HTTP jest obecnie najpopularniejszym i najbardziej rozpowszechnionym protokołem służącym do komunikacji klientów z serwerami. Jego początki nie zakładały tak wszechstronnego użycia jak ma to miejsce obecnie. Niemal wszystkie aplikacje i urządzenia korzystające z sieci internet korzystają z tego rozwiązania.

Pierwszą wersję protokołu HTTP stworzył Tim Berners-Lee, i zakładała ona jak najprostszą komunikację. Został zaprojektowany do przesyłania dokumentów hipertekstowych i oparty na protokołach TCP/IP i oznaczony numerem wersji 0.9. Funkcjonalności jakie zaimplementowano w protokole przedstawiają się następująco:

* Żądanie klienta to prostu ciąg znaków ASCII
* Żądanie klienta zakończone było znakiem nowego wiersza (CRLF)
* Odpowiedź z serwera była prostym strumieniem tekstowym
* Odpowiedź była sformatowana w języku HyperText Markup Language (HTML)
* Po zakończeniu przesyłania dokumentu połączenie było zamykane

W popularnych serwerach takich jak Apache czy Nginx wciąż dostępna jest obsługa protokołu HTML 0.9. Jego działanie jest traktowane raczej jako ciekawostka i wykorzystanie do niekonwencjonalnych zadań.

|  |
| --- |
| $> telnet website.org 80  Connected to xxx.xxx.xxx.xxx  GET /rfc/rfc1945.txt HTTP/1.0  (plain-text response)  (connection closed) |

Powyższe żądanie składa się z metody GET oraz ścieżki do żądanego dokumentu, natomiast odpowiedź jest zwykłym dokumentem HTML. Najprostsze połączenie jakie można stworzyć.

## HTTP 1.0

Największy rozwój specyfikacji protokołu HTML nastąpił w latach 1991 – 1995. W tym czasie powstawało takie oprogramowanie jak przeglądarki internetowe, narzędzia do komunikacji. To był także czas rozwoju infrastruktury całego internetu. Pożądanie nowych funkcjonalności rozwijającego się internetu wymagało czegoś więcej niż HTTP 0.9, który nie zadowalał programistów tworzących aplikacje inne niż klient-serwer = żądanie-odpowiedź.

W Maju 1996 roku zbiór „dobrych praktyk i wzorców” został opublikowany przez „HTTP Working Group” (HTTP-WG) pod nazwą RFC 1945 i była to pierwsza specyfikacja protokołu HTTP 1.0. Specyfikacja ta była jednak jedynie informacyjna, a sam protokół do chwili obecnej nie doczekał się oficjalnej, formalnej dokumentacji i nie jest także oficjalnym standardem, który obowiązuje w internecie.

|  |
| --- |
| $> telnet website.org 80  Connected to xxx.xxx.xxx.xxx  GET /rfc/rfc1945.txt HTTP/1.0 ❶  User-Agent: CERN-LineMode/2.15 libwww/2.17b3  Accept: \*/\*  HTTP/1.0 200 OK❷  Content-Type: text/plain  Content-Length: 137582  Expires: Thu, 01 Dec 1997 16:00:00 GMT  Last-Modified: Wed, 1 May 1996 12:45:26 GMT  Server: Apache 0.84  (plain-text response)  (connection closed) |

❶ - Numer wersji protokołu HTTP oraz następujący po nim nagłówek

❷ - Status odpowiedzi oraz następujący po nim nagłówek

* Przykładowe klasy odpowiedzi Status-Code(PRZYPIS):  
  "200"   ; OK   
  "201"   ; Created    
  "202"   ; Accepted    
  "204"   ; No Content  
  "302"   ; Found  
  "403"   ; Forbidden  
  "404"   ; Not Found

W przeciwieństwie do poprzedniej wersji nastąpiło kilka zmian:

* Żądanie może się składać z kilku pól nagłówka oddzielonych znakami nowego wiersza
* Obiekt z odpowiedzią jest poprzedzony wierszem ze statusem odpowiedzi
* Obiekt zawierający odpowiedź zawiera własny zestaw pól nagłówka oddzielonych znakami nowego wiersza
* Po każdym żądaniu połączenie między klientem a serwerem jest zamykane

Nagłówki żądań oraz odpowiedzi są zapisane w kodzie ASCII, lecz obiekt odpowiedzi może być każdego innego typu np. plikiem tekstowym, plikiem HTML, obrazem itp. Specyfikacja RFC opisuje nie tylko negocjację typu mediów ale także kodowanie treści, autoryzację, autoryzację serwera proxy, obsługę zestawów znaków, formaty daty, typ komunikatów wieloczęściowych, zapis do pamięci podręcznej itd.

## HTTP 1.1

Praca nad dokumentowaniem protokołu w wersji 1.0 trwała około pięciu lat (do 1999 roku). Równolegle trwały prace nad przekształceniem technologii w nowszą wersję oznaczoną jako HTTP 1.1 i to ona jako pierwsza została oficjalnie opublikowana w styczniu 1997 roku w specyfikacji RFC 2068. Dwa lata po tej publikacji zostało wprowadzonych kilka usprawnień w wydajności lecz kod protokołu pozostał taki sam, jednak nowsza wersja specyfikacji to RFC 2616.

Podobnie jak to miało miejsce podczas przechodzenia z wersji 0.9 na nowszą, również teraz znaleziono wiele problemów i wprowadzono usprawnienia w stosunku do wersji 1.0. Przykładowe optymalizacje to między innymi kolejkowanie żądań, transmisja danych w blokach, żądania przesyłania zakresu bajtów, kodowanie przesyłanych danych czy podtrzymywanie połączenia.

Poniżej znajduje się przykładowa sesja HTTP 1.1:

|  |
| --- |
| $> telnet website.org 80  Connected to xxx.xxx.xxx.xxx  GET /index.html HTTP/1.1 ❶  Host: website.org  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_7\_4)... (snip)  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  Accept-Encoding: gzip,deflate,sdch  Accept-Language: en-US,en;q=0.8  Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.3  Cookie: \_\_qca=P0-800083390... (snip)  HTTP/1.1 200 OK ❷  Server: nginx/1.0.11  Connection: keep-alive  Content-Type: text/html; charset=utf-8  Via: HTTP/1.1 GWA  Date: Wed, 25 Jul 2012 20:23:35 GMT  Expires: Wed, 25 Jul 2012 20:23:35 GMT  Cache-Control: max-age=0, no-cache  Transfer-Encoding: chunked  100 ❸  <!doctype html>  (snip)  100  (snip)  0 ❹  GET /favicon.ico HTTP/1.1 ❺  Host: www.website.org  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_7\_4)... (snip)  Accept: \*/\*  Referer: http://website.org/  Connection: close ❻  Accept-Encoding: gzip,deflate,sdch  Accept-Language: en-US,en;q=0.8  Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.3  Cookie: \_\_qca=P0-800083390... (snip)  HTTP/1.1 200 OK ❼  Server: nginx/1.0.11  Content-Type: image/x-icon  Content-Length: 3638  Connection: close  Last-Modified: Thu, 19 Jul 2012 17:51:44 GMT  Cache-Control: max-age=315360000  Accept-Ranges: bytes  Via: HTTP/1.1 GWA  Date: Sat, 21 Jul 2012 21:35:22 GMT  Expires: Thu, 31 Dec 2037 23:55:55 GMT  Etag: W/PSA-GAu26oXbDi  (icon data)  (connection closed) |

❶ - Żądanie pliku HTML wraz z metadanymi opisującymi kodowanie, zestaw znaków i ciasteczka.

❷ - Blok z odpowiedzią na pierwsze żądanie HTML.

❸ - Liczba bajtów w bloku danych wyrażona jako liczba szesnastkowa w kodzie ASCII (256 bajtów)

❹ - Koniec strumienia bloków danych z odpowiedzią.

❺ - Żądanie pliku ikony przesłany tym samym połączeniem TCP

❻ - Informacja do serwera, że połączenia nie będzie ponownie wykorzystane.

❼ - Odpowiedź z plikiem ikony i zamknięcie połączenie

Od razu rzucają się w oczy dwa obiekty żądań TCP, co wcześniej było niemożliwe poprzez zamykane połączenia po zakończeniu żądania. Podtrzymywanie i ponowne wykorzystanie tego samego połączenia znacznie umożliwia znacznie szybsze dostarczenie mu odpowiedzi. W celu zamknięcia połączenia wykorzystany jest nagłówek (Connection: Close).

W ulepszonej wersji protokołu HTTP 1.0 też została wprowadzona funkcja podtrzymywania połączenia poprzez użycie nagłówka (Connection: Keep-Alive). Nagłówek ten jednak nie jest wymagany jeśli serwer aplikacji webowej korzysta z nowszej wersji protokołu.

## HTTP 2.0

Coraz powszechniejsze i bardziej zaawansowane aplikacje webowe, już nawet nie statyczne czy interaktywne strony WWW, wymagają także większej wydajności. W odpowiedzi na to z pozoru banalne zapotrzebowanie powstaje protokół HTTP w wersji 2.0. Zakłada on zmniejszenie opóźnień, zwiększenie wydajności transmisji danych oraz uzyskanie większej przepływności. W tym momencie nie planuje się zmieniać takich cech protokołu jak nagłówki HTTP oraz ich wartości i funkcje.

Wszystkie aplikacje webowe, które w tej chwili już zostały uruchomione na serwerach w momencie pełnego wdrożenia nowego protokołu sieciowego będą z niego korzystały bez konieczności wprowadzania zmian w kodzie aplikacji. Ważne jest aby serwer oraz klient obsługiwały to połączenie. Zmiany w działaniu aplikacji powinny być jednak zauważalne jeśli prócz serwera, aplikacja jest dobrze skonfigurowana i wolna od błędów jakie może wskazać np. „Snail Project” przygotowany specjalnie w celu wykonania testu wydajnościowego aplikacji. Warto także pamiętać, że tworząc najbardziej nowoczesną aplikację warto zadbać o kompatybilność wsteczną protokołu. Serwer Apache (jeśli programista nie skonfiguruje połączenia inaczej) zawsze wykorzystuje najwyższą możliwą wersję protokołu HTTP.

http://www.drzewo-wiedzy.pl/?page=artykul&id=70