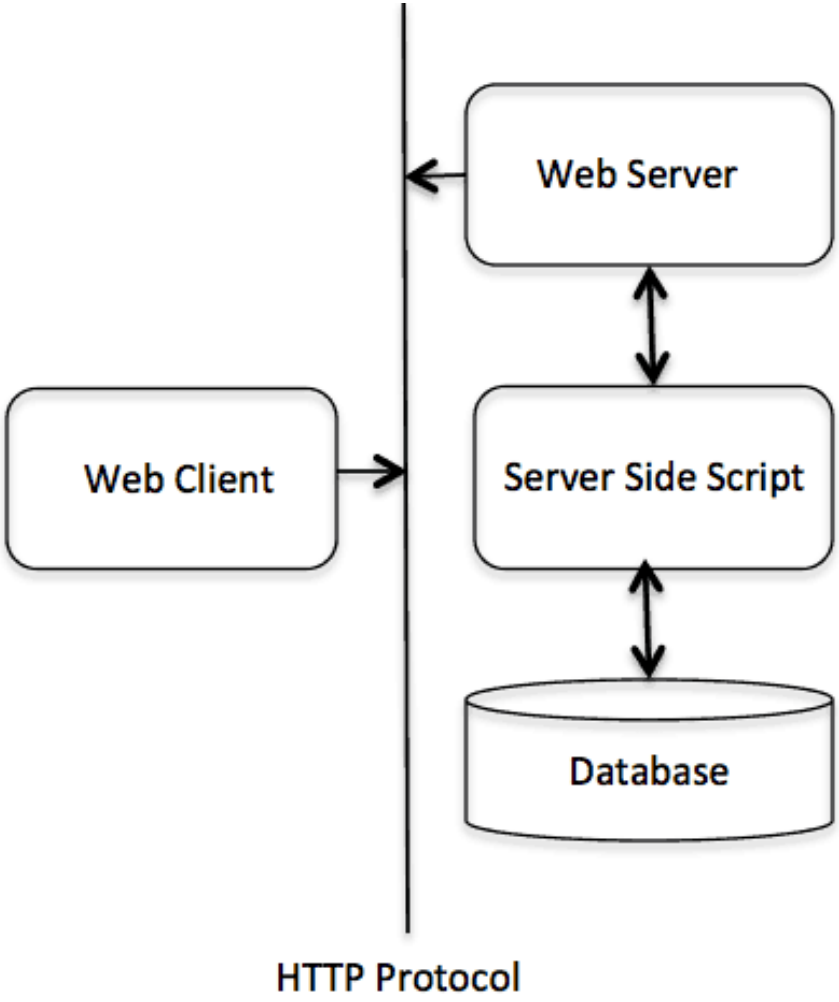
Протокол HTTP

Протокол HTTP

- 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)

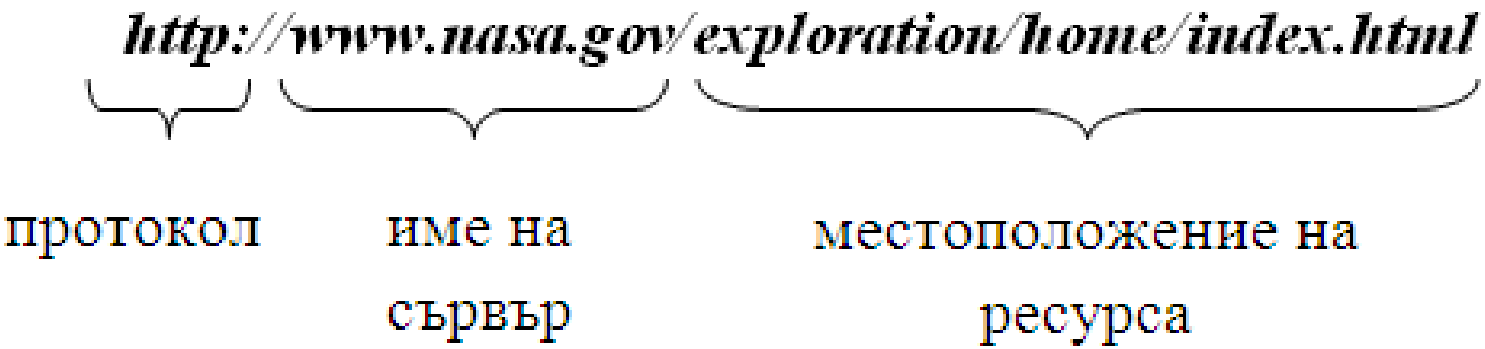
Протокол HTTP

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



URL

- 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:



Първи версии на HTTP

- 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.

Характеристики на HTTP

- 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.) – мултиплексиранена заявки и отговори

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

- Повишават ефективността на HTTP чрез премахване на повечето от разходите за комуникация за HTTP/1.0.
- Стават стандарт за комуникациите между клиентите и сървърите в Web.
- Създаване на постоянна конекция – клиентът е инициаторът на връзката с Web сървъра.

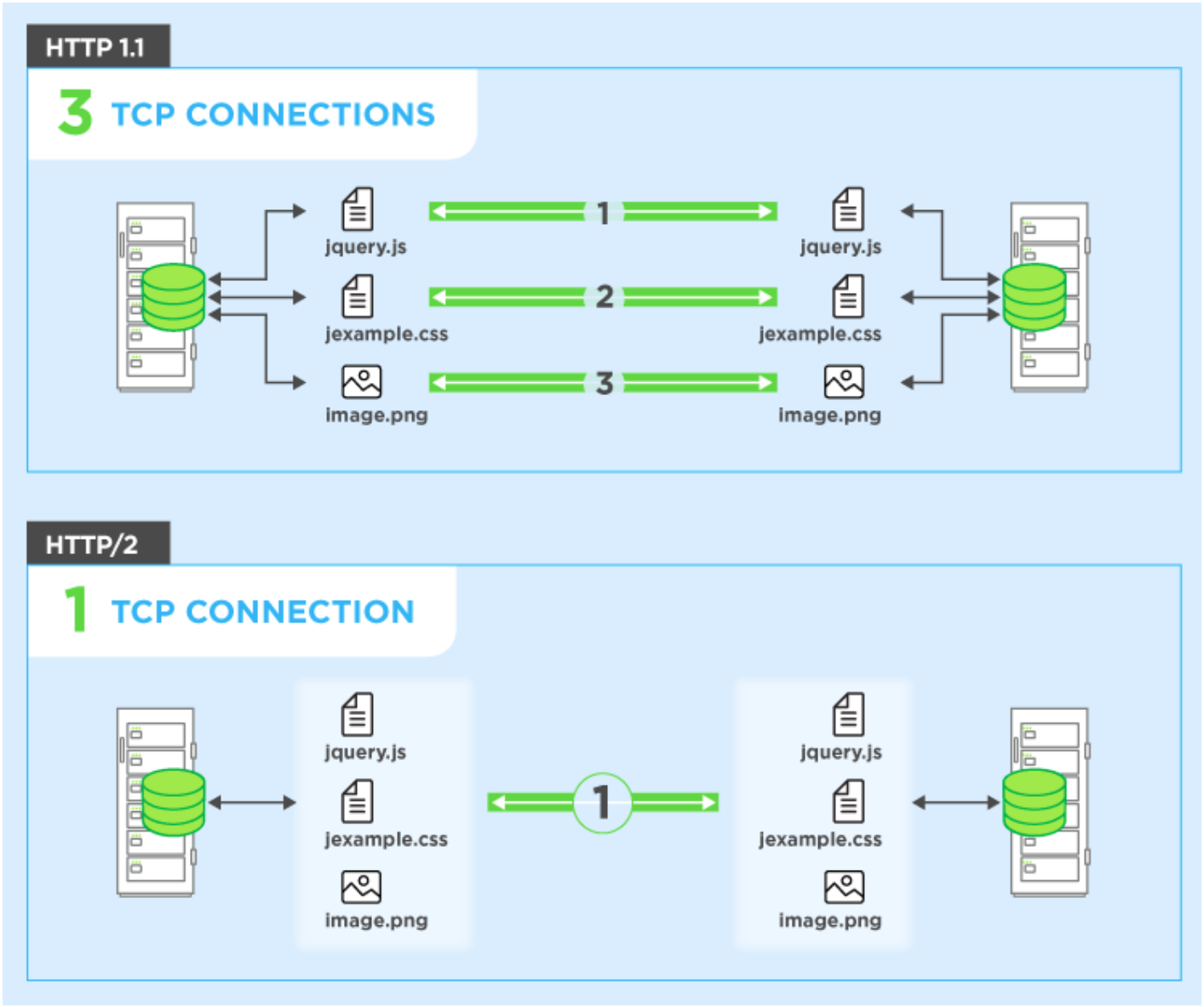
HTTP 2.0

- Бинарен протокол
- 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

HTTP 3.0

- QUIC (Quick UDP Internet Connections) протокол на Google
- Множество мултиплексирани потоци от данни.
- Контролът на потока данни е на приложно ниво
- Оценява пропускателната способност във всяка посока
- Възможност за Forward Error Correction
- През 2018г. получава официалното име HTTP 3.0
- Има експериментални реализации

HTTP 1.1 / 2.0


HTTP 1.1
3 TCP CONNECTIONS
jquery.js
jexample.css
image.png
1
2
3
jquery.js
jexample.css
image.png
HTTP/2
1 TCP CONNECTION
jquery.js
jexample.css
image.png
1
jquery.js
jexample.css
image.png

HTTP 1.1 съобщения

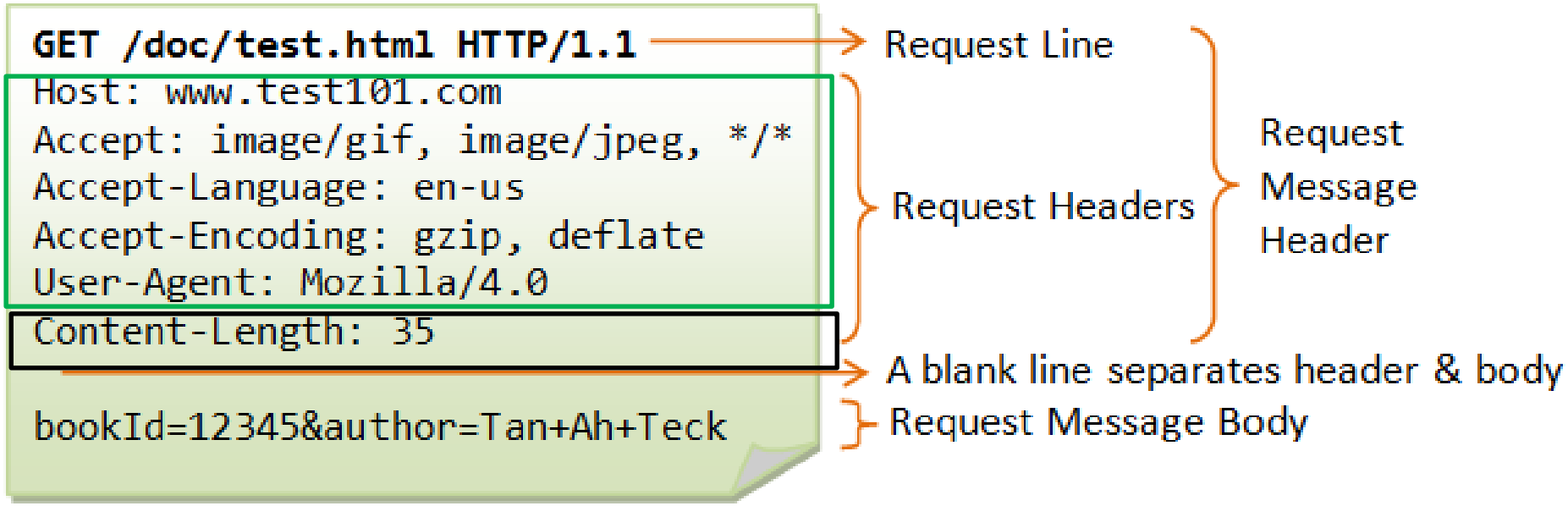
Съобщение-заявка

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

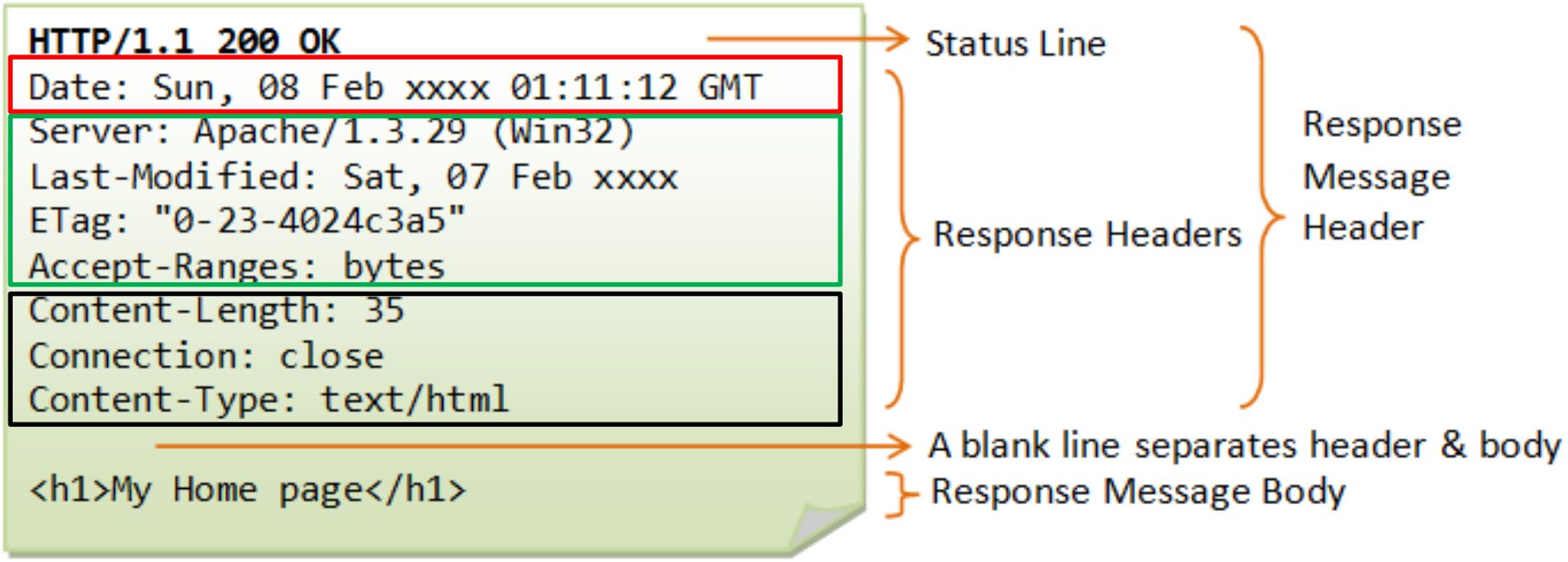
Съобщение-отговор

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

HTTP 1.1 заявка



HTTP 1.1 отговор



HTTP методи

- GET – изисква страница от web сървъра 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
- CONNECT – за превключване между тунели с SSL
- TRACE – да изпрати заявка до дестинацията (сървър или първия прокси), която да отговори на клиента

HTTP статус кодове

- Статус 2xx – Success

(200 – OK, 201 – created…)

- Статус 3xx – Redirection

(300 – Multiple Choices, 305 – Use Proxy)

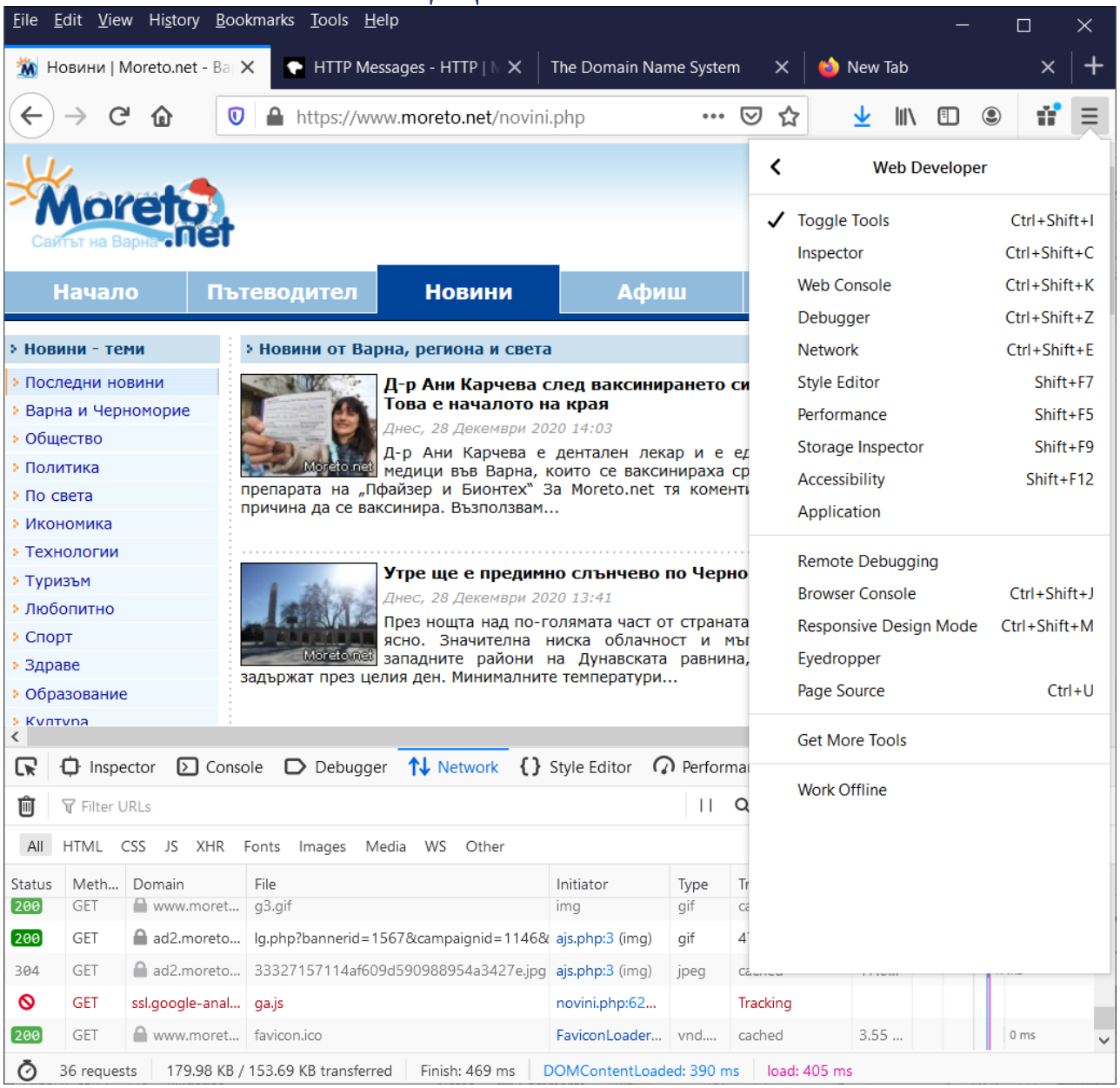
- Статус 4xx – Client Error

(400 – Bad request, 404 – Not Found)

- Статус 5xx – Server Error

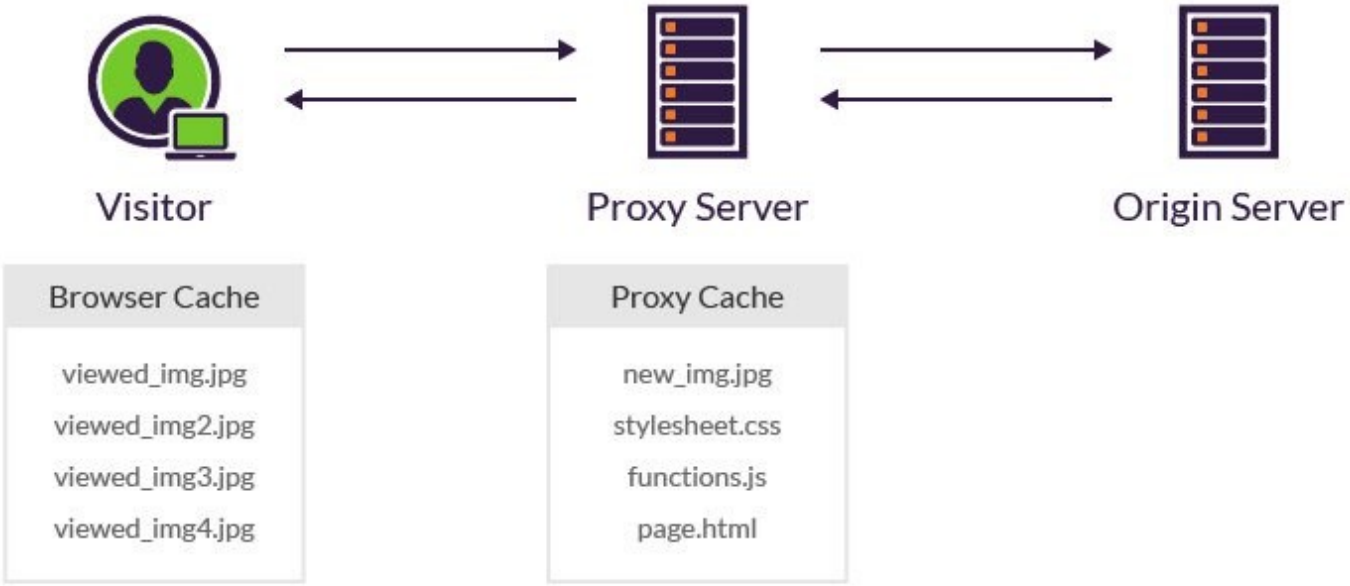
(500 – Internal Error, 501 – Not implemented)

HTTP диагностика



Кеширане

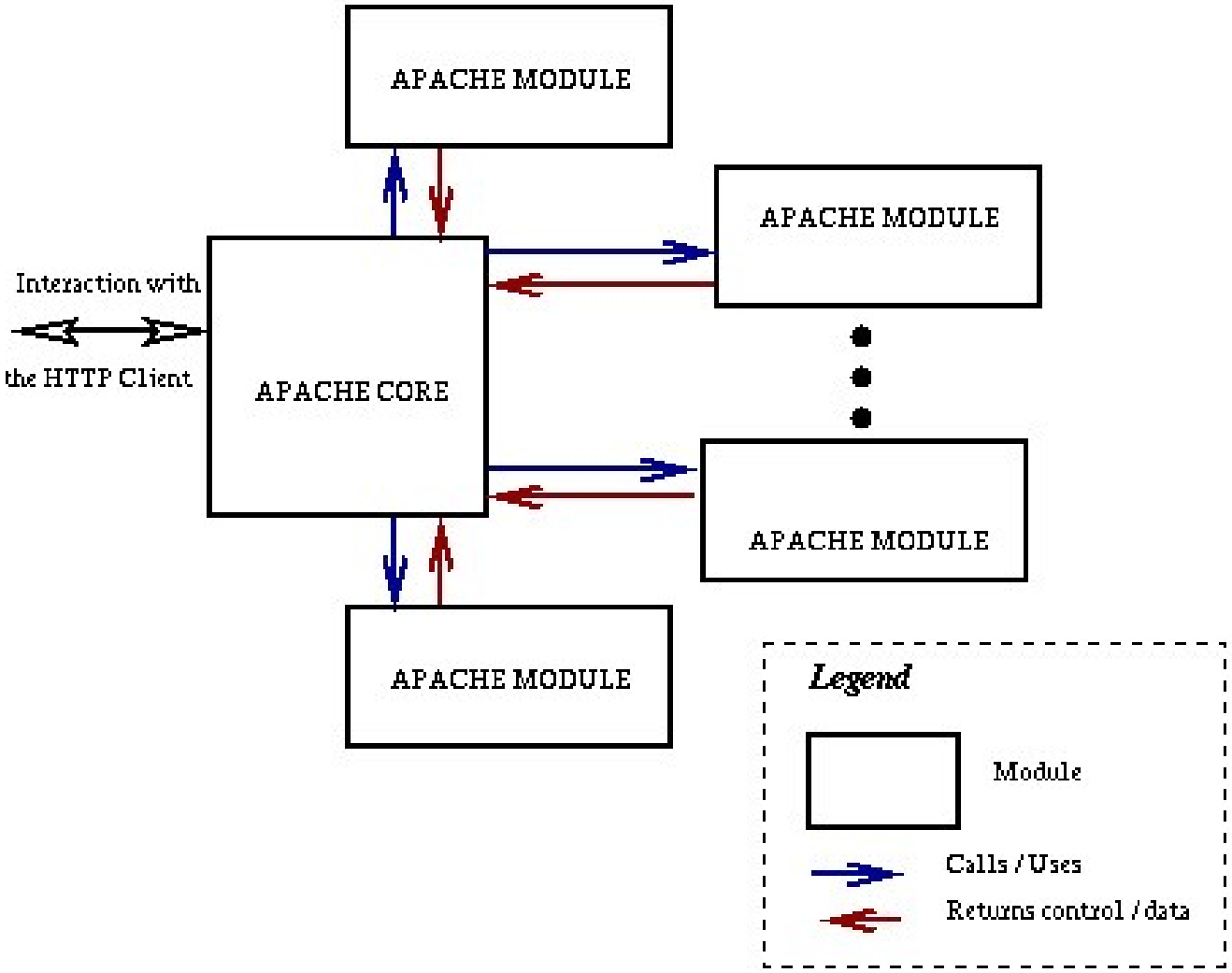
Основната идея на кеширането е, че потребителите имат склонност да изискват едни и същи документи многократно. Поради факта, че някои документи не се променят (или не толкова често), е по-ефективно те просто да се заредят от локалния кеш, отколкото всеки път да се изискват от отдалечения Web сървър.



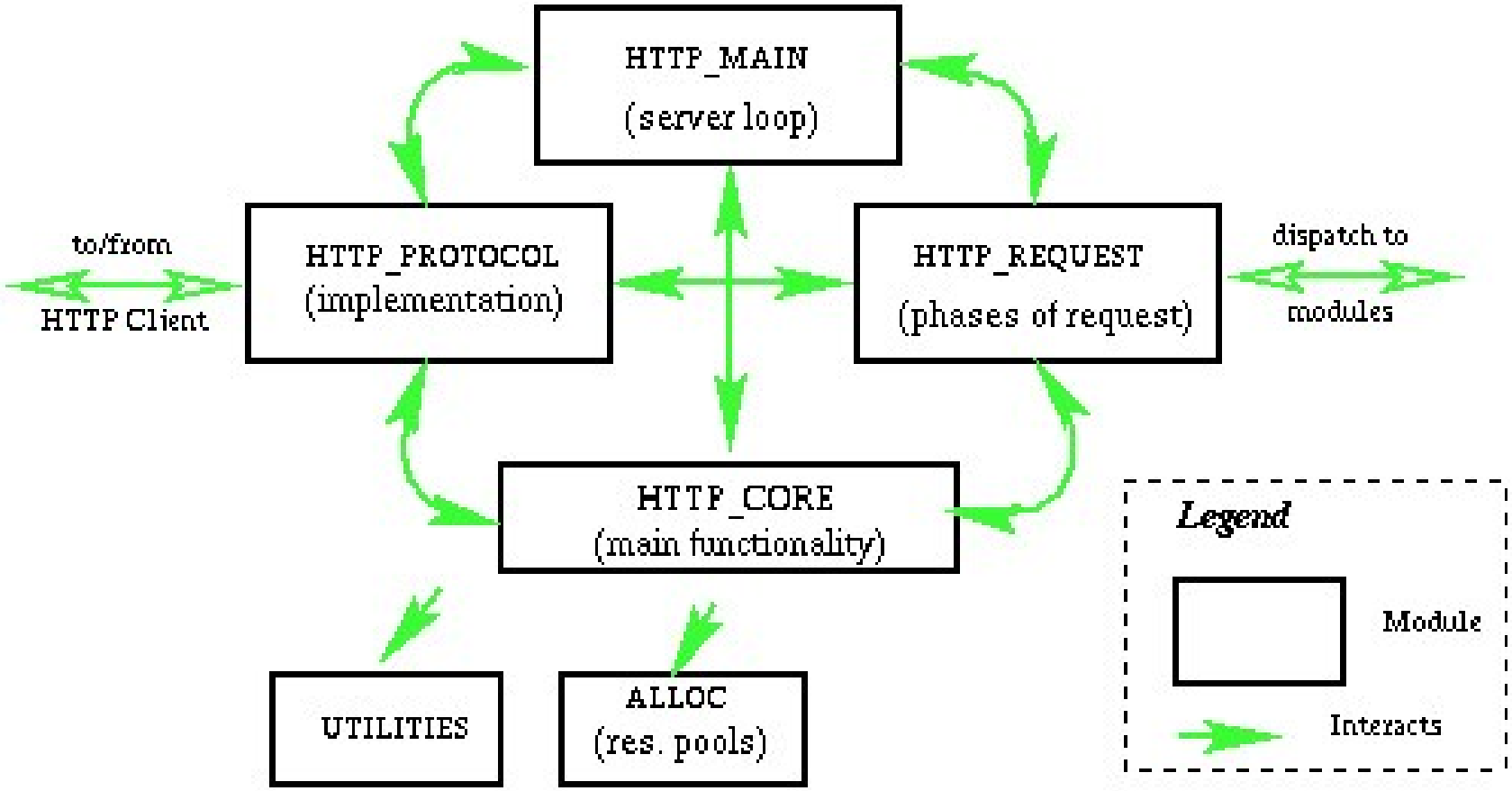
Apache web сървър

- Apache е един от най-популярните Web сървъри.
- 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.

Apache архитектура



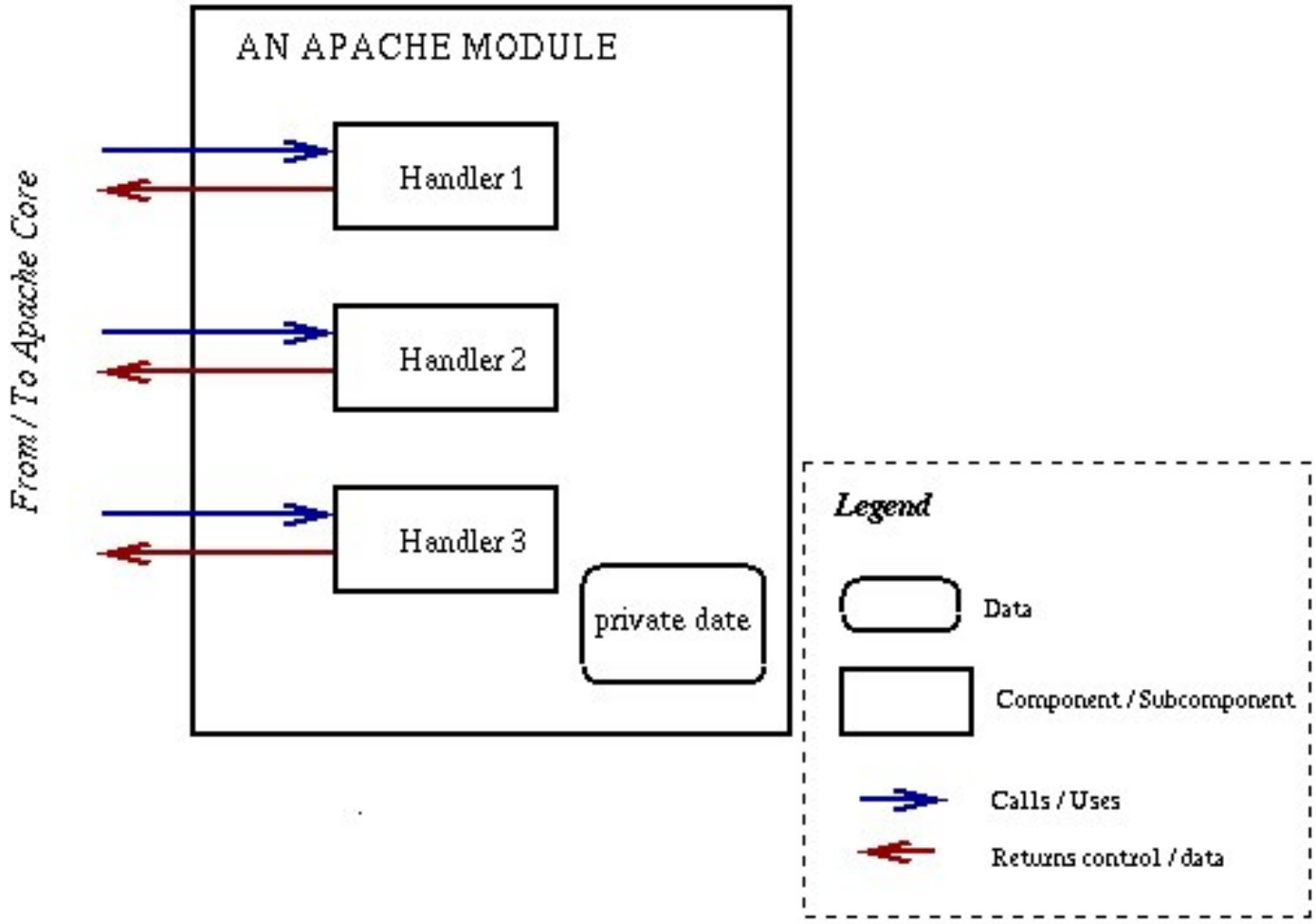
Ядро на Apache



Модули на Apache

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

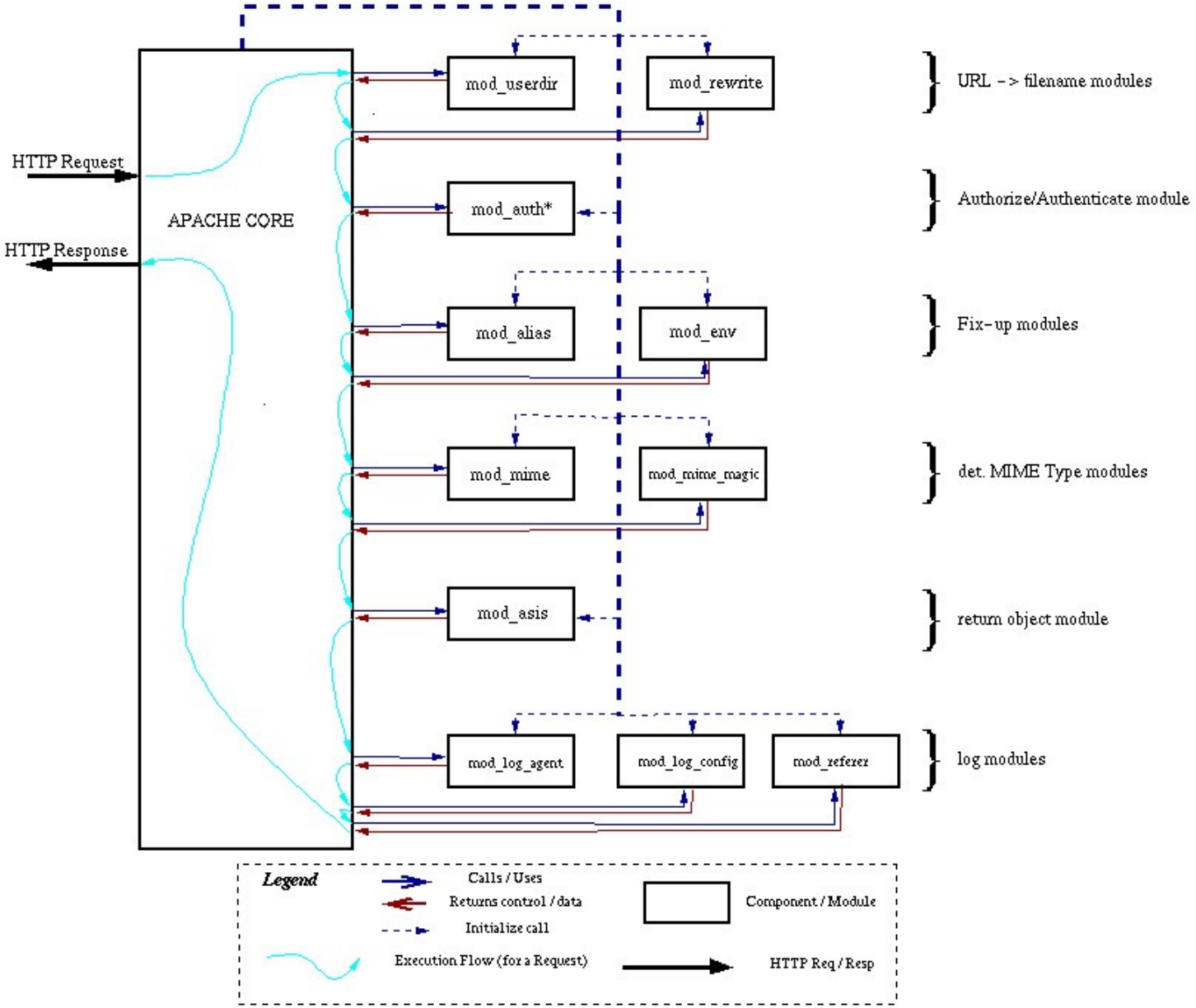
Модули на Apache



Обработка на заявка - фази

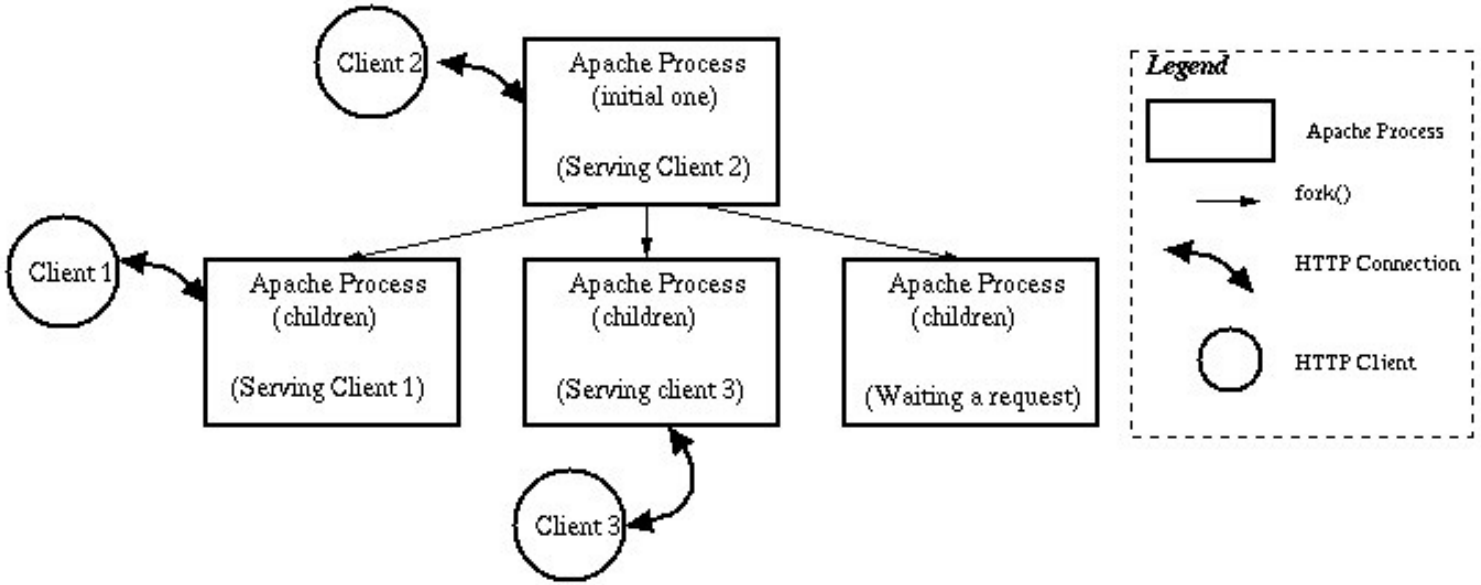
- Преобразуване на URL във файлово име;
- Проверка за достъп, базирана на адреса на хоста или друга достъпна информация;
- Получаване на потребителския ID от HTTP заявката и валидирането му;
- Оторизация на потребителя;
- Определяне на MIME типа на изисквания обект (content type, encoding, language);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

Обработка на заявка



Паралелност в Apache

- Типично TCP/IP сървърите стартират (fork) нов процес, който да обработи идващите от клиентите заявки.
- Apache използва различна техника – *persistent server processes*. Той създава още от началото фиксиран брой дъщерни процеси (под Windows това са нишки).



Конфигуриране на Apache

- За конфигуриране на Apache се използват текстови конфигурационни файлове, съдържащи множество директиви. Основния конфигурационен файл е */etc/httpd/httpd.conf*. Този файл се прочита от сървъра при стартиране или рестарт.
- Последователността на обработване на конф. файлове е
  1. httpd.conf
  2. .htaccess
  3. .htpasswd

Конфигуриране на Apache

- От версия 2 Apache има модулна структура на конфигурационния си файл. Някои от настройките са изнесени в отделни файлове 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 конфигурационен 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 групират по секции (контейнери)
- Контейнерите са 2 типа:
  - Обработват се за всяка заявка. Директивите в тях се прилагат само ако заявката съвпадне с контейнера;
  - Условни. <IfDefine>, <IfModule>, <IfVersion>. Те се обработват само по време на стартиране и рестартиране на сървъра. Ако посочените условия са „истина”, директивите в тях се прилагат.

```
<Directory "/srv/httpd/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>
```

Основна конфигурация

- **ServerRoot “/some_directory”** – кореннова директория, където се търсят файлове с относителни пътища от някои директиви като Include, LoadModule
- **DocumentRoot “/var/www/…”** – от която директория се търсят документите за заявките
- **Listen 80** – порт на който сървърът слуша за заявки. Ако е различен от 80, клиентът трябва да включи този порт в заявката (http://www.same.com:8080).
- **User apache**
- **Group apache** – потребител и група от името на които Apache се изпълнява. Препоръчва се създаване на отделен потребител и група.
- **ServerName www.alabala.com** - текущото регистрирано в DNS име на сървъра
- **DirectoryIndex index.html** – указва списък с ресурси, който ще се върне на клиента ако той е описал в URL „/” в края ако този файл го няма

Контрол на достъпа до ресурси

- Контролът на достъпа е чрез директивите **Allow** или **Deny**.
- **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 *alabala.com*

Allow from *all*

Контрол на достъпа до директории

- От версия 2.0 Apache има рестриктивна политика за достъп към директориите по подразбиране:

<**Directory /> -** указва всички директории под коренновата
Options FollowSymLinks - сървъра ще следва символните линкове в директорията
AllowOverride None – указва, че никои опции няма да бъдат препокрити от конфигурационния файл .htaccess
Order deny,allow - дефинира последователността за достъп – първо забрана
Deny from all – забранява достъп от всякъде
</Directory>

- Трябва допълнително изрично да се разрешава това, което е необходимо

Контрол на достъпа 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

Позволява повече от един web сайт да се изпълняват на същата машина. Два типа виртуални хостове:

- **IP базирани** – при тях се използва IP адреса на конекцията за да се определи коректния ВХ. Трябва да има отделен IP за всеки хост.
- **базирани на име** – клиентът трябва да включи името на хоста като част от HTTP хедъра. Така различни хостове могат да споделят едни и същи IP. Този тип е по-използван.

Виртуални хостове - конфигуриране

Виртуалните хостове се конфигурират във файла */etc/httpd/extra/httpd-vhosts.conf*.

Той трябва да се включи в основния конфигурационен файл с директивата **include**.

За да се използват именувани ВХ, трябва да се определи IP адреса на хоста и порта. Това става с директивата **NameVirtualHost *:80**
Ако (нормално) се използва само един IP този аргумент (*) е достатъчен.

Виртуални хостове - пример

```
NameVirtualHost *:80

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

    <VirtualHost *:80>
            ServerName www.otherdomain.com
            DocumentRoot /www/otherdomain
    </VirtualHost>
```

Тестване на виртуалните хостове:

 # httpd -S

Въпроси ?