

24. **Хипертекстов протокол в Интернет**

Основното е хипертекста

Най-мощното приложение на Internet е **WWW** (world wide web).

Като технология и идея е създадено от физика **Тим Бърнър Лий** през **1990** година с цел да улесни комуникацията на група учени.

web-технологията се базира на **хипертекста**.

Хипертекст

Това е текст, който съдържа в себе си информация как да бъде изобразен на екрана.

Изобразяването става чрез специална програма, наречена хипертекстов `browser`.

Хипертекст, URL, хиперлинк

Хипертекстът се оформя като **съвкупност** от **страници**.

Всяка страница си има уникално **URL** – уникален адрес, който еднозначно указва **местоположението на страницата в Internet**.

Хиперлинк – под част от текста стои URL на друга страница.

Т.е хипертекстовите страници съдържат **препратки** към други хипертекстови страници.

Не се изисква тези страници да се намират на същия сървър.

HTTP - за комуникация между сървър и клиент

Web - паяжина – страниците, които са пръснати и са свързани като паяжина чрез хиперлинковите връзки.

Browser е клиентът, който изтегля и изобразява страниците.

Web server е сървърът, който съхранява страниците.

За комуникация между browser-и и сървъри е създаден протокола **HTTP** (hyper text transfer protocol).

URL

URL съдържа протокол, име на домейн и пътя на страницата върху диска на сървъра.

При осъществяване на връзка между browser-а и сървъра по URL-то първо се търси по името на сървъра, а после по пътя върху диска на сървъра.

Страницата физически се изтегля от сървъра, предава се на browser-а и той я изобразява.

Една страница се прехвърля в рамките на една HTTP-сесия.

URL (пример)

foo://example.com:8042/over/there/index.dtb?name=ferret#nose

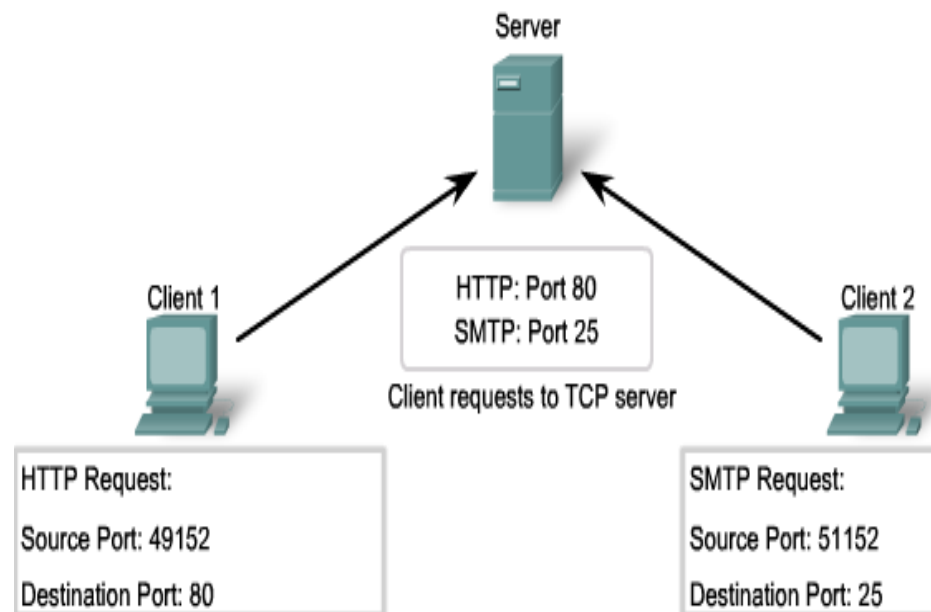
\ / _____/ _/_____/_____/ _____/ _/
scheme authority port path filename query fragment

HTTP – TCP/80

Протоколът HTTP
се базира на TCP.

HTTP клиентът
отваря сесия на
произволен порт
с номер по-голям
от 1024.

HTTP сървърът
слуша за заявки
на порт 80.



HTTP - request-response protocol

HTTP – request-response (заявка-отговор) протокол.

Клиентът отправя съобщение с HTTP заявка към сървъра.

Сървърът (съдържание, ресурси - HTML файлове и др.) връща съобщение с отговор:

- информация за състоянието - хедъра и заявеното съдържание – в тялото.

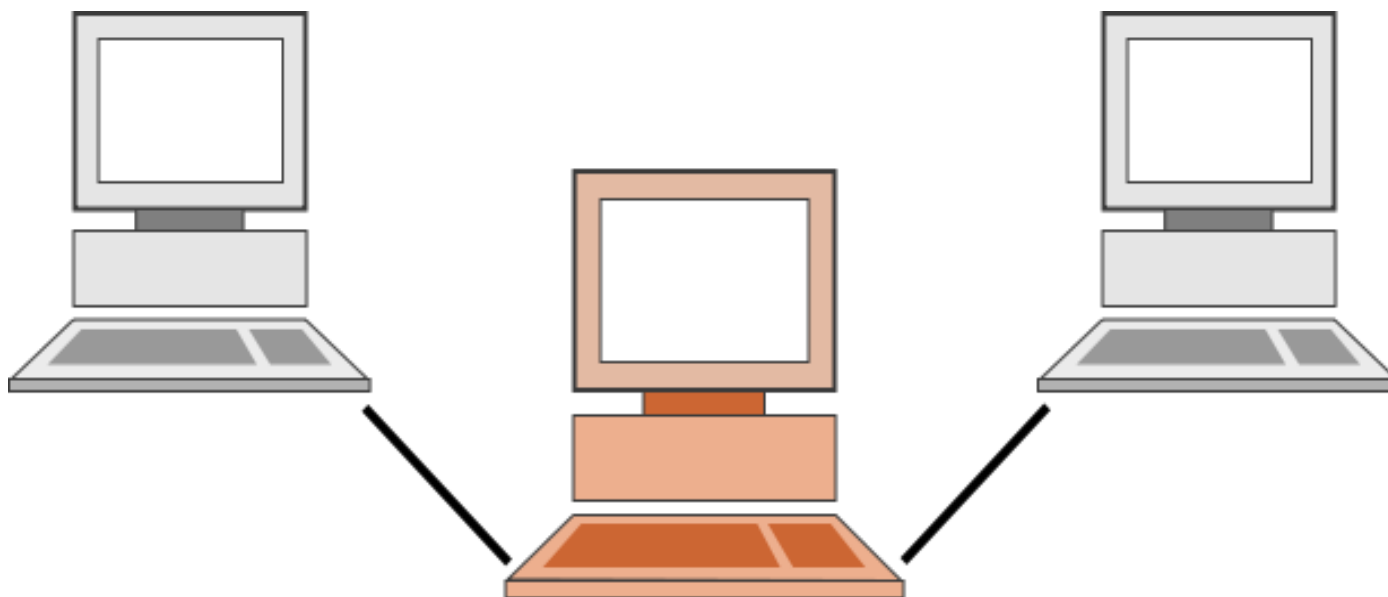
Посредници

HTTP допуска "посредници" с цел повишаване на производителността:

- **web cache** сървъри – предоставят съдържание за сметка на "оригиналния";
- **proxy** сървъри – прави заявки за сметка на клиента.

HTTP е **stateless** протокол. Сървърът не задържа информация или състояние за всеки отделен потребител през цялото време на изпълнение на всяка заявка.

Proxy сървър



Версия 1.0 на HTTP

При първите версии на HTTP протокола за изпълнението на всеки метод се прави отделна HTTP сесия.

Тя отваря TCP съединение, праща нещо, след това затваря последователно съединението и сесията.

Имаме една сесия, едно съединение.

Не е ефективно

Не е ефективно. Много служебен трафик.

Ако за всяко GET се затваря съединението, а предстои **четене на серия от страници**.

Правенето на всяко ново съединение:

- **resolving** на URL-адреса;
- на TCP ниво **3-way handshake**;
- договаряне на HTTP сесия.

Колко време да е отворена сесията

Не може всяко обръщение към дадена
страница да отваря сесия към даден сървър
и да я държи

$$\Delta t \approx \infty$$

Нормално би било сесията да приключи с
изтеглянето на страницата.

Версия 1.1 на HTTP

Request -1	Response- 1	Request-2	Respose-2	Timeout
------------	-------------	-----------	-----------	---------

При версия 1.1 на HTTP тези недостатъци са преодолени:

- на едно TCP съединение отговарят няколко сесии (няколко команди).
- различни хипертекстови страници, но с едно TCP съединение.

Създава се „канал“ в пълен дуплекс: в едната посока - заявки, а в обратната – отговори.

HTTP сесия

HTTP сесията е последователност от **request-response** транзакции.

HTTP клиентът инициира заявка. Установява съединение на **порт 80/TCP**.

HTTP сървърът “слуша” на **порт 80** за съобщение със заявка от клиент.

При получаване на заявка сървърът връща **"HTTP/1.1 200 OK"** и **съобщение**, което съдържа:

- търсения **ресурс** или
- съобщение за **грешка**.

Съобщение Request

request съобщението се състои:

- * ред Request, напр. `GET /images/logo.png HTTP/1.1`, т.е заявява от сървъра ресурса `/images/logo.png`
- * хедъри, напр. `Accept-Language: en`
- * празен ред
- * тяло на съобщението (евентуално)

Редовете да завършват с `<CR><LF>`

HTTP методи

HTTP дефинира **девет метода** (наричани и "verbs" - глаголи), които показват какво действие да се изпълни върху желания ресурс.

Ресурсът е файл или данни, получени от изхода на програма на сървъра.

HTTP request чре3 telnet

```
josh@blackbox: ~  
File Edit View Terminal Tabs Help  
josh@blackbox:~$ telnet en.wikipedia.org 80  
Trying 208.80.152.2...  
Connected to rr.pmtpa.wikimedia.org.  
Escape character is '^]'.  
GET /wiki/Main_Page http/1.1  
Host: en.wikipedia.org  
  
HTTP/1.0 200 OK  
Date: Thu, 03 Jul 2008 11:12:06 GMT  
Server: Apache  
X-Powered-By: PHP/5.2.5  
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate  
Content-Language: en  
Vary: Accept-Encoding, Cookie  
X-Vary-Options: Accept-Encoding;list-contains=gzip, Cookie;string-contains=enwikiToken;string-contains=enwikiLoggedOut;string-contains=enwiki_session;  
string-contains=centralauth Token;string-contains=centralauth_Session;string-contains=centralauth_LoggedOut  
Last-Modified: Thu, 03 Jul 2008 10:44:34 GMT  
Content-Length: 54218  
Content-Type: text/html; charset=utf-8  
X-Cache: HIT from sq39.wikimedia.org  
X-Cache-Lookup: HIT from sq39.wikimedia.org:3128  
Age: 3  
X-Cache: HIT from sq38.wikimedia.org  
X-Cache-Lookup: HIT from sq38.wikimedia.org:80  
Via: 1.0 sq39.wikimedia.org:3128 (squid/2.6.STABLE18), 1.0 sq38.wikimedia.org:80 (squid/2.6.STABLE18)  
Connection: close  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
    <meta name="keywords" content="Main Page,1778,1844,1863,1938,1980 Summer Olympics,2008,2008 Guizhou riot,2008 Jerusal  
...  
... This content has been removed to save space  
...  
"Non-profit organization">nonprofit</a> <a href="http://en.wikipedia.org/wiki/Charitable_organization" title="Charitable organization">charity</a>.<b  
r /></li>  
    <li id="privacy"><a href="http://wikimediafoundation.org/wiki/Privacy_policy" title="wikimedia:Privacy policy">Privac  
y policy</a></li>  
    <li id="about"><a href="/wiki/Wikipedia:About" title="Wikipedia:About">About Wikipedia</a></li>  
    <li id="disclaimer"><a href="/wiki/Wikipedia:General_disclaimer" title="Wikipedia:General disclaimer">Disclaimers</a>  
</li>  
  </ul>  
</div>  
</div>  
  
  <script type="text/javascript">if (window.runOnloadHook) runOnloadHook();</script>  
<!-- Served by srv93 in 0.050 secs. --></body></html>  
Connection closed by foreign host.  
josh@blackbox:~$
```

Основен метод GET

Основният метод е **GET**.

Като аргумент му се подава адреса на страницата върху диска на сървъра.

Страниците могат да се кешират върху прохусървъри, затова в метода GET има if-условие.

Метод HEAD

Всяка хипертекстова страница има **заглавие**:

- възрастта на страницата;
- къде се намира;
- датата на последната модификация и др.

Методът **HEAD**, за разлика от GET, взима само заглавието на страницата.

Полезен е за извличане на мета-информация.

Методи PUT и POST

Методът **PUT** служи за прехвърляне на страница върху сървъра:

- обмен на два етапа – първо се дава адреса на страницата, след това се прехвърля самата страница.

Методът **POST** $\approx$ PUT, но той добавя новите данни към съществуващ адрес:

- създава се нов ресурс или се обновява съществуващ.

Методът TRACE

Методът **TRACE** връща от сървъра получените данни по заявката.

Той се използва **за тестване** – да видим дали сървърът е получил това, което сме изпратили.

Методът **DELETE** изтрива заявения ресурс.

OPTIONS проверява функционалността (методите) на web сървъра за дадено URL:

- **иска '*'** вместо конкретен ресурс.

Методи CONNECT и PATCH

CONNECT конвертира request съединението в прозрачен TCP/IP тунел, за да минат SSL-криптирани комуникации (**HTTPS**) през некриптирано HTTP прокси.

PATCH – за частични модификации върху ресурс.

HTTP трябва да реализират **поне GET и HEAD**, когато е възможно - **и OPTIONS**.

HTTP Secure

Hypertext Transfer Protocol Secure (**HTTPS**) е:

HTTP с SSL/TLS

за криптирани комуникации и сигурна идентификация на web сървър.

HTTPS се използва за **е-разплащания** и други поверителни операции

(напр. СУСИ:

<https://susi4.uni-sofia.bg/>)

S-HTTP

HTTPS да не се бърка със Secure HTTP (**S-HTTP**) - RFC 2660.

(S-HTTP криптира само данните в страницата, POST полета. Не засяга инициране на съединението. S-HTTP може да споделя един порт с HTTP. Некриптираният хедър определя дали съдържанието е криптирано или не.)

TLS / SSL

Transport Layer Security (**TLS**) и предшественика му Secure Sockets Layer (**SSL**) са криптографски протоколи.

TLS и SSL криптират сегментите, които се пренасят през сесията на транспортния протокол.

TLS се базира на SSL

TLS е IETF стандарт (последно [RFC 5246](#)) и се базира на по-ранни SSL версии, разработени от Netscape Corporation:

- TLS 1.0 (SSL 3.1)

- TLS 1.1 (SSL 3.2)

- TLS 1.2 (SSL 3.3)

HTTPS URLs започват с "[https://](#)" и използват [порт 443](#) по подразбиране.

TLS сесия

TLS клиент и сървър изпълняват “ръкостискането”, параметри за установяване на сигурно съединение.

- * Клиентът се свързва към сървър с TLS възможности, заявявайки сигурна конекция, представя списък с поддържани CipherSuites (шифри и хеш функции).

- * От списъка сървърът избира най-силните, които поддържа и уведомява клиента за решението си.

- * Сървърът връща идентификатора си под формата на цифров сертификат, съдържащ име, доверената “трета страна” - certificate authority (CA) и публичния си криптиращ ключ.

TLS сесия

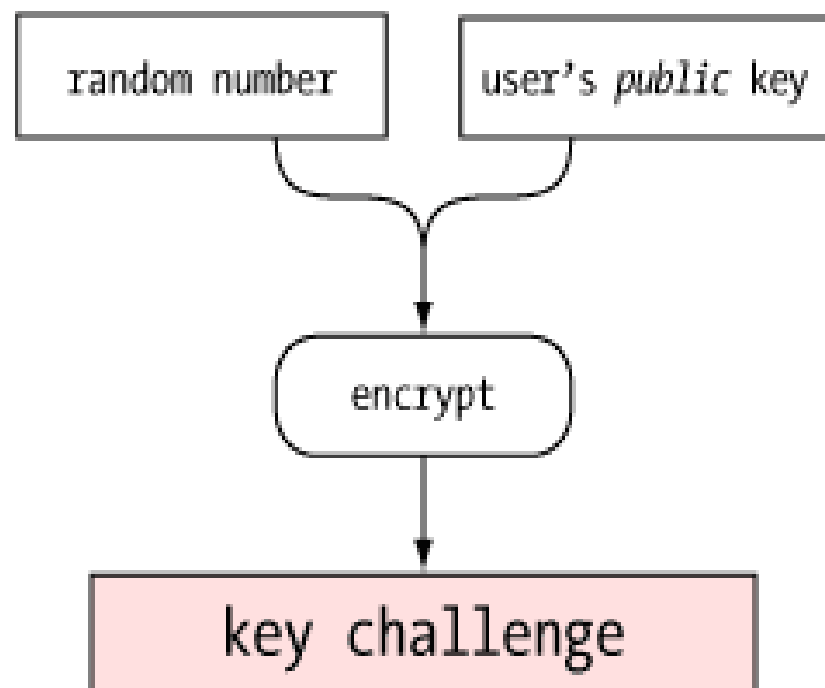
- * Клиентът може да се обърне към СА, за да потвърди сертификата.

- * За да генерира ключове за сесията, клиентът криптира случайно число с публичния ключ на сървъра и изпраща получения резултат. Само сървърът със своя частен ключ ще може да го декриптира.

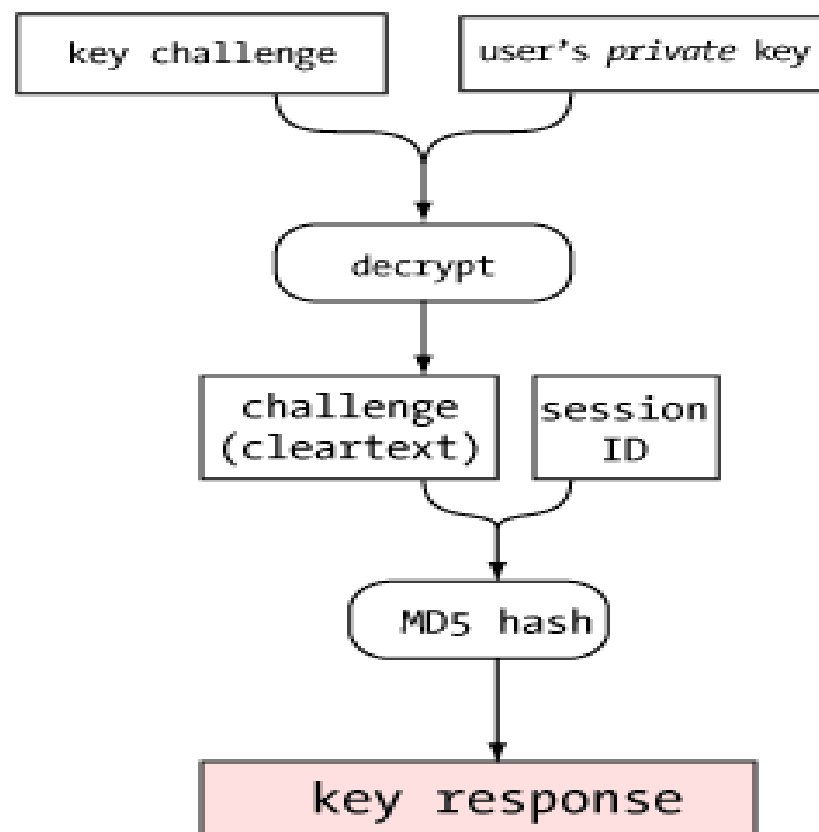
С това приключва “ръкостискането” и започва защитената сесия.

Публичен и частен ключ

Key Challenge Creation



Key Response Generation



HTTPS сесия

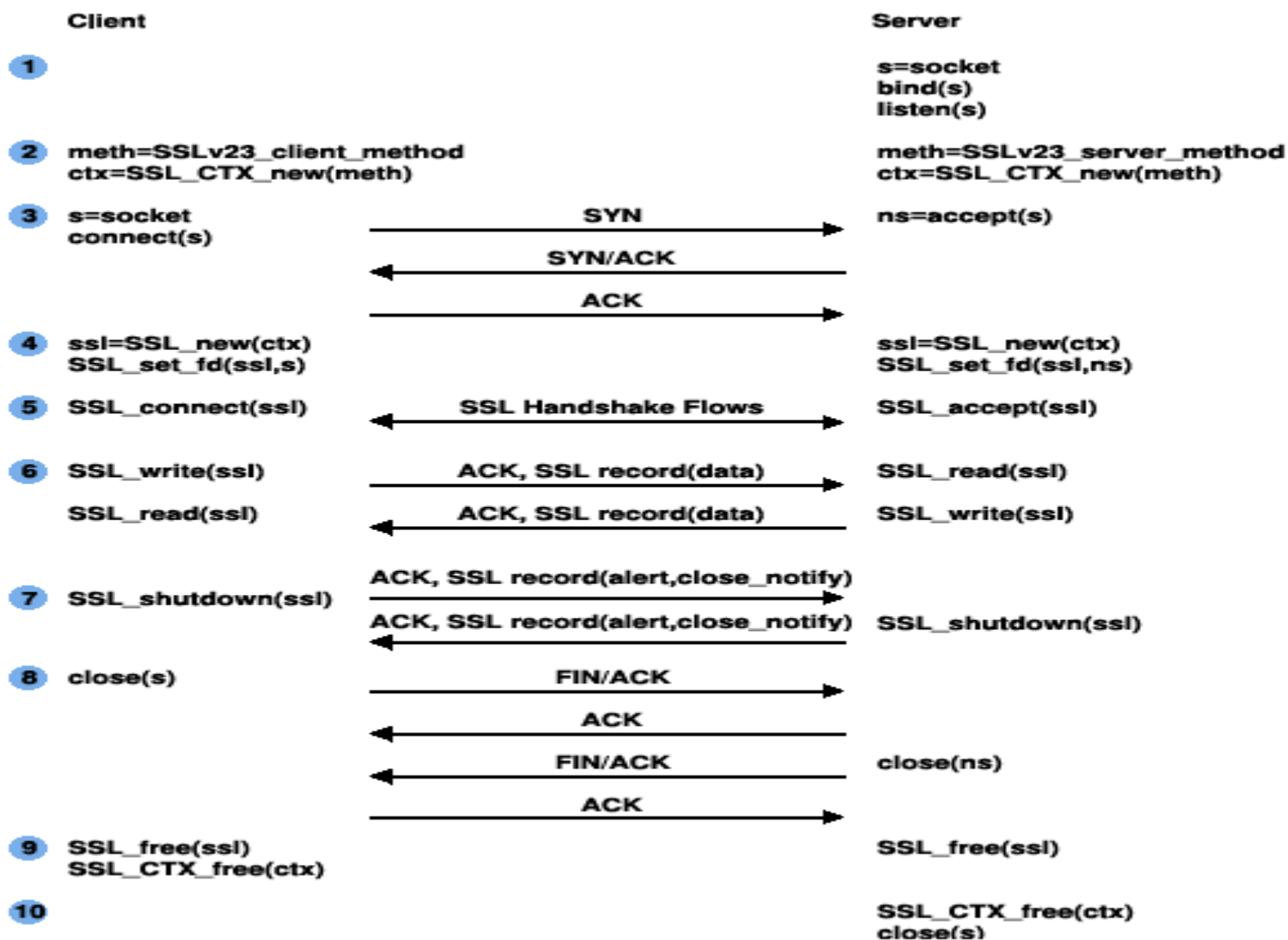
Можем да се доверим на HTTPS съединение, ако са изпълнени всички изброени по-долу условия:

1. Потребителят е сигурен, че неговия браузър правилно реализира HTTPS с правилно инсталирани предварително CAs.
2. Потребителят се доверява на CAs, гарантиращи легитимността на съответни web сайтове.
3. web сайтът има валиден сертификат, т.е подписан от доверена трета страна.

HTTPS Сесия

4. Сертификатът коректно идентифицира web сайта.
5. Или междинните хопове по Internet са доверени, или потребителят се доверява криптиращия слой (TLS или SSL), че не е податлив на “прослушване” или “разбиване”.

SSL Session Flow



SSL session flow

- Стъпки 1, 3, и 8 - TCP без SSL.
- Стъпки 2, 4, 5, 7, и 9 - само със SSL.
- Стъпки 6 и 10 – зависи дали се използва или не SSL.

Да се доверим?



Secure Connection Failed

svn.boost.org uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is unknown.

(Error code: sec_error_unknown_issuer)

- This could be a problem with the server's configuration, or it could be someone trying to impersonate the server.
- If you have connected to this server successfully in the past, the error may be temporary, and you can try again later.

[Or you can add an exception...](#)

Виртуални сървъри

Зад един IP адрес може да има няколко имена на сървъри (т.нар. **виртуални сървъри**).

За клиента те са различни сървъри, но са на една и съща машина.

Възможно е също на една машина:

- **>1** виртуални сървъри — **всеки с $\neq$ IP и име.**

HTTP сървъри...

В момента се използват два HTTP сървъра:

`apache` - в UNIX/Linux и

`IIS` на Microsoft.

Apache

Apache HTTP Server Project е **open-source** решение за UNIX/Linux, може и Windows.

Apache **httpd** е най-популярният web сървър в Internet от **Април 1996**.

Apache Version 2.4.2

Binding (обвързване) с IP адреси и портове:

След стартиране `httpd` се обвързва (**слуша**) с порт и адрес локалната машина и чака заявки.

По подразбиране слуша на всички адреси и на порт 80. Но може да му се зададат определени адреси и портове.

В комбинация с **Virtual Host**: как `httpd` да реагира на различни IP адреси, имена и портове.

Директивата Listen

Директивата **Listen** казва на сървъра да приема входящи заявки само на определен(и) порт(ове) или комбинации **адрес-порт**:

- ако е зададен **само portNo.**, сървърът слуша на дадения порт и на **всички интерфейси**;
- ако е зададен **IP и portNo.**, сървърът слуша на дадения **порт и интерфейс**.

Файлът `httpd.conf` може да съдържа **>1 директива Listen**. Сървърът ще отговаря на заявки от всеки от зададените адреси и портове.

Apache v. 2.3. Binding. Примери.

Искаме сървърът да приема конекции и на порт 80, и на порт 8000, на всички интерфейси:

```
Listen 80
```

```
Listen 8000
```

Сървърът да приема конекции на порт 80 за един интерфейс, и порт 8000 за друг:

```
Listen 192.0.2.1:80
```

```
Listen 192.0.2.5:8000
```

Слуша по IPv6

IPv6 адресите трябва да бъдат заградени в квадратни скоби:

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

Определяне на протокол с Listen

По подразбиране https е на порт 443, а http – на всички останали портове.

Протокол се дефинира само ако работи на нестандартни портове. Напр., **https – на порт 8443:**

```
Listen 192.170.2.1:8443 https
```

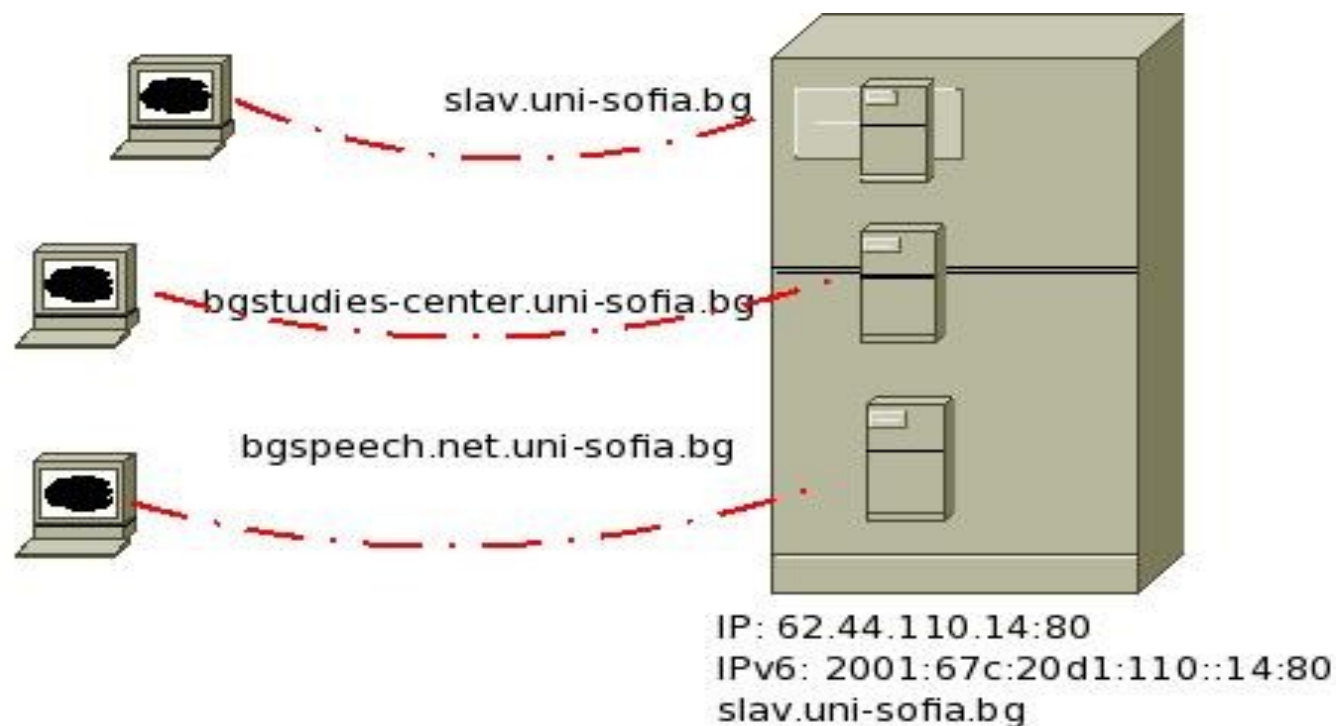
Конфигуриране

Apache HTTP Server се конфигурира, като директивите се поставят в обикновени текстови файлове.

Главният конфигурационен файл е **httpd.conf**.

В следващия пример с директивата **include** в **httpd.conf** се казва кои са **конф. файлове** на виртуалните хостове. (IP адресът е един и същ.)

Виртуални сървъри (пример)



Пример на Virtual Host

```
less /etc/httpd/conf/httpd.conf
```

```
...
```

```
Include "/etc/httpd/sites/*.conf"
```

```
less /etc/httpd/sites/62.44.110.14_80_slav.uni-sofia.bg.conf
```

```
## Default Virtual Host Configuration
```

```
NameVirtualHost 62.44.110.14:80
```

```
NameVirtualHost 2001:67c:20d1:110::14:80
```

Пример (прод.)

```
<VirtualHost 62.44.110.14:80>
```

```
    Include /etc/httpd/sites/slav.uni-  
sofia.bg.httpd
```

```
</VirtualHost>
```

```
<VirtualHost 2001:67c:20d1:110::14:80>
```

```
    Include /etc/httpd/sites/slav.uni-  
sofia.bg.httpd
```

```
</VirtualHost>
```

Пример (прод.)

```
<VirtualHost 62.44.110.14:80>
```

```
    Include /etc/httpd/sites/bgstudies-  
center.uni-sofia.bg.httpd
```

```
</VirtualHost>
```

```
<VirtualHost 2001:67c:20d1:110::14:80>
```

```
    Include /etc/httpd/sites/bgstudies-  
center.uni-sofia.bg.httpd
```

```
</VirtualHost>
```

Пример (прод.)

```
<VirtualHost 62.44.110.14:80>
```

```
    Include
```

```
/etc/httpd/sites/bgspeech.net.httpd
```

```
</VirtualHost>
```

```
<VirtualHost 2001:67c:20d1:110::14:80>
```

```
    Include
```

```
/etc/httpd/sites/bgspeech.net.httpd
```

```
</VirtualHost>
```

Пример (прод.)

```
less /etc/httpd/sites/virtual_host_global.conf*
```

```
...
```

```
Listen 62.44.110.14:80
```

```
Listen [2001:67c:20d1:110::14]:80
```

```
NameVirtualHost 62.44.110.14:80
```

```
NameVirtualHost [2001:67c:20d1:110::14]:80
```

*Автоматично генериран

Браузъри

По-известни HTTP браузъри по ред на поява:

Mosaic – 1993 г.;

Netscape Navigator и Netscape Communicator, 1994

Internet Explorer 1, 1995

Opera, 1996

Mozilla Navigator, 2002

Safari (за Apple Macintosh), 2003

Mozilla Firefox, 2004

Google Chrome, 2008