

Въведение

TCP/IP, DNS & домейни, WWW, HTTP, браузъри, инструменти за разработка.

Владислав Илиев / SAP Labs Bulgaria

10-ти Октомври 2016

Public



Съдържание



История



Мрежови основи



Моделът клиент-сървър



Web Browser



HTTP



Инструменти

Кой и защо изобретява Интернет?

История

Древна Гърция

Фидипид (Phidippides) – герой на древна Гърция.

През 490 преди Христа пробягва 34.5 километра от Маратон до Атина за един ден, за да предупреди за отстъпващите перси.



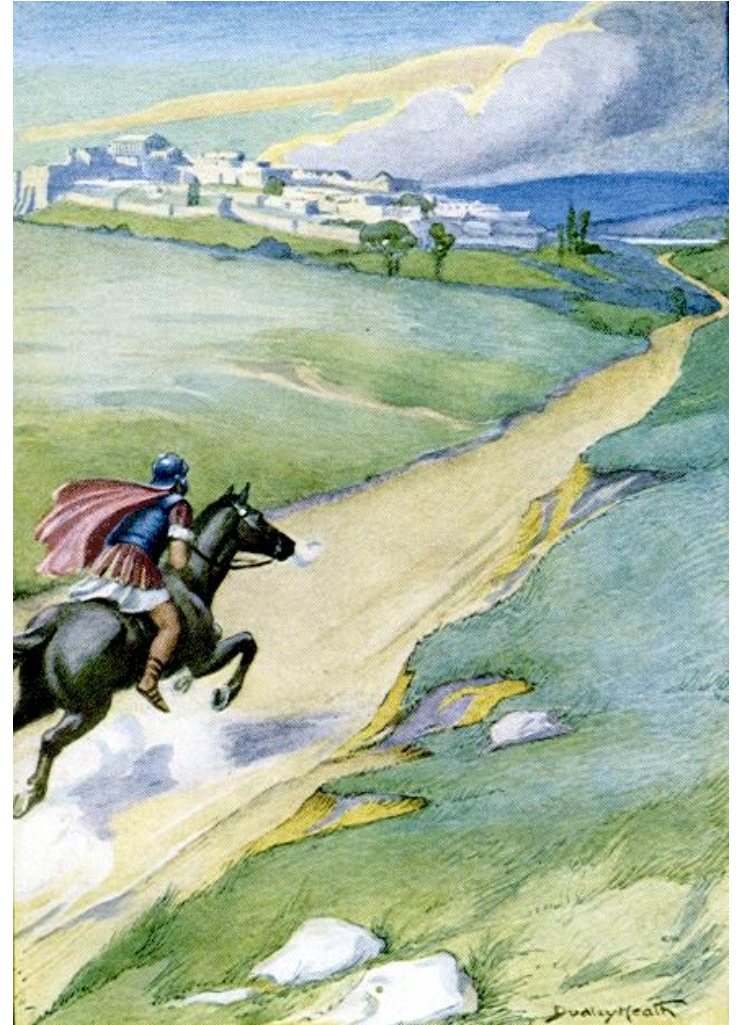
История

Римската Империя

Римската империя от 1^{ви} век преди
Христа използва мрежа от конни станции.

Всяка станция е имала 40 коне готови за
транспортиране на съобщения.

За един ден е можело да се предаде
съобщение на 80-160 мили(128-256км.)



История

Измерване на мрежите

Latency (латентност)

- Времето от изпращането на един бит до неговото пристигане
- еднопосочна, двупосочна
- в секунди (или секунди за разстояние)

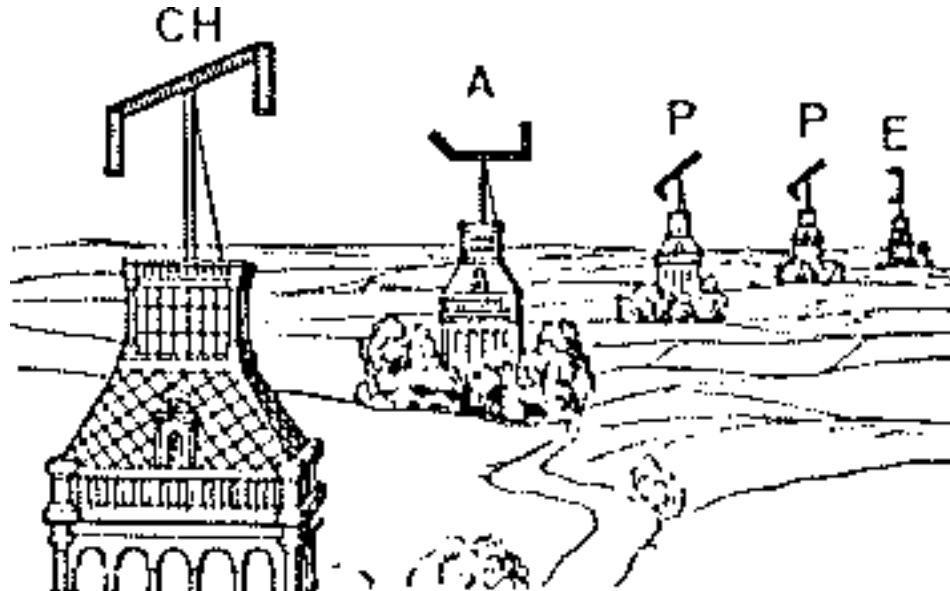
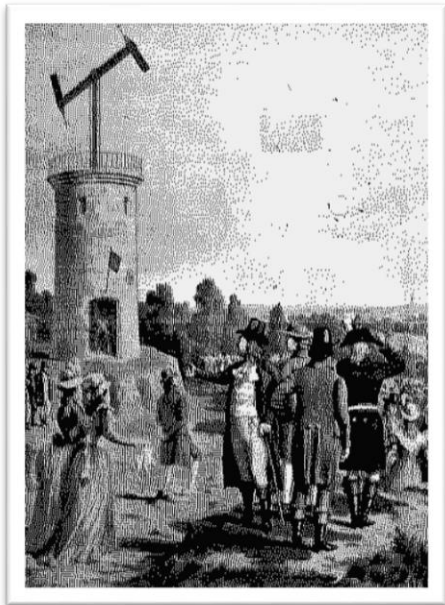
Bandwidth (пропускателност/честотна лента)

- Колко информация може да се предаде за дадено време, в бита/с



История

Семафорната мрежа на Чапе



Клауд Чапе – французин

Париж – Лил, 12 юли 1793 година

Париж – Туолон | 1805 | 750 | 2 | 13

История

Advanced Research Projects Agency Network (ARPANet)

J.C.R. Licklider от университета MIT пръв предлага идеята за глобална мрежа от компютри през 1962.

Leonard Kleinrock от университета MIT и UCLA предлага идеята за “packet switching”, която заляга в основата на интернет връзките.

Lawrence Roberts свързва чрез телефонна линия компютър от Масачузец с компютър в Калифорния през 1965 година.

През 1966 година се премества в DARPA и създава мрежата ARPANET.

На 29 октомври 1969 година, Charley Kline се опитва да изпрати съобщението “login” по ARPANET мрежата, но системата първоначално се срива, когато написва “g”.

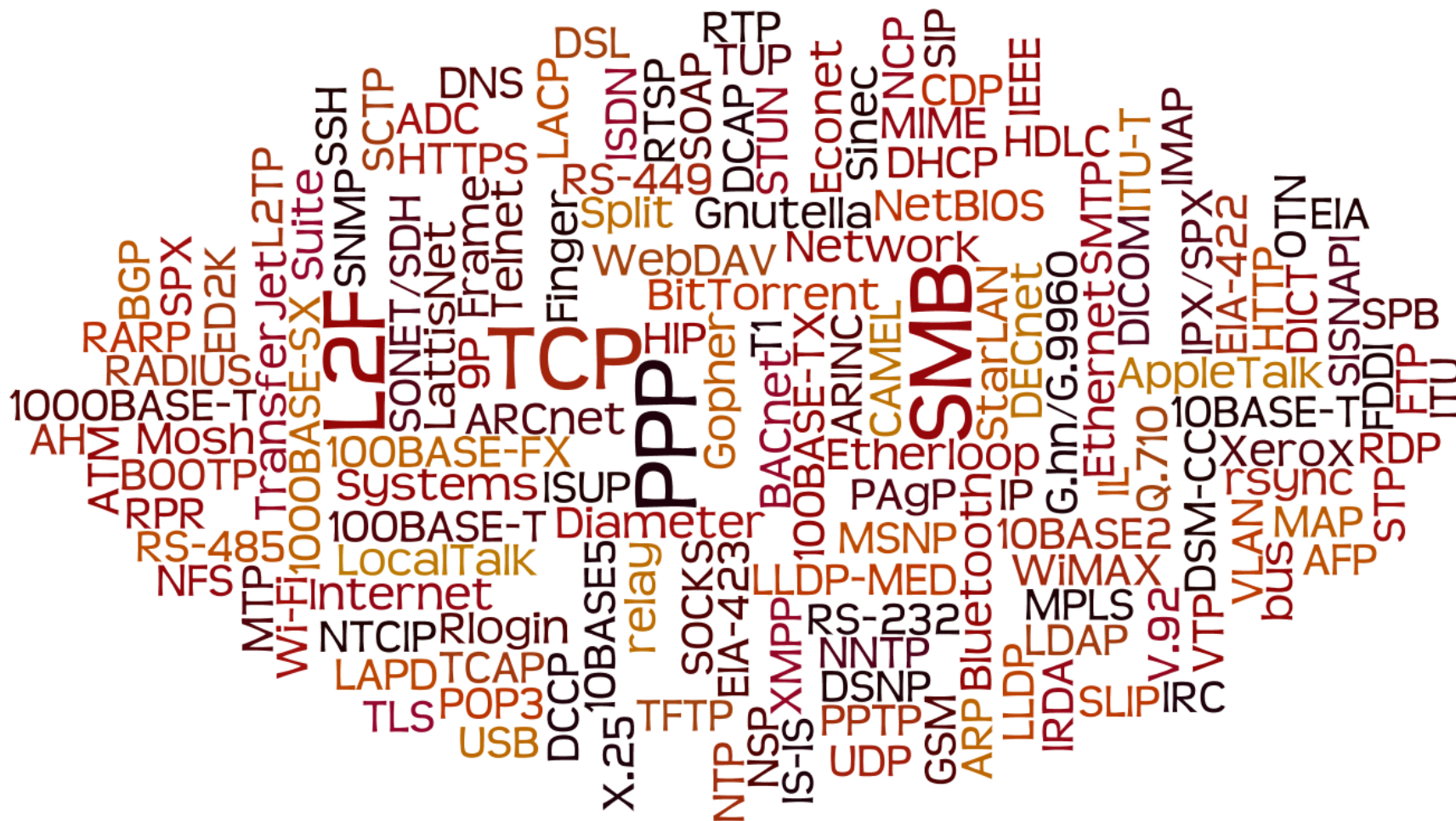
История

Развитие

	Латентност	Честотна лента	Сваляне на една песен
Семафорна мрежа	13 минути	16 b/m	~1824 дена
56K MODEM	40 ms	56 kbit/s	~12 минути
Ethernet	20 ms	10 Mbit/s	~3 секунди
Wireless 802.11g	5 ms	54 Mbit/s	~600 милисекунди
Fast Ethernet	60 μ s	100 Mbit/s	~300 милисекунди
100 Gigabit Ethernet	5 μ s	100 Gbit/s	~300 микросекунди

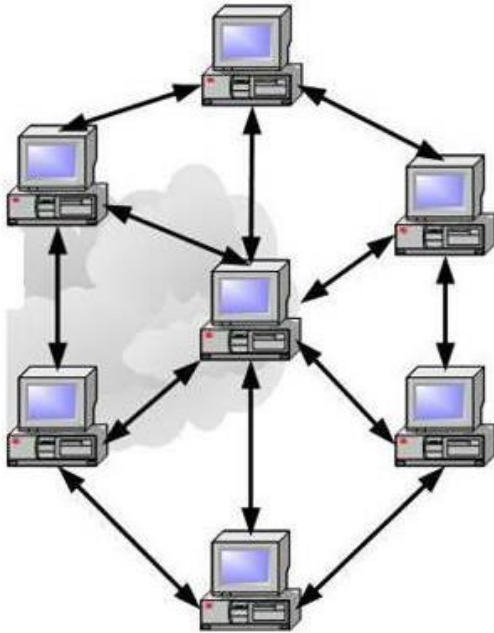
История

Протоколи



История

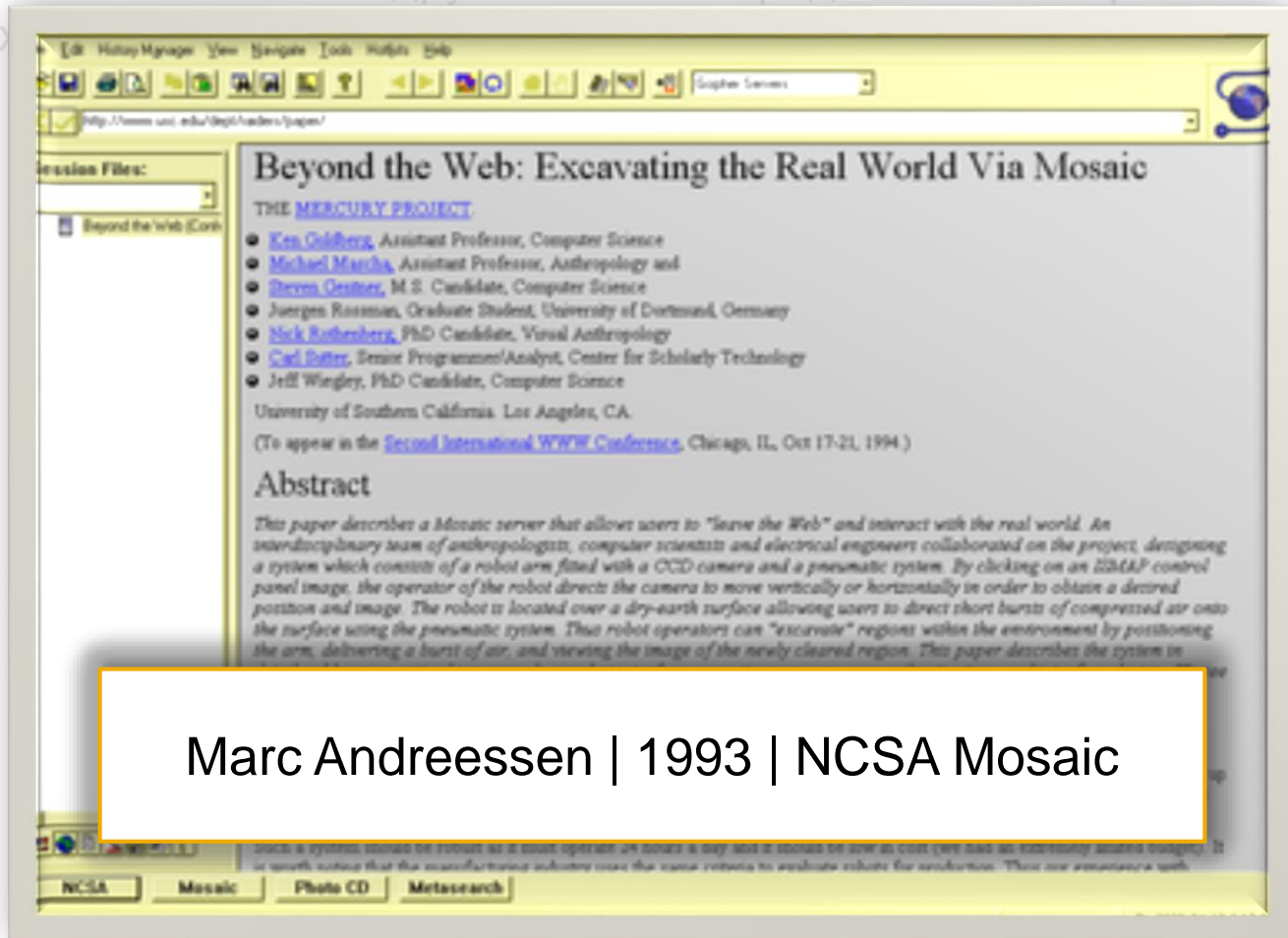
The Internet vs WWW



The Internet vs World Wide Web (WWW)

История WWW

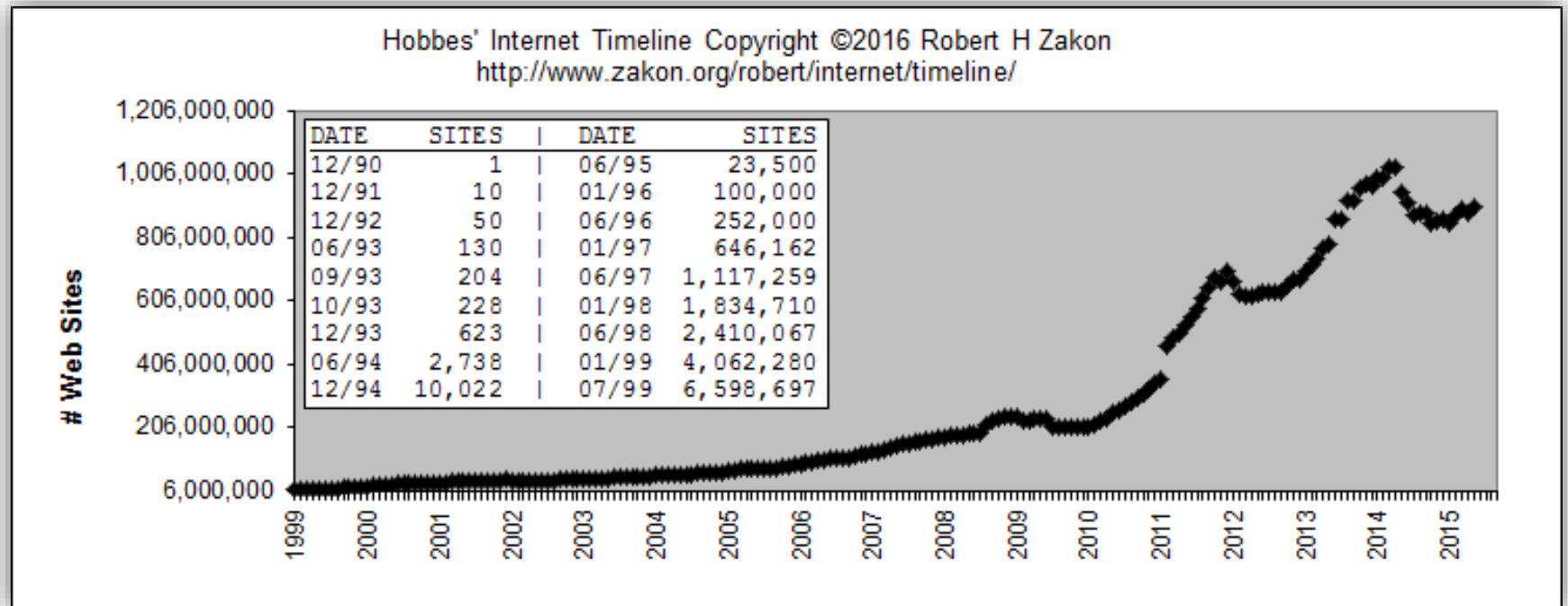
През 1989 Tim Berners-Lee и други от CERN представят нов протокол базиран на "Hypertext"



Marc Andreessen | 1993 | NCSA Mosaic

История

Успеха на WWW



Съдържание



История



Мрежови основи



Моделът клиент-сървър



Web Browser



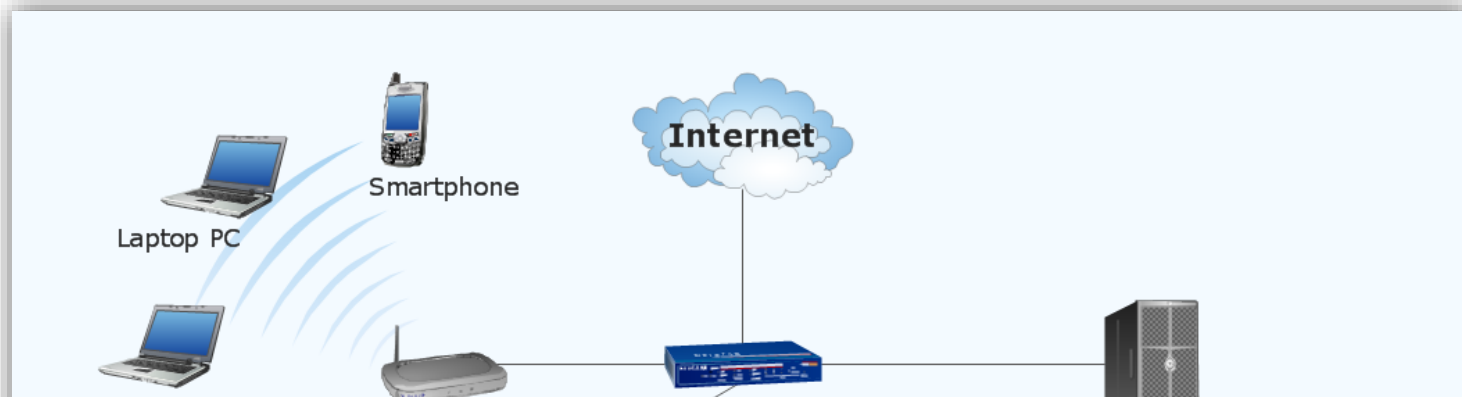
HTTP



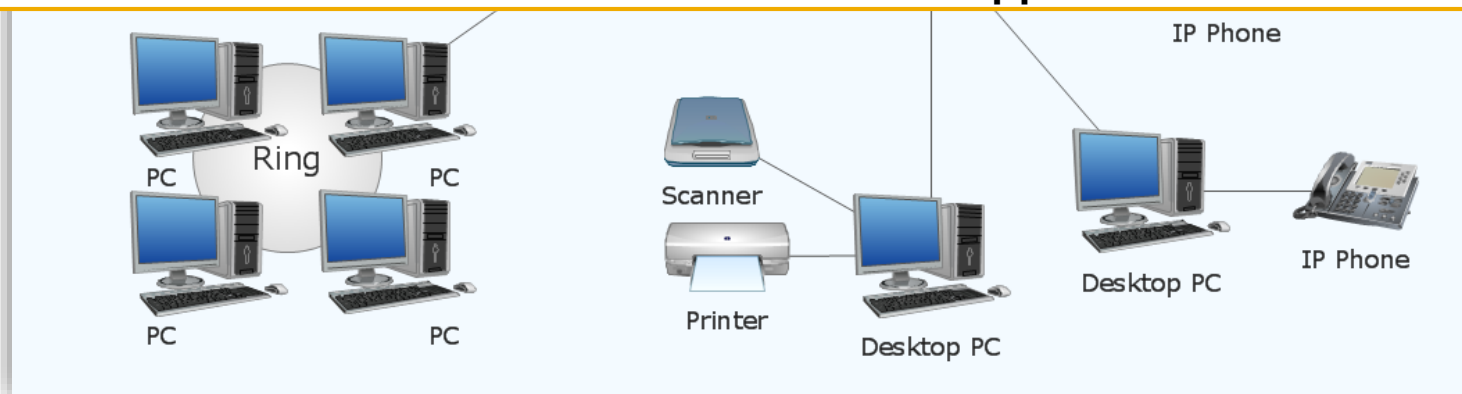
Инструменти

Мрежови основи

Мрежата



**„КАК ВСИЧКИ ТЕЗИ УСТРОЙСТВА ПРЕДАВАТ ИНФОРМАЦИЯ
КЪМ И ОТ ИНТЕРНЕТ И ПОМЕЖДУ СИ“?**



Мрежови основи

OSI (Open Systems Interconnection) модел

№	Слой	Описание	Протоколи
7	Application	Позволява на потребителските приложения да заявяват услуги или информация, а на сървър приложенията — да се регистрират и предоставят услуги в мрежата.	DNS, FTP, HTTP, NFS, NTP, DHCP, SMTP, Telnet
6	Presentation	Конвертиране, компресиране и криптиране на данни.	TLS/SSL
5	Session	Създаването, поддържането и терминирането на сесии. Сигурност. Логически портове.	Sockets
4	Transport	Грижи се за целостта на съобщенията, за пристигането им в точна последователност, потвърждаване за пристигане, проверка за загуби и дублиращи се съобщения.	TCP, UDP
3	Network	Управлява на пакетите в мрежата. Рутинане. Фрагментация на данните. Логически адреси.	IPv4, IPv6, IPX, ICMP
2	Data Link	Предаване на фреймове от един възел на друг. Управление на последователността на фреймовете. Потвърждения. Проверка за грешки. MAC.	ATM, X.25, DSL, IEEE 802.11
1	Physical	Отговаря за предаването и приемането на неструктурирани потоци от данни по физическият носител. Кодирание/декодирание на данните. Свързване на физическият носител.	IEEE 802.11, IEEE 1394, Bluetooth

Мрежови основи

Идентифициране на приложение

Как идентифицираме един компютър в мрежата?



10.199.199.200

IP Адрес

50430

Порт

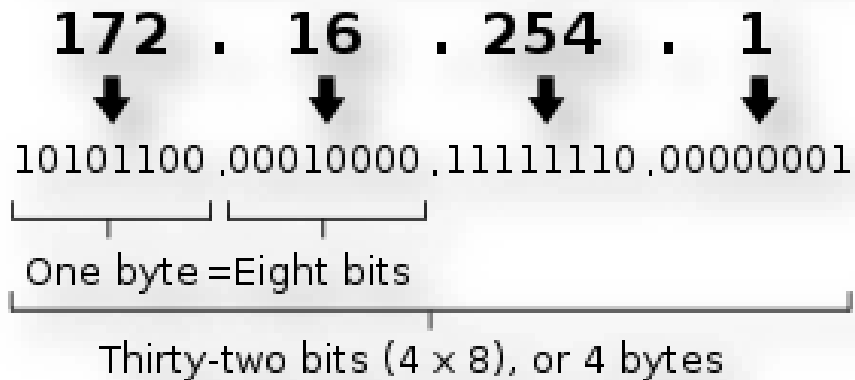
Socket

Мрежови основи

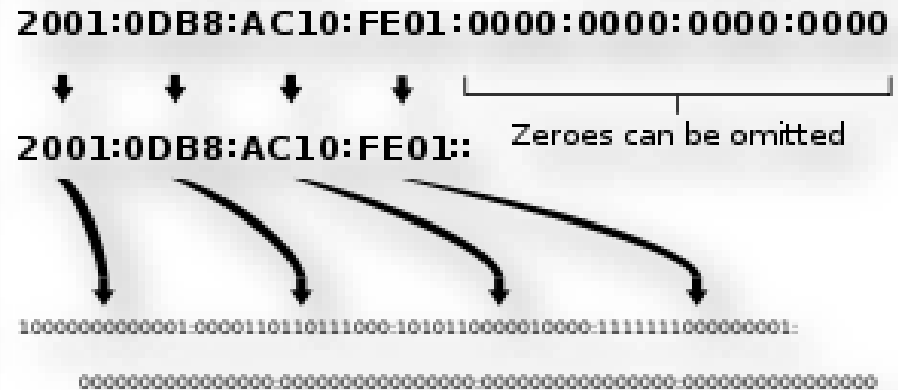
IP протокол 1/2

- Логически адрес
- Internet Protocol
- IPv4, IPv6
- Subnet Mask

An IPv4 address (dotted-decimal notation)



An IPv6 address (in hexadecimal)



Мрежови основи

IP протокол 2/2 (класове адреси)

1 9 2 . 1 6 8 . _ _ 0 . _ _ 1

?

C

Class A	<u>0</u> xxxxxx	Host	Host	Host
Class B	<u>10</u> xxxxxx	Network	Host	Host
Class C	<u>110</u> xxxxx	Network	Network	Host

255.0.0.0

255.255.0.0

255.255.255.0

MACKA

Class	Left-most Bit	Starting IP Address	Last IP Address
A	0xxx	0.0.0.0	127.255.255.255
B	10xx	128.0.0.0	191.255.255.255
C	110x	192.0.0.0	223.255.255.255
D	1110	224.0.0.0	239.255.255.255
E	1111	240.0.0.0	255.255.255.255

Мрежови основи

Портове

- В общият случай, компютърът има една физическа връзка към мрежата. По тази връзка се изпращат и получават данни от/за всички приложения. Портовете се използват, за да се знае кои данни за кое приложение са.
- Предадените данни по мрежата, винаги съдържат в себе си информация за компютъра и порта към които са насочени.
- Портовете се идентифицират с 16 битово число, което се използва от UDP и TCP протокола, да идентифицират за кое приложение се предназначени данните.
- Портовете могат да бъдат от номер 0 до номер 65 535.
- Портове с номера от 0 до 1023 са известни като “well-known ports”. За да се използват тези портове от вашето приложение, то трябва да се изпълнява с администраторски права.

Мрежови основи

TCP/UDP

Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Надеждност: TCP е протокол, който се основава на връзки (connections). Когато един файл се изпрати, той със сигурност ще бъде доставен, освен ако връзката не се прекрати. Ако част от пакетите бъдат изгубени, те ще бъдат предадени отново.	Надеждност: UDP е пакетно ориентиран протокол. Когато се изпрати съобщение по мрежата, не е сигурно дали то ще бъде доставено или дали ще запази своята цялост.
Подредба: Ако се изпратят две последователни съобщения, протокола гарантира, че те ще се получат в реда в който са изпратени.	Подредба: Протокола не гарантира, че съобщенията ще се получат в реда в който са изпратени.
Тежък: Протокола изисква допълнителни мрежови трафик, за потвърждения и изпращане отново на пакети, които са се загубили по мрежата или които не са доставени в правилният ред.	Лек: Няма подредба на съобщенията или потвърждаване за получените пакети.
Streaming: Данните се четат като "stream, ". Може да се изпратят/получат няколко пакета едновременно.	Datagrams: Пакетите се изпращат индивидуално.
Примери: HTTP, SMTP, FTP, SSH и други.	Примери: DNS, IPTV, VoIP, TFTP и други.

Мрежови основи

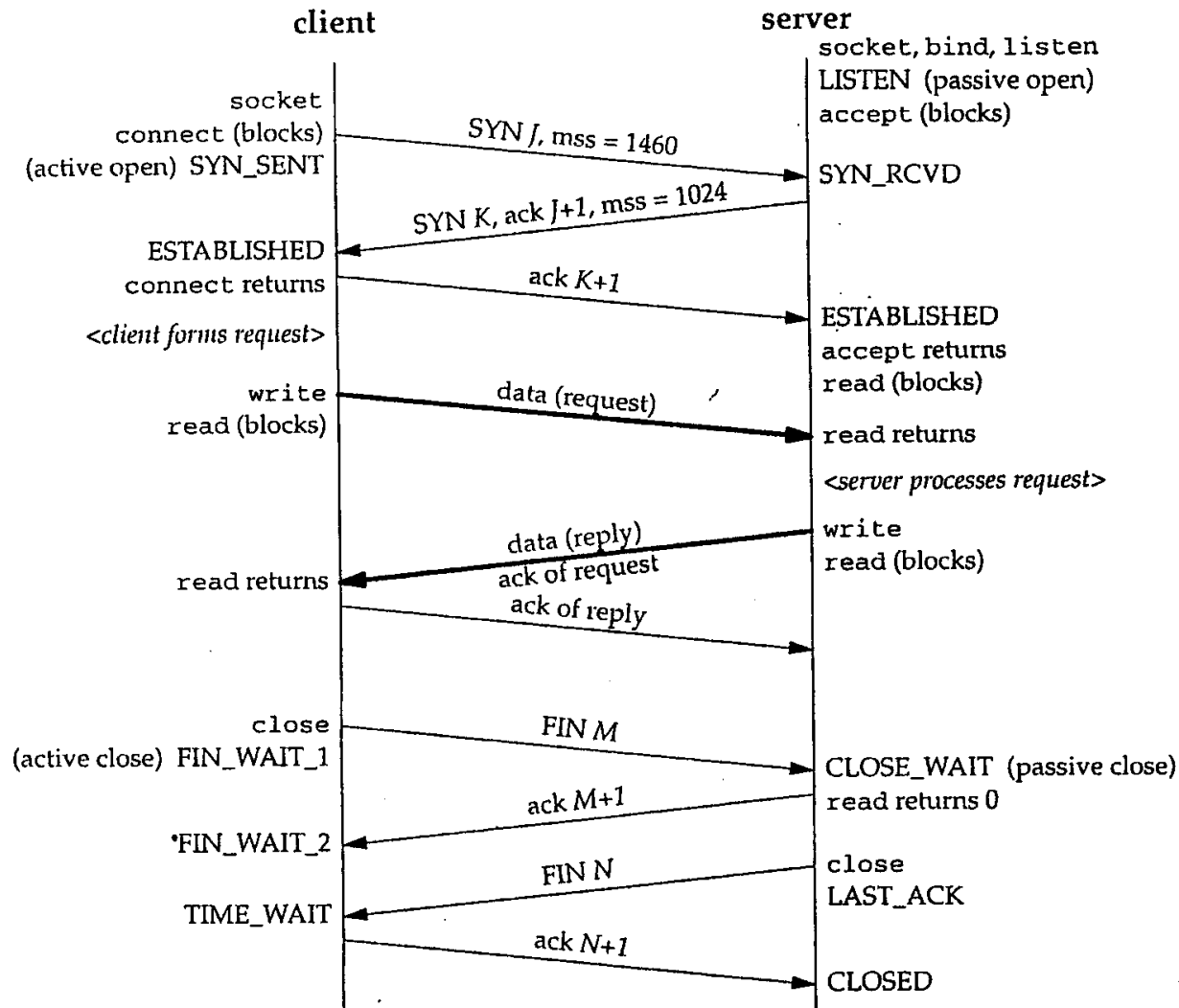
Кога да използваме UDP?

Поради характеристиките на UDP протокола, е добре да се използва при:

- **Broadcast или multicast приложения.**
- **При предаване на данни, които ще бъдат заменени скоро с нови данни**
- **При предаване на данни, които не са критични.**
- **Приложения които ще обработват огромно количество заявки (request), които не създават сесия.**
- **Не трябва да се използва за трансфер на голямо количество данни.**

Мрежови основи

Как работи TCP протокола



Съдържание



История



Мрежови основи



Моделът клиент-сървър



Web Browser



HTTP

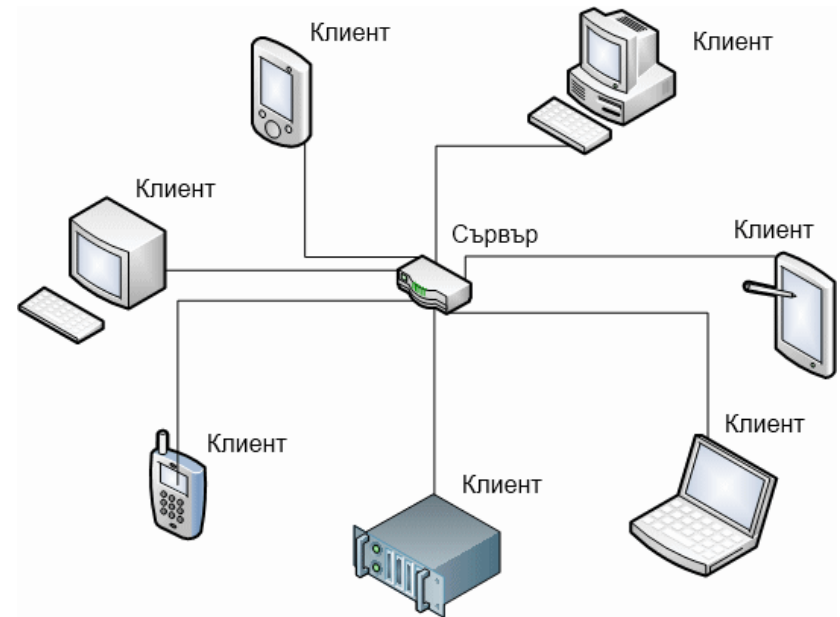
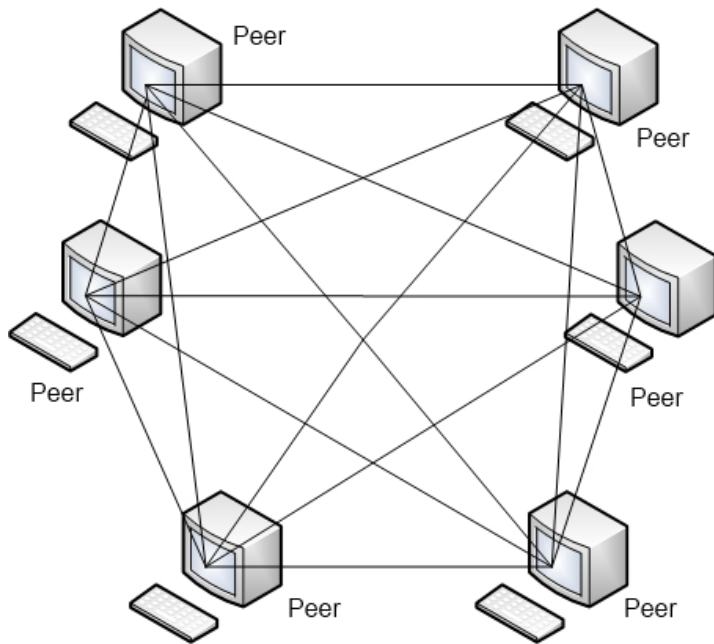


Инструменти

Моделът клиент-сървър

Архитектурни модели

- **Клиент-сървър** е разпределен изчислителен модел, при който част от задачите се разпределят между доставчиците на ресурси или услуги, наречени **сървъри** и консуматорите на услуги, наречени **клиенти**.



- **Peer-to-peer** е разпределен архитектурен модел на приложение, при който задачите се разпределят по еднакъв начин между всички участници (peer, node). Всеки участник е едновременно и **клиент** и **сървър**.

Моделът клиент-сървър

Клиент-сървър – Клиенти

Според наличната функционалност в клиента:

- **Rich** клиенти.
- **Thin** клиенти.

Според семантиката (протокол):

- **Web** клиенти – Браузери (Chrome, Firefox, IE).
- **Mail** клиенти – POP/SMTP клиенти (MS Outlook, Lotus notes).
- **FTP** клиенти – Total Commander, Filezilla, WinSCP.
- ...

Моделът клиент-сървър

Клиент-сървър - Сървъри

Файл сървър (Windows, Samba, UNIX NFS, OpenAFS).

DB сървър (MySQL, PostgreSQL, Oracle, MS SQL Server, Mongo DB, HANA).

Mail сървър (MS Exchange, GMail, Lotus Notes).

Name сървър (DNS).

FTP сървър (ftpd, IIS).

Print сървър.

Game сървър.

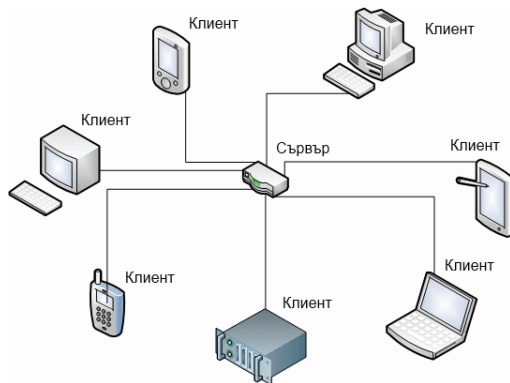
Web сървър (Apache, GWS, MS IIS, nginx).

Application сървър (Tomcat, GlassFish, SAP NetWeaver, JBoss, Oracle, Node.js).

...

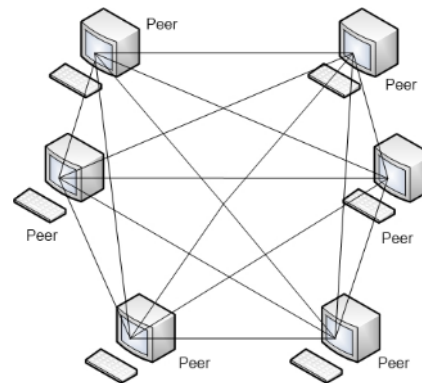
Моделът клиент-сървър

Предимства и недостатъци



Клиент-сървър

- Single Point Of Failure (SPOF).
- Увеличаването на броя на клиентите води до намаляване на производителността.
- 70-95% от времето, през което работи сървъра е idle.



Peer-to-peer

- + Няма SPOF.
- + Няма намаляване на производителността при увеличаване на клиентите.
- Проблеми със сигурността.
- Риск от умишлена промяна на съдържание.
- Липса на контрол върху съдържанието и възможност за загуба на съдържание.
- Труден процес на поддръжка.

Съдържание



История



Мрежови основи



Моделът клиент-сървър



Web Browser



HTTP



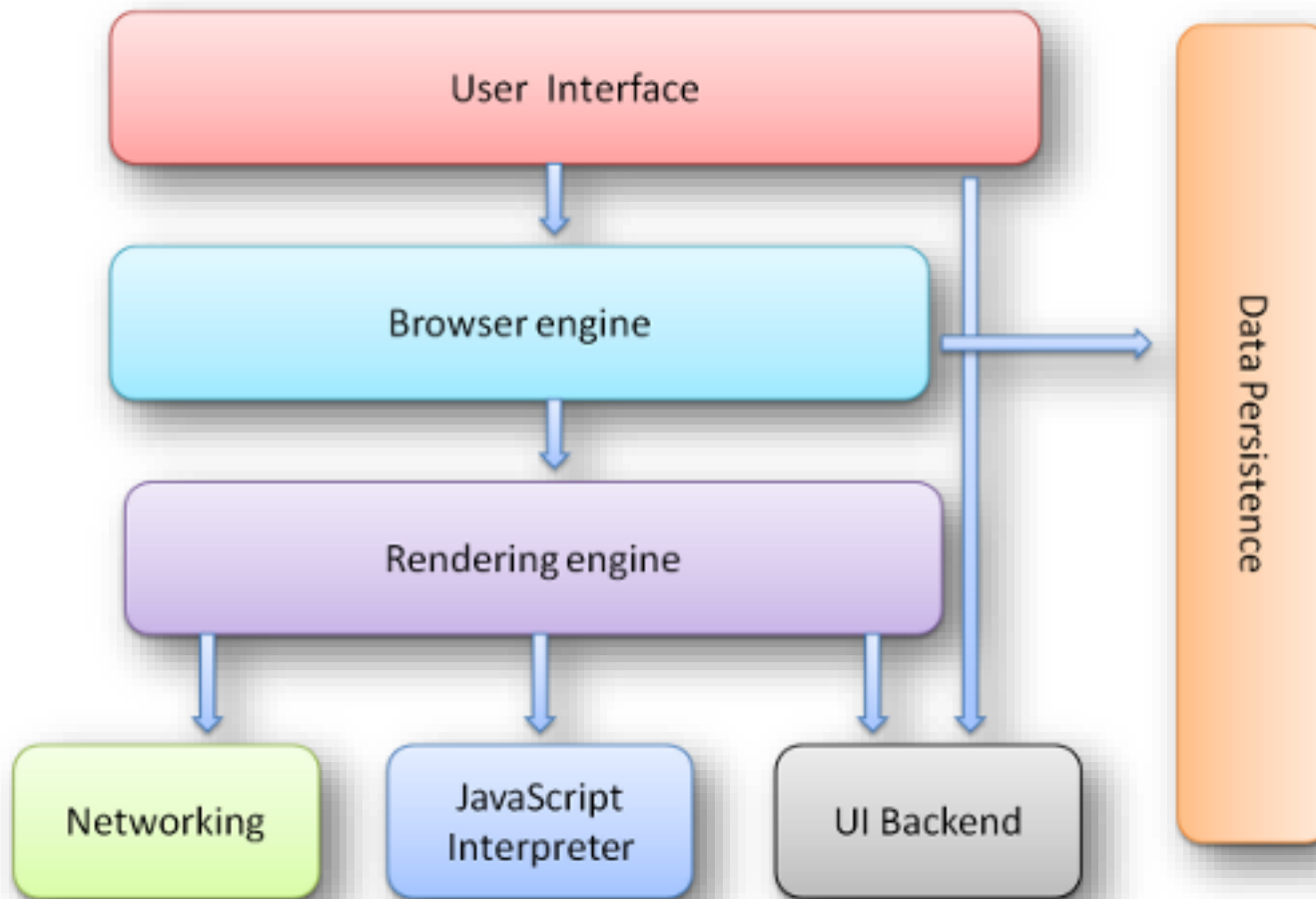
Инструменти

Web Browser



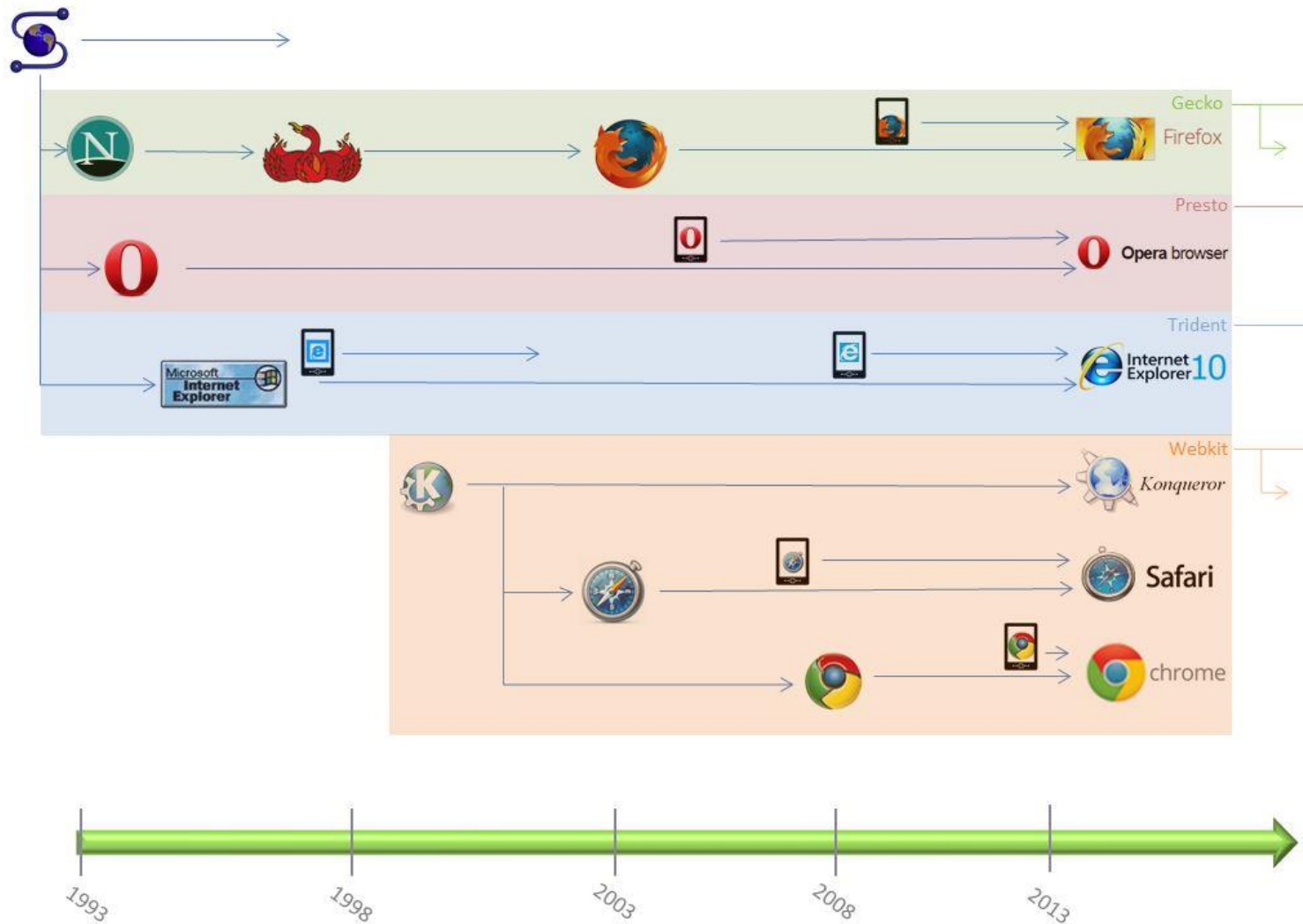
Web Browser

Архитектура



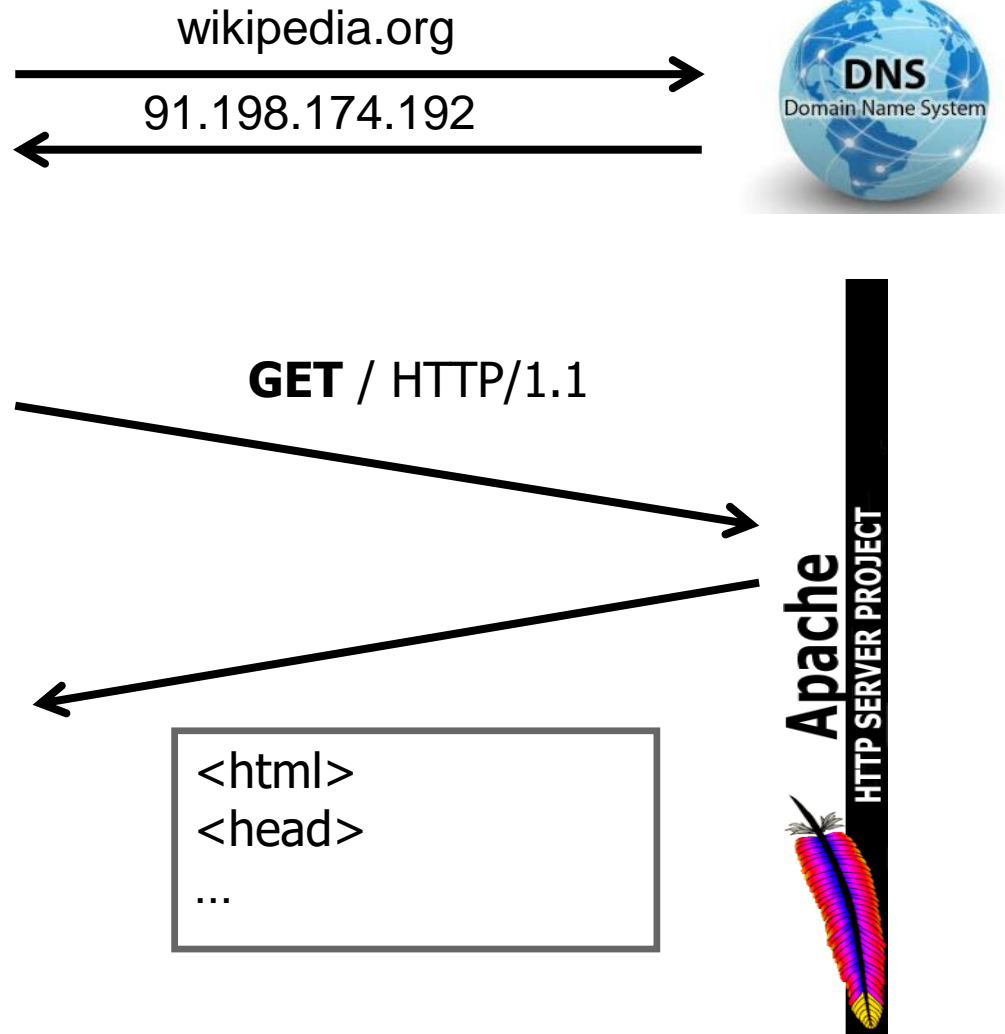
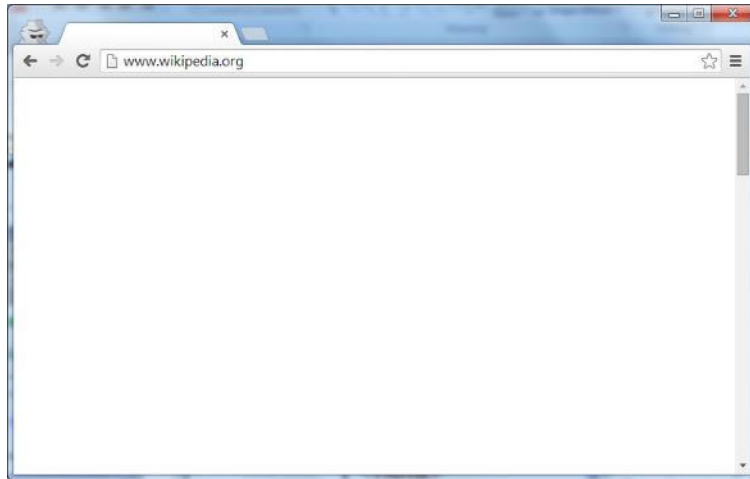
Web Browser

Browser and Rendering Engines



Web Browser

Как функционира



Web Browser

DNS

DNS е глобална, дистрибутирана, надеждна база от данни, която се състои от три компонента:

- ❖ “name space”
- ❖ Сървъри, които предоставят този “name space”.
- ❖ Клиенти (resolvers), който изпращат заявки към сървърите.

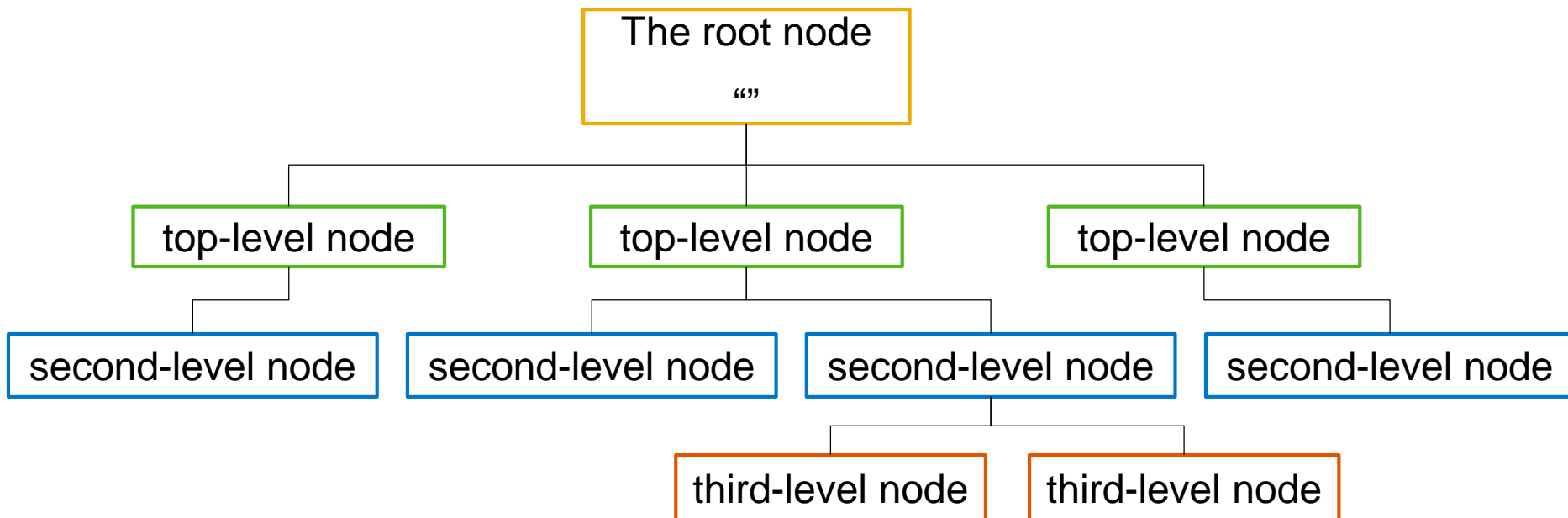
Web Browser

DNS / name space

“name space” е структурата на DNS базата от данни.

Всеки възел от дървото, трябва да има име, не по дълго от 63 байта.

Корена има име null, записано като “”.



Web Browser

DNS / domain names

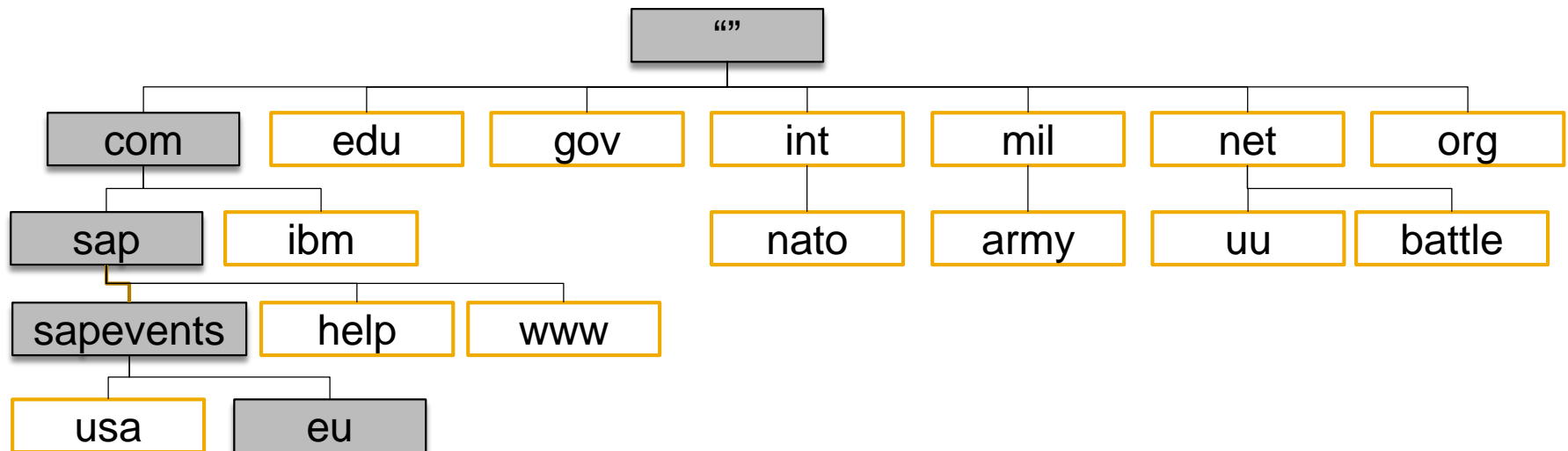
Домейн име е последователност от имена от даден възел до корена на дървото разделени с точки:

eu.sapevents.sap.com

Максималната дълбочина е 127 нива, а максималната дължина на едно домейн име е 255 символа.

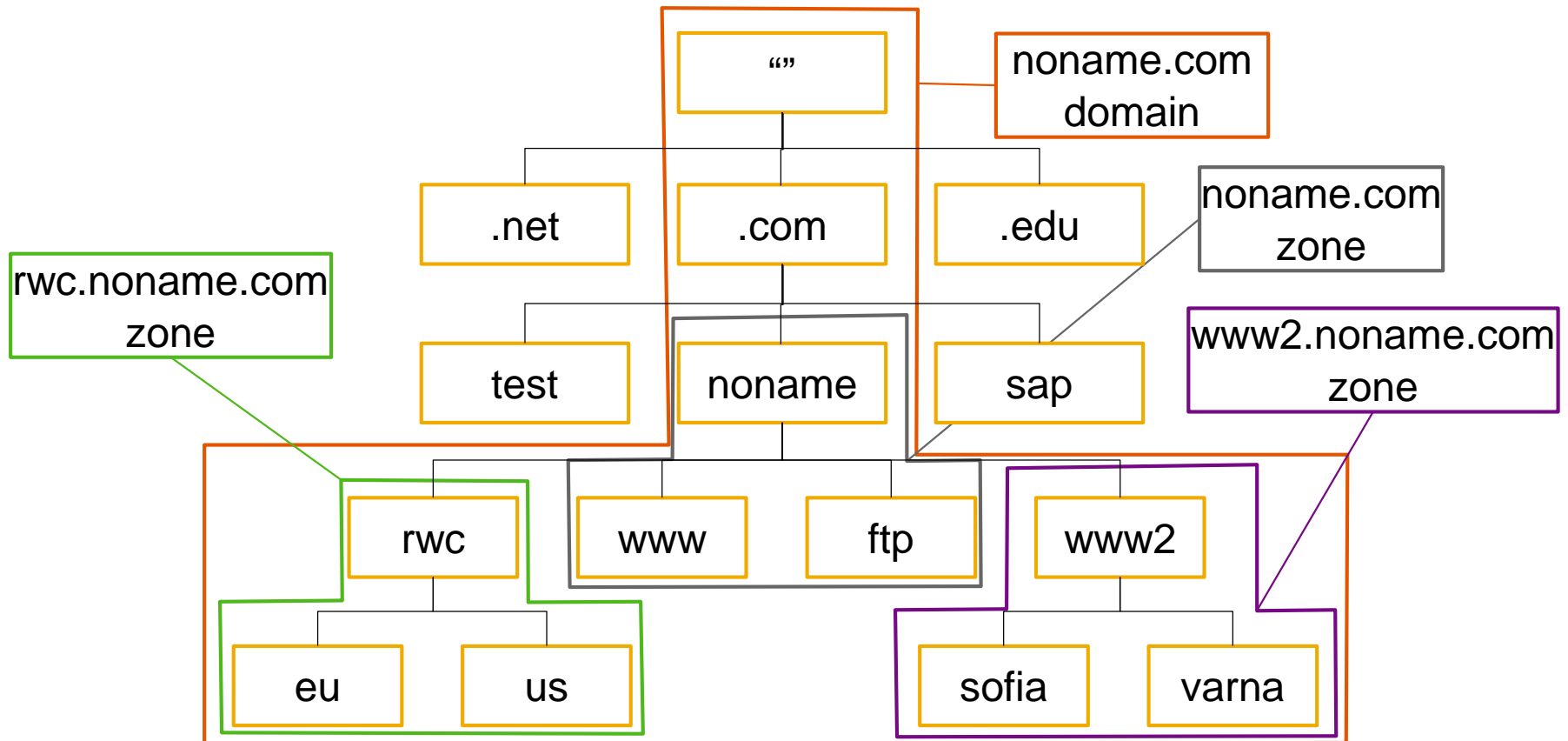
Домейн името определя разположението на възела в дървото.

FQDN (Fully Qualified Domain Name) – абсолютно домейн име.



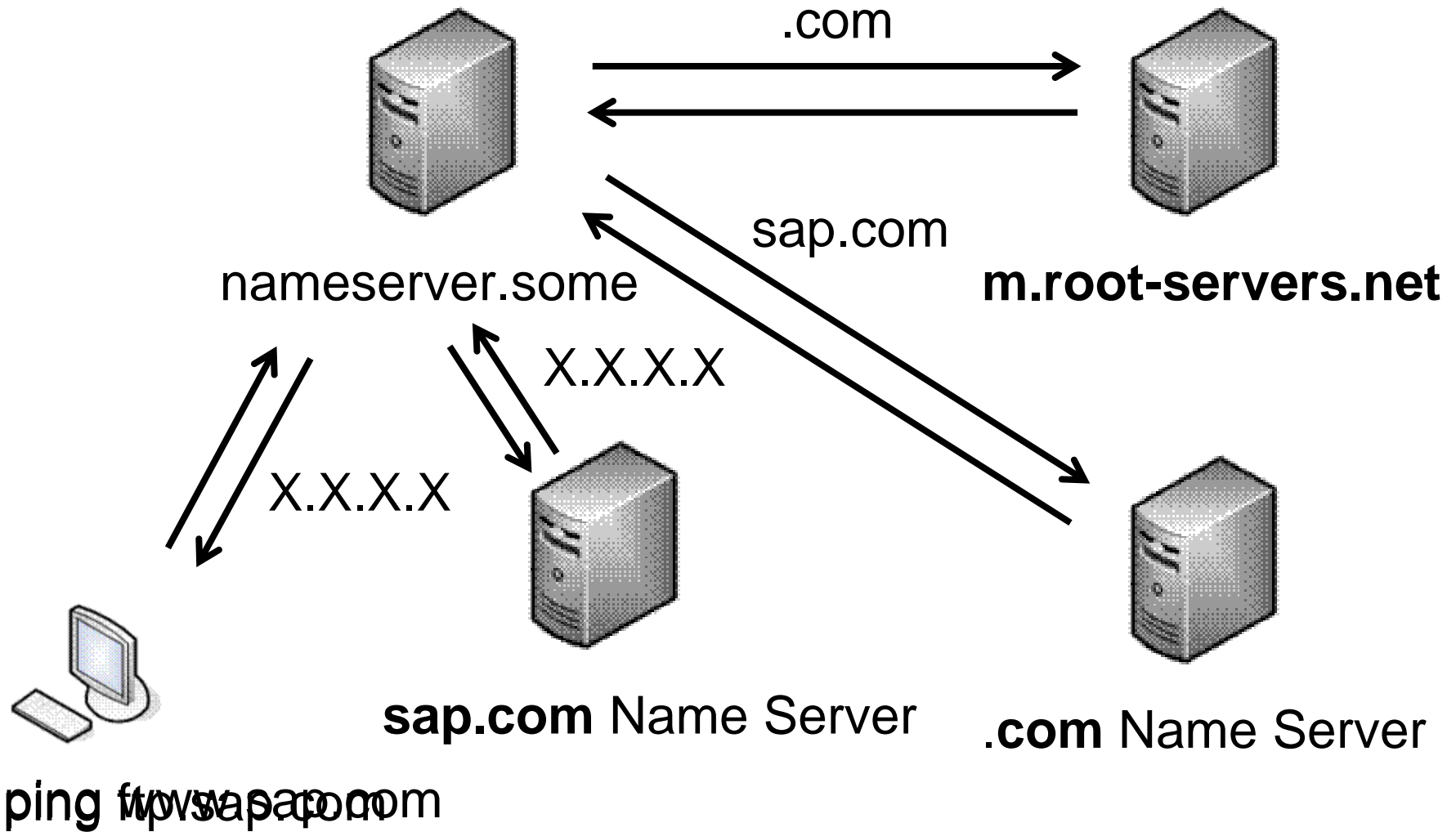
Web Browser

DNS / zones



Web Browser

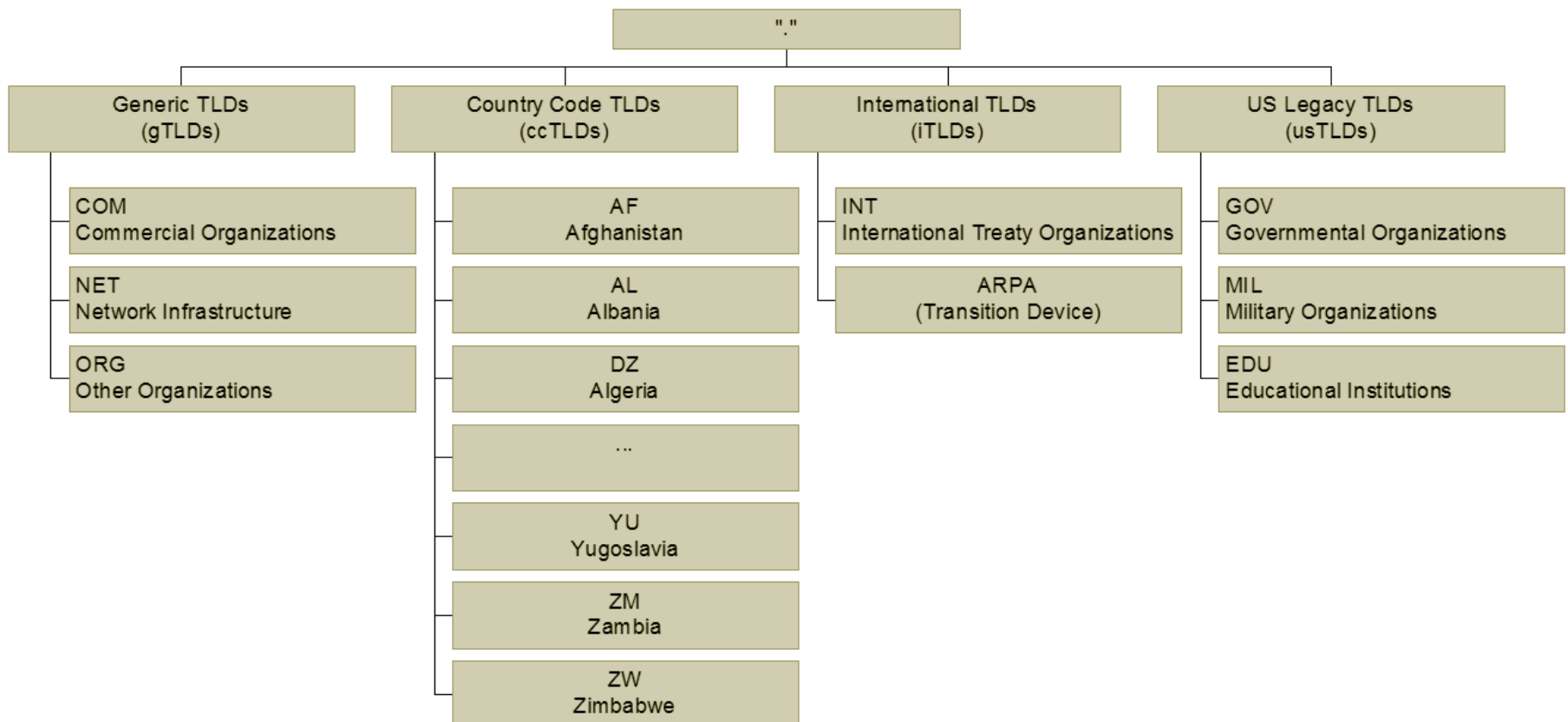
DNS / как функционира



Web Browser

DNS / TLD

TLD – Top Level Domains



Web Browser

DNS / root name servers

Root name server-ите има адресите на всички TLD.

Nameserver	Operated by
A	Verisign (US East Coast)
B	University of S. California –Information Sciences Institute (US West Coast)
C	Cogent Communications (US East Coast)
D	University of Maryland (US East Coast)
E	NASA (Ames) (US West Coast)
F	Internet Software Consortium (US West Coast)
G	U. S. Dept. of Defense (ARL) (US East Coast)
H	U. S. Dept. of Defense (DISA) (US East Coast)
I	Autonomica (SE)
J	Verisign (US East Coast)
K	RIPE-NCC (UK)
L	ICANN (US West Coast)
M	WIDE (JP)

Съдържание



История



Мрежови основи



Моделът клиент-сървър



Web Browser



HTTP



Инструменти

HTTP

Характеристики на HTTP

- **Приложен протокол** – като транспортен протокол, почти винаги се ползва TCP/IP, в редки случаи и UDP. По подразбиране слуша на порт 80.
- **Модел „Заявка-Отговор“ (“Request-Response”)** - служи за комуникационен канал в „Клиент-Сървър“ архитектура, като следва строги правила за ред и формат на съобщенията между участниците.
- **Не пази състояние (Stateless)** – всяка клиентска заявка е независима сама по себе си. Сървърът не обвързва логически серия заявки от определен клиент. Това води до липса на вграден в протокола механизъм за поддържане на сесии.
- **HTTP Транзакция** - /опростен модел – без преизползване на конекцията/
 1. Клиентът отваря комуникационен канал (TCP сокет)
 2. Изпращане на заявка от клиента към сървъра
 3. Сървърът връща отговор на клиента
 4. Затваряне на сокет-а от сървъра.

HTTP

Видове HTTP съобщения

- **Заявка** – инициатор е клиентът – подава информация на сървъра, достъп до който ресурс иска да получи и каква операция иска да извърши с него (и евентуални входни параметри). **Клиент** (условно наречен User-Agent в HTTP) може да бъде всяко софтуерно приложение, спазващо правилата на протокола на комуникация.
- **Отговор** – изпраща се от веб сървъра, като резултат от изпълнението на клиентска заявка. Под **веб сървър** разбираме софтуерно приложение, служещо като доставчик на дадени услуги върху определени негови ресурси.

HTTP

HTTP Заявка

- **Начален ред**

HTTP Метод (Глагол) – указва типът операция, която клиентът иска да извърши със заявеният ресурс.

URL – уникален локатор на заявеният ресурс

Версия на HTTP – версията на протокола, която ще се ползва за комуникация

GET en.wikipedia.org/w/index.php HTTP/1.1

- **Хедъри** - опционални (HTTP 1.1 задължава специфицирането на хедър HOST в заявката). Възможно е да дефинира множество хедъри, като всеки от тях заема точно един ред и следва форматът: “Име на хедър: Стойност на хедър”

Connection: Keep-Alive

Host: en.wikipedia.org

- **Данни (тяло)** – опционални, може да съдържат множество редове, включително и празни

HTTP

HTTP Заявка / Пример

```
GET /w/index.php?search=Students&title=Special%3ASearch HTTP/1.1
```

```
Accept: application/x-ms-application, image/jpeg,  
application/xaml+xml, image/gif, image/pjpeg, application/x-ms-  
xbap, */*  
Referer: http://en.wikipedia.org/wiki/Main_Page  
Accept-Language: en-US  
User-Agent: Mozilla/4.0  
Accept-Encoding: gzip, deflate  
Host: en.wikipedia.org  
Connection: Keep-Alive
```

<Празен ред>

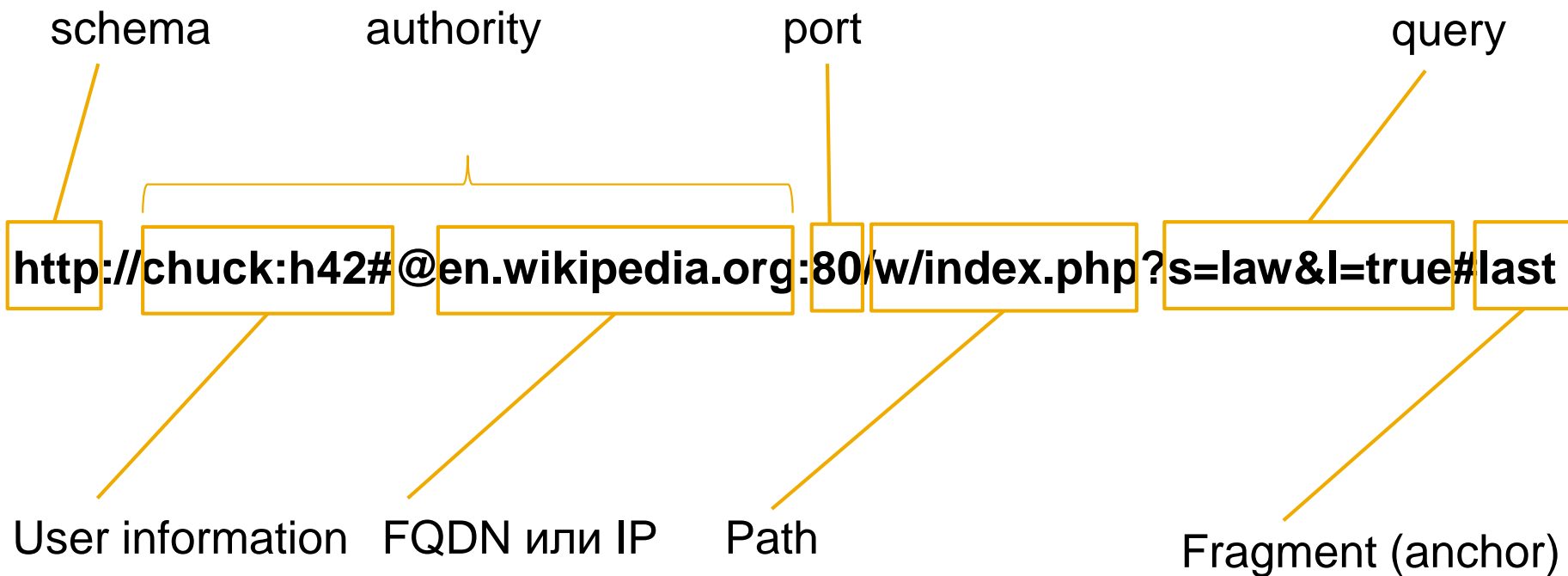
<GET заявките нямат тяло!>

HTTP

HTTP Ресурси

Унифициран локатор на ресурси (URL)

- Стандартизиран адрес на даден мрежов ресурс (документ или страница).
- Всяка уеб страница е идентифицирана уникално чрез URL



HTTP

Видове HTTP методи

- **GET** – за зареждане на ресурс от сървъра
- **POST** - изпраща данни (например от HTML форма) за обработка от сървъра. Данните се съдържат в тялото на заявката
- **HEAD** - идентичен с GET, с разликата, че отговорът няма да върне тяло, а само хедъри
- **PUT** – ъплоудва специфичен ресурс
- **DELETE** – трие специфичен ресурс
- **TRACE** – указва на сървъра да върне низа на заявката в тялото на отговора
- **OPTIONS** – казва на сървъра да му върне всички позволени методи за даден ресурс
- **CONNECT** – за работа с проксита
- **PATCH** – за подмяна на части от ресурса

HTTP

HTTP Отговор

- **Начален ред** – съдържа 3 елемента, разделени с празно пространство помежду си:

Версия на HTTP

Статус код – обяснява резултата на изпълнението на заявката

Причина – кратко обяснение на статус-кода

HTTP/1.1 200 OK

- **Хедъри**

Date: Sat, 06 Oct 2015 15:08:15 GMT

Server: Apache

- **Данни (Тяло)** – отговорите обикновено връщат данни, като тук най-често се съдържа **HTML документът**, получен на базата на клиентската заявка.

HTTP

HTTP Отговор / Пример

HTTP/1.1 200 OK

```
Date: Sat, 17 Nov 2012 15:08:15 GMT
Server: Apache
X-Content-Type-Options: nosniff
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: en
Vary: Accept-Encoding, Cookie
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Encoding: gzip
Content-Length: 8582
```

<Празен ред>

<HTML>Пропускаме документа за простота!</HTML>

HTTP

HTTP Статус Кодове / 1

Трицифрени кодове, идентифициращи какъв е резултът от обработката на клиентската заявка

Групирани са в 5 категории, на базата на цифрата на стотиците

1. Група 100 – те са чисто информационни, не дават индикация дали заявката е била успешна или не. Служат за „временни“ кодове, т.е. заявката е пристигнала, но сървърът не е готов с резултата все още:

100 Continue
101 Switching protocols

2. Група 200 – сървърът е обработил успешно клиентската заявка

200 OK
206 Partial content

HTTP

HTTP Статус Кодове / 2

3. Група 300 – ресурсът е наличен, но е разположен на друго място

301 Moved permanently
307 Temporary redirect

304 Not Modified

4. Група 400 - клиентска грешка

400 Bad Request
401 Not Authorized

404 Not Found
408 Request Timeout

5. Група 500 - сървърна грешка

500 Internal Server Error
503 Service Unavailable

501 Not Implemented

HTTP

HTTP Хедъри / 1

- **Основни (General headers)** – могат да се ползват едновременно и в заявки, и в отговори. Съдържат информация (мета-данни) за самото съобщение или за метода на комуникация

Connection: keep-alive

Date: Sat, 17 Nov 2016 16:08:15 GMT

- **Заявка (Request headers)** – специфични са само за заявките и могат да съдържат данни за самата заявка или за клиента

Accept: text/html

Accept-Charset: utf-8

Accept-Language: en-US

User-Agent: Mozilla/4.0

HTTP

HTTP Хедъри / 2

- **Отговор (Response headers)** - съдържат информация (мета-данни) за сървъра и формата на съобщението

Server: Apache

Allow: GET, HEAD

- **Същински (Entity headers)** – информация за самото съдържание на данни (тяло) и/или за ресурса, заявен от клиента:

Content-Language: en

Content-Encoding: gzip

Content-Length: 8582

Last-Modified: Tue, 15 Nov 2016 12:45:26 GMT

HTTP

HTTP Хедъри / User Agent

User Agent е софтуер, който извършва действие от името на потребителя:

- E-mail клиенти.
- Web Browser-и.
- Месинджъри: Skype, WhatsApp.
- ...

Примерен низ за Google Chrome Web Browser:

Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/53.0.2785.143 Safari/537.36

HTTP

Кеширане

- **Браузър кеш** - механизъм за временно съхранение на ресурси, с цел по-бързият им достъп.
- **Срок на годност (“expiration period”)** – целта е да се елиминират HTTP заявки, които биха получили еднакъв документ като отговор. За целта браузър кешът трябва да знае за колко дълго може да се “довери” на кешираният документ.

Cache-Control: max-age=3600

Expires: Thu, 21 Oct 2015 16:00:00 GMT

- **Актуалност на данните (“validation”)** – сървърът предоставя възможност на клиента да провери дали кешираните му ресурси са били променени.

Last-Modified: 01 Dec 2012 16:00:00

If-Modified-Since: 01 Dec 2012 16:00:00

HTTP

HTTP Сесии

Сесията е концепция, която позволява да се поддържа връзка (състояние) между 2 или повече http requests, изпратени към даден сървър в Internet.



HTTP

HTTP Сесии / Механизми

- **Cookies (бисквитки)**

- **Hidden fields forms**

HTML страницата трябва да съдържа скрита (hidden) форма:

```
<INPUT TYPE="HIDDEN" NAME="jsessionId" VALUE="...">
```

- **URL Rewriting (презаписване)**

Може да добавите в края на всяко URL данни, които да унифицират сесията, за да може сървъра да прочете тези данни и да намери вашата сесия.

```
http://<my_java_site>?jsessionid=I_am_unique_session_identifier
```

HTTP

Cookies

- Cookie-тата са малки текстови файлове генерирани от сървъра и изпратени на клиента в header-ите.

Как работят бисквитките:

1. Клиентът изпраща request към сървъра.
2. Сървърът отговоря и в header-ите на response-а праща към клиента cookie-тата, които ще се ползват за проследяване на сесията

Примерен отговор (response) на apache tomcat web container-а съдържа header:

Set-Cookie: JSESSIONID=ACFF1B473DAB71CD27AA16049D61265E;

3. Всеки следващ request, изпратен от клиента, трябва да съдържа Cookie header-а, за да може сървърът да намери сесията на клиента

Cookie: JSESSIONID=ACFF1B473DAB71CD27AA16049D61265E

HTTP

Cookies / Атрибути

Cookie-тата са дефинирани в RFC 2109.

Атрибути:

- **Comment** - обикновено се използва от програмистите, за да обосноват нуждата от ползването на cookie-то.
- **Domain** – определя домейна, за който cookie-то е валидно и ще бъде изпращано.
- **Max-age** – задава lifetime-а на cookie-то в секунди. След като изтече валидността на cookie-то, клиентът не трябва да го праща повече.
- **Path** – специфицира subset от URL-та, където cookie-то може да бъде изпращано.
- **Secure** – този атрибут указва, че cookie-то може да бъде трансферирано само по https протокола.
- **HttpOnly** – когато този атрибут е добавен, cookie-то не може да бъде четено или променяно от JavaScript
- **Version** – цяло число, което определя на коя версия на RFC-то отговоря cookie-то.

HTTP

HTTP2

- Защо е нужен?
- HTTP/2.0 или HTTP/2?
- Какви са разликите с HTTP/1.x?
 - двоичен;
 - напълно multiplexed;
 - паралелизъм само с една TCP връзка за всеки origin;
 - компресия на хедъри;
 - Разрешава “push” от сървъра обратно към клиента без предхождаща заявка.
- Кои браузери поддържат HTTP2 към момента?
 - Firefox, Chrome. Opera, Safari, Internet Explorer, Edge

Съдържание



История



Мрежови основи



Моделът клиент-сървър



Web Browser



HTTP

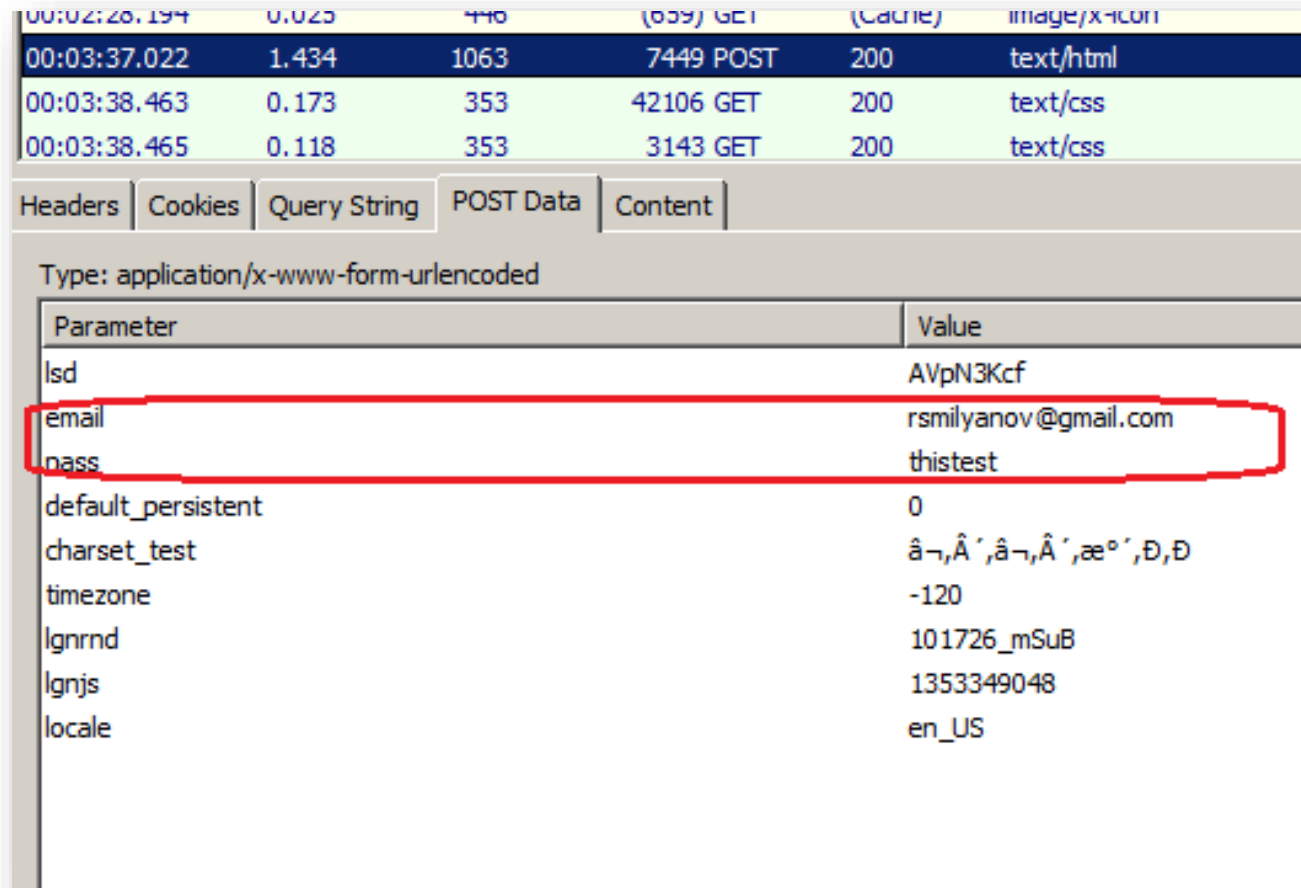


Инструменти

Инструменти

HttpFox plugin

HttpFox plugin за Mozilla Firefox за проследяване на HTTP трафика.



The screenshot displays the HttpFox plugin interface. The top section shows a list of HTTP requests. The bottom section shows the details of a selected POST request, including a table of parameters. The 'email' and 'pass' parameters are highlighted with a red rectangle.

Time	Size	Length	Method	Status	MIME Type
00:02:28.194	0.023	440	(639) GET	(Cache)	image/x-icon
00:03:37.022	1.434	1063	7449 POST	200	text/html
00:03:38.463	0.173	353	42106 GET	200	text/css
00:03:38.465	0.118	353	3143 GET	200	text/css

Headers Cookies Query String POST Data Content

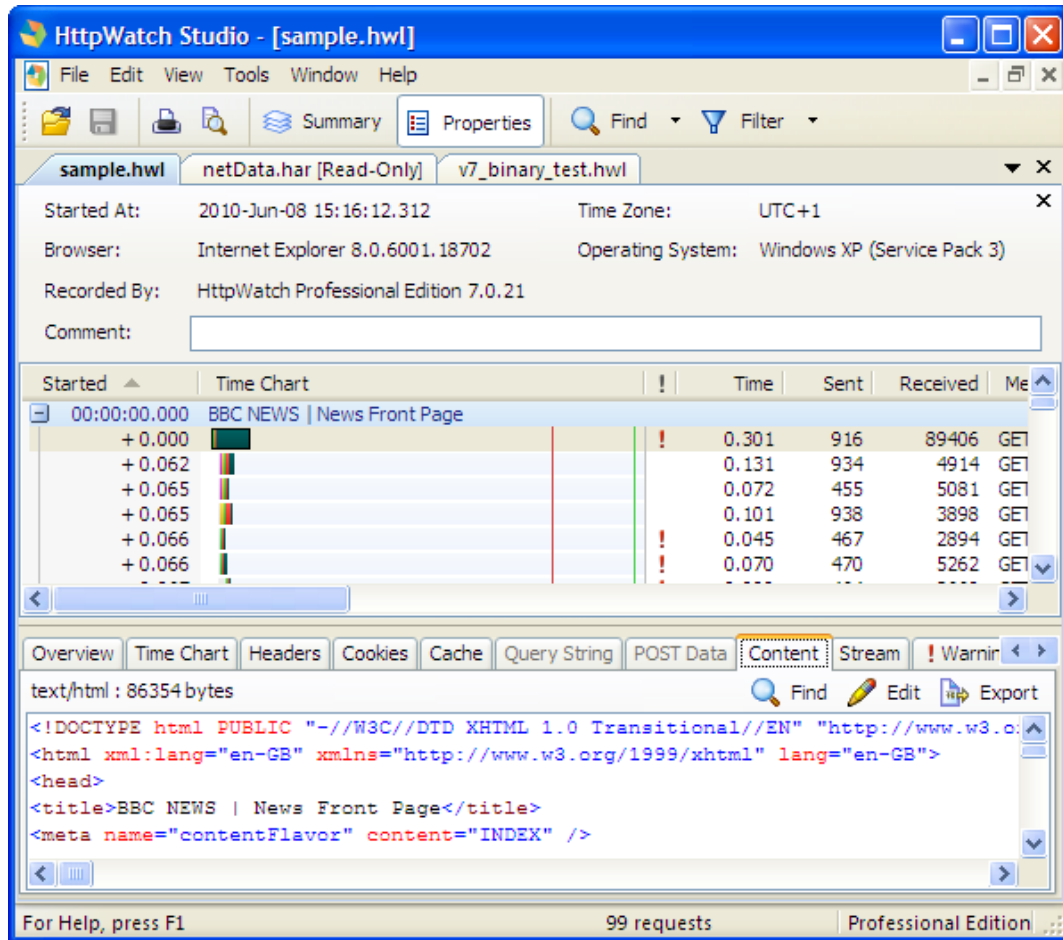
Type: application/x-www-form-urlencoded

Parameter	Value
lsd	AVpN3Kcf
email	rsmilyanov@gmail.com
pass	this test
default_persistent	0
charset_test	â¬,Â´,â¬,Â´,æ°,Ð,Ð
timezone	-120
lgnrnd	101726_mSuB
lgnjs	1353349048
locale	en_US

Инструменти

HttpWatch

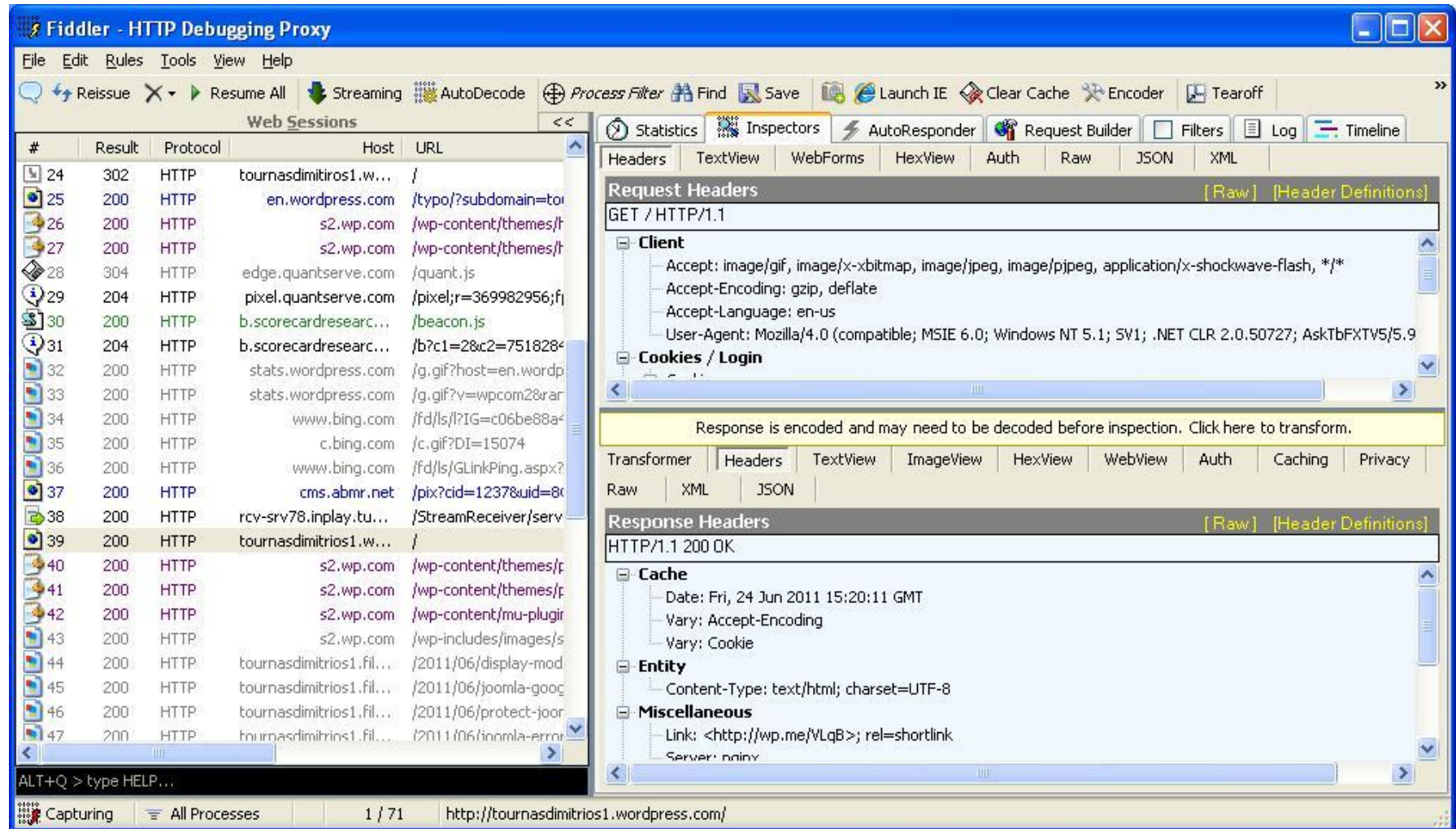
HttpWatch за проследяване на HTTP трафика.



Internet Explorer
Firefox
iPhone, iPad

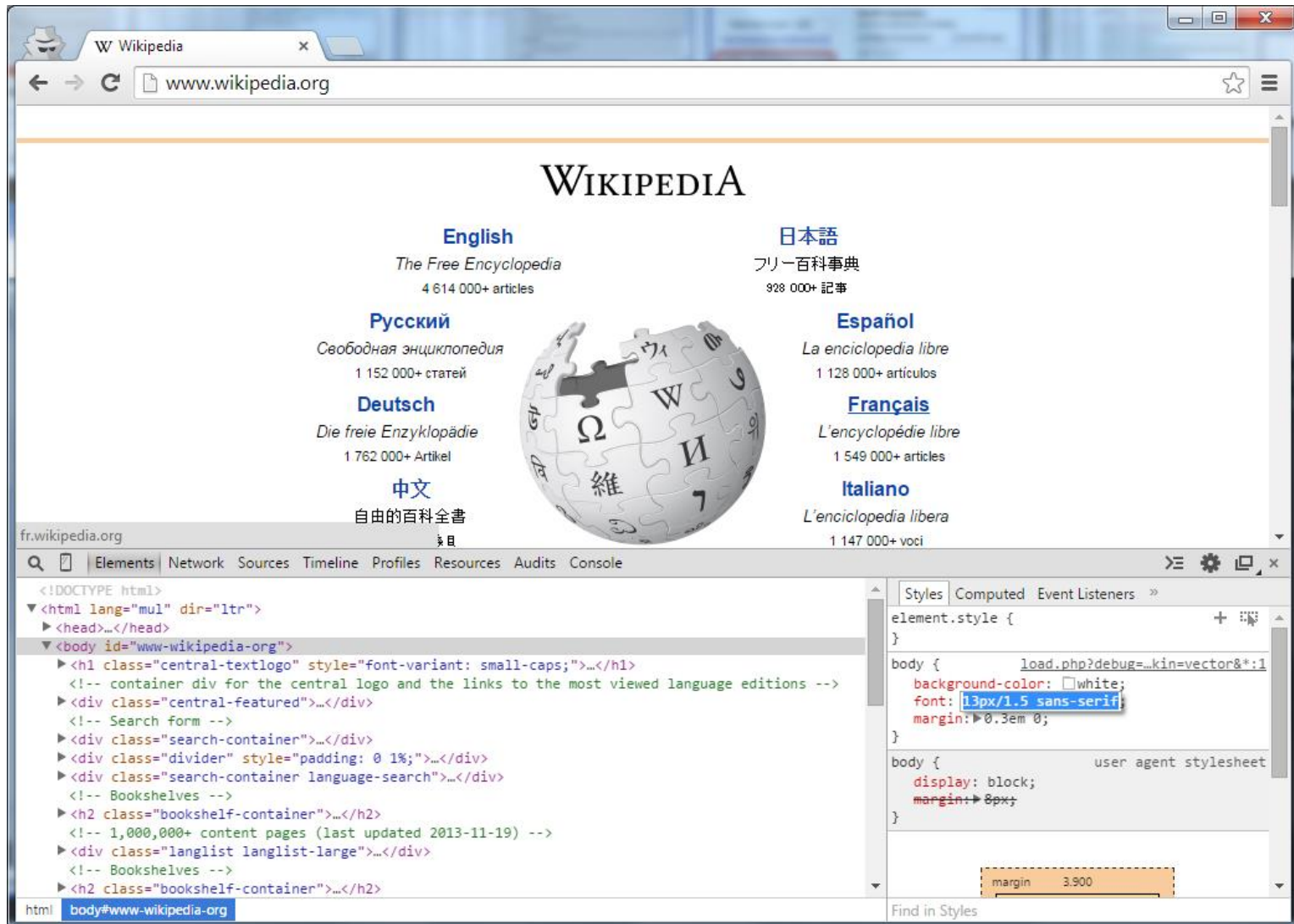
Инструменти

Fiddler



Инструменты

Chrome Developer Tools - F12



Инструменти

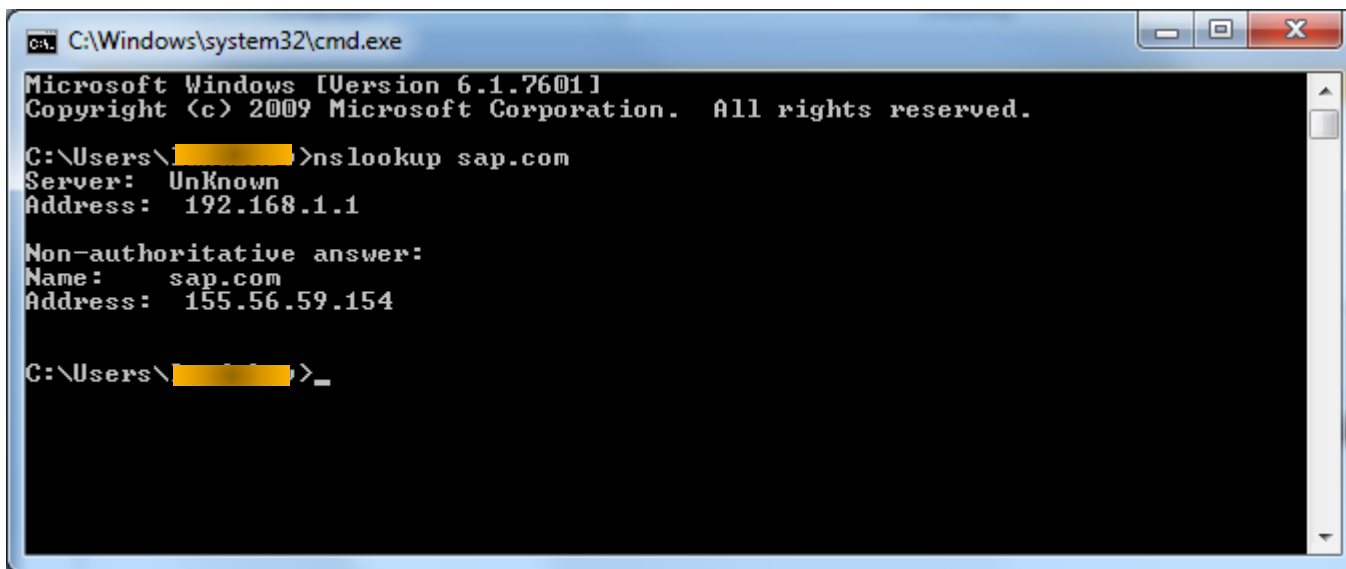
Command Line Tools

nslookup – за тестване на DNS. Преобразува име към IP адрес.

ping – проверява дали даден ресурс е наличен.

tracert – показва през какви сървъри минава даде конекция, докато стигне дестинацията.

netstat – показва какви конекции са отворени от и към дадена система.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\>nslookup sap.com
Server: UnKnown
Address: 192.168.1.1

Non-authoritative answer:
Name:    sap.com
Address: 155.56.59.154

C:\Users\>
```



Благодаря за вниманието!

За контакти:
Владислав Илиев
e-mail: vladislav.iliev@sap.com

ИСПОЛЗВАНА ЛИТЕРАТУРА

- ❖ <http://www.w3.org/Protocols/rfc2616/rfc2616.html>
- ❖ <http://www.f5.com/flash/wbt/http-basics-i/player.html>
- ❖ <http://www.f5.com/flash/wbt/http-basics-ii/player.html>
- ❖ https://en.wikipedia.org/wiki/Claude_Chappe
- ❖ <http://www.walthowe.com/navnet/history.html>
- ❖ <https://en.wikipedia.org/wiki/ARPANET>
- ❖ <http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/>
- ❖ <http://blog.trustiv.co.uk/2013/04/yet-another-fork-road>
- ❖ <http://cse.unl.edu/~ylu/csce855/notes/DNS.ppt>