Web сървър

проф. д-р инж. Венета Алексиева

Основни моменти

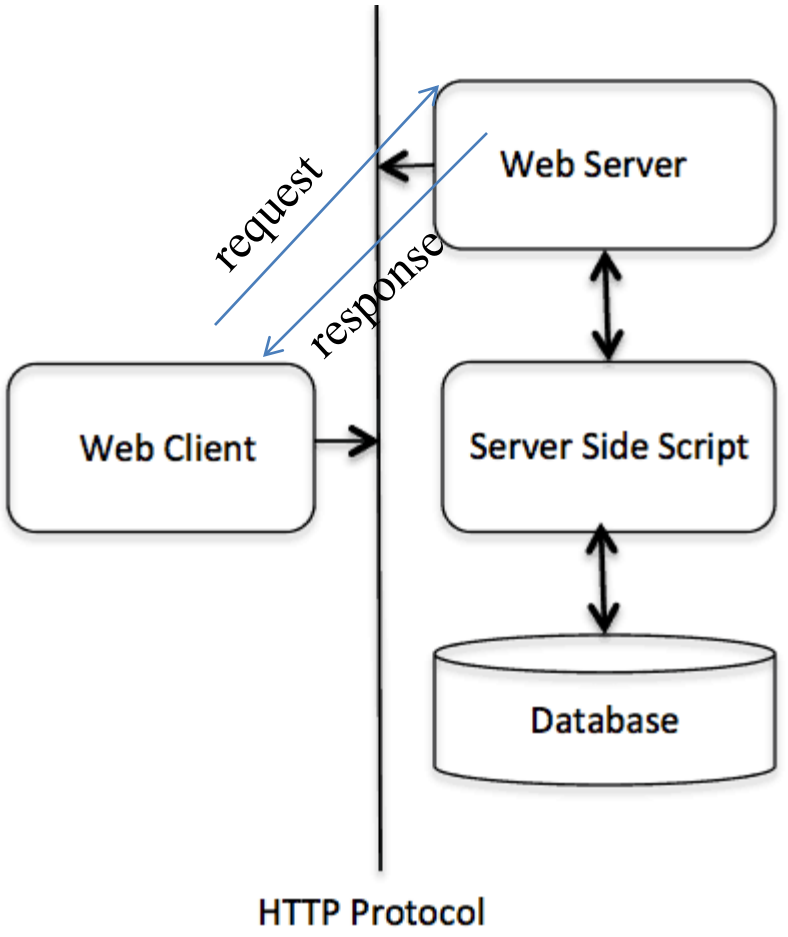
- HTTP протокол
- Web сървър
- Конфигуриране на Apache

HTTP протокол

- Протокол, позволяващ обмен на хипермедийна информация в Интернет.
- Използва се за реализирането на World Wide Web (WWW).
- Поддържа CGI (*Common Gateway Interface*), SSL (*Secure Socket Layer*) и виртуални домейни.
- Сървърният компонент е известен като “web сървър”, а клиентският – „browser”.
- За повишаване на сигурността се използва криптиране чрез протокол HTTPS.
- Използва се порт 80 (за HTTPS – 443).

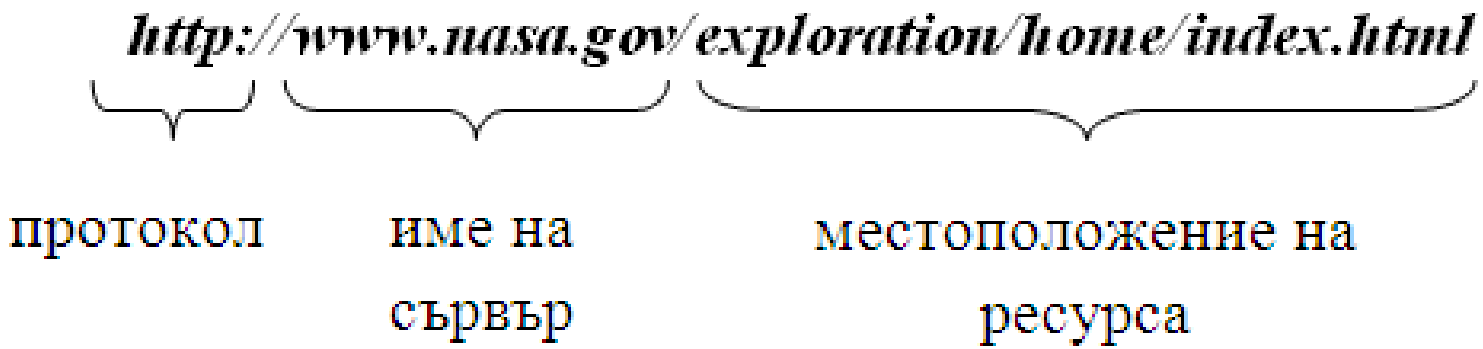
HTTP протокол

- Протокол от типа „заявка-отговор”.
- Протоколът се реализира чрез съобщения в ASCII текст.
- При изпращане на заявка се използва задължително FQDN.



HTTP

- Всеки хипермедиен обект се идентифицира посредством т.н. *Uniform Resource Locator (URL)*.
- Той се състои от три основни компонента: протокол, име на сървър и местоположение на ресурс:



HTTP преди

- HTTP/1.0 всяка TCP конекция включва само един обмен.
- При HTTP/1.1 са възможни множество обмени.
- В зората на WWW хипертекстовите документи са били основно ТЕКСТОВИ документи.
- Една сесия е била достатъчна, тъй като целият ресурс е бил в един файл.
- През 1990 започва използването на хипермедийни документи, с включени графики и други файлове.

HTTP характеристики

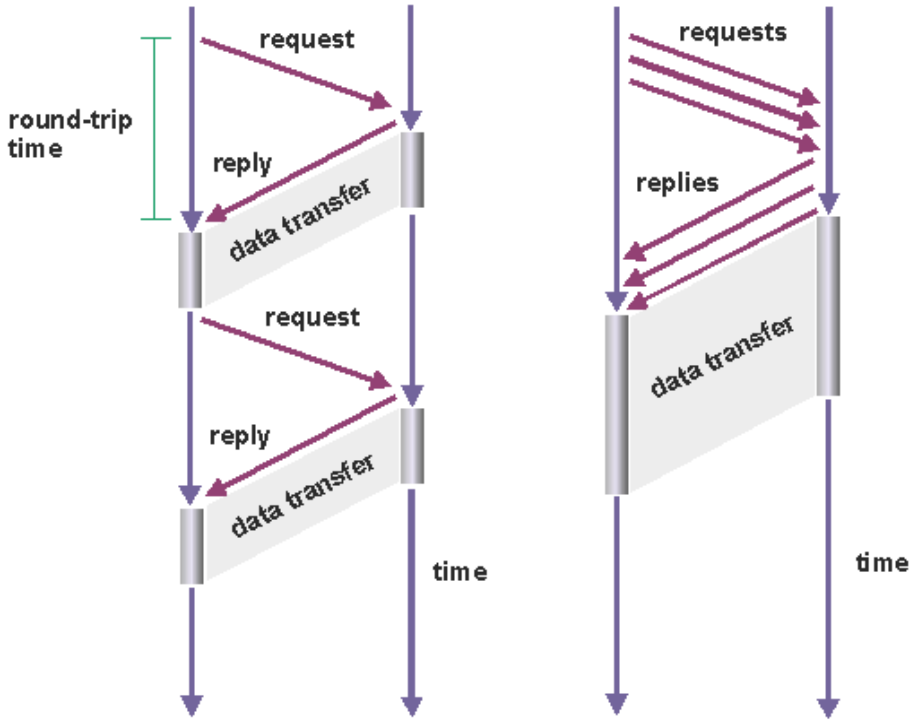
- Connectionless – доставя се ресурса и се прекъсва сесията
- Независим от преносната среда
- Stateless- не поддържа информация за състоянието, не помни взаимодействието с клиента
- Проблем, когато сесията трябва да бъде удължена – решава се със сесии и cookies
- Няма механизъм за управление на диалога
- Управлението на потока от данни – от TCP
- HTTP1.0 – временни конекции и се затварят
- HTTP1.1- постоянни конекции и не се затварят
- HTTP2.0 (от 2015г.) –мултиплексиране на заявки и отговори
- HTTP3.0 (от 2020г.) - използва различен транспортен протокол, способен да възстанови пакет с минимална загуба на производителност

Transitory (временни) конекции

- Те са били единственият тип, поддържан от оригиналния HTTP/0.9.
- Неефикасни са когато клиентът трябва да направи множество заявки към един и същ сървър. Това е характерно за съвременните хипертекстови документи, съдържащи референции към изображения или медийни файлове.
- С временните конекции, всяка от тези отделни заявки от страна на клиента изисква да се установи отделна TCP конекция между клиента и сървъра.
- Всяка конекция изисква ресурси от страна на сървъра и заемане на мрежата.

Persistent (постоянни) конекции

- Те повишават ефективността на HTTP чрез премахване на повечето от разходите за комункация за HTTP/1.0.
- Те стават стандарт за комуникациите между клиентите и сървърите в Web.
- Създаване на постоянна конекция- клиентът е инициаторът на връзката с Web сървъра.



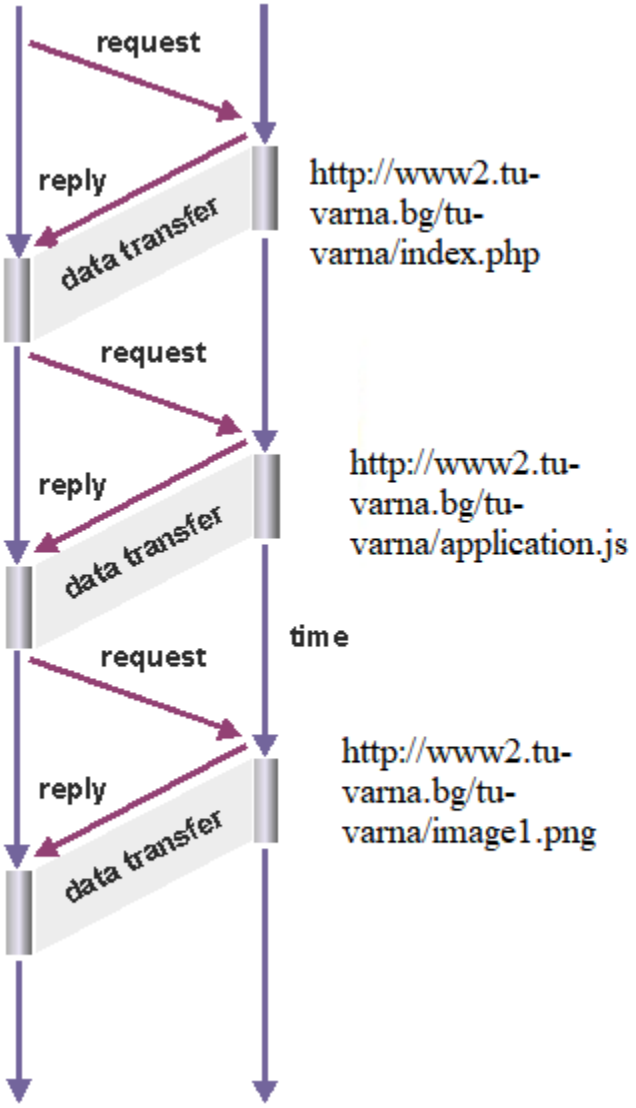
HTTP2.0

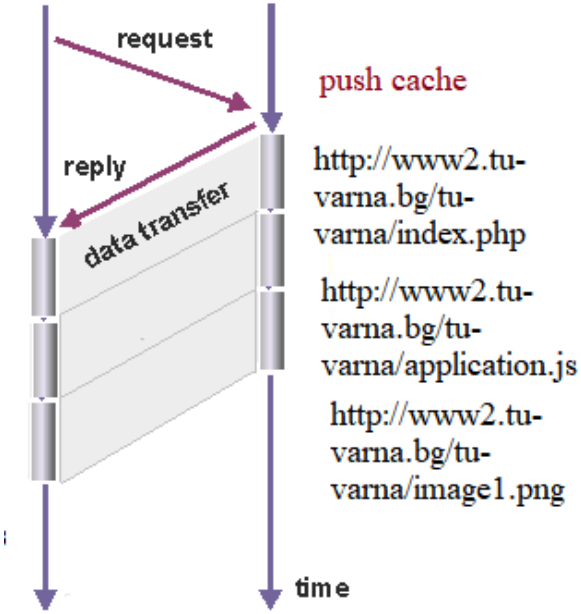
- Базиран на SPDY протокол на Google
- Стандартизиран през май 2015г.
- Използва TLS (SSL)
- Компресира хедърите
- Приоритизира заявките
- Изпраща корелативни (свързани) HTTP ресурси, за да минимизира заявките към тях в една TCP конекция

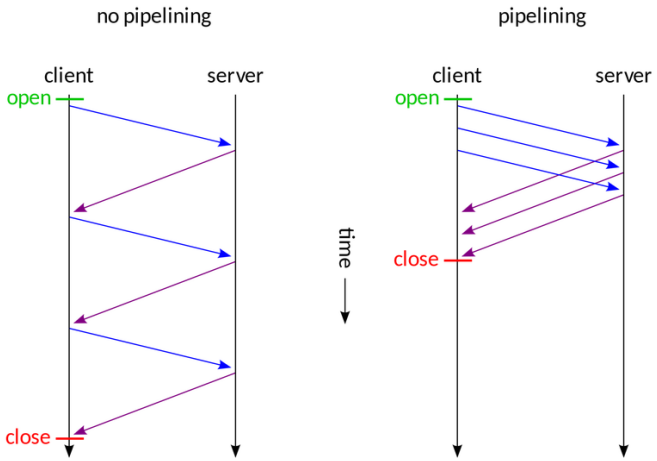
HTTP3.0

- Пренася се от транспортен протокол Quick UDP Internet Connections (QUIC), разработен от Google, различен от TCP, с предимства като:
  - вградено мултиплексиране,
  - per-stream flow control,
  - установява конекция с low-latency
  - включва TLS 1.3, а не го ползва като отделен слой.
- Запазва предимствата на HTTP2.0:
  - Компресира хедърите
  - Приоритизира заявките
  - Изпраща корелативни (свързани) HTTP ресурси, за да минимизира заявките към тях

HTTP1.1 и HTTP2.0/3.0

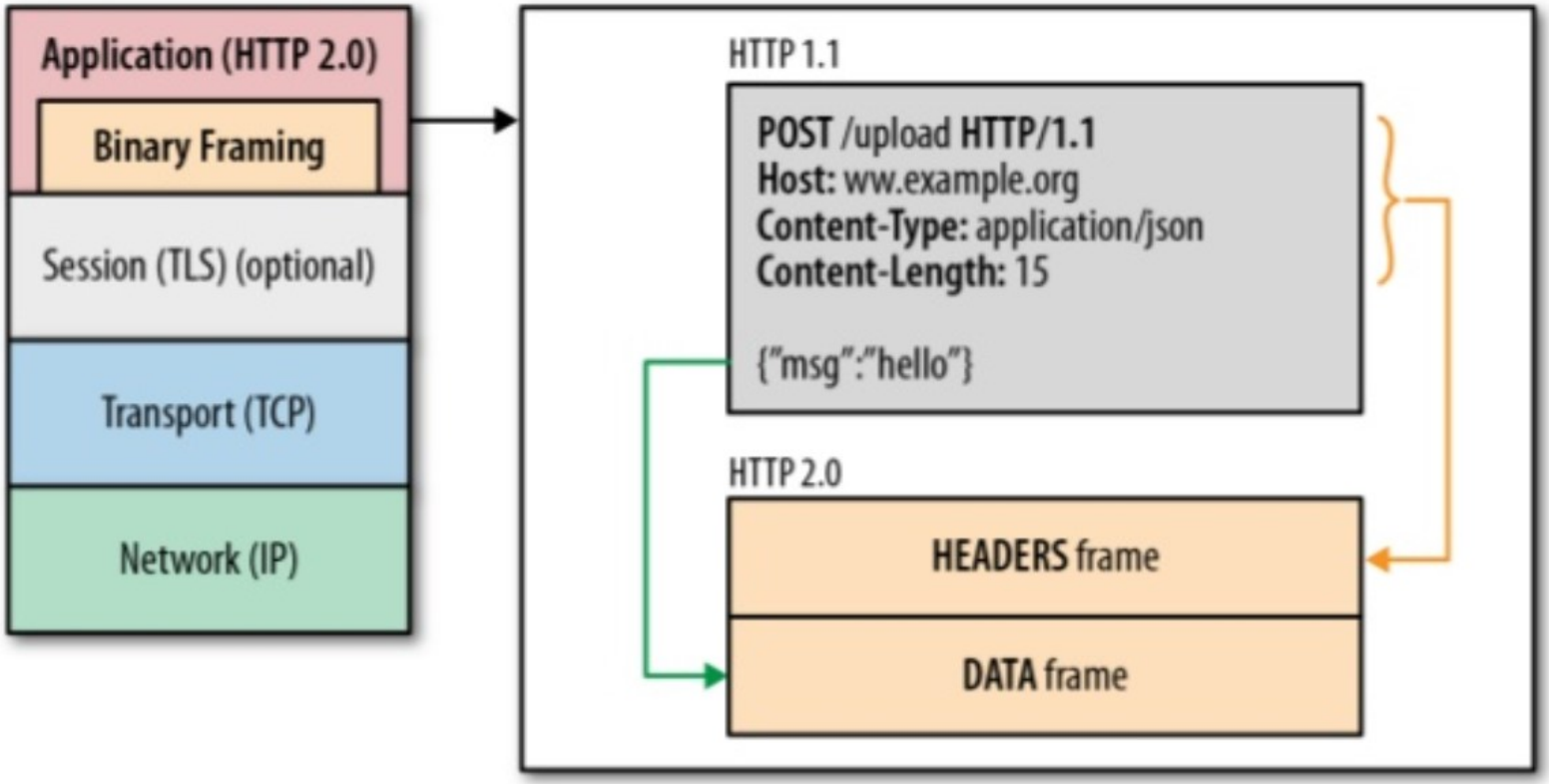




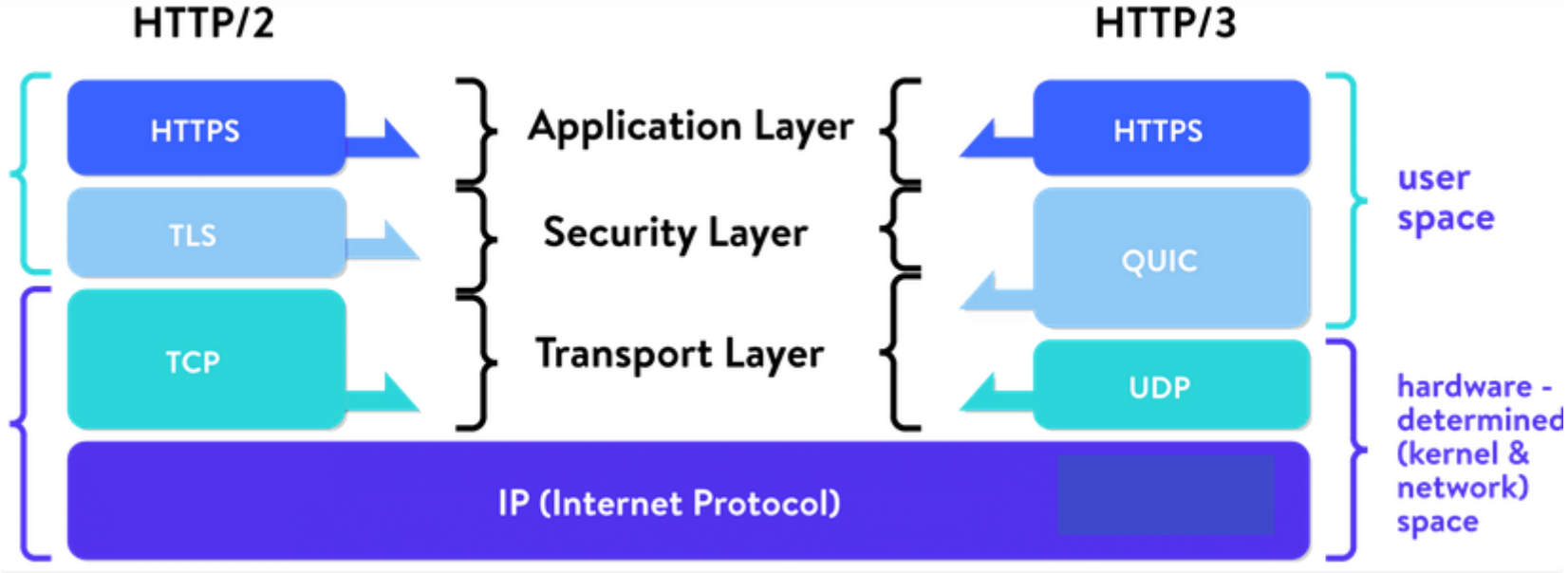


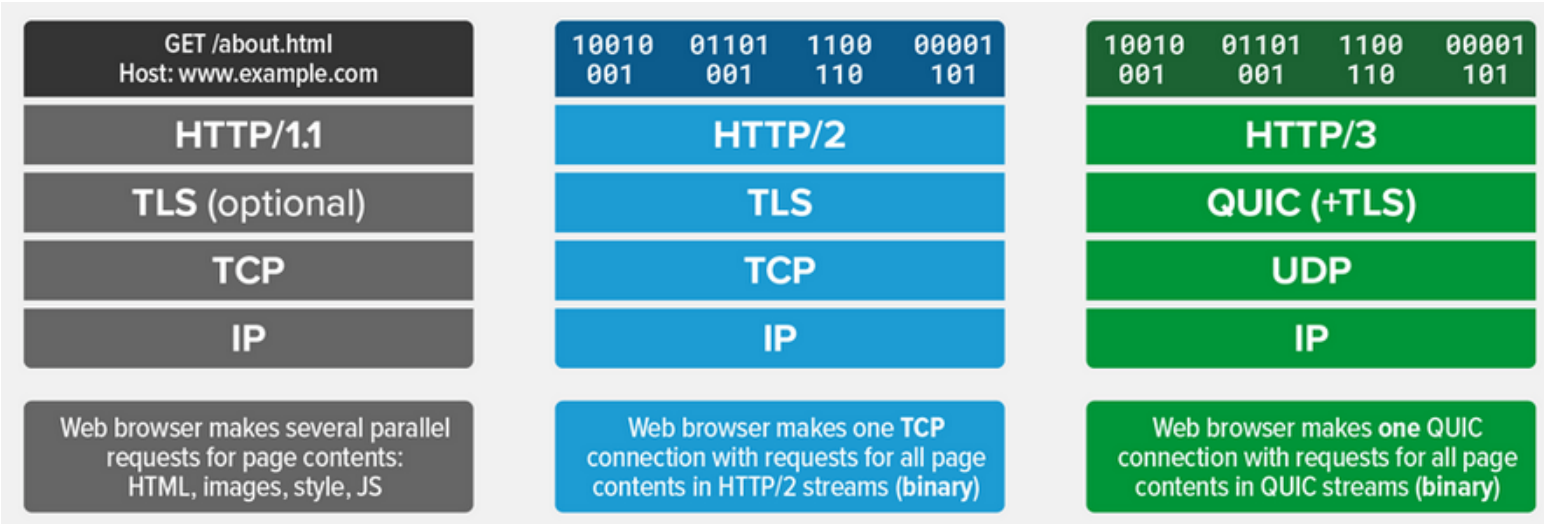
HTTP1.1 и HTTP2.0/3.0

- Приоритизация
- Flow control
- Server push

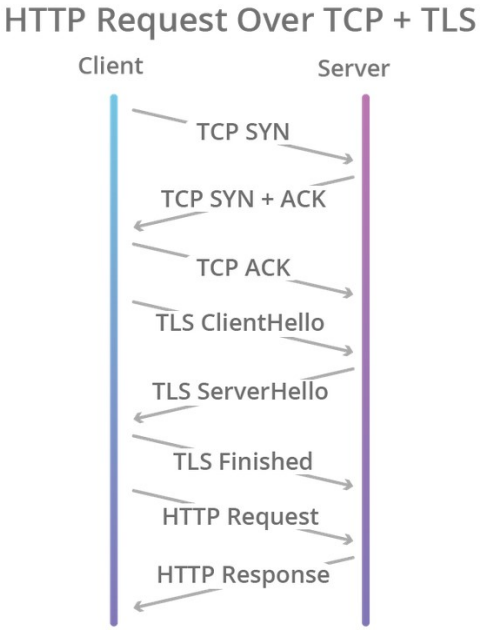


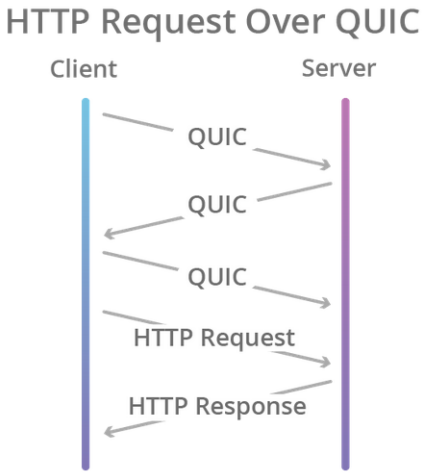
HTTP2.0 и HTTP3.0 протоколен стек





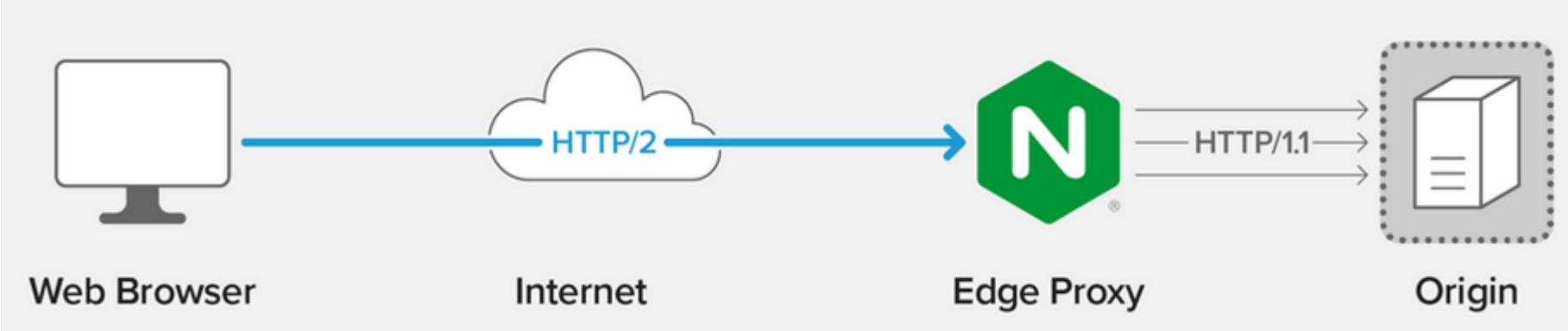
Договаряне на версията на HTTP





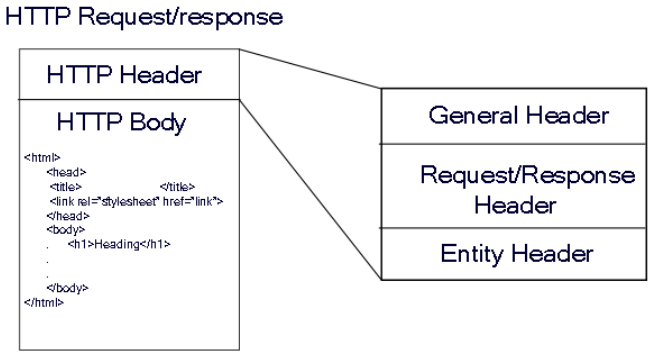
- При HTTP2.0 се използва TLS handshake, за да се открие дали клиентът и сървърът са способни да комуникират през HTTP2.0 преди връзката да бъде установена.
- Клиентът установява TCP връзка за първоначалната HTTP заявка.
- Отговорът от сървър, който поддържа HTTP3.0, включва в хедъра Alt-Svc, за да посочи UDP порта, на който слуша за HTTP3.0 трафик.
- Браузърът помни кои сайтове поддържат QUIC, за да елиминират времето за размяна на служебни съобщения за договаряне на връзката.

Хибридна употреба

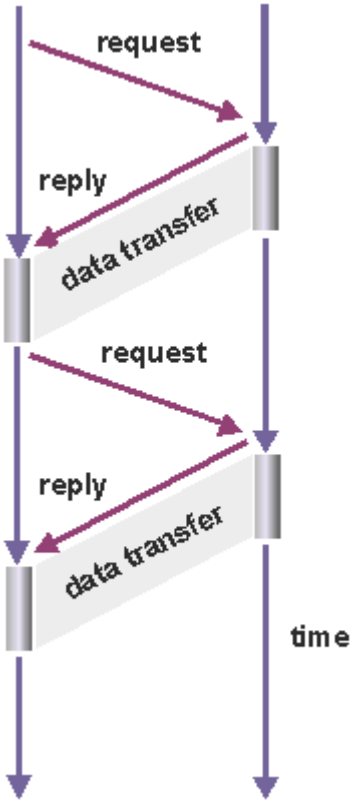


HTTP1.1 съобщения- общ формат

- HTTP Generic Message Format- текстов. Текстовите редове завършват с CR/LF контролни символи.




```
<start-line>
<message-headers>
<empty-line>
[<message-body>]
[<message-trailer>]
```



- Хедърите винаги се изпращат като регулярен текст.
- Тялото може да бъде както текст, така и 8-битова двоична информация, в зависимост от естеството на данните.

HTTP1.1 съобщения- общ вид

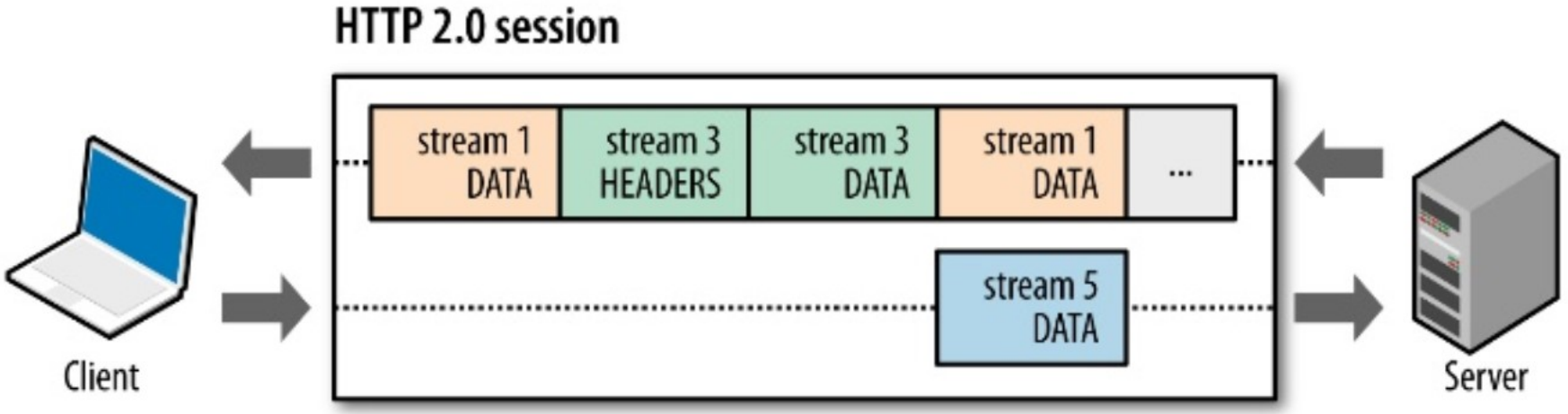
Съобщение-заявка:

```
<request-line>
<general-headers>
<request-headers>
<entity-headers>
<empty-line>
[<message-body>]
[<message-
 trailers>]
```

Съобщение-отговор:

```
<status-line>
<general-headers>
<response-
 headers>
<entity-headers>
<empty-line>
[<message-body>]
[<message-
 trailers>]
```

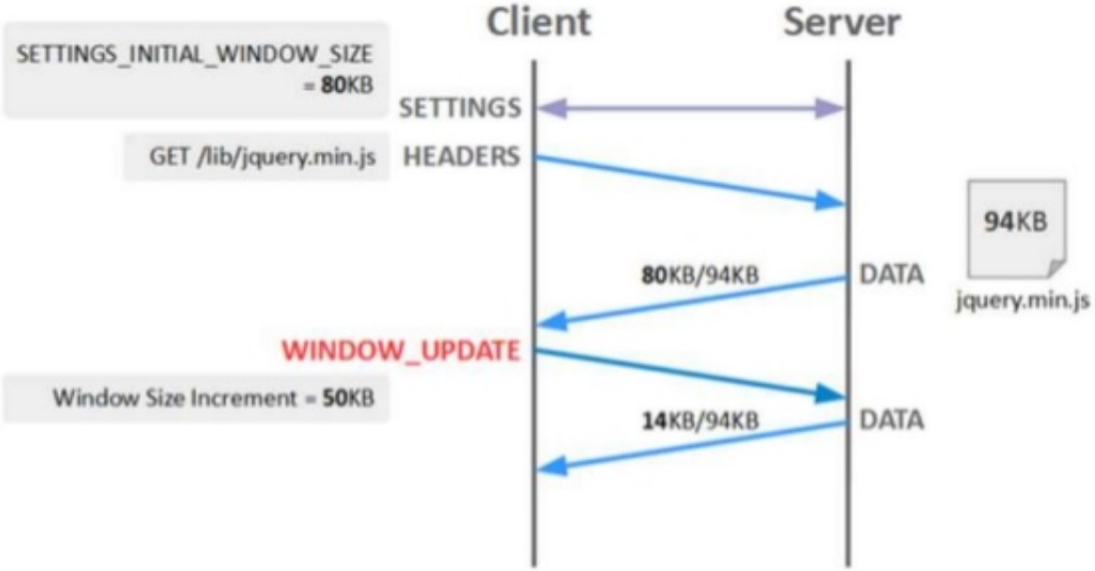
HTTP2.0/3.0- поток от съобщения



- В една TCP конекция
- Потоците са мултиплексирани, като комуникацията е разделена във фреймове – headers, data….
- Фреймовете са приоритизирани от сървъра
- Има flow control
- Всеки поток и конекция започва с 64KB window size, само DATA фрейма може да намали window size с window_update фрайм

Представени са 3 активни потока (1,3,5)

HTTP2.0-flow control



- Само DATA кадри са обект на контрол на потока
- С фрейма WINDOW_UPDATE и клиентът, и сървърът могат самостоятелно да рекламират броя октети, които са готови да получат и тогава партньорът трябва да спазва тази наложена стойност.
- Това е цяло число от 1 до 2^{31} -1. по подразбиране е 65535.

HTTP1.0 headers-request/response

Headers Cookies Request Response Timings

Filter Headers Block Resend

GET http://www2.tu-varna.bg/tu-varna/

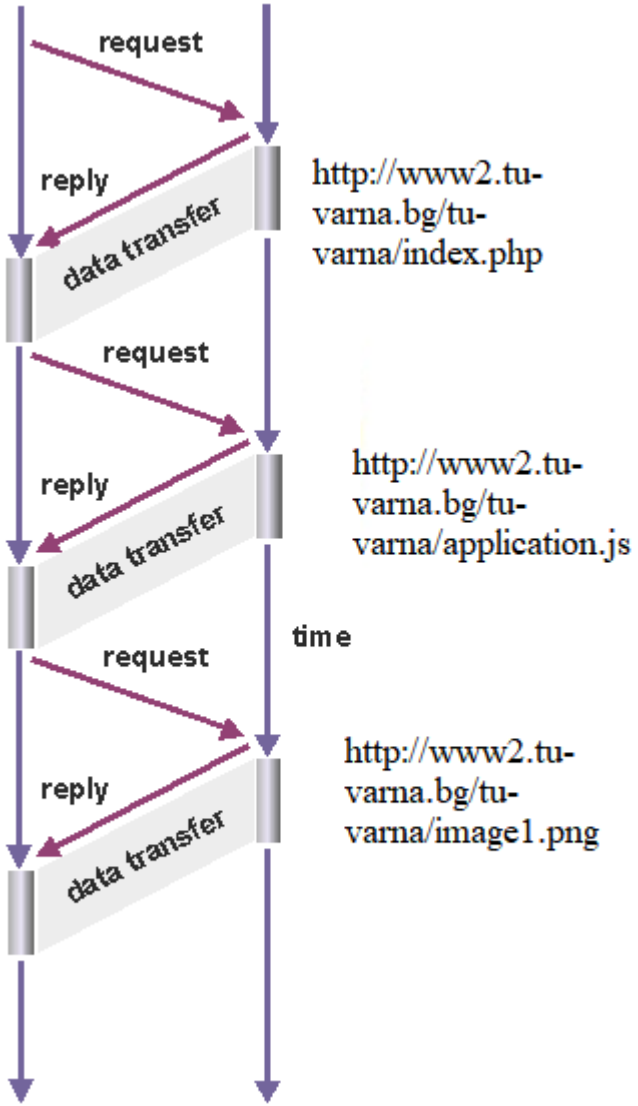
Status 200 OK
Version HTTP/1.1
Transferred 12.89 KB (67.06 KB size)

Response Headers (611 B) Raw

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 12586
Content-Type: text/html; charset=utf-8
Date: Fri, 10 Jul 2020 17:33:40 GMT
Expires: Wed, 17 Aug 2005 00:00:00 GMT
Keep-Alive: timeout=5, max=500
Last-Modified: Fri, 10 Jul 2020 17:33:41 GMT
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Pragma: no-cache
Server: Apache/2.2.15 (CentOS)
Set-Cookie: 0c514fc5e08e5f3179897355fc646331=mdqq17rh0r4cg7jtf1c6olluf5; path=/; HttpOnly
Vary: Accept-Encoding
X-Powered-By: PHP/5.6.40

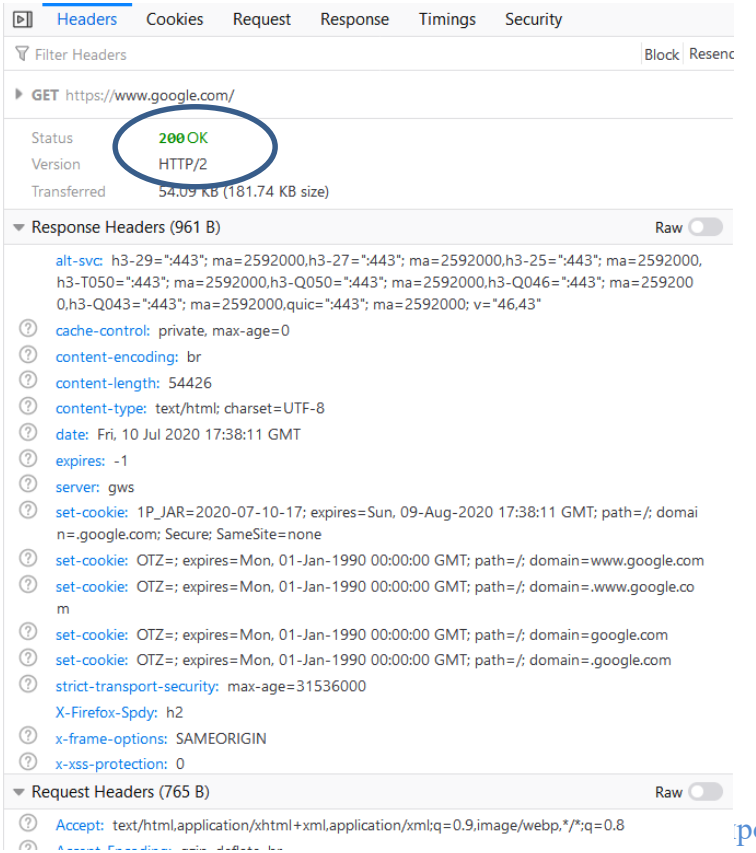
Request Headers (386 B) Raw

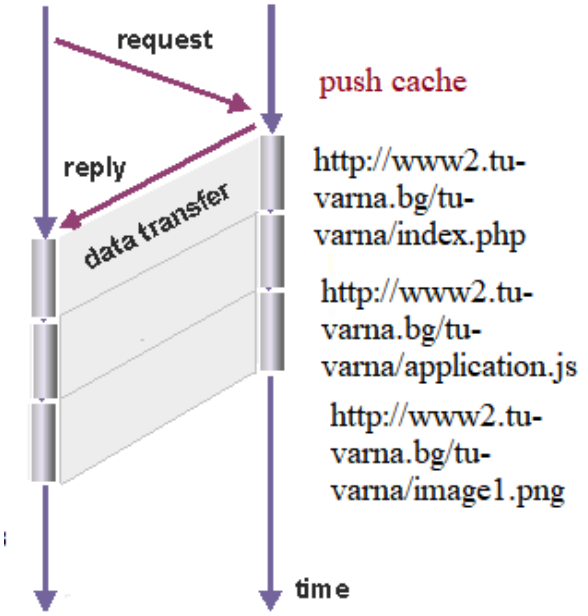
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Cookie: _ga=GA1.2.903615407.1507820477
Host: www2.tu-varna.bg
Upgrade-Insecure-Requests: 1



HTTP2.0 headers-request/response

- Заявките са:
  - Следствие от конкретно действие на потребителя (кликване върху хиперлинк)
  - В резултат от предишно действие (референция към inline изображение в документа).
- Една заявка води до един отговор –указва резултата от изпълненото от сървъра действие и често съдържа обект (файл или ресурс) в message-body.





HTTP body

Headers | Cookies | Params | Response | Timings | Stack Trace | Security

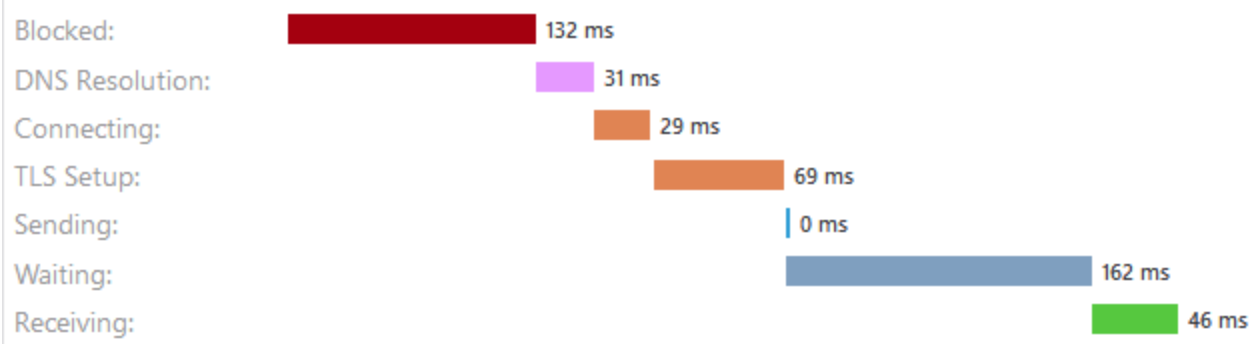
Response payload

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <script>
5          //
6  /* Version f1b9622831c5f758b69f8c4fafbe9659 v:4.5.0.773, c:5eb1d893d60371496c
7  (function(e){(function(e){(function(e){var g=function(){function b(){}b.addEv
8  a.toString(16);return"0000".substr(a.length)+a};return a(c[0])+a(c[1])+"_"+a(
9  ""))return JSON.stringify(a);var d=Object.toJSON;return b.e(d)&&'{"a":[1,2,3,
10 "*")};return b}();e.j=g;var p=new g;g.addEventListener(window,"message",funct
11
12     </script>
13 </head>
14 <body>
15 Ready
16 </body>
17 </html>
```

Headers | Cookies | Request | Response | Timings | Security

Queued: 0 ms | Started: 132 ms | Downloaded: 473 ms

Request Timing



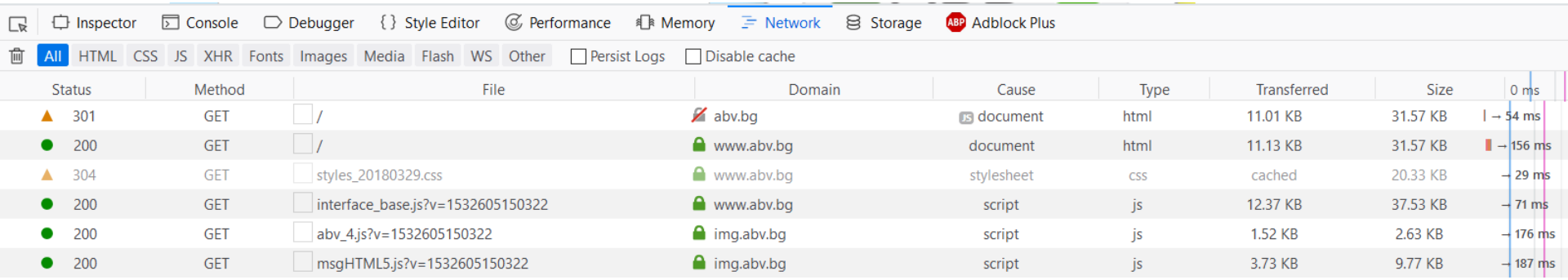
HTTP методи

- При отваряне на web страница, браузърът може да изпълни множество заявки към един или няколко сървъра за изтегляне на различните ресурси, описани в страницата.
- GET- изисква страница от web сървъра с даден URI (Uniform Resource Identifier)
- HEAD- като GET, но не връща body в отговора
- POST - изпращат се данни
- PUT- изпращат се (upload) ресурси към сървъра
- OPTIONS - определя опциите и/или изискванията към ресурсите на сървъра, без да предприема действие
- DELETE – изтрива посочения ресурс (index.html)
- CONNECT – за превключване между тунели с SSL
- TRACE – да изпрати заявка до дестинацията (сървър или първия прокси), която да отговори на клиента

HTTP параметри

- **HTTP Version**
  - HTTP/1.0
- **Uniform Resource Identifiers (URI)**
  - http://abc.com:80/~smith/home.html
  - http://ABC.com/%7Esmith/home.html
  - http://ABC.com:/%7esmith/home.html
- **Date/Time Formats**
- **Character Sets**
  - US-ASCII
  - ISO-8859-1
  - ISO-8859-7
- **Content Encodings**
  - Accept-encoding: gzip
- **Media Types**
  - Accept: image/gif
- **Language Tags**
  - en, en-US, en-cockney

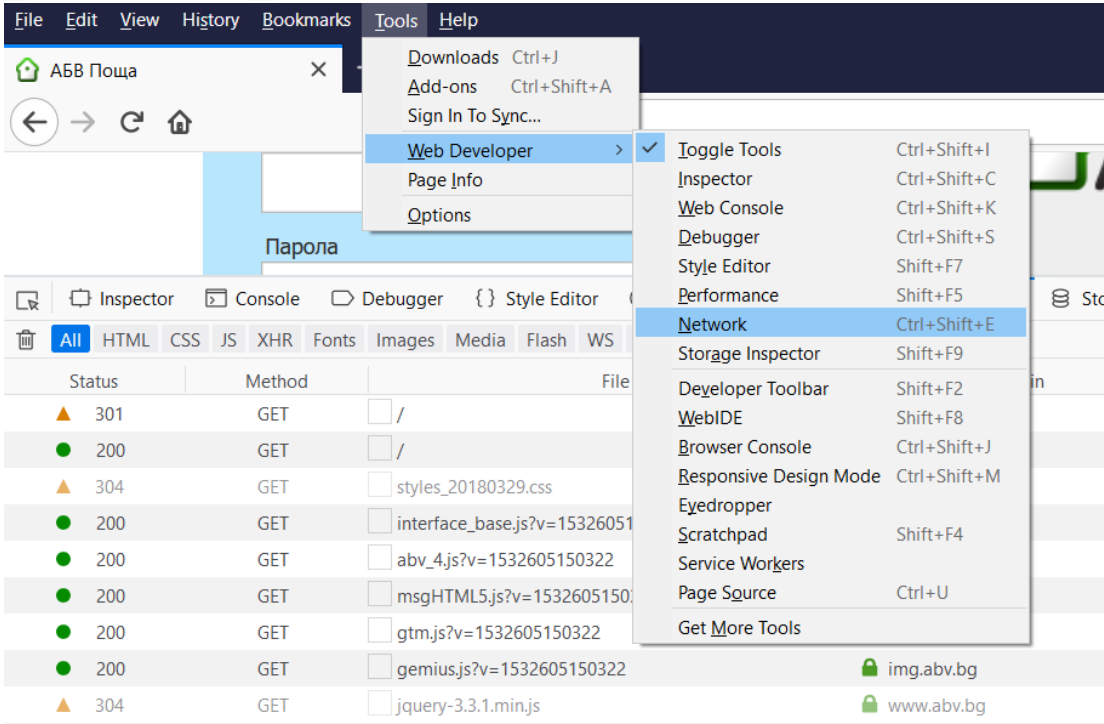
HTTP статус кодове

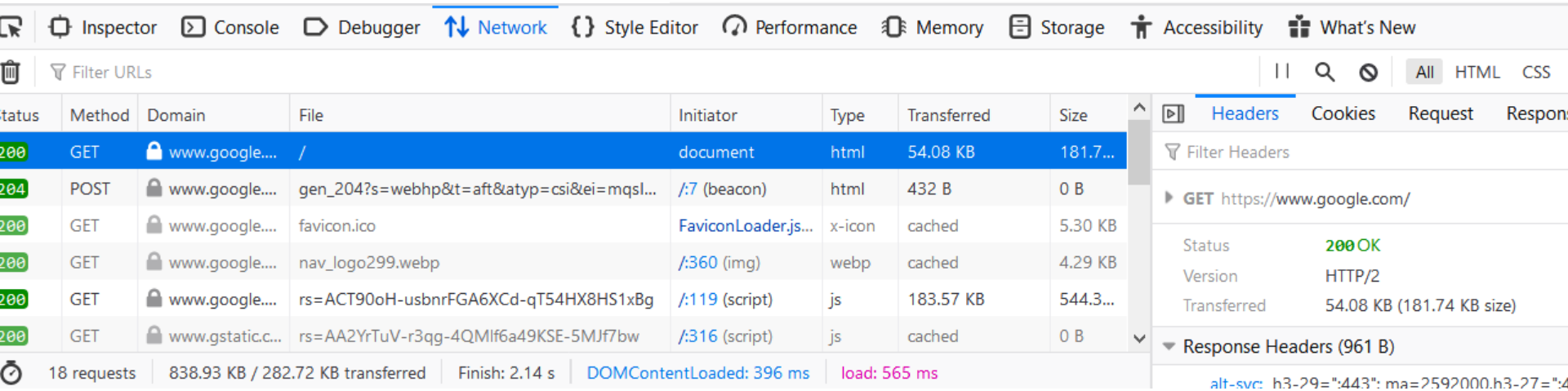


- Статус **2xx** – Success
  (200-OK,201-created…)
- Статус **3xx** – Redirection
  (300-Multiple Choices, 305-Use Proxy)
- Статус **4xx** – Client Error
  (400-Bad request, 404-Not Found)
- Статус **5xx** – Server Error
  (500-Internal Error, 501-Not implemented)

Примери

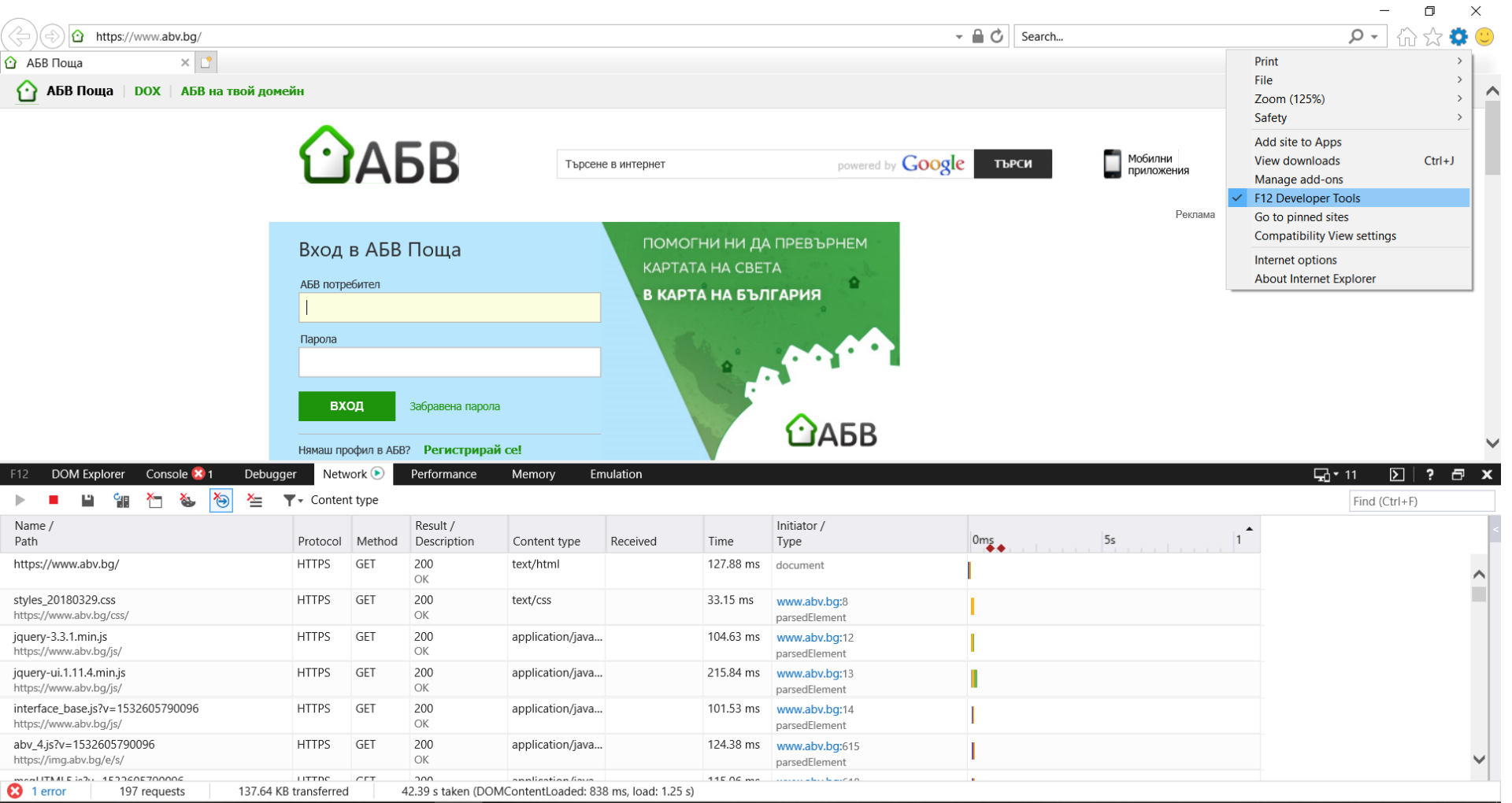
- С Mozilla Firerox





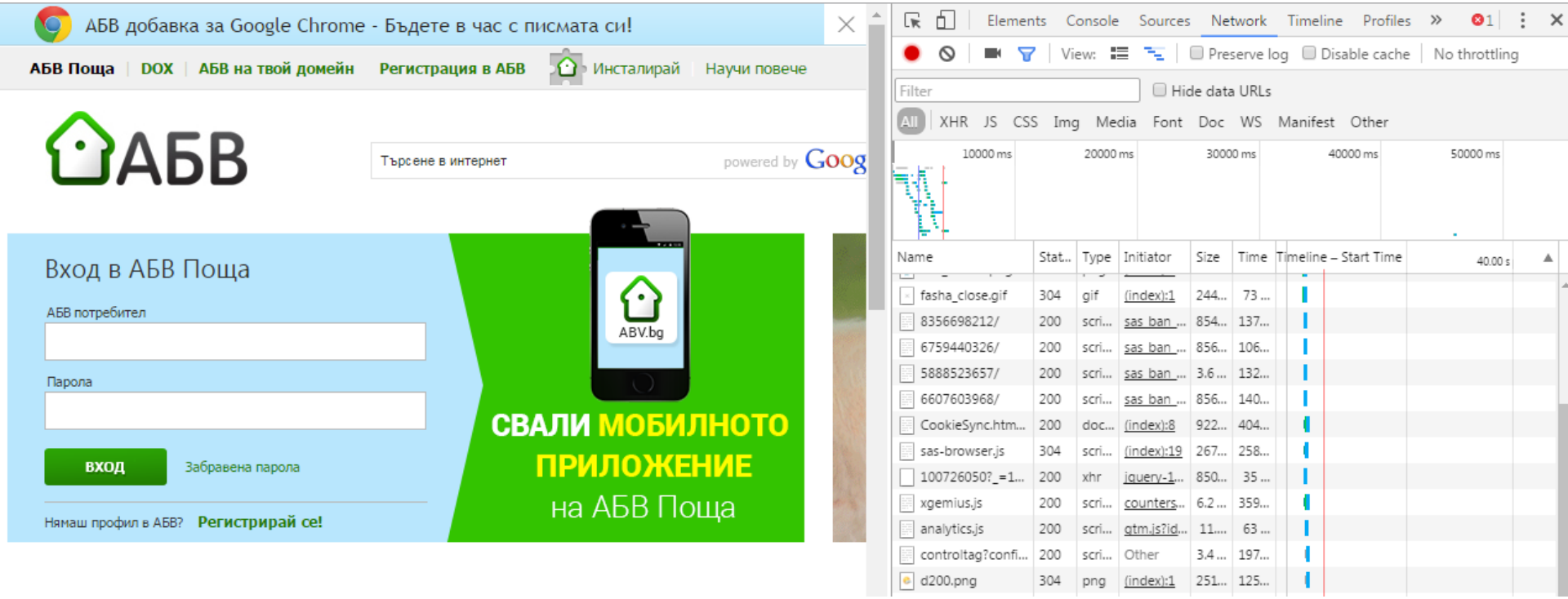
Примери

- С Internet Explorer



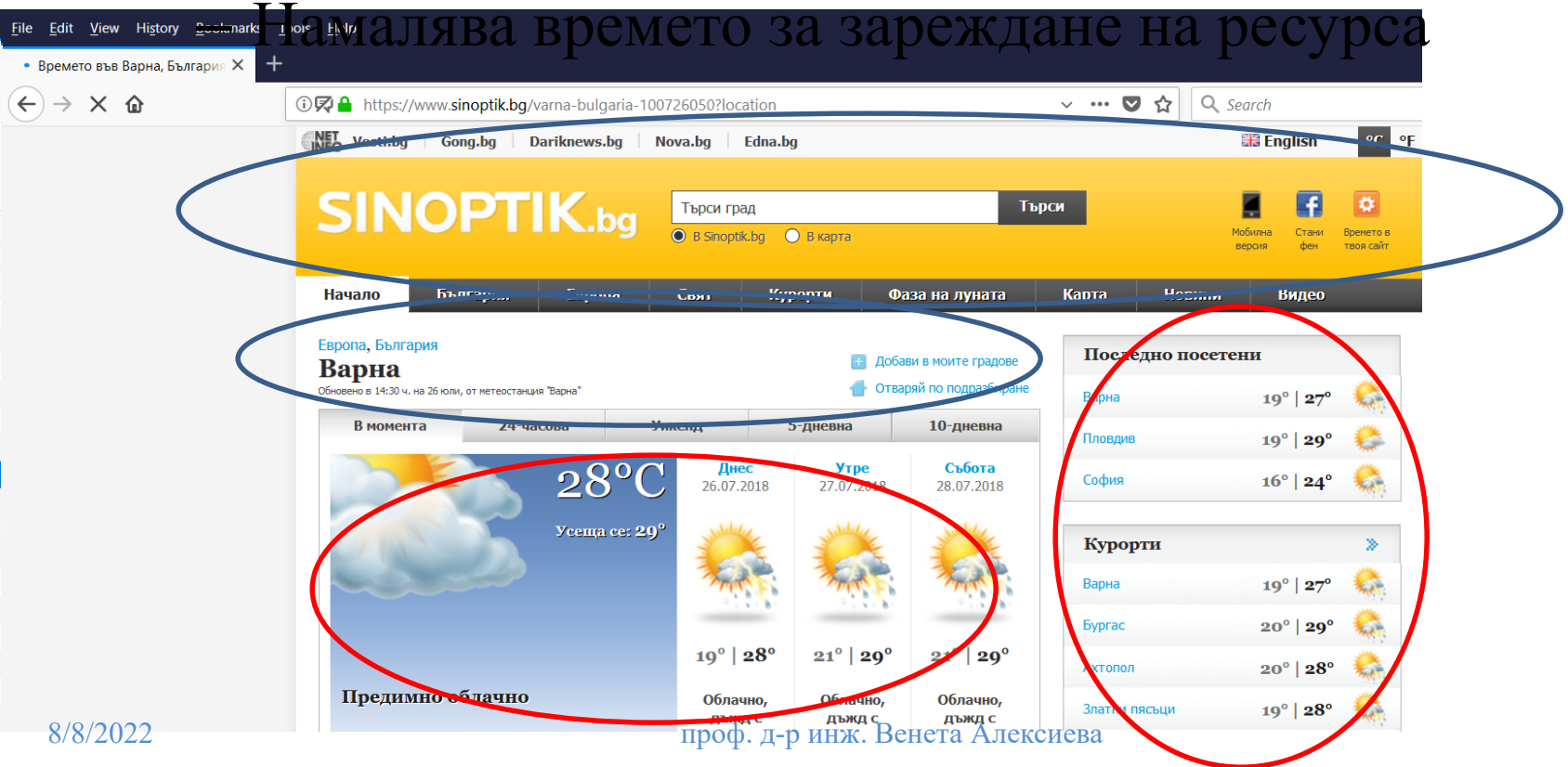
Примери

- С GoogleChrome

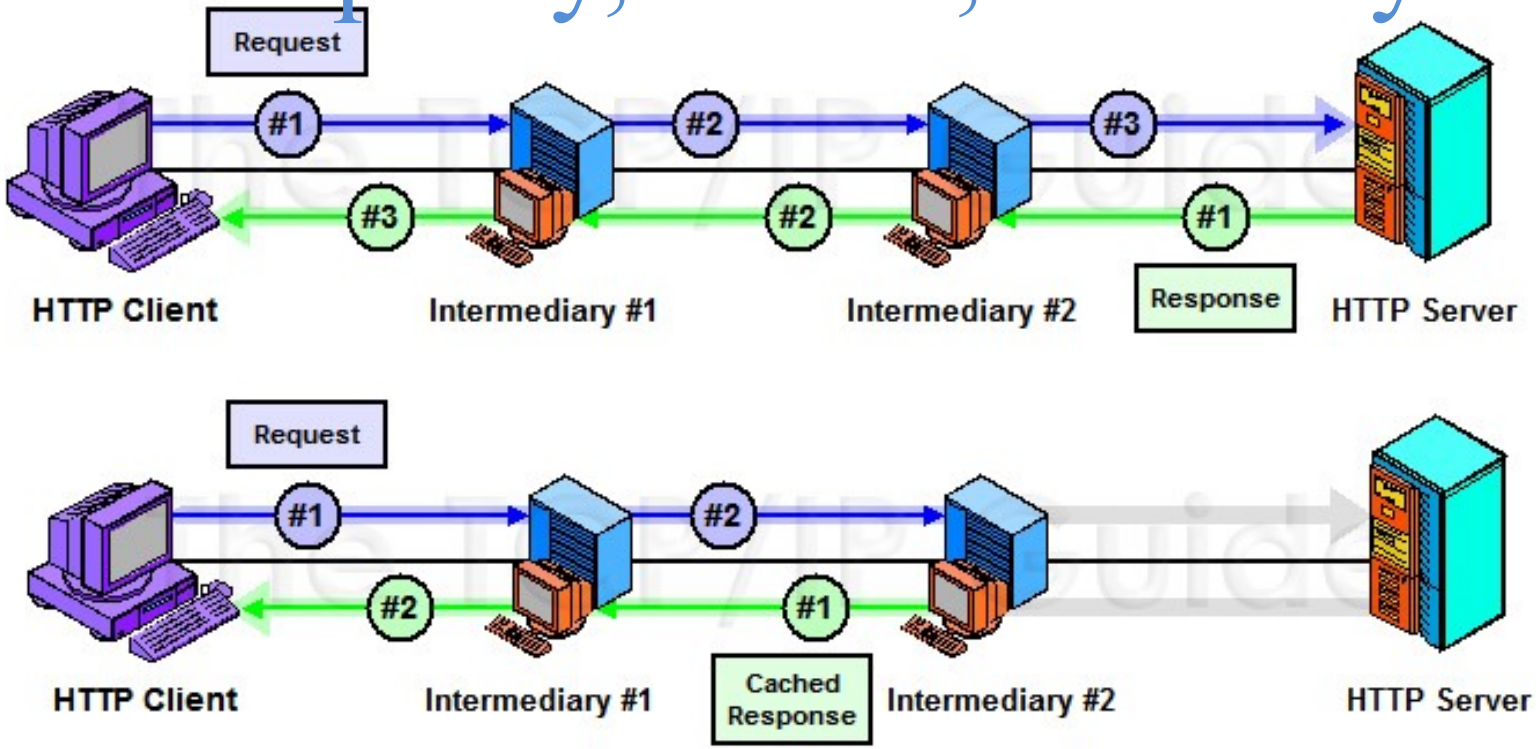


HTTP кеширане

- Потребителите изискват едни и същи документи многократно.
- Повишава се ефективността на протокола като:
  - редуцира натоварването на мрежите
  - Намалява времето за зареждане на ресурса



Web proxy, tunnel, Gateway



1. Заявка от клиента. Клиентът изпраща HTTP заявка към междинното устройство.
2. Заявка от посредника. Посредникът обработва заявката, ако е необходимо прави промени в нея и я препраща към реалния сървър.

I. Отговор от сървъра. Сървърът обработва заявката и изпраща отговор, който се връща на посредника.

II. Отговор от посредника. Може да промени отговора и го праща към клиента.

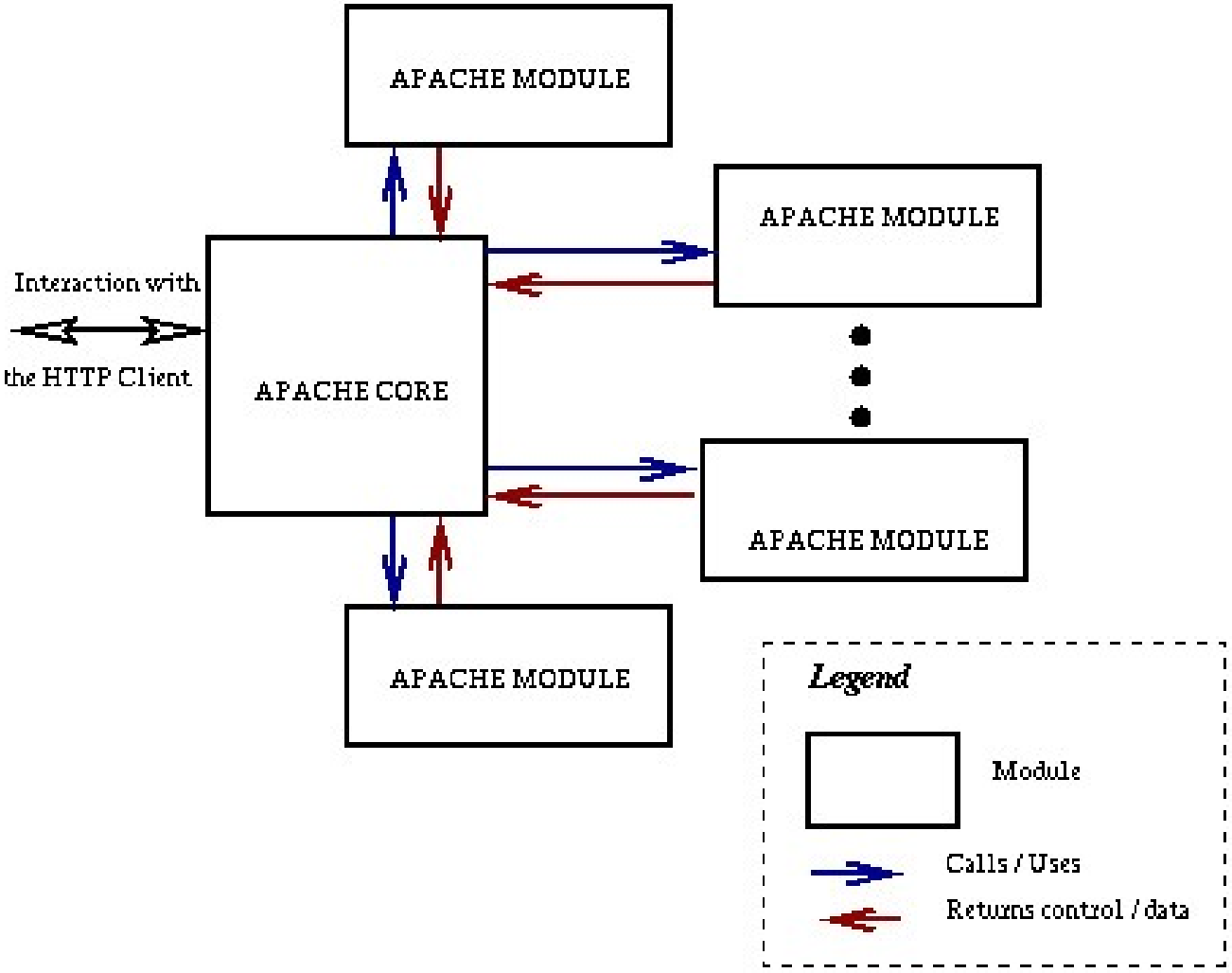
Apache Web сървър

- Apache е един от най-популярните Web сървъри.
- От 1996 и е базиран на NCSA http 1.3 (National Center for Supercomputing Applications).
- Apache е open source проект.
- Разработен под Linux, има и реализация под Windows
- С модулна архитектура
  - Всеки модул реализира само една част от функционаността при обслужването на клиент.
  - За пълното обслужване на заявка са необходими повече от един модул.
  - Всеки отделен модул няма информация за останалите модули.
  - Ядрото реализира основната функционалност на сървъра.

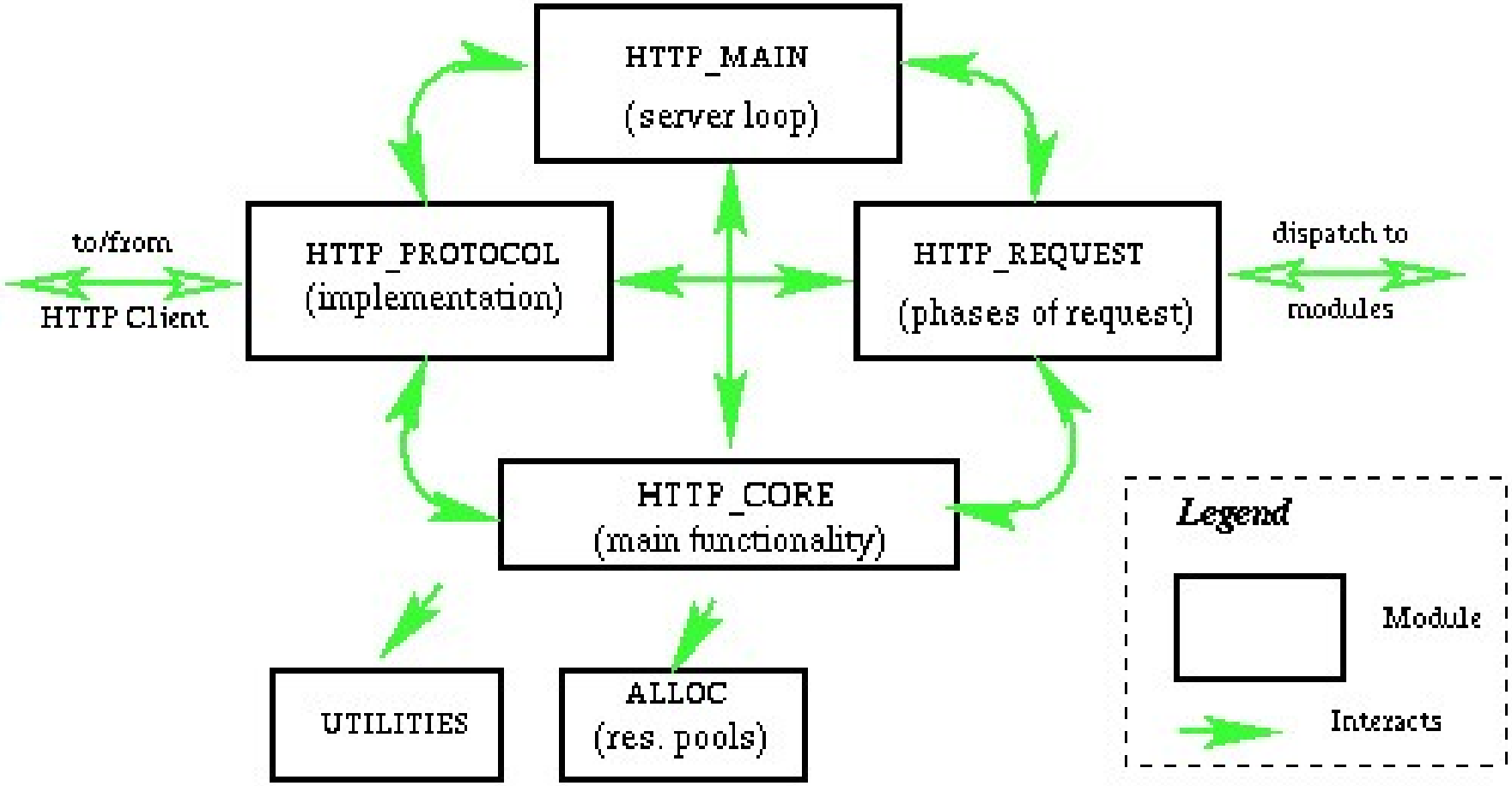
Предимства на модулната архитектура

- бързодействие при обслужване на заявките
- ефективно използване на системни ресурси и памет
- по-добра защитеност от гледна точка на сигурността

Apache архитектура

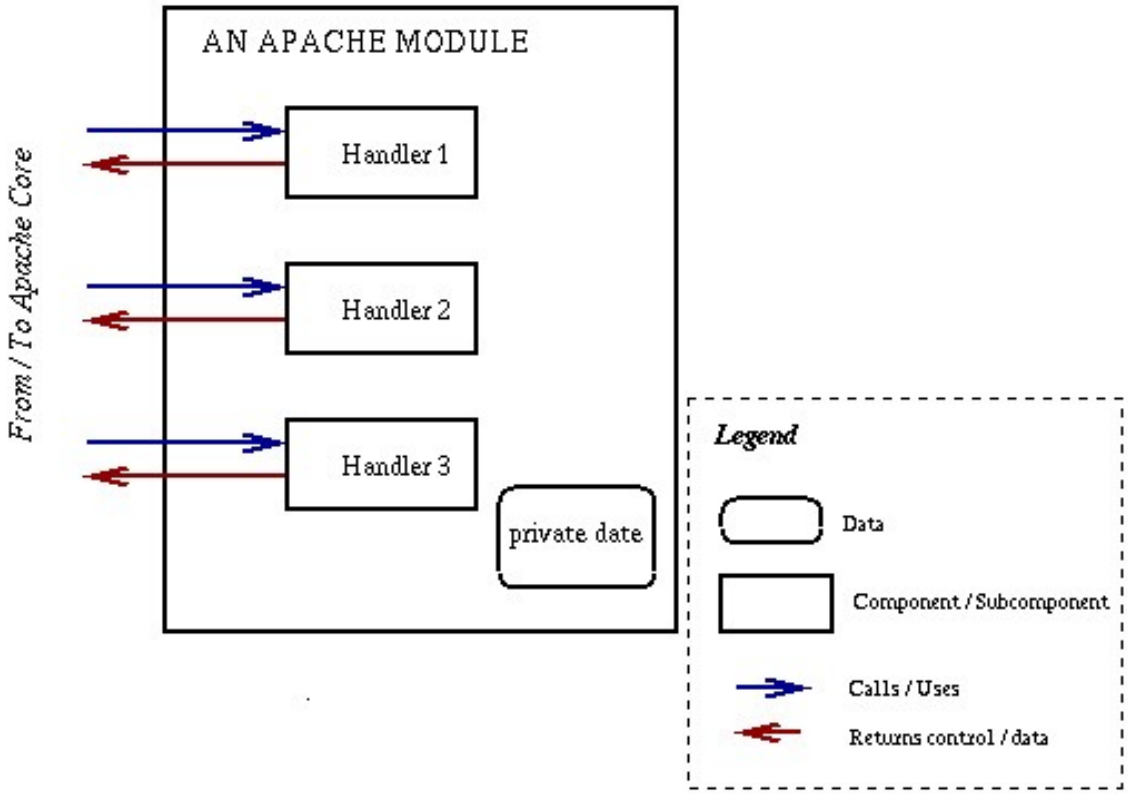


Роля на ядрото



Модули на Apache

- Модулите не взаимодействат директно помежду си, а само през ядрото.
- Всеки модул реализира определена фаза от заявката.
- Handler – действие, което трябва да се изпълни от сървъра.



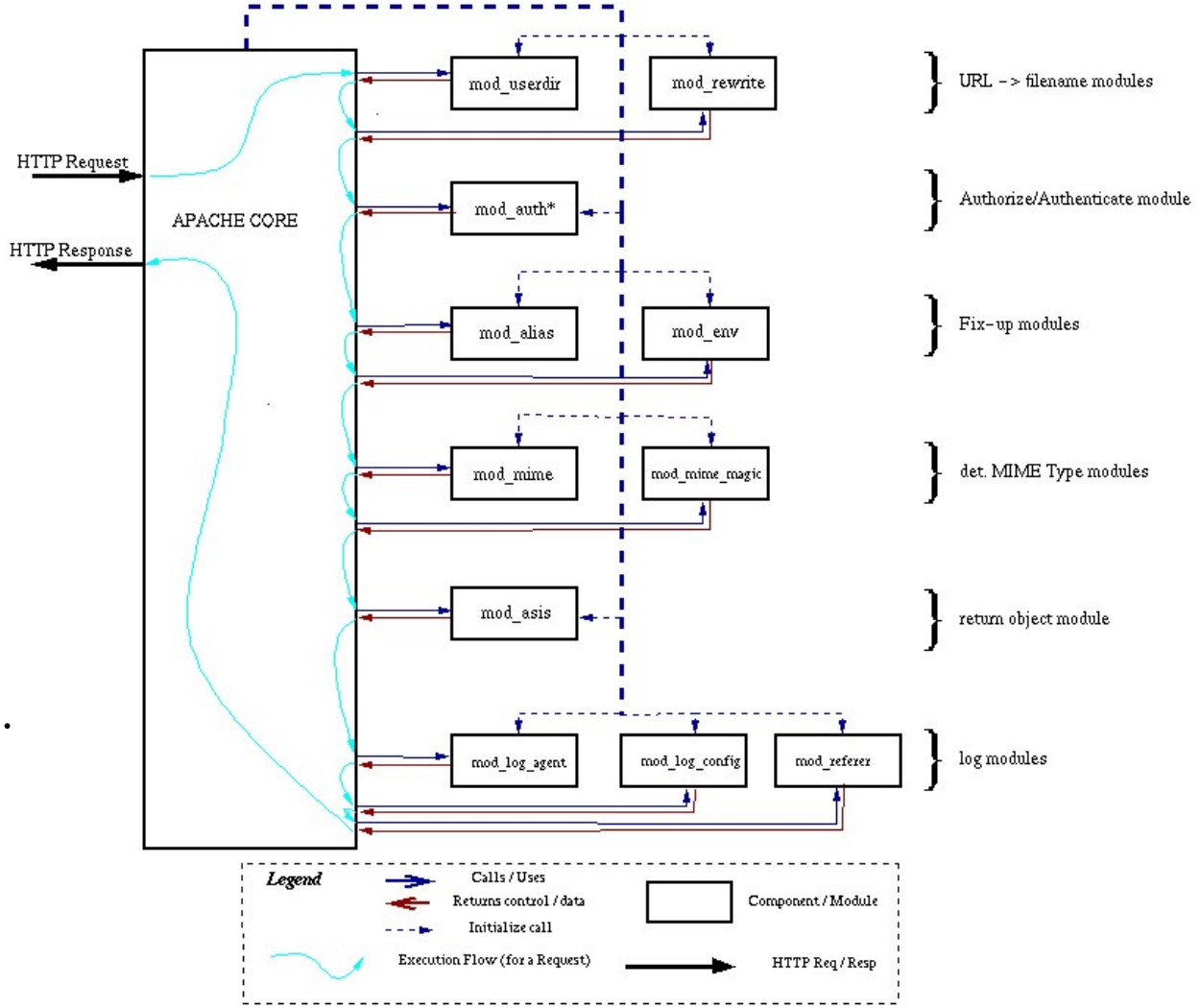
Фази на заявката

- Преобразуване на URL във файлово име;
- Проверка за достъп, базирана на адреса на хоста или друга достъпна информация;
- Получаване на потребителския ID от HTTP заявката и валидирането му;
- Ауторизация на потребителя;
- Определяне на MIME типа на изисквания обект (content type, encoding, language);
- Корекции (напр., замяна на псевдонимите с актуалния път);
- Изпращане на действителните данни обратно към клиента;
- Log-ване на заявката.

Път на обработка на заявката

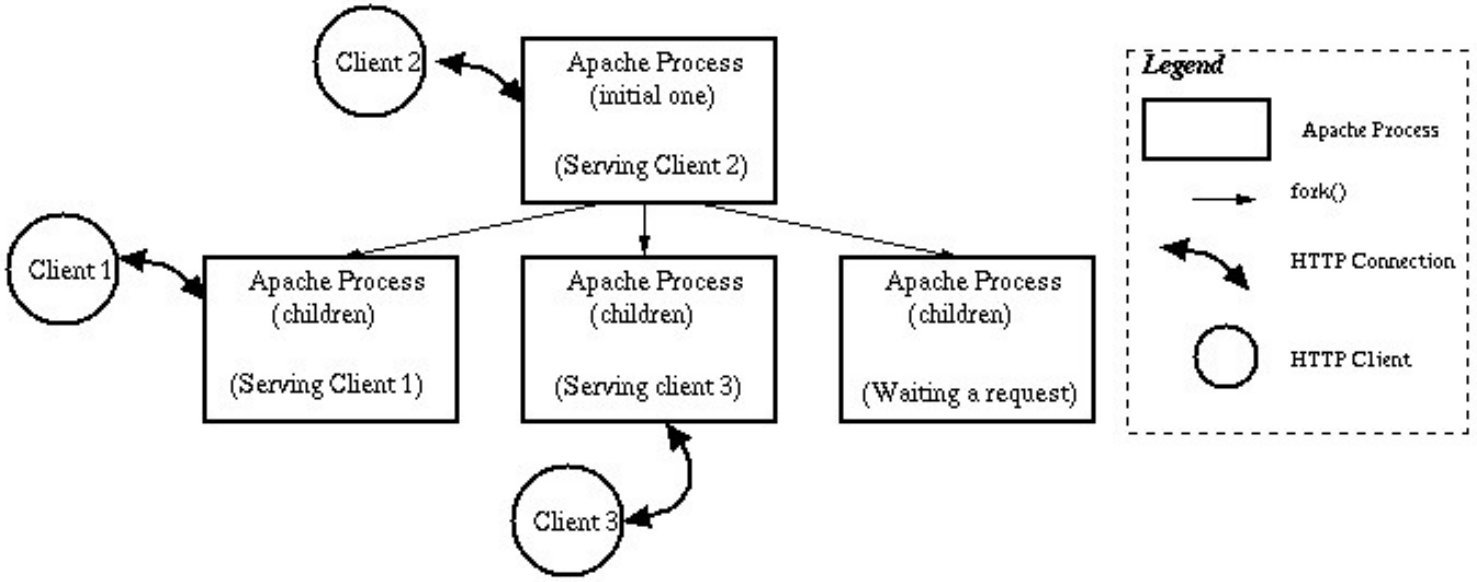
- Няма как да се прескочат фазите

За определени фази може да се извика само един модул (хендлър в модул). Такива фази са:
• ауторизация,
•автентикация,
•връщане на актуалния обект към клиента
•транслиране на UPI в име на файл.
Други фази изискват повече от един хендлър. Когато хендлър върне код за грешка могат да се извикат всички хендлъри.



Паралелност в Apache

- TCP/IP сървърите стартират (fork) нов процес за заявката – голям брой процеси
- Apache използва различна техника - persistent server processes. Той създава още от началото фиксиран брой дъщерни процеси (под Windows това са нишки).



Виртуални хостове

- Apache може да отговаря за повече от едно име, всяко присвоено на един или няколко IP адреса на машината.
- Множество IP адреси могат да бъдат асоциирани с физически мрежови интерфейси или асоциирани с виртуални мрежови (симулирани чрез логически устройства от ОС).
- Apache позволява да се укаже под кое име хостът е бил описан и да се прилагат различни конфигурационни опции.

Конфигуриране на Apache

/etc//httpd/httpd.conf

- прочита се от сървъра при стартиране или рестарт
- първоначално се прочита основният конфигурационен файл *httpd.conf*.
- При достъп до директория се прочита (ако съществува) намиращият се в нея конфигурационен файл за достъп *.htaccess*.
- Модулите са в директорията */etc/httpd/extra*:

  httpd-autoindex.conf *httpd-languages.conf* *httpd-ssl.conf*
  httpd-dav.conf *httpd-manual.conf* *httpd-userdir.conf*
  httpd-default.conf *httpd-mpm.conf* *httpd-vhosts.conf*
  httpd-info.conf *httpd-multilang-errordoc.conf*
- Включват се в основния файл с **include**:

```
include /etc/httpd/extra/httpd-vhosts.conf
```

Директиви

- На отделен ред може да се укаже само една директива.
- Символът за пренасяне е „\”.
- Коментарите се указват със символа „#” като текстът до края на реда се игнорира.
- Директивите не зависят от регистъра на буквите, но при някои техни параметри това е приложимо.
- Директивите могат да се прилагат към целия сървър или частично към отделни директории, файлове и хостове.
- Директивите могат да се групират по секции (контейнери).

Контейнери

- Директивите могат да се обработват по групи, наречени контейнери.
- Типове контейнери:
  - Които се обработват за всяка заявка;
  - Които се обработват само по време на стартиране и рестартиране на сървъра- Условни - **IfDefine**, **IfModule**, **IfVersion**. Ако посочените условия са „истина”, директивите в тях се прилагат.

```
<SectionName [option] >
  Директиви в секцията ...
</SectionName>
...
```

Проверка за коректност

apachectl –t

Пример:

```
    ServerAdmin root@mydomain.com
    ServerRoot "/usr"
#кореннова директория, където се търсят файлове с относителни пътища
#от някои директиви като Include, LoadModule
    Listen 80
#порт на който сървърът очаква заявки. Ако е различен от 80, клиентът
#трябва да включи този порт в заявката (http://www.same.com:8080).

    LoadModule authn_file_module /lib/mod_authn_file.so
    ...

    User apache
    Group apache
    ServerName www.mydomain.com
#текущото регистрирано име на сървъра в DNS
    DocumentRoot "/var/www/htdocs"
#  в посочената директория се търсят документите за заявките
    DirectoryIndex index.html index.php

    ErrorLog "/var/log/httpd/error_log"
```

Условно използване на модули

- Ако е налице модул с указаното име, директивите в условната секция ще бъдат приложени.
- Възможно е задаване на отрицание в тестовото условие чрез използване на знака „!” пред името на модула.

```
<IfModule dir_module>
  DirectoryIndex index.html
  DirectoryIndex index.php
</IfModule>
```

Контрол на достъпа до файлове и директории

- Контролът на достъпа е чрез директивите **Allow** (разреши) или **Deny** (забрани).
- **Order** задава реда на тяхното прилагане.
- Източникът на заявките е конкретен IP адрес, част от адрес, име на домейн, част от домейн или всичко.

```
Order deny, allow
Deny from 200.100.50.5
Deny from 10.20.30
Deny from spammers.domain.com
Deny from example.com
Allow from all
```

Допълнителен контрол на достъпа

- Файловете за допълнителен контрол на достъпа до директории и файлове *.htaccess* и *.htpasswd* е необходимо да се забранят да са видими от клиентите чрез следната секция:

```
<FilesMatch "^\.ht">
  Order allow,deny
  Deny from all
</FilesMatch>
```

Достъп до директории

- От версия 2.0 Apache има рестриктивна политика за достъп към директориите по подразбиране.
- Директиви, които се прилагат единствено по отношение на зададената директория, нейните поддиректории и съдържащите се в тях файлове:

```
<Directory "/var/named/htdocs/cisco">
        Списък от директиви ...
 </Directory>
```

Пример:

```
<Directory />
  AllowOverride None
  Options FollowSymLinks
  Order deny, allow
  Deny from all
</Directory>
```

Достъп до файлове

- Файловете за допълнителен контрол на достъпа до директории и файлове *.htaccess* и *.htpasswd* трябва да се забранят да се виждат от клиентите чрез следната секция:

<FilesMatch"^\.ht">

#всички айлове, започващис “.ht”

Order *allow,deny*

Deny from *all*

</FilesMatch>

Виртуални хостове

- Apache притежава възможност повече от един web сайтове да се изпълняват на една и съща машина. Това е известно като поддържане на виртуални хостове.
- Възможни са два типа виртуални хостове:
  - **IP-базирани -** . За всеки виртуален хост е необходимо наличието на отделен IP адрес
  - **базирани на име** (именувани)- клиентът трябва да включи името на хоста като част от HTTP хедъра. Така различни виртуални хостове споделят един IP адрес, но трябва да имат отделна регистрация в DNS.

Конфигуриране на виртуални хостове

/etc/httpd/extra/httpd-vhosts.conf

Но трябва да се включи с `include` в */etc//httpd/httpd.conf*

- За да се използват именувани виртуални хостове, трябва да се определи IP адреса на хоста и порта. Това става с директивата **NameVirtualHost *:80**. Ако (нормално) се използва само един IP адрес, този аргумент (*) е достатъчен.
- За всеки виртуален хост се създават отделни секции чрез директивата **VirtualHost**. Нейният аргумент трябва да е аналогичен на този в **NameVirtualHost**.
- Проверка на виртуалните хостове:

 # httpd -S

Пример за виртуални хостове

За всеки виртуален хост е зададена отделна директория, в която ще са разположени файловете за неговия сайт.

```
NameVirtualHost *:80

<VirtualHost *:80>
   ServerName www.domain1.com
   DocumentRoot /var/www/htdocs/domain1
   ServerAlias *.domain1.com
   ErrorLog /var/log/httpd/domain1.log
</VirtualHost>

<VirtualHost *:80>
   ServerName www.domain2.com
   DocumentRoot /var/www/htdocs/domain2
   ServerAlias server.domain2.com
   ErrorLog /var/log/httpd/domain2.log
</VirtualHost>
```

Въпроси ?

Благодаря за вниманието !