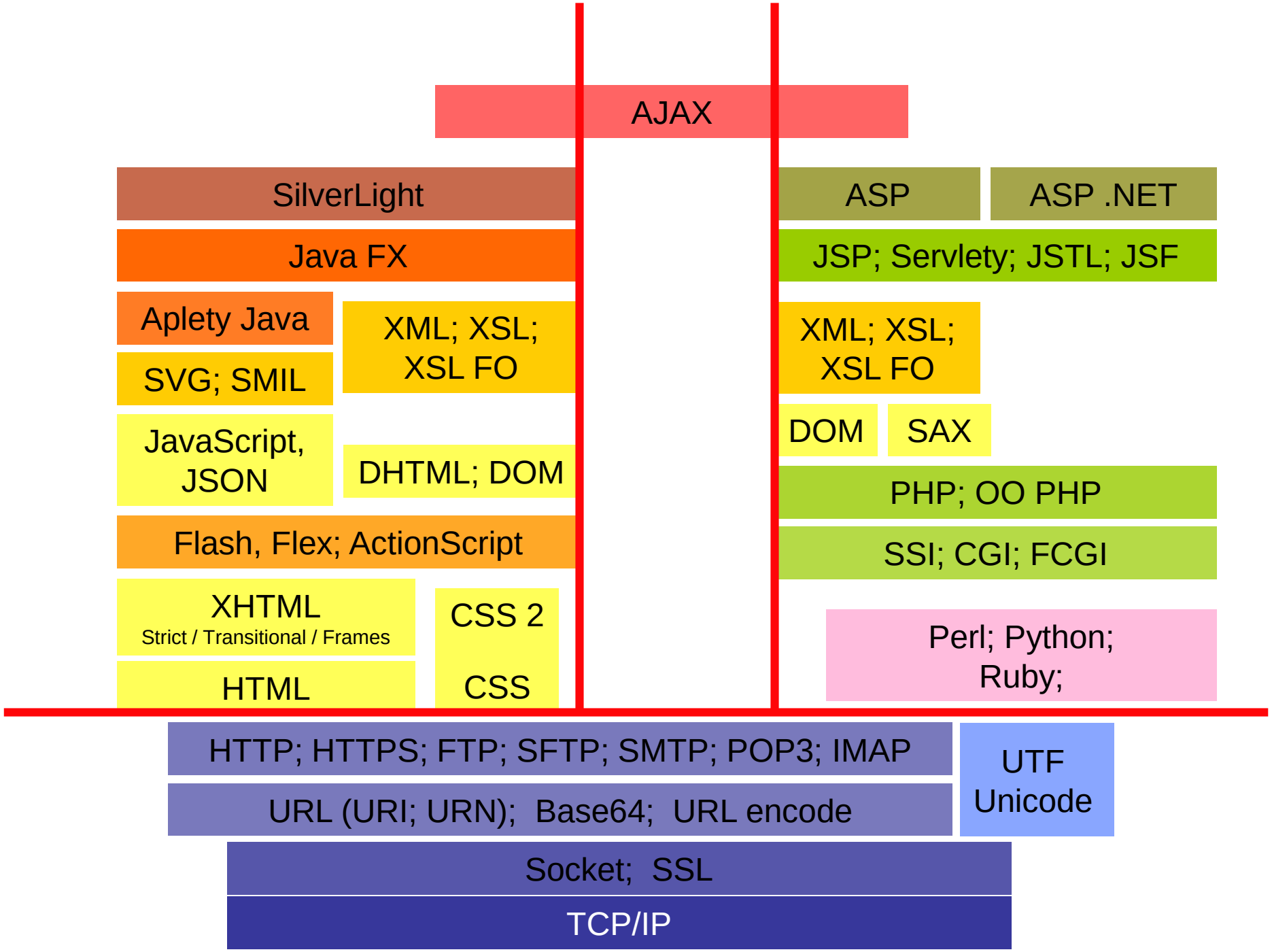
Architektury Usług Internetowych
Protokół HTTP

Tomasz Dziubich

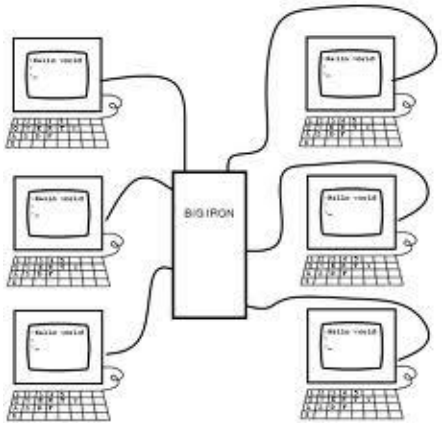
4.12.2016

Plan wykładu

- Protokół SMTP
- Rozszerzenie MIME
- Podstawy protokołu HTTP
- Protokół HTTPS
- Protokół HTTP 2.0

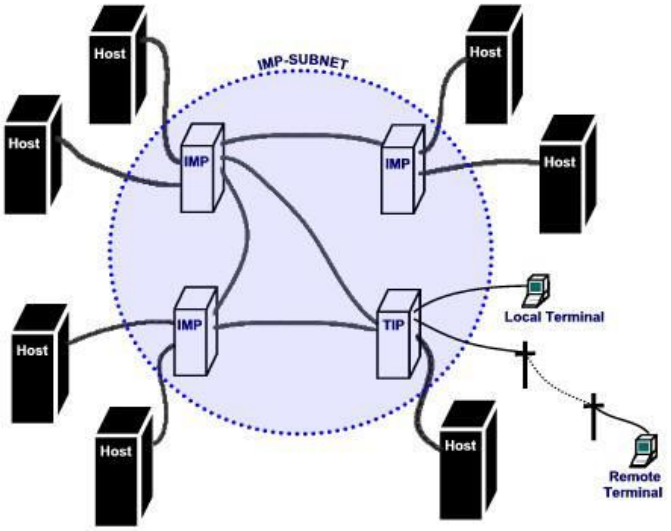

AJAX
SilverLight
Java FX
Aplety Java
SVG; SMIL
JavaScript, JSON
XML; XSL; XSL FO
DHTML; DOM
Flash, Flex; ActionScript
XHTML
Strict / Transitional / Frames
HTML
CSS 2
CSS
ASP
ASP .NET
JSP; Servlety; JSTL; JSF
XML; XSL; XSL FO
DOM
SAX
PHP; OO PHP
SSI; CGI; FCGI
Perl; Python; Ruby;
HTTP; HTTPS; FTP; SFTP; SMTP; POP3; IMAP
URL (URI; URN); Base64; URL encode
UTF Unicode
Socket; SSL
TCP/IP

Rozwój architektury Internetu



Mainframe (wielodostęp)

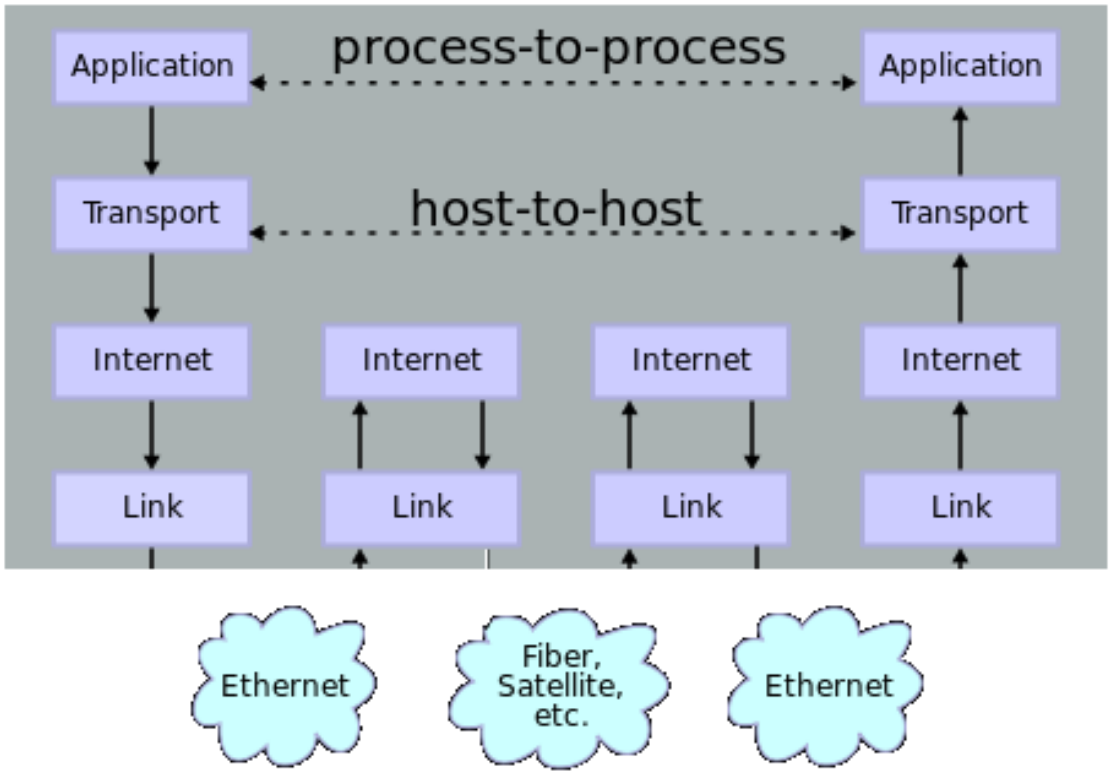
- Pierwsze aplikacje poczty internetowej (1965-1971) - sendmail



Protokoły internetowe (IP/TCP)

- Program telnet (1969)
- Połączenie komputerów z różnych instytucji (1974)
- DNS (Domain Name System)

Model warstwowy i protokoły



HTTP, SMTP, FTP

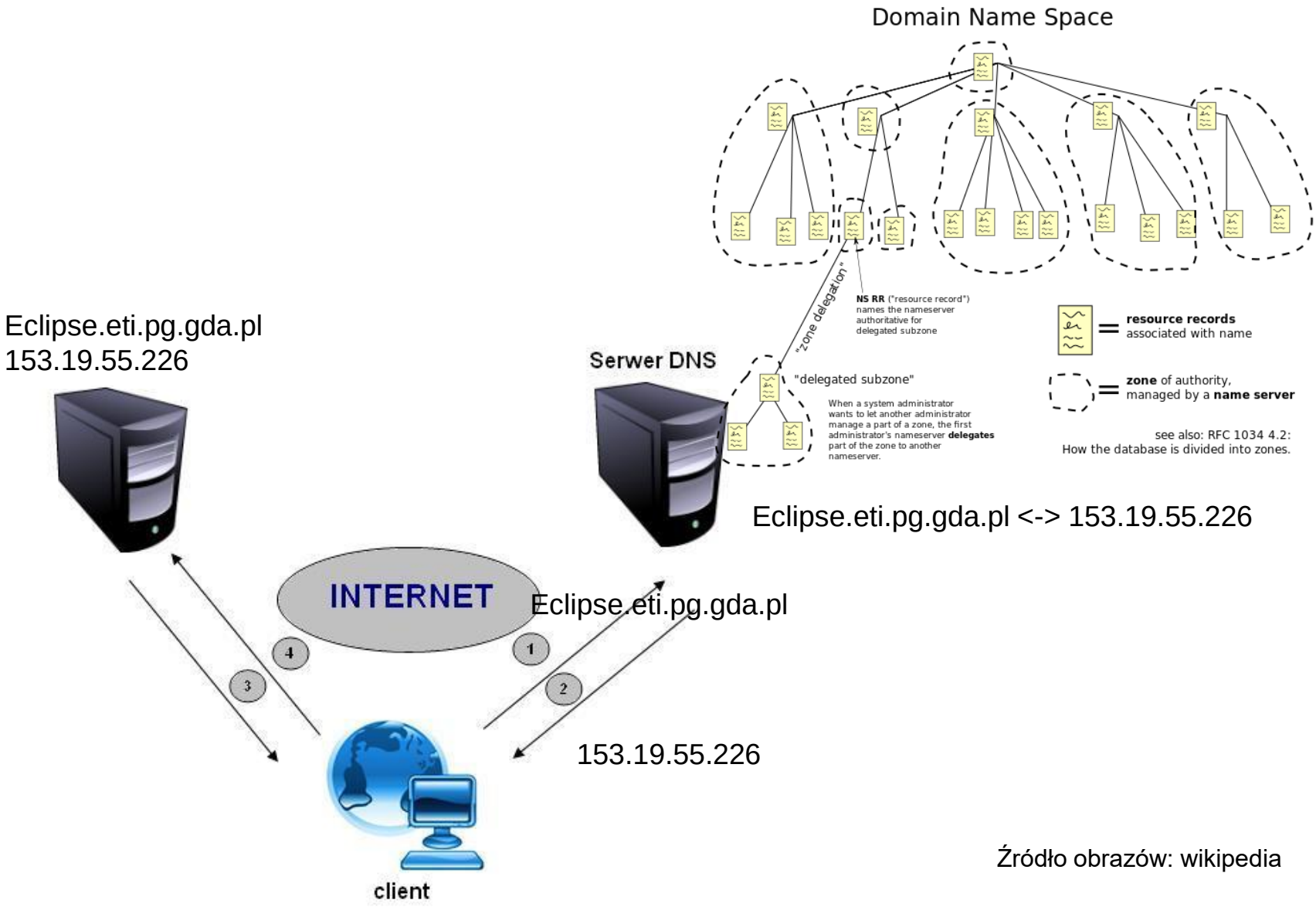
Port

TCP, UDP

Adres

IP

Usługa nazewnicza DNS



Źródło obrazów: wikipedia

SMTP: Simple Mail Transfer Protocol

MIME: Multipurpose Internet Mail Extension

HTTP: Hypertext Transfer Protocol

URL(URI): Uniform Resource Locator
HTML: Hypertext Mark-up Language

HTTPS: Secure HTTP

.

Źródła informacji o protokołach

RFC - Request For Comments

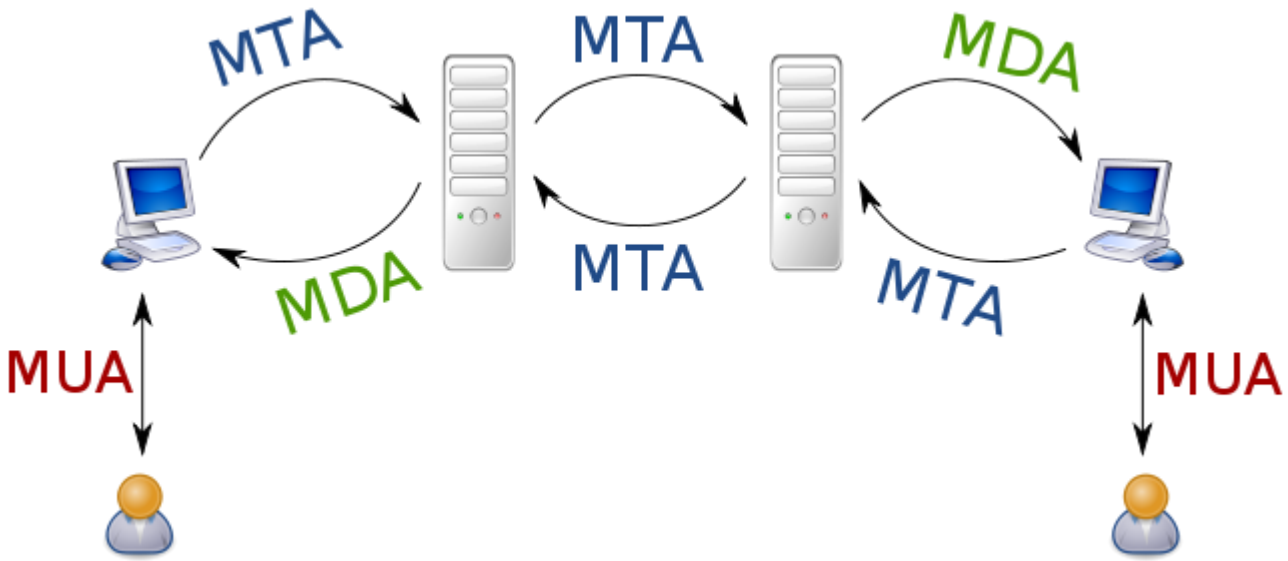
Źródło informacji głównie o protokołach usług i standardach obowiązujących w wymianie informacji w Internecie. Są to numerowane dokumenty (specyfikacje i opisy): np. HTTP wersja 1.0 - RFC1945, wersja 1.1 - RFC2616

Niektóre dokumenty aktualizują inne lub czynią je "przestarzałymi", np. RFC2616 zaktualizował RFC2068

Przykładowe lokalizacje zawierające dokumenty RFC:
http://rfc.net/
http://www.rfc-editor.org/
http://sunsite.icm.edu.pl/pub/doc/rfc/
.

Schemat działania poczty elektronicznej

Uproszczony schemat nadawania



Wysłanie:
Mail User Agent (MUA) – klient poczty elektronicznej – wysyła do MTA.
Mail Transfer Agent (MTA) – serwer poczty elektronicznej – odbiera pocztę od MUA/MSA.
Mail Delivery Agent (MDA) – pobiera pocztę od MTA i przekazuje do skrzynek.

Odczyt:
Mail Submission Agent (MSA) – odbiera pocztę od MUA i wysyła do MTA.
Mail Access Agent (MAA) – pobiera pocztę ze skrzynek i wysyła do MRA.
Mail Retrival Agent (MRA) – pobiera pocztę od MAA.

POLITECHNIKA GDAŃSKA

SMTP - Simple Mail Transfer Protocol

Simple Mail Transfer Protocol (SMTP)- protokół do nadawania poczty elektronicznej.

Dokumentacja w [RFC 821].

Domyślny port: 25

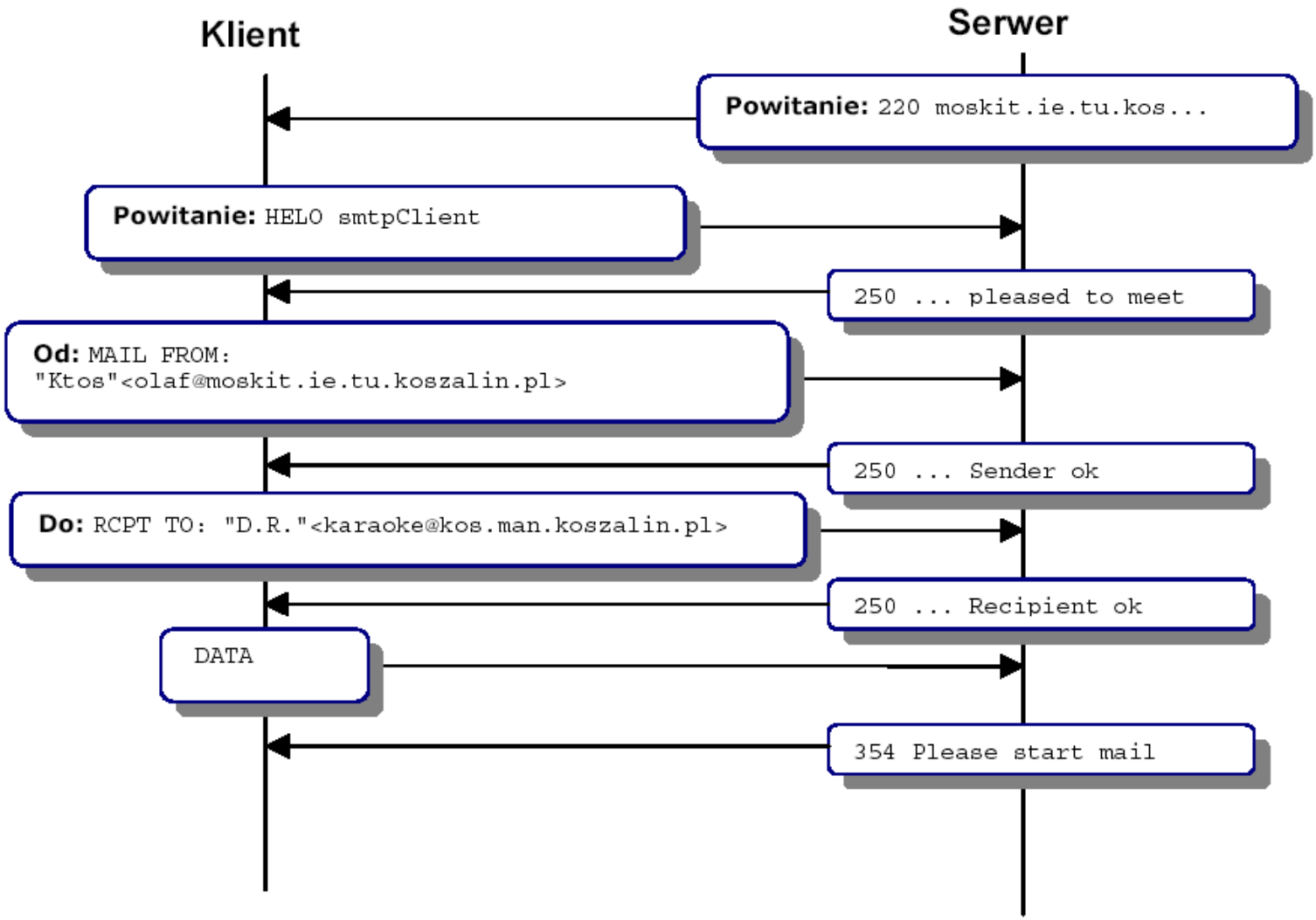
Składa się z komend oraz treści wiadomości (tzw. ciało).

Treść składa się nagłówków (Date:, Subject:, To:, From:) i treści właściwej.

Komendy protokołu przeważnie kończą się znakiem (znakami) nowej linii. Wyjątek to np. DATA

POLITECHNIKA GDAŃSKA

SMTP - Przebieg podstawowej sesji



POLITECHNIKA GDAŃSKA

SMTP - Przebieg podstawowej sesji

1. Po nawiązaniu połączenia przez klienta serwer wysyła informację powitalną.

2. Klient opcjonalnie wysyła HELO nazwa lub EHLO nazwa, gdzie nazwa "przedstawia" serwerowi klienta. Komenda EHLO umożliwia skorzystanie z rozszerzonych funkcji protokołu.

3. Po każdej komendzie serwer odpowiada statusem jej wykonania. Status jest postaci: trzycyfrowy numer i dodatkowa informacja tekstowa, np. 220 Service ready ESMTP, 250 ok

4. W celu wysłania poczty klient specyfikuje nadawcę za pomocą MAIL FROM: e-mail. Serwer odpowiada.

5. Następnie klient specyfikuje odbiorców za pomocą kolejnych komend RCPT TO: e-mail. Serwer odpowiada po każdej z nich.

6. Wiadomość zaczyna się po użyciu komendy DATA. Po potwierdzeniu klient może wyspecyfikować wiadomość, która kończy się (ale nie zawiera) sekwencją nowa_linia, znak '.', nowa_linia Po wystąpieniu tej sekwencji serwer odpowiada statusem.

7. Po wysłaniu wiadomości można zakończyć pracę komendą QUIT lub wysłać nastepną wiadomość. Nie trzeba wtedy specyfikować jeszcze raz nadawcy.

SMTP – Pierwotne wady

1. Brak kodowania znaków narodowych (np. Latin2, Mazovia, cp-1250)

2. Brak możliwości przekazywania treści binarnych (np. obrazki czy programy)

-> rozwiązaniem było wprowadzenie MIME (kodowanie i zawartość)

Standard MIME (*Multipurpose Internet Mail Extensions*) definiuje różne sposoby kodowania i przesyłania wiadomości.

Kodowanie MIME (1)

Quoted-printable:

Każdy znak 8-bitowy może być reprezentowany przez sekwencję =numer_heksadecymalny_kodu_ASCII. Jeżeli numer zawiera cyfry heksadecymalne A,B,C,D,E,F to są dozwolone tylko wielkie znaki
Większość znaków US-ASCII (do kodu 127) nie musi być reprezentowana przez powyższą sekwencję

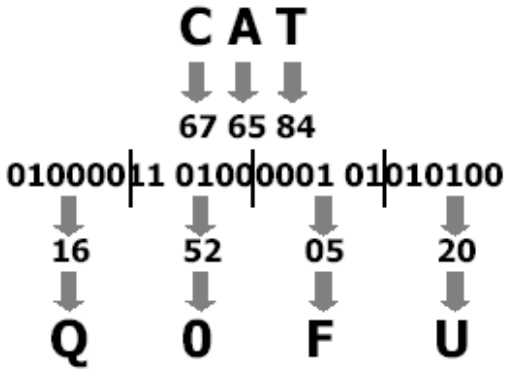
```
Tekst źródłowy:  Kiedyś to będzie brzmiało jaśniej
Quoted-printable:  Kiedy=B6 to b=EAdzie brzmia=B3o ja=B6niej
```

Base64

Każda sekwencja trzech znaków 8-bitowych (czyli 24 bitów) zamieniana jest na sekwencję czterech liczb 6-bitowych, z których każda stanowi indeks znaku 64-znakowego alfabetu (zawiera wielkie i małe litery US_ASCII, cyfry, znak + i znak /). Kolejne znaki tego alfabetu są wynikiem kodowania.

Dodatkowy znak specjalny '=' służy np. do uzupełniania brakujących bajtów w sekwencjach 3-bajtowych

6-Bit Value	Encoding
0	A
1	B
2	C
...	...
60	8
61	9
62	+
63	/




```
Tekst źródłowy:      Kiedyś to będzie brzmiało jaśniej
Base64:              S2llZHm2IHRvIGLqZHppZSBicnptaWGzbyBqYbZuaWVqDQo=
```

https://www.base64encode.org/

Kodowanie nagłówków w SMTP wykorzystując MIME

Kodowanie wartości nagłówków

tzw. encoded word

Format: `=?charset?kodowanie?zakodowany_tekst?=`

Przykład: `Subject: =?iso-8859-2?Q?Czy_to_jest_zrozumia=EAe?=`

Q – quoted-printable
B – base64

Opis zawartości treści w MIME

MIME określa format pliku (jego zawartość), który będzie przesyłany jako treść (załącznik), np.

text/html,

image/jpg,

application/pdf.

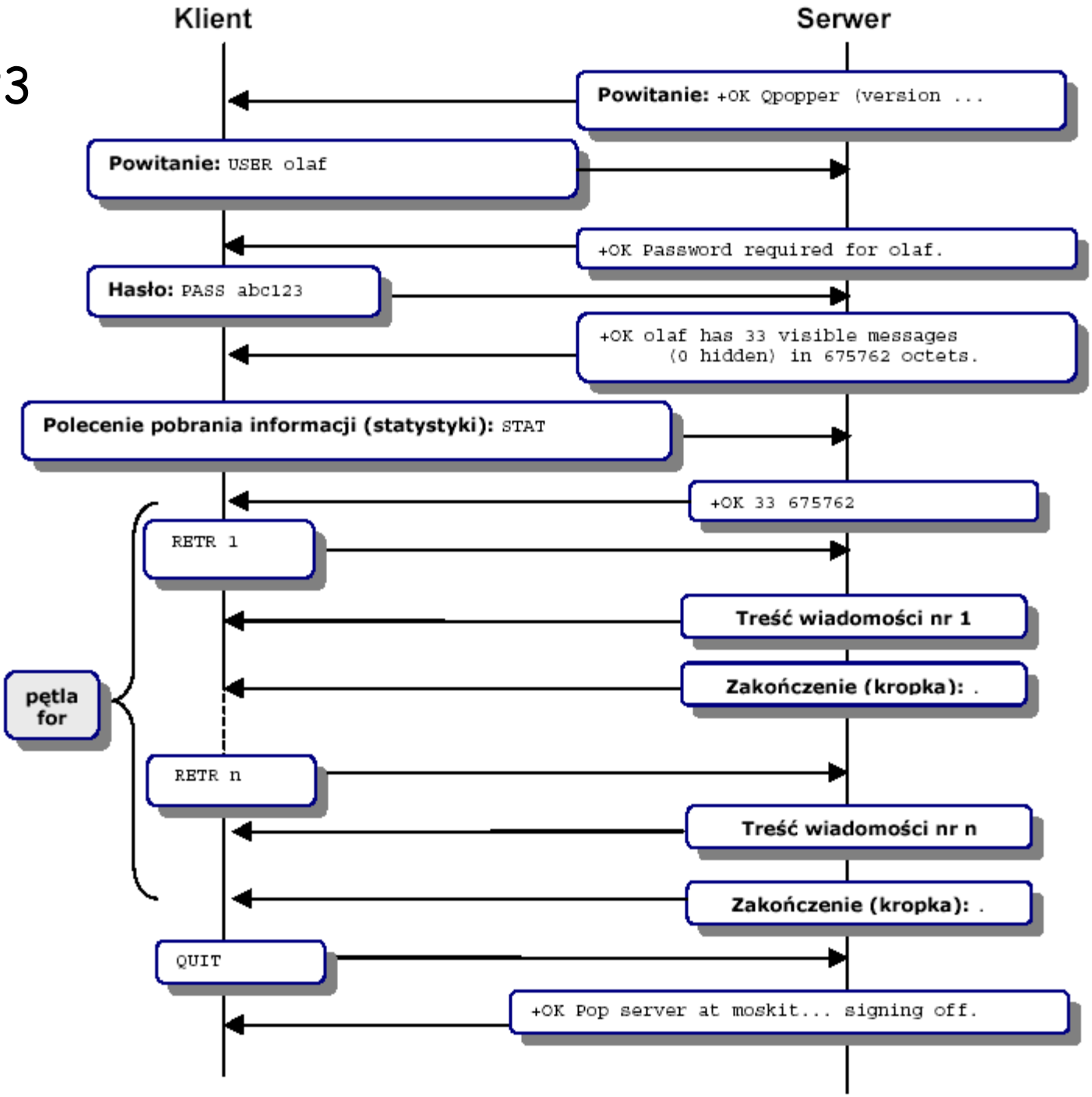
Aplikacja po stronie serwera/klienta musi określić poprawny typ MIME, który jest przesyłany

```
From: ja@domena.pl
To: ty@domena.pl
Subject: Test
MIME-Version: 1.0

Content-Type: multipart/mixed; boundary="ToJestSeparator"
This is a message with multiple parts in MIME format.
--ToJestSeparator
Content-Type: multipart/alternative; boundary=„SeparatorZagniezdzony"
--SeparatorZagniezdzony
Content-Type: text/plain; charset="iso-8859-2" Content-Transfer-Encoding: quoted-printable
To jest tre=B6=E6
--SeparatorZagniezdzony
Content-Type: text/html; charset="iso-8859-2" Content-Transfer-Encoding: quoted-printable
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> <HTML><HEAD> <META http-equiv=3DContent-Type content=3D"text/html; charset=3Diso-8859-2"></HEAD> <BODY><FONT face=3DArial size=3D2>To jest tre=B6=E6</FONT></BODY></HTML>
--SeparatorZagniezdzony--
--ToJestSeparator
Content-Type: image/gif; name="obrazek.gif" Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename="obrazek.gif"
QXBsaWthY2plIGludGVybmV0b3dlIHOxIHRoZSBiZXN0Lg==
--ToJestSeparator--
```

Przykład - 1317817977.M869390P17179V0000000000000904I00000000004D812D_0.eclipse,S=2219868%3A2,S

POP3


Klient
Serwer
Powitanie: +OK Qpopper (version ...
Powitanie: USER olaf
+OK Password required for olaf.
Hasło: PASS abc123
+OK olaf has 33 visible messages
(0 hidden) in 675762 octets.
Polecenie pobrania informacji (statystyki): STAT
+OK 33 675762
RETR 1
Treść wiadomości nr 1
Zakończenie (kropka): .
pętla for
RETR n
Treść wiadomości nr n
Zakończenie (kropka): .
QUIT
+OK Pop server at moskit... signing off.

IMAP

Internet Message Access Protocol

Bardziej zaawansowany niż POP

Umożliwia zarządzanie folderami i plikami na serwerze pocztowym

Lepszy niż POP, gdy użytkownik pracuje na różnych maszynach

Możliwe pobieranie tylko nagłówków a nie treści całej wiadomości

Znalazł zastosowanie w oprogramowaniu dla list dyskusyjnych (NNTP)

RFC 2060 (RFC 1730)

Różnice IMAP – POP 3

Protokół POP wymaga , aby w tym samym czasie do danego konta pocztowego podłączony był jeden klient.

IMAP pozwalaja wykryć zmiany dokonane przez inne podłączone w tym samym czasie klienty.

IMAP umożliwia pobieranie wskazanych części wiadomości elektronicznej, niekoniecznie całej wiadomości.

Protokół IMAP implementuje system flag w taki sposób, że każdy z podłączonych klientów widzi zmiany statusów dokonane przez inne klienty.

Opcje wyszukiwania i nawigacji po folderach na serwerze.

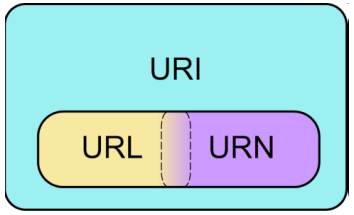
Pojęcie URI, URL i URN

URL – Uniform Resource Locator
Ustandaryzowana nazwa **adresu dla zasobu** (dokumentu, obrazu, pliku …) w sieci.

[RFC2396, RFC1738] - podzbiór URI

URL składa się z dwóch części:
[schemat protokołu] [separator np. ://] [określenie zasobu specyficzne dla schematu]

http://www.imdb.com
ftp://astro.caltech.edu



URI [RFC 2369] jest elementem, który wskazuje (**w sposób szczególny**) na zasób w internecie. Może to być nazwa, adres lub obydwa atrybuty.

URI jest, zazwyczaj krótkim, łańcuchem znaków, zapisanym zgodnie z określoną w standardzie składnią. Łańcuch ten określa nazwę lub adres zasobu, którego dotyczy dany URI.

URI składa się z URL(adresu, lokalizacji) i/lub URN(nazwa) .

URN (Uniform Resource Name) jest to URI, który wskazuje na przestrzeń (namespace) zasobu, bez określenia gdzie się on znajduje i jaki jest schemat dostępu (NID i NSS) np. nazwa książki.

Typy URL

Zdefiniowane są dwa typy URL: ogólny i względny

Składnia ogólnego URL

```
protokół://lokalizacja_serwera:port/ścieżka_na_serwerze?parametry
http://my.host:1234/resource?p1=1&p2=2

protokół://użytkownik:hasło@LokalizacjaSerwera:port/ścieżkaNaSerwerze
ftp://nobody:unknown@ftp.site/index.html

schemat:użytkownik@lokalizacja_serwera
mailto:user@my.mail

schemat://user:pass@adres.com:8042/gdzies/index.php?param=animal&name=mysz#paragraf \ /
\______/\_________/ \__/\_________/ \___/ \_/ \_________/ \_________/ \__/
scheme userinfo hostname port path filename extension parameter(s) query fragment
```

Pominięcie niektórych części (np. portu) powoduje przyjęcie rozwiązań domyślnych

Względny URL [RFC2396, RFC1808] - w zasadzie odpowiada lokalizacji zasobu na serwerze, z dopuszczeniem znaków specjalnego znaczenia "." i ".."

Hypertext Transfer Protocol (HTTP)

Protokół warstwy aplikacyjnej

Domyślny port przyjmuje wartość 80

Protokół typu żądanie-odpowiedź (request/response) pomiędzy klientami i serwerami

Klienty: zazwyczaj przeglądarki

Serwery www (HTTP servers, Web servers): oprogramowanie, które zazwyczaj działa jako tzw. demon (daemon), oczekuje na żądania od klientów i przesyła w odpowiedzi dokumenty z serwera.

Bezstanowy (stateless)

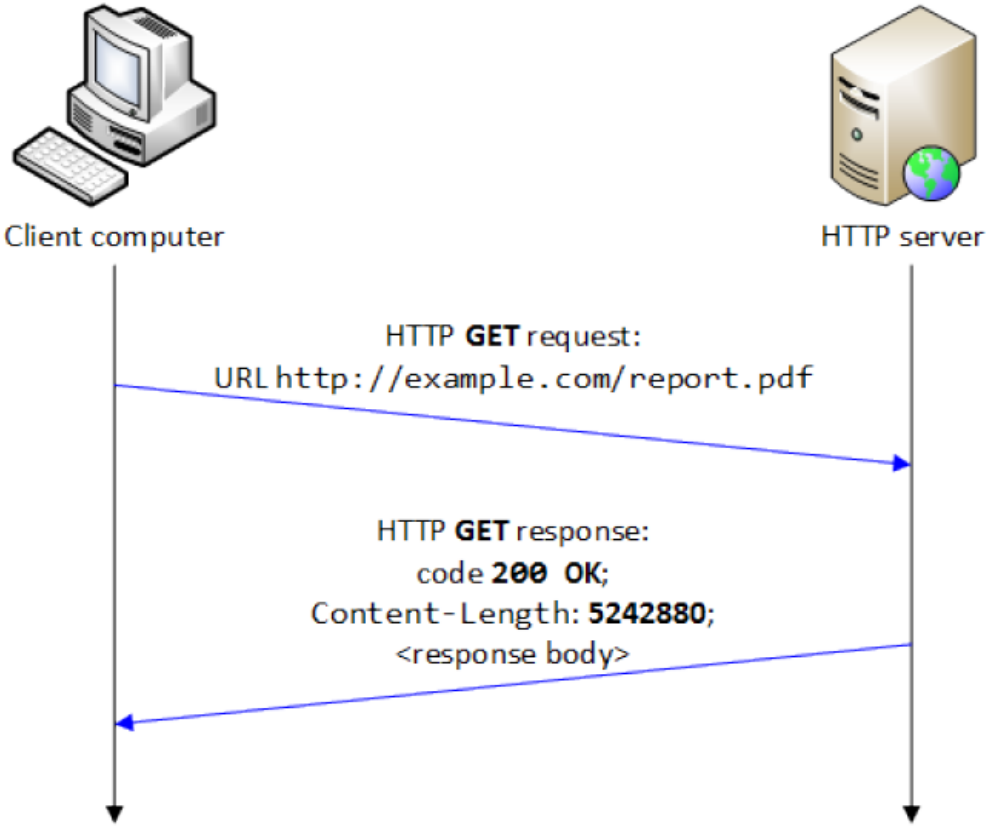
Przesyłane komunikaty mają określoną strukturę.

Zarówno żądanie i odpowiedź składają się z:

- Komendy (z dokładnie określonym formatem)
- Nagłówków (niekiedy opcjonalne)
- Opcjonalna treść oddzielona od komendy i nagłówków pojedynczą pustą linią (tzw. entity body)

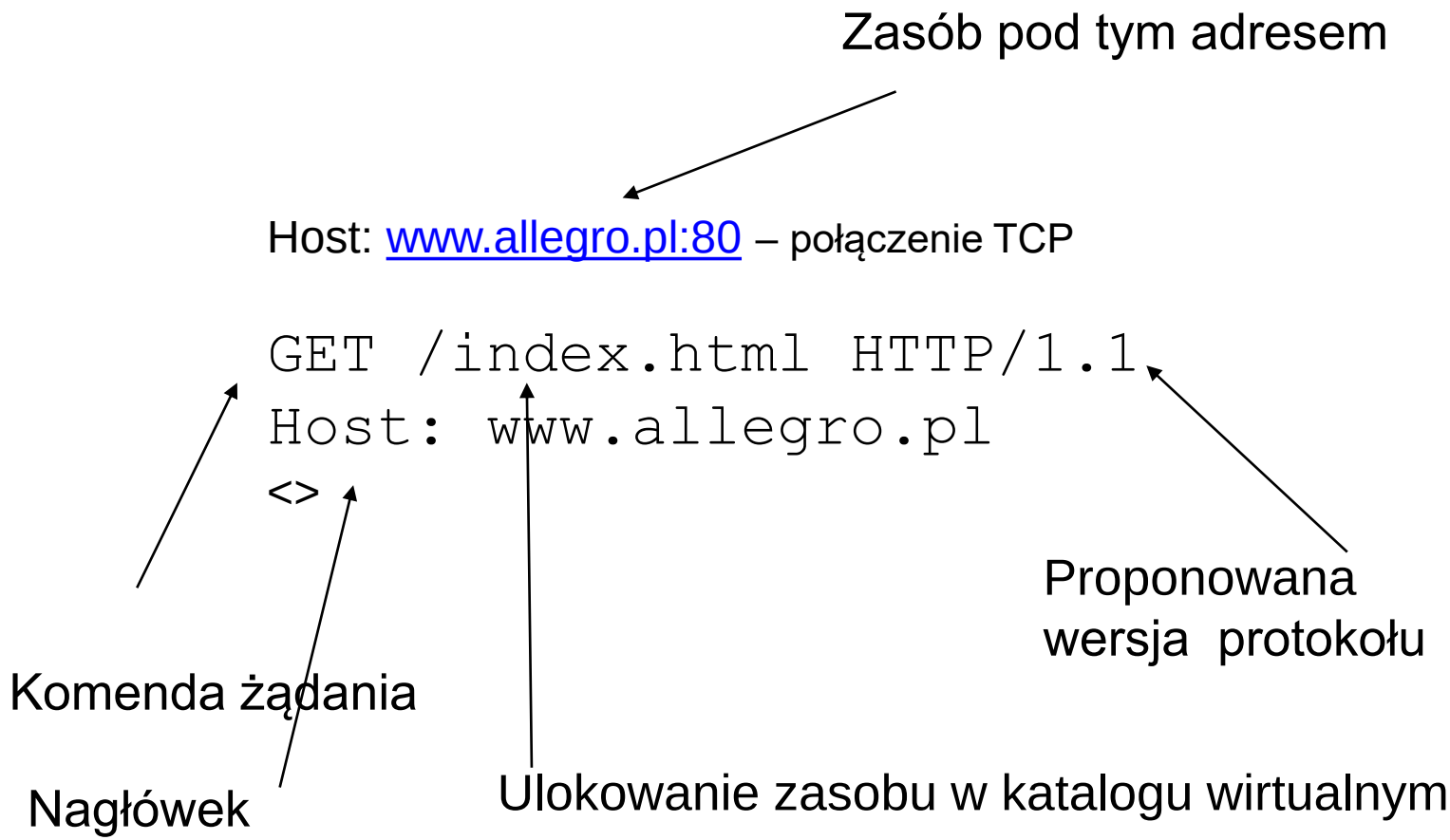
POLITECHNIKA GDAŃSKA

Proces pobierania zasobu z sieci



Na bazie: https://zoompf.com/blog/2010/03/performance-tip-for-http-downloads,

Żądanie klienta



POLITECHNIKA GDAŃSKA

Typy komend

- GET - uzyskanie zasobu o podanym URI
- POST – (od HTTP 1.0) przesłanie danych do zasobu o podanym URI, dane zawarte są w treści żądania
- HEAD - (od HTTP 1.0) jak GET, ale w odpowiedzi uzyskujemy sam nagłówek
- LINK - (tylko HTTP 1.0) i UNLINK - (tylko HTTP 1.0) – tworzy relacje między zasobami

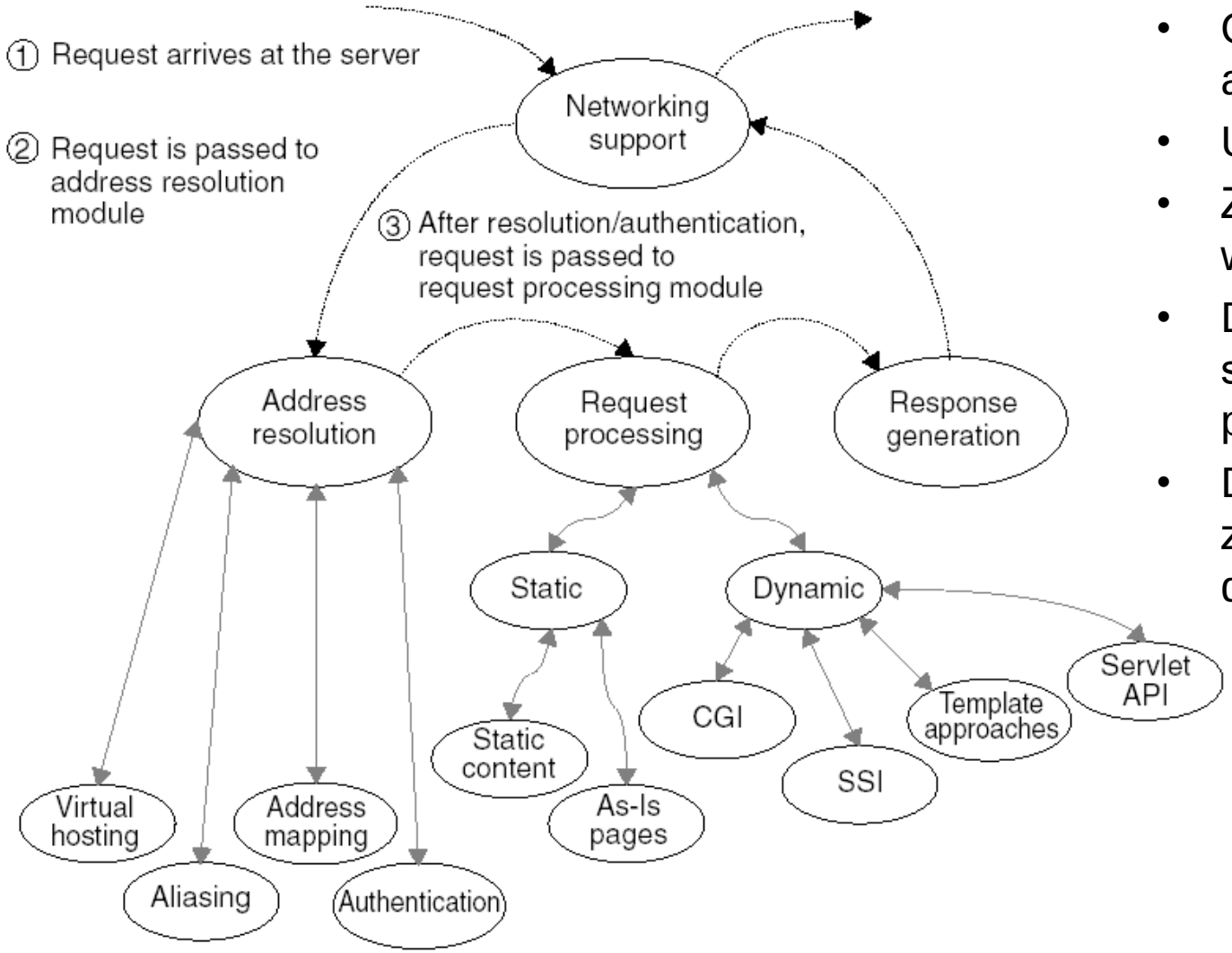
- OPTIONS (od HTTP 1.1) - uzyskanie informacji o metodach akceptowalnych przez dany zasób lub serwer
- PUT (od HTTP 1.1) - umieszczenie na serwerze zasobu w podanym URI, zawartość zasobu określona jest przez treść żądania
- DELETE (od HTTP 1.1) - usunięcie z serwera zasobu o podanym URI
- TRACE (od HTTP 1.1) - wykonanie rodzaju testu przesłania wiadomości w pętli zwrotnej na samym serwerze - odpowiedź do faktycznego klienta zawiera to co klient przesłał
- CONNECT (od HTTP 1.1) – używana przez proxy do tworzenia tuneli

POLITECHNIKA GDAŃSKA

Komenda GET i POST

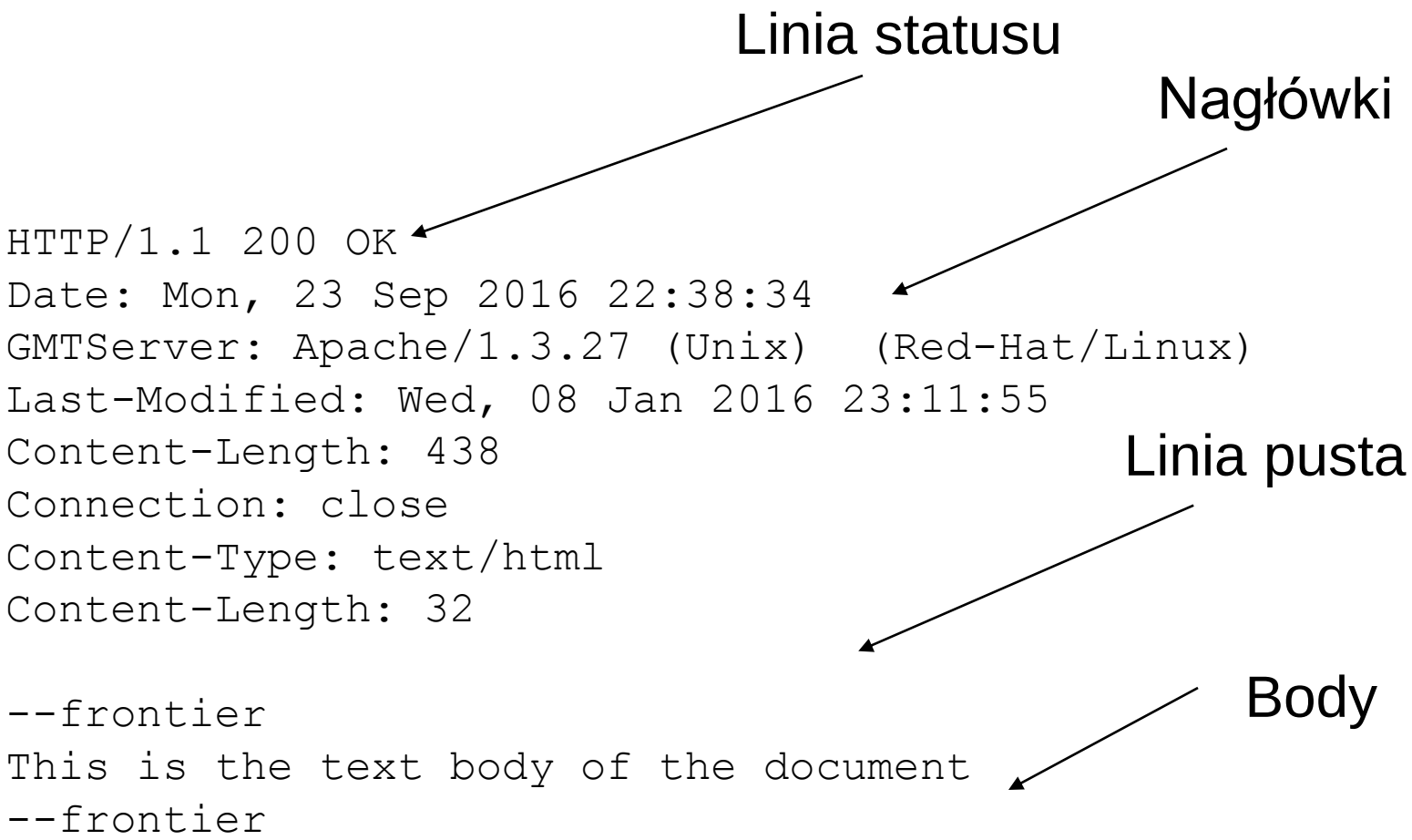
- GET służy do pobrania określonego zasobu (zazwyczaj pierwsze żądanie)
  - Razem z URL można dołączyć dodatkowe parametry
    http://server/file?variable=val&variable2=val
  - Większość serwerów www ma limit długości żądania (od 1024 – 4096 znaków)
  - Format przesyłanych danych to URL encode - symbole zastępowane %XX (np. %2E), spacje zamieniane na + (w starszych wersjach)
- POST - brak ograniczenia z GET
  - Wysyłanie dodatkowych informacji (paramterów) oprócz właściwego żądania
  - Przesyłane informacje są w kodowane w wybranym standardzie MIME lub URL encode

Podstawowe operacje serwera HTTP/WWW



- Odwzorowywanie adresu
- Uwierzytelnianie
- Zarządzanie wirtualnymi hostami
- Dostarczanie statycznej zawartości pliku
- Dostarczanie zawartości dynamicznej

Odpowiedź serwera



POLITECHNIKA GDAŃSKA

Format odpowiedzi

- Pierwsza linia odpowiedzi ma postać

WERSJA_HTTP KOD_ODPOWIEDZI_(STATUSU) PRZYCZYNA_ODPOWIEDZI

  - Wersja HTTP: wersja obsługiwana przez serwer, nie musi odpowiadać wersji w żądaniu, ciąg znaków HTTP/1.0 lub HTTP/1.1
  - Kod odpowiedzi: numeryczny, trzyznakowy, pierwsza cyfra określa klasę odpowiedzi
  - Przyczyna odpowiedzi: krótki ciąg znaków wyjaśniający zwrócenie takiego, a nie innego kodu odpowiedzi

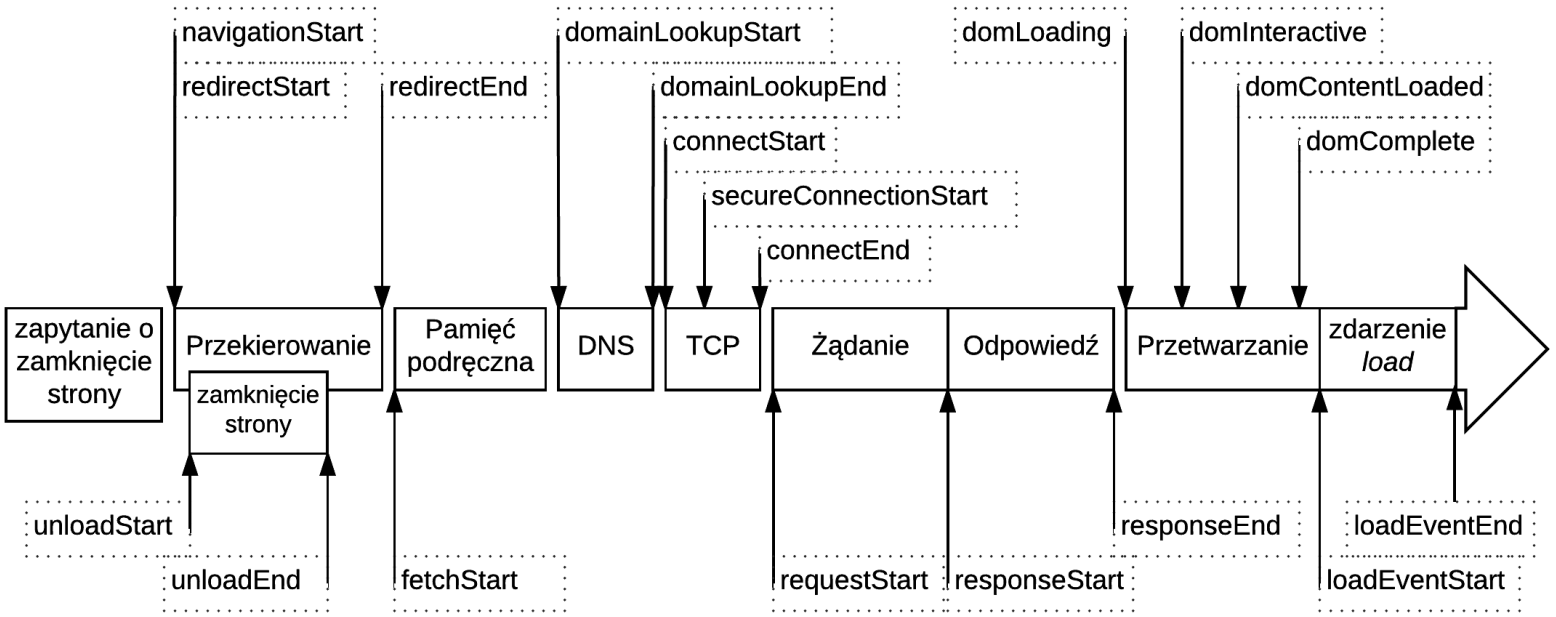
POLITECHNIKA GDAŃSKA

Klasy/kody odpowiedzi

- 1xx: klasa informacyjna, żądanie zostało odebrane i jego przetwarzanie jest kontynuowane, np. 100 - kontynuacja procesu
- 2xx: klasa odpowiedzi pozytywnych, żądanie zostało poprawnie odebrane, zrozumiane i zaakceptowane np. 200 - po prostu OK
- 3xx: klasa przekierowania, należy podjąć dodatkową czynność, aby ukończyć żądanie, np. 301 - zasób przeniesiony na stałe, 302 - zasób przeniesiony czasowo, 304 - brak modyfikacji
- 4xx - klasa błędów klienta, żądanie zostało niepoprawnie sformułowane lub nie
- może zostać spełnione np. 400 - nieprawidłowe zapytanie, 403 - dostęp do zasobu zabroniony, 404 - zasób nie znaleziony
- 5xx - klasa błędów serwera, wystąpił błąd po stronie serwera podczas przetwarzania poprawnego żądania. np. 500 - wewnętrzny błąd serwera, 501 - metoda niezaimplementowana

POLITECHNIKA GDAŃSKA

Typowe etapy przetwarzania HTTP



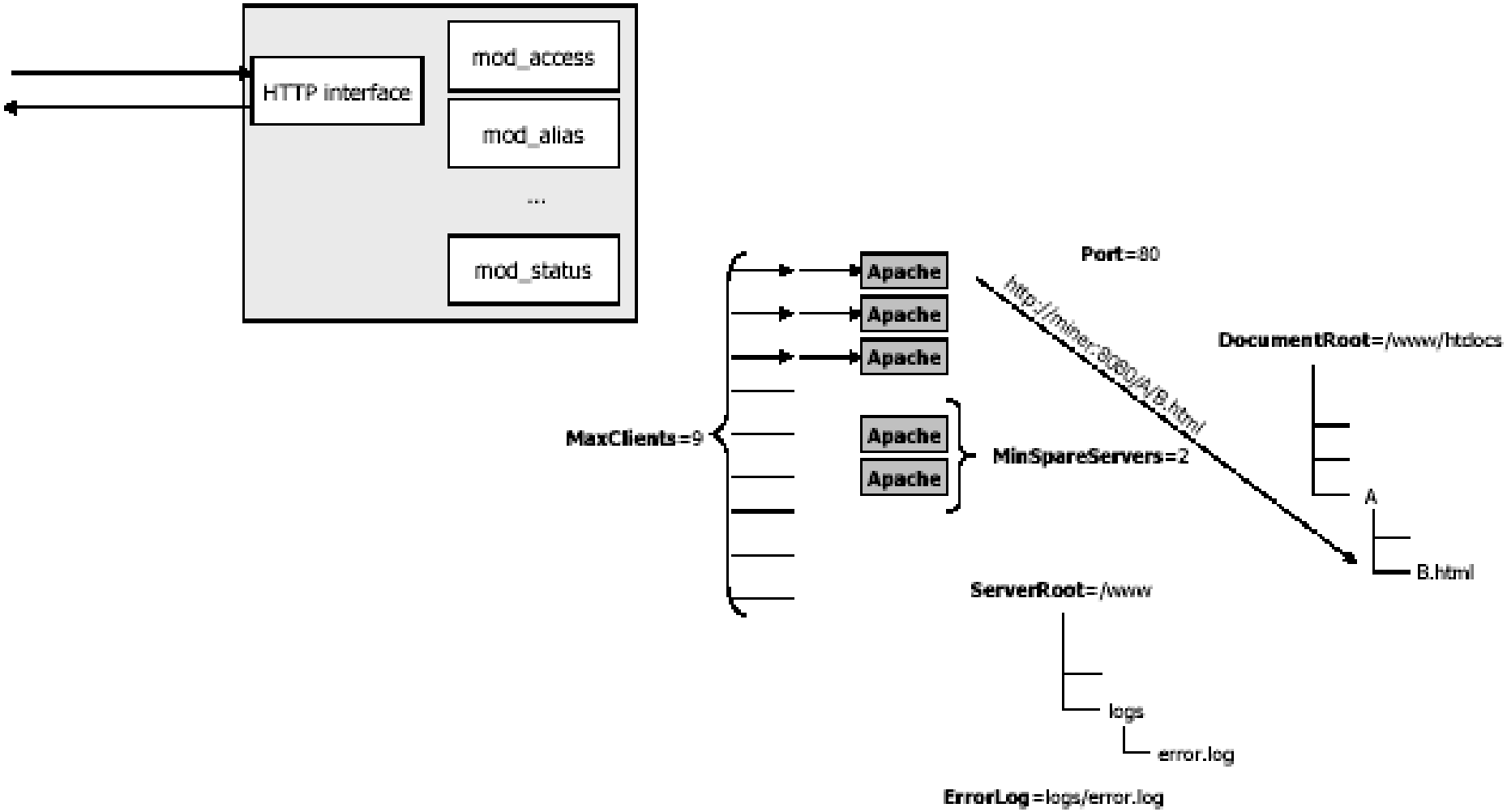
POLITECHNIKA GDAŃSKA

Różnice w wersjach protokołu HTTP

- trwałe połączenia
- potokowość z ich wykorzystaniem trwałych połączeń,
- zniesienie limitu dwóch jednoczesnych połączeń z danym serwerem
- dopracowany mechanizm pamięci podręcznej cache,
- mechanizm wirtualnych hostów - serwowanie innych zasobów w zależności od nazwy hosta w polu nagłówkowym Host z jednego adresu IP
- przesyłanie wiadomości o nieznanym z góry rozmiarze,
- przesyłanie fragmentów zasobu,
- porzucenie wsparcia dla HTTP/0.9
- nowe komendy
- URI zasobu ogólny lub względny w HTTP 1.0, w 1.1 tylko względny (ogólny może wystąpić tylko w przypadku łączenia przez serwer proxy)

POLITECHNIKA GDAŃSKA

Budowa modułowa serwera Apache



Przykładowa konfiguracja wirtualnego hosta

1. http://www.neurozen.com/index.html

2. http://www.neurozen.com/test?a=1&b2

3. http://www.neurozen.com/images/news.gif

```
<VirtualHost www.neurozen.com>
        ServerName www.neurozen.com
        ServerAdmin webmaster@neurozen.com
        Alias /test /servlet/test
        Alias /images /static/images
        DocumentRoot /www/docs/neurozen
                ErrorLog logs/neurozen-error-log
        CustomLog logs/neurozen-access-log common
</VirtualHost>
```

Dokumentacja HTTP

- HTTP 0.9, HTTP 1.0 RFC1945
- HTTP 1.1 (RFC 2616)
  - RFC7230 — HTTP/1.1 Message Syntax and Routing,
  - RFC7231 — HTTP/1.1 Semantics and Content,
  - RFC7232 — HTTP/1.1 Conditional Requests,
  - RFC7233 — HTTP/1.1 Range Requests,
  - RFC7234 — HTTP/1.1 Caching,
  - RFC7235 — HTTP/1.1 Authentication.
- HTTP 2.0
  - RFC7540 — Hypertext Transfer Protocol Version 2 (HTTP/2)
  - RFC7541 — HPACK: Header Compression for HTTP/2

Wybrane aspekty HTTP - Nagłówki

Wyróżnia się 4 kategorie nagłówków (1.1)

- ogólne (ang. general header fields), które maja¸ zastosowanie zarówno dla żądania jak i odpowiedzi: `Date, Pragma`,
- dotyczące żądania (ang. request header fields), które pozwalają przekazać serwerowi dodatkowe informacje na temat żądania i klienta: `Authorization, From, If-Modified-Since, Referer, User-Agent`,
- dotyczące odpowiedzi (ang. response header fields), które pozwalają przekazać klientowi dodatkowe informacje na temat serwera i lokalizacji zasobu: `Location, Server, WWW-Authenticate`,
- dotyczące zasobu (ang. entity header fields), które definiują opcjonalne metainformacje: `Allow, Content-Encoding, Content-Length, Content-Type, Expires, Last-Modified`.

Wybrane aspekty HTTP - Nagłówki

- **Host** Żądanie (HTTP 1.1)
  - Nazwa i ewentualnie port serwera. Wymagane przy łączeniu bezpośrednim, bez proxy.
- **Location** Odpowiedź
  - URI zasobu w przypadku przekierowania.
- **Content-Type** Żądanie/Odpowiedź
  - Typ MIME przesyłanej treści
- **Content-Length** Żądanie/Odpowiedź
  - Długość w bajtach przesyłanej treści
- **Accept** Żądanie
  - Specyfikacja typów MIME zasobów akceptowalnych przez klienta
- **Allow** Odpowiedź
  - Lista metod akceptowalnych przez serwer lub zasób
- **User-Agent** Żądanie
  - Identyfikator przeglądarki
- **Referer** Żądanie
  - URI zasobu, z poziomu którego wystąpiło żądanie
- **Connection** Żądanie/Odpowiedź (HTTP 1.1)
  - Rodzaj połączenia: Keep-Alive lub Close

Wybrane aspekty HTTP – Wiadomość porcjowana

Nagłówek: Transfer-Encoding: chunked

- Żądanie/Odpowiedź HTTP 1.1
- Określa "porcjowane" kodowanie przesyłanej wiadomości. Wiadomość składa się z porcji (chunks), z których każda poza ostatnią jest postaci:

rozmiar[CRLF]

dane[CRLF]

Ostatnia porcja jest postaci:

0[CRLF]

opcjonalna stopka[CRLF]

[CRLF]

Wybrane aspekty HTTP – Wiadomość porcjowana

```
HTTP/1.1 200 OK
Date: Thu, 06 May 2005 17:35:41 GMT
Server: Apache/1.3.23 (Unix) mod_ssl/2.8.7 OpenSSL/0.9.6c PHP/4.1.0 mod_perl/1.26
X-Powered-By: PHP/4.1.0
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html

7e
<html>
<head>
<title>Params</title>
</head>
<body>
<p>Params:</p>
<div>param1 = 1</div>
<div>param2 = 2</div>
</body>
</html>
0
```

Nagłówek we wszystkich odpowiedziach (za wyjątkiem 1xx)

Żądanie

```
If-Modified-Since:  Fri, 31 Dec 1999 23:59:59 GMT
If-Modified-Since:  Friday, 31-Dec-99 23:59:59 GMT
If-Modified-Since:  Fri Dec 31 23:59:59 1999
```

Odpowiedź klasy 200 lub 304

```
HTTP/1.1 304 Not Modified
Date: Fri, 31 Dec 1999 23:59:59 GMT
[blank line]
```

Bezpieczny HTTP, czyli HTTPS

Bezpieczna wersja HTTP-S (od Secure).

Kodowanie sesji poprzez użycie protokołów SSL (Secure Socket Layer) lub TLS (Transport Layer Security).

Domyślny port 443

SSL (Secure Socket Layer) - protokół kryptograficzny opracowany przez Netscape
W ramach sesji SSL używa się kryptografii przy pomocy algorytmów asymetrycznego (para kluczy: publiczny i prywatny) oraz symetrycznego (jeden klucz)

Wszystkie dane przed wysłaniem kodowane są własnym kluczem symetrycznym

Bezpieczeństwo transmisji zależy głównie od rozmiaru kluczy. Za wystarczające uznaje się (jeszcze) 128 bitów dla klucza sesji i 1024 dla klucza publicznego

Komunikacja HTTPS

Przebieg sesji połączenia przez SSL:

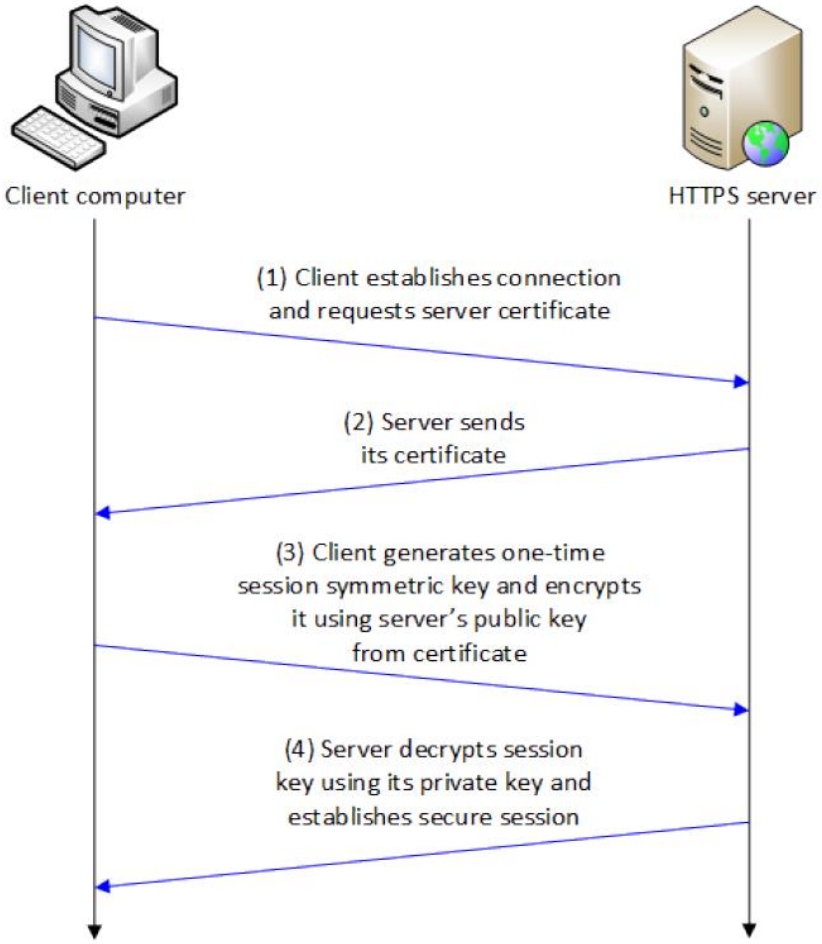
1. Wymiana certyfikatów (CA) pomiędzy klientem i serwerem. CA zawiera uwierzytelnione informacje o właścicielach i klucze publiczne. Przeglądarka może podać własny CA, jeżeli brakuje CA użytkownika

2. Uzgodnienie algorytmu kryptograficznego opartego o klucz symetryczny - wybierany jest zawsze najmocniejszy algorytm znany obu stronom

3. Wygenerowanie osobno przez klienta i serwer tzw. symetrycznego klucza sesji

4.Wymiana kluczy sesji zakodowanych kluczem publicznym "rozmówcy"

5. Odkodowanie kluczy sesji przy pomocy własnego klucza prywatnego i rozpoczęcie wymiany danych



Na bazie: https://zoompf.com/blog/2010/03/performance-tip-for-http-downloads,

Protokół HTTP 2.0

Bazuje na protokole SPDY, opracowanym przez Google

Opublikowany w RFC 7540, w maju 2015

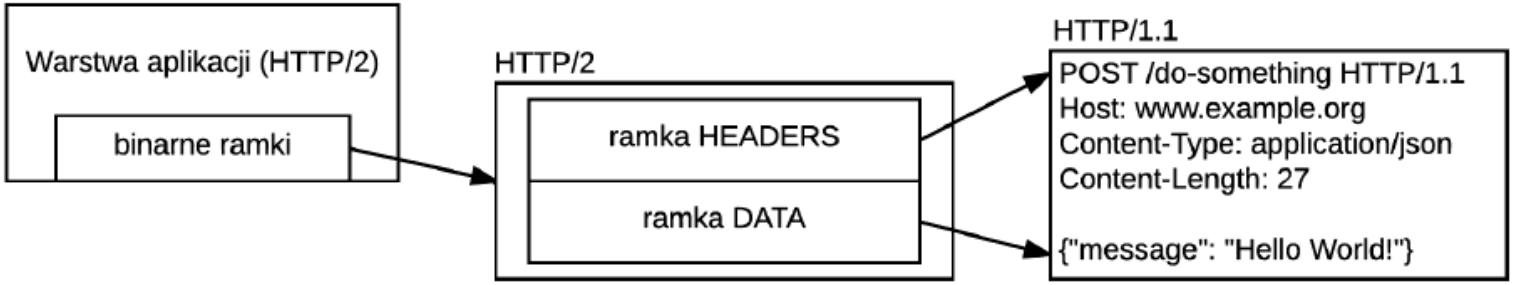
Obecnie ok. 10% serwisów korzysta z niego (09.2016)

Zachowuje wysoką kompatybilność z 1.1 (komendy, statusy, większość nagłówków, adresy URI)

Zmniejszenie opóźnienia w ładowaniu stron poprzez

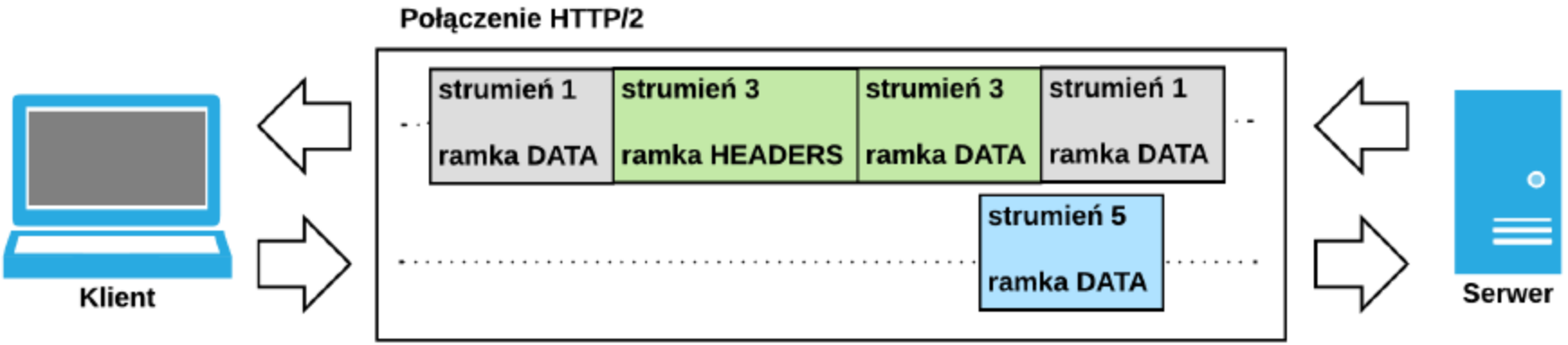
- kompresję danych w nagłówkach
- rozwiązanie blokowania pierwszego żądania
- ładowanie elementów w sposób jawnie równoległy
- wykorzystanie pojedynczego połączenia na poziomie gniazd do transmisji wielu żądań i odpowiedzi.
- wykorzystanie protokołu UDP
- wykorzystanie mechanizmu push

Ramka HTTP 2.0



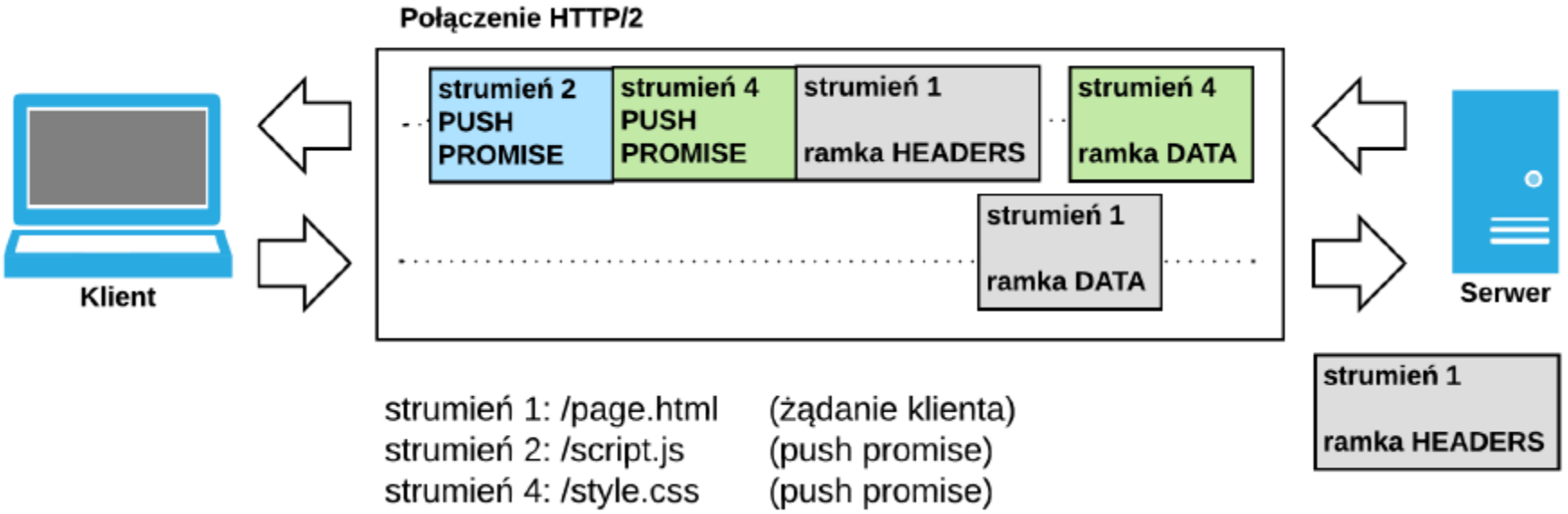
Źródło: na podstawie High Performance Browser Networking,Ilya Grigor, O'Reilly Media, 2013

Przesyłanie ramek w obrębie dwukierunkowego połączenia



Źródło: na podstawie High Performance Browser Networking,Ilya Grigor, O'Reilly Media, 2013

Mechanizm *server-push*

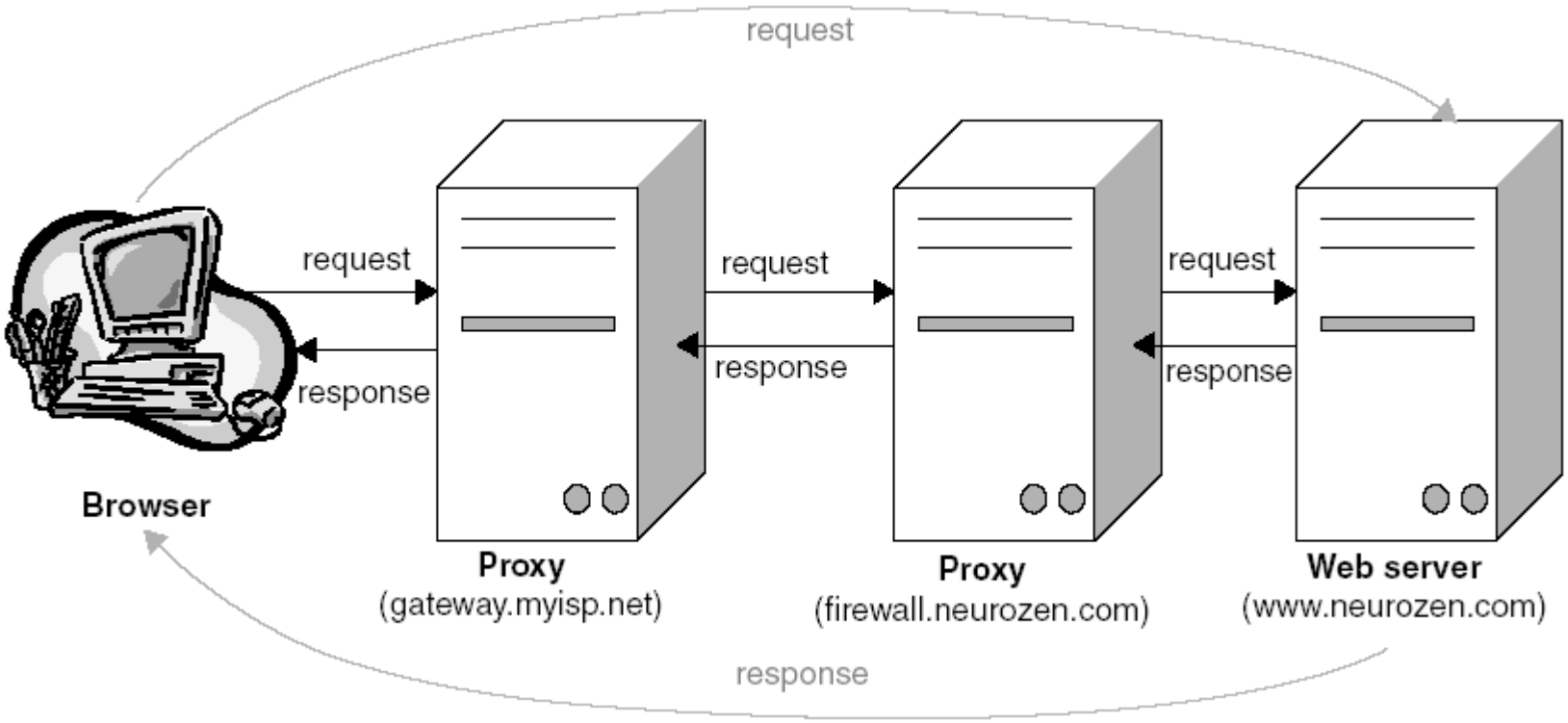


Źródło: na podstawie High Performance Browser Networking,Ilya Grigor, O'Reilly Media, 2013

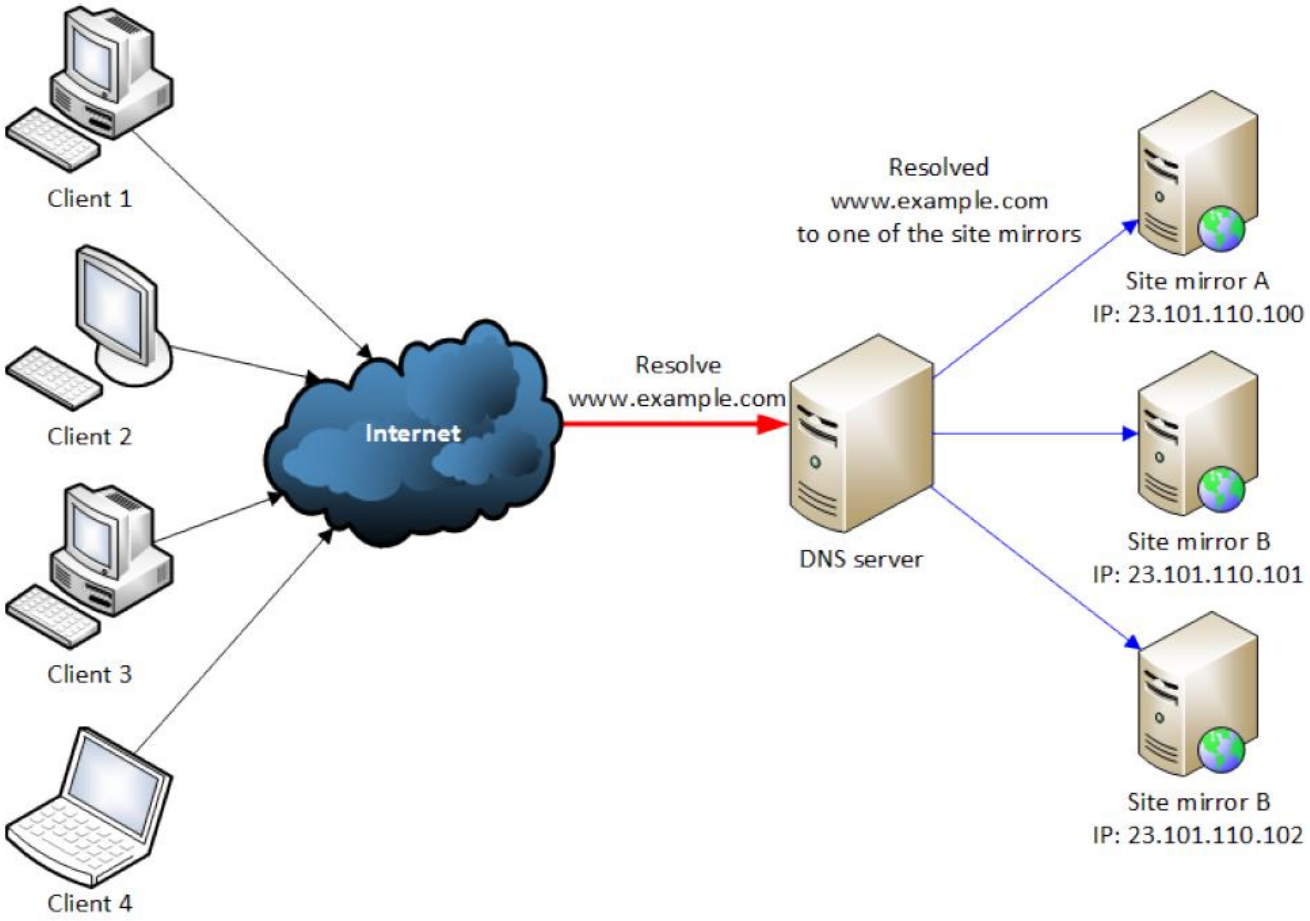
Zestawienie różnic między HTTP1.0 a HTTP 2.0

cecha	**HTTP/1.1**	**HTTP/2**
format	tekstowy	binarny
kompresja nagłówków	nie	tak
liczba jednoczesnych połączeń z serwerem	1-8	1
efektywne wykorzystanie TCP	nie	tak
priorytetyzacja i multipleksacja	częściowa (występuje problem *head-of-line blocking*)	pełna
wymagane szyfrowane połączenie	nie	nie, w praktyce tak
wysyłanie przez serwer dodatkowych zasobów	nie	tak, *server push*
liczba RTT wymagana do nawiązania połączenia	3	3 dla TCP; 1 (często 0) dla QUIC

Rzeczywisty kanał komunikacyjny



Mirror-sites



Serwer to punkt centralny systemu rozproszonego - wrażliwość na przeciążenia => ataki typu DoS (Denial of Service)/DDoS

Przykłady szczegółowych implementacji

Server	Security				Dynamic content[a]								
	basic	digest	SSL/TLS	virtual hosting	CGI	FCGI	Java servlet	SSI	ISAPI	Runs in user space or kernel space	Administration console	IPv6	HTTP/2
AOLserver	Yes	No	Yes[b][c][d][1]	Yes	Yes	No	No	Yes	Unknown	user	Unknown	Unknown	Unknown
Apache HTTP Server	Yes	Yes	Yes[e][c][2][f][3]	Yes	Yes	Yes	No[g]	Yes	Yes[h]	user	Yes[i]	Yes	Yes
Apache Tomcat	Yes	Yes	Yes[j][4]	Yes	Yes	No	Yes	Yes	No[k]	user	Yes	Yes[l]	Unknown
Internet Information Services	Yes	Yes	Yes	Yes	Yes	Yes	No[r]	Yes	Yes	kernel and user[8]	Yes	Yes	Yes
Jetty	Yes	Yes	Yes	Yes	Yes	Unknown	Yes	Unknown	Unknown	user	Unknown	Unknown	Yes
lighttpd	Yes	Yes	Yes[c][9]	Yes	Yes	Yes	No[g]	Yes	No	user	No	Yes	Unknown
nginx	Yes	Yes (module)	Yes	Yes	No	Yes	No[10]	Yes	No	user	Yes[11]	Yes[12]	Yes[13]

Na bazie: https://en.wikipedia.org/wiki/Comparison_of_web_server_software

Tryby pracy serwera WWW

- Przetwarzanie typu wejście/wyjście - serwer WWW przekazuje do modułu oprogramowania (rozszerzenie) żądanie klienta i traktuje wynik działania modułu jako pełną odpowiedź przekazywaną klientowi

- Przetwarzanie wsadowe - serwer WWW przekazuje klientowi statyczny zasób (np. kod HTML), przetwarzając tylko niektóre, specjalnie oznaczone fragmenty; wynik przetwarzania umieszczany jest w ramach zasobu zamiast danego fragmentu

- CGI podstawowy interfejs wymiany danych pomiędzy serwerem WWW i oprogramowaniem; potocznie - moduły w języku Perl, C lub skrypty systemu operacyjnego działające w ramach przetwarzania typu wejście/wyjście

- Serwlety j. Java - poza implementacją CGI - większość możliwości środowiska Java i kilka specyficznych udoskonaleń obecnie część większego standardu J2EE

- ASP - pierwsza poważna technologia przetwarzania wsadowego w zasadzie tylko serwery WWW Microsoft (IIS)
- ASP.NET

- PHP najpopularniejsza i najbardziej dynamicznie rozwijająca się technologia przetwarzania wsadowego pod wieloma względami lepsze od ASP, ale część możliwości zależna od systemu operacyjnego i zewnętrznych bibliotek

- JSP "odpowiedź" j. Java na PHP razem z serwletami tworzą środowisko łatwe do projektowania i "podziału kompetencji"

Mechanizm interfejsu wspólnej bramki CGI

CGI - Common Gateway Interface:

- Pierwszy mechanizm oferowania dynamicznej zawartości

- Mechanizm CGI zakłada, że podczas odbioru żądania jest uruchamiany nowy proces (exec – w unix'ach), do którego zostanie przekazany określony zbiór parametrów

- Typowy scenariusz działania:
  1. odczytanie parametrów żądania (ze standardowego wejścia albo ze zmiennych środowiskowych);
  2. przetworzenie danych;
  3. wyprowadzenie komunikatu http (na standardowe wyjście).

CGI – zmienne środowiskowe

Dwie grupy:

Związane z serwerem i żądaniem

- REMOTE_ADDR, REMOTE_HOST, REMOTE_USER
- QUERY_STRING
- SERVER_SOFTWARE, SERVER_NAME, SERVER_PORT
- SERVER_PROTOCOL, GATEWAY_INTERFACE
- REQUEST_METHOD (np. POST lub GET)
- PATH_INFO (virtual) PATH_TRANSLATED, SCRIPT_NAME

Związane z nagłówkami protokołu HTTP

- HTTP_AUTH_TYPE (zastosowana metoda uwierzytelniania)
- HTTP_CONTENT_TYPE, HTTP_CONTENT_LENGTH
- HTTP_ACCEPT_LANGUGE, HTTP_ACCEPT, HTTP_USER_AGENT
- HTTP_COOKIE

CGI – przykład (1)

```
<HTML>
<HEAD><TITLE>Simple Form</TITLE></HEAD>
<BODY>
<H2>Simple Form</H2>
<FORM ACTION=http://serwer.pl/cgi-bin/zip.cgi  METHOD="post">
Zip Code: <INPUT SIZE="5" NAME="zip">
Name: <INPUT SIZE="30" NAME="name">
<INPUT TYPE="submit" VALUE="set zip">
</FORM>
<BODY>
</HTML>
```

```
POST http://serwer.pl/cgi-bin/zip.cgi HTTP/1.1
Host: mysite.org
User-Agent: Mozilla/4.75 [pl] (WinNT; U)
Content-Length: 29
Content-Type: application/x-www-form-urlencoded
Remote-Address: 127.0.0.1
Remote-Host: demo

zip=84240&name=Kubus+Puchatek
```

CGI – przykład (1)

```
<HTML>
<HEAD><TITLE>Simple Form</TITLE></HEAD>
<BODY>
<H2>Simple Form</H2>
<FORM ACTION=http://serwer.pl/cgi-bin/zip.cgi  METHOD="post">
Zip Code: <INPUT SIZE="5" NAME="zip">
Name: <INPUT SIZE="30" NAME="name">
<INPUT TYPE="submit" VALUE="set zip">
</FORM>
<BODY>
</HTML>
```

```
POST http://serwer.pl/cgi-bin/zip.cgi HTTP/1.1
Host: mysite.org
User-Agent: Mozilla/4.75 [pl] (WinNT; U)
Content-Length: 29
Content-Type: application/x-www-form-urlencoded
Remote-Address: 127.0.0.1
Remote-Host: demo

zip=84240&name=Kubus+Puchatek
```

CGI – przykład (2)

```
#!/usr/local/bin/perl
sub ReadFormFields { ... }
sub PrintFormFields {
my $fieldsRef = shift;
my $key, $value;
print "Content-Type:
    text/html\n\n";
print
    "<html>\n<head><title>hello</
    title></head>\n";
print "<body>\n";
foreach $key (keys(%$fieldsRef))
    {
$value = $$fieldsRef{$key};
print "<h3>$key: $value</h3>\n";
}
print "</body>\n</html>\n";
}

&ReadFormFields(\%fields);
&PrintFormFields(\%fields);
exit 0;
```

```
sub ReadFormFields
{
# ustawienie parametrów przekazanych do
    ReadFormFields
my $fieldsRef = shift;
my($key, $val, $buftmp, @buf parm);

#odczyt żądania
$buftmp = " ";
read(STDIN,$buftmp,$ENV{'CONTENT LENGTH'});
$buftmp = $ENV{QUERY STRING} if (!$buf tmp);
@bufparm = split(/&/,$buf tmp);

#podział parametrów żądania na pary klucz -
    wartość
foreach $parm (@bufparm) {
    ($key, $val) = split(/=/,$parm);

    # zamiana plusów i znaków specjalnych %XX
    $val =~ s/\+/ /g;
    $val =~ s/%([a-fA-F0-9][a-fA-F0-9])/
    pack("C",hex($1))/ge;

    # użycie \0 do odseperowania pół
    $$fieldsRef{$key} .= '\0' if
    (defined($$fieldsRef {$key}));
    $$fieldsRef{$key} .= $val;
    }
return($fieldsRef);
}
```

CGI - wady

- Ograniczona sprawność działania: konieczność załadowania procesu przy obsłudze każdego żądania stanowi znaczne obciążenie
- Problem współdzielenia zasobów
- Możliwość wywoływania poleceń po stronie serwera tworzy istotne zagrożenie dla bezpieczeństwa
- Przeplatanie kodu programu i znaczników HTML
- Zależny od serwera
- Potrzeba utworzenia sesji internetowej

Server Side Includes (SSI) - idea

- Jak sugeruje nazwa, w tym wypadku kod jest wstawiany w strukturę dokumentu HTML.
- Odpowiednie fragmenty dokumentu są ponownie generowane przy każdym odwołaniu.
- Dynamika dokumentu bez konieczności stosowania CGI; generowane są tylko wymagane fragmenty dokumentu.
- Możliwe zastosowania:
  - Licznik odwołań;
  - Data i czas; data ostatniej modyfikacji dokumentu;
  - Wkomponowywanie innego pliku lub rezultatu wywołania programu CGI.
- Rozszerzenia specyfikacji obejmują m. in.:
  - Instrukcje warunkowe (if … else);
  - Wysyłanie poczty.

Server Side Includes (SSI) - składnia

- Pliki wymagające takiego przetwarzania wyróżniane zwykle specjalnym rozszerzeniem: .sht .shtm lub .shtml.

- Elementy interpretowane umieszczone są w specjalnych znacznikach wewnątrz kodu HTML. Przed wysłaniem zawartość jest przeglądana i osadzone znaczniki są interpretowane. Format znacznika (*token*) SSI jest następujący: <!--*#polecenie zbiór_zmiennych* --> czyli :
  - **<!--#** Identyfikator otwierający znacznik SSI;
  - **polecenie** może przybierać wartości: echo, include, fsize, flastmod, exec, config, oraz (rozszerzenia): if, goto, label, break
  - **zbiór zmiennych** jedna lub więcej para **nazwa_zm = "wartosc"**. Umieszczane tu dane są zależne od kontekstu polecenia.
  - **-->** obowiązkowy identyfiaktor zamykający.

Server Side Includes (SSI) – przykłady poleceń

- **config** - Określa sposoby parsowania plików:
  - errmsg – postać komunikatu o błędzie.
  - timefmt – format wyświetlania czasu.
  - sizefmt – Sposób wyświetlania rozmiaru plików.
- **include** – wstawienie do bieżącego dokumentu innej wersji:
  - virtual – ścieżka wirtualna na serwerze.
  - file – ścieżka względna w głąb bieżącego katalogu.
- **echo** – wyświetlenie wartości podanej zmiennej.
- **fsize** – wyświetla rozmiar pliku.
- **flastmod** – data ostatniej modyfikacji.
- **exec** – wykonanie zadanej komendy systemowej lub skryptu CGI:
  - cmd – uruchomienie podanego łańcucha jako komendy.
  - cgi – uruchomienie zadanego skryptu CGI.

Server Side Includes (SSI) – przykłady

```
<html>
<head><title>hello</title></head>
<body>
<!--#exec cgi http://serwer.pl/cgi-
   bin/zip-ssi.cgi -->
</body>
</html>
```

```
#!/usr/local/bin/perl
sub ReadFormFields { . . . }
sub PrintFormFields
{
my $fieldsRef = shift;
my $key, $value;
foreach $key (keys(%$fieldsRef)) {
$value = $$fieldsRef{$key};
print "<h3>$key: $value</h3>\n";
}
}
&ReadFormFields(\%fields);
&PrintFormFields(\%fields);
exit 0;
```

Internet Server Application Programming Interface (ISAPI)

- Technologia wprowadzona przez firmę Microsoft (inny rodzaj NSAPI).
- Definiuje tzw. rozszerzenia (*extensions*) i filtry (*filters*).
- Zapewnia równie dużą elastyczność jak CGI.

- W przeciwieństwie do CGI proces obsługi rezyduje w pamięci, a nie jest każdorazowo ładowany.

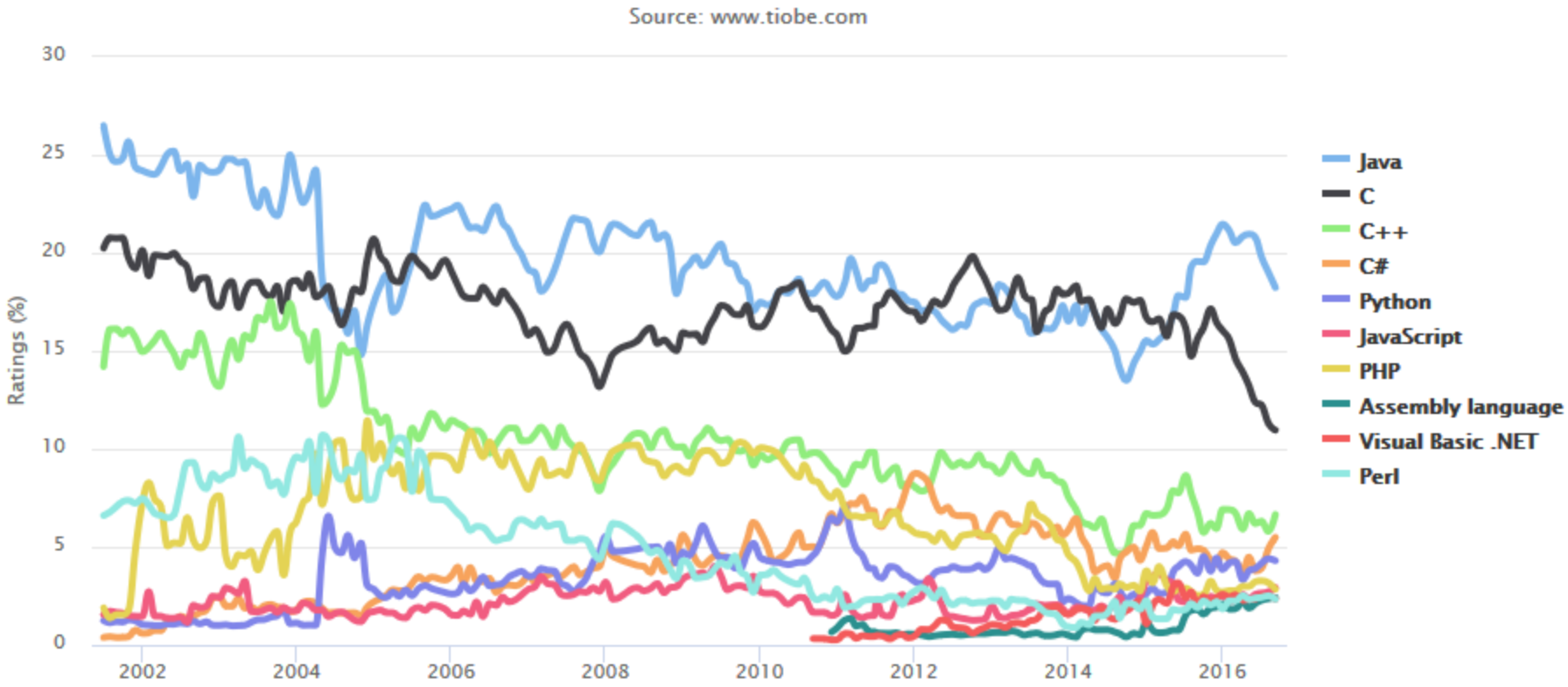
- Tak jak CGI, ISAPI daje możliwość wywoływania funkcji systemowych (API) i innych dostarczonych przez wybrany język programowania.

- Brak przenośność kodu (zależny od API konkretnego dostawcy) – określanej jako *proprietary API*

- Przykład -> plik isapi.cpp

POLITECHNIKA GDAŃSKA

Popularność języków programowania

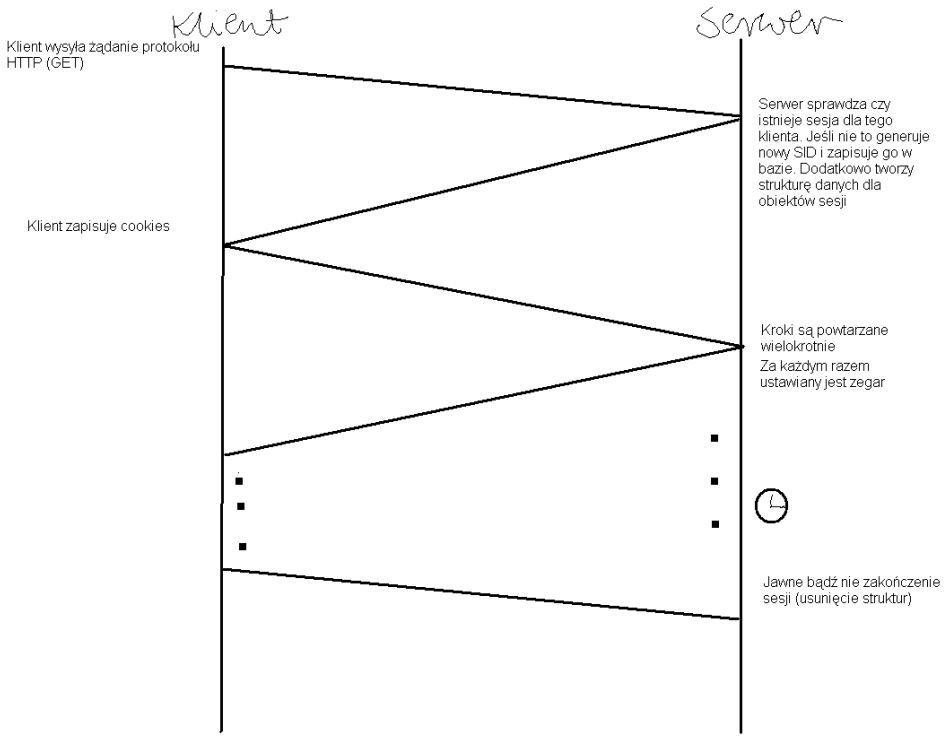


http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Sesja internetowa i nagłówek Cookie

Potrzebna gdy serwer chce śledzić poczynania użytkownika (tzn. jego kolejne żądania)
- Utrzymywanie koszyka zakupów użytkownika
- Zapisywanie informacji o autoryzacji dostępu

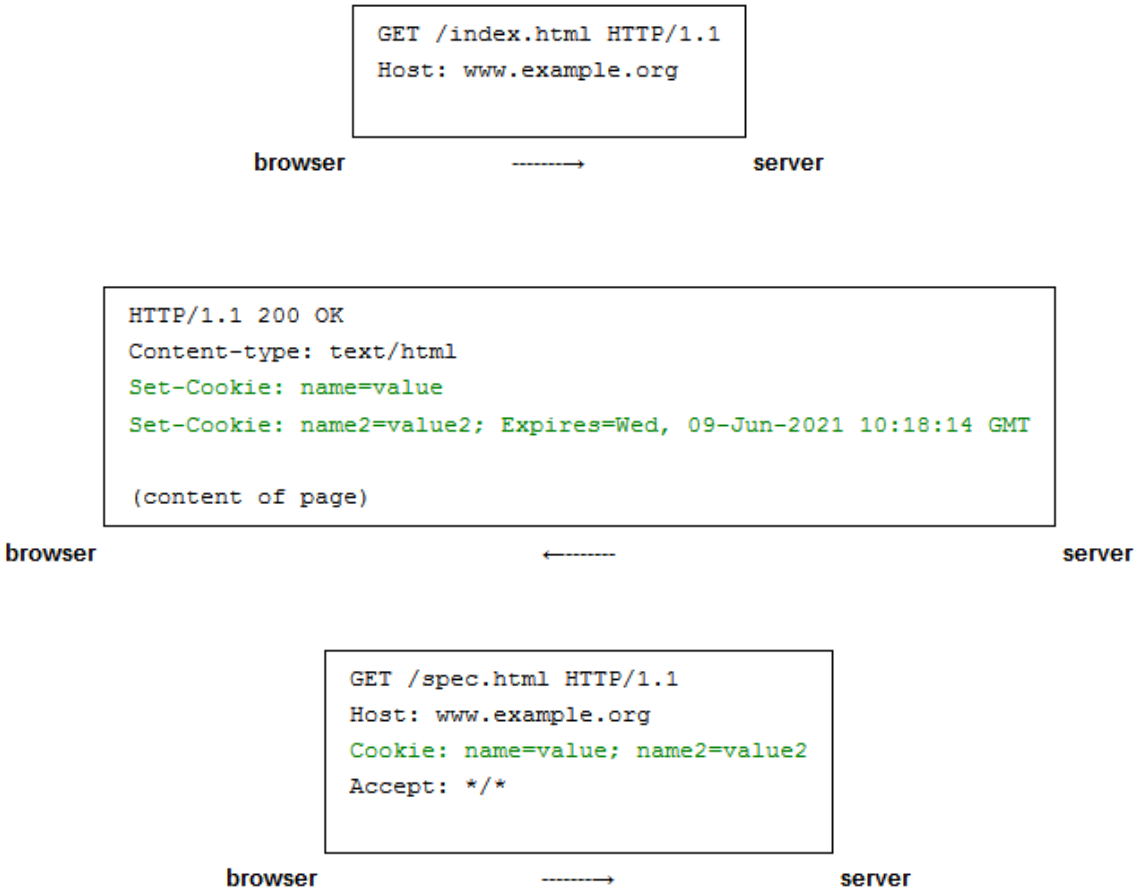
Sesje są problemem dla serwerów HTTP
- Brak mechanizmu sesji HTTP (bezstanowość)
- PHP/ASP.NET/JSP musi dostarczać API sesji
- Współdzielenie sesji przez wszystkie przeglądarki (jednego typu)
- RFC 2965 lub 6265



Śledzenie i utrzymanie sesji internetowej

Identyfikator sesji powinien być przechowywany po stronie serwera i stronie klienta
Identyfikatory sesji po stronie serwera przechowywane są w pamięci, pliku lub bazie danych
Istnieją 3 sposoby na śledzenie sesji u klienta

1. Cookies
2. Ukryte pola formularza
3. Przepisywanie URL'a



POLITECHNIKA
GDAŃSKA

HISTORIA MĄDROŚCIĄ
PRZYSZŁOŚĆ WYZWANIEM