
Warstwa aplikacji

część 1: HTTP

Sieci komputerowe

Wykład 9


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>

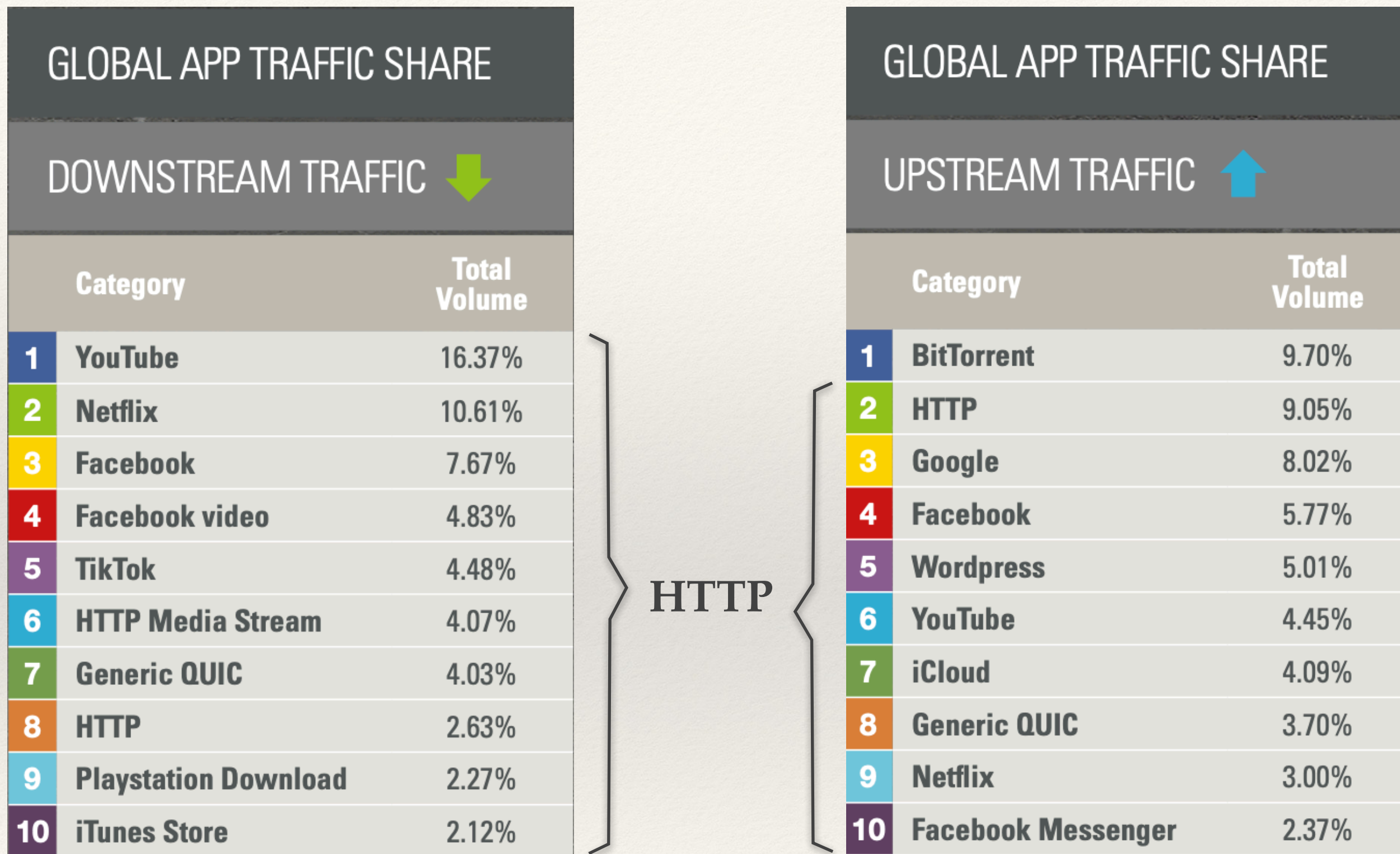
Trochę statystyk: aplikacje (2022)

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%

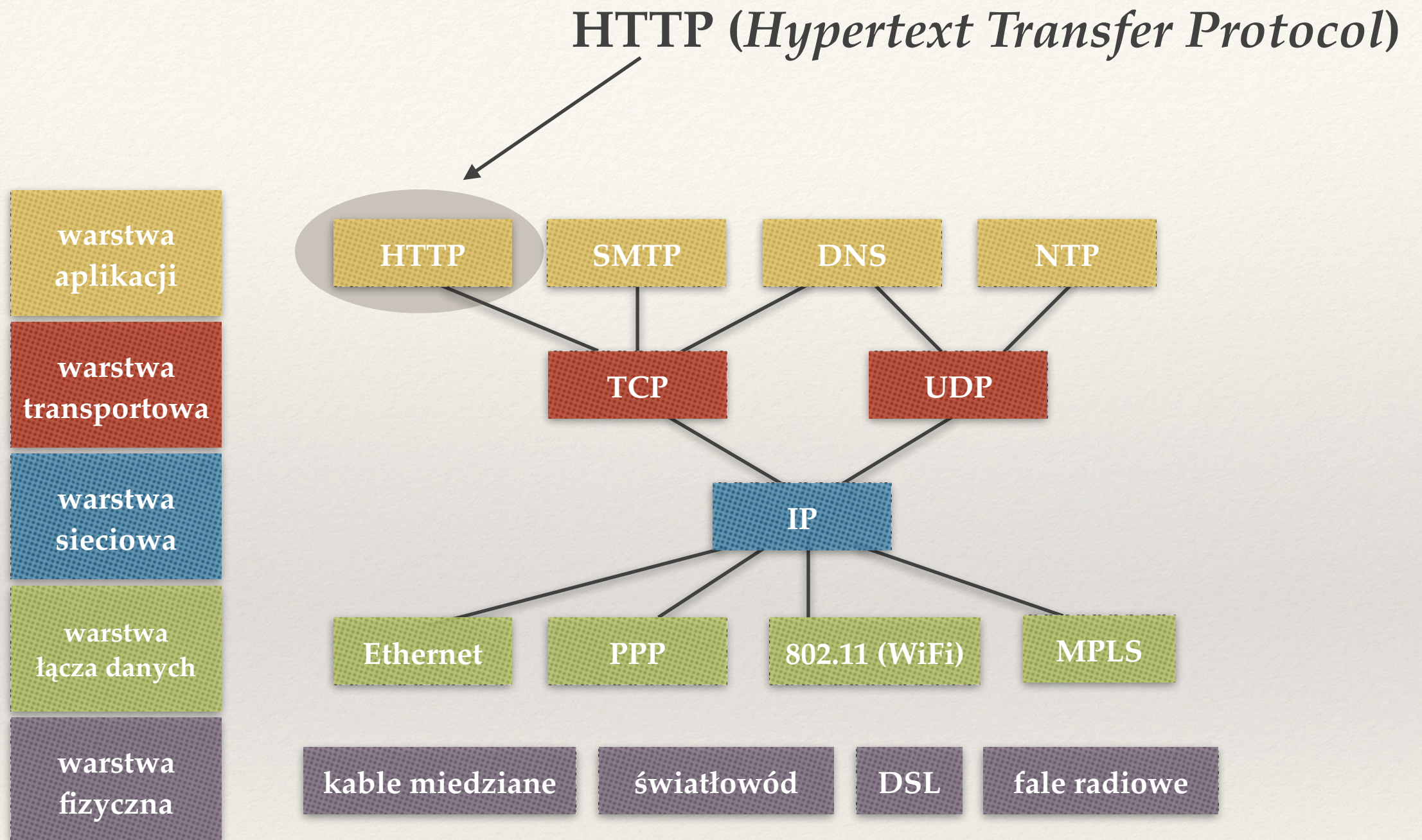
Obrazek ze raportu <https://www.sandvine.com/phenomena>

Trochę statystyk: aplikacje (2022)



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

Protokoły w Internecie



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*)

- ❖ Indentyfikuje 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: `http://en.wikipedia.org:443/wiki/HTTP/3`

Pobieranie strony WWW krok po kroku (1)

- ❖ Przeglądarka WWW dostaje URL
- ❖ URL jest rozbijany na człony.
- ❖ 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

Pobieranie strony WWW krok po kroku (2)

- ❖ Serwer analizuje żądanie, pobiera z dysku odpowiedni plik.
- ❖ Serwer sprawdza typ MIME 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

PLIK (w tym przypadku dokument HTML)

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

Odpowiedzi HTTP

Typy odpowiedzi:

- ❖ 1xx: informacyjne
- ❖ 2xx: sukces (200 = OK)
- ❖ 3xx: przekierowania
- ❖ 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:
 - 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.

Dynamika po stronie klienta WWW

- ❖ Javascript: prosty obiektowy interpretowany język zintegrowany z HTML.
 - ✦ Współcześnie wzbogacany przez różne biblioteki (React, Angular, Vue, ...)
- ❖ (Wycofywane) aplikacje Flash, Silverlight, aplety Javy (wykonanie realizowane przez odpowiednie wtyczki do przeglądarki).

Dynamika po stronie serwera WWW

URL może wskazywać na program generujący kod HTML.

- ❖ Popularne „frameworki” do tworzenia aplikacji po stronie serwera: Django, Flask (Python), Spring (Java), Laravel (PHP), Node.js (Javascript), Phoenix (Erlang), Ruby on Rails (Ruby), ...
- ❖ Dany program może komunikować się z serwerem WWW
 - ✦ za pośrednictwem IPC: standardy CGI / FastCGI (*Common Gateway Interface*)
 - ✦ za pomocą API: np. interfejs WSGI
- ❖ Formularze, przekazywanie parametrów (metody GET i POST).
- ❖ Cookies = utrzymywanie stanu sesji.

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 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.

Wydajne HTTP

Połączenia trwałe (1)

❖ do HTTP 1.0

- ♦ Każda para żądanie-odpowiedź w osobnym połączeniu TCP.
- ♦ Nawiązywanie połączenia TCP = duży narzut czasowy.
- ♦ Zazwyczaj przeglądarka pobiera wiele dokumentów naraz (np. strona WWW + obrazki).

❖ 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)

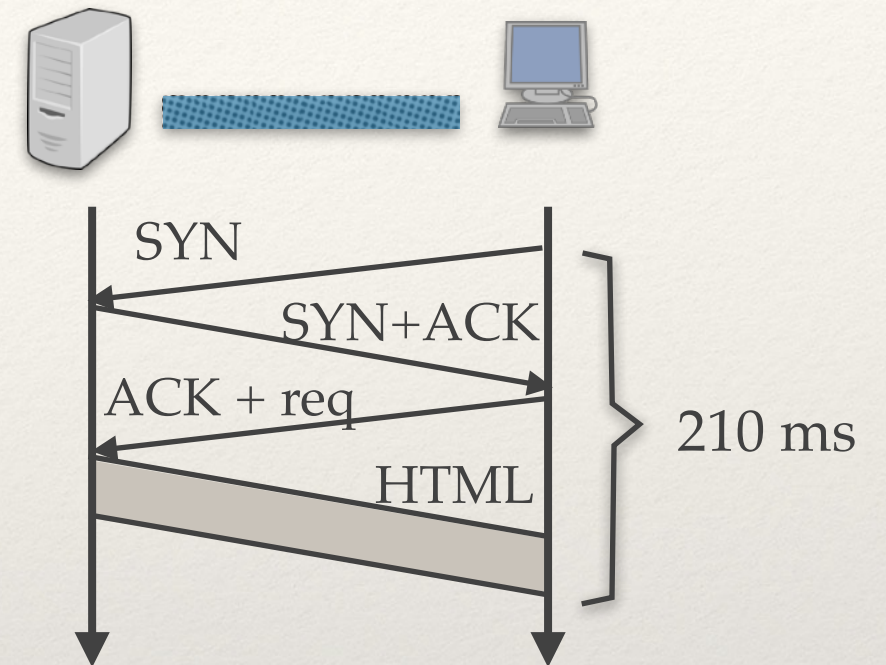
Przykład:

- ❖ 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).

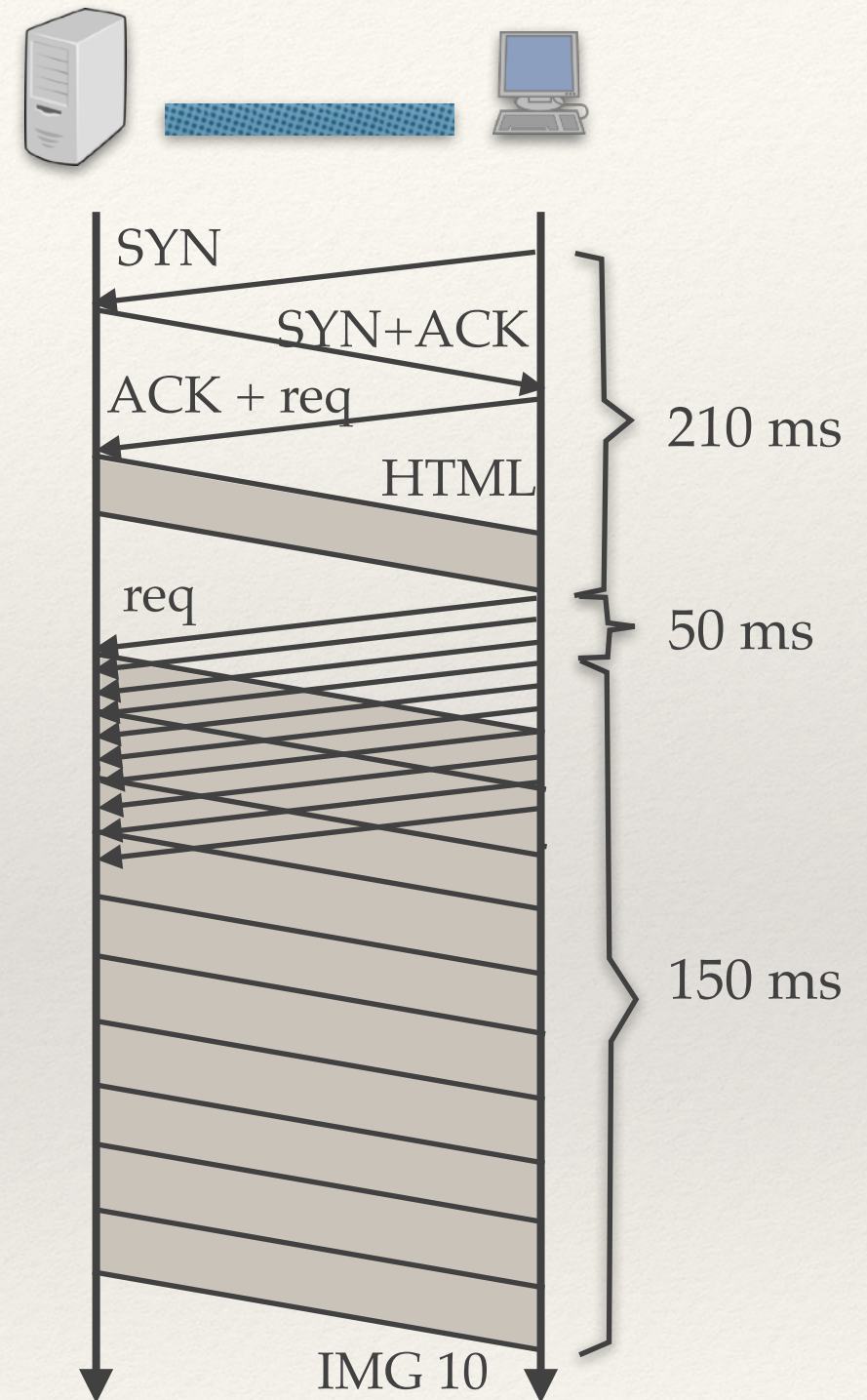
- ❖ Otrzymanie strony HTML: 210 ms.
- ❖ Pobieranie każdego z obrazków: kolejne: 210 ms.
- ❖ Usprawnienie: dwa równoległe połączenia do serwera → pobieranie 10 obrazków trwa $210 \text{ ms} * (10/2) = 1050 \text{ ms}$.
- ❖ Całkowity czas: $210 + 1050 = 1260 \text{ ms}$.



Połączenia trwałe (4)

HTTP/1.1 (połączenia trwałe).

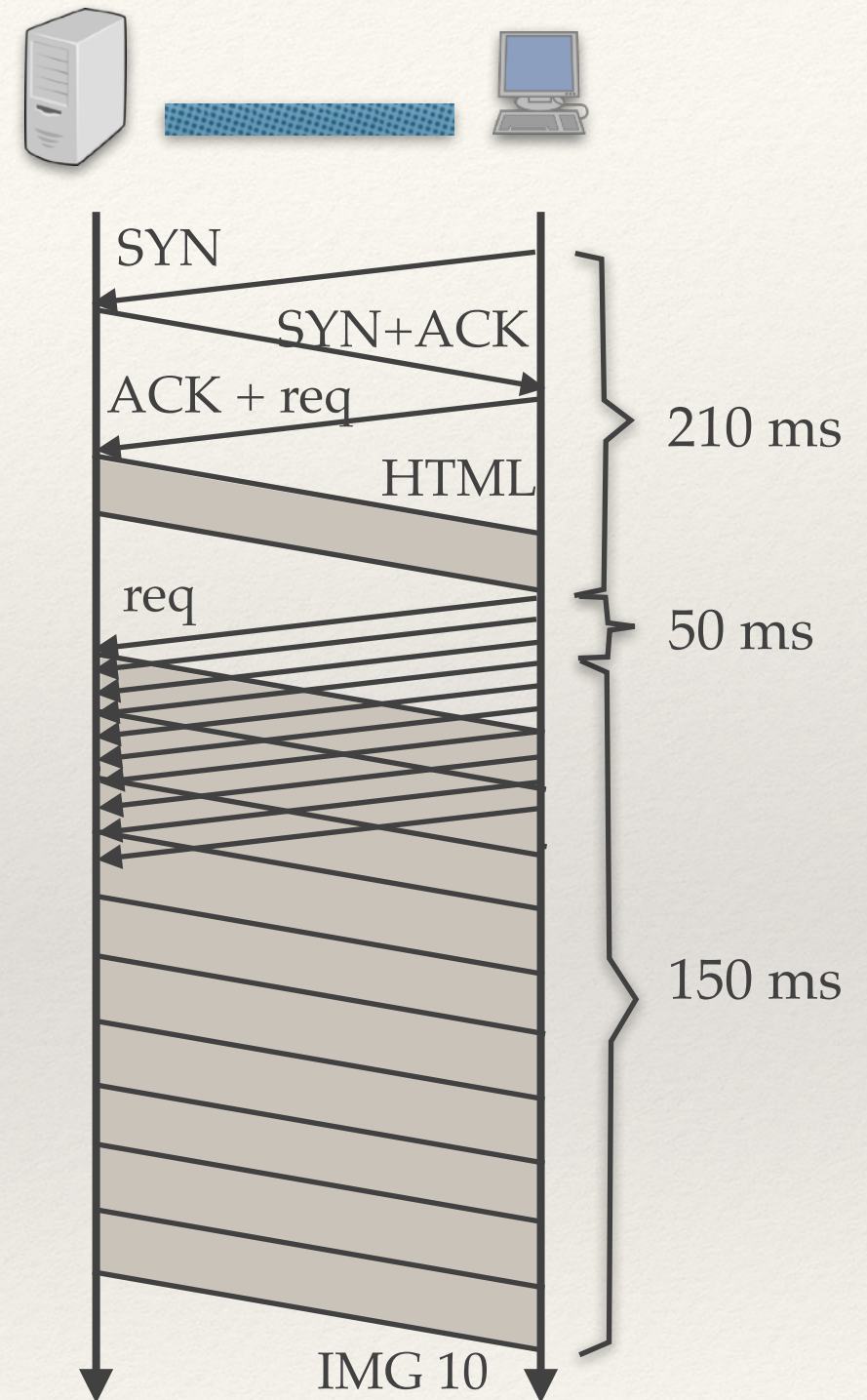
- ❖ Otrzymanie strony HTML: 210 ms.
- ❖ Wysłanie zapytania o obrazek nr 1: 50 ms.
- ❖ Wysyłanie obrazków: $50 + 10 * 10 = 150$ ms.
- ❖ Całkowity czas: $210 + 200 = 410$ ms.



Połączenia trwałe (4)

HTTP/1.1 (połączenia trwałe).

- ❖ Otrzymanie strony HTML: 210 ms.
 - ❖ Wysłanie zapytania o obrazek nr 1: 50 ms.
 - ❖ Wysyłanie obrazków: $50 + 10 * 10 = 150$ ms.
 - ❖ Całkowity czas: $210 + 200 = 410$ ms.
-
- ❖ Niepotrzebne dodatkowe połączenia TCP.
 - ❖ Jedno połączenie: okno TCP szybciej rośnie.



Zapytanie warunkowe GET

- ❖ W nagłówku podajemy:

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

- ❖ Możliwe odpowiedzi:

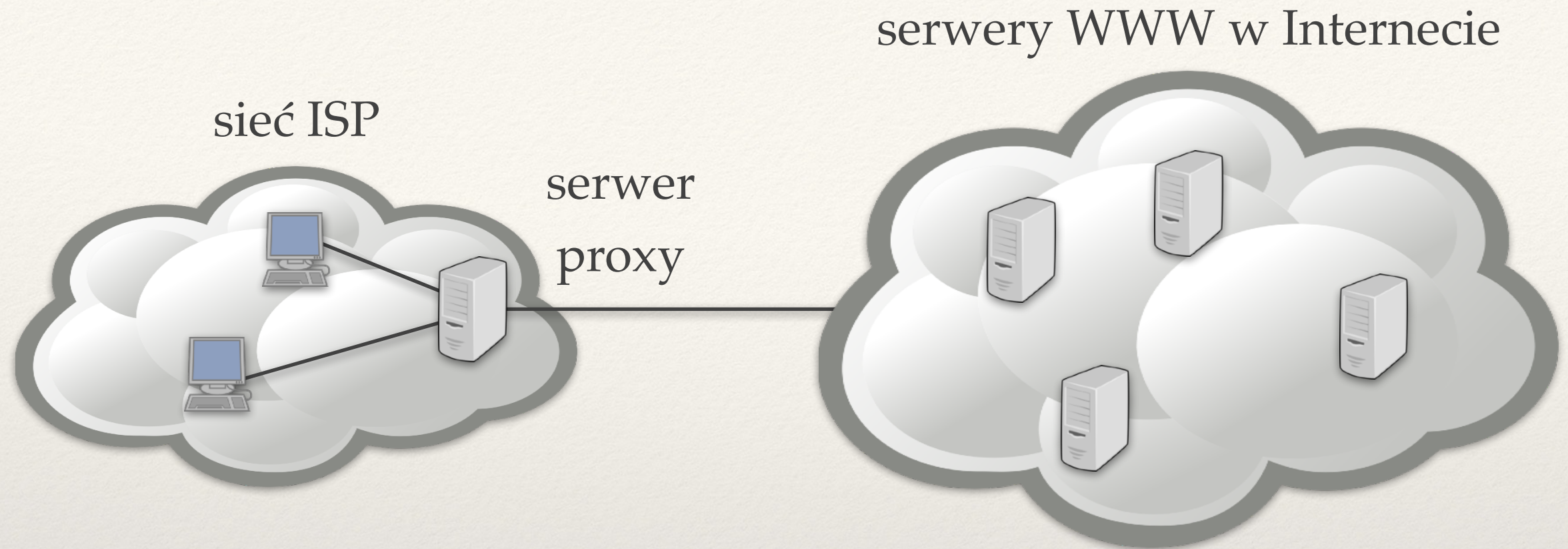
- ♦ 200 OK

- ♦ 304 Not Modified

Pamięć podręczna w przeglądarce WWW

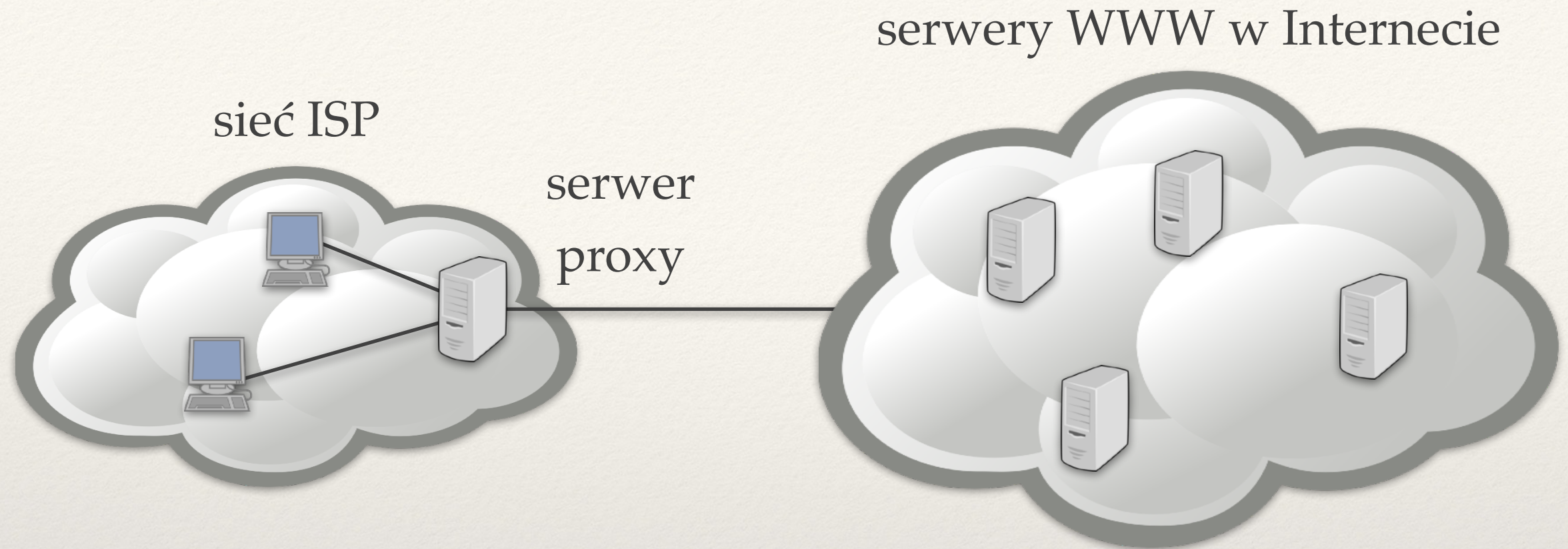
- ❖ Warunkowe zapytanie GET
- ❖ Serwer może umieszczać w nagłówku odpowiedzi pola:
 - ♦ `Expires:` (do kiedy można trzymać dokument w pamięci podręcznej) → można całkowicie pominąć żądanie strony.
 - ♦ `Cache-Control: no-cache` (nigdy nie trzymaj w pamięci podręcznej)

Serwery 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.

Serwery proxy (2)



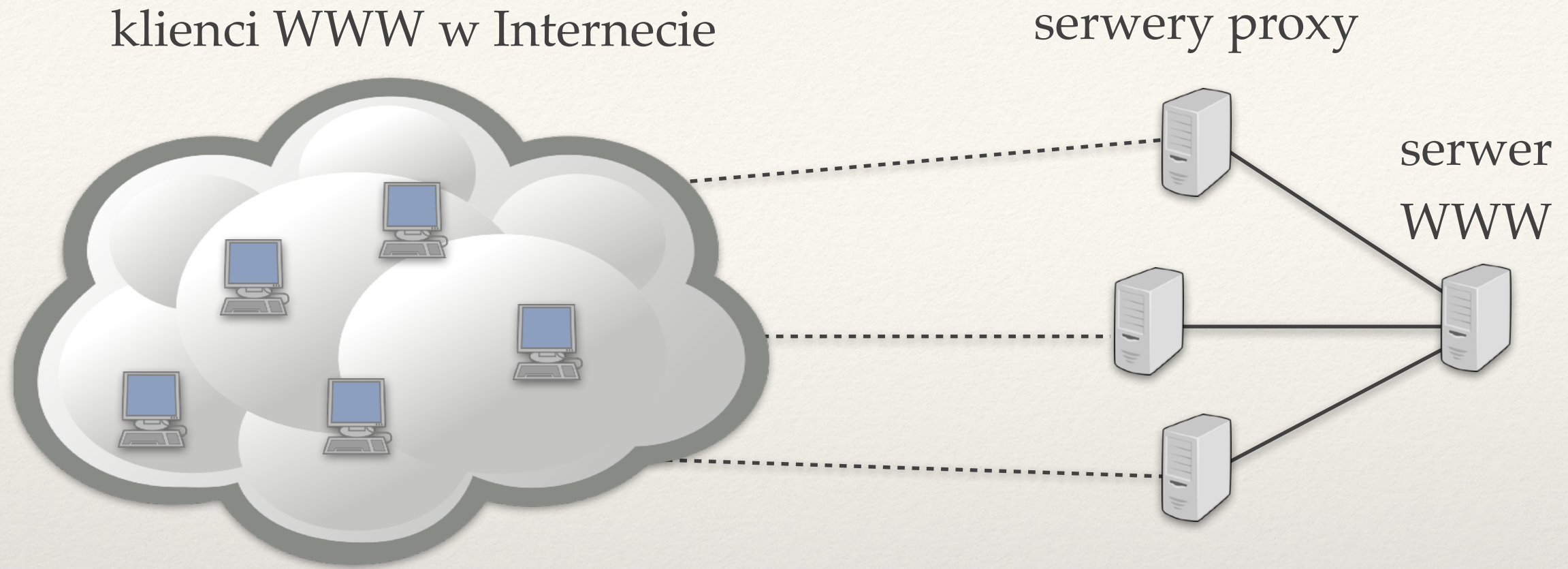
Serwer proxy

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

Dygresja: anonimizujące serwery proxy

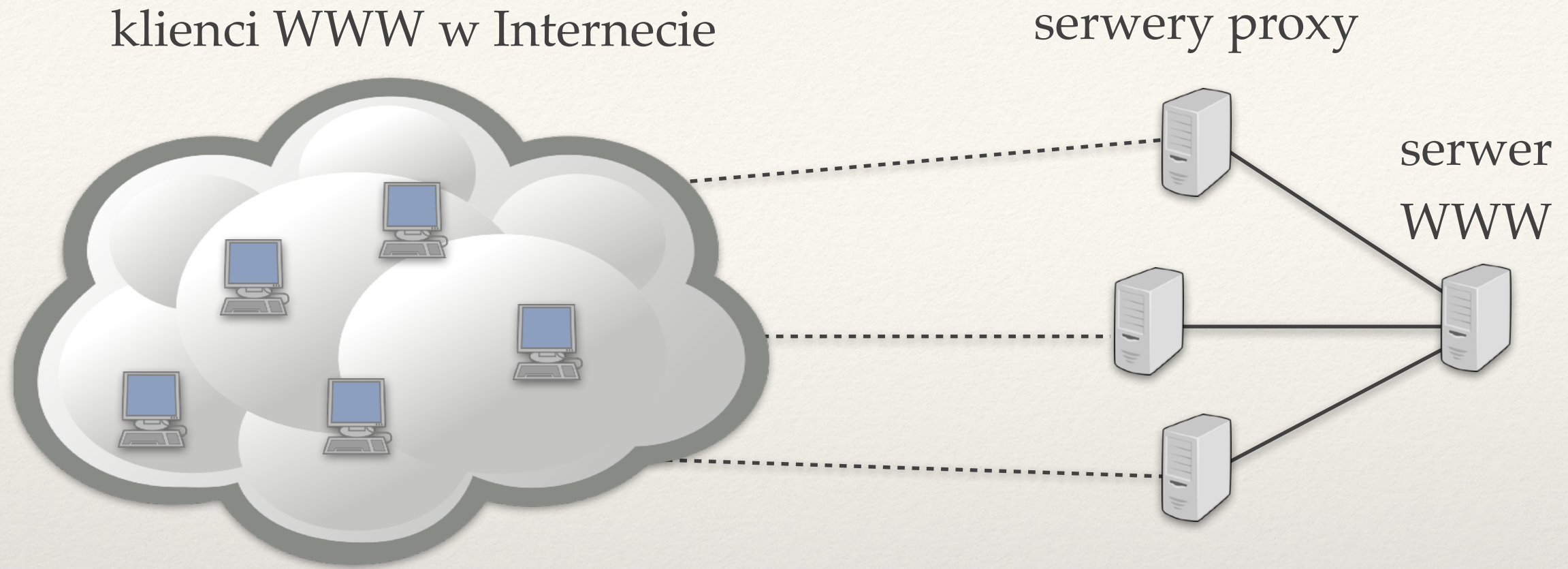
- ❖ Serwer proxy dodaje do żądania HTTP dodatkowe pola.
 - ♦ `X-Forwarded-For`: adres IP.
 - ♦ `Via`: adres IP proxy.
- ❖ Anonimizujące serwery proxy:
 - ♦ Nie dodają takich nagłówków.
 - ♦ Zwykle płatne.

Odwrotne proxy (1)



- ❖ Wykorzystywane przez dostawców treści.
- ❖ Zmniejszają obciążenie samego serwera WWW.
- ❖ Adresy IP serwerów proxy podawane zazwyczaj przez DNS jako adresy IP przy rozwiązywaniu nazwy serwera WWW.
- ♦ Serwery DNS zazwyczaj zwracają listę adresów IP w losowej albo cyklicznej kolejności (np.: `host -a google.com`)

Odwrotne proxy (2)



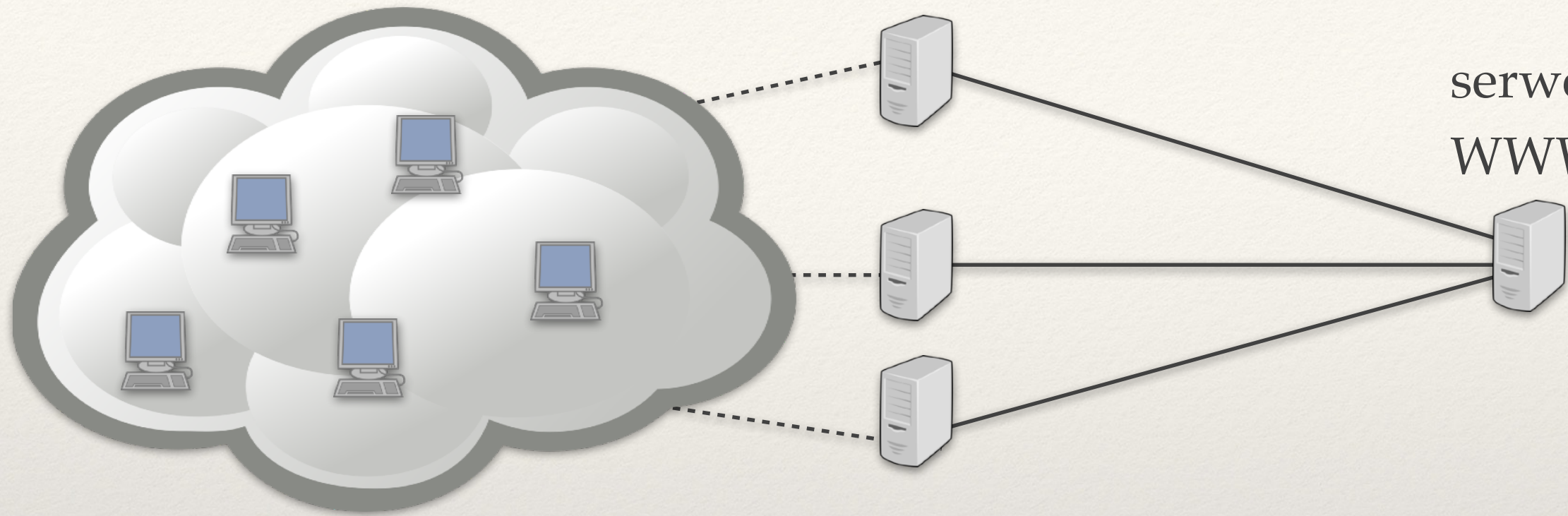
- ❖ Zysk dla klienta i dostawcy treści.
- ❖ Ale wciąż duże opóźnienie w przesyłaniu pakietów pomiędzy klientami i serwerami proxy.
- ❖ Jak opłacalnie przysunąć serwery proxy do klientów?

CDN (Content Distribution Networks)

klienci WWW w Internecie

serwery proxy CDN

serwer
WWW



- ❖ Serwery proxy obsługiwane przez osobną organizację (obsługuje wiele serwerów WWW).
 - ♦ Akamai, Limelight, ...
 - ♦ Setki tysięcy serwerów proxy.
- ❖ CDN utrzymuje również serwery DNS: umożliwiają wybieranie bliskiego serwera proxy.

HTTP jako dodatkowa 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, Flickr, ...)
 - ◆ 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 video

Dawniej specjalizowane protokoły, np. RTSP (kamery IP).

Współcześnie ruch video enkapsulowany jako dane protokołu HTTP.

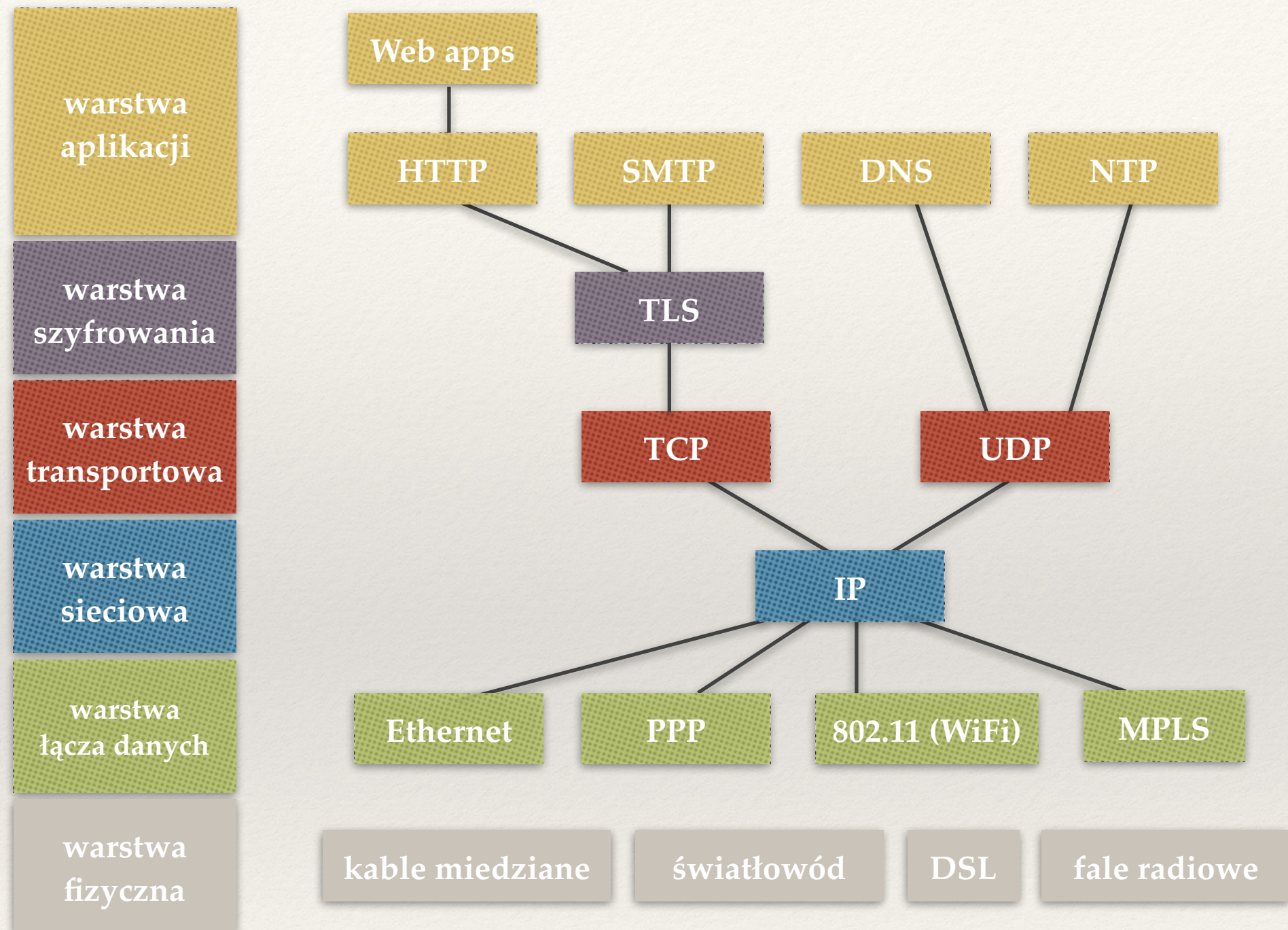
- ❖ DASH = Dynamic Adaptive Streaming over HTTP
 - ♦ Youtube, Netflix, HBO, Amazon Prime, ...
- ❖ HLS = HTTP Live Streaming
 - ♦ Twitch, HBO, Apple TV, ...

HTTP/2 i HTTP/3

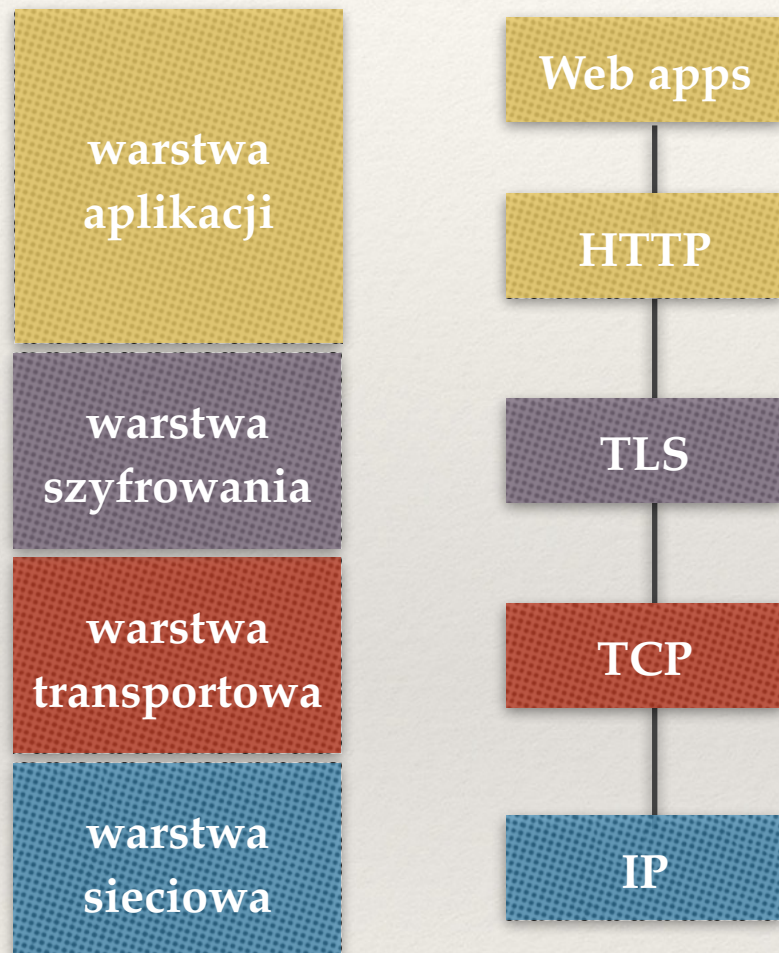
HTTP/2

- ❖ 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 1.2)

Protokoły w Internecie

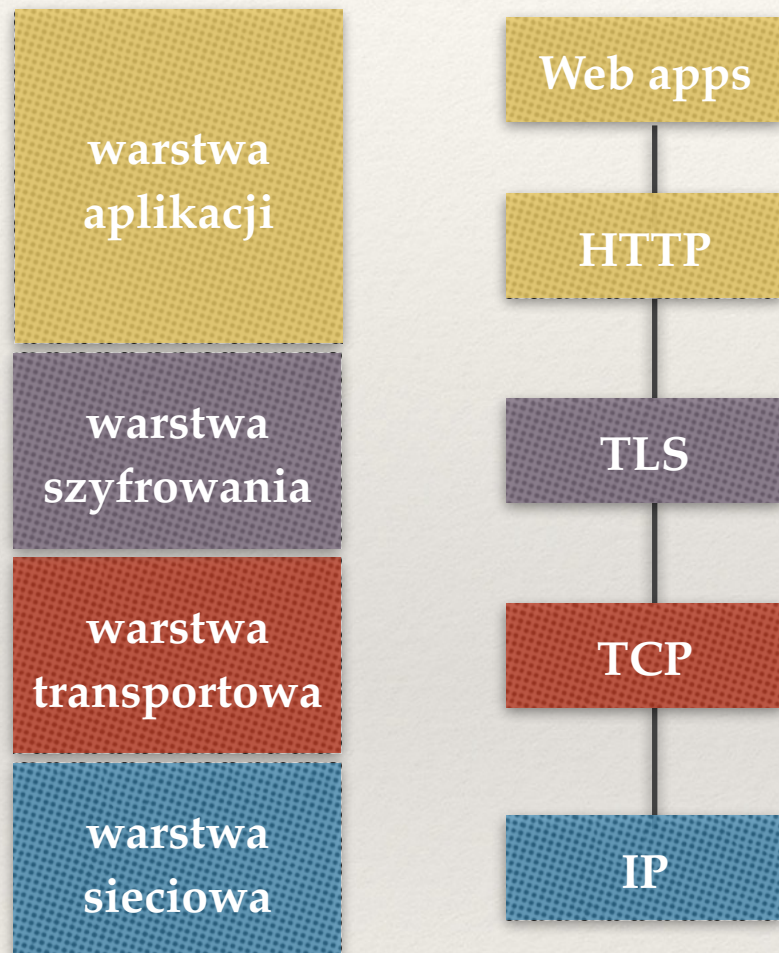


Protokoły w Internecie



Obecnie większość ruchu sieciowego
wykorzystuje kombinację
IP - TCP - TLS - HTTP

Model warstwowy

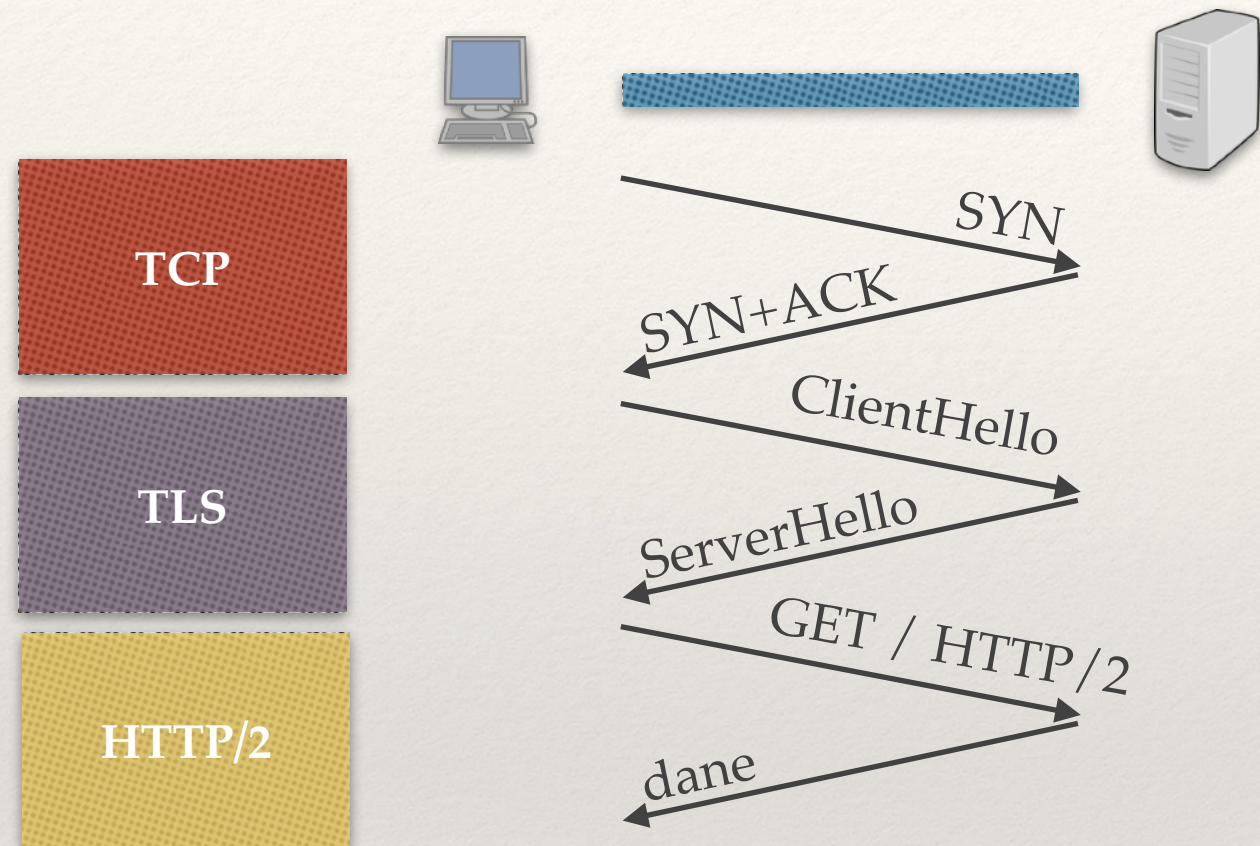


Dlaczego mamy model warstwowy?

- ❖ 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

Typowa sytuacja: TCP + TLS + HTTP/2

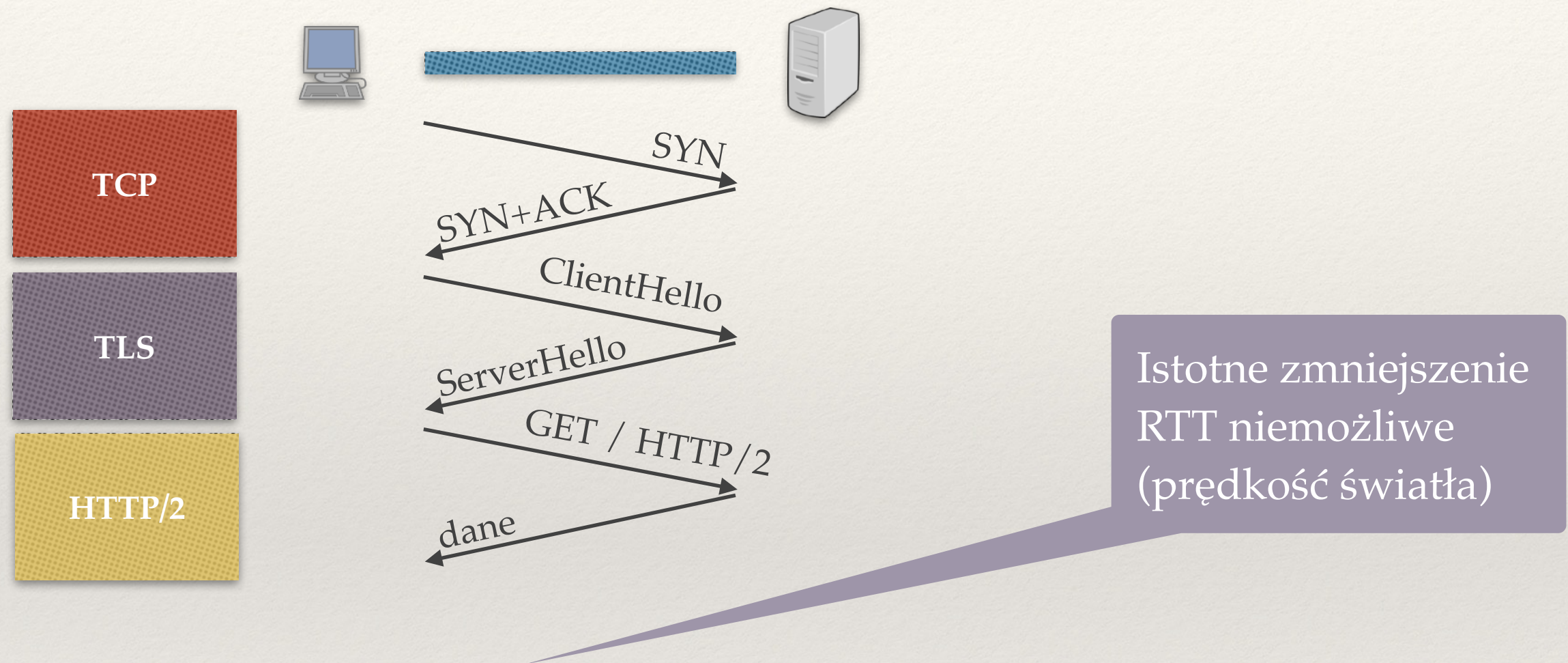


3 x RTT zanim klient zobaczy początek danych!

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

Problem #1: trzy nawiązywania połączenia

Typowa sytuacja: TCP + TLS + HTTP/2



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

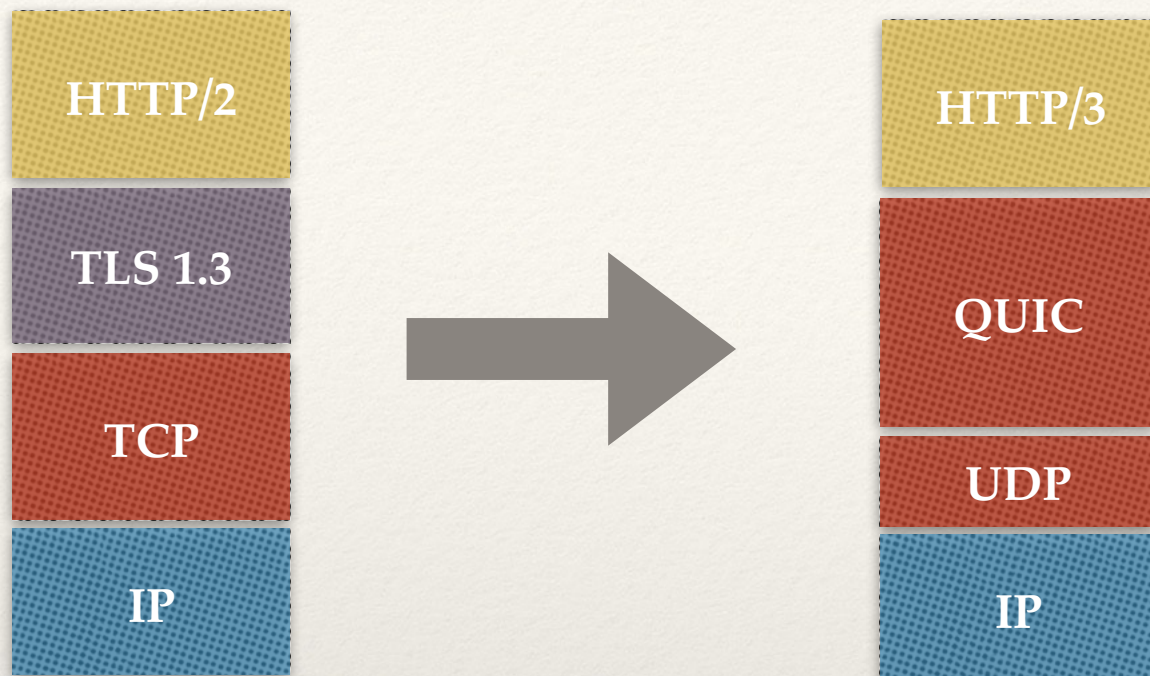
HTTP/2 wprowadza multiplexing zapytań i odpowiedzi:

- ❖ 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:

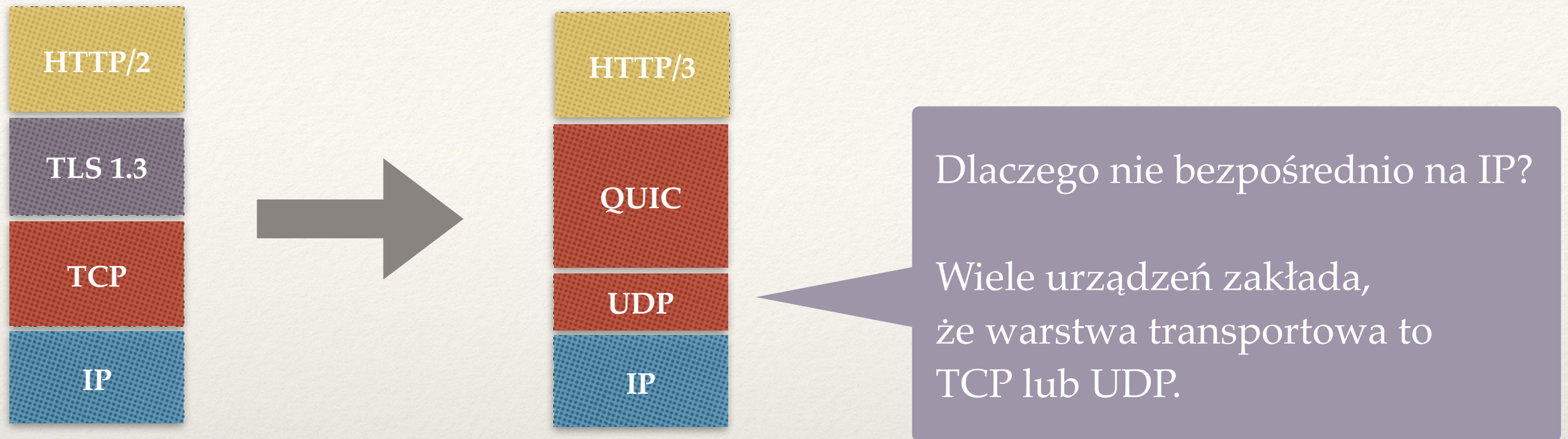
- ❖ Ginie segment zawierający mało istotny obrazek.
- ❖ Kolejne segmenty z innymi odpowiedziami HTTP docierają do klienta.
- ❖ Warstwa HTTP ich nie dostaje, bo TCP musi zrekonstruować strumień.

QUIC: nowy protokół transportowy



- ❖ Zintegrowany TLS 1.3.
- ❖ Własna kontrola przeciążenia (obecnie skopiowana z TCP).
- ❖ Zaimplementowany w przestrzeni użytkownika (TCP jest w jądrze)
- przewidywana szybsza ewolucja.
- ❖ HTTP/3 = HTTP/2 over QUIC.

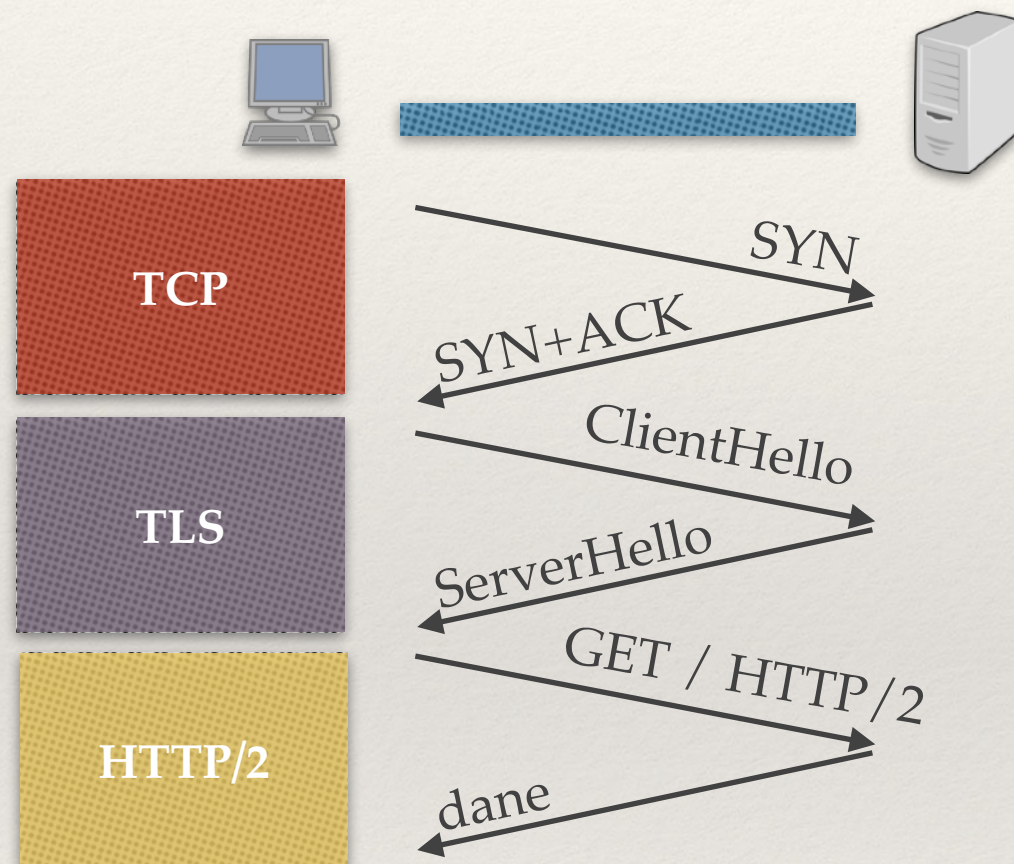
QUIC: nowy protokół transportowy



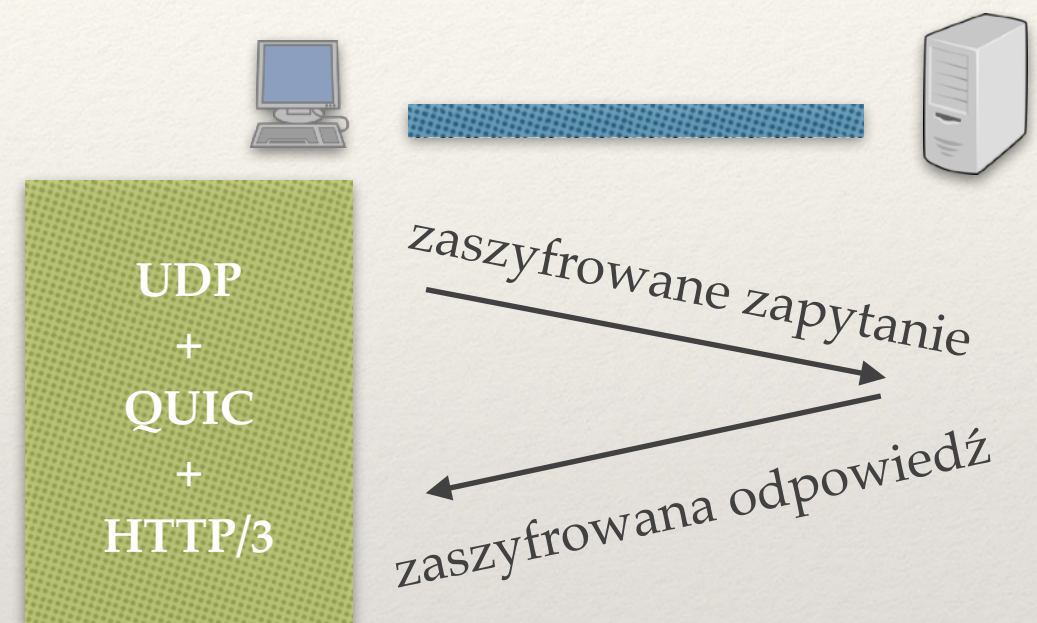
- ❖ Zintegrowany TLS 1.3.
- ❖ Własna kontrola przeciążenia (obecnie skopiowana z TCP).
- ❖ Zaimplementowany w przestrzeni użytkownika (TCP jest w jądrze) - przewidywana szybsza ewolucja.
- ❖ HTTP/3 = HTTP/2 over QUIC.

Połączenia

TCP + TLS + HTTP/2



UDP + QUIC + HTTP/3



Wykorzystanie

Wykorzystanie jako procent całości ruchu HTTP (2021)

- ❖ HTTP / 1.1: ~26%
- ❖ HTTP / 2: ~50%
- ❖ HTTP / 3: ~24%

Jak doszło do tak szybkiej adopcji nowego protokołu?

Kto “posiada” warstwę aplikacji? (2022)

Application Group		Total Volume
1	Google	20.99%
2	Facebook	15.39%
3	Netflix	9.39%
4	Apple	4.18%
5	Amazon	3.68%
6	Microsoft	3.32%
TOTAL		56.96%

Amazon

Amazon Prime, Twitch, Amazon.com, Alexa, Amazon Glacier, Amazon Music, Chime?

Apple

AppleTV+, iTunes, iCloud, Apple Software Update, FaceTime, Apple Music, Apple Watch (40m). Apple.com, iCloud Photo Stream, Mac App Store, Siri

Microsoft

Xbox Live, Windows Update, Skype, Outlook 365, Office 365, SharePoint, OneDrive, Windows Store, LinkedIn, Microsoft TEAMS

Netflix

Pure play video but moving into Mobile Gaming

Facebook (Meta)

Facebook, Instagram, Facebook Video, WhatsApp, Facebook Messenger, Oculus Rift

Google (Alphabet)

YouTube, Waze, Google Cloud, Google Play, Google Search, Google Docs, Google Drive, DoubleClick, Gmail, Google Home, Crashlytics, Nest, Looker, Fitbit

Lektura dodatkowa

- ❖ Kurose & Ross: rozdział 2.
- ❖ Tanenbaum: rozdział 7.
- ❖ HTTP/1.1: `https://tools.ietf.org/html/rfc2616`
- ❖ HTTP/2: `https://tools.ietf.org/html/rfc7540`
- ❖ HTTP/3: `https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34`

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.
- ❖ Wymień parę możliwych odpowiedzi HTTP wraz z ich znaczeniem.
- ❖ Po co w nagłówku żądania HTTP/1.1 podaje się pole `Host`?
- ❖ Do czego służą pola `Accept`, `Accept-Language`, `User-Agent`, `Server`, `Content-Length`, `Content-Type` w nagłówku 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?
- ❖ Do czego służą arkusze stylów CSS?
- ❖ Wymień parę możliwości uzyskiwania dynamicznych stron WWW.
- ❖ 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 skłonić klienta, żeby łączył się z serwerem proxy a nie bezpośrednio ze stroną WWW?
- ❖ Jakie informacje dołączane są przez serwer proxy do zapytania?
- ❖ Co to są anonimowe serwery proxy?
- ❖ W jakim celu powstał protokół QUIC? Jakie funkcje spełnia?