

HTTP клиент

проф. д-р инж. Христо Вълчанов

<http://cs.tu-varna.bg>

Протокол HTTP

- Не зависи от типа на данните (media independent).
- Не поддържа съхраняване на състоянието (stateless).
- Използва TCP порт 80.

HTTP съобщения

- Базиран е на модела клиент-сървър.
- Използва съобщения в текстов формат.

Start-line
Headers
Empty-line
Message-body

HTTP заявки

- Заявка към HTTP сървър.
- Указват се методи за определени действия (GET, HEAD, ...).
- Примерна GET заявка:

```
GET /index.html HTTP/1.1
```

```
Host: www.myserver.com
```

HTTP отговори

- Отговор от HTTP сървър.

Status-line
Response-headers
Empty-line
Message-body

```
HTTP/1.1 200 OK
Date: Mon, 7 Jan 2015 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2014 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

```
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Тестване комуникацията с HTTP сървър

- Използва се **telnet** клиент.
- От командния ред се изпълнява

```
> telnet A.B.C.D 80
```

Тестване комуникацията с HTTP сървър

- Въвежда се Request-line (няма промпт):

GET index.html HTTP/1.1

- На следващия ред се въвежда адреса на сървъра:

HOST: A.B.C.D

- Въвеждат се два пъти символите за нов ред CR/LF

Тестване комуникацията с HTTP сървър

```
Administrator: Command Prompt

HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Mon, 03 Nov 2014 07:06:26 GMT
Accept-Ranges: bytes
ETag: "4b301faf34f7cf1:0"
Server: Microsoft-IIS/7.5
Date: Thu, 08 Jan 2015 14:21:25 GMT
Content-Length: 689

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>IIS7</title>
<style type="text/css">
<!--
body {
    color:#000000;
    background-color:#B3B3B3;
    margin:0;
}

#container {
    margin-left:auto;
    margin-right:auto;
    text-align:center;
}

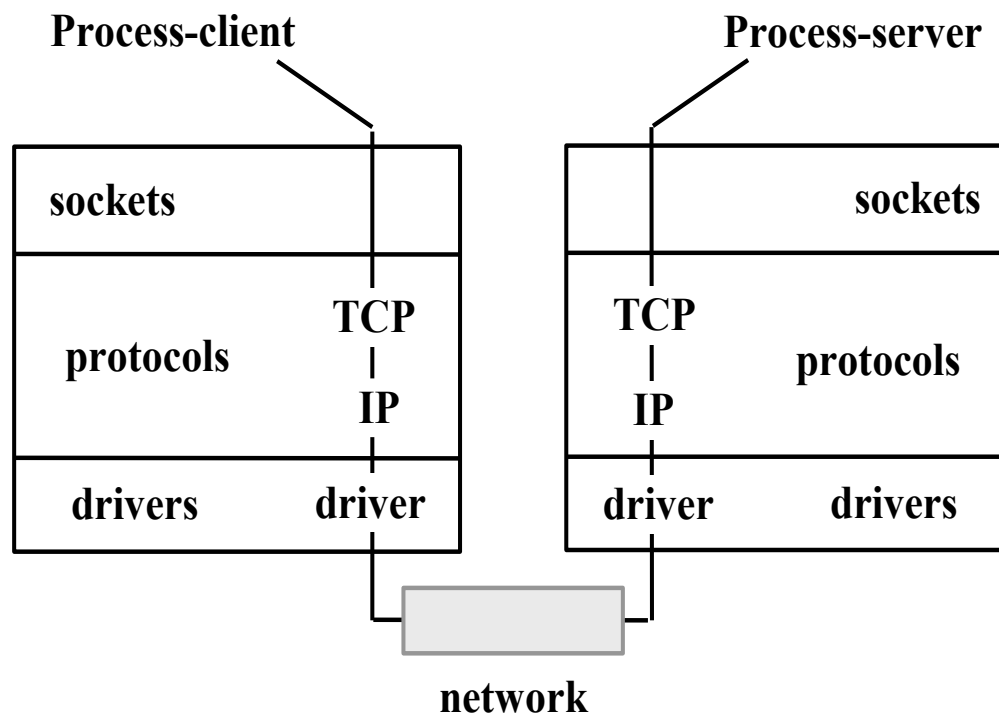
a img {
    border:none;
}
-->
</style>
</head>
<body>
<div id="container">
<a href="http://go.microsoft.com/fwlink/?linkid=66138&clcid=0x409"><img src=
"welcome.png" alt="IIS7" width="571" height="411" /></a>
</div>
</body>
</html>

Connection to host lost.
```


HTTP клиент

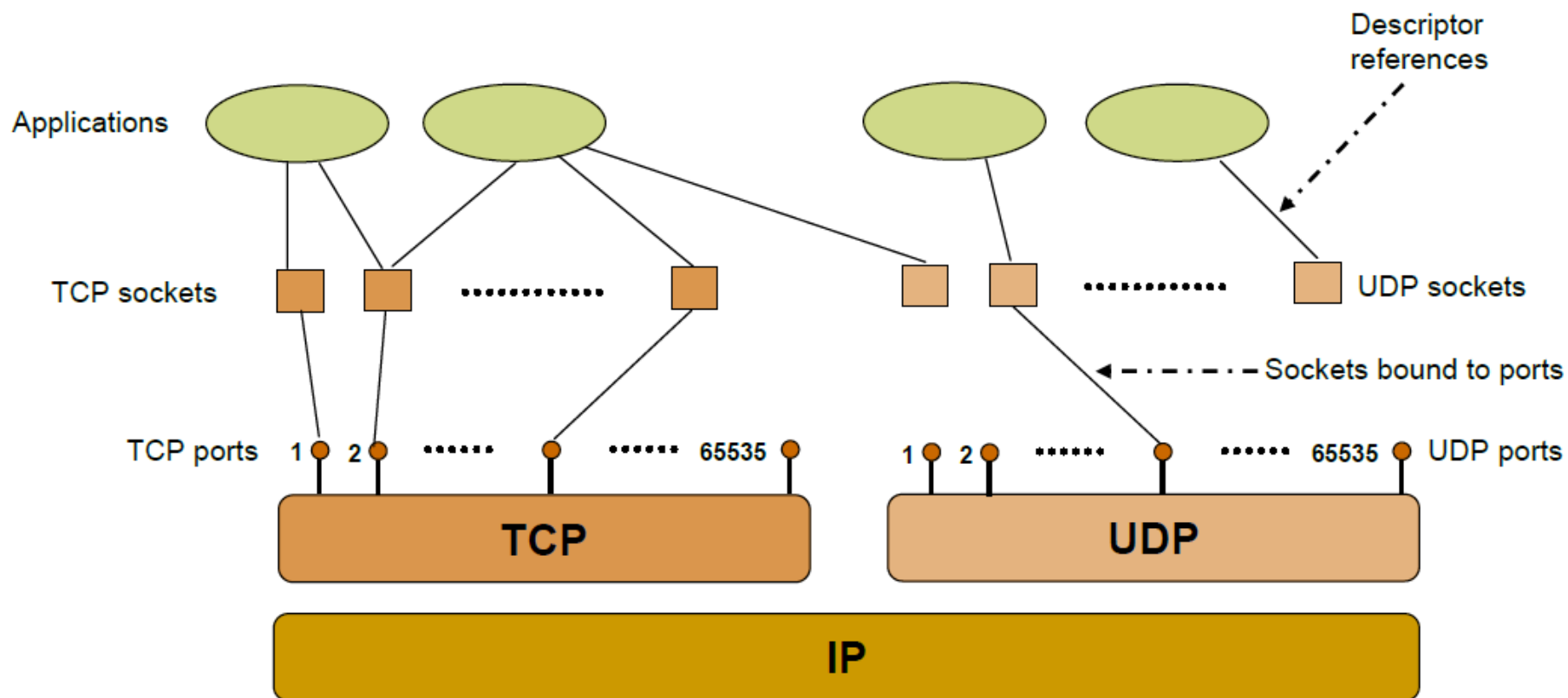
- Използва се механизма на сокетите.
- Библиотека за реализация – Winsock2.
- Език за програмиране C/C++

Сокети



- Предоставят двупосочна комуникация между два процеса от тип point-to-point.

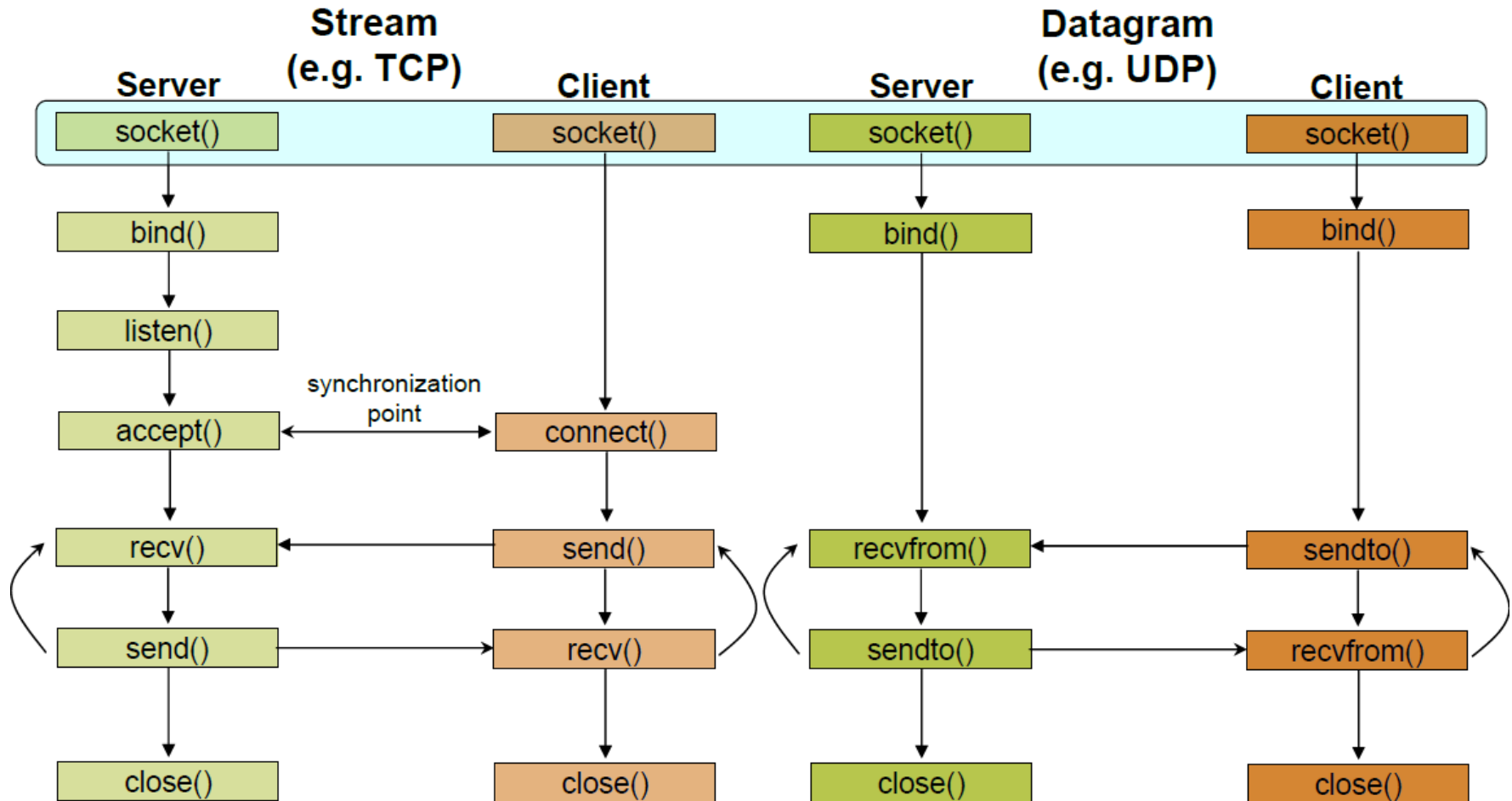
Сокети в Интернет домейна



Типове сокети

- Stream socket
- Datagram socket
- Sequential packet socket
- Raw socket

Комуникация клиент-сървър



Конструиране на съобщенията – подреждане на байтовете

- Адресите и портовете са integer
- Различните машини или ОС използват различно подреждане на байтовете:
 - Little-endian – старшите байтове напред;
 - Big-endian – младшите байтове напред.

Конструиране на съобщенията – подреждане на байтовете

192	168	0	20
-----	-----	---	----

Big-endian

192	A
168	A+1
0	A+2
20	A+3

192	168	0	20
-----	-----	---	----

Little-endian

20	A
0	A+1
168	A+2
192	A+3

Конструиране на съобщенията – решение

- **Host-byte ordering** – подреждането на байтовете в машината (Little-endian или Big-endian)
- **Network-byte ordering** – подреждането на байтовете в мрежата (винаги Big-endian)

Функции

- `u_long htonl (u_long x)`
- `u_short htons (u_short x)`
- `u_long ntohl (u_long x)`
- `u_short ntohs (u_short x)`
- При Big-endian машини не се извършват никакви действия
- При Little-endian машини се обръща реда на байтовете

HTTP клиент - последователност

1. Инициализиране на Winsock.
2. Създаване на сокет.
3. Осъществяване на конекция към сървър.
4. Изпращане и получаване на данни.
5. Затваряне на сокет и деактивиране на Winsock.

Библиотека Winsock

- Необходима библиотека ***ws2_32.lib***
- Включване на библиотеката:

```
#include <winsock2.h>
```

```
#include <ws2tcpip.h>
```

```
// Link with ws2_32.lib
```

```
#pragma comment (lib, "ws2_32.lib")
```

```
...
```

Инициализиране на библиотеката

```
int main(int argc , char *argv[]) {  
    // Инициализиране на поддръжката на сокети  
    unsigned short wVersionRequested;  
    WSADATA wsaData;  
    int err;  
  
    wVersionRequested = MAKEWORD( 2, 2 );  
    err = WSAStartup(wVersionRequested, &wsaData);  
    if ((err != 0) || (LOBYTE(wsaData.wVersion) != 2 ||  
        HIBYTE(wsaData.wVersion) != 2 )) {  
        printf("Winsock dll failed.\n");  
        return 1;  
    }  
    printf("Winsock Initialized.\n");  
  
    return 0;  
}
```

Създаване на сокет

```
int main(int argc , char *argv[]) {  
    int s;  
  
    ...  
  
    if((s=socket(AF_INET,SOCK_STREAM,0))==INVALID_SOCKET) {  
        printf("Could not create socket.\n");  
        return 1;  
    }  
  
    printf("Socket created.\n");  
  
    return 0;  
}
```

Структура за свързване към сървър

```
struct sockaddr {  
    u_short    sa_family;    // communication domain  
    char       sa_data[14];  // protocol-specific address  
};
```

Структура за свързване към сървър

```
struct sockaddr_in {  
    short      sin_family;    // Комуникационна  
    област  
    u_short    sin_port;      // Номер на порт  
    struct in_addr sin_addr; // IP адрес  
    char sin_zero[8];         // Не се използва  
};  
  
struct in_addr {  
    u_long    s_addr;         // 32 бита IP адрес  
};
```

Свързване към сървър

```
int main(int argc , char *argv[])
{
    int s;
    struct sockaddr_in server;

    ...

    server.sin_addr.s_addr=inet_addr("77.125.71.106");
    server.sin_family = AF_INET;
    server.sin_port = htons( 80 );

    // Свързване към отдалечен сървър
    if ( connect(s, (struct sockaddr *)&server,
                sizeof(server)) == SOCKET_ERROR ) {
        printf("Connect error.\n");
        return 1;
    }

    printf("Connected.\n");

    return 0;
}
```


Изпращане и получаване на данни

```
int send(int sd, const char *buf, int len, int flags);
```

```
int recv(int sd, char *buf, int len, int flags);
```

Затваряне на сокет

```
int s, res;
char recvbuf[BUFLen];
int recvbuflen = BUFLen;
...
// Затваря конекцията като няма повече да се изпращат данни
res = shutdown(s, SD_SEND);
if (res == SOCKET_ERROR) {
    printf("shutdown failed.\n");
    closesocket(s);
    WSACleanup();
    return 1;
}
do { // Получава докато отсрещната страна не затвори конекцията
    res = recv(s, recvbuf, recvbuflen, 0);
    if (res > 0)
        printf("Bytes received: %d\n", res);
    else if (res == 0)
        printf("Connection closed\n");
    else
        printf("recv failed.\n");
} while (res > 0);
closesocket(s); // Завършване
WSACleanup(); // Освобождава Winsock dll
```

Резолване на имена

- Необходимо е URL да се преобразува в IP адрес
- Използва се функцията `gethostbyname()`.
- Резултат с IP информацията:

```
struct hostent {  
    char *h_name;          // Официално име на хост  
    char **h_aliases;      // Списък с псевдоними  
    int h_addrtype;        // Тип на адреса на хоста  
    int h_length;          // Дължина на адреса  
    char **h_addr_list;    // Списък с адреси  
};
```

Резолване на имена - пример

```
char *hostname = "www.google.com";
char ip[100];
struct hostent *he;
struct in_addr **addr_list;
int i;

if ( (he = gethostbyname( hostname ) ) == NULL) {
    printf("gethostbyname failed.\n");
    return 1;
}

// Преобразува h_addr_list към in_addr,
// тъй като h_addr_list също има ip адреси само
// в long формат
addr_list = (struct in_addr **) he->h_addr_list;
for(i = 0; addr_list[i] != NULL; i++) {
    strcpy(ip , inet_ntoa(*addr_list[i]) );
    printf("%s resolved to: %s\n" , hostname , ip);
}

...
```

Въпроси?