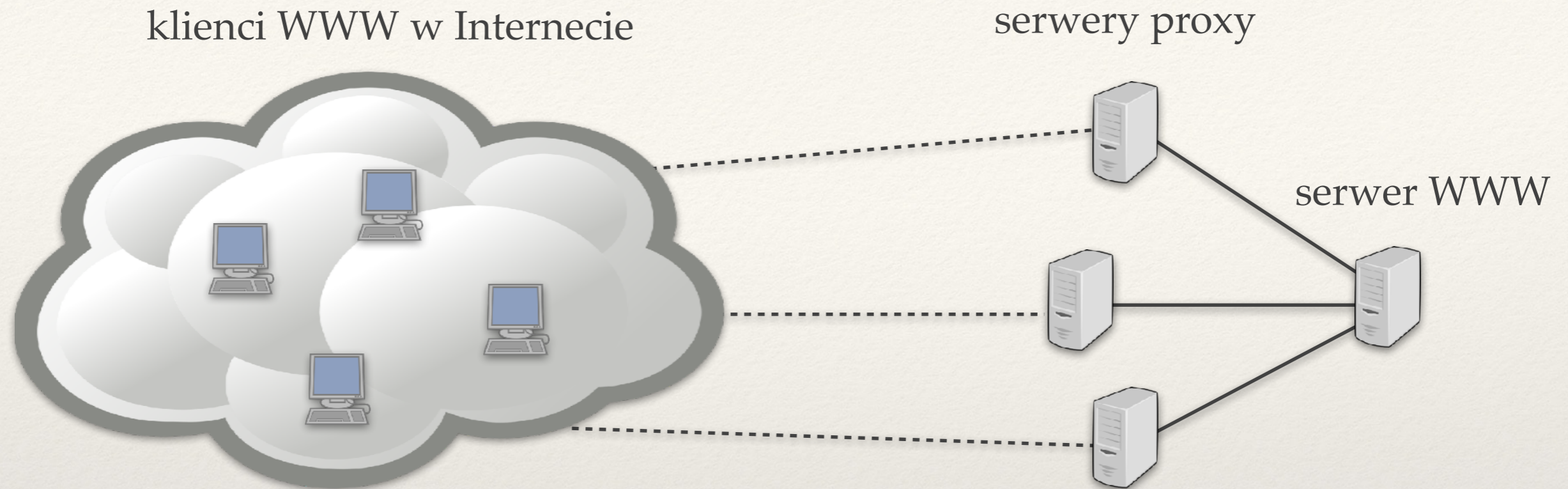
- ❖ Przeglądarka wysyła zapytanie HTTP do serwera proxy.
- ❖ Proxy w razie potrzeby łączy się z serwerem HTTP.
- ❖ Serwer proxy odpowiada używając stron przechowywanych w swojej pamięci podręcznej.
- ❖ W razie potrzeby przeglądarka może wymusić pominięcie proxy.

Serwer proxy (2)



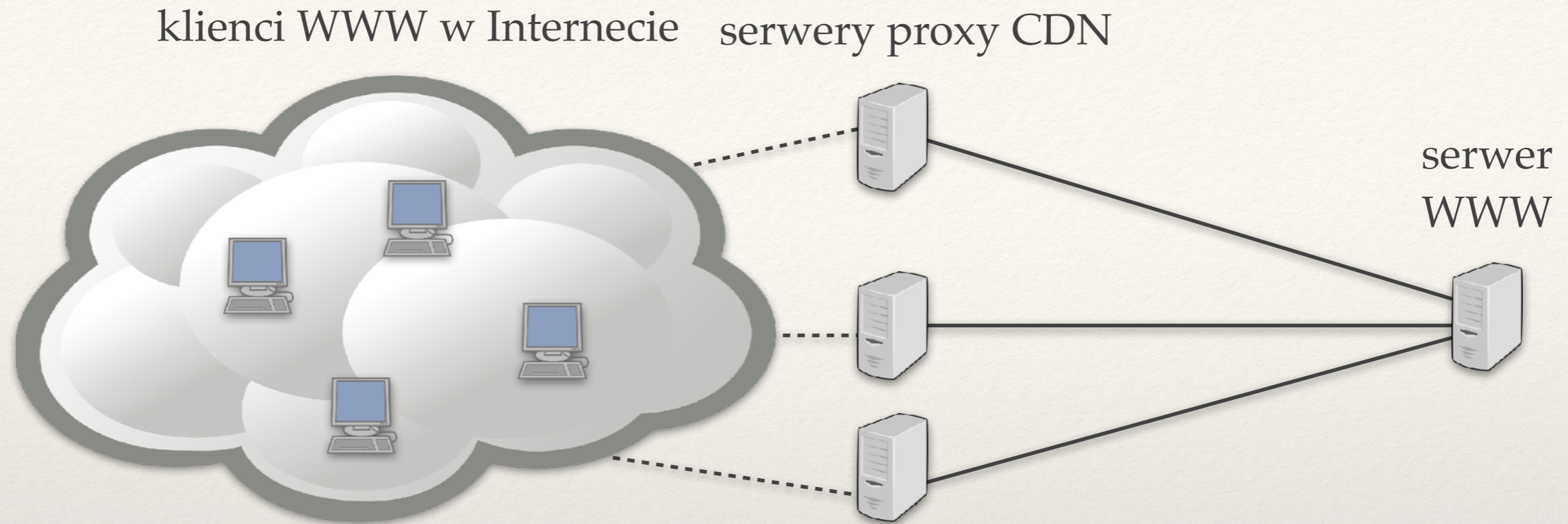
- ❖ Wpisywany w ustawieniach przeglądarki HTTP.
- ❖ Czasem wymuszany przez ISP (integrowany z routerem obsługującym ruch z danej sieci).
- ❖ Korzyści: głównie dla ISP (ograniczenie ilości danych).

Odwrotne proxy



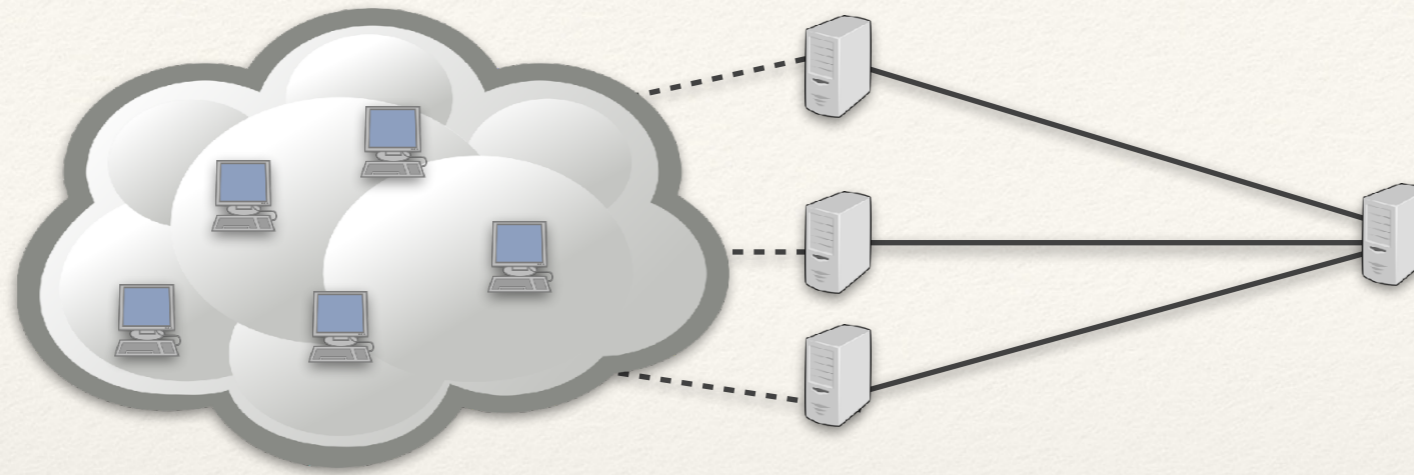
- ❖ Wykorzystywane przez dostawców treści (zmniejszają obciążenie serwera WWW).
- ❖ Zysk dla klienta i dostawcy treści.
- ❖ Ale: duże opóźnienie w przesyłaniu pakietów pomiędzy klientami i serwerami proxy.
- ❖ **Jak opłacalnie przysunąć serwery proxy do klientów?**

CDN (Content Distribution Networks)



- ❖ Serwery proxy obsługiwane przez osobną organizację (obsługuje wiele serwerów WWW).
 - ♦ Cloudflare, Akamai, ...
 - ♦ Setki tysięcy serwerów proxy.
- ❖ **Jak sprawić, żeby klient zamiast do serwera WWW trafił do serwera proxy?**

CDN: znajdowanie serwera proxy



Jak skierować klienta do odpowiedniego (bliskiego) serwera CDN?

- ❖ Dawno temu: klient pobiera stronę z serwera WWW zawierającą (automatycznie generowane) URL-e do obrazków na serwerach proxy.
- ❖ Później: zamiast strony WWW klient dostaje przekierowanie do odpowiedniego serwera proxy (*HTTP redirect*).
- ❖ Współcześnie: za pomocą serwerów DNS (utrzymywanych przez CDN): zapytanie o daną domenę zwraca adres IP serwera proxy.

Dygresja: anonimizujące serwery proxy

- ❖ **Serwer proxy dodaje do żądania HTTP dodatkowe pola.**
 - ◆ `X-Forwarded-For`: adres IP.
 - ◆ `Via`: adres IP proxy.
- ❖ **Anonimizujące serwery proxy:**
 - ◆ Nie dodają takich nagłóweków.
 - ◆ Zwykle płatne.

HTTP jako
„warstwa transportowa”

REST

- ❖ Pisanie poprawnych programów korzystających z TCP jest niełatwe.
- ❖ Jak wykorzystać HTTP do przesyłania danych?
- ❖ Testowego klienta (przeglądarkę www) mamy za darmo.

- ❖ **REST**
 - ◆ Zautomatyzowany dostęp do niektórych serwisów WWW (eBay, Amazon, Twitter, ...).
 - ◆ REST (*Representational State Transfer*) tworzenie usługi sieciowej wykorzystując metody (GET, PUT, POST, DELETE) protokołu HTTP.
 - ◆ REST nie jest standardem, raczej filozofią.
 - ◆ Łatwy do zautomatyzowania, czytelny dla człowieka.

Przesyłanie nieinteraktywnego video

Współcześnie video enkapsulowane jako dane protokołu HTTP.

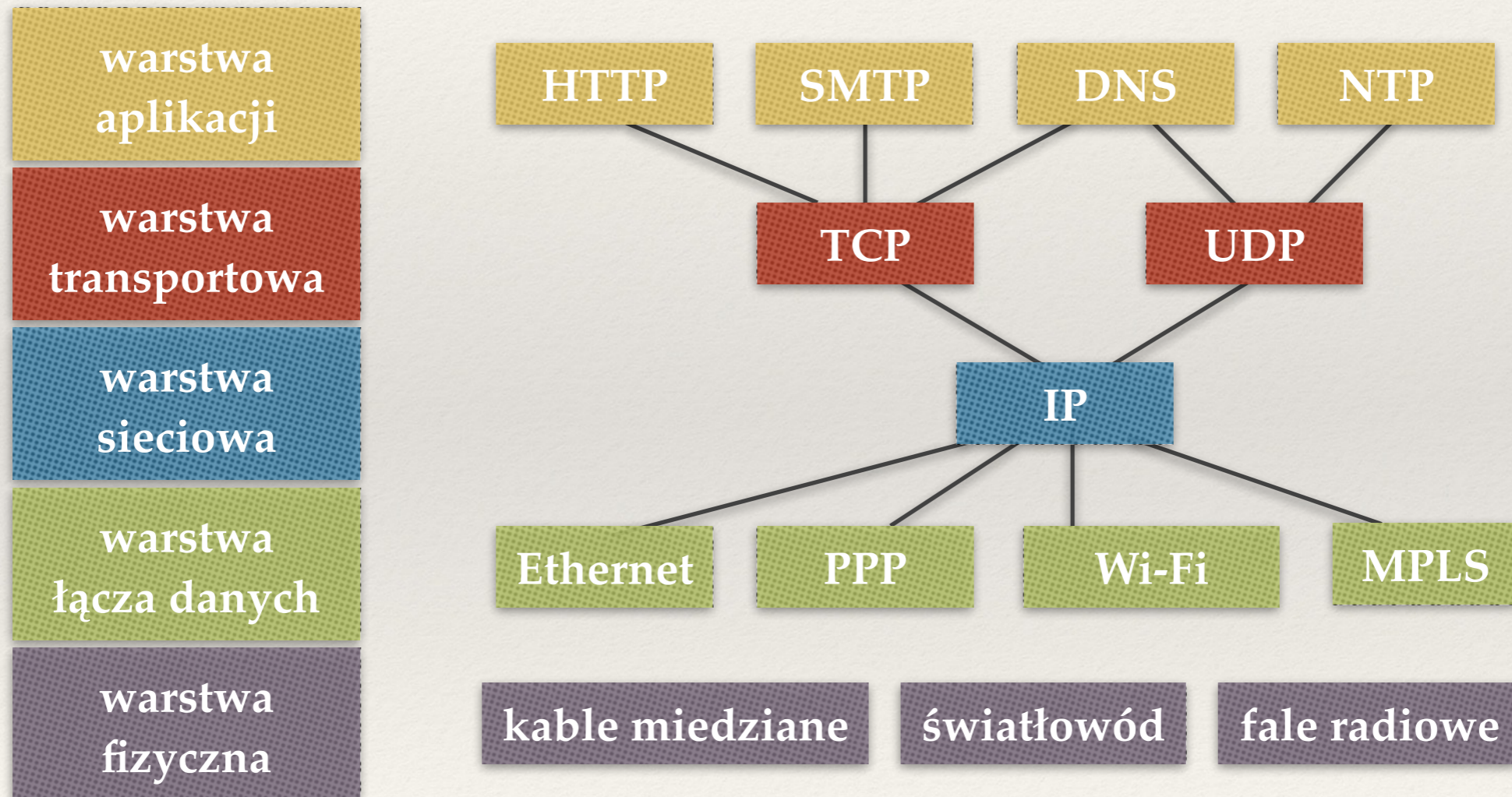
- ❖ DASH = Dynamic Adaptive Streaming over HTTP.
 - ◆ Youtube, Netflix, HBO, Amazon Prime, ...
- ❖ HLS = HTTP Live Streaming.
 - ◆ Twitch, HBO, Apple TV, ...
- ❖ Wykorzystują pola Range: `bytes=xxx-yyy` w nagłówku zapytania HTTP.

HTTP/2 i HTTP/3

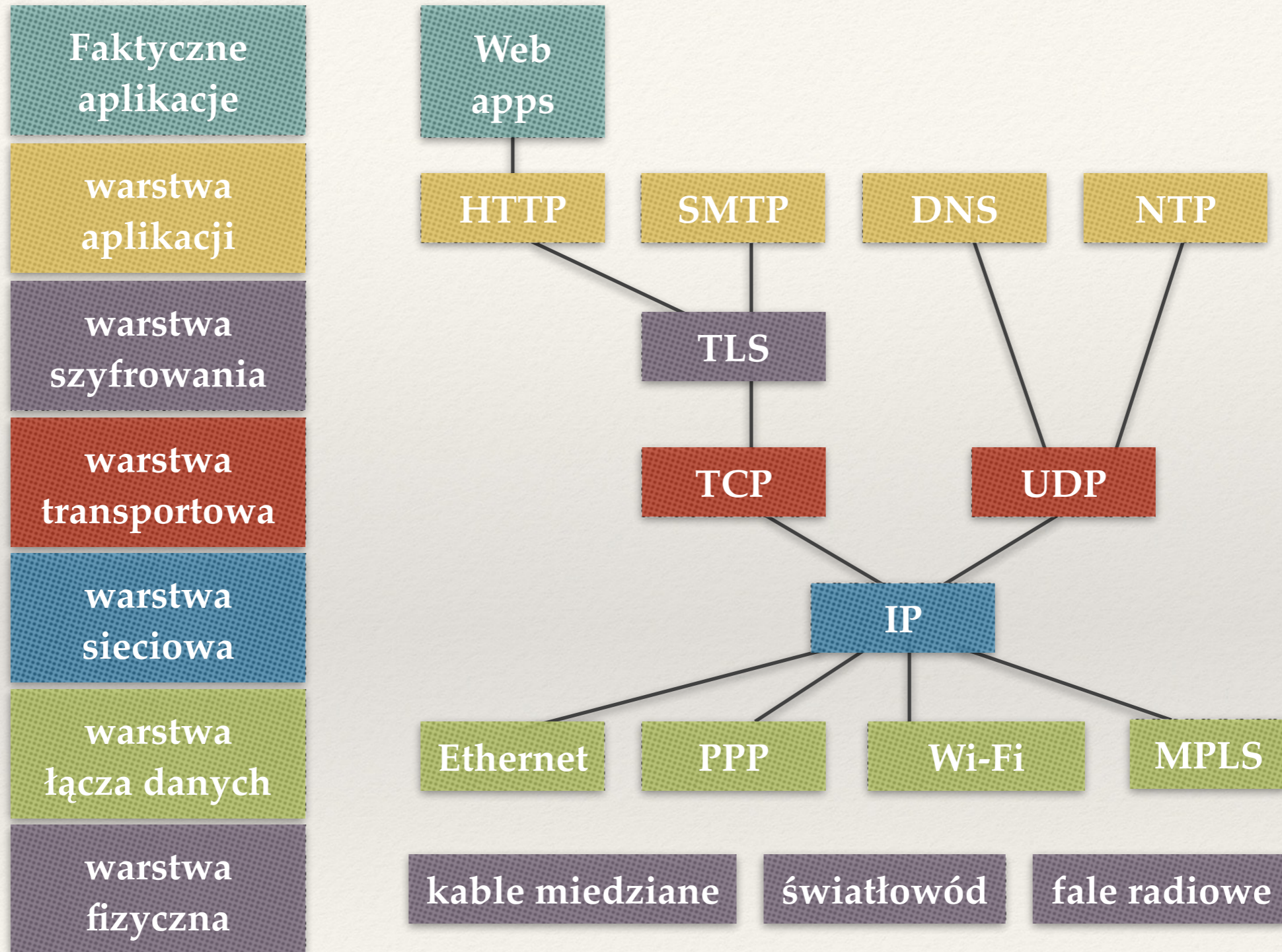
HTTP/2

- ❖ Binarny protokół.
- ❖ Kolejowanie żądań (*pipelining*, obecny już w HTTP / 1.1) + przesyłanie odpowiedzi w innej kolejności niż żądania.
- ❖ Server push: wysyłanie odpowiedzi na niezadane zapytania.
- ❖ Usuwanie powtarzających się nagłówków.
- ❖ Kompresja.
- ❖ Obowiązkowe szyfrowanie (TLS \geq 1.2).

Protokoły w Internecie

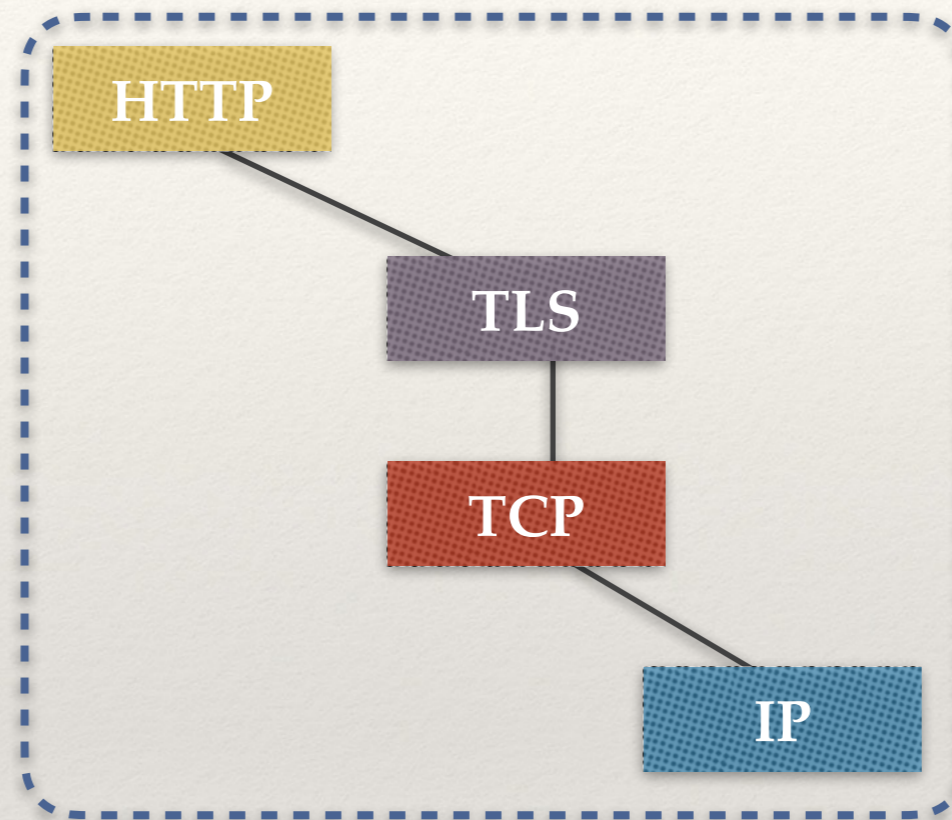


Protokoły w Internecie (dokładniej)

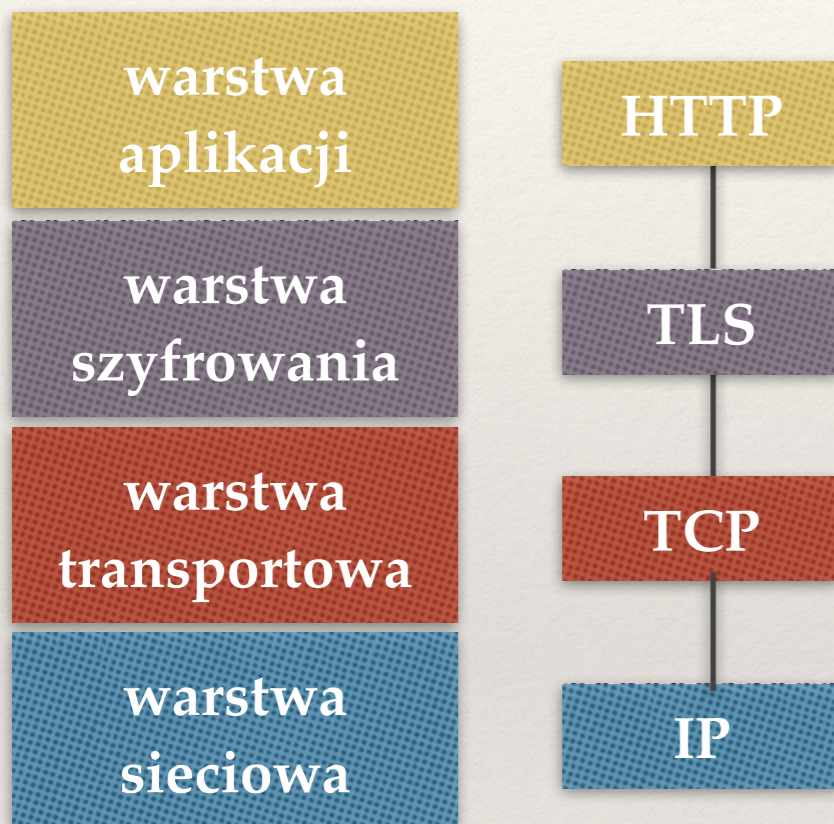


Dominująca technologia

85-90% wolumenu
ruchu sieciowego



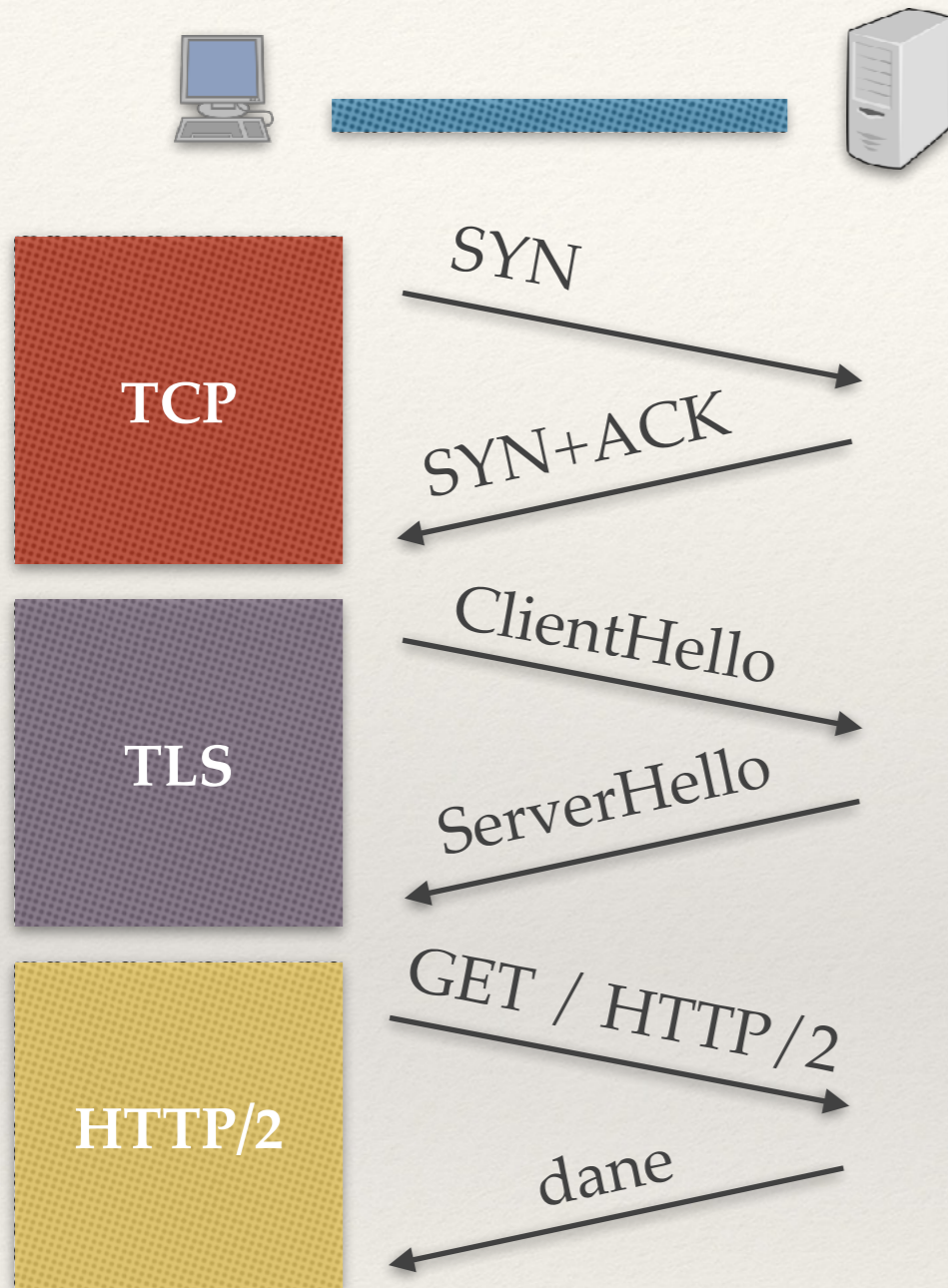
Model warstwowy



Dlaczego mamy model warstwowy?

- ❖ Względy historyczne: różne warstwy powstawały w różnych latach.
- ❖ Warstwy abstrakcji i podział zadań (np. powyżej warstwy transportowej nie musimy się przejmować niezawodnością dostarczania i przeciążeniem sieci / odbiorcy).

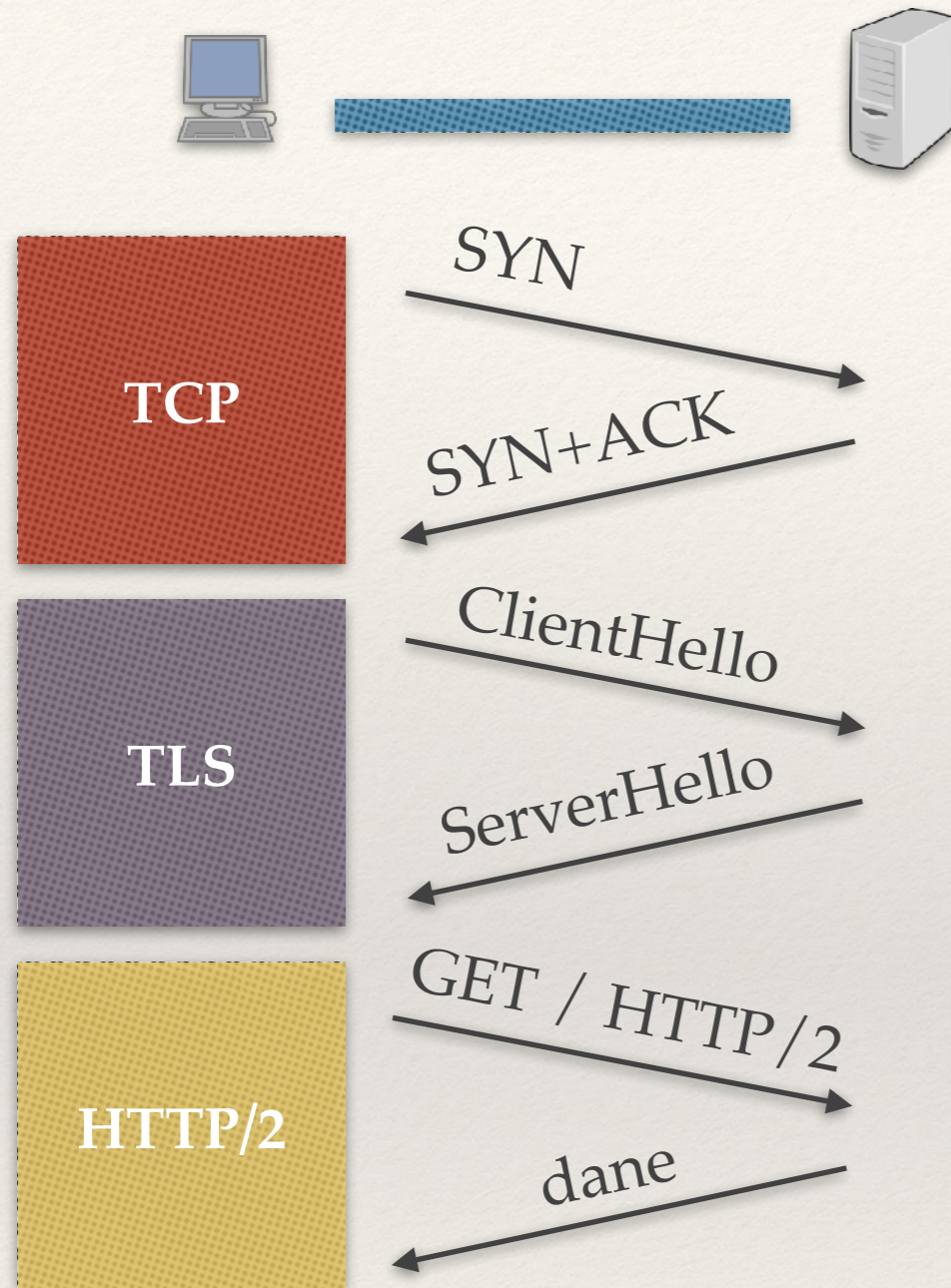
Problem 1: trzy nawiązywania połączenia



3 x RTT zanim klient zobaczy początek danych.

- ❖ Jeśli jest to pierwsze połączenie z danym serwerem, to może trwać jeszcze dłużej (wymiana kluczy kryptograficznych w TLS).

Problem 1: trzy nawiązywania połączenia



niemożliwe istotne zmniejszenie RTT
(prędkość światła)

3 x RTT zanim klient zobaczy
początek danych.

- ❖ Jeśli jest to pierwsze połączenie z danym serwerem, to może trwać jeszcze dłużej (wymiana kluczy kryptograficznych w TLS).

Problem 2: TCP utrzymuje kolejność

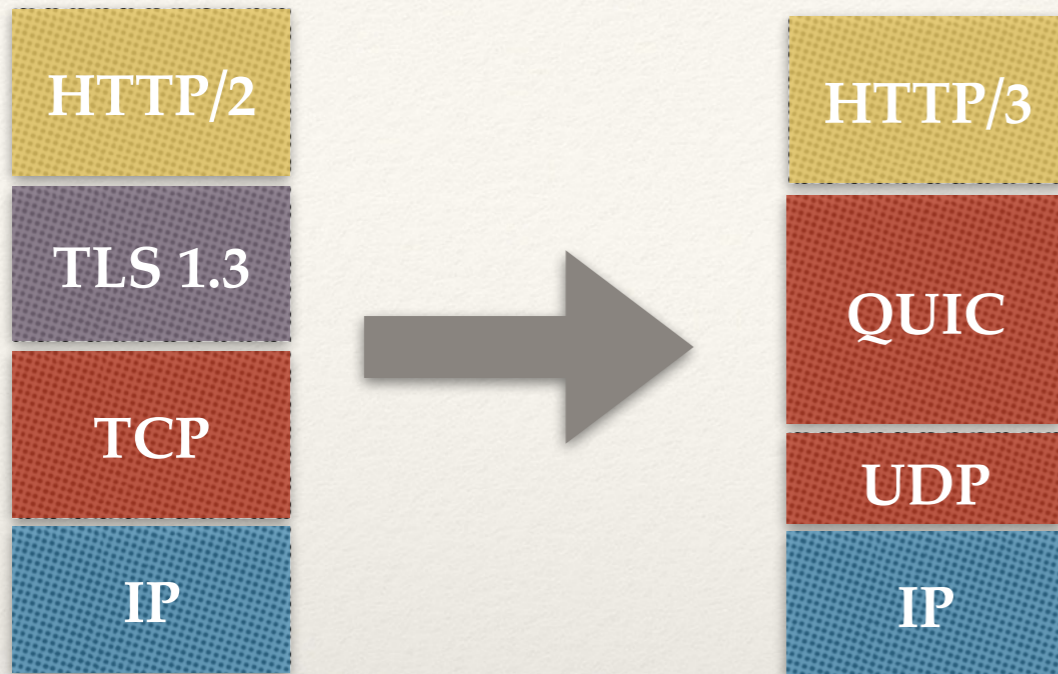
HTTP/2 wprowadza multiplexing zapytań i odpowiedzi:

- ❖ Może przesyłać odpowiedzi w innej kolejności niż pytania.
- ❖ Server push: wysyłanie odpowiedzi na niezadane zapytania.

Ale te informacje wkładane są w pojedynczy strumień danych, którym zarządza TCP. Przykład:

- ❖ Ginie segment zawierający mało istotny obrazek.
- ❖ Kolejne segmenty z innymi odpowiedziami HTTP docierają do klienta.
- ❖ Warstwa HTTP ich nie dostaje, bo TCP musi zrekonstruować strumień.

QUIC: nowy protokół transportowy



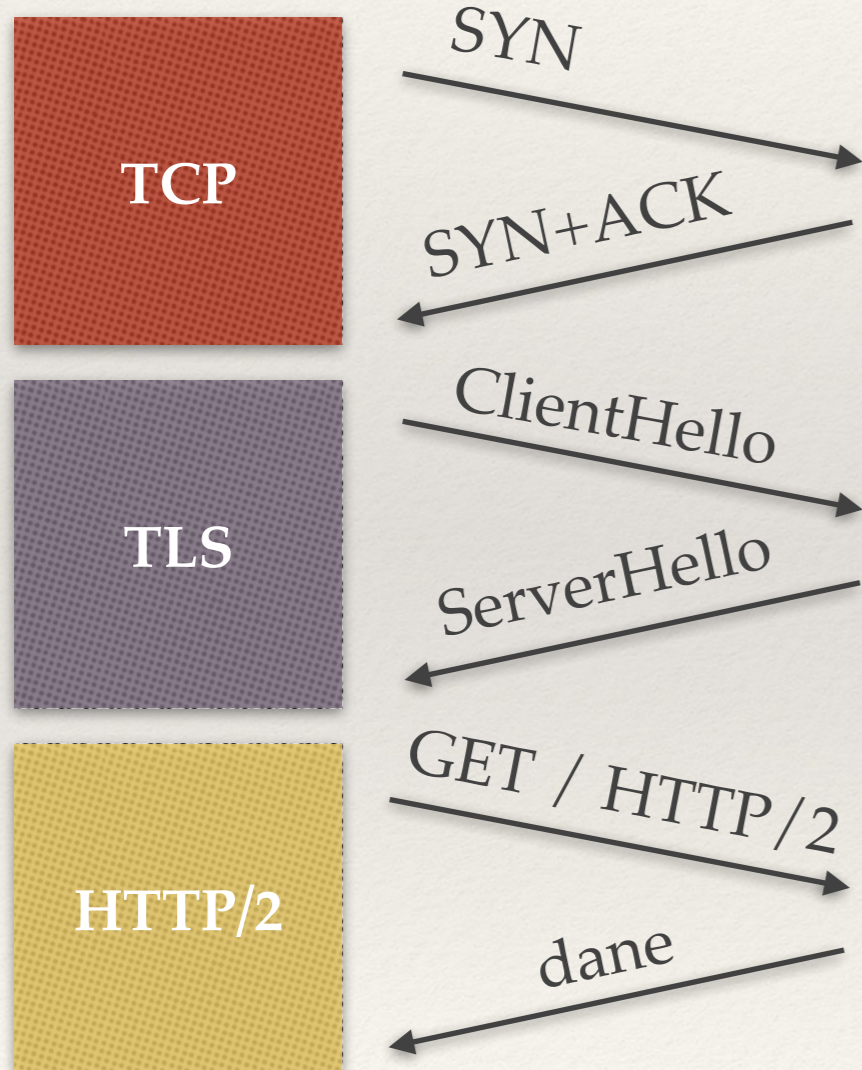
Dlaczego nie bezpośrednio na IP?

Bo wiele urządzeń zakłada, że w. transportowa to TCP lub UDP.

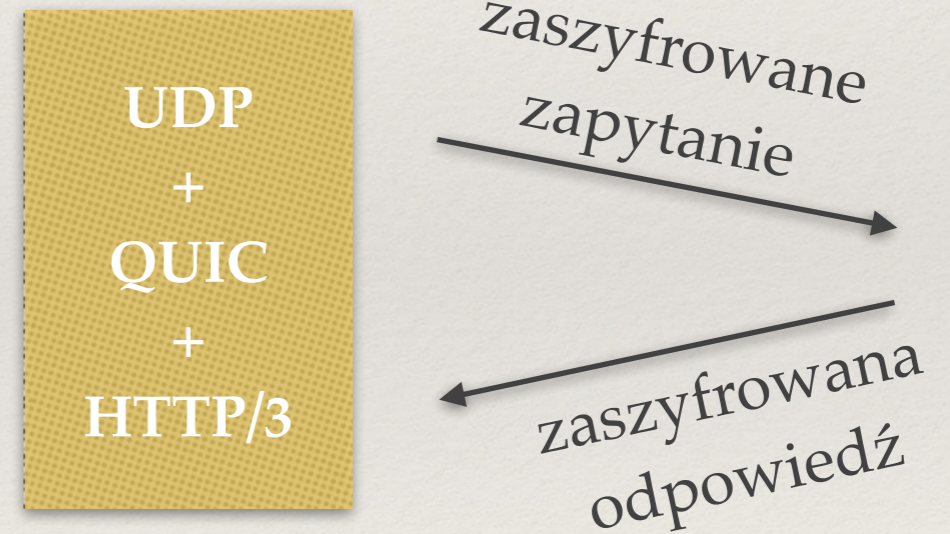
- ❖ Zintegrowane szyfrowanie TLS 1.3.
- ❖ Zaimplementowany w przestrzeni użytkownika (TCP jest w jądrze) → przewidywana szybsza ewolucja.
- ❖ $\text{HTTP/3} = \text{QUIC} + \text{HTTP/2}$.

Połączenia

TCP + TLS + HTTP/2



UDP + QUIC + HTTP/3



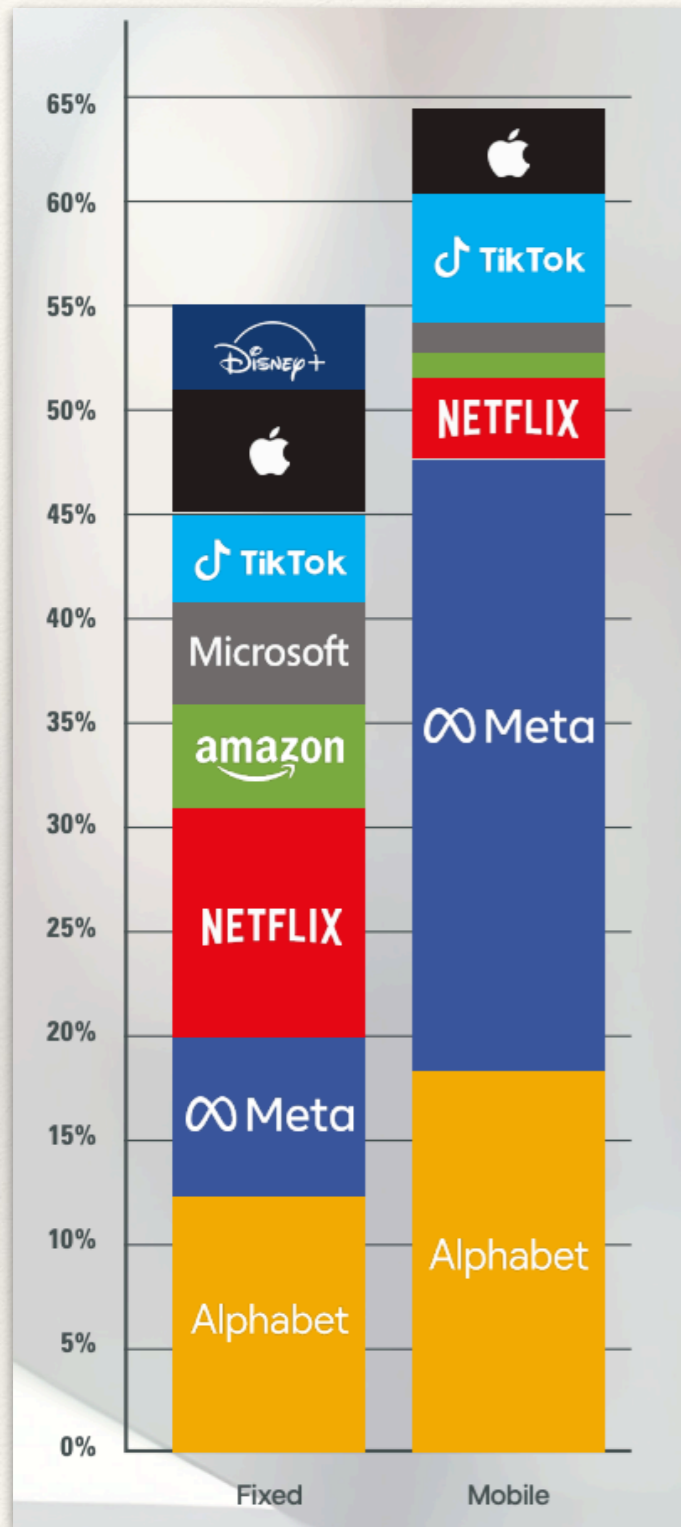
Wykorzystanie

Wykorzystanie jako procent całości ruchu HTTP (2025):

- ❖ HTTP / 1.1: ~35%
- ❖ HTTP / 2: ~50%
- ❖ HTTP / 3: ~15%

Jak doszło do tak szybkiej adopcji nowych wersji?

Udział treści poszczególnych firm (2025)



Nowszych wersji HTTP używają:

- ❖ powyższe firmy
- ❖ sieci CDN

Lektura dodatkowa

- ❖ Kurose & Ross: rozdział 2.
- ❖ Tanenbaum: rozdział 7.
- ❖ Dokumentacja HTTP:
 - ♦ 1.1: <https://datatracker.ietf.org/doc/html/rfc2616>
 - ♦ 2: <https://datatracker.ietf.org/doc/html/rfc9113>
 - ♦ 3: <https://www.rfc-editor.org/rfc/rfc9114.html>
- ❖ QUIC Tutorial: <https://www.youtube.com/watch?v=31J8PoLW9iM>
- ❖ Raport Cloudflare za 2025:
<https://radar.cloudflare.com/year-in-review/2025>

Zagadnienia

- ❖ Opisz budowę adresu URL. Opisz budowę adresu URL w przypadku schematu `http`.
- ❖ W jakim celu serwer WWW ustawia typ MIME dla wysyłanej zawartości? Podaj kilka przykładów typów MIME.
- ❖ Po co w nagłówku żądania `HTTP/1.1` podaje się pole `Host`?
- ❖ Do czego służą pola `Accept`, `Accept-Language`, `User-Agent`, `Referer`, `Server`, `Content-Length`, `Content-Type`, `Range` w nagłówku `HTTP`?
- ❖ Jak implementuje się przechowywanie stanu w komunikacji `HTTP`?
- ❖ Jak wygląda warunkowe zapytanie `GET` protokołu `HTTP`?
- ❖ Jakie znasz kody odpowiedzi protokołu `HTTP`?
- ❖ Na czym polegają połączenia trwałe w `HTTP/1.1`? Do czego służy opcja `Connection: close` w nagłówku `HTTP`?
- ❖ Po co stosuje się metodę `POST`?
- ❖ Co to jest technologia `REST`?
- ❖ Do czego służą serwery proxy?
- ❖ Co to jest odwrotne proxy? Co to jest `CDN`?
- ❖ Jak sprawić, żeby klient połączył się z serwerem proxy a nie bezpośrednio ze stroną `WWW`?
- ❖ Jakie informacje dołączane są przez serwer proxy do zapytania?
- ❖ Jakie usprawnienia wprowadza protokół `HTTP/2`?
- ❖ W jakim celu powstał protokół `QUIC`? Jakie funkcje spełnia?