

Web сървър

проф. д-р инж. Венета Алексиева

ОСНОВНИ МОМЕНТИ

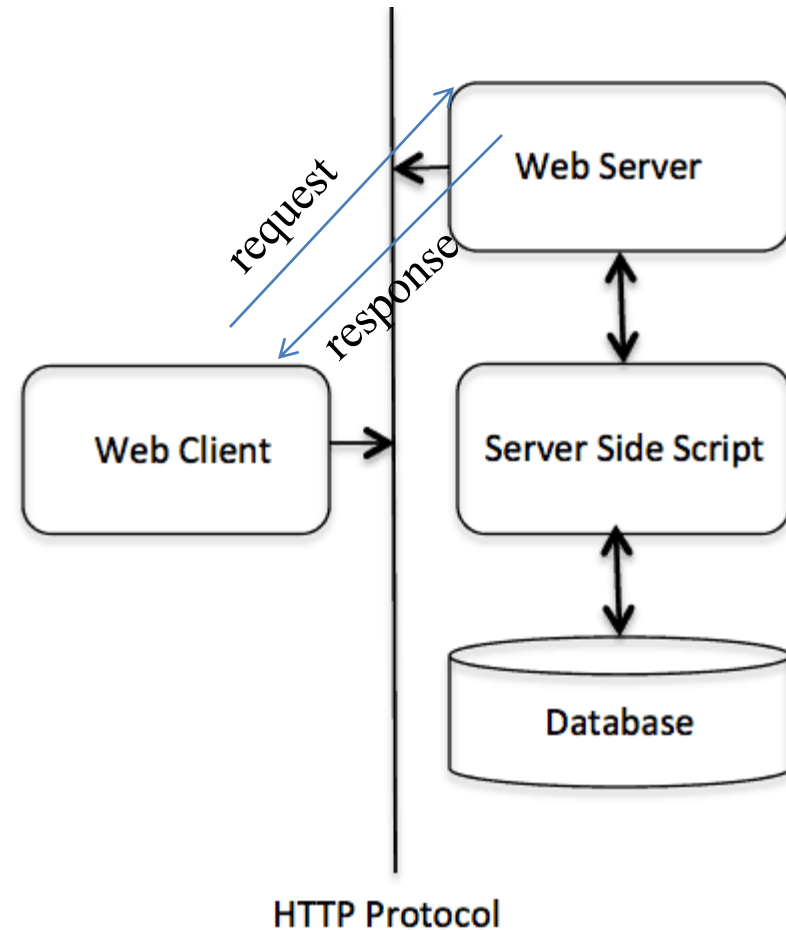
- HTTP протокол
- Web сървър
- Конфигуриране на Apache

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://www.nasa.gov/exploration/home/index.html



протокол име на сървър местоположение на ресурса

HTTP преди

- HTTP/1.0 всяка ТСР 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

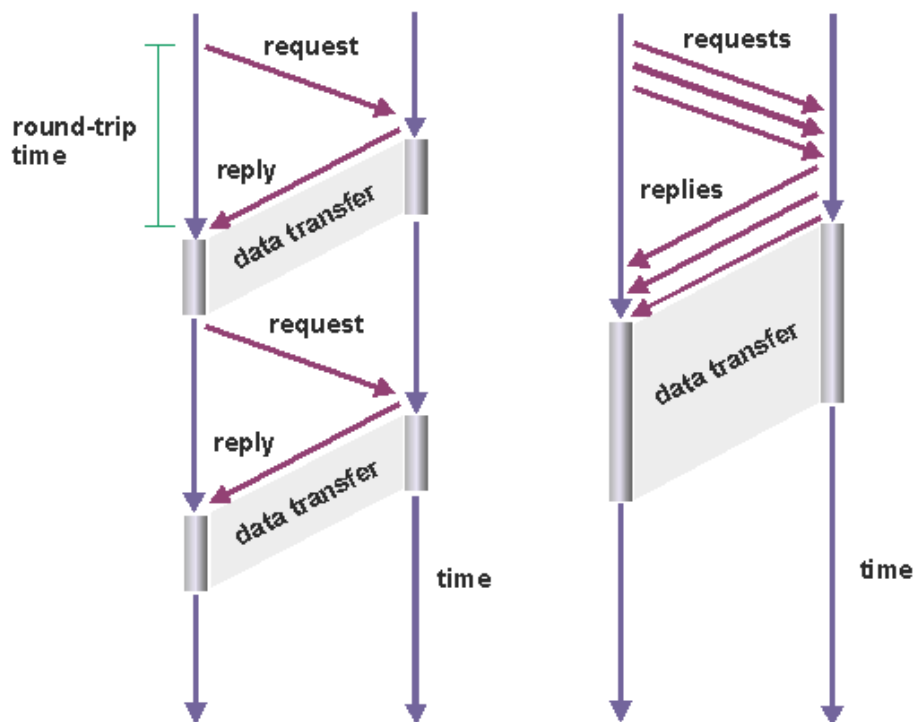
- Connectionless – доставя се ресурса и се прекъсва сесията
- Независим от преносната среда
- Stateless- не поддържа информация за състоянието, не помни взаимодействието с клиента
- Проблем, когато сесията трябва да бъде удължена – решава се със сесии и cookies
- Няма механизъм за управление на диалога
- 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 ТСП 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 комуникация 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.



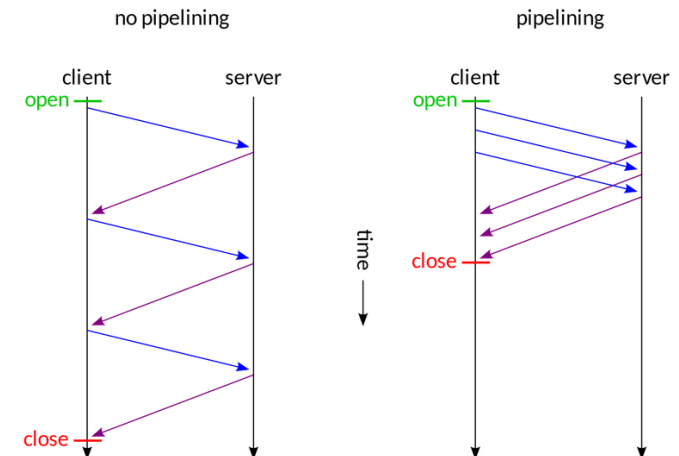
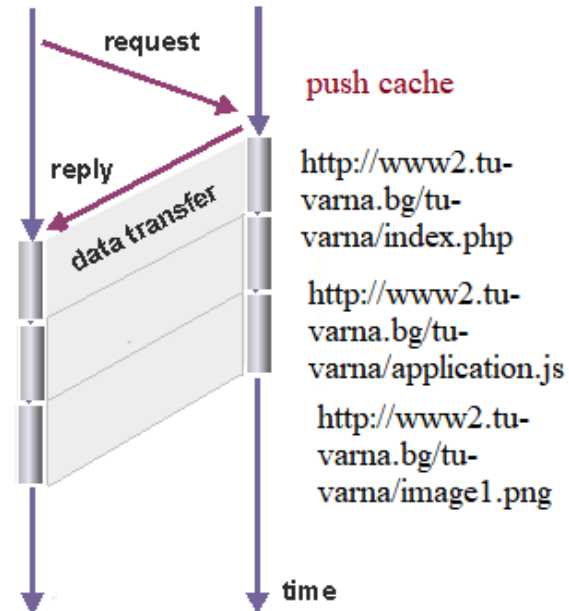
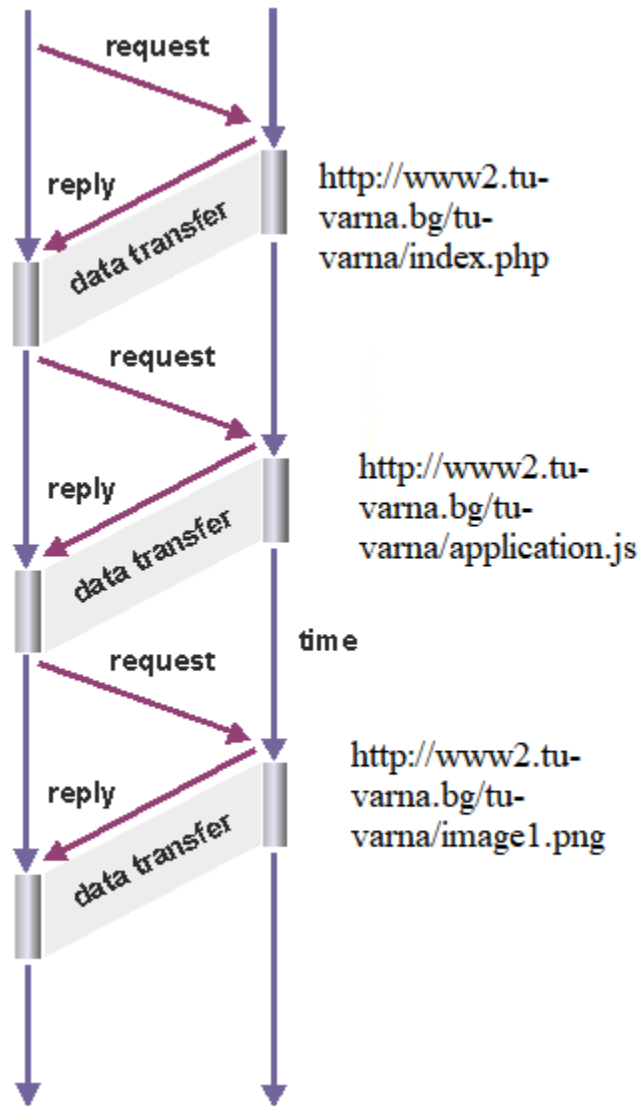
HTTP2.0

- 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

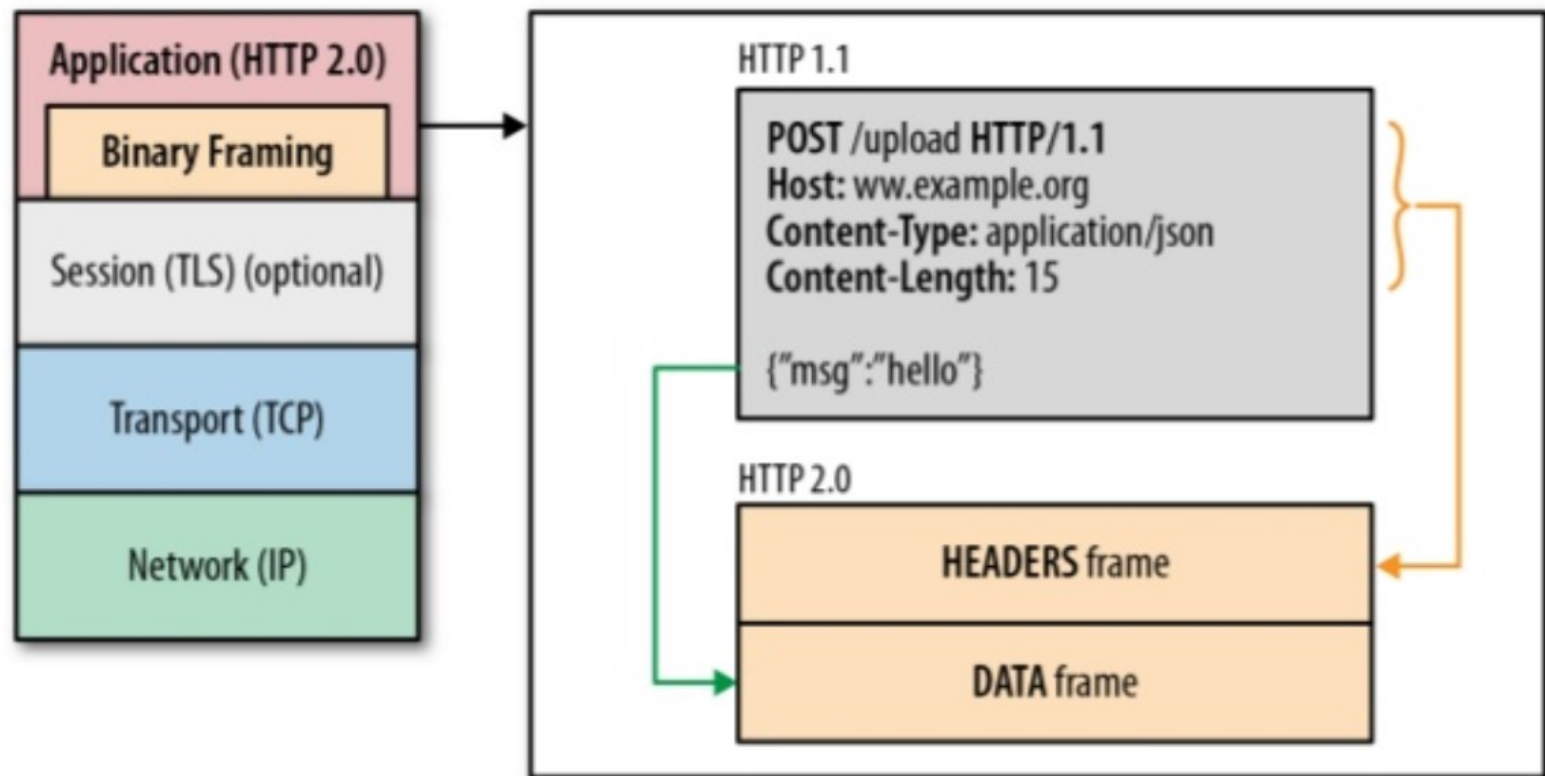
- Пренася се от транспортен протокол Quick UDP Internet Connections (QUIC), разработен от Google, различен от TCP, с предимства като:
 - вградено мултиплексиране,
 - per-stream flow control,
 - установява конекция с low-latency
 - включва TLS 1.3, а не го ползва като отделен слой.
- Запазва предимствата 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

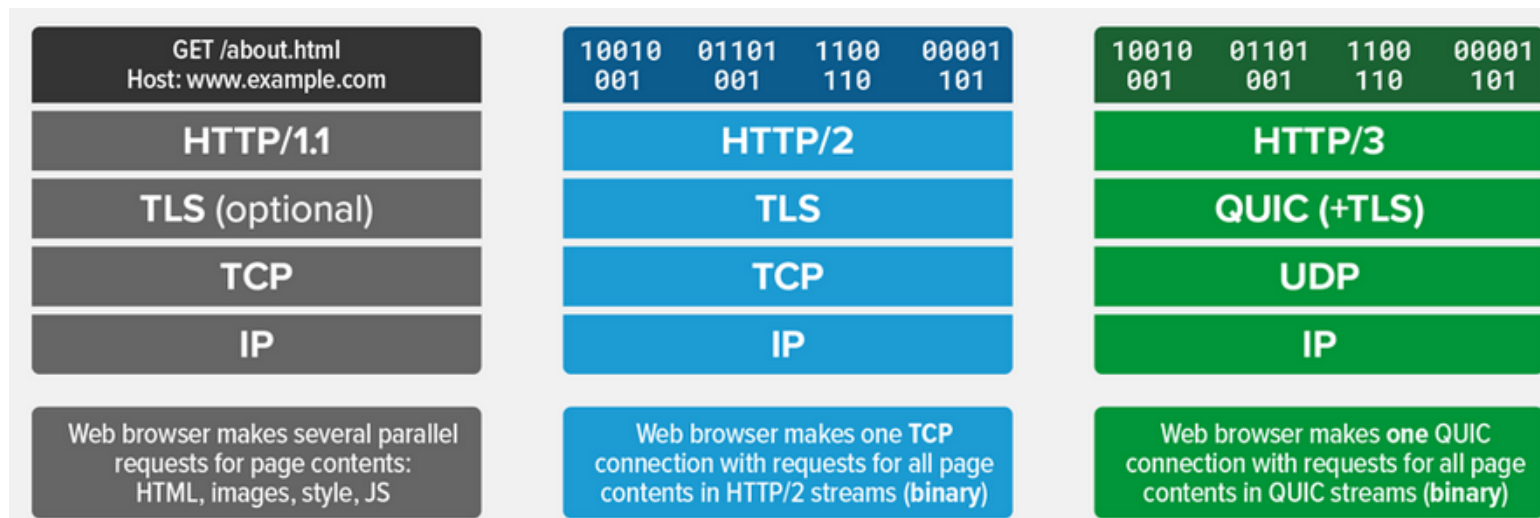
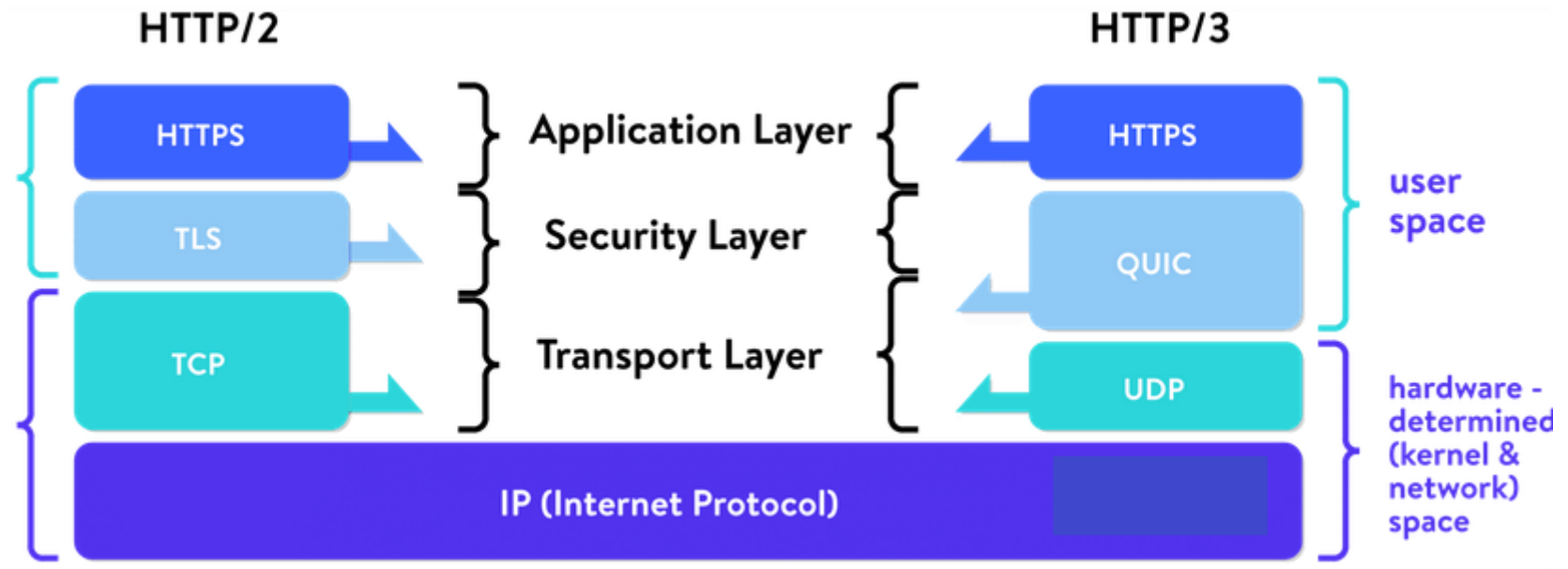


HTTP1.1 и HTTP2.0/3.0

- Приоритизация
- Flow control
- Server push

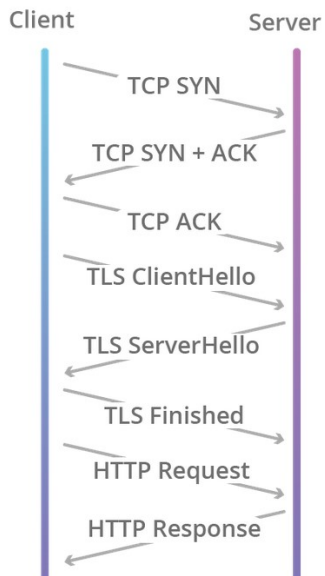


HTTP2.0 и HTTP3.0 протоколен стек

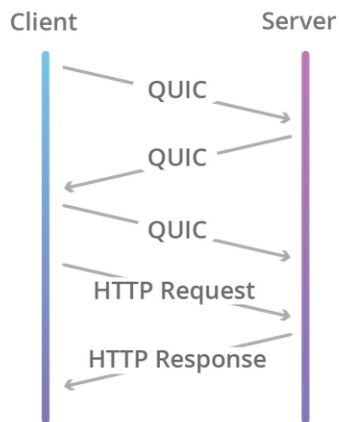


Договаряне на версията на HTTP

HTTP Request Over TCP + TLS

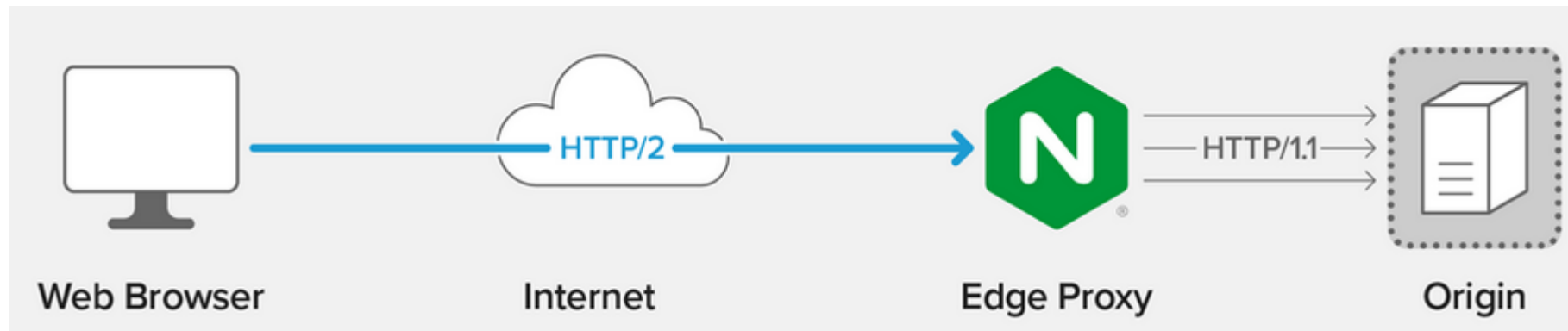


HTTP Request Over QUIC



- 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.

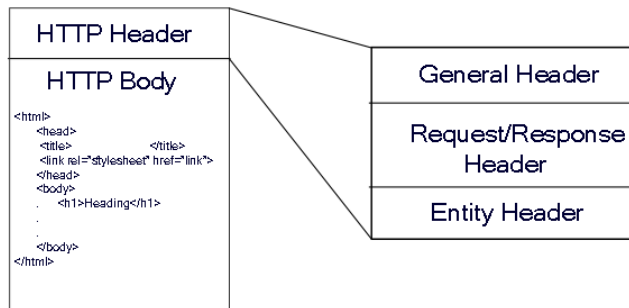
Хибридна употреба



HTTP1.1 съобщения- общ формат

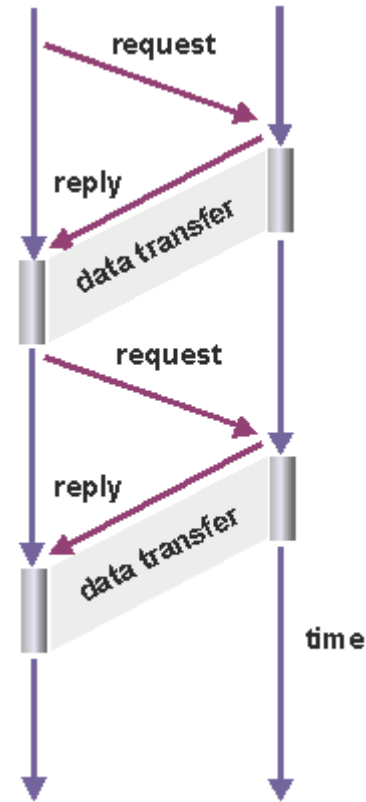
- HTTP Generic Message Format- текстов.
Текстовите редове завършват с CR/LF
контролни символи.

HTTP Request/response



`<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>

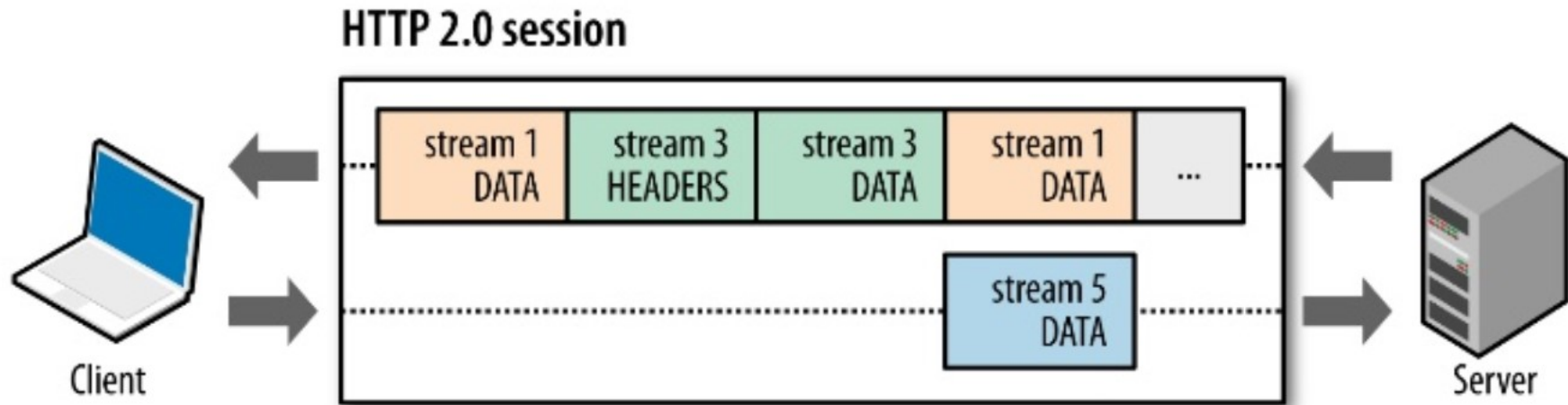
<entity-headers>

<empty-line>

[<message-body>]

[<message-trailers>]

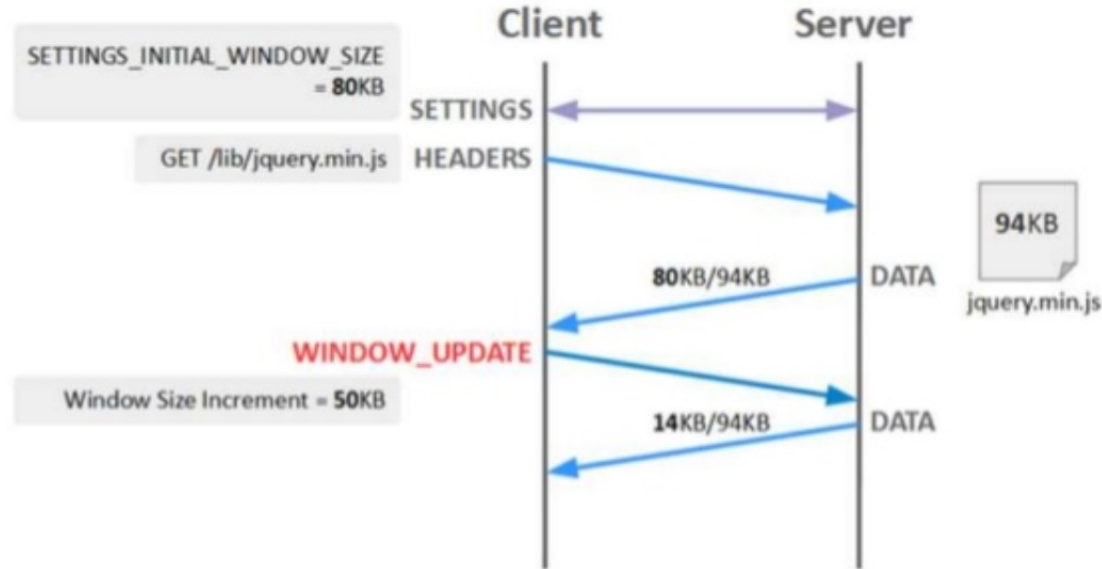
HTTP2.0/3.0- поток от съобщения



- 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.

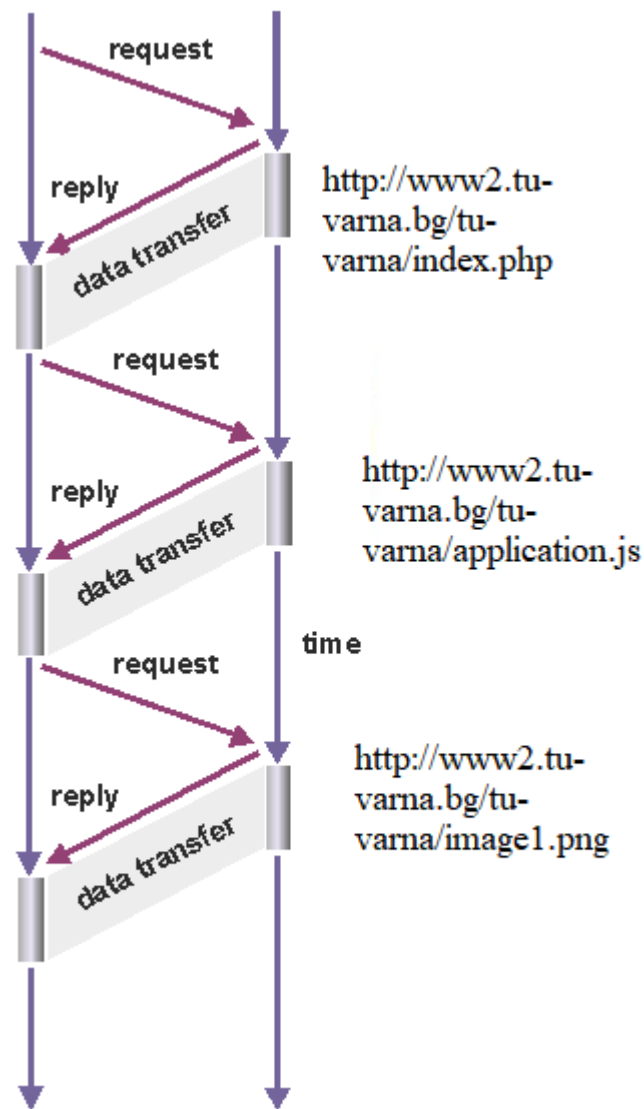
HTTP1.0 headers-request/response

The screenshot shows the 'Headers' tab in a web browser's developer tools. The request is a GET to `http://www2.tu-varna.bg/tu-varna/`. The status is **200 OK** (circled in blue) and the version is **HTTP/1.1**. The response headers include:

- 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

The request headers include:

- 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.

The screenshot shows the Chrome DevTools Network tab with the 'Headers' sub-tab selected. The request is a GET to <https://www.google.com/>. The status is 200 OK. The response headers are expanded, showing various metadata like cache-control, content-encoding, content-length, content-type, date, expires, server, set-cookie, strict-transport-security, X-Firefox-Spdy, x-frame-options, and x-xss-protection.

GET <https://www.google.com/>

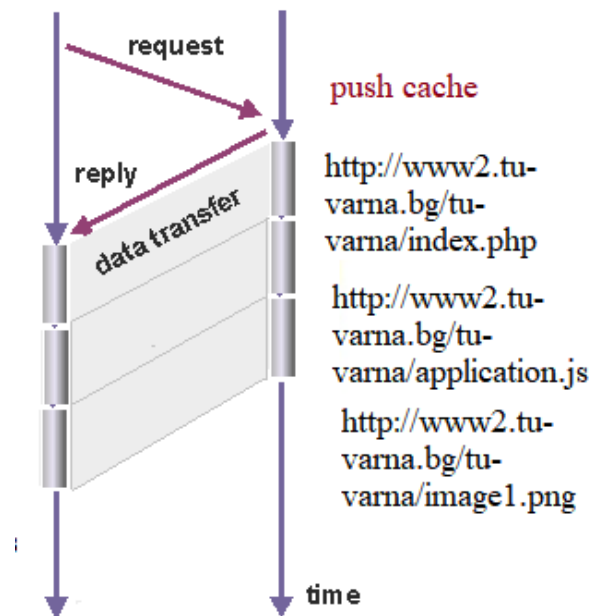
Status: 200 OK
Version: HTTP/2
Transferred: 54.09 KB (181.74 KB size)

Response Headers (961 B)

- alt-svc: h3-29=":443"; ma=2592000,h3-27=":443"; ma=2592000,h3-25=":443"; ma=2592000,h3-T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
- cache-control: private, max-age=0
- content-encoding: br
- content-length: 54426
- content-type: text/html; charset=UTF-8
- date: Fri, 10 Jul 2020 17:38:11 GMT
- expires: -1
- server: gws
- set-cookie: 1P_JAR=2020-07-10-17; expires=Sun, 09-Aug-2020 17:38:11 GMT; path=/; domain=.google.com; Secure; SameSite=none
- set-cookie: OTZ=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=www.google.com
- set-cookie: OTZ=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=.www.google.com
- set-cookie: OTZ=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=google.com
- set-cookie: OTZ=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=.google.com
- strict-transport-security: max-age=31536000
- X-Firefox-Spdy: h2
- x-frame-options: SAMEORIGIN
- x-xss-protection: 0

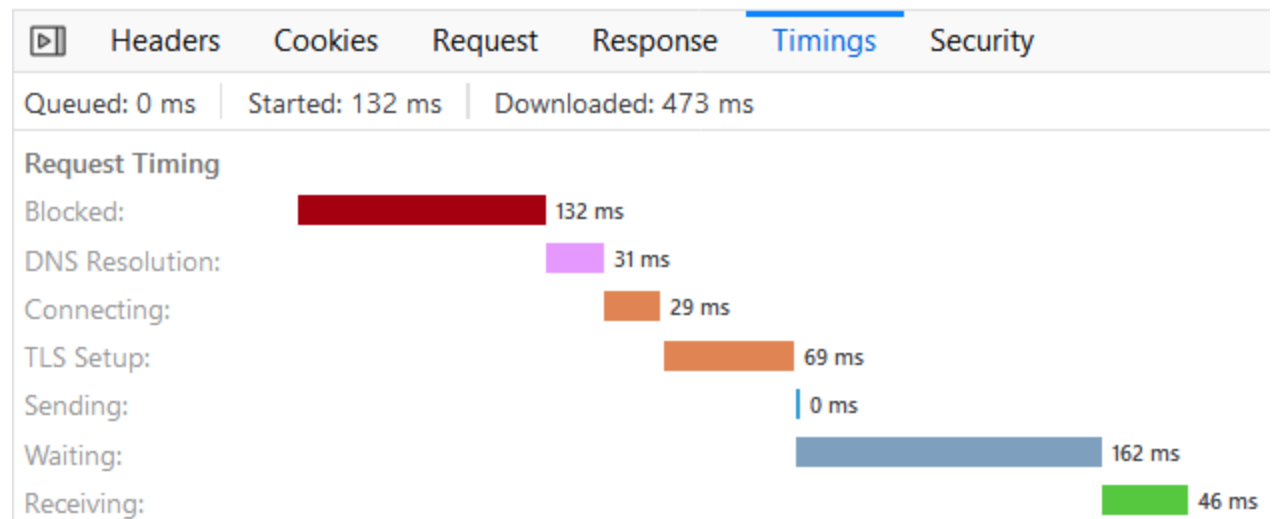
Request Headers (765 B)

- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8



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>					



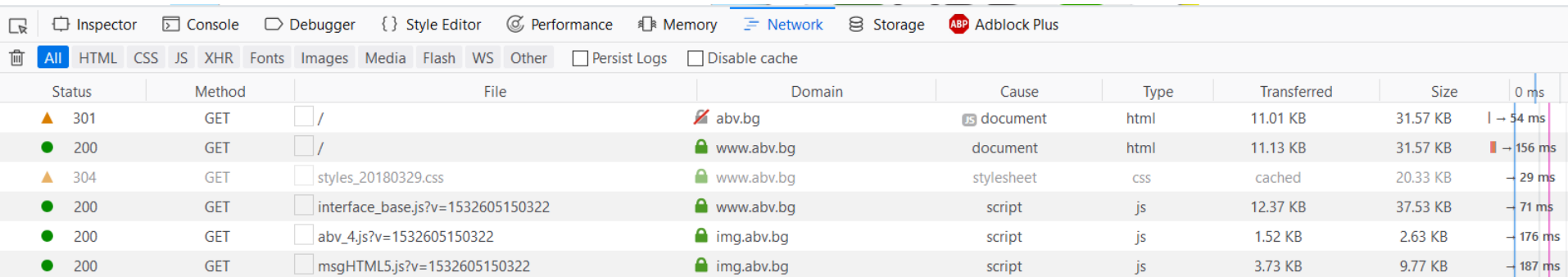
HTTP методи

- 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 сървър 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 сървър
- OPTIONS - определя опциите и/или изискванията към ресурсите на сървър, 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



The screenshot shows the Network tab of a web browser's developer tools. It displays a list of HTTP requests with columns for Status, Method, File, Domain, Cause, Type, Transferred, Size, and Time. The requests are as follows:

Status	Method	File	Domain	Cause	Type	Transferred	Size	Time
301	GET	/	abv.bg	document	html	11.01 KB	31.57 KB	54 ms
200	GET	/	www.abv.bg	document	html	11.13 KB	31.57 KB	156 ms
304	GET	styles_20180329.css	www.abv.bg	stylesheet	css	cached	20.33 KB	29 ms
200	GET	interface_base.js?v=1532605150322	www.abv.bg	script	js	12.37 KB	37.53 KB	71 ms
200	GET	abv_4.js?v=1532605150322	img.abv.bg	script	js	1.52 KB	2.63 KB	176 ms
200	GET	msgHTML5.js?v=1532605150322	img.abv.bg	script	js	3.73 KB	9.77 KB	187 ms

- 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 Firefox

The screenshot shows the Mozilla Firefox browser interface with the Developer Tools open. The 'Tools' menu is open, highlighting the 'Web Developer' option. The 'Network' tab is selected in the Developer Tools, showing a list of requests. The 'Headers' sub-tab is active, displaying the response headers for a GET request to 'https://www.google.com/'.

Tools Menu:

- Downloads Ctrl+J
- Add-ons Ctrl+Shift+A
- Sign In To Sync...
- Web Developer** >
- Page Info
- Options

Web Developer Sub-menu:

- Toggle Tools Ctrl+Shift+I
- Inspector Ctrl+Shift+C
- Web Console Ctrl+Shift+K
- Debugger Ctrl+Shift+S
- Style Editor Shift+F7
- Performance Shift+F5
- Network Ctrl+Shift+E**
- Storage Inspector Shift+F9
- Developer Toolbar Shift+F2
- WebIDE Shift+F8
- Browser Console Ctrl+Shift+J
- Responsive Design Mode Ctrl+Shift+M
- Eyedropper
- Scratchpad Shift+F4
- Service Workers
- Page Source Ctrl+U
- Get More Tools

Network Tab:

Status	Method	File
301	GET	/
200	GET	/
304	GET	styles_20180329.css
200	GET	interface_base.js?v=15326051
200	GET	abv_4.js?v=1532605150322
200	GET	msgHTML5.js?v=1532605150
200	GET	gtm.js?v=1532605150322
200	GET	gemius.js?v=1532605150322
304	GET	jquery-3.3.1.min.js

Headers Tab:

GET https://www.google.com/

Status: 200 OK

Version: HTTP/2

Transferred: 54.08 KB (181.74 KB size)

Response Headers (961 B)

alt-svc: h3-29=":443"; ma=2592000,h3-27=":4

Примери

- C Internet Explorer

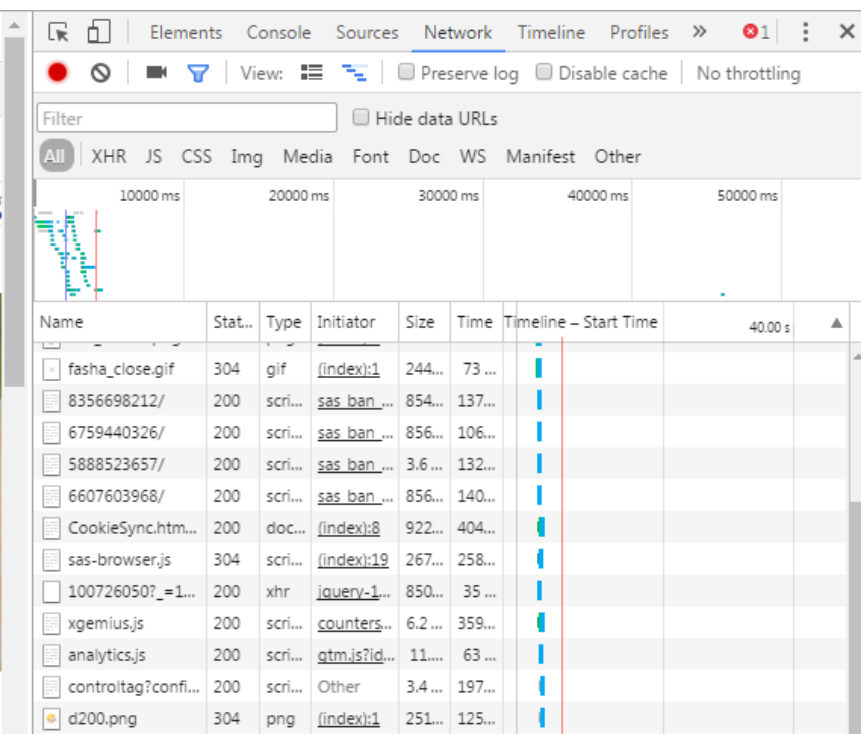
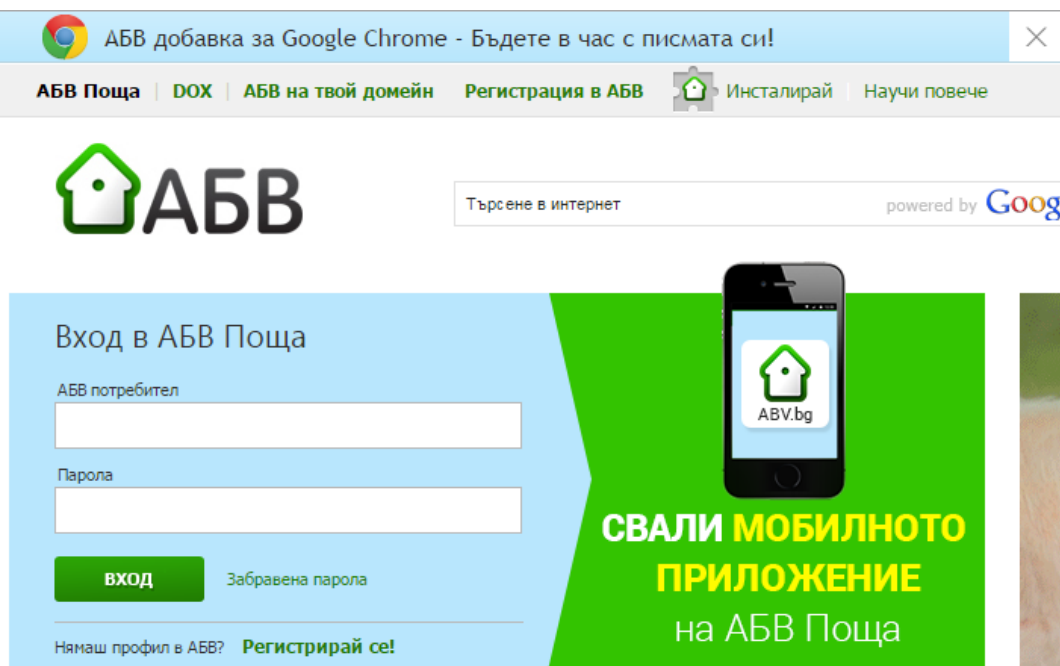
The screenshot shows the Internet Explorer browser window displaying the ABV website. The address bar shows <https://www.abv.bg/>. The page content includes the ABV logo, a search bar, and a login section titled "Вход в АБВ Поща". The F12 Developer Tools menu is open, showing options like Print, File, Zoom, and F12 Developer Tools (which is selected). The Network tab is active, showing a list of requests.

Name / Path	Protocol	Method	Result / Description	Content type	Received	Time	Initiator / Type
https://www.abv.bg/	HTTPS	GET	200 OK	text/html		127.88 ms	document
styles_20180329.css https://www.abv.bg/css/	HTTPS	GET	200 OK	text/css		33.15 ms	www.abv.bg:8 parsedElement
jquery-3.3.1.min.js https://www.abv.bg/js/	HTTPS	GET	200 OK	application/java...		104.63 ms	www.abv.bg:12 parsedElement
jquery-ui.1.11.4.min.js https://www.abv.bg/js/	HTTPS	GET	200 OK	application/java...		215.84 ms	www.abv.bg:13 parsedElement
interface_base.js?v=1532605790096 https://www.abv.bg/js/	HTTPS	GET	200 OK	application/java...		101.53 ms	www.abv.bg:14 parsedElement
abv_4js?v=1532605790096 https://img.abv.bg/e/s/	HTTPS	GET	200 OK	application/java...		124.38 ms	www.abv.bg:615 parsedElement

1 error 197 requests 137.64 KB transferred 42.39 s taken (DOMContentLoaded: 838 ms, load: 1.25 s)

Примери

- С GoogleChrome



НТТР кеширане

- Потребителите изискват едни и същи документи многократно.
- Повишава се ефективността на протокола като:
 - редуцира натоварването на мрежите

Намалява времето за зареждане на ресурса

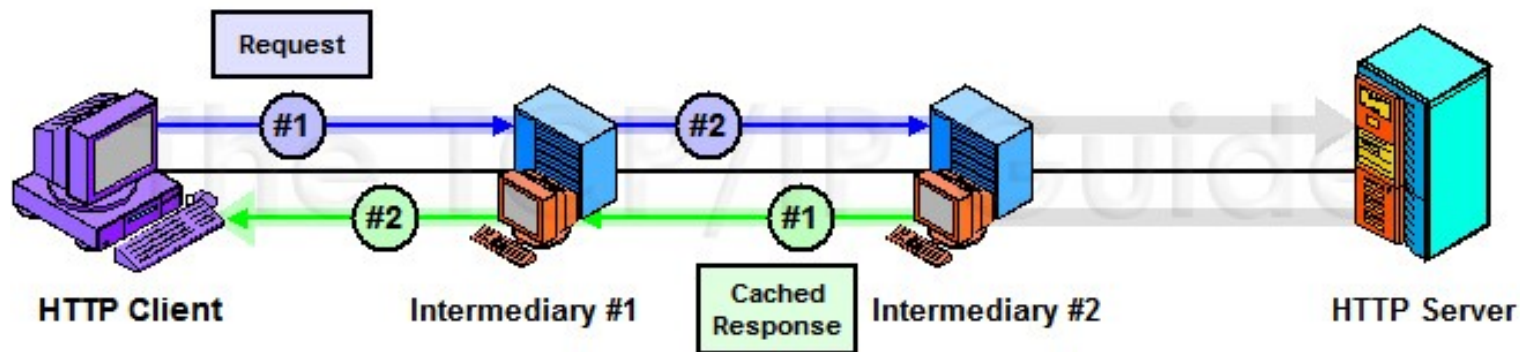
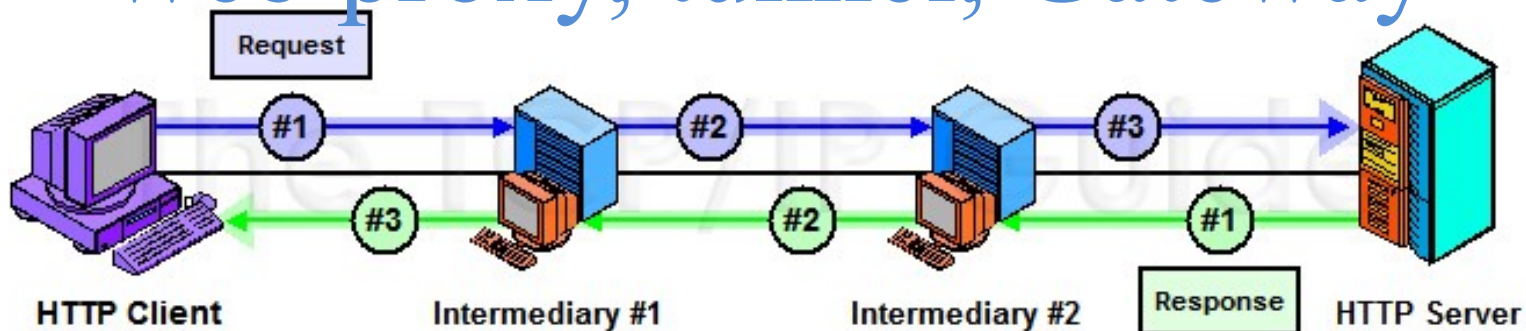
The screenshot shows the SINOPTIK.bg website interface. The URL bar displays <https://www.sinoptik.bg/varna-bulgaria-100726050?location>. The page features a yellow header with the SINOPTIK.bg logo and a search bar. Below the header, there is a navigation menu with links like 'Начало', 'България', 'Европа', 'Свят', 'Курорти', 'Фаза на луната', 'Карта', 'Новости', and 'Видео'. The main content area shows weather data for 'Варна' (Varna), Bulgaria, updated at 14:30 on July 26, 2018. It includes a current weather section with a temperature of 28°C and a forecast for the next 10 days. A table on the right lists 'Последно посетени' (Recently visited) locations and a 'Курорти' (Resorts) section.

В момента	24-часова	Уикенд	5-дневна	10-дневна
28°C Усеща се: 29° Предимно облачно				
	Днес 26.07.2018	Утре 27.07.2018	Събота 28.07.2018	
	19° 28° Облачно, дъжд с	21° 29° Облачно, дъжд с	21° 29° Облачно, дъжд с	

Последно посетени		
Варна	19° 27°	☀️
Пловдив	19° 29°	☀️
София	16° 24°	☀️

Курорти		
Варна	19° 27°	☀️
Бургас	20° 29°	☀️
Духтопол	20° 28°	☀️
Златни пясъци	19° 28°	☀️

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.

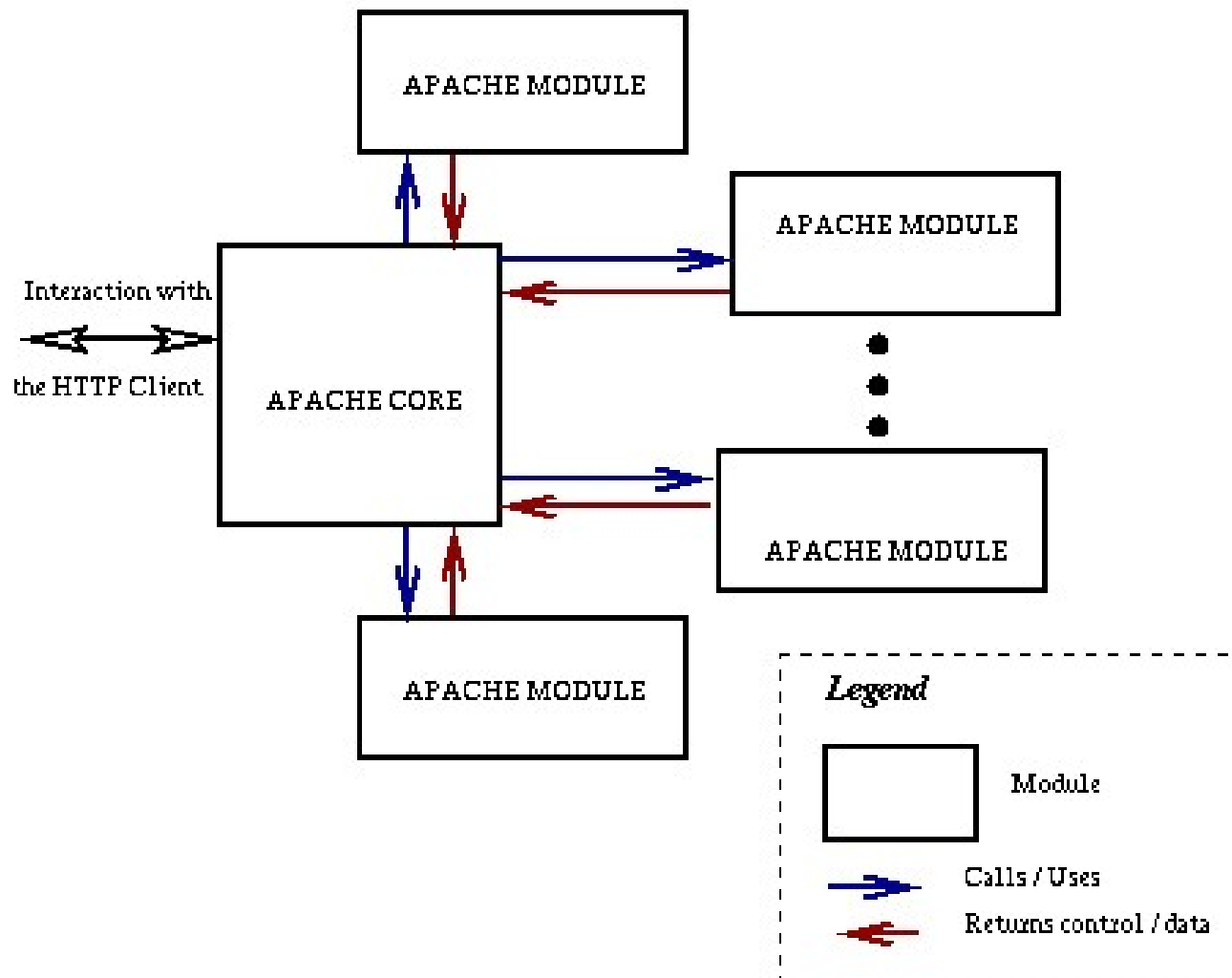
Apache Web сървър

- 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 функционалността 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.

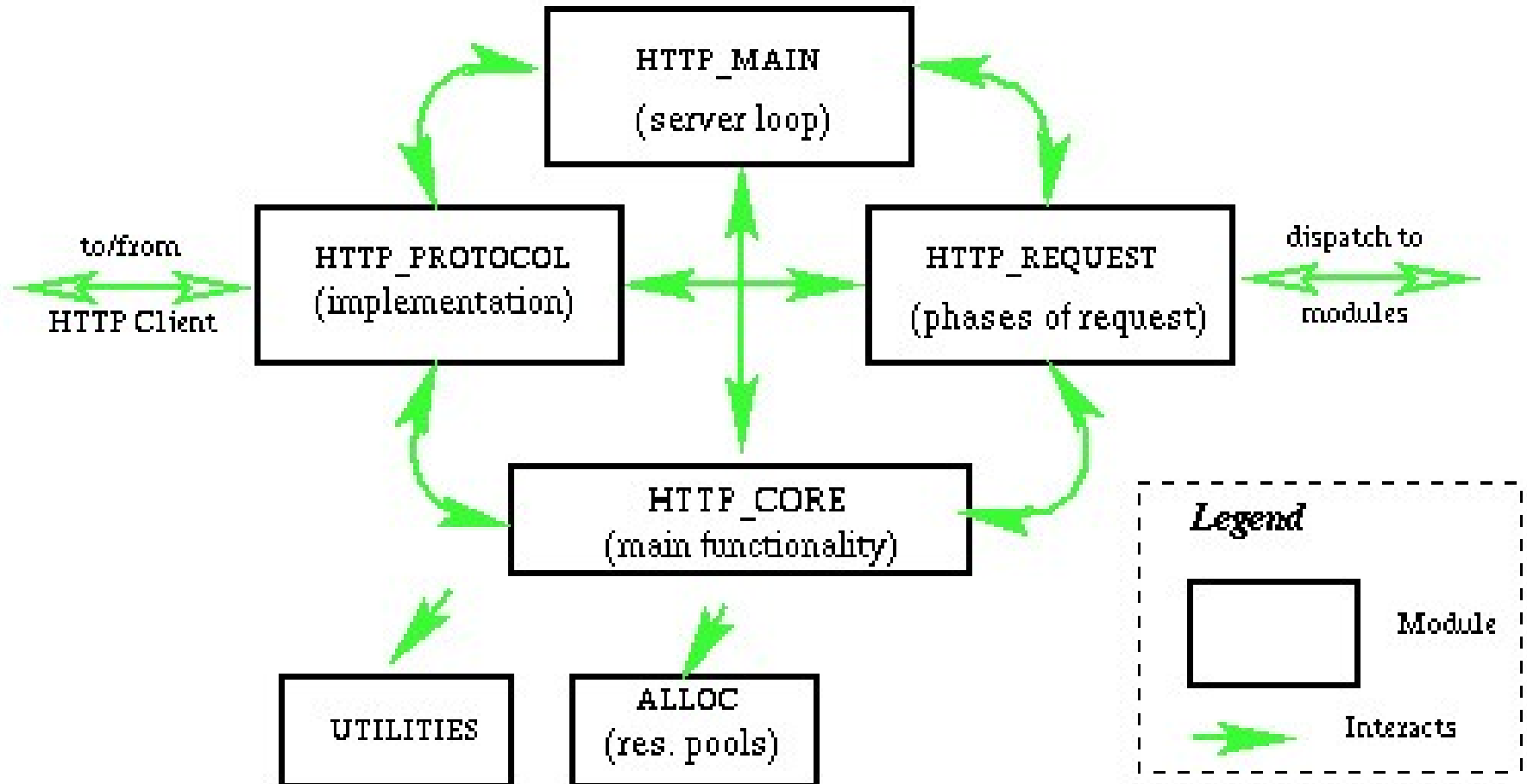
Предимства на модулната архитектура

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

Apache архитектура

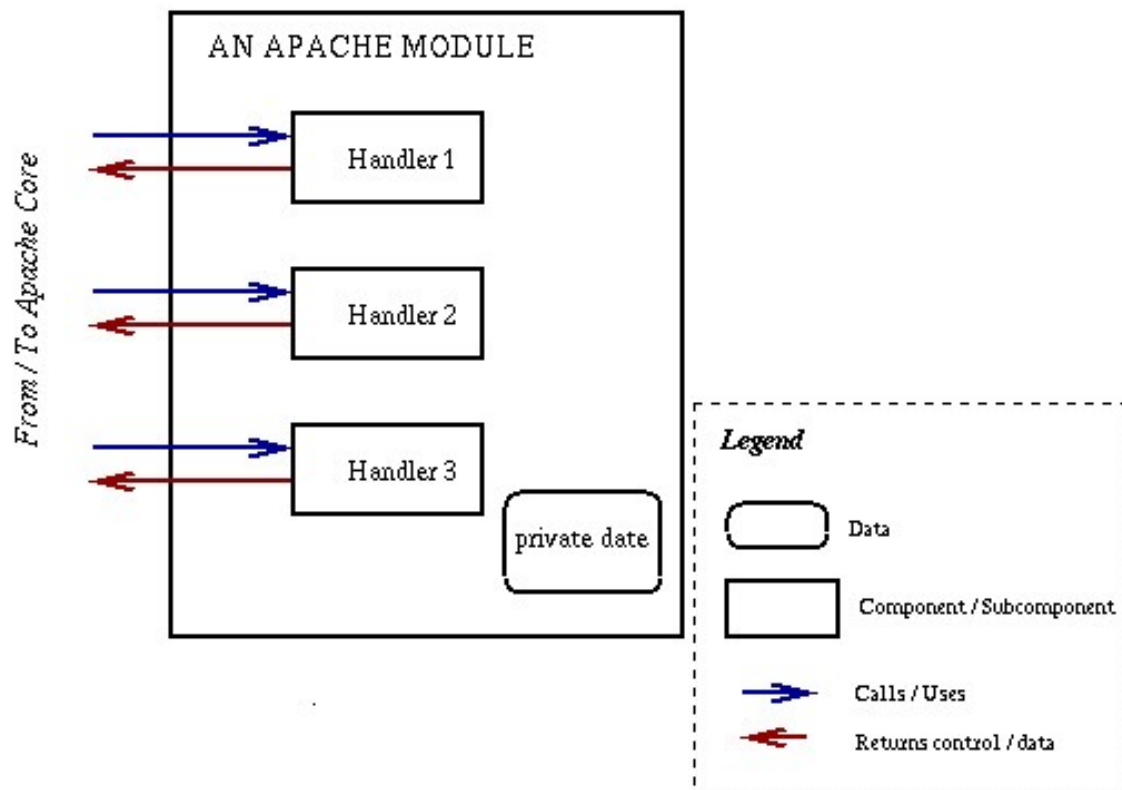


Роля на ядрото



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.

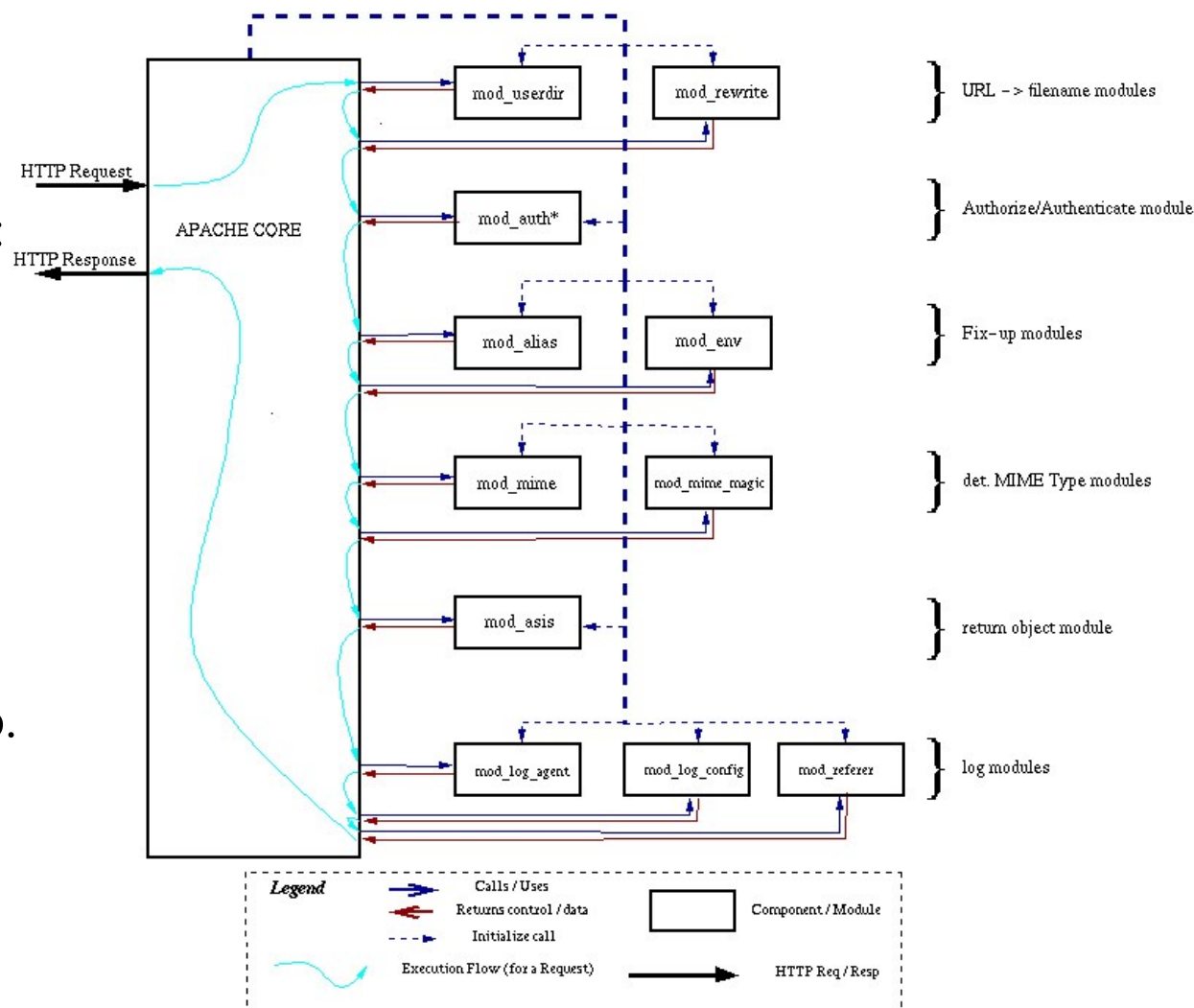
Път на обработка на заявката

- Няма как да се прескочат фазите

За определени фази може да се извика само един модул (хендлър в модул). Такива фази са:

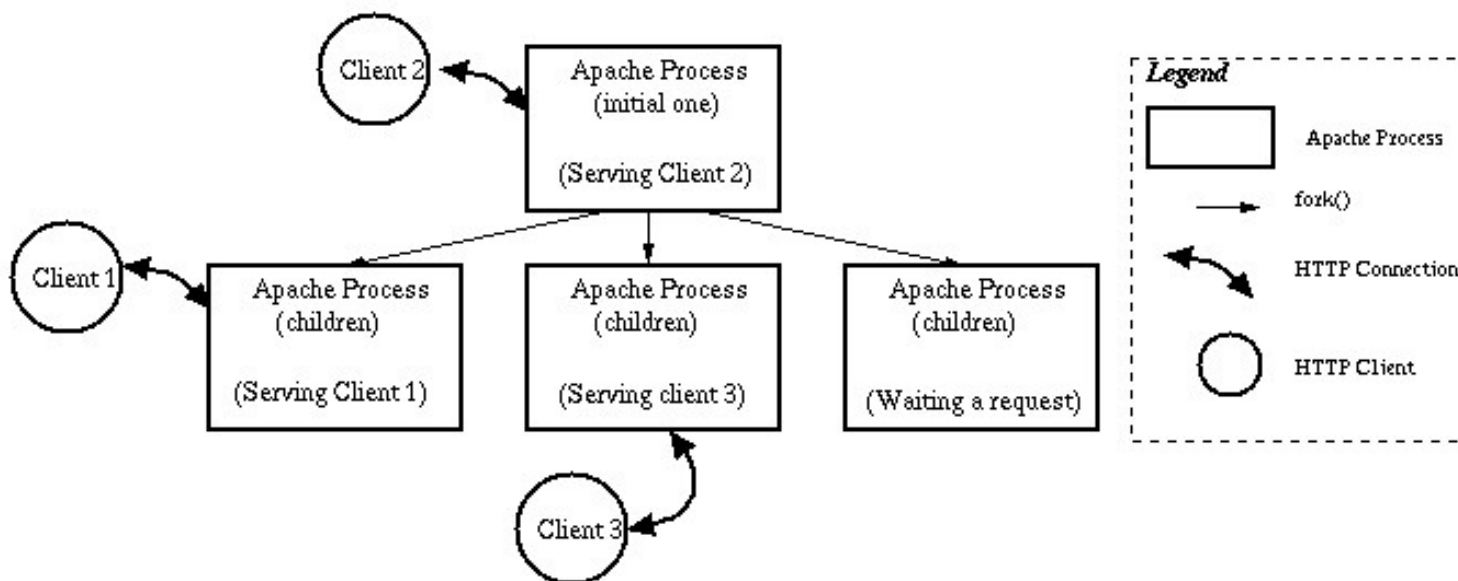
- авторизация,
- автентикация,
- връщане на актуалния обект към клиента
- транслиране на URI в име на файл.

Други фази изискват повече от един хендлър. Когато хендлър върне код за грешка могат да се извикат всички хендлъри.



Паралелност в Apache

- ТСР/ІР 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 файлове, започващи с “.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/html/docs/domain1
```

```
    ServerAlias *.domain1.com
```

```
    ErrorLog /var/log/httpd/domain1.log
```

```
</VirtualHost>
```

```
<VirtualHost *:80>
```

```
    ServerName www.domain2.com
```

```
    DocumentRoot /var/www/html/docs/domain2
```

```
    ServerAlias server.domain2.com
```

```
    ErrorLog /var/log/httpd/domain2.log
```

```
</VirtualHost>
```

Въпроси ?

Благодаря за вниманието !