

Paweł Rajba

[pawel@cs.uni.wroc.pl](mailto:pawel@cs.uni.wroc.pl)

<http://pawel.ii.uni.wroc.pl/>

# Komunikacja w Internecie, protokół HTTP

# Agenda

- Komunikacja w Internecie, model warstwowy
- Adresowanie i przekazywanie pakietów, DNS
- Wprowadzenie do HTTP
- Adresy zasobów
- Rodzaje zawartości, Negocjacja treści
- Komunikacja
- Buforowanie
- HTTP Request/Response, Nagłówki
- Stan aplikacji i sesja, cookies

# Kluczowe instytucje Internetu

- Lista z Wikipedii: ([https://en.wikipedia.org/wiki/List\\_of\\_Internet\\_organizations](https://en.wikipedia.org/wiki/List_of_Internet_organizations))
  - IAB (Internet Architecture Board)
  - IANA (Internet Assigned Numbers Authority)
  - ICANN (Internet Corporation for Assigned Names and Numbers)
  - IESG (Internet Engineering Steering Group)
  - IETF (Internet Engineering Task Force)
  - IRTF (Internet Research Task Force)
  - ISOC (Internet Society)
  - NANOG (North American Network Operators' Group)
  - NRO (Number Resource Organization)
  - W3C (World Wide Web Consortium)
- Ciekawa prezentacja:
  - <https://www.slideshare.net/PeterREgli/internet-organization>
- Słowniczek od Google:
  - [https://support.google.com/domains/topic/3365481?hl=pl&ref\\_topic=6279308](https://support.google.com/domains/topic/3365481?hl=pl&ref_topic=6279308)

# Kluczowe instytucje Internetu

- Internet Corporation for Assigned Names and Numbers (ICANN)
  - Strona WWW: <https://www.icann.org/>
  - Ustalanie reguł, wg. których przydzielane są domeny i adresy IP
  - Nadzór nad działaniem serwerów DNS na całym świecie
  - Przyznawanie parametrów protokołom w Internecie
  - Zarządzanie i ustalanie listy domen najwyższego poziomu (TLD), np. .com, .org, ...
  - ICANN prowadzi dwie inne organizacje: InterNIC i IANA
    - InterNIC zajmuje się usługami rejestracji domen
      - WWW: <http://www.internic.net/>

# Kluczowe instytucje Internetu

- Internet Assigned Numbers Authority (IANA)
  - Strona WWW: <https://www.iana.org/>
  - Poprzednik ICANN, obecnie wsparcie dla ICANN na poziomie operacyjnym
  - Zarządzanie strefą root zone & TLD
  - Przydzielanie domen i adresów IP
  - Rejestr protokołów i katalogi uzupełniające
    - Np. MIME types

Domain Names	Number Resources	Protocol Assignments
Management of the DNS Root Zone (assignments of ccTLDs and gTLDs) along with other functions such as the .int and .arpa zones. <ul style="list-style-type: none"><li>■ Root Zone Management</li><li>■ Database of Top Level Domains</li><li>■ .int Registry</li><li>■ .arpa Registry</li><li>■ IDN Practices Repository</li></ul>	Coordination of the global IP and AS number spaces, such as allocations made to Regional Internet Registries. <ul style="list-style-type: none"><li>■ IP Addresses &amp; AS Numbers</li><li>■ Network abuse information</li></ul>	The central repository for protocol name and number registries used in many Internet protocols. <ul style="list-style-type: none"><li>■ Protocol Registries</li><li>■ Apply for an assignment</li><li>■ Time Zone Database</li></ul>

# Kluczowe instytucje Internetu

- The Internet Engineering Task Force (IETF)
  - Strona WWW: <https://www.ietf.org/>
  - Koordynacja przez WG prac nad standardami w Internecie (i inną dokumentacją techniczną)
    - Dokumenty RFC – Request for Comments
  - Powiązania do
    - The Internet Society (ISOC)
    - The Internet Architecture Board (IAB)
    - The Internet Research Task Force (IRTF)

# Kluczowe instytucje Internetu

- The Internet Society (ISOC)
  - Strona WWW: <https://www.internetsociety.org/>
  - „To provide leadership in Internet-related standards, education, access, and policy”
  - Wpływa na wiele innych organizacji, np. IETF, IANA

# Kluczowe instytucje Internetu

- Internet Architecture Board (IAB)
  - Strona WWW: <https://www.iab.org/>
  - Funkcja doradcza dla ISOC
  - „Architectural Oversight” nad standardami
  - Odpowiedzialni za edycja i publikację RFC
  - ... i wiele innych
    - [https://www.iab.org/wiki/index.php/IAB\\_Job\\_Description](https://www.iab.org/wiki/index.php/IAB_Job_Description)



# Komunikacja w Internecie

- Internet to globalna sieć
  - Sieć łącząca sieci (LAN, WAN, MAN)
- Mamy ogromną różnorodność urządzeń, które się komunikują
  - Komputery, telefony, telewizory, komunikatory, sensory, lodówki, kamery, pralki, etc.
- ... i różnorodność technologii
  - Kable, WiFi, 3G/4G/4G LTE/5G, światłowody, satelity, bluetooth, NFC, RFID
- Obecne trendy
  - IoT, Cloud computing, XaaS (wszystko jako usługa)

# Komunikacja w Internecie

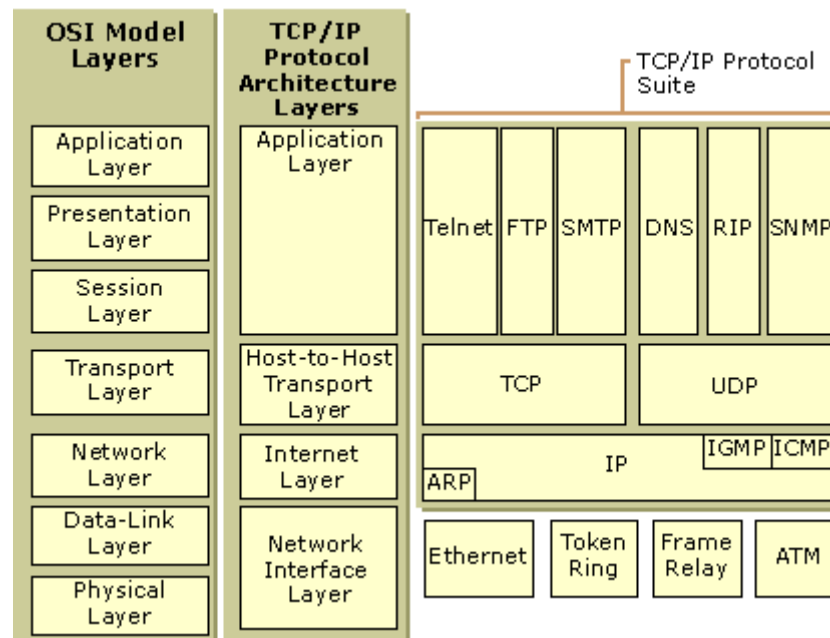
- Aby w przeglądarce pojawił się serwis WWW, mamy kilka kluczowych składowych:
  - Model warstwowy sieci ISO/OSI lub TCP/IP
  - Adresowanie
  - Przekazywanie pakietów (routing)
  - DNS
  - Serwer WWW

# Model warstwowy


- Każda warstwa
  - Ma określony zakres zadań
  - Komunikuje się tylko z warstwami sąsiednimi
- Taki model ułatwia
  - Implementację
    - Każdą warstwę można rozpatrywać niezależnie
  - Wyłapywanie błędów
  - Enkapsulację/dekapsulację przesyłanych danych
    - Każda warstwa określa własne rozszerzenia

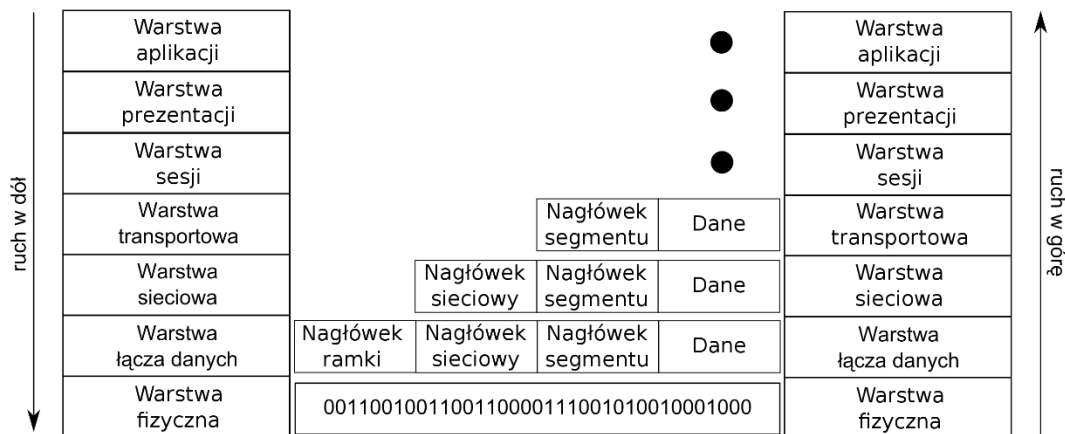
# Model warstwowy

- Model ISO/OSI i TCP/IP



# Model warstwowy

- Enkapsulacja/dekapsulacja
  - Nomenklatura
    - Transportowa
      - TCP: Segmenty
      - UDP: Datagramy
    - Sieciowa
      - Pakiety
    - Łączy danych
      - Ramki
- 
- The diagram illustrates the seven layers of the OSI model, arranged vertically from top to bottom. A vertical arrow on the left points downwards, labeled 'ruch w dół' (downward movement). The layers are: Warstwa aplikacji, Warstwa prezentacji, Warstwa sesji, Warstwa transportowa, Warstwa sieciowa, Warstwa łącza danych, and Warstwa fizyczna. To the right of the layers, there are two columns of text: 'Nazwa' (Name) and 'Format danych' (Data format). The 'Nazwa' column contains the names of the layers, and the 'Format danych' column contains the corresponding data formats: 'Dane' (Data) for the top three layers, 'Segment' (Segment) for the transport layer, 'Pakiet' (Packet) for the network layer, 'Rama' (Frame) for the data link layer, and 'Bity' (Bits) for the physical layer.
- |   | Nazwa                | Format danych |
|---|----------------------|---------------|
| 1 | Warstwa aplikacji    | Dane          |
| 2 | Warstwa prezentacji  | Dane          |
| 3 | Warstwa sesji        | Dane          |
| 4 | Warstwa transportowa | Segment       |
| 5 | Warstwa sieciowa     | Pakiet        |
| 6 | Warstwa łącza danych | Rama          |
| 7 | Warstwa fizyczna     | Bity          |



# Adresowanie

- Podstawą adresowania globalnego jest adres IP
  - Występuje w wersjach 4 i 6 (dalej skupimy się na v4)
- Adres to 32 bity podzielony na 4 grupy
  - Np. 124.112.3.4
- Adres dzielimy na część sieci i hosta
  - Np. 124.112.3.4
    - 124.112: część/ID sieci
    - 3.4: część/ID hosta
- Podział jest określony przez maskę, która określa liczbę bitów dla ID sieci

# Adresowanie

- Klasy adresów
  - A: maska 8 bitów
  - B: maska 16 bitów
  - C: maska 24 bity
- Zapisy adresów
  - 131.112.2.3/255.255.255.0
  - 131.112.2.3/24 (CIDR)

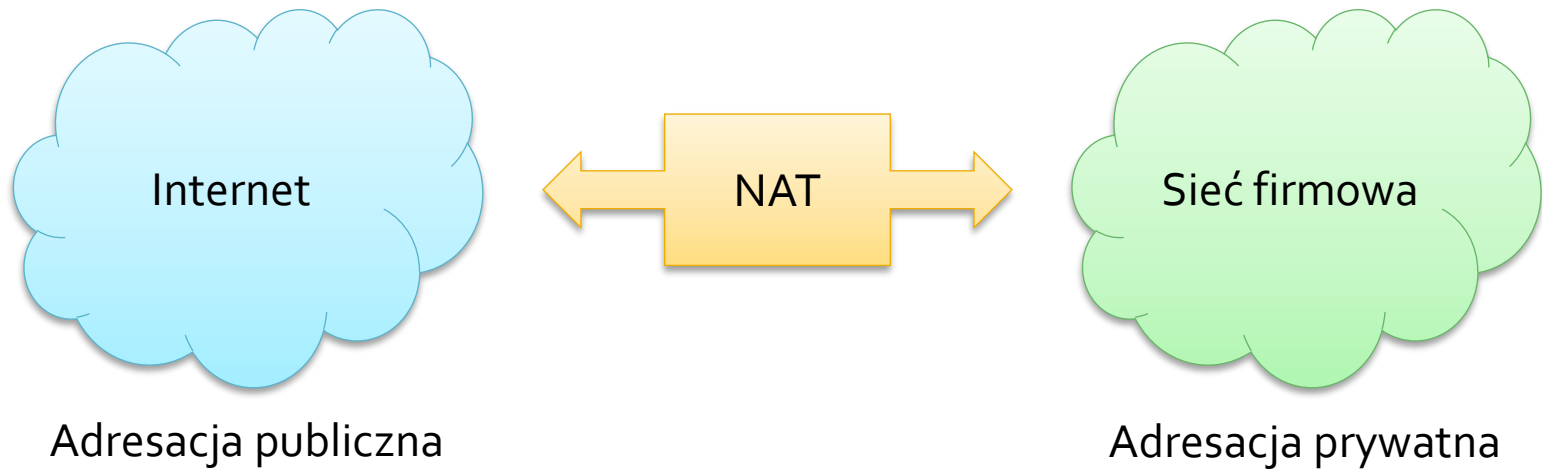
# Adresowanie

- Adresy publiczne i prywatne
  - Niektóre adresy są zarezerwowane do adresacji lokalnej w firmach
  - Każda klasa adresów ma odpowiedni zakres
    - A: 10.0.0.0/8
    - B: 172.16.0.0/16
    - C: 192.168.X.0/24
  - Pojęcie NAT



# Adresowanie

- Adresy publiczne i prywatne

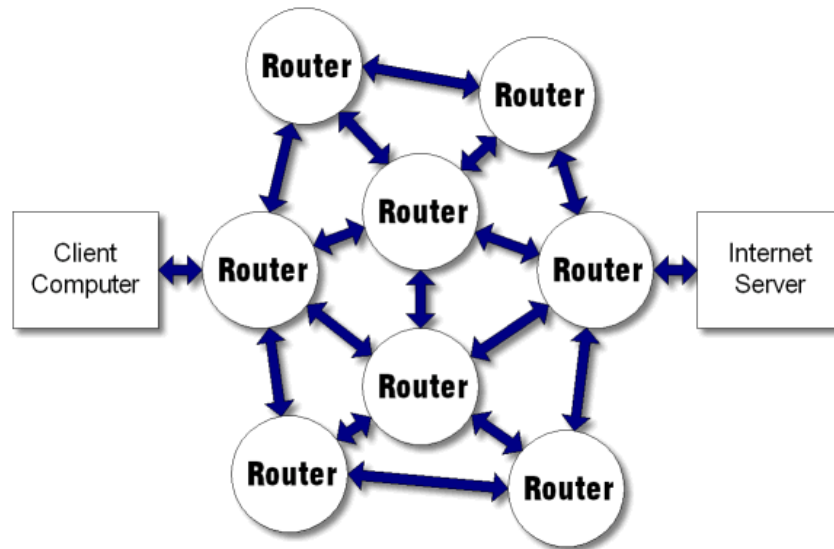


# Adresowanie

- Oprócz adresu IP mamy również porty
  - Definiowane w warstwie transportowej
  - Zakres 0-65535
  - Wiele usług ma predefiniowane porty domyślne
    - Ale można je zmienić, np. RDP przez 443 (dlaczego?)
- Adresowanie lokalne
  - Adres 127.0.0.1
  - Adresy MAC (np. 00:0A:E6:3E:FD:E1)
  - Protokoły ARP i RARP
  - Więcej: [https://pl.wikipedia.org/wiki/Adres\\_MAC](https://pl.wikipedia.org/wiki/Adres_MAC)

# Przekazywanie pakietów

- Najpierw dostarczanie lokalne
- Potem tablica routingu
- ... a potem brama domyślna i w świat



# DNS

- Ludziom łatwiej zapamiętać nazwę `www.wp.pl`, niż adres `212.77.98.9`
- A komputerom łatwiej operować na adresie `212.77.98.9` niż poprzez nazwę `www.wp.pl`
- Zamiana nazwy na adres to
  - forward resolution,
- a zamiana adresu na nazwę to
  - reverse resolution

# DNS

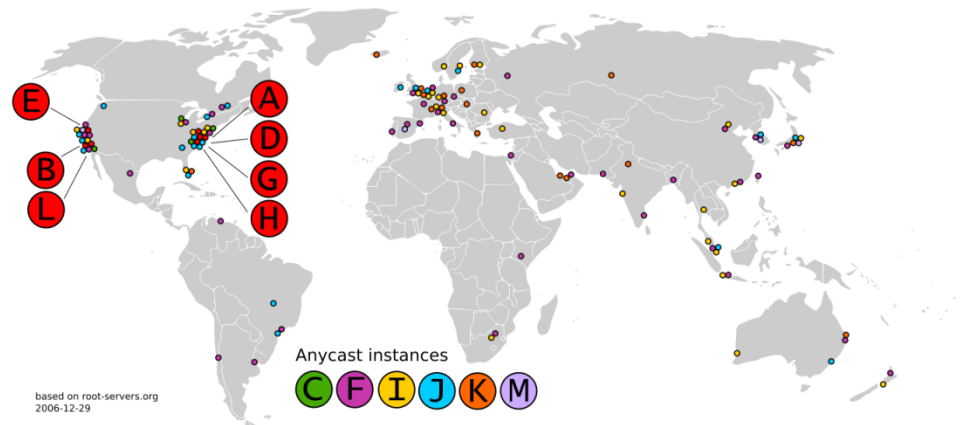
- Na początku był centralny plik hosts.txt, który każdy mógł lokalnie skopiować
- Z czasem plik robił się coraz większy i aktualizacja stawała się kłopotliwa
- ... i większy ... i jeszcze większy ...
- ... aż w końcu w 1984 r. powstał DNS
- Główny cel: zamiana nazw na adresy IP

# DNS

- Działa w modelu klient-serwer
- Rozproszona, odporna na błędy, baza danych
- Struktura drzewa
  - W korzeniu jest root domain, empty-string
  - Poziom 1 to tzw. TLD (top level domains)
    - W lipcu 2015 było 1058 TLDs
  - Drzewo podzielone jest na strefy określające odpowiedzialność za dany fragment drzewa

# DNS

- Dla root domain jest 13 logicznych serwerów
  - A.root-servers.net
  - B.root-servers.net
  - C.root-servers.net
  - D.root-servers.net
  - ...
- Każdy system ma te adresy zapisane



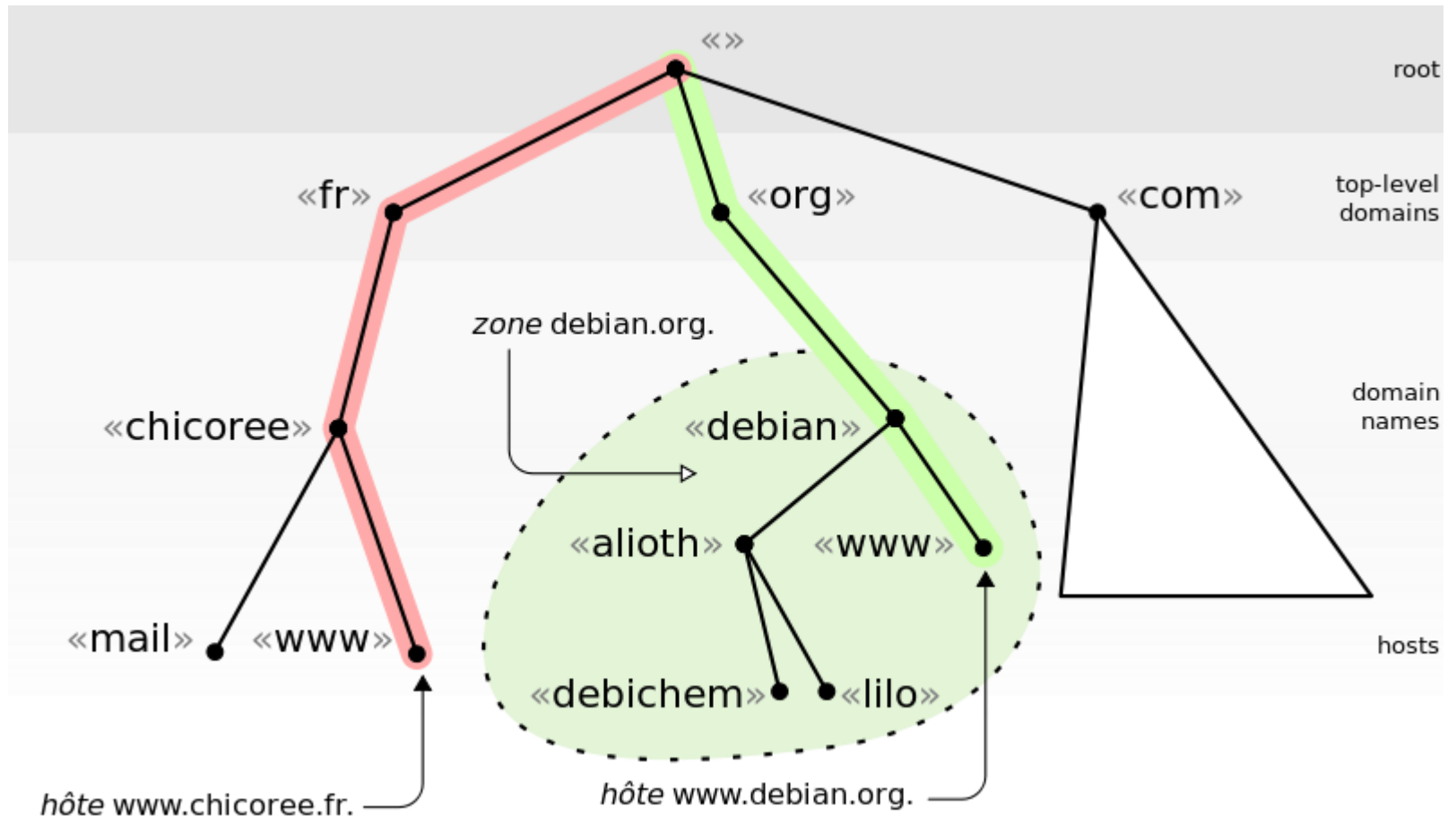
List of Root Servers

HOSTNAME	IP ADDRESSES	MANAGER
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California (ISI)
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	VeriSign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

<https://www.iana.org/domains/root/servers>

[https://commons.wikimedia.org/wiki/File:DNS\\_Tree.svg](https://commons.wikimedia.org/wiki/File:DNS_Tree.svg)

# DNS





# DNS

- Rodzaje rozwiązywania nazw
  - **Iteracyjne:** klient odpytuje serwery z kolejnych poziomów drzewa DNS zaczynając od roota
  - **Rekurencyjne:** klient pyta wybrany serwer DNS, a on wykonuje odpytywanie i dostarcza kompletnej odpowiedzi

# DNS

## ■ TTL

- Każdy rekord ma zdefiniowany time-to-live, po którym powinno nastąpić odświeżenie danych
- Duży TTL zmniejsza liczbę zapytań
- Mały TTL daje szybszą propagacją zmian

# DNS

- Rodzaje rekordów
  - A, AAAA – mapowanie nazwy hosta na IPv4 lub v6
  - NS – lista serwerów DNS dla danej strefy
  - CNAME – alias do nazwy
  - MX – lista serwerów SMTP dla strefy

# DNS

- Domena odwrotna
  - Zamiana adresu IP na nazwy
  - Zdefiniowane za pomocą rekordów PTR
  - Wykorzystywana „sztuczna” domena in-addr.arpa
- Przykład
  - Adres: 8.8.4.4
  - Rekord PTR: 4.4.8.8.in-addr.arpa  
... wskazuje na google-public-dns-b.google.com

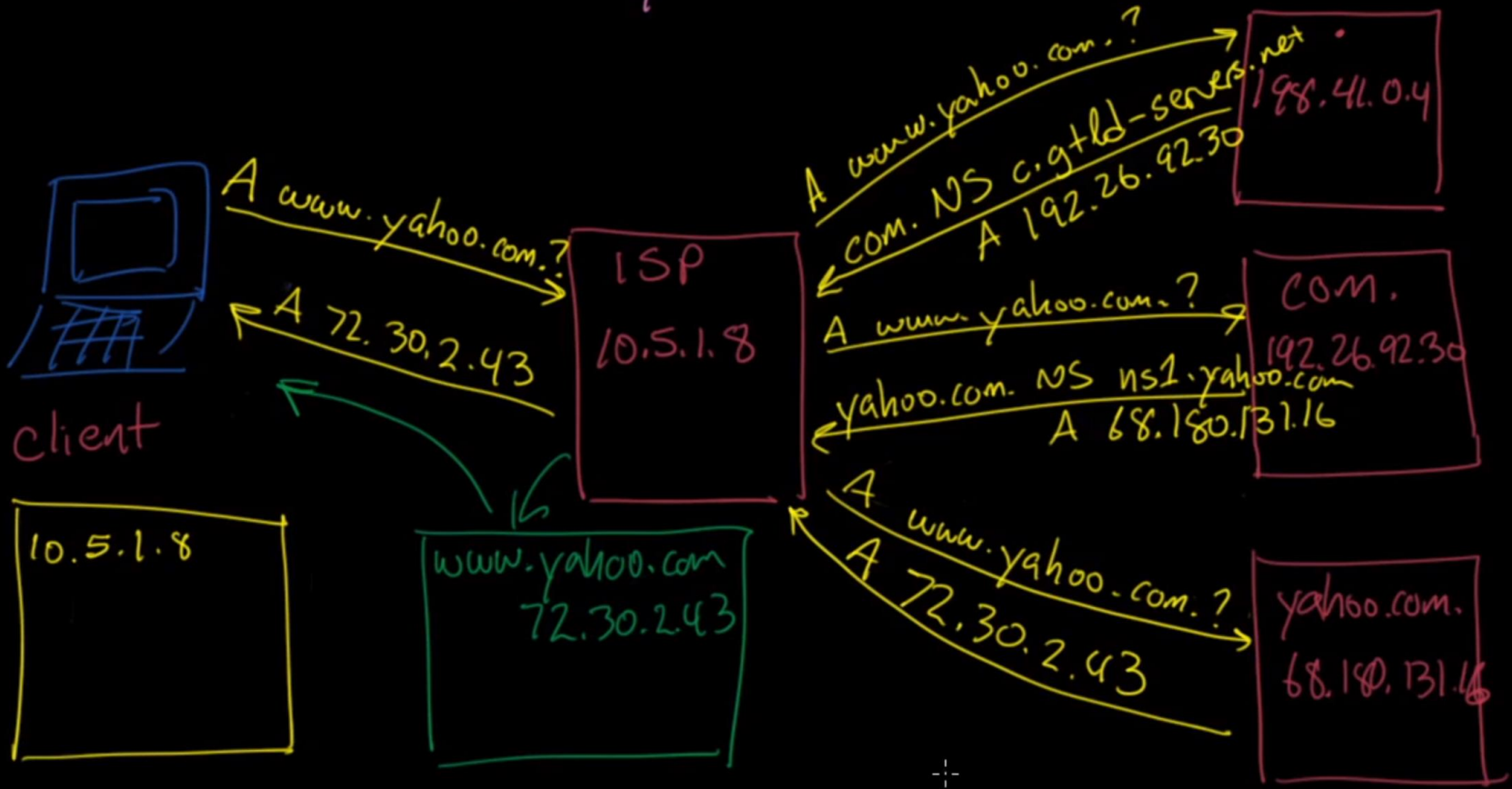
# DNS

DNS Resolution, Step by Step

## DNS

## Resolving Host to IP Address

www.yahoo.com → 72.30.2.43



# DNS

- Narzędzia

- nslookup

- <https://www.whois.com/whois/>

- <https://mxtoolbox.com/DNSLookup.aspx>

# Historia HTTP

- Początki w roku 1989
  - Wraz z pojawieniem się WWW
- Wersje
  - Pierwsza oficjalna wersja to HTTP 0.9 (1991)
  - HTTP 1.0 (1996), HTTP 1.1 (1997), HTTP 2 (2015)
    - W 2014 pojawiły nowe specyfikacje wersji 1.1
- Protokół tekstowy w modelu klient-serwer
- Stworzony z myślą o przesyłaniu hipertekstu
  - Czyli zbioru stron połączonych odnośnikami
- Usługa HTTP zwykle dostępna na porcie 80
  - Wersja zaszyfrowana: port 443

# HTTP 2.0

- HTTP/2
  - Oparty o protokół Google'a SPDY (Speedy)
  - Główne zmiany w stosunku do wersji 1.1
    - Protokół binarny
    - Przesyłania wielu zasobów jednocześnie bez otwierania nowych połączeń (multiplexing)
    - Kompresja nagłówków
    - Mechanizm „push” dla serwera
- Test zgodności z HTTP/2
  - <https://tools.keycdn.com/http2-test>
- Demo
  - <https://http2.akamai.com/demo>



# HTTP 2.0

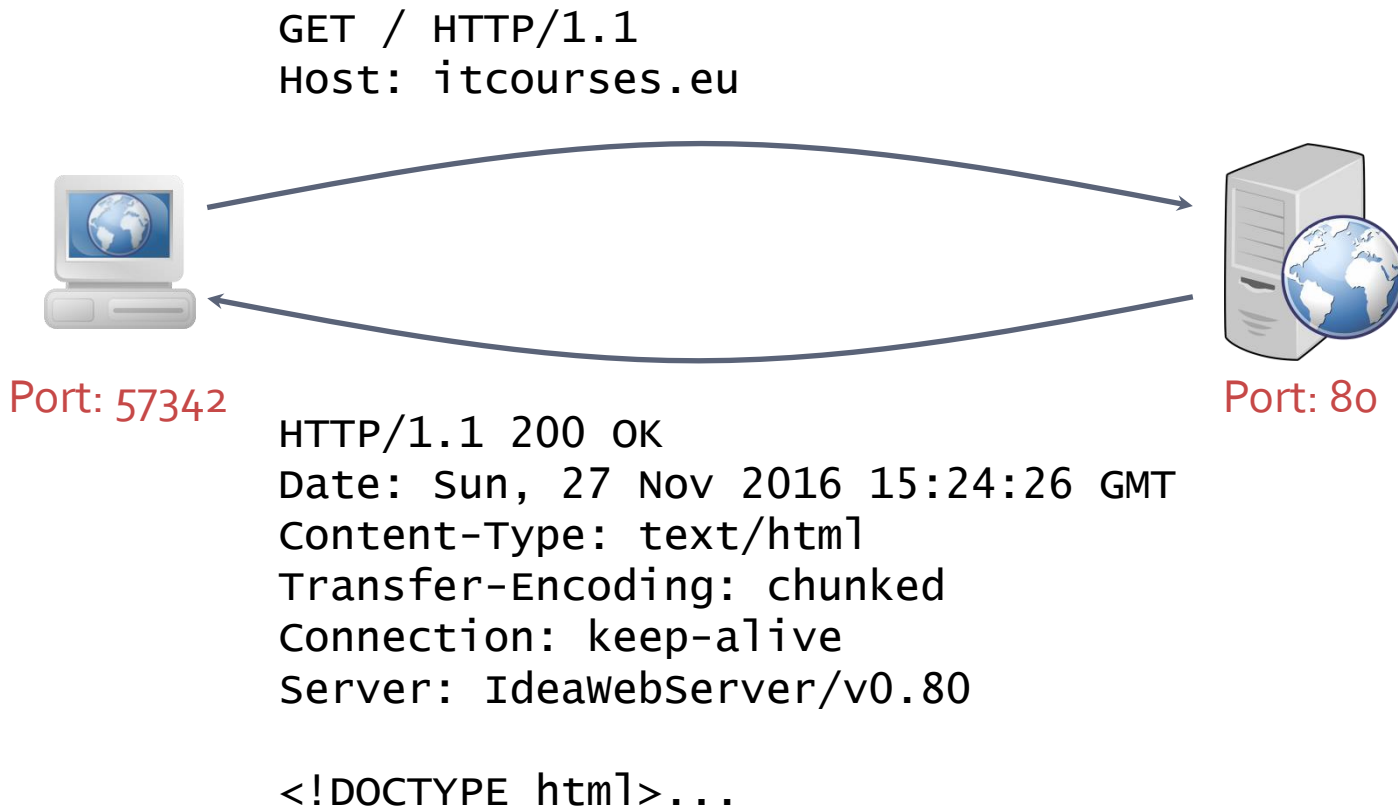
- Warto poczytać:
  - Wprowadzenie:
    - <https://developers.google.com/web/fundamentals/performance/http2/>
  - Inne wprowadzenie
    - <https://http2.akamai.com/>
  - Ciekawe FAQ: <https://http2.github.io/faq/>
  - Krytyczne artykuły:
    - <http://www.dobreprogramy.pl/HTTP2-to-triumf-polityki-a-nie-techniki.-Google-moze-sie-cieszyc-uzytkownicy-niekoniecznie,News,61166.html>
    - <https://queue.acm.org/detail.cfm?id=2716278>

# HTTP a warstwy sieci

- W większości modeli warstwowych HTTP jest w warstwie aplikacji



# HTTP – przykład komunikacji



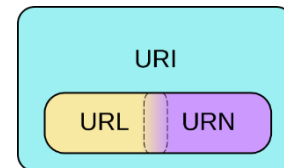
# HTTP

- DEMO: aplikacja w MS Azure
  - Przygotowanie
  - Konfiguracja
  - Dostęp

# Adresy zasobów

## ■ URI

- Ujednolicony sposób adresowania zasobów
- Specyfikacja: <http://www.ietf.org/rfc/rfc3986.txt>
  - Zastępuje wiele innych specyfikacji, m.in. dla URL, Relative URL, itd.



## ■ URI vs. URL vs. URN

- Uniform Resource Identifier vs. Locator vs. Name
- URN – nazwisko, URL – adres
  - URN – identyfikuje zasób, URL – określa jak znaleźć
- URI – URL lub URN, specyfikacja dopuszcza dodatkowe formaty

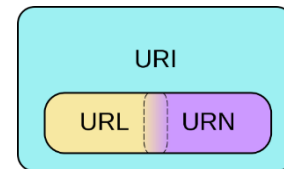
# Adresy zasobów

## ■ URL

- Wskazuje lokalizację zasobu
- Schemat & przykład
  - `<scheme>://<authority><path>?<query>#fragment`
  - `http://localhost:8080/path?q=text#wyniki`

## ■ URN

- Identyfikuje zasób poprzez nazwę w określonej przestrzeni nazw
- Schemat & przykład
  - `urn:<NID>:<NSS>`
  - `urn:isbn:0451450523`
    - książka The Last Unicorn z 1968, określona po numerze książki



# Adresy zasobów

- Uwagi:
  - Fragmenty nie są wysyłane do serwera
  - Lokalizacja fizyczna vs. Dynamiczna
    - /index.php?since=month vs. /blog/last-entries
  - Czasami można się spotkać z adresami typu
    - //ajax.microsoft.com/ajax/jquery/jquery-1.3.2.min.js
      - Jest to poprawny URI, tzw. Relative Reference (relatywny w odniesieniu do protokołu)
      - Więcej
        - <https://tools.ietf.org/html/rfc3986#section-4.2>
        - <http://blog.httpwatch.com/2010/02/10/using-protocol-relative-urls-to-switch-between-http-and-https/>

# Adresy zasobów

- Przykłady innych schematów
  - <ftp://user:pass@serwer.pl:21/dokument.txt>
  - <mailto:pawel@ii.uni.wroc.pl>
  - <news://pl.comp.os.linux/>
  - <telnet://156.17.4.4/>



# Adresy zasobów

- URL encoding
  - „Safe characters” – znaki dozwolone w adresach
  - „Unsafe characters” – wymagają zakodowania
    - Kodowanie za pomocą „%”, przykładowo
      - Spacja            %20
      - !                    %21
      - "                    %22
      - #                    %23
      - \$                    %24
      - %                    %25
  - Zgodnie z RFC3986 (URI Generic Syntax) znaki „safe” (czyli inaczej unreserved) to:
    - unreserved = ALPHA / DIGIT / "-" / "." / "\_" / "~"

# Rodzaje zawartości

- Standard MIME
  - Określony przez type/subtype
    - application/json
    - image/png
    - image/gif
    - text/xml
    - text/html
    - text/plain
  - Repozytorium
    - <http://www.iana.org/assignments/media-types/media-types.xhtml>
  - Rozszerzenie jest ostatnim miejscem, po którym rozpoznawany jest rodzaj zawartości
    - Chociaż mapowania są częścią konfiguracji serwera WWW

# Negocjacja treści

- Treść może być dostępna
  - W różnych formatach
    - np. możemy wybrać, czy chcemy XML czy JSON
  - W różnych językach
    - Przykład:  
wchodzimy na [www.google.com](http://www.google.com) ustawiając język na
      - pl
      - de
      - fr
      - en-gb

# Komunikacja

- Jedna transakcja
  - HTTP Request
  - HTTP Response

# Komunikacja

- Narzędzia
  - telnet (putty)
    - telnet host 80 (kwestia set localecho)
  - Fiddler
  - Postman
  - Narzędzia developerskie Google (network)
- DEMO: obejrzymy sobie powyższe

# Komunikacja

- Pobieranie strony
  - Obecne strony mają dużo zasobów – przyspieszenie pobieranie poprzez równoczesne połączenia
    - Specyfikacja: max 2 jednoczesnych połączeniach do hosta
    - Stan obecny: przeglądarki otwierają więcej połączeń

# Komunikacja

- Trwałe połączenie
  - Mechanizm pozwala na wiele żądań w ramach jednego połączenia
    - Wskazówka: ile zasobów tworzy jedną stronę WWW?
  - Serwer może zdecydować, czy to akceptuje

# Komunikacja

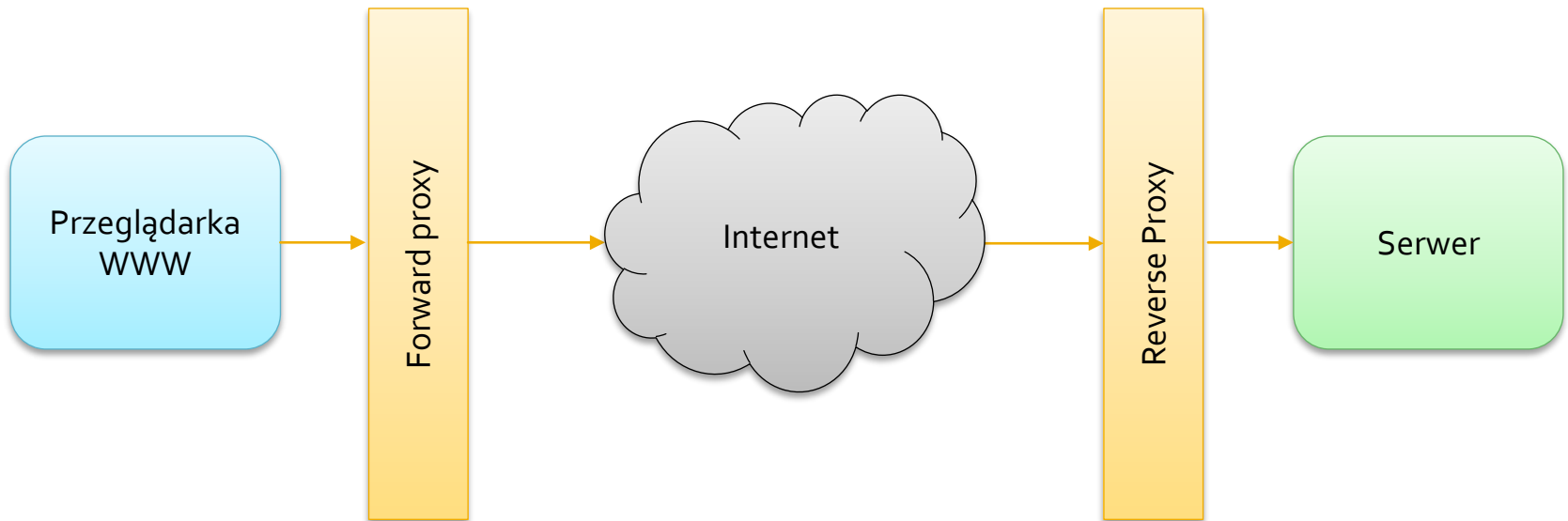
- Forward Proxy (czyli po prostu proxy)
  - Serwer pośredniczący pomiędzy klientem a docelowym serwerem
    - Jest po stronie klienta
  - Zastosowanie
    - Przyspieszenie transmisji
    - Filtrowanie treści, usuwanie tajnych informacji
    - Blokowanie dostępu do facebooka, twittera, allegro, ...



# Komunikacja

- Reverse proxy
  - Jest po stronie serwera
  - Zastosowanie
    - Loadbalancing
    - Kierowanie żądań o statyczne zasoby do dedykowane serwera (tzw. asset server)
    - Dodatkowa warstwa zabezpieczająca
    - Nakładanie dodatkowych elementów jak kompresja, SSL
    - Buforowanie często żądanych zasobów

# Komunikacja



# Buforowanie (cache)

- Metody HTTP a buforowanie
  - GET – buforowane, POST, PUT, DELETE – nie
- Public cache
  - Zasoby są buforowane na serwerach proxy
- Private cache
  - Zasoby są buforowane w przeglądarce
- Co wpływa na buforowanie
  - Cache-Control: public, private, no-cache, no-store
    - Dodatkowo można dodać
      - max-age: liczba-sekund
      - must-revalidate
  - Expires: data (np. w przeszłości)
  - Pragma: no-cache
  - Etag: „hash” zasobu – jeśli się zmieni, zasób musi zostać pobrany
- Sztuczka: dodanie ?wersja=1.2 do CSS, JS, itd.

# HTTP Request

- Struktura żądania
  - [metoda] [zasób] [wersja]  
[nagłówki]  
[treść]
  - Przykład
    - GET / HTTP/1.1  
Host: itcourses.eu

# HTTP Request

- Podstawowe metody
  - GET – pobranie zasobu
  - POST – aktualizacja zasobu
  - PUT – wgranie zasobu
  - DELETE – usunięcie zasobu
  - HEAD – pobranie nagłówków dla zasobu
- Pozostałe metody:
  - HEAD – tylko nagłówki, bez treści
  - TRACE – „echo”
  - OPTIONS – lista dozwolonych metod dla zasobu
  - CONNECT – tworzenie tunelu, cele diagnostyczne
  - PATCH – częściowa modyfikacja zasobu

# HTTP Request

- Metoda HTTP może być
  - Bezpieczna (safe) – nie modyfikuje zasobu
  - Idempotentna – jedno- i wielokrotne wykonanie daje taki sam stan zasobu
  - Buforowalna – wyniki jest sens buforować
- Podsumowanie

HTTP Method ↕	RFC ↕	Request Has Body ↕	Response Has Body ↕	Safe ↕	Idempotent ↕	Cacheable ↕
GET	<a href="#">RFC 7231</a>	No	Yes	Yes	Yes	Yes
HEAD	<a href="#">RFC 7231</a>	No	No	Yes	Yes	Yes
POST	<a href="#">RFC 7231</a>	Yes	Yes	No	No	Yes
PUT	<a href="#">RFC 7231</a>	Yes	Yes	No	Yes	No
DELETE	<a href="#">RFC 7231</a>	No	Yes	No	Yes	No
CONNECT	<a href="#">RFC 7231</a>	Yes	Yes	No	No	No
OPTIONS	<a href="#">RFC 7231</a>	Optional	Yes	Yes	Yes	No
TRACE	<a href="#">RFC 7231</a>	No	Yes	Yes	Yes	No
PATCH	<a href="#">RFC 5789</a>	Yes	Yes	No	No	Yes

# HTTP Request

- Kilka popularnych nagłówków
  - Referer
  - User-Agent
  - Accept
  - Accept-Language
  - Cookie
  - If-Modified-Since
  - Date

# HTTP Response

- Struktura odpowiedzi
  - [wersja] [status] [opis]  
[nagłówki]  
[body]
  - Nagłówki X-cos
    - Niestandardowe dla poszczególnych serwerów



# HTTP Response

- Kody odpowiedzi
  - 100-199 : Informacja
  - 200-299 : Żądanie zostało przetworzone poprawnie
  - 300-399 : Żądanie zostało przeadresowane
  - 400-499 : Błąd po stronie klienta (np. błędny URL)
  - 500-599 : Błąd po stronie serwera

# HTTP Response

- Często spotykane statusy
  - 200 OK
  - 206 Partial Content
  - 301 Moved Permanently
  - 302 Moved Temporarily
  - 304 Not Modified
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
  - 405 Method Not Allowed
  - 500 Internal Server Error
  - 503 Service Unavailable

# Nagłówki

- Nagłówki HTTP dzielimy na 4 grupy
  - Pola ogólnego przeznaczenia
  - Pola nagłówka żądania (klient)
  - Pola nagłówka odpowiedzi (serwer)
  - Pola nagłówka zawartości
- Reguły
  - Wielkość znaków w nazwach pól nie ma znaczenia
  - Kolejność pól również nie ma znaczenia
- Dalej przejrzymy kilka nagłówków

# Nagłówki ogólnego przeznaczenia

- Cache-Control – określa sposób buforowania
  - Przykładowe wartości żądania
    - no-cache, no-store, max-age
  - Przykładowe wartości odpowiedzi
    - public, private, no-cache, no-store, must-revalidate, max-age
- Connection – rodzaj połączenia
  - Connection: { keep-alive | close }
- Date – moment wysłania żądania/odpowiedzi
  - Date: format-daty
    - Czas jest zawsze względem GMT
    - Format powinien być zgodny z RFC 1123;

# Nagłówki żądania

- Host
- Accept
- Accept-Language
- If-Modified-Since
- User-Agent
- Accept-Charset
- Authorization
- Range
- Priorytety określone przez q-range, np.
  - Accept: typ/podtyp [q=ranga] [,...] ( $0 \leq \text{ranga} \leq 1$ )
    - Accept: text/\*, image/gif
    - Accept-Charset – określa preferowane alfabety

# Nagłówki odpowiedzi

- Accept-Ranges
- Retry-After
- Set-Cookie
- Etag
- Server
- WWW-Authenticate
- Location

# Nagłówki pola zawartości

- Content-Encoding
- Content-Language
- Content-Length
- Content-Range – określa pobrany fragment
  - Content-Range: początek-koniec/rozmiar
- Content-Type
- Expires
- Last-Modified

# Wysyłanie danych do serwera

- Dwa główne sposoby:
  - Metodą GET
    - Polega na dołączeniu danych dla URL-a
    - Pozwala na utrzymywanie kontekstu strony
      - Inaczej mówiąc, można skopiować URL i komuś podesłać
    - Trzeba pamiętać o limicie przesyłanych danych
    - Nie wolno tak przysyłać haseł (dlaczego?)
  - Metodą POST
    - W ten sposób powinno się wysyłać dane z formularzy
    - Jedyne sposoby na przesłanie danych binarnych np. plików



# Wysyłanie danych do serwera

- Użycie metody POST
  - Dane są kodowane zwykle na dwa sposoby
    - x-www-form-urlencoded
    - multipart/formdata (format MIME, RFC1867)
  - Przykład, wysyłanie prostych danych
    - POST /htdocs/processdata.php HTTP/1.1  
Host: localhost  
Content-type: application/x-www-form-urlencoded  
Content-length: 48  
  
imie=Jan&nazwisko=Kowalski&miasto=Warszawa

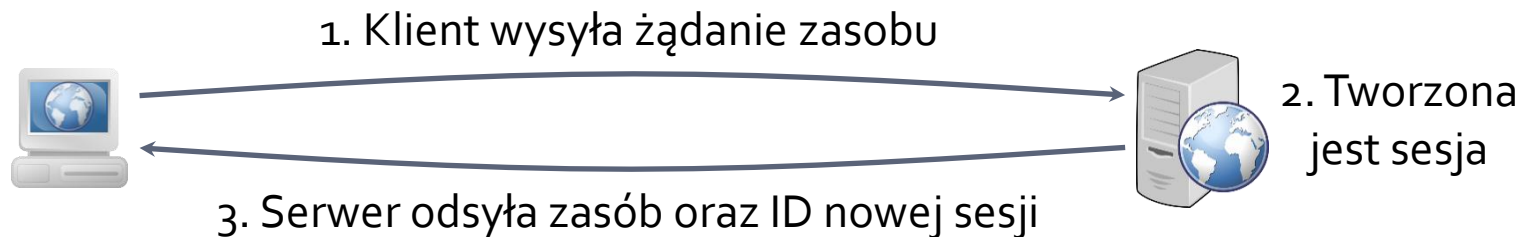
# Stan aplikacji i sesja

- Każda para request-response stanowi jedną transakcję
  - Czyli nie ma żadnego związku pomiędzy kolejnymi żądaniami
- Większość aplikacji jest „stanowa”
  - ... czyli gdzieś stan trzeba pamiętać
- Stan może być pamiętany po stronie
  - Klienta
    - Stan po stronie klienta vs. bezstanowość protokołu HTTP
  - Serwera
    - Potrzebne rozwiązanie...

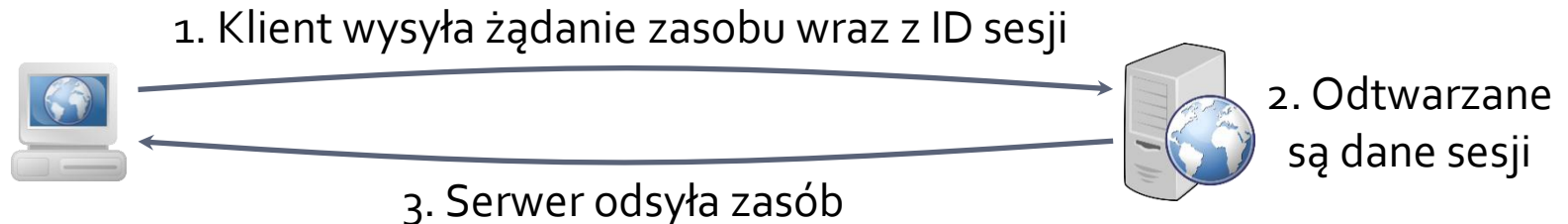
# Stan aplikacji i sesja

- Stan po stronie serwera i sesje

- Utworzenie sesji

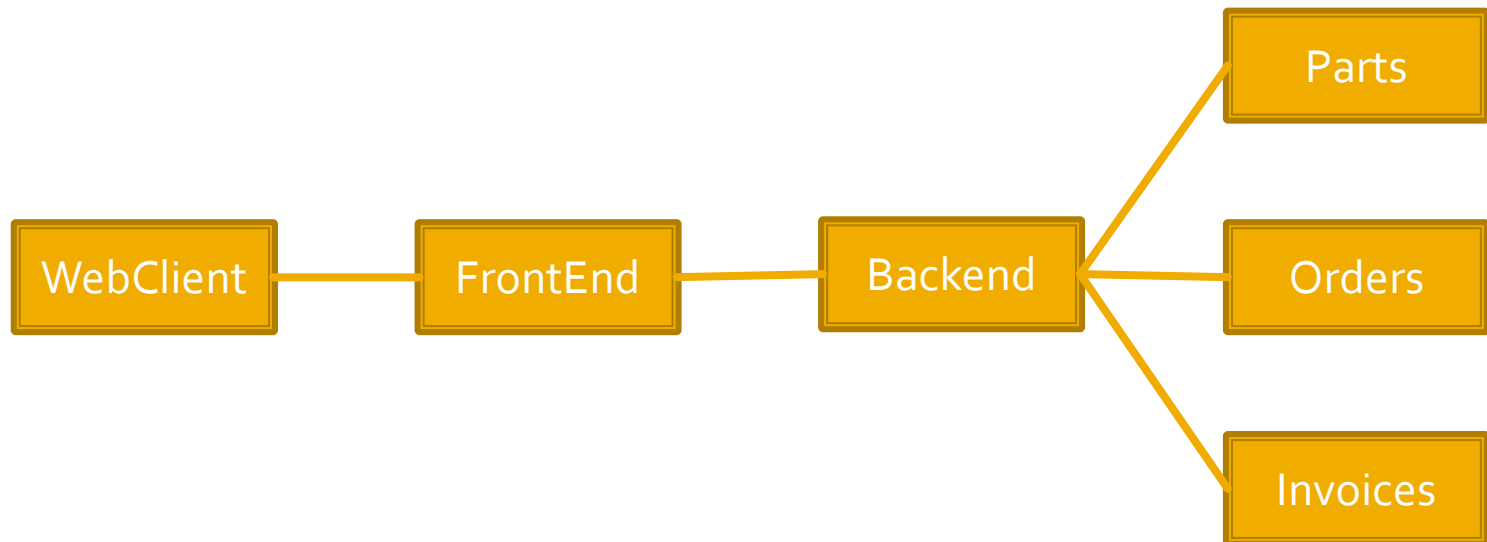


- Odtworzenie sesji



# Stan aplikacji i sesja

- A co w przypadku bardziej złożonych aplikacji?
  - Zalety „bezstanowości”



# Stan aplikacji i sesja

- Jak przekazać identyfikator?
  - Parametr GET
  - Cookies
- Pojęcie „sticky session”

# Cookies

- Co to są „ciastka”? RFC6265
  - Rozmiar ciastka: do 4KB
- Po stronie klienta
  - Cookie: SESSIONID=aasoizirj
- Po stronie serwera
  - Set-Cookie: SESSIONID=aasoizirj;  
domain=kursy24.eu;  
path=/

# Cookies

- Znaczenie parametrów
  - domain
  - path
- Ciastka sesyjne i trwałe
  - Expires
- Dodatkowe parametry:
  - Secure
  - HttpOnly

# Cookies

---

- Obejrzymy sobie ciastka w Google Chrome



# Narzędzia

---

- Postman
- Fiddler

# Na koniec

- Co mogło być, ale nie było, ale może jeszcze będzie:
  - DNSSec i ogólnie więcej o DNSie
  - Nagłówek Authorization i uwierzytelnianie
  - HTTPS, szyfrowanie, certyfikaty i PKI
  - RESTful, czy więcej o metodach, kodach odpowiedzi i innym spojrzeniu na WWW