Warstwa aplikacji

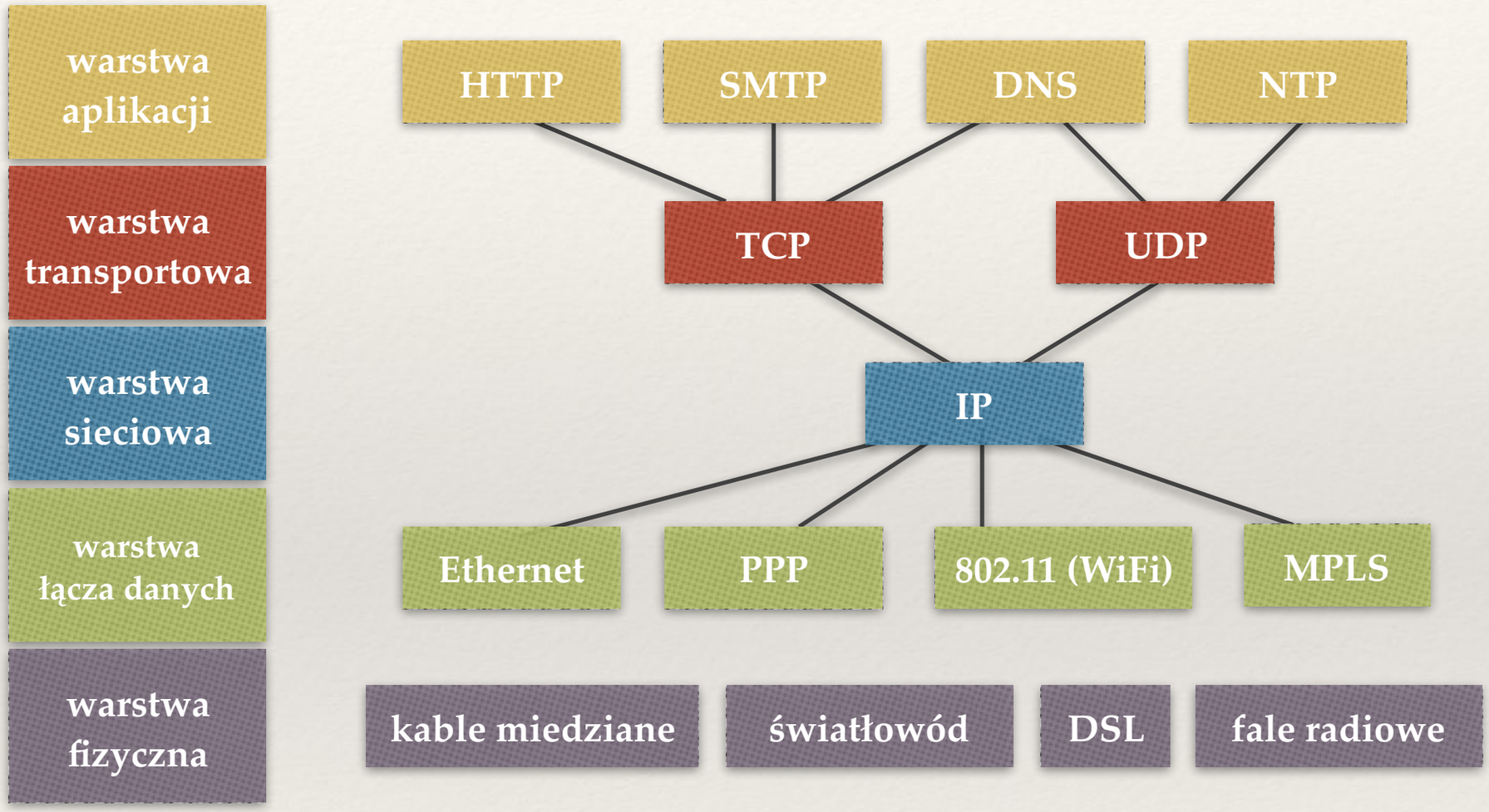
część 1

Sieci komputerowe

Wykład 8

Marcin Bieńkowski

Protokoły w Internecie



Trzy najpopularniejsze zastosowania

- DNS (*Domain Name System*)
  - Zamienia nazwy symboliczne na adresy IP i z powrotem.
- HTTP (*Hypertext Transfer Protocol*)
  - Przesyłanie danych w architekturze klient-serwer.
- BitTorrent
  - Przesyłanie danych w architekturze peer-to-peer.

DNS

Nazwy symboliczne a adresy IP

- Większości ludzi łatwiej zapamiętać jest nazwę symboliczną
  - `www.ii.uni.wroc.pl` → `156.17.4.11`
  - `atm-wro-pb1-wro-br1.devs.futuro.pl` → `62.233.154.25`.

- Dodatkowa warstwa pośrednia: można zmienić adres IP (zmiana ISP) a nazwa domeny pozostaje taka sama.

/etc/hosts

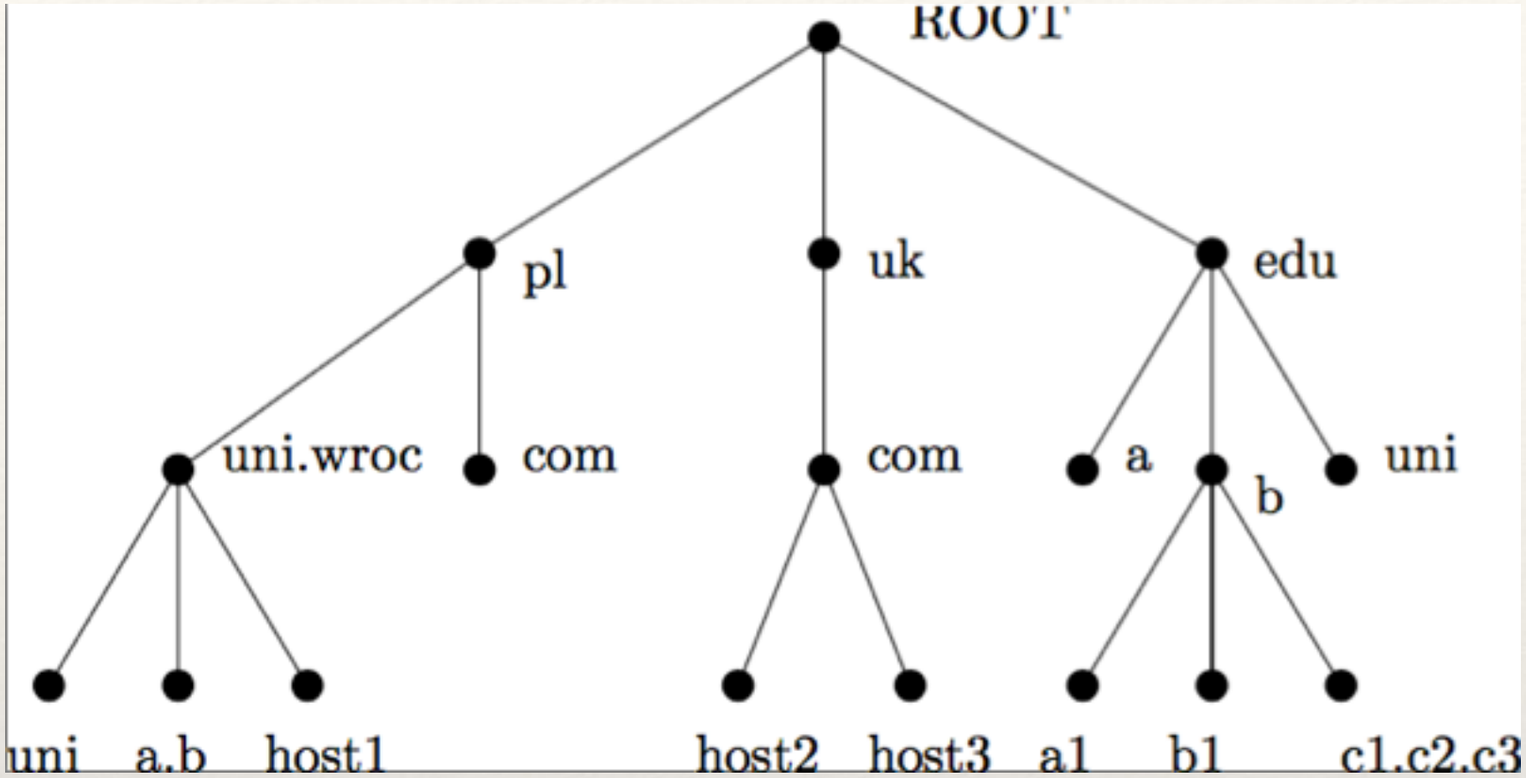
- Można takie odwzorowanie zapisać lokalnie (plik `/etc/hosts`).

- W początkach Internetu:
  - Pojedynczy i centralnie przechowywany plik HOSTS.TXT.
  - Każdy mógł go pobrać i zapisać do pliku `/etc/hosts`.
  - Aktualizacje HOSTS.TXT przez email do administratora.
  - Problemy z koordynacją, aktualizacją, dostępem, skalowalnością.

Cele DNS

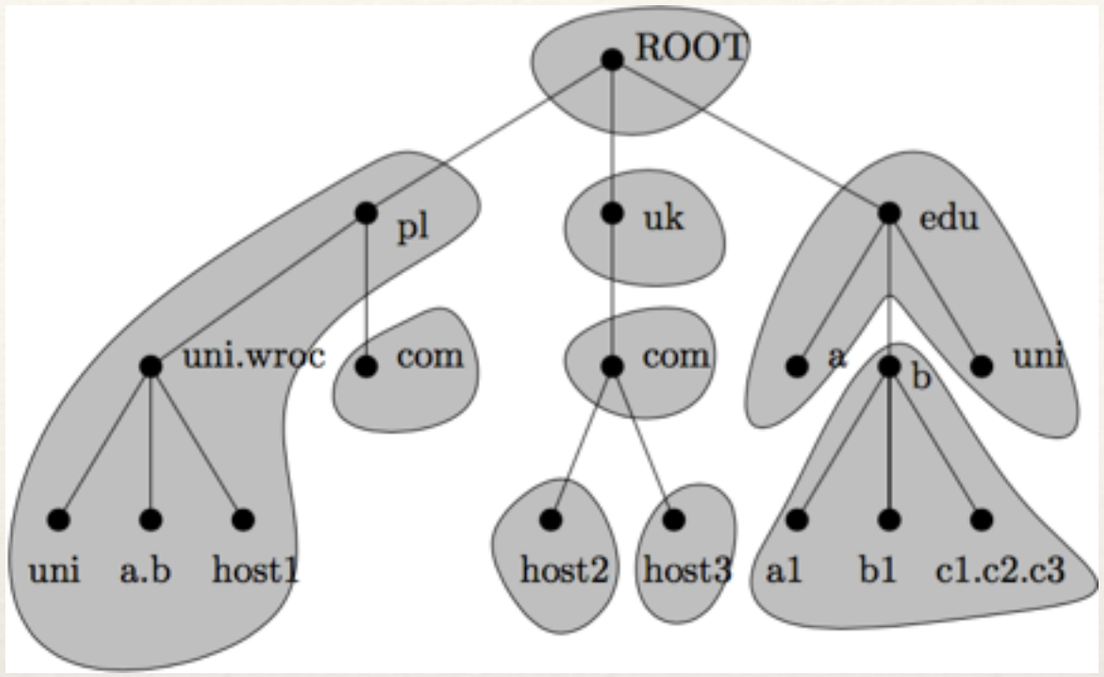
- Przekształcanie nazw na adresy (lub ogólnie: na inne informacje)
- Obsługiwanie olbrzymiej liczby rekordów (ok. 300 mln nazw, nie licząc poddomen).
- Rozproszone zarządzanie.
- Odporne na błędy pojedynczych serwerów.

Hierarchia nazw domen



- Korzeń drzewa oznaczany kropką.
- Pierwszy poziom drzewa = domeny TLD (*top level domain*)
  - `.pl`, `.uk`, `.org`, `.com`, `.guru`, `.pizza`, ...
- Może istnieć `host.poddomena.domena`, choć nie istnieje wpis `poddomena.domena`.

Rozproszone zarządzanie



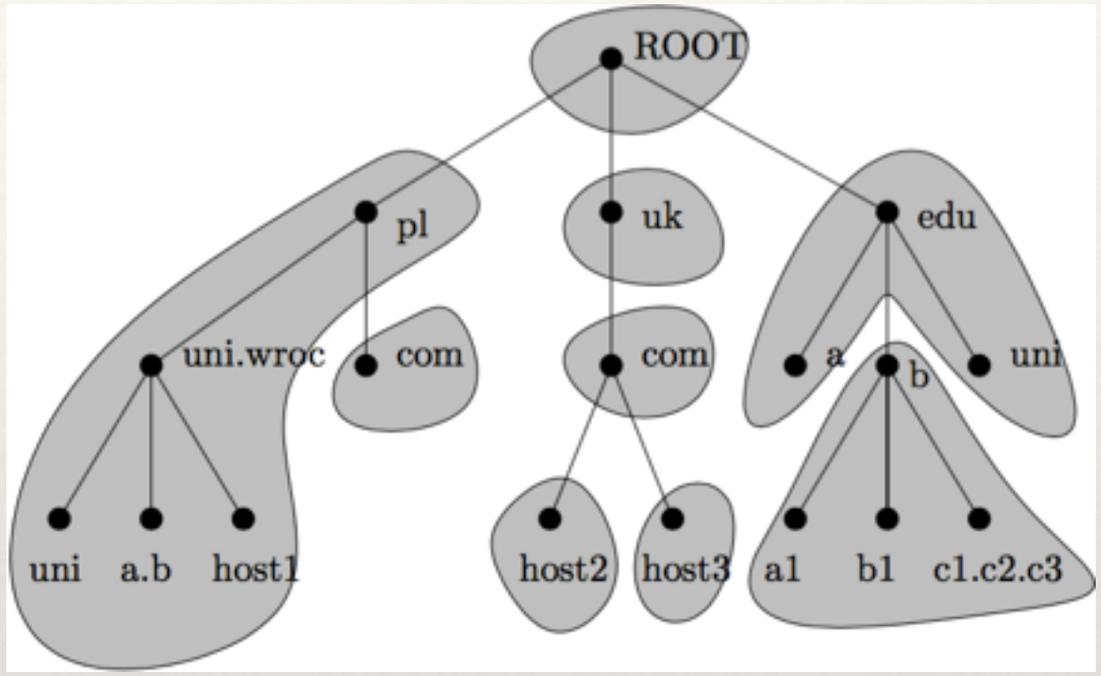
Strefa

- Spójny fragment drzewa
- Najmniejsza jednostka administracyjną DNS, odrębnie zarządzana.
- Właściciel strefy = serwer(y) DNS (zazwyczaj 2-5).
- Taki serwer ma informacje o nazwach w strefie i serwerach odpowiedzialnych za strefy podrzędne.
  - Krawędzie w dół = delegacje.

Serwery główne

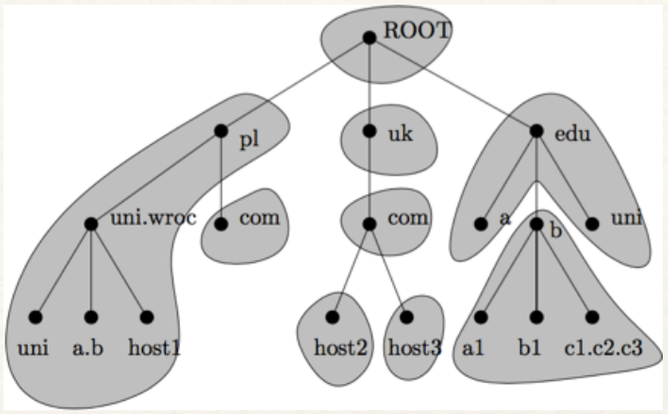
13 serwerów *głównych* dla strefy .

```
A.ROOT-SERVERS.NET. = 198.41.0.4

B.ROOT-SERVERS.NET. = 192.228.79.201

C.ROOT-SERVERS.NET. = 192.33.4.12

D.ROOT-SERVERS.NET. = 128.8.10.90

...
```



Taką informację musimy wpisać ręcznie (rzadko modyfikowana, jest w standardowych plikach konfiguracyjnych).

Rozszyfrowywanie nazw (resolving)



Chcemy: poznać adres IP dla `a1.b.edu`.

- Pytamy jeden z serwerów nazw dla "`.`",
  np. `E.ROOT-SERVERS.NET` o adresie IP `192.203.230.10`.
  → Serwer nie zna, ale mówi, że serwerem nazw dla `edu` jest foo.edu o adresie IP = 1.2.3.4.

- Pytamy `foo.edu`.
  → Serwer nie zna, ale mówi, że serwerem nazw dla `b.edu` jest `foo.bar.uni.edu` o adresie IP = `5.6.7.8`.

- Pytamy foo.bar.uni.edu → Serwer `foo.bar.uni.edu` odpowiada: `4.8.2.4` (bo jest serwerem nazw dla strefy zawierającej `a1.b.edu`).

Rozszyfrowywanie iteracyjne i rekurencyjne

- **Rozszyfrowywanie iteracyjne** = klient przechodzi drzewo DNS zaczynając od korzenia (jak na poprzednich slajdach).

- **Rozszyfrowywanie rekurencyjne** = pytamy resolver DNS, a on w naszym imieniu wykonuje odpytywanie.

- **Resolver DNS** = to co wpisujemy w polu „serwer DNS“ w konfiguracji sieci naszego komputera.
  - Dla poprawy wydajności, zapisuje zwracane wyniki w pamięci podręcznej.
  - Może być też serwerem DNS (odpowiedzialnym za jakąś strefę).

demonstracja

Informacje przechowywane w strefie

Najczęściej:

- Adresy IP (rekordy `A` i `AAAA`)

```
google.pl.    A       216.58.209.67
google.pl.    AAAA    2a00:1450:401b:801::2003
```

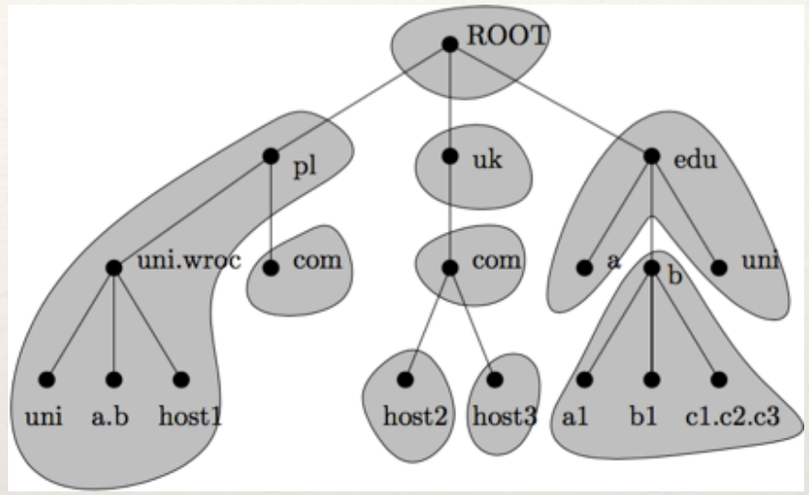
- Aliasy nazw (rekordy `CNAME`)

```
host1.uni.wroc.pl.    CNAME a.b.c.uni.wroc.pl.
```

Opis strefy: delegacje

- Dla poddomeny `com.pl` należy umieścić wpis, że należy do innej strefy:

```
com.pl.          NS    ns1.com.pl.
com.pl.          NS    ns2.pl.
ns1.com.pl.      A     10.0.8.1
ns2.pl.          A     10.20.30.40
```



- Ostatnie dwa wpisy = wpisy sklejające.

Domena odwrotna

- **Odwrotna konwersja: adres IP → nazwa domeny.**
  - Wykorzystuje rekord PTR.
  - Sztuczna domena `in-addr.arpa`, której poddomenami są klasy lub adresy IP.

- **Przykładowo:**
  - strefa `33.22.11.in-addr.arpa` zawiera informacje na temat sieci `11.22.33.0/24`
  - w szczególności zawiera wpis `44.33.22.11.in-addr.arpa PTR` nazwa.domena.org

demonstracja

Odpytywanie DNS w programie

```
int getaddrinfo(const char *domain, const char *service,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

```
struct addrinfo {
  int ai_family;
  int ai_socktype;
  …
  struct sockaddr *ai_addr;
  struct addrinfo *ai_next;
};
```

zazwyczaj `AF_INET` albo `AF_INET6`

`SOCK_STREAM`, `SOCK_DGRAM`, …

result jest listą struktur `addrinfo`

Najprostszy przypadek użycia:

```
struct addrinfo* result;
int getaddrinfo(„www.example.com“, NULL, NULL, &result);
```

Kod odpytujący DNS

```
int main(int argc, char* argv[])
{
    struct addrinfo* result;
    struct addrinfo hints = {
        .ai_family = AF_INET,
        .ai_socktype = SOCK_STREAM,
    };
    getaddrinfo(argv[1], NULL, &hints, &result);

    for (struct addrinfo* r = result; r != NULL; r = r->ai_next) {
        struct sockaddr_in* addr = (struct sockaddr_in*)(r->ai_addr);
        char ip_address[20];
        inet_ntop (AF_INET, &(addr->sin_addr), ip_address,
            sizeof(ip_address));
        printf ("%s\n", ip_address);
    }
}
```

Brak obsługi błędów, plików nagłówkowych, etc.

filtrujemy niechciane informacje

demonstracja

cały kod programu

HTTP

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, ftp, mailto, …)
  - część zależna od rodzaju zasobu

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

URL dla schematu http

- Po dwukropku:
  - `//`
  - nazwa DNS serwera
  - opcjonalnie `:port`
  - `/`
    - identyfikator zasobu wewnątrz serwera
      - niekoniecznie ścieżka do pliku,
      - `/` w identyfikatorze wskazuje na hierarchię.

- Przykład: `http://www.ii.uni.wroc.pl:80/~mbi/dyd/sieci_16s/`

Pobieranie strony WWW krok po kroku (1)

- Przeglądarka WWW dostaje URL
- URL jest rozbijany na człony (zakładamy, że schemat = `http`).
- Nawiązuje połączenie TCP z serwerem WWW na porcie 80.
- Wysyła żądanie HTTP:

```
GET /~mbi/dyd/sieci_16s/ HTTP/1.1
Host: www.ii.uni.wroc.pl
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, etc.). Przykłady:
  - `text/plain`
  - `text/html`
  - `image/jpeg`
  - `video/mpeg`
  - `application/msword` — dokument `.doc(x)`
  - `application/pdf` — dokument PDF
  - `application/octet-stream` — ciag bajtów bez interpretacji

Pobieranie strony WWW krok po kroku (3)

- Serwer wysyła odpowiedź:

```
HTTP/1.1 200 OK
Server: Apache/2.2.21 (Unix) … OpenSSL/0.9.8k
Last-Modified: Wed, 20 Apr 2016 21:58:30 GMT
Content-Length: 5387
Content-Type: text/html

PLIK (w tym przypadku dokument HMTL)
```

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

Zapytanie warunkowe GET

- W nagłówku podajemy:

```
If-Modified-Since: Wed, 20 Apr 2016 23:27:04 GMT
```

- Możliwe odpowiedzi:
  - `200 OK`
  - `304 Not Modified`

- Umożliwia implementację pamięci podręcznej w przeglądarce.

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 dokumentu, HTML określa zazwyczaj tylko strukturę).
    - czcionki
    - …
  - elementy osadzone są pobierane przez kolejne żądania HTTP i wyświetlane przez przeglądarkę.

Połączenia trwałe (1)

- Nawiązywanie połączenia TCP = duży narzut czasowy.
- Zazwyczaj przeglądarka pobiera wiele dokumentów naraz (np. strona WWW + obrazki)
- Standard HTTP 1.1: połączenie jest domyślnie otwarte.
- Zamknięcie połączenia po odpowiedzi na żądanie w którym umieścimy wiersz `Connection: close`.

Połączenia trwałe (2)

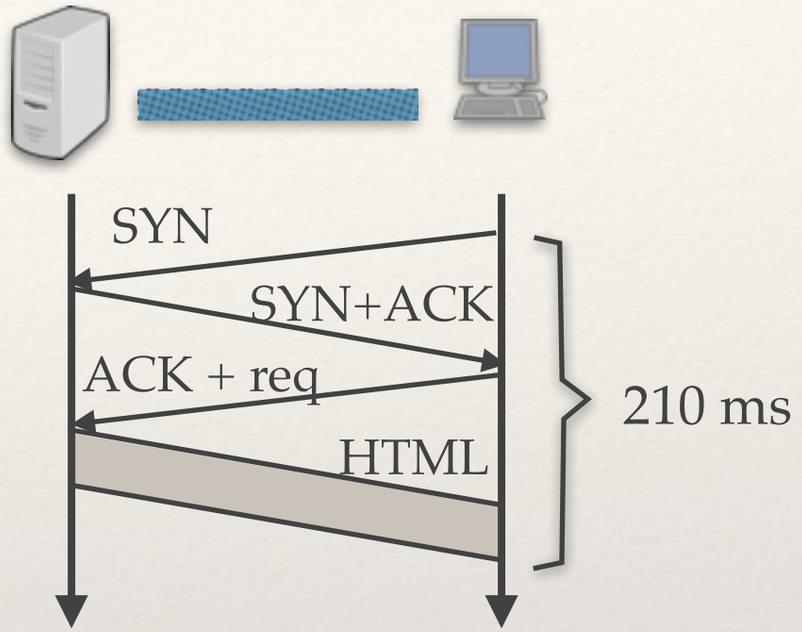
Przykład:

- 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).

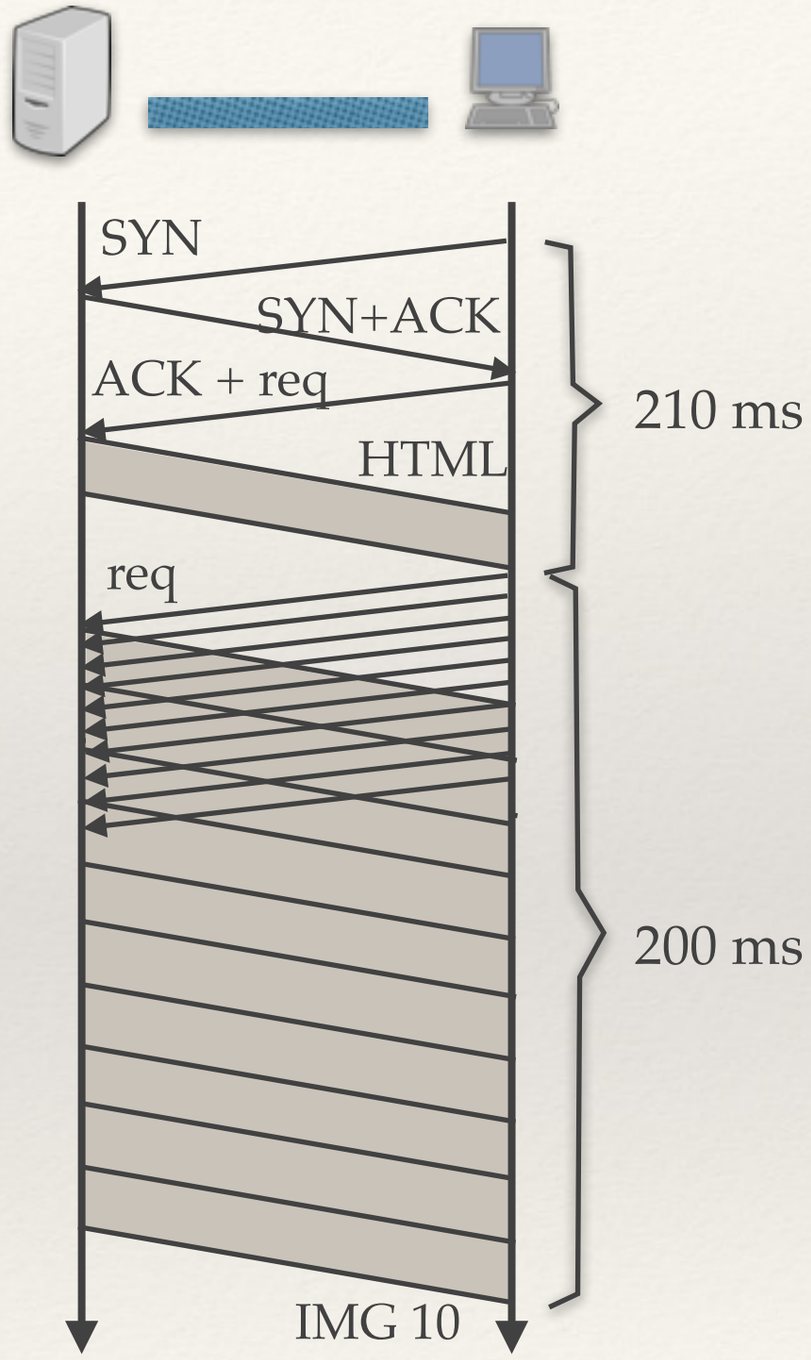
- Otrzymanie strony HTML: 210 ms.
- Pobieranie każdego z obrazków: kolejne: 210 ms.
- Zazwyczaj dwa równoległe połączenia do serwera → pobieranie 10 obrazków trwa 210 ms * (10/2) = 1050 ms.
- Całkowity czas: 210 + 1050 = 1260 ms.



Połączenia trwałe (4)

HTTP/1.1 (połączenia trwałe).

- Otrzymanie strony HTML: 210 ms.
- Wysłanie zapytań o 10 obrazków: 50 ms.
- Wysyłanie obrazków: 50 + 10 * 10 = 150 ms.
- Całkowity czas: 210 + 150 = 360 ms.
- Niepotrzebne dodatkowe połączenia TCP



HTTP/2

- Oparty na SPDY (protokół zaproponowany przez Google)
- Binarny protokół.
- Przetwarzanie żądań w innej kolejności niż nadchodzą.
- Usuwanie powtarzających się nagłówków
- Kompresja.
- …

Dynamiczne strony WWW

Dynamika po stronie klienta:

- Javascript: prosty obiektowy interpretowany język zintegrowany z HTML.
- Aplikacje Flash, Silverlight, aplety Javy (wykonanie realizowane przez odpowiednie wtyczki do przeglądarki).

Dynamika po stronie serwera:

- URL może wskazywać na program, którego wynikiem działania jest HTML.
  - CGI (*Common Gateway Interface*): standard umożliwiający wykonanie dowolnego zewnętrznego programu.
  - Mechanizmy zintegrowane z serwerem WWW (PHP, JSP, ASP, mod_perl, ...)
- 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.

demonstracja

HTTP jako warstwa transportowa

- Pisanie poprawnych programów wykorzystujących TCP nie jest trywialne.
- A może 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

BitTorrent

Peer-to-peer

P2P kontra klient-serwer

- Wszystkie komputery są jednocześnie klientami i serwerami.
- Każdy komputer może nawiązywać połączenia z innymi (bez pośrednictwa znanego serwera).
- Brak centralnego miejsca z danymi:
  - Lepsza skalowalność i niezawodność
  - Autonomia, brak administracji, ale trudniejsze zagwarantowanie współpracy całości.
- Problemy jeśli oba komputery są za NAT.

Udostępnianie pliku

Udostępnianie pliku X

- Związujemy z plikiem X plik `.torrent`, umieszczany np. na stronie WWW.
- Plik X dzielony na kawałki rzędu 32 KB - 16 MB.
- Plik `.torrent` zawiera informacje takie jak adres IP *trackera* oraz funkcje skrótu dla wszystkich kawałków.

Podłączanie się do sieci

- Pobieramy z jakiegoś (zewnętrznego) serwisu plik `.torrent`
- Łączymy się z zadanym trackerem.
- Tracker zna wszystkich członków i udostępnia adresy niektórych (ok. 50).
- Po jakimś czasie możemy prosić o kolejne adresy członków.

Pobieranie pliku

- Dwa typy członków sieci:
  - *Seeder*: ma wszystkie kawałki pliku,
  - *Leecher*: ma tylko niektóre kawałki.

animacja

- Każdy z członków sieci ma pewną liczbę slotów na udostępnianie pliku.
  - *Seeder* udostępnia chętnym po kolei.
  - *Leecher* udostępnia chętnym pod warunkiem, że dadzą mu coś w zamian.
  - Wyjątek: jeśli klient mówi, że jest nowy, to dostaje kawałek od leechera za darmo.

- Po pobraniu kawałka sprawdzamy jego poprawność za pomocą funkcji skrótu z pliku `.torrent`.
- Oficjalny klient ma wbudowaną politykę „najpierw najrzadsze kawałki“.

Lektura dodatkowa

- Kurose & Ross: rozdział 2.
- Tanenbaum: rozdział 7.