Socket Programming

Sockets Programming

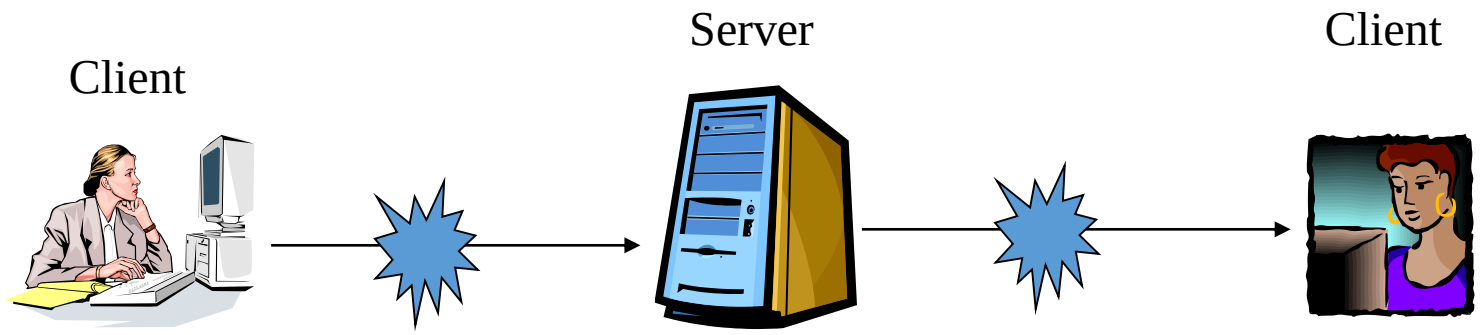
- Client-Server изчисления
- Какво са sockets?
- Sockets Programming in Java
- Пример

Client/Server изчисления

- Общ сценарий:
- Хост компютрите(clients, typically desk top computers) си взаимодействат с потребителите, с цел :
  - Събира входни данни от потребителите
  - Предоставя информация за потребителите
- Друг тип хост компютъри(сървъри) се използват за управление и обработка на големи данни
- Пример е Web: Client (Browser) & Server (HTTP server)

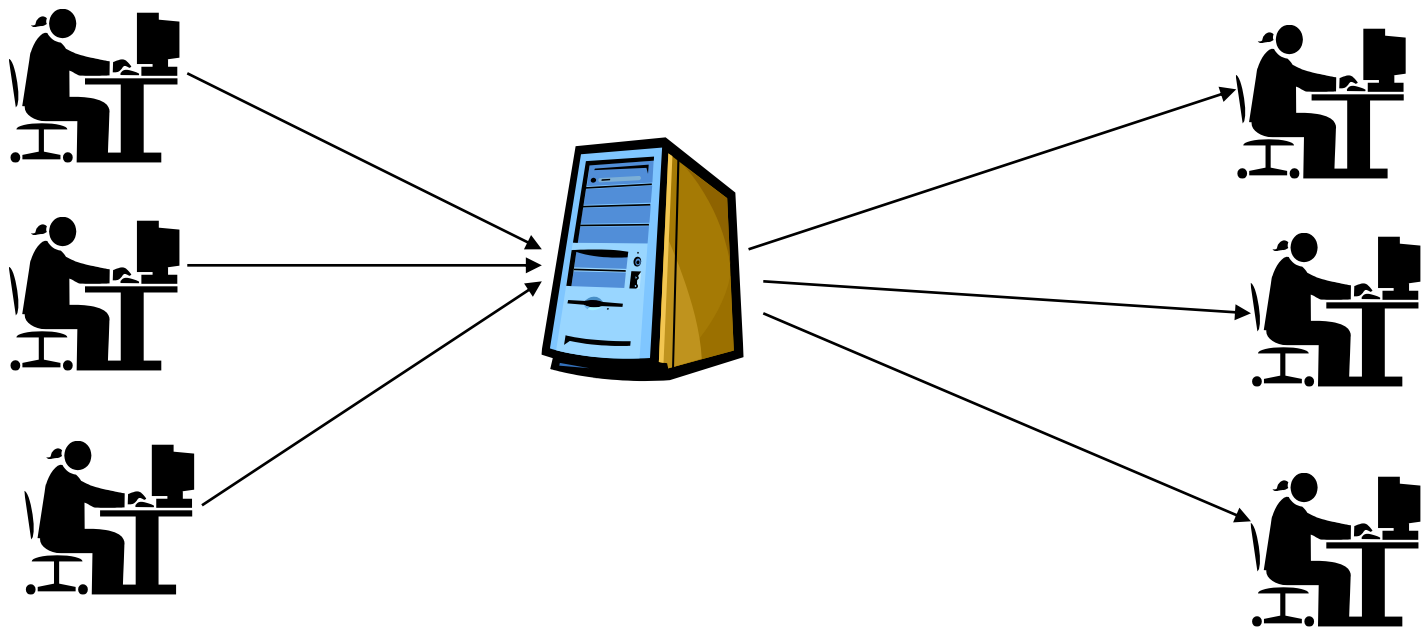
Client/Server Computing

- Other examples:
  - E-mail



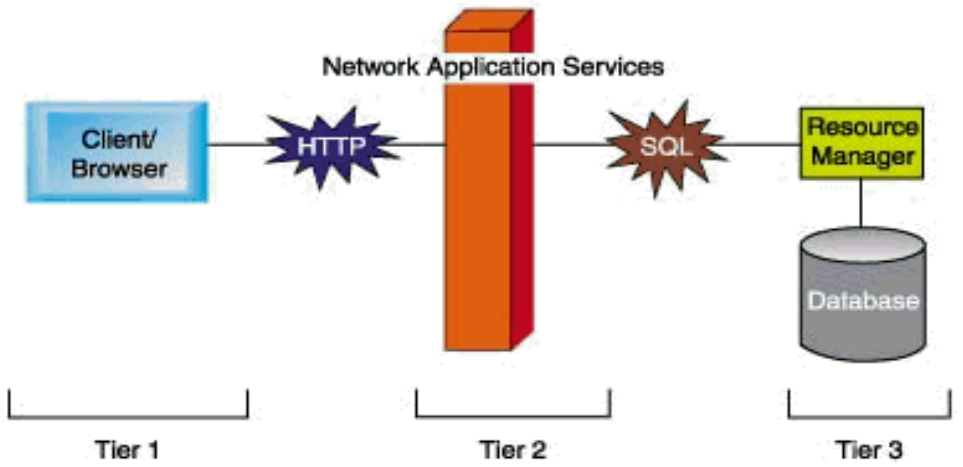
Client/Server Computing

- Other examples:
  - Chatroom



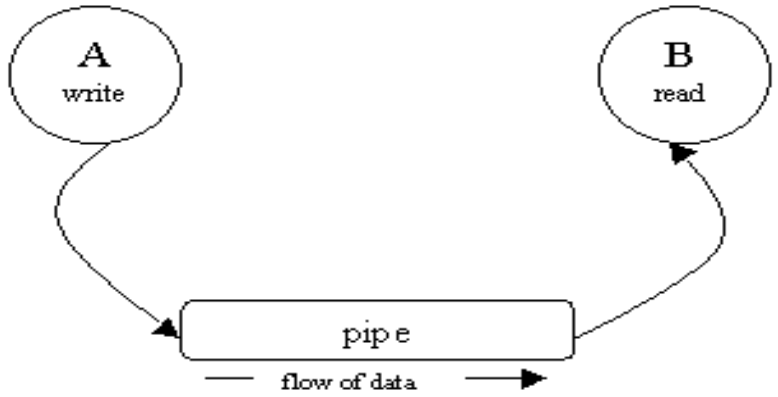
Tiered Client/Server Architecture

- 1-tier: single program
- 2-tier: client/server (e.g. the Web)
- 3-tier: application logic and databases on different servers (e.g. the Web with CGI and databases)



Комуникация клиент-сървър

- Два процеса от една машина могат да комуникират помежду си чрез pipe



- Pipe - това е pseudo-file, който свързва 2 процеса заедно

Комуникация клиент-сървър

- Два несвързани процеса могат да комуникират чрез файлове (process A записва във файл, а process B чете от него)

- Как ще изглежда ситуацията, когато два процеса, които са стартирани върху различни машини трябва да комуникират? РЕШЕНИЕ: Berkeley sockets.

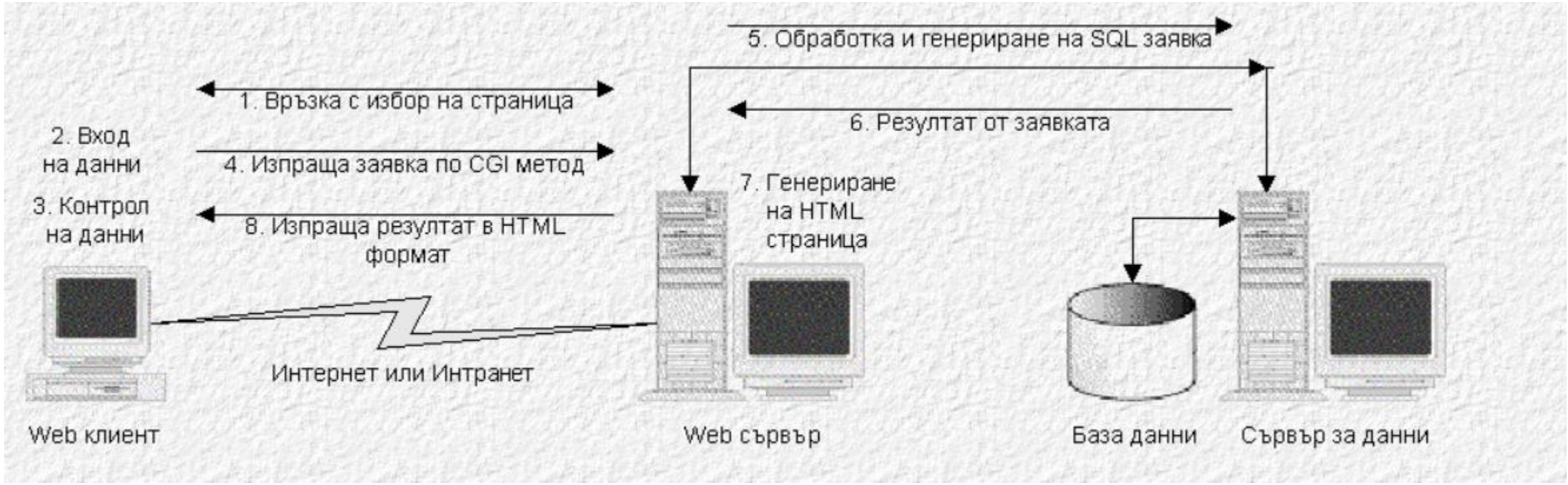
Какво представлява сокет

- Сокет – крайната точка на комуникацията
- Или: интерфейса между потребителя и мрежата
- Преди да се предават данни двата сокета трябва да се свържат (case of TCP)
- Видове връзки:
  - TCP, UDP, Multicast
- Типове сокети:
  - SOCK_STREAM, SOCK_DGRAM, SOCK_RAW

Класическият Клиент-Сървър има относително проста схема на функциониране. Клиентът управлява и извършва по-голямата част от обработките, с помощта на едно или повече приложения. Обработката може да включва:

- управление на графичен интерфейс;
- управление на взаимодействията с потребителя;
- изчисления и локална преработка на информацията;
- визуализация на данните;
- други процедури.

Универсалният Клиент-Сървър е основан на Web технология.



Универсалният Клиент-Сървър

Действията на едно приложение в средата на универсален клиент-сървър се състоят от последователност от операции:

Установява се връзка между HTTP сървъра и Web клиента и се извлича съответната страница;

Данните се въвеждат в HTML формуляри, изобразени от Web браузър;

Скриптов език контролира на място въведената информация *(Javascript, VBScript)*;

Заявката се изпраща към HTTP сървър по CGI метод;

Обработват се данните от заявката в сървъра и се генерират SQL заявки;

Достъп до данните посредством сървъра на БД *(SQL сървър, ...)*;

Генерира се HTML страница с получените данни. Страниците се генерират от сървър за да бъдат интерпретирани от клиент;

Изпращат се резултатите при клиента и се визуализират от браузър.

Стандарти във Web

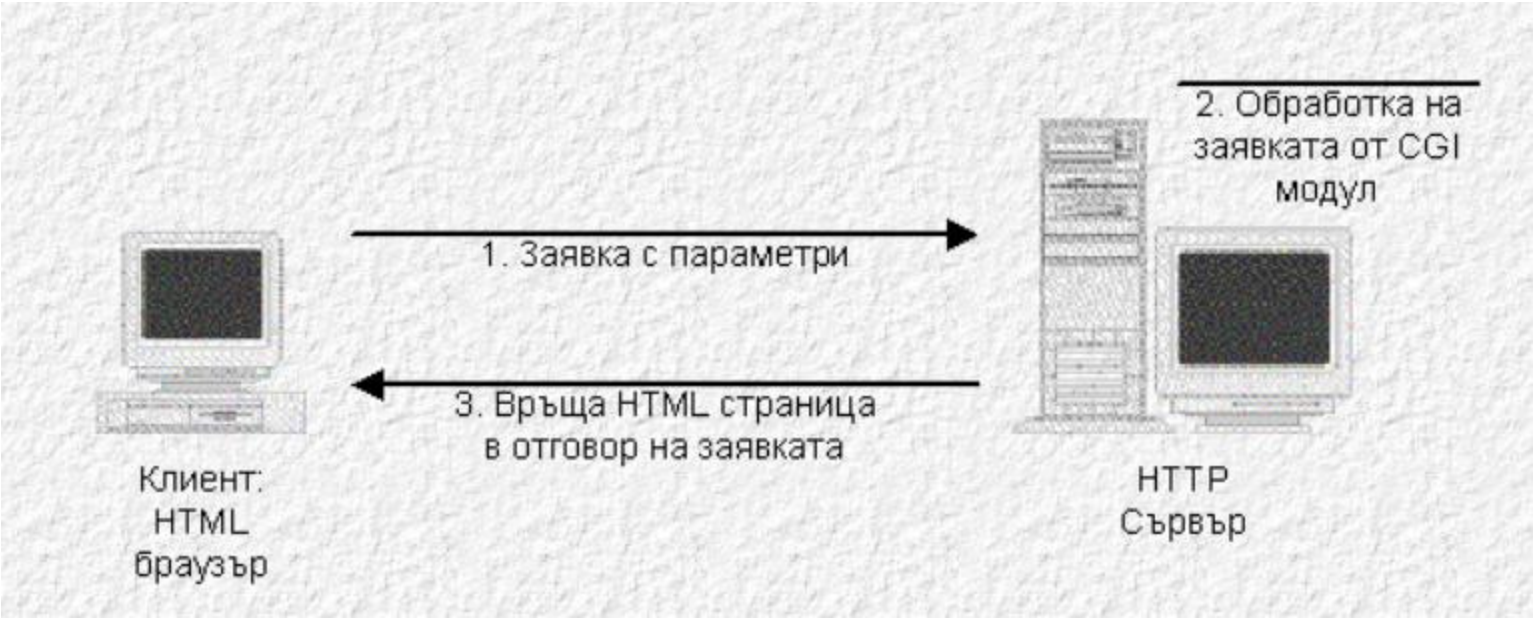
- Протокол HTTP;
- Скриптов език за описание на документи HTML;
- Норма-интерфейс за изпълнение на външни програми в HTTP сървър - CGI.

Протокол HTTP

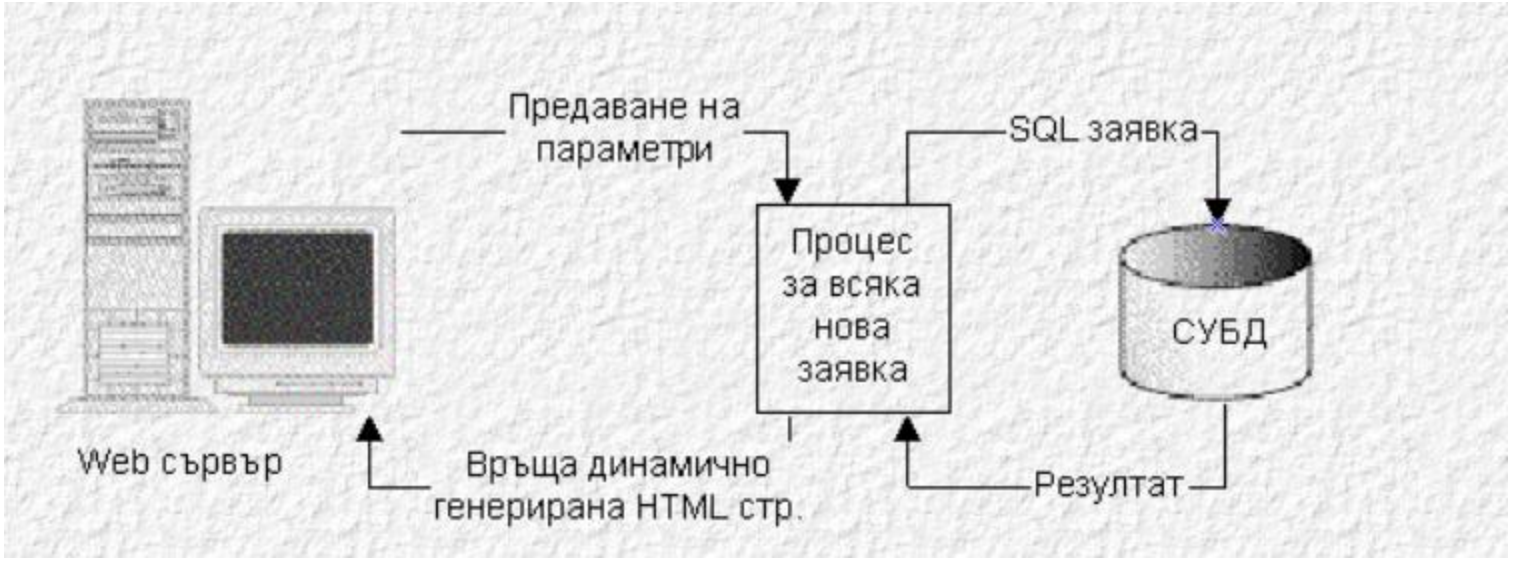
HTTP *(**H**yper **T**ext **T**ransfert **P**rotocol)* е протокол за обмен на документи между сървър и клиент, работещи под управление на Интернет протоколи (TCP/IP). В Интернет мрежа това позволява бързото разпространение на хипермедийни документи. Функционирането на протокола HTTP се свързва с проста схема на "въпрос-отговор". Клиентът излъчва заявка към сървъра, на което той отговаря

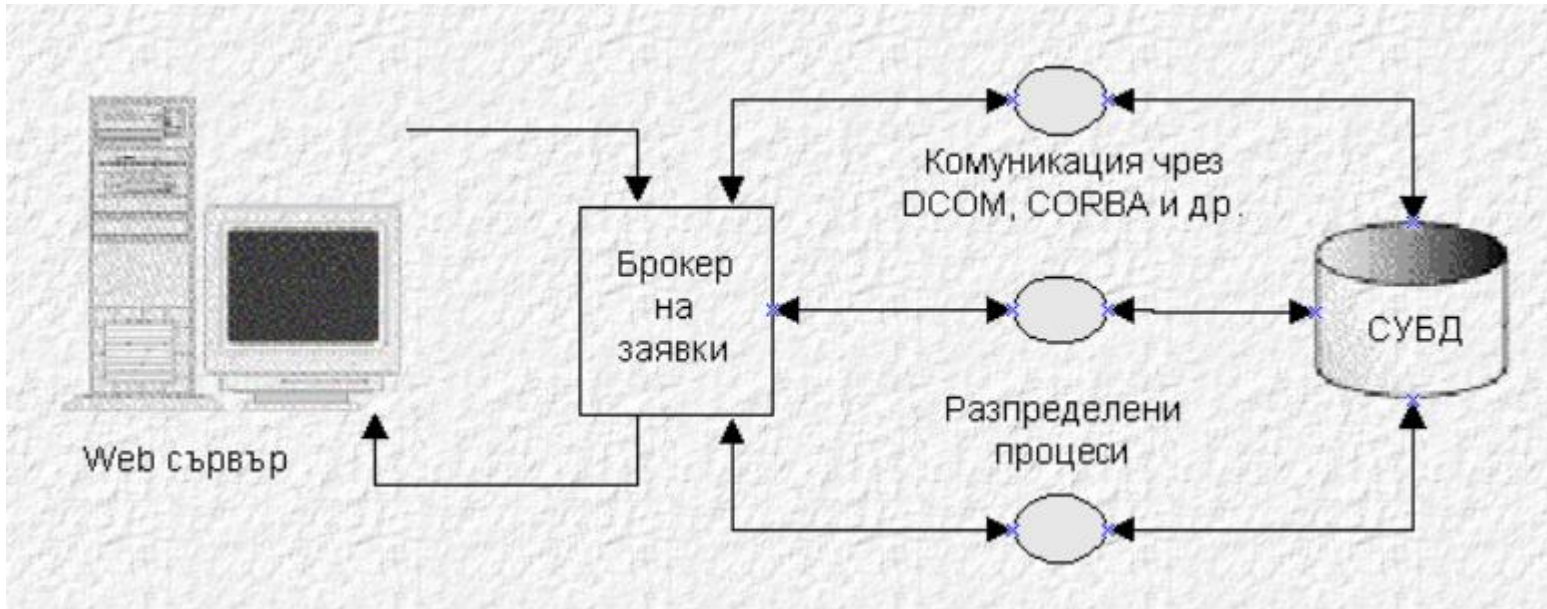
- ***Език HTML***
- **HTML** *(**H**yper**T**ext **M**arkup **L**angage)* е скриптов език за генериране на преносими документи по Интернет. Преносимостта се осигурява от възприетия стандарт за езика (HTML 3.2, 4.0, 4.01, DHTML) от всички компютърни платформи и производители на Web браузъри *(Виж раздела "Мултимедийни документи")*. Езиковите средства са съставени от мета-дескриптори за описание на различни текстови фрагменти и параграфи, вмъкване в текста на графични, звукови и видео компоненти, както и средства за описание на връзки към други документи. Записани в текстови файлове с разширения *.htm* или *.html* докуметите се предават на клиента, който ги интерпретира и изобразява в полето на браузъра. Чистото съдържание на един HTML файл може да се разглежда с произволен текстов редактор (Notepad в Windows, vi в Unix). Езикът HTML в този смисъл не е средство за изграждане на интерфейс, а само средство за стилово описание на документи. Динамични и диалогови елементи се изграждат с допълнителни средства.
- ***CGI интерфейс***
- **CGI** *(**C**ommon **G**ateway **I**nterface)* е програмен интерфейс, позволяващ изпълнението на външни програми от един HTTP сървър. Това е стандарт от ниско програмно ниво, съвместим с почти всички видове сървъри от тип HTTP. Предназначението му е да позволи обработката на клиентски заявки от сървъра. CGI програмите се изпълняват на сървъра с параметри зададени от клиент. Програмният им код се реализира от *shell* скриптове, код на езика PERL, C или C++. Предпочитан в тази област е езикът PERL. Езиковите му средства са максимално адаптирани за нуждите и функциите на този род приложения. Със средствата на CGI програма по заявка на клиента в сървъра се генерира HTML документ, който му се изпраща обратно за визуализация от неговия браузър

CGI интерфейс



CGI интерфейса е станал своеобразен стандарт за всички производители на Web сървъри

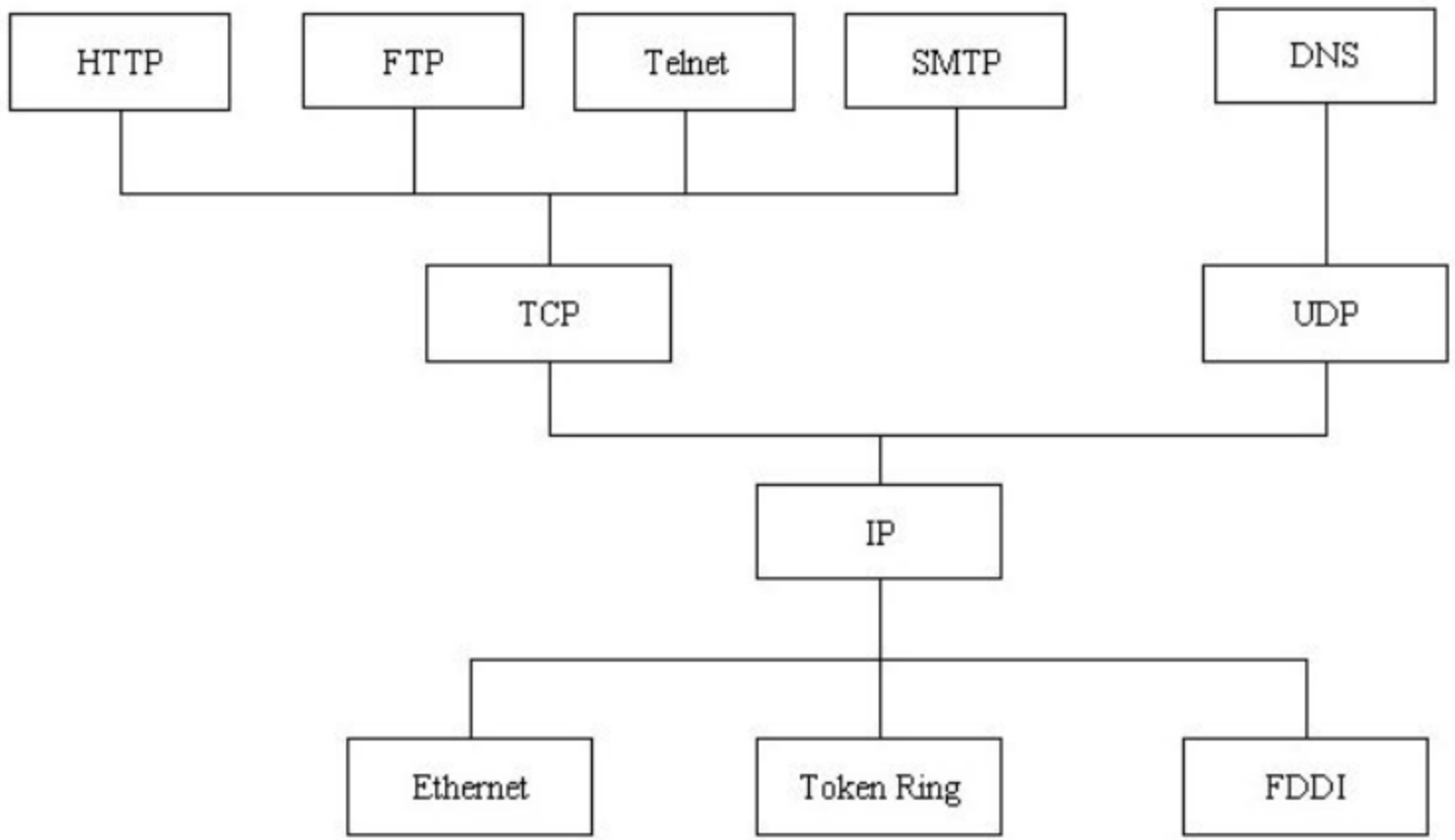



Комуникация чрез
DCOM, CORBA и др.
Брокер
на
заявки
СУБД
Разпределени
процеси
Web сървър

Как работи Интернет. Основи на TCP/IP мрежите

#	ниво	описание и протоколи
7	Application (приложно ниво)	Осигурява на приложните програмисти интерфейс към мрежовата инфраструктура, осигурена от по-долните слоеве. Протоколите от това ниво задават форматите и правилата за обмяна на данни между комуникиращите приложения. Типични протоколи на това ниво са: HTTP, SMTP, POP3, FTP, SNMP, FTP, DNS, NFS и др.
6	Presentation (представително ниво)	Осигурява общ формат, унифицирано канонично представяне на пренасяните данни, което е еднакво за всички платформи и е разбираемо за по-долните слоеве. Типични протоколи или по-точно схеми за унифицирано представяне на данни от това ниво са XDR, ASN.1, SMB, AFP.
5	Session (сесийно ниво)	Организира и синхронизира прозрачната обмяна на информация между два процеса в операционните системи на комуникиращите машини. Типични протоколи от това ниво са: RPC, NetBIOS, CCITT X.225 и др.
4	Transport (транспортно ниво)	Осигурява поддръжката на комуникационни канали за данни между две машини. Позволява пренасяне не само на отделни пакети, но и на по-големи обеми данни. Осигурява прозрачност и надеждност на преноса на данни. Грижи се за започване, поддръжка и прекратяване на комуникацията между машините участнички. Типични протоколи на това ниво са: TCP, UDP, RTP, SPX, ATP.
3	Network (мрежово ниво)	Осигурява пренасяне на единици информация (пакети) между две машини в дадена мрежа, всяка от които има уникален мрежов адрес. Не е задължително двете машини да са пряко свързани една с друга и затова мрежовото ниво осигурява маршрутизиране на пакетите от една машина към друга с цел достигане на крайната цел. Типични протоколи на това ниво са IP, IPv6, ICMP, IGMP, X.25, IPX и др.
2	Data Link (свързващо ниво)	Осигурява директно пренасяне на информация между две мрежови комуникационни устройства (например две мрежови карти или два модема), управлява физическото ниво и се грижи за корекция на грешки възникнали в него. Типични протоколи са Ethernet, Token ring, PPP, Frame relay, ISDN и др.
1	Physical (физическо ниво)	Осигурява физическото пренасяне на информацията. Може да се реализира от радиовълни, оптични кабели, лазери и др.

#	ниво	описание и протоколи
4	Application (приложно ниво)	Осигурява на приложните програмисти интерфейс към мрежовата инфраструктура, осигурена от транспортния и Интернет слоевете. Протоколите от това ниво задават форматите и правилата за обмяна на данни между комуникиращите си приложения. Типични протоколи на това ниво са: HTTP, SMTP, POP3, FTP, SNMP, FTP, DNS, NFS и др.
3	Transport (транспортно ниво)	Осигурява поддръжката на комуникационни канали за данни между две приложения (евентуално на отдалечени машини). Позволява пренасяне не само на отделни пакети, но и на по-големи обеми данни. Осигурява прозрачност и надеждност на преноса. Грижи се за започване, поддръжка и прекратяване на комуникацията между процесите участници. В това ниво се използват само два протокола: TCP и UDP.
2	Internet (Интернет ниво)	Осигурява пренасяне на единици информация (пакети) между две машини в дадена мрежа, всяка от които има уникален адрес (IP адрес). Не е задължително двете машини да са пряко свързани една с друга и затова Интернет нивото осигурява маршрутизиране на пакетите от една машина към друга с цел достигане на крайната цел. На това ниво работят протоколите IP, IPv6, ICMP и IGMP.
1	Link (свързващо ниво)	Осигурява директно пренасяне на информация между две мрежови комуникационни устройства (например две мрежови карти или два модема). Типични протоколи са Ethernet, Token ring, PPP, Frame relay, ISDN и др.

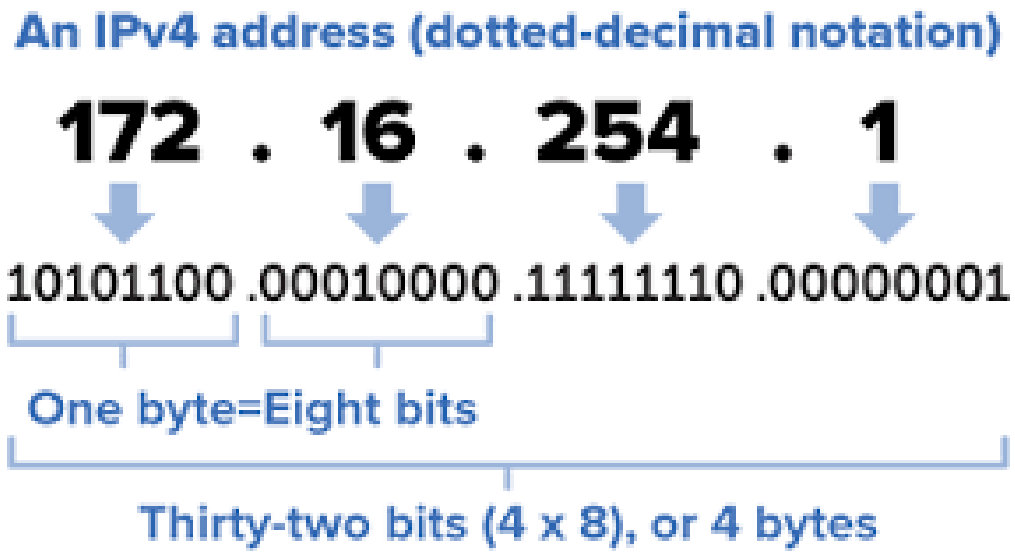


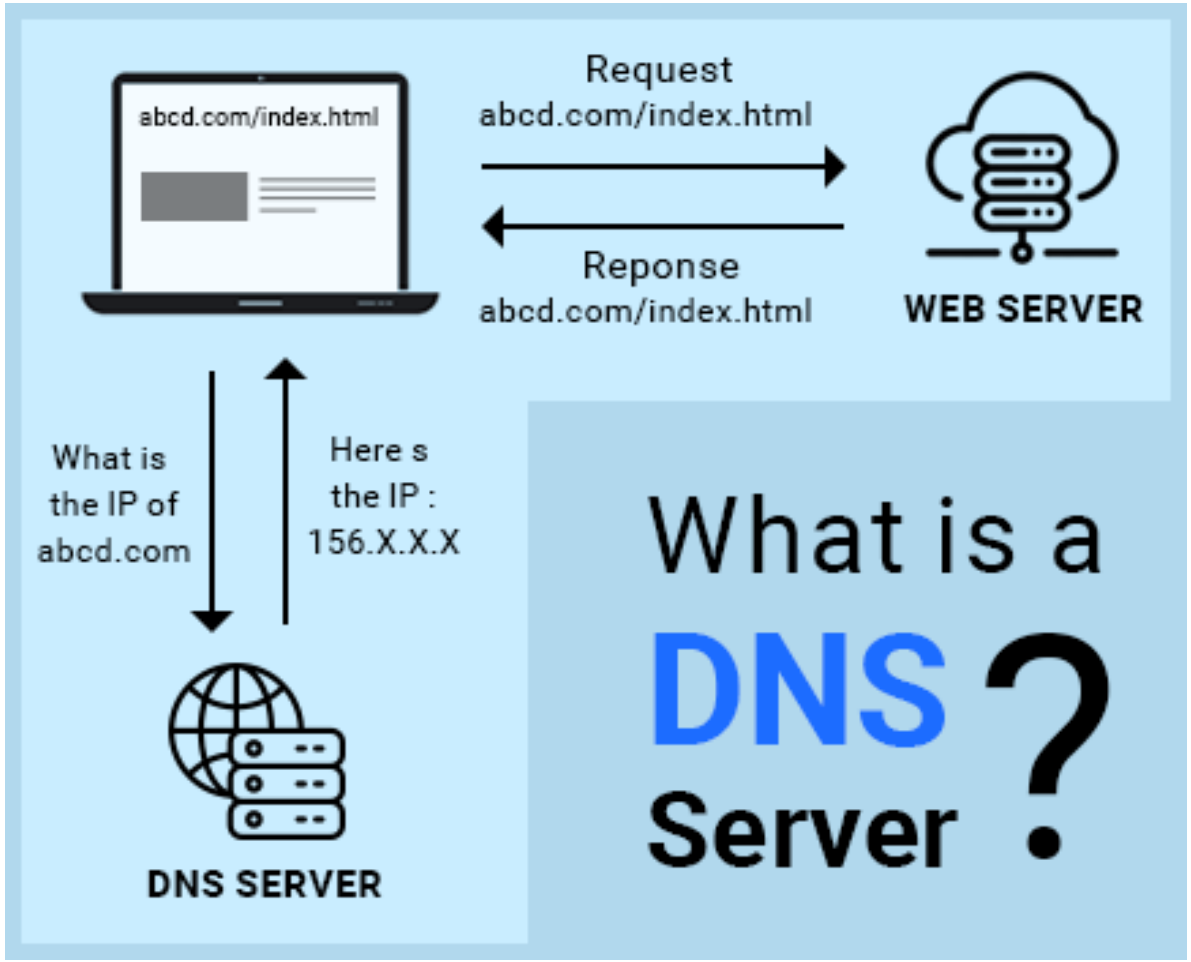
Примери на протоколи от приложния слой

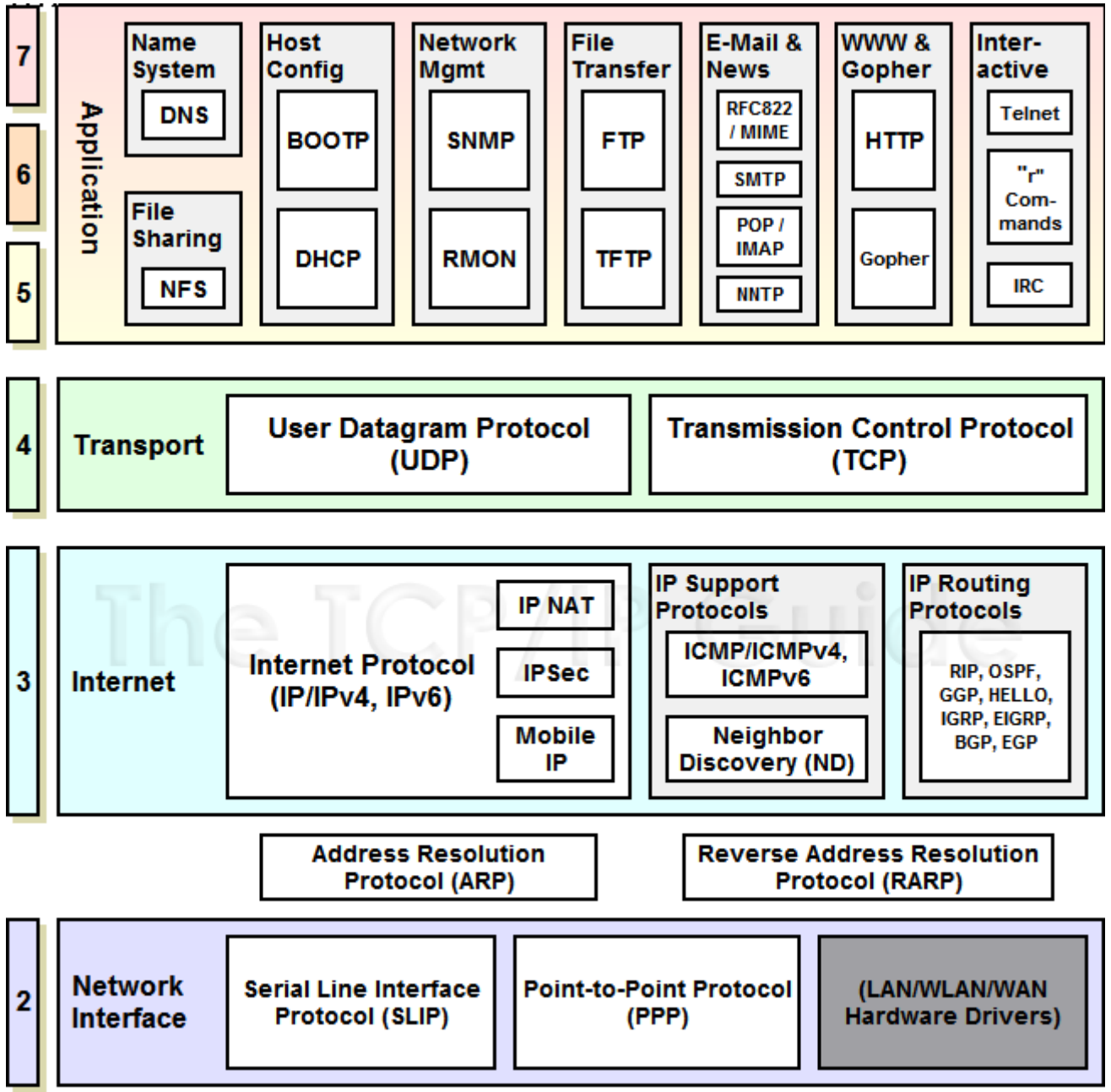
IP (Internet protocol) - комутация на пакети(datagrames). Не съществува стандартно Java API позволяващо директна манипулация на протокола IP

UDP (User Datagram Protocol) - предаване на съобщения, бърза връзка, няма връзка между източник и получател, няма гаранция за получаване

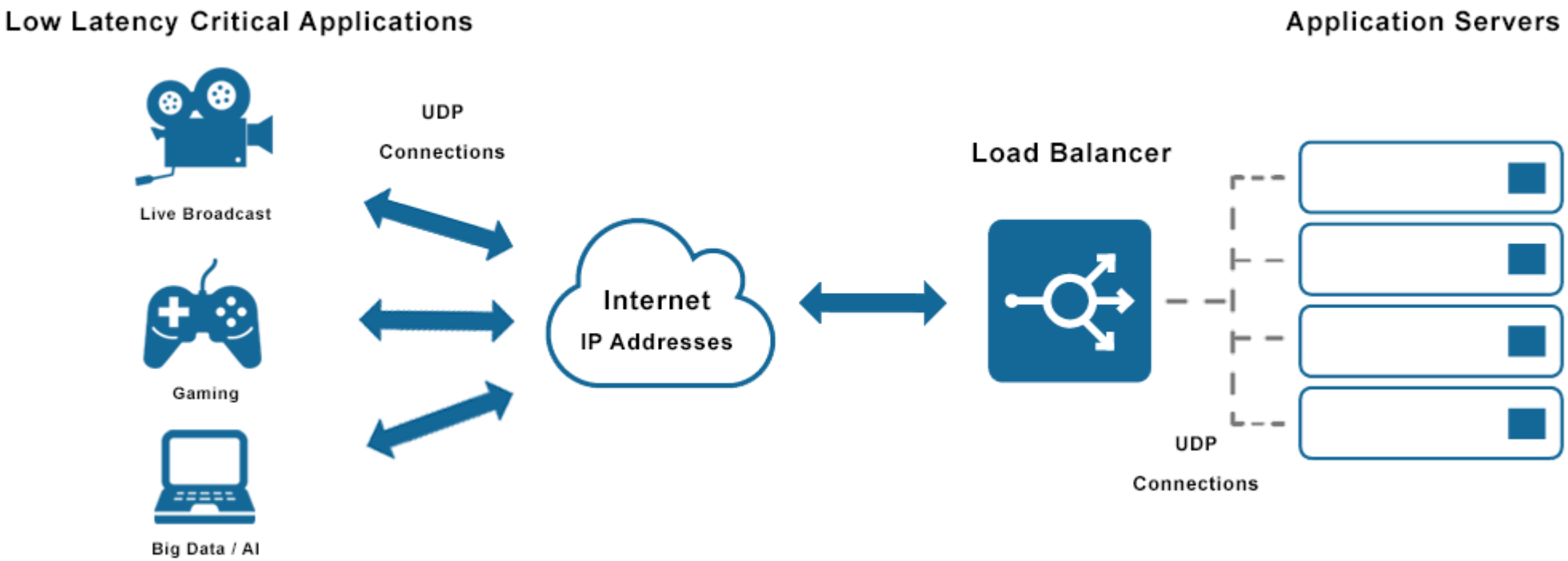
TCP (Transmission Control Protocol) - трансмисия на поток, връзка между сървър и получател, гаранция за получаване.

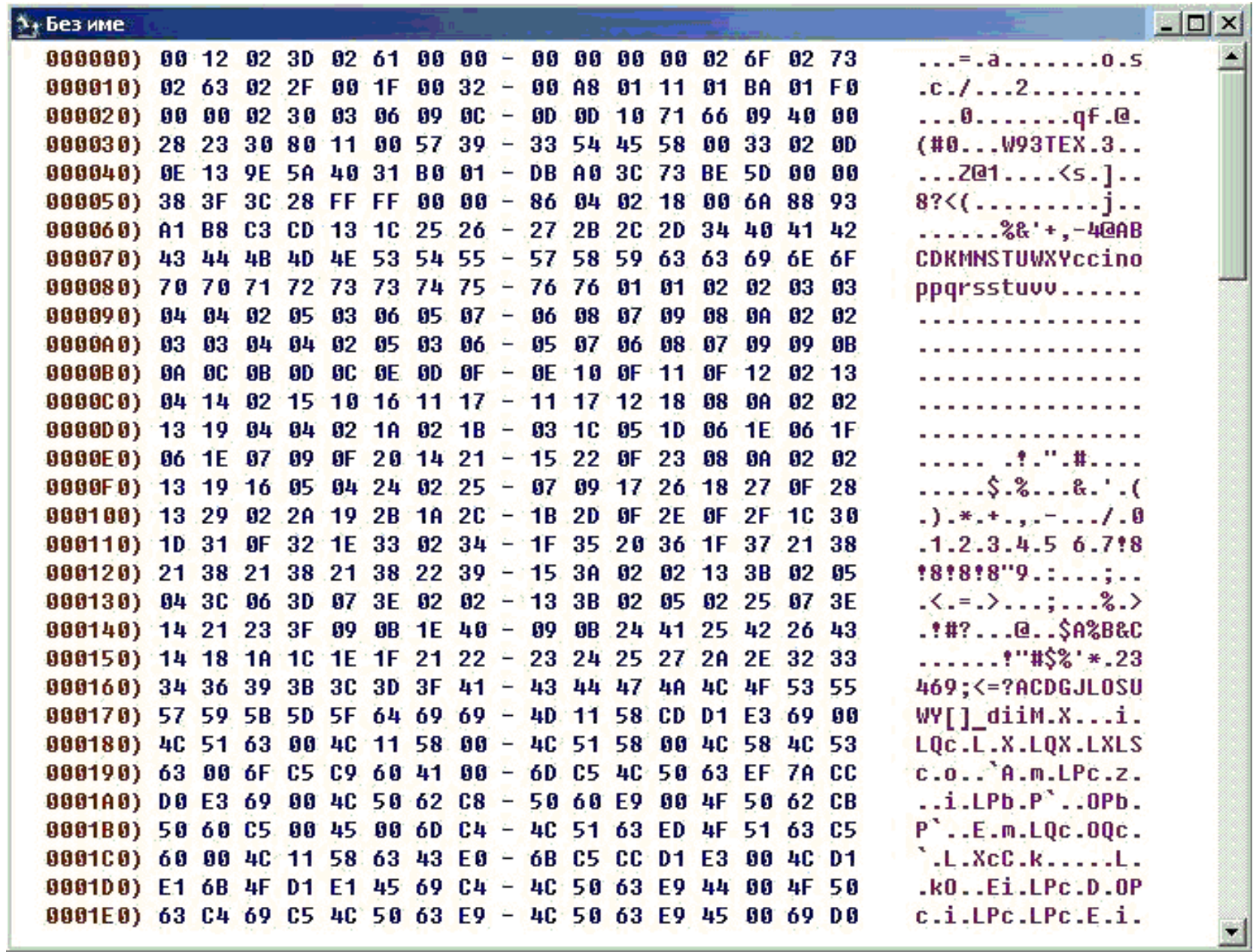

An IPv4 address (dotted-decimal notation)
172 . 16 . 254 . 1
10101100 .00010000 .11111110 .00000001
One byte=Eight bits
Thirty-two bits (4 x 8), or 4 bytes

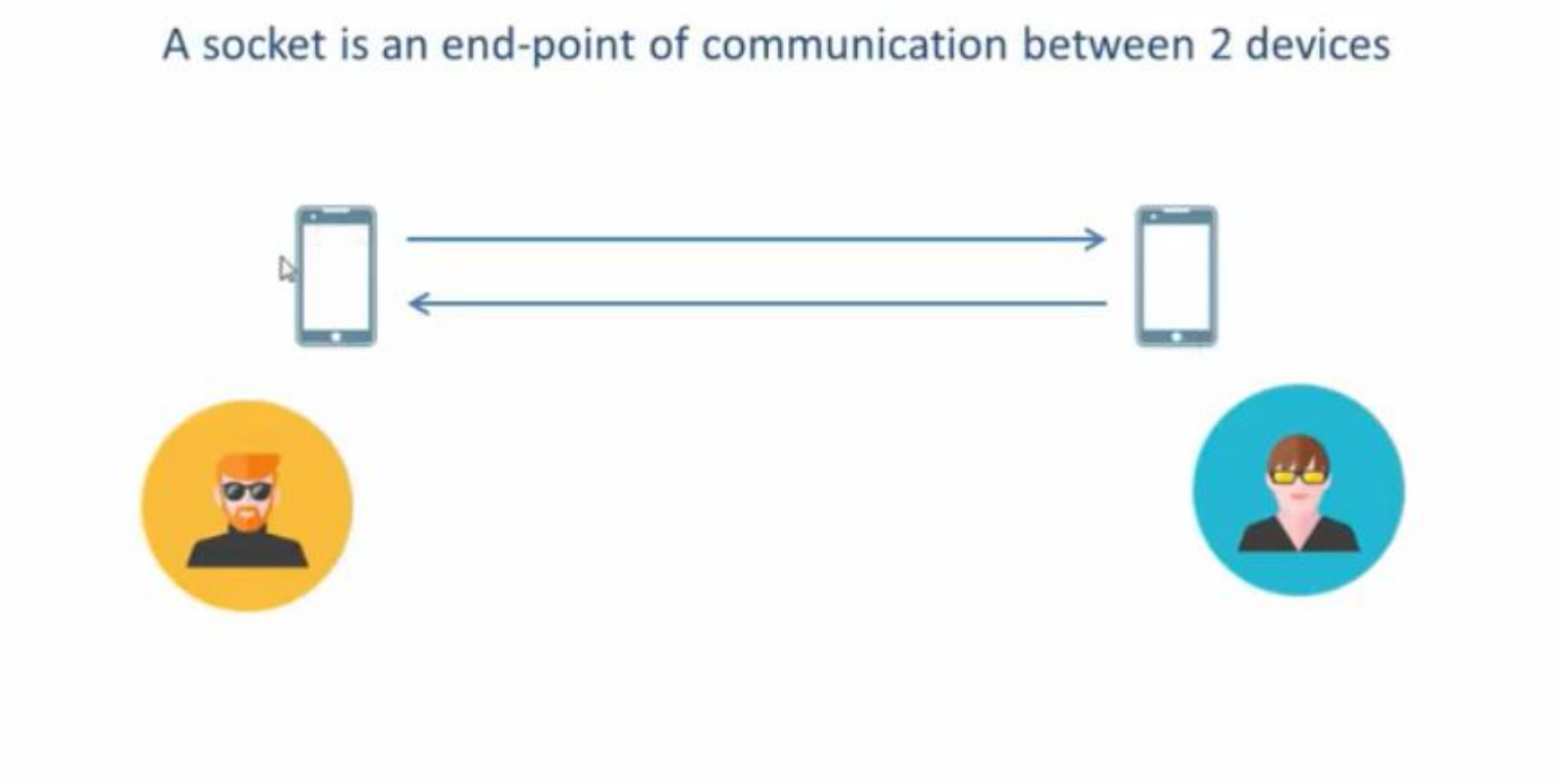

abcd.com/index.html
Request
abcd.com/index.html
Reponse
abcd.com/index.html
WEB SERVER
What is
the IP of
abcd.com
Here s
the IP :
156.X.X.X
DNS SERVER
What is a
DNS
Server ?


7
6
5
Application
Name System
DNS
File Sharing
NFS
Host Config
BOOTP
DHCP
Network Mgmt
SNMP
RMON
File Transfer
FTP
TFTP
E-Mail & News
RFC822 / MIME
SMTP
POP / IMAP
NNTP
WWW & Gopher
HTTP
Gopher
Inter-active
Telnet
"r" Com-mands
IRC
4
Transport
User Datagram Protocol (UDP)
Transmission Control Protocol (TCP)
3
Internet
Internet Protocol (IP/IPv4, IPv6)
IP NAT
IPSec
Mobile IP
IP Support Protocols
ICMP/ICMPv4, ICMPv6
Neighbor Discovery (ND)
IP Routing Protocols
RIP, OSPF, GGP, HELLO, IGRP, EIGRP, BGP, EGP
Address Resolution Protocol (ARP)
Reverse Address Resolution Protocol (RARP)
2
Network Interface
Serial Line Interface Protocol (SLIP)
Point-to-Point Protocol (PPP)
(LAN/WLAN/WAN Hardware Drivers)

Протоколът UDP



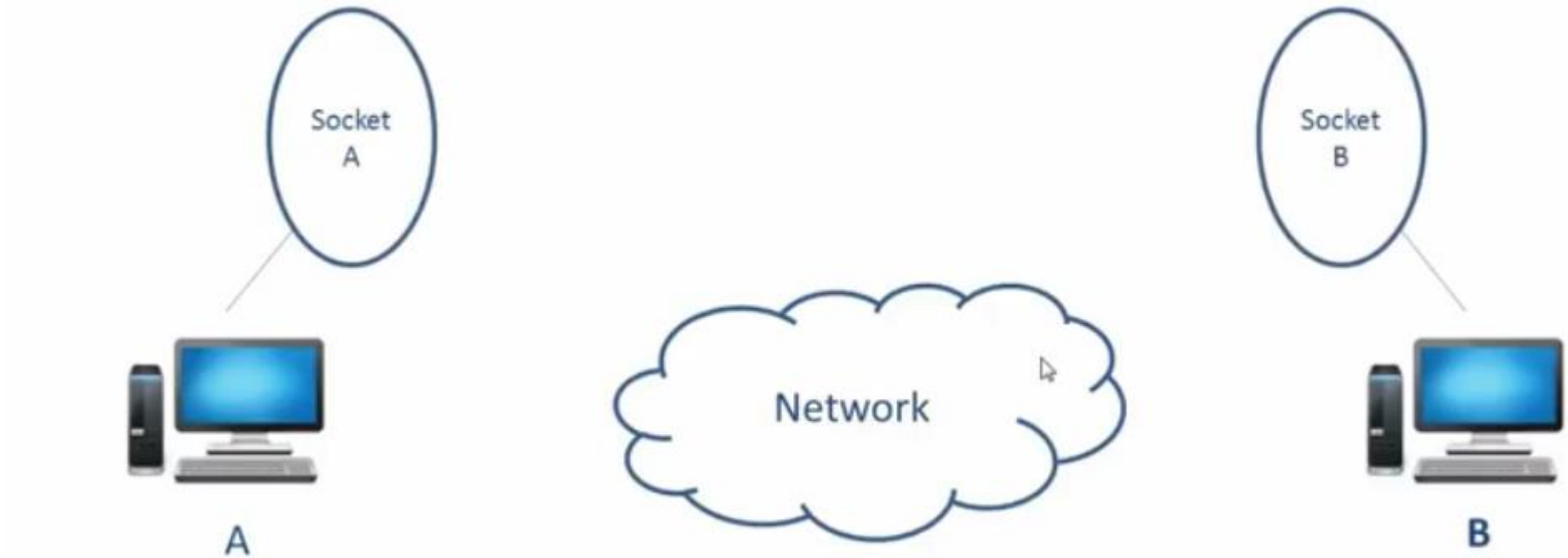

Без име
000000) 00 12 02 3D 02 61 00 00 - 00 00 00 00 02 6F 02 73 ...=.a.....o.s
000010) 02 63 02 2F 00 1F 00 32 - 00 A8 01 11 01 BA 01 F0 .c./...2........
000020) 00 00 02 30 03 06 09 0C - 0D 0D 10 71 66 09 40 00 ...0.....qf.@.
000030) 28 23 30 80 11 00 57 39 - 33 54 45 58 00 33 02 0D (#0...W93TEX.3..
000040) 0E 13 9E 5A 40 31 B0 01 - DB A0 3C 73 BE 5D 00 00 ...Z@1....<s.]..
000050) 38 3F 3C 28 FF FF 00 00 - 86 04 02 18 00 6A 88 93 8?<(.........j..
000060) A1 B8 C3 CD 13 1C 25 26 - 27 2B 2C 2D 34 40 41 42 ......%&'+,-4@AB
000070) 43 44 4B 4D 4E 53 54 55 - 57 58 59 63 63 69 6E 6F CDKMNSTUWXYccino
000080) 70 70 71 72 73 73 74 75 - 76 76 01 01 02 02 03 03 ppqrsstuvv......
000090) 04 04 02 05 03 06 05 07 - 06 08 07 09 08 0A 02 02 ....
0000A0) 03 03 04 04 02 05 03 06 - 05 07 06 08 07 09 09 0B ....
0000B0) 0A 0C 0B 0D 0C 0E 0D 0F - 0E 10 0F 11 0F 12 02 13 ....
0000C0) 04 14 02 15 10 16 11 17 - 11 17 12 18 08 0A 02 02 ....
0000D0) 13 19 04 04 02 1A 02 1B - 03 1C 05 1D 06 1E 06 1F ....
0000E0) 06 1E 07 09 0F 20 14 21 - 15 22 0F 23 08 0A 02 02 ..... .!."."#....
0000F0) 13 19 16 05 04 24 02 25 - 07 09 17 26 18 27 0F 28 .....$.%...&.'.(
000100) 13 29 02 2A 19 2B 1A 2C - 1B 2D 0F 2E 0F 2F 1C 30 .).*.+.,.-.../.0
000110) 1D 31 0F 32 1E 33 02 34 - 1F 35 20 36 1F 37 21 38 .1.2.3.4.5 6.7!8
000120) 21 38 21 38 21 38 22 39 - 15 3A 02 02 13 3B 02 05 !8!8!8"9.:...;..
000130) 04 3C 06 3D 07 3E 02 02 - 13 3B 02 05 02 25 07 3E .<.=.>...;...%.>
000140) 14 21 23 3F 09 0B 1E 40 - 09 0B 24 41 25 42 26 43 .!#?...@..$A%B&C
000150) 14 18 1A 1C 1E 1F 21 22 - 23 24 25 27 2A 2E 32 33 ......!"#$%'*.23
000160) 34 36 39 3B 3C 3D 3F 41 - 43 44 47 4A 4C 4F 53 55 469;<=?ACDGJLOSU
000170) 57 59 5B 5D 5F 64 69 69 - 4D 11 58 CD D1 E3 69 00 WY[]_diiM.X...i.
000180) 4C 51 63 00 4C 11 58 00 - 4C 51 58 00 4C 58 4C 53 LQc.L.X.LQX.LXLS
000190) 63 00 6F C5 C9 60 41 00 - 6D C5 4C 50 63 EF 7A CC c.o..`A.m.LPc.z.
0001A0) D0 E3 69 00 4C 50 62 C8 - 50 60 E9 00 4F 50 62 CB ..i.LPb.P`..OPb.
0001B0) 50 60 C5 00 45 00 6D C4 - 4C 51 63 ED 4F 51 63 C5 P`..E.m.LQc.OQc.
0001C0) 60 00 4C 11 58 63 43 E0 - 6B C5 CC D1 E3 00 4C D1 `.L.XcC.k.....L.
0001D0) E1 6B 4F D1 E1 45 69 C4 - 4C 50 63 E9 44 00 4F 50 .kO..Ei.LPc.D.OP
0001E0) 63 C4 69 C5 4C 50 63 E9 - 4C 50 63 E9 45 00 69 D0 c.i.LPc.LPc.E.i.

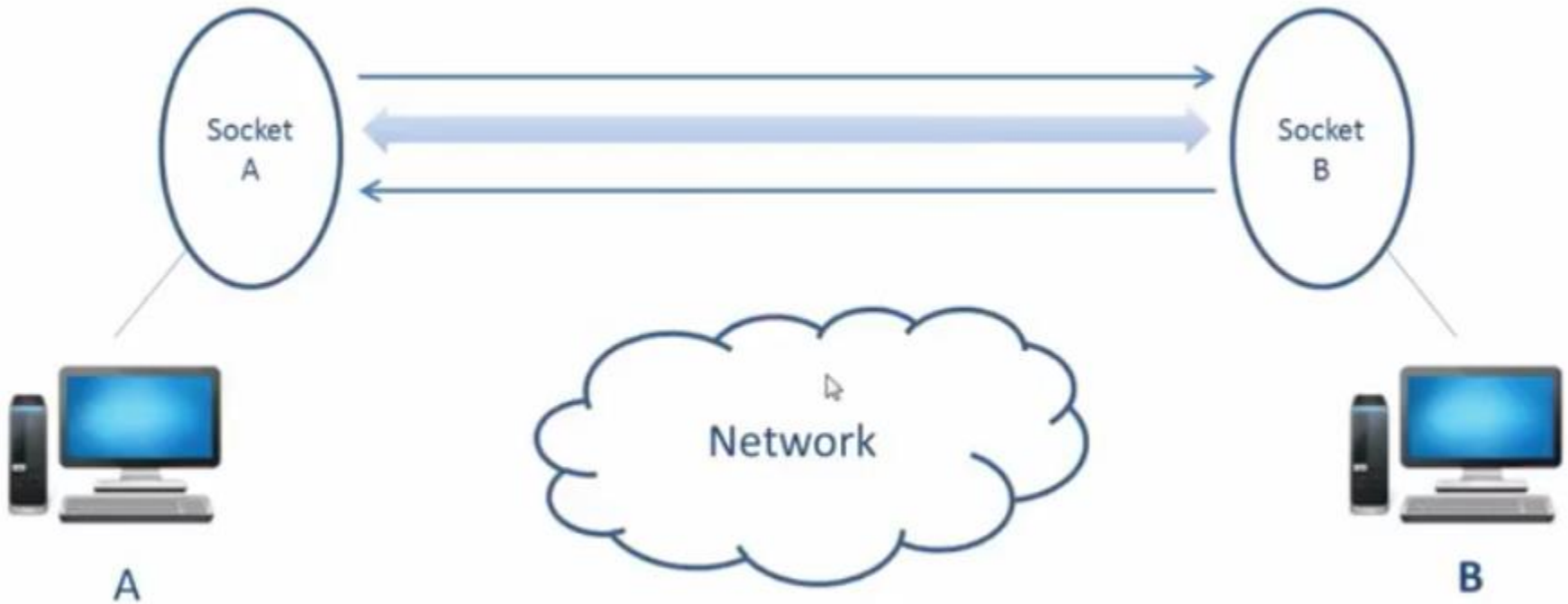


- Internet Explorer = 212.50.1.81:1033 ↔ 212.50.1.1:80 = Apache Web Server
- Internet Explorer = 212.50.1.81:1037 ↔ 212.50.1.1:80 = Apache Web Server
- Outlook Express = 212.50.1.81:1042 ↔192.92.129.4:110 = Microsoft Exchange POP3 Server

Сървъри и клиенти

- **Клиентските приложения** (наричани още **клиенти**) се свързват към сървърските като отварят сокет връзка към тях. За целта те предварително знаят техните IP адреси и портове.
- **Сървърските приложения** (наричани още **сървъри**) “слушат на определен порт” и чакат клиентско приложение да се свърже към тях. При пристигане на заявка за връзка от някой клиент на порта, на който сървърът слуша, се създава сокет за връзка между клиента на неговия порт източник и сървъра на неговия порт получател.


Socket
A
Network
Socket
B
A
B


Socket
A
Socket
B
Network
A
B

Sockets

- Местоположението на съобщенията се задава в адреса на сокета
- Всеки адрес на сокета има комуникационен идентификатор:
  - Internet address
  - Port number
- Броят на портовете е целочислено число, което е различно за отделните сървиси на една и съща машина
- Портовете от 0 .. 1023 са резервирани;

Портове:

- някои “известни” портове:
  - 21: ftp
  - 23: telnet
  - 80: http
  - 161: snmp
- Ако сте в Linux, проверете /etc/services file за списък портове и сървиси, свързани с тези портове;

порт	протокол	услуга
21	FTP	Услуга за достъп до отдалечени файлове. Използва се от FTP клиенти (например Internet Explorer, GetRight, CuteFTP, wget)
25	SMTP	Услуга за изпращане на E-mail. Използва се от E-mail клиенти (например Outlook Express, Mozilla Mail, pine)
80	HTTP	Услуга за достъп до Web-ресурси. Използва се от Web-браузъри (например Internet Explorer, Mozilla, lynx)
110	POP3	Услуга за извличане на E-mail от пощенска кутия. Използва се от E-mail клиенти (например Outlook Express, Mozilla Mail, pine)

Which transport protocol (TCP v. UDP)

- TCP -- Transmission Control Protocol
- UDP -- User Datagram Protocol
- What should I use?
  - TCP is a ***reliable*** protocol, UDP is not
  - TCP is ***connection-oriented***, UDP is ***connectionless***
  - TCP incurs overheads, UDP incurs fewer overheads
  - UDP has a size limit of 64k, in TCP no limit

UDP Socket programming

- UDPе прост и ефективен, но не надежден
- Комуникацията се осъществява по symmetric manner: двата края изпращат и получават съобщения, следвайки протокол

UDP Socket Programming

- Докато не се създаде връзка, всяко съобщение трябва да съдържа в себе си и адреса на получателя.
- В Java, съобщенията са инстанции на класа `DatagramPacket`

The DatagramPacket class

- Constructors:
  - One for sending datagrams:

```
DatagramPacket{byte buffer[], int length, InetAddress,
  int port}
```

  - One for receiving datagrams:

```
DatagramPacket{byte buffer[], int length}
```

The DatagramPacket class

- The useful methods are the accessors:

```
InetAddress getAddress()
Int getPort()
Byte[] getData()
Int getLength()
```

Изпращане и получаване с Datagram packets: използвайки сокети

- Двата края трябва да създадат сокети за да коминикират
- Sockets ще бъдат „връзката” с хоста и портовете
- Получателят трябва да “подслушва” портовете на sender
- Sender може да използва приоритетни портове

The DatagramSocket class

- Constructors:
  - One for the sender (randomly chosen port number):

```
DatagramSocket() throws SocketException
```

  - One for the receiver (port needs to be specified):

```
DatagramSocket(int port) throws SocketException
```

The DatagramSocket class

- Sending and receiving messages:

```
void send(DatagramPacket packet) throws
  IOException
void receive(DatagramPacket packet) throws
  IOException
```

- Close the socket when you're done!

```
void close()
```

Receiving UDP packets

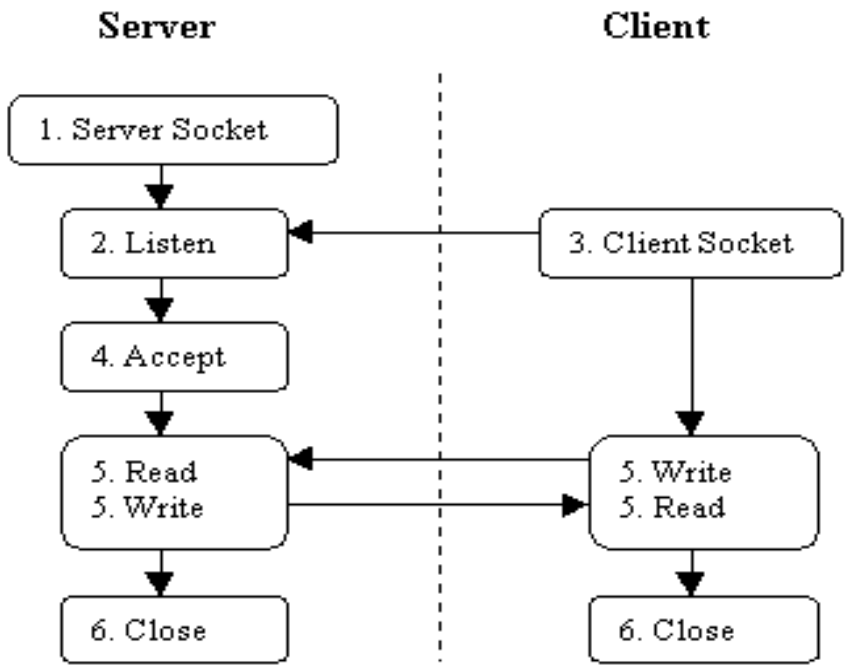
```
DatagramSocket socket = new DatagramSocket(port);
Byte buffer[] = new byte[65508];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
Socket.receive(packet);
InetAddress fromAddress = packet.getAddress();
int fromPort = packet.getPort();
int length = packet.getLength();
byte[] data = packet.getData();
socket.close();
```

Sending UDP Packets

```
DatagramSocket socket = new DatagramSocket();

DatagramPacket packet = new DatagramPacket(data, data.length,
  InetAddress.getByName("eureka.sce.carleton.ca"), 1728);

socket.send(packet);

socket.close();
```

TCP Socket Communication

- Sequence of steps normally taken to set up socket communication and exchange data between C/S



Пакет java.net

Interfaces	Classes	Exceptions
ContentHandlerFactory	Authenticator	MalformedURLException
FileNameMap	ContentHandler	ProtocolException
SocketImplFactory	DatagramPacket	SocketException
SocketOptions	DatagramSocket	BindException
URLStreamHandlerFactory	MulticastSocket	ConnectException
	DatagramSocketImpl	NoRouteToHostException
	InetAddress	UnknownHostException
	NetPermission	UnknownServiceException
	PasswordAuthentication	
	ServerSocket	
	Socket	
	SocketImpl	
	SocketPermission	
	URL	
	URLClassLoader	
	URLConnection	
	HttpURLConnection	
	JarURLConnection	
	URLDecoder	
	URLEncoder	
	URLStreamHandler	

Пакетът предлага множество класове за комуникация в Internet (и Intranet), работа с URL, дефиниция на нови протоколи.
По специално могат а бъдат създавани и използвани

- *URL връзки*
- *Stream socket връзки (TCP)*
- *Datagram socket връзки (UDP)*

Идентификация на машина в мрежата (по-точно на мрежова карта)

Адрес IP - DNS форма - refg.tu-sofia.bg
“dotted quad” форма - 194.141.64.254

32 бита в IPv4:

Ако най-старшите битове на адреса са:
0xxx: мрежа от клас **A** - **над 16 000 000 адреса**
10xx: мрежа от клас **B** - 65533 адреса
110xx: мрежа от клас **C** - **254 адреса**

CIDR -- **C**lassless **I**nter**D**omain **R**outing RFC1519
Supernetting - ако са необходими 1000 адреса - могат да се използват 4 мрежи от клас C

```
192.60.128.0   (11000000.00111100.10000000.00000000)  Class C subnet address
192.60.129.0   (11000000.00111100.10000001.00000000)  Class C subnet address
192.60.130.0   (11000000.00111100.10000010.00000000)  Class C subnet address
192.60.131.0   (11000000.00111100.10000011.00000000)  Class C subnet address
-----------------------------------------------------
192.60.128.0   (11000000.00111100.10000000.00000000)  Supernetted Subnet address
255.255.252.0  (11111111.11111111.11111100.00000000)  Subnet Mask
192.60.131.255 (11000000.00111100.10000011.11111111)  Broadcast address
```

Подмрежата 192.60.128.0 включва всички адреси от 192.60.128.0 до 192.60.131.255. Идентификацията на мрежата е 22 бита а адресите вътре в мрежата - 10 бита.

CIDR нотация:

```
192.60.128.0, Subnet Mask 255.255.252.0
```

или:

```
192.60.128.0/22
```

128 бита в IPv6

Клас InetAddress - Internet адрес (DNS и “dotted quad” форма)

Не притежава публични член променливи и конструктори

Някои член функции

за създаване на обект от класа:

static InetAddress getByName(String host) - Определя IP адреса на машина (host).
static InetAddress[] getAllByName(String host) - Определя всички IP адреси на машината host.
public static InetAddress getByAddress(byte[] addr) - *addr* в IPv4 е 4 байта, в IPv6 - 16.
static InetAddress getLocalHost() - адреса на локалната машина - еквивалентен на:.
getByName(null) ~ getByName("localhost") ~ getByName("127.0.0.1")

други:

String getHostAddress() - IP адрес във форма на низ : "%d.%d.%d.%d".
String getHostName() - името на машината на този адрес.
boolean isMulticastAddress() - проверява дали не е multicast адрес.

```
import java.net.*;
public class NSLookupApp {
  public static void main(String args[]) {
    try {
         if(args.length!=1){
           System.out.println("Usage: java NSLookupApp hostName");
           return;
         }
         InetAddress host = InetAddress.getByName(args[0]);
         String hostName = host.getHostName();
         System.out.println("Host name: "+hostName);
        System.out.println("IP address: "+host.getHostAddress());
    }
    catch(UnknownHostException ex) {
         System.out.println("Unknown host");
         return;
    }
  }
}
```

```
java NSLookupApp localhost
Host name: localhost
IP address: 127.0.0.1
```

Упражнение: изведете на екран всички адреси на amazon.com

Класът URL

Напомняне

URL: Uniforme Resource Locator.

Низ от символи, която позволява локализирането по уникален начин един източник от Internet - протокол, машина(адрес IP+номер на порт), път на достъп(patl

протокол: //машина[:порт]/път/име

https://ff.tu-sofia.bg/JavaBg/Network/Reseau.html

Описание

Съдържа методи за отваряне и създаване на Web страници. Не притежава променливи.

Конструктори

Обект от класа URL може да се създаде с конструктор

- с параметри -протокол, машина и път
- с параметър - низ от символи (адрес URL)
- или с параметри - друг обект URL и път:

URL url1 = new URL("https", "ff.tu-sofia.bg", "/futurs-etudiants/");

URL u2 = new URL("https://ff.tu-sofia.bg/");

URL u3 = new URL(u1, "Java/Network.htm"); // http://ff.tu-sofia.bg/Java/Network.html

Всички конструктори могат да генерират MalformedURLException.

Пример:

```
import java.net.MalformedURLException;
import java.net.URL;

public class Test
{
    public static void main(String[] args)
            throws MalformedURLException {

        // creates a URL with string representation.
        URL url1 =
        new URL("https://ff.tu-sofia.bg"+
            "/sites/default/files/documents_ff/FICHE_INSCRIPION
-IC_002.pdf");

        // creates a URL with a protocol,host name,and path
        URL url2 = new URL("https", "ff.tu-sofia.bg",
                    "/futurs-etudiants/");

        //create a URL with another URL and path
        URL url3 = new URL("https://ff.tu-sofia.bg");
        URL url4 = new URL(url3, "/futurs-etudiants/");
```

```
        // print the String representation of the URL.
        System.out.println(url1.toString());
        System.out.println(url2.toString());
        System.out.println();
        System.out.println("Different components of the URL");

        // Retrieve the protocol for the URL
        System.out.println("Protocol url1: " + url1.getProtocol());

        // Retrieve the host name of the URL
        System.out.println("Hostname url1:  " + url1.getHost());

        // Retrieve the default port
        System.out.println("Default port url1: " +
url1.getDefaultPort());



        // Retrieve the path of URL
        System.out.println("Path url4: " + url4.getPath());

        // Retrieve the file name
        System.out.println("File url1: " + url1.getFile());
    }
}
```

Към URL могат да се закачват класове за потоков вход изход.

Пример: копиране на бинарен файл от зададен URL:

```
import java.io.*;
import java.net.*;
public class Test1 {
          public static void main(String arg[]) {
              int abyte;
              try{
                URL url = new URL("https://upload.wikimedia.org/wikipedia/commons/thumb/d/d9/"+
                    "Big_Bear_Valley%2C_California.jpg/1200px-Big_Bear_Valley%2C_California.jpg");
                InputStream fi= url.openStream();
                FileOutputStream fo= new FileOutputStream("picture.jpg");
                System.out.println("srarting ...");
                while((abyte= fi.read())!=-1)fo.write(abyte);
                fi.close();
                fo.close();
                System.out.println("created");

              }catch(MalformedURLException me) {
                  System.out.println("MalformedURLException: "+ me);
              }catch (IOException ioe){
                   System.out.println("IOException: "+ ioe);
              }

        }
}
```

С използване на буфер:

```
import java.io.*;
import java.net.*;
public class Test2 {
          public static void main(String arg[]) throws MalformedURLException, IOException{
              byte buf[]=new byte[1024];
              int len;
                URL url = new URL("https://upload.wikimedia.org/wikipedia/commons/thumb/d/d9/"+
                    "Big_Bear_Valley%2C_California.jpg/1200px-Big_Bear_Valley%2C_California.jpg");
                InputStream fi= url.openStream();
                FileOutputStream fo= new FileOutputStream("picture1.jpg");
                System.out.println("srarting ...");
                while((len= fi.read(buf))!=-1)fo.write(buf,0,len);
                fi.close();
                fo.close();
                System.out.println("created");
        }
}
```

Пример: работа с текстов файл - извеждане на кода на URL страницата на екрана. Използване на клас филтър за текст.

```
import java.io.*;
import java.net.*;

class ReadURL{
  public static void main(String[] a){
    try{
      URL url = new URL("https://nbu.bg");
      BufferedReader br = new BufferedReader(new
                          InputStreamReader(url.openStream()));
      String line;
      while((line=br.readLine()) !=null) {
        System.out.println(line);
      }
      br.close();
    }catch(MalformedURLException me) {
       System.out.println("MalformedURLException: "+ me);
    }catch (IOException ioe){
       System.out.println("IOException: "+ ioe);
    }
  }
}
```

Charset encoding

При четене на текстова информация в интернет може да се появи проблем с нейното кодиране. В зависимост от кодирането текстът може да бъде визуализиран по различен начин. Това не е проблем при латинските букви, но останалите се представят по различен начин в зависимост от използваното кодиране. Обикновено кодирането се задава в началото на страницата в meta tag. Например :

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Кодирането на страницата може да се получи програмно, чрез анализ на страницата или чрез използване на класа URLConnection.

В Java кодирането се управлява от клас **Charset**. При създаването на текстови потоци (входни и изходни) без указване на кодирането, се използва подразбиращото се кодиране на Java виртуалната машина.

```
import java.nio.charset.Charset;

public class Test{
    public static void main(String[] arg) {
        System.out.println("Local character encoding:
"+Charset.defaultCharset().displayName());
    }
}
```

Подразбиращото се кодиране се задава преди стартиране на дадена апликация, кешира се и на практика не може да бъде променено от програмата. Проблеми няма, само ако кодиранията на страницата и на локалната машина съвпадат, или ако на страницата има само символи, чиито ASCII и UNICODE кодове съвпадат.

При разлика в кодирането е необходимо при създаването на входния текстови поток да се укаже използваното кодиране на страницата.

```
import java.io.*;
import java.net.*;
import java.nio.charset.*;

class ReadURL{
  public static void main(String[] a){

    try{
      URL url = new URL("https://nbu.bg");
      BufferedReader br = new BufferedReader(new
                          InputStreamReader(url.openStream(),Charset.forName("UTF-8")));
      String line;
      while((line=br.readLine()) !=null) {
        System.out.println(line);
      }
      br.close();
    }catch(MalformedURLException me) {
       System.out.println("MalformedURLException: "+ me);
    }catch (IOException ioe){
       System.out.println("IOException: "+ ioe);
    }
  }
}
```

Клас URLConnection

Този клас осигурява по-добър контрол върху обмена на информацията на URL страницата

Конструиране - на две стъпки 1) конструира се URL 2) използва се функцията openConnection()

URL tu = new URL("https://www.tu-sofia.bg");

URLConnection tu_c = tu.openConnection();

Класът притежава методи за прочитане на параметри на страницата като:

тип на информацията getContentType(), която извежда използвания charset:

```
import org.omg.CORBA.SystemException;

import java.io.IOException;
import java.net.*;

public class URLc {
    public static void main(String[] arg) throws
    MalformedURLException,IOException {
        URL url = new URL( spec: "https://tu-sofia.bg/");
        URLConnection urlc=url.openConnection();
        System.out.println(urlc.getContentType());
    }

}
```

Sockets Programming in Java

- Streams
  - The basic of all I/O in Java is the data stream
  - A pipeline of data
    - put info into the pipeline (write) and get it (read)
- Programming with Sockets (TCP)
  - Opening a Socket
  - Creating a data input stream
  - Creating a data output stream
  - Closing the socket(s)

Opening a socket

- Client-side:

```
Socket myClient = null;
  try {
     MyClient = new Socket(“host”, PotNumber);
  } catch (UnknownHostException uhe) {
     uhe.printStackTrace();
  }
```

- “host” can be symbolic name or IP address

Opening a socket

- Server-side

```
ServerSocket myService = null;
try {
  myService = new ServerSocket(portNumber);
} catch (UnknownHostException uhe) {
  uhe.printStackTrace();
}
Socket serviceSocket;
serviceSocket = myService.accept();
```

Creating an input stream

- Client-side:

```
BufferedReader is = null;
 try {
    is = new BufferedReader(new
     InputStreamReader(myClient.getInputStream()));
 } catch (IOException ioe) {
    ioe.printStackTrace();
 }
```

Creating an input stream

- Server-side:

```
BufferedReader is = null;
try {
  is = new BufferedReader(new
InputStreamReader(serviceClient.getInputStream()));
} catch(IOException ioe) {
  ioe.printStackTrace();
}
```

Creating an output stream

- Client-side:

```
DataOutputStream os = null;
try {
   os = new
      DataOutputStream(myClient.getOutputStream());
} catch (IOException e) {
   e.printStrackTrace();
}
```

Creating an output stream

- Server-side:

```
DataOutputStream os = null;
try {
   os = new
DataOutputStream(serviceClient.getOutputStream());
} catch(IOException e) {
   e.printStackTrace();
}
```

Closing sockets

- Client-side:

```
try {
   os.close();
   is.close();
   myClient.close();
} catch(IOException e) {
    e.printStrackTrace();
}
```

Closing sockets

- Server-side:

```
try {
   os.close();
   is.close();
   serviceSocket.close();
   myService.close();
} catch(IOException e) {
   e.printStackTrace();
}
```

Прост TCP сървър

DateServer.java

```
import java.util.Date;
import java.io.OutputStreamWriter;
import java.io.IOException;
import java.net.Socket;
import java.net.ServerSocket;

public class DateServer {
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(2002);
        while (true) {
        Socket socket = serverSocket.accept();
            OutputStreamWriter out =
                new OutputStreamWriter(
                    socket.getOutputStream());
            out.write(new Date()+ "\n");
            out.close();
            socket.close();
        }
    }
}
```

```
telnet localhost 2002
```

Резултатът е получената от сървъра дата:

```
Wed Mar 03 20:31:05 EET 2004
```

Прост TCP клиент

DateServerClient.java

```
import java.io.*;
import java.net.Socket;

public class DateServerClient {
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("localhost", 2002);
        BufferedReader in = new BufferedReader(
            new InputStreamReader(
                socket.getInputStream() ) );
        System.out.println("The date on the server is: " +
            in.readLine());
        socket.close();
    }
}
```

```
The date on the server is: Wed Mar 03 20:34:12 EET 2004
```

Обработка на изключения

```
java.net.ConnectException: Connection refused: connect
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at java.net.PlainSocketImpl.doConnect(
        PlainSocketImpl.java:305)
    at java.net.PlainSocketImpl.connectToAddress(
        PlainSocketImpl.java:171)
    at java.net.PlainSocketImpl.connect(
        PlainSocketImpl.java:158)
    at java.net.Socket.connect(Socket.java:426)
    at java.net.Socket.connect(Socket.java:376)
    at java.net.Socket.<init>(Socket.java:291)
    at java.net.Socket.<init>(Socket.java:119)
    at DateServerClient.main(DateServerClient.java:6)
Exception in thread "main" Process terminated with exit code 1
```

- **Четенето от сокет е блокираща операция**
- **Писането във сокет също е блокираща операция**
- **Нормално прекъсване на TCP връзка**
- **Внезапно прекъсване на TCP връзка**

Multi-threaded Servers

- A server should be able to serve multiple clients simultaneously