Technologie Internetu

Protokół HTTP

Aleksander Denisiuk

denisjuk@pja.edu.pl

Polsko-Japońska Akademia Technik Komputerowych

Wydział Informatyki w Gdańsku

ul. Brzegi 55

80-045 Gdańsk

Protokół HTTP

Najnowsza wersja tego dokumentu dostępna jest pod adresem `http://users.pja.edu.pl/~denisjuk/`

HyperText Transfer Protocol

- Protokół warstwy aplikacji
- Architektura klient-serwer: żądanie — odpowiedź
- Jest podstawowym protokołem World Wide Web
- Jest wykorzystywany jako *transport* dla innych protokołów: XML-RPC, SOAP, WebDAV

Zasoby

- Podstawowe obiekty: *zasoby,* wskazane przez Uniform Resource Identifier w żądaniu klienta: pliki, obiekty abstrakcyjne
- Klient w żądaniu może wskazać sposób prezentacji zasoba: kodowanie, format, język, etc
- Możliwa jest wymiana danych binarnych (mimo że protokół jest tekstowy)

Bezstanowy

- Schemat żądanie-odpowiedź
- Identyfikacja zasobów poprzez absolutny URI
- W protokole nie jest zapamiętywana informacja o poprzednich tranzakcjach
- Możliwe jest zapamiętywanie przez oprogramowanie: klient, serwer

Zalety

- Prostota realizacji
- Rozszerzalność: nieznane nagłówki będą ignorowane
- Rozpowszechnioność

Wady

- Duży rozmiar przekazywanych komunikatów
  - Cachowanie danych u klienta
  - Kompresja danych
  - Proxy-serwery
  - Diff-kodowanie
- Brak nawigacji
  - Strony *Site map*
  - Pliki `Sitemap`
  - protokół `WebDAV`
- Brak rozproszoności
  - protokół `HTTP-NG`

Oprogramowanie — Serwery

nginx	$36{,}91\%$
Apache	$25{,}45\%$
Microsoft	$12{,}51\%$
Google	$3{,}42\%$

- Dane Netcraft za kwiecień 2020 (Active sites)

Oprogramowanie — Przeglądarki

Chrome	$80{,}7\%$
Firefox	$8{,}5\%$
Internet Explorer/Edge	$3{,}5\%$
Safari	$4{,}1\%$
Opera	$1{,}6\%$

- Dane W3CSchools za maj 2020

Oprogramowanie — Inne klienty

- Przeglądarki tekstowe
- Przeglądarki mobilne
- Wirtualne mapy
- Maszyny indeksujące
- Menadżery pobierań
- Pobieranie aktualizacji oprogramowania

Historia HTTP

HTTP 1991 Tim Berners-Lee

HTTP/0.9 1992, pierwsza specyfikacja

HTTP/1.0 1996, RFC 1945

HTTP/1.1 1999 — obecny standard

- trwałe (persistent) połączenia
- konieczne wysyłanie nazwy hosta w żądaniu

SPDY 2012, Google

HTTP/2.0 2015. Hypertext Transfer Protocol Bis Working Group (The Internet Engineering Task Force): protokół binary, semantyka zgadza się z HTTP/1.1

- dalej omawiany jest tylko HTTP/1.1

Struktura protokołu

- Każdy komunikat (zapytanie i odpowiedź) składa się z trzech części, przekazywanych we wskazanej kolejności:
  1. Linia startowa *(Starting line)* — określa typ komunikatu
  2. Nagłówki *(Headers)* — zawiera meta-dane
  3. Pusta linia
  4. Dane *(Body)*.
- Tylko linia startowa jest wymagana
- W żądaniu wymagany jest również nagłówek, zawierający nazwę hosta

Linia startowa żądania

```
METODA URI HTTP/1.1
```

METODA nazwa żądania

URI zasób

HTTP/1.1 wersja protokołu

- Przykład:

```
GET /i/PJWSTK_logo.gif HTTP/1.1
```

Linia startowa odpowiedzi

```
HTTP/1.1 Kod Komentarz
```

HTTP/1.1 wersja protokołu

Kod trzycyfrowy kod opracowania żądania

Komentarz tekstowy komentarz, opcjonalny

- Przykład:

```
HTTP/1.1 200 OK
```

Metody żądania

- każdy serwer powiniem obsługiwać metody `GET` i `HEAD`
- jeżeli serwer nie rozpoznał metody żądania, on zwraca kod odpowiedzi 501 (Not implemented)
- jeżeli serwer rozpoznał metodę, ale one nie może zostać zastosowana do zasobu, zwraca się kod odpowiedzi 405 (Method not allowed)
- w obu przypadkach serwer w odpowiedzi dołącza hagłówek `Allowed` z listą zaimplementowanych metod

Metody żądania. OPTIONS

- Opytanie serwera o zaimplementowanych metodach
- Przykład:

```
OPTIONS * HTTP/1.1
Host: www.pjwstk.edu.pl

HTTP/1.1 200 OK
Date: Sat, 26 Mar 2011 21:17:24 GMT
Server: Apache
Allow: GET,HEAD,POST,OPTIONS
Content-Length: 0
Content-Type: text/plain
```

Metody żądania. GET

- Pobieranie zasobu
- Przykład:

```
GET /i/PJWSTK_logo.gif HTTP/1.1
Host: www.pjwstk.edu.pl

HTTP/1.1 200 OK
Date: Sat, 26 Mar 2011 21:22:11 GMT
Server: Apache
Last-Modified: Tue, 10 Mar 2009 06:28:42 GMT
ETag: "2c01951a-114c-464bddcbfda80"
Accept-Ranges: bytes
Content-Length: 4428
Content-Type: image/gif

GIF89a...
```

Metody żądania. HEAD

- Pobieranie meta-danych o zasobie
- Przykład:

```
HEAD /i/PJWSTK_logo.gif HTTP/1.1
Host: www.pjwstk.edu.pl

HTTP/1.1 200 OK
Date: Sat, 26 Mar 2011 21:22:11 GMT
Server: Apache
Last-Modified: Tue, 10 Mar 2009 06:28:42 GMT
ETag: "2c01951a-114c-464bddcbfda80"
Accept-Ranges: bytes
Content-Length: 4428
Content-Type: image/gif
```

Inne metody żądania

`POST` wysyłanie danych na serwer do przetwarzania

`PUT` umieszczenie wysłanych danych na serwerze

`PATCH` uzupełnienie zasobu na serwerze

`DELETE` usuwanie zasobu na serwerze

`TRACE` zwraca otrzymane zapytanie (można zobaczyć, jak ono zostało zmodyfikowano na pośrednich serwerach)

Klasy kodów odpowiedzi

1xx Kody informacyjne

2xx Kody powodzenia

3xx Kody przekierowania

4xx Kody błędu aplikacji klienta

5xx Kody błędu serwera

Kody informacyjne

100 Continue

110 Connection Timed Out

111 Connection refused

Kody powodzenia

200 OK

201 Created

206 Partial Content

Kody przekierowania

300 Multiple Choices

301 Moved Permanently

307 Temporary Redirect

303 See Other

Kody błędu aplikacji klienta

400 Bad Request

401 Unauthorized

403 Forbidden

404 Not Found

413 Request Entity Too Large

Kody błędu serwera

500 Internal Server Error

503 Service Unavailable

Nagłówki HTTP

- `Nazwa: Wartość`
- Cztery rodzaje:
  1. General Headers, zarówno w żądaniu jak i w odpowiedzi
  2. Request Headers — w żądaniu
  3. Response Headers — tylko w odpowiedzi
  4. Entity Headers — określają zawartość

Ogólne nagłówki

`Date` — data generacji komunikata

`Transfer-Encoding` — sposób kodowania komunikata przy transmisji

Nagłówki żądania

`Accept` — lista typów akceptowalnych przez klienta zasobów

`Accept-Charset` — lista akceptowalnych kodowań znaków

`Accept-Ecnoding` — lista akceptowalnych sposobów kodowania zasobu przy transmisji

`Host` — nazwa hosta (jest wymagany), pozwala na organizację wirtualnych hostów

`If-Modified-Since` — jeżeli zasób został zmodyfikowany

`Range` — część zasobu

`User-Agent` — nazwa klienta

Nagłówki odpowiedzi

`Accept-Ranges` — akceptowalne jednostki części zasobów

`ETag` — unikatowy identyfikator (dla cachowania)

`Location` — URI dla przekierowania

`Allow` — lista obsługiwanych metod

`Server` — nazwa serwera

Nagłówki zawartości

`Content-Encoding` — sposób kodowania zasobu przy transmisji (`gzip`, `deflate`)

`Content-Language` — lista języków zasobu

`Content-Length` — rozmiar zasobu w bajtach

`Content-Range` — fragment zasobu

`Content-Type` — format i soposób reprezentacji zasobu

HTTP URI

```
http_URL = "http:" "//" host [ ":" port ]
     [ path ]  [ "?" query ] [ "#" fragment ]
```

- Domyślnie `port` 80,
- `path` zależy od ustawień serwera (`index.html`, `index.php`, `defaut.htm`, etc
- `http://gdansk.pjwstk.edu.pl/`
- `http://192.168.1.1/~denisjuk#edukacja`
- `http://192.168.1.1/?search=qw&erty=op`
- `http://[::192.9.5.5]/ipng`

Typy zasobów

- sposób przekazywania:
  - statyczne
  - dynamiczne (być może z użyciem parametryzacji)
- do klasyfikacji treści służy pojęcie *typu medium (medium type)*

Typ MIME

- Internet Media Type aka Multipurpose Internet Mail Extensions
- Opracowane istotnie w RFC 2046 dla SMTP
- Używane obecnie w HTTP, RTP, SIP, etc
- Specyfikacja typu medium:
  - `typ/podtyp[; parametryOpcjonalne]`
- Przykłady specyfikacji:
  - `text/plain; charset=UTF-8`
  - `application/javascript`
  - `image/png`

Typy MIME

`application`: dla specyficznych potrzeb

- `application/atom+xml`
- `application/javascript` (IE≥ 9)
- `application/pdf`
- `application/vnd.oasis.opendocument.text`
- `application/vnd.ms-excel`
- `application/x-latex`
- `application/x-rar-compressed`
- `application/x-shockwave-flash`
- `application/x-font-ttf` nie jest zarejestrowanym typem

Typy MIME, cd

audio :

- audio/vorbis
- audio/mpeg
- audio/x-ms-wma

image :

- image/png
- image/jpeg
- image/svg+xml

video :

- video/mpeg
- video/ogg

Typy MIME, cdd

`text` :

- `text/css`
- `text/csv`
- `text/html`
- `text/javascript` (przestarzały)
- `text/plain`
- `text/xml`

`multipart` :

- `multipart/mixed` (Email)
- `multipart/form-data`

IANA

- Internet Assigned Numbers Authority
- http://www.iana.org/assignments/media-types
- można zarejestrować własne typy (podtypy)

Serwer a typ mediów

- Serwer informuje klientów o typie medium zasobu
- Klient na podstawie tej informacji może stosownie opracować zasób
- Serwery rozpoznają typy udostępnianych przez:
  - *magiczną liczbę* (magic number)
  - rozszerzenie nazwy pliku z zasobem
  - jawne przypisanie typu do zasobu w konfiguracji serwera, bądź w treści hipertekstowej

Przykład żądania `multipart`

```
POST /send-message.html HTTP/1.1
Host: mail.example.com
Referer: http://mail.example.com/send-message.html
User-Agent: BrowserForDummies/4.67b
Content-Type: multipart/form-data; boundary="Asrf456BGe4h"
Content-Length: cały rozmiar
Connection: keep-alive
Keep-Alive: 300

--Asrf456BGe4h
Content-Disposition: form-data; name="DestAddress"

bv@example.com
--Asrf456BGe4h
Content-Disposition: form-data; name="MessageTitle"

Greetings!
```

Przykład żądania `multipart`, cd

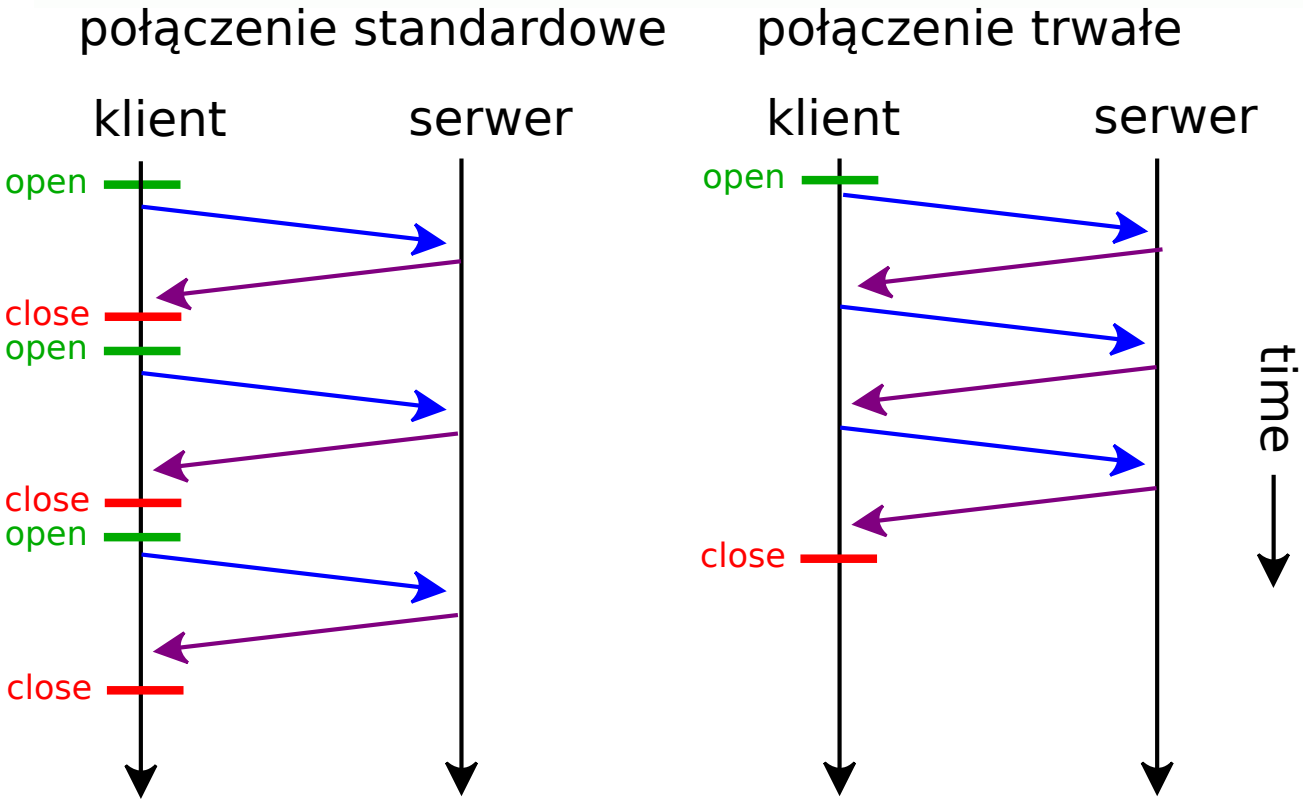
```
--Asrf456BGe4h
Content-Disposition: form-data; name="MessageText"

I am Shehu Musa Abacha, cousin to the former
Nigerian dictator Sani Abacha.
Please, find attached my photos
--Asrf456BGe4h
Content-Disposition: form-data; name="AttachedFile1"; filename="musa.jpg"
Content-Type: image/jpeg

dane binarne
--Asrf456BGe4h
Content-Disposition: form-data; name="AttachedFile2"; filename="sani.jpg"
Content-Type: image/jpeg

dane binarne
--Asrf456BGe4h--
```

Połączenia trwałe



Połączena trwałe

- Nagłówek `Connection: persistent`

  HTTP/1.0 niestandardowa opcja implementowana przez wiele klientów/serwerów

  HTTP/1.1 połączenia są domyślnie trwałe (`Connection: close`)

- Proxy używają trwałych połączeń tylko z serwerem

- Połączenia w każdej chwili mogą zostać przerwane zarówno przez serwer jak i przez klienta

Połączena trwałe. Zasady

- jeśli klient nie chce wysyłać kolejnych żądań w ramach połączenia, powinien wysłać nagłówek `Connection: close`
- serwer nie powinien zamykać połączenia w trakcie wysyłania odpowiedzi
- przed zamknięciem połączenia serwer powinien odpowiedzieć na co najmniej jedno żądanie
- aplikacja HTTP/1.1 powinna radzić sobie w sytuacjach niespodziewanego (asynchronicznego) zamknięcia połączenia
  - ponowić żądanie automatycznie
  - spytać użytkownika („efekty uboczne”)

Połączena trwałe. Potoki

- wysyłanie w ramach połączenia trwałego kilku żądań po kolei, bez oczekiwania na odpowiedź
- Odpowiedzi powinny być generowane w takiej samej kolejności
- HTTP nie udostępnia mechanizmu numerowania komunikatów
- W ramach potoku klient nie powinien wysyłać żądań, które mogą powodować efekty uboczne

DEMO

```
HEAD /images/logo_pjwstk.gif HTTP/1.1
Host: gdansk.pjwstk.edu.pl

GET /images/logo_pjwstk.gif HTTP/1.1
Host: gdansk.pjwstk.edu.pl

GET /favicon.ico HTTP/1.1
Host: www.pjwstk.edu.pl

GET /templates/pjwstk/favicon.ico HTTP/1.1
Host: www.pja.edu.pl

OPTIONS * HTTP/1.1
Host: www.pjwstk.edu.pl
```