

OTVORENO RA^UNARSTVO

MRE@NO PROGRAMIRANJE

MARIO @ AGAR

Otvoreno računarstvo

Mrežno programiranje

- Utičnice (*sockets*), Spojna komunikacija
- Programiranje TCP klijenta, Programiranje TCP poslužitelja
- Nespojna komunikacija
- Utičnice u Javi
- Struktura mrežnih aplikacija

Mario Žagar



Utičnice (*Sockets*)

- Općenit mehanizam međuprocene komunikacije
 - između procesa na istom računalu
 - između procesa na različitim računalima
- Berkeley sockets API: 4.2 BSD UNIX (1983)
 - API – apstrakcija mrežnih utičnica, jezik C
 - licencirano do 1989 – AT&T
 - *de facto* norma
 - ekvivalentni API postoje za većinu programskih jezika (pa i za PHP i Javu)

TCP/IP utičnice

- Svaka utičnica određena trojkom (adresa, protokol, vrata)
- Adresa poslužiteljske utičnice stalna:
 - kako bi se inače znali spojiti na neku uslugu?
 - web poslužitelj FER-a: (www.fer.hr, 80)
- Klijentska utičnica dinamički dodijeljena
 - utičnica je bitna samo za (kratkog) životnog vijeka klijenta
 - dodjelu vrši operacijski sustav računala po izboru



Spojna komunikacija

Spojna komunikacija (TCP)

- Ispravljanje grješaka nastalih u podatkovnom sloju
 - gubitak ili uvišestručenje paketa
 - poredak paketa
- Napredne funkcije:
 - upravljanje tokom podataka (TCP *sliding window*)
- Nedostaci:
 - dodatni resursi računala
 - dodani mrežni promet (stvaranje i zatvaranje veze, upravljanje vezom, retransmisije ...)

Izravan pristup Web poslužitelju



```
(lokalno pokrenut IIS na WinXP)
telnet localhost 80
GET / HTTP/1.0 <enter>
<enter>
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.1
Date: Sun, 27 Jan 2008 16:47:04 GMT
X-Powered-By: ASP.NET
Location: localstart.asp
Connection: Keep-Alive
Content-Length: 121
Content-Type: text/html
Set-Cookie: ASPSESSIONIDQCQRTSQT=DDINBNBCBCCONJIPOCFNMMIKP; path=/
Cache-control: private

<head><title>Object moved</title></head>
<body><h1>Object Moved</h1>This object may be found <a
HREF="">here</a>.</body>
```

Programiranje TCP klijenta

Jednostavan Web klijent



```
C:\WINDOWS\system32\cmd.exe

>php webclientSimple.php localhost 80 /
len = 424
-----
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.1
Date: Sun, 27 Jan 2008 17:18:45 GMT
X-Powered-By: ASP.NET
Location: localstart.asp
Connection: Keep-Alive
Content-Length: 121
Content-Type: text/html
Set-Cookie: ASPSESSIONIDQCQRTSQT=JDINBNCBJKDFDJDMGKKDEGCJ; path=/
Cache-control: private

<head><title>Object moved</title></head>
<body><h1>Object Moved</h1>This object may be found <a HREF="">here</a>.</body>

>_
```

Programski kôd Web klijenta

```
...

$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);

socket_connect($socket, $argv[1], $argv[2]);

socket_write($socket, "GET /$argv[3] HTTP/1.0\r\n\r\n");

$data = socket_read($socket, 100000);

socket_close($socket);

echo "len = " . strlen($data);
echo "\n-----\n";
echo $data . "\n";
?>
```



webclientSimple.php

Stvaranje utičnice

```
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
```

- Stvara klijensku TCP/IP utičnicu
 - domena: AF_INET – IPv4 TCP/IP (postoje i AF_INET6, AF_UNIX)
 - tip: SOCK_STREAM – pouzdana full-duplex komunikacija (TCP)
 - protokol korištene domene: SOL_TCP
 - klijentske utičnice ne mogu prihvaćati veze
- Vraća resurs (utičnicu) ili FALSE

Ostvarivanje veze

```
socket_connect($socket, $host, $port);
```

- Pokušava spajanje na poslužiteljsku utičnicu
 - \$socket – klijentska utičnica
 - \$host – IPv4 adresa poslužitelja
 - \$port – vrata poslužitelja
- Vraća TRUE ili FALSE

Pisanje

```
$sent = socket_write($socket, $data [, $dataLen]);
```

- Piše podatke u utičnicu (šalje na drugi kraj veze)
 - \$socket – utičnica
 - \$data – slani podaci
 - \$dataLen – duljina slanih podataka (opcionalan parametar)
- Vraća broj stvarno poslanih podataka ili FALSE

Čitanje



```
$data = socket_read($socket, $dataLen [, $type]);
```

- Čita raspoložive podatke iz utičnice
 - \$socket – utičnica
 - \$dataLen – maksimalan broj okteta za pročitati
 - \$type – tip podataka
 - PHP_BINARY_READ – za čitanje binarnih podataka
 - PHP_NORMAL_READ – čitanje staje na \n ili \r
- Vraća pročitane podatke ("" za kraj podataka) ili FALSE

Zatvaranje veze

```
socket_close($socket) ;
```

- Zatvara utičnicu (i vezu ako postoji), oslobađa resurse
- Nema povratne vrijednosti

Fragmentacija podataka (I)

- Duljina IP paketa ograničena (~1500 okteta)
 - dulji podaci se fragmentiraju u više paketa i takvi šalju prema odredišnom procesu
 - TCP sloj brine se za poredak paketa i retransmisiju izgubljenih paketa
 - na odredištu se vrši defragmentacija i punjenje izlaznog međuspremnik utičnice

Fragmentacija podataka (II)

- Svi podaci koji su prosljeđeni u pozivu `socket_write()` ne moraju biti i odmah poslani
 - što ako smo probali poslati više no što stane u izlazni međuspremnik?
 - što ako je mreža zagušena?
- Podaci pročitani sa `socket_read()` ne moraju biti i svi očekivani podaci koji tvore logičku cjelinu
 - npr. html kod čitave web stranice, znakovi naredbe zaključeni s `\r\n, ...`)
 - nisu svi paketi još pristigli, izlazni međuspremnik sadrži samo dio ...

Dobra praksa (I)

- Kod svih operacija sa utičnicama obavezno provjeravati izlazno stanje korištenih funkcija!
 - grješka u komunikaciji može se desiti u bilo kojem trenutku (Murphy bi rekao: u najnezgodnijem trenutku)
 - pratiti koliko je stvarno podataka poslano
 - pratiti koliko je podataka primljeno
 - koristiti vlastiti međuspremnik za prihvatanje svih podataka neke logičke cjeline pogodne za daljnju obradu (npr. čitavog retka koji sadrži naredbu)

Dobra praksa (II)

- Oslobađati resurse na kraju njihova korištenja
 - operacijski sustav ima na raspolaganju ograničen broj utičnica!
- Prethodni primjer je radio ispravno jer:
 - je mala količina podataka slana prema poslužitelju (naredbe HTTP-a nisu dugačke, ali problem bi mogao nastati s velikim brojem podataka u zaglavlju naredbe)
 - mala količina podataka (424 okteta) primljena kao odgovor poslužitelja (nikako nije pravilo, html stranice znaju biti vrlo dugačke)

Posljedice loše prakse...

duljina sadržaja pročitano
iz međuspremnik (<1500 !)

web poslužitelj tvrdi da je
duljina sadržaja 4431 (metoda
brojanja okteta!)

sadržaj web stranice je
nepotpun!

```

C:\WINDOWS\system32\cmd.exe

C:\>php webContent\Simple.php localhost 80 /localstart.asp
len = 1460
-----
HTTP/1.1 401 Access Denied
Server: Microsoft-IIS/5.1
Date: Sun, 27 Jan 2008 18:12:35 GMT
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
WWW-Authenticate: Basic realm="127.0.0.1"
Content-Length: 4431
Content-Type: text/html

<?DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html dir=ltr>

<head>
<style>
a:link                {font:8pt/11pt verdana; color:FF0000}
a:visited              {font:8pt/11pt verdana; color:#4e4e4e}
</style>

<META NAME="ROBOTS" CONTENT="NOINDEX">

<title>You are not authorized to view this page</title>

<META HTTP-EQUIV="Content-Type" Content="text-html; charset=Windows-1252">
</head>

<script>
function Homepage()
{
    <!--
    // in real hits, urls get returned to our script like this:
    // res://shdocvw.dll/http_404.htm#http://www.DocURL.com/bar.htm

    //For testing use DocURL = "res://shdocvw.dll/http_404.htm#https://www.m
icrosoft.com/bar.htm"
    DocURL=document.URL;

    //this is where the http or https will be, as found by searching for ://
    but skipping the res://
    protocolIndex=DocURL.indexOf("://",4);

    //this finds the ending slash for the domain server
    serverIndex=DocURL.indexOf("/",protocolIndex + 3);

    //for the href we need a valid URL to the domain. We search for the # s
ymbol to find the beginning
    //of the true URL, and add 1 to skip it - this is the BeginURL value. We
    use serverIndex as the end marker.
    //urlresult=DocURL.substring(protocolIndex - 4,s
    >

```

... i primjer dobrog klijenta

```

$socket = @socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
if( $socket == FALSE ) reportError();

$status = socket_connect($socket, $argv[1], $argv[2]);
if( $status == FALSE ) reportError();

$req = "GET /$argv[3] HTTP/1.0\r\n\r\n";
$req_sent = 0;
do {
    $status = socket_write($socket, substr($req, $req_sent));
    if( $status == FALSE ) reportError();
    $req_sent += $status;
} while ( $req_sent != strlen($req) );

$data = "";
while( ($rbuff = socket_read($socket, 1500)) != FALSE )
    $data .= $rbuff;

socket_close($socket);

```

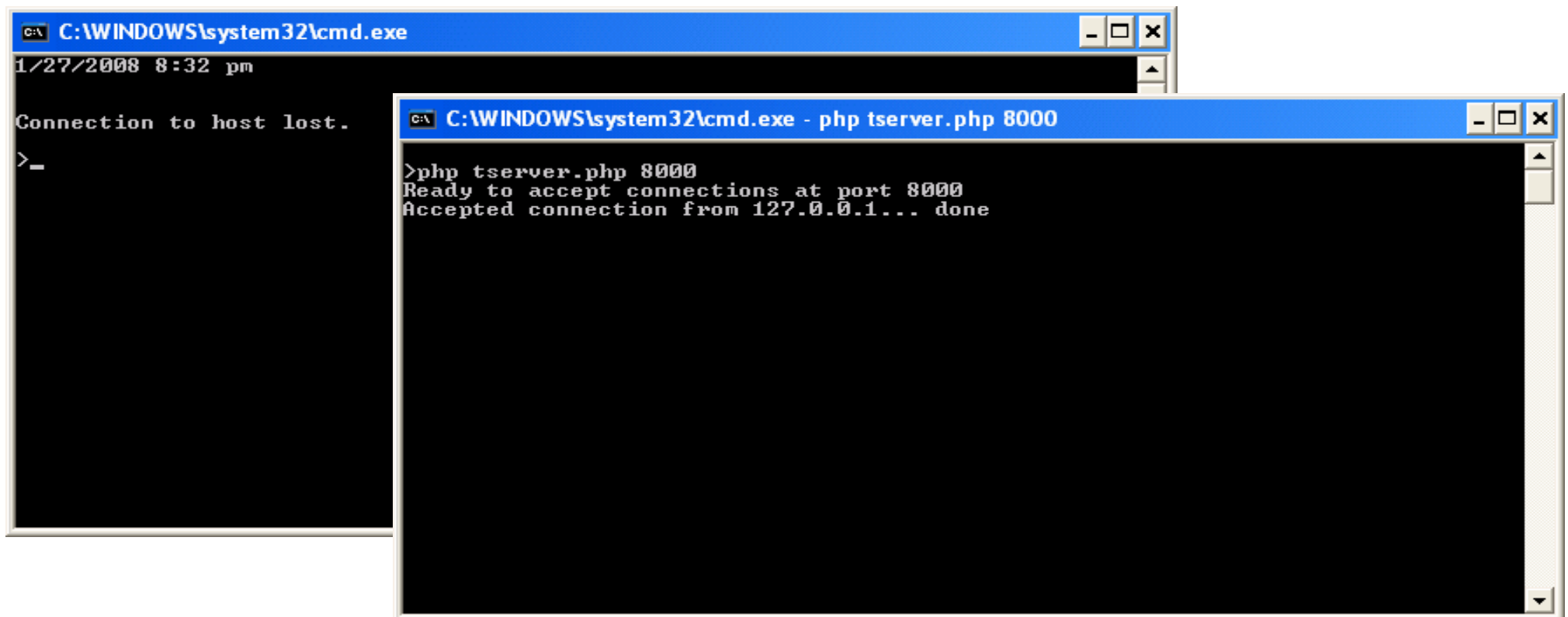


webclientFull.php

Programiranje TCP poslužitelja

Jednostavan poslužitelj

- Poslužitelj prihvaća vezu od strane klijenta, čeka 2 sekunde, šalje vrijeme na poslužitelju, zatvara vezu
- Spajanje na poslužitelj – telnet



The image shows two overlapping Windows command prompt windows. The background window has a title bar 'C:\WINDOWS\system32\cmd.exe' and shows the date and time '1/27/2008 8:32 pm'. It displays the message 'Connection to host lost.' followed by a prompt '>_'. The foreground window has a title bar 'C:\WINDOWS\system32\cmd.exe - php tserver.php 8000'. It shows the command '>php tserver.php 8000' being executed, followed by the output: 'Ready to accept connections at port 8000' and 'Accepted connection from 127.0.0.1... done'.

```
C:\WINDOWS\system32\cmd.exe
1/27/2008 8:32 pm

Connection to host lost.
>_

C:\WINDOWS\system32\cmd.exe - php tserver.php 8000
>php tserver.php 8000
Ready to accept connections at port 8000
Accepted connection from 127.0.0.1... done
```

Programski kod poslužitelja

```
...
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
socket_bind($socket, '127.0.0.1', $argv[1]);
socket_listen($socket, 2);
echo "Ready to accept connections at port $argv[1]\n";

while(TRUE) {

    $cSocket = socket_accept($socket);
    $peerName = "";
    socket_getpeername($cSocket, $peerName);
    echo "Accepted connection from $peerName... ";
    sleep(2); //"processing delay" :)

    socket_write($cSocket, date('n/j/Y g:i a') . "\n");
    socket_close($cSocket);
    echo "done\n";
}
socket_close($socket);
?>
```



tserver.php

Vezanje utičnice

```
socket_bind($socket, $addr[, $port]);
```

- Veže utičnicu stvorenu sa `socket_create()` za navedenu adresu (`$addr`) i vrata (`$port`)
 - jedno računalo može imati više mrežnih sučelja, IP adresa
 - bez eksplicitnog vezivanja za jedan par (host, port) operacijski sustav bi te informacije stvorio nasumično
- Vraća TRUE ili FALSE
 - vrata mogu biti prethodno zauzeta
 - vrata 0-1023 rezervirana za potrebe sustava

Poslužiteljski mod utičnice

```
socket_listen($socket, $backlog);
```

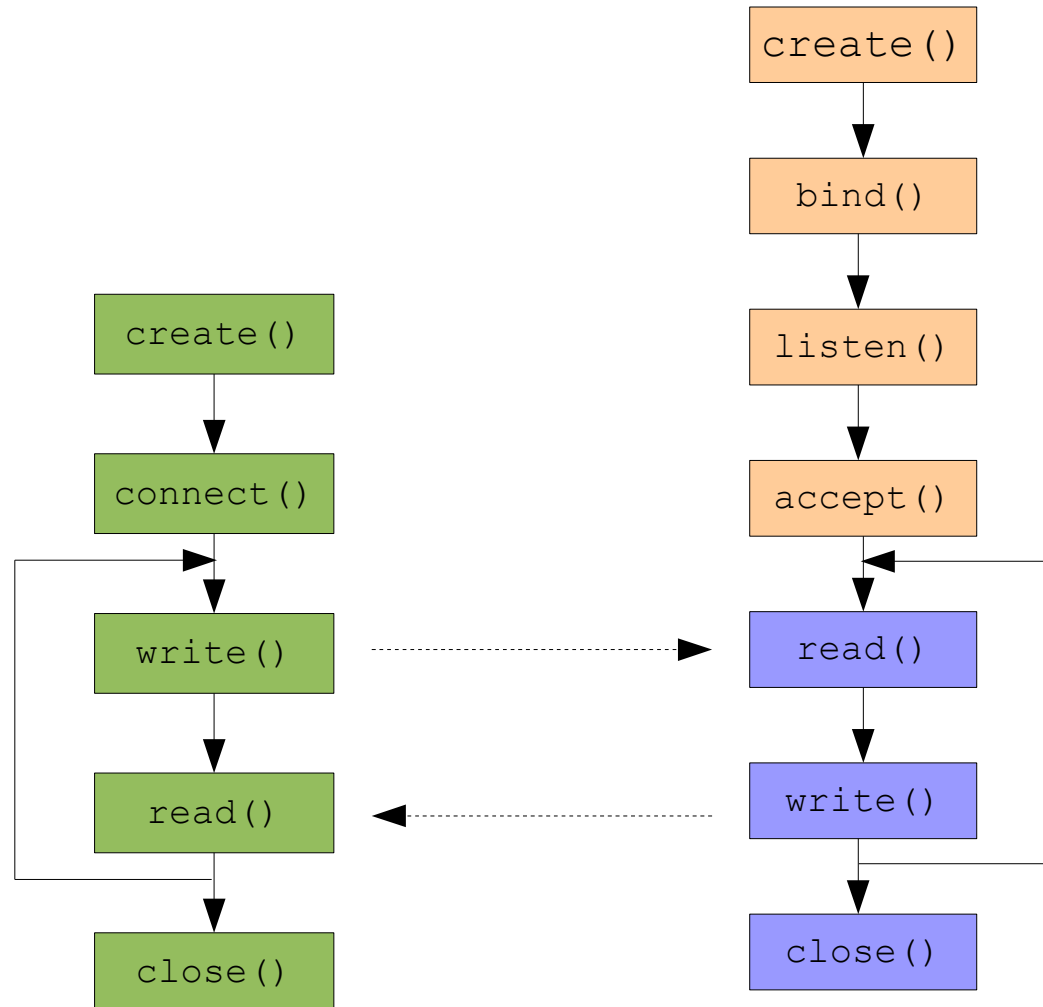
- Pretvara utičnicu u poslužiteljsku
 - \$socket – utičnica
 - utičnica mora biti stvorena i vezana na adresu
 - \$backlog – duljina reda čekanja na vezu
 - istovremeno može biti aktivna samo jedna veza s klijentom
 - ostali zahtjevi za vezom se spremaju u red čekanja
 - ukoliko je red čekanja popunjen, daljnji zahtjevi se odbijaju
- Vraća TRUE ili FALSE

Prihvaćanje veze

```
$cSocket = socket_accept($socket) ;
```

- Prihvaća vezu na priključnici \$socket i preusmjerava je na utičnicu \$cSocket
 - funkcija blokira izvođenje do prispjeća zahtjeva za vezu
 - za komunikaciju s klijentom koristi se novostvorena utičnica
 - postojeća utičnica može prihvaćati nove veze
- Vraća novu utičnicu ili FALSE

Redoslijed korištenja naredaba



Složeniji poslužitelj

- Stvara se trajna veza, poslužitelj odgovara na naredbe `time` (vraća trenutno vrijeme) i `close` (zatvara vezu s klijentom)
- Klijent - telnet

The image shows two overlapping Windows command prompt windows. The background window is titled 'C:\WINDOWS\system32\cmd.exe' and shows a telnet session. The foreground window is titled 'C:\WINDOWS\system32\cmd.exe - php tserver.php 8000' and shows the server's output.

```

C:\WINDOWS\system32\cmd.exe
1/27/2008 9:06 pm
time
1/27/2008 9:06 pm
times
not understood: times
close
server: closing connection
Connection to host lost.
>_

C:\WINDOWS\system32\cmd.exe - php tserver.php 8000
>php tserver.php 8000
Ready to accept connections at port 8000
Accepted connection from 127.0.0.1.
Connection with 127.0.0.1 closed.
  
```

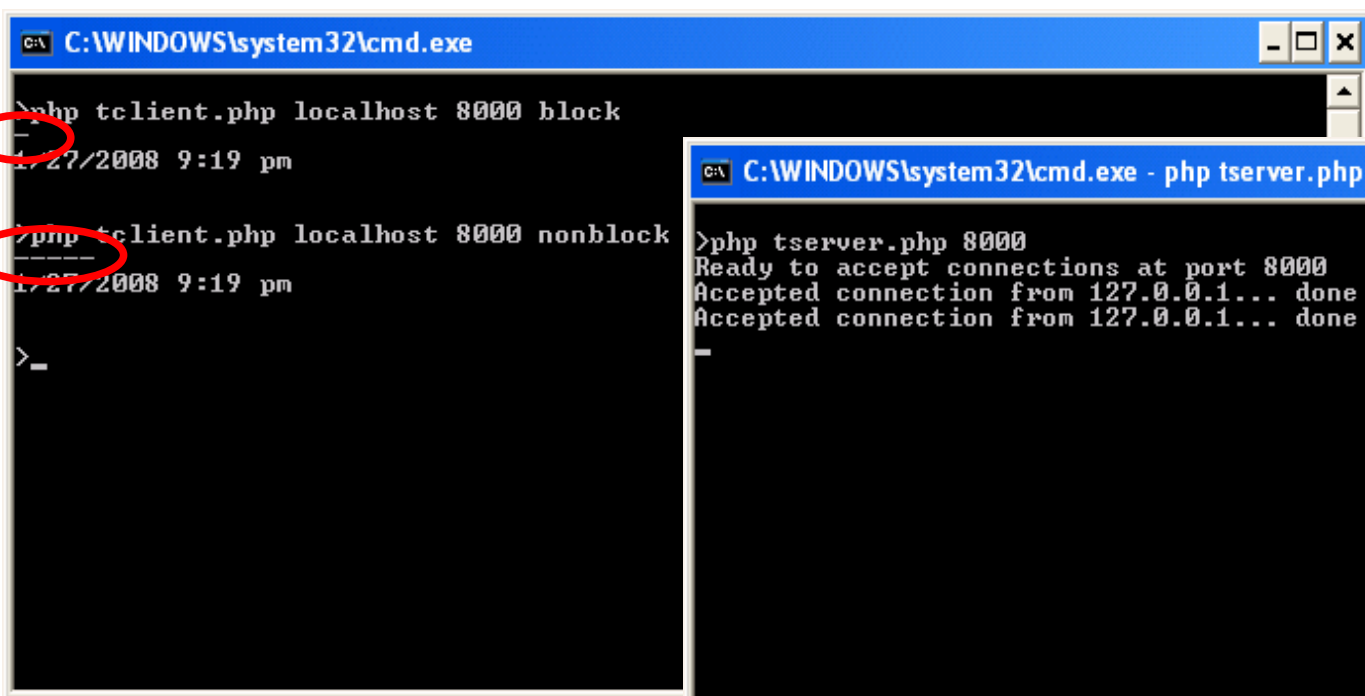
At the bottom right, there is a small icon of a PHP file and a text box containing the filename `tserver.php`.

Usporedne veze na poslužitelj

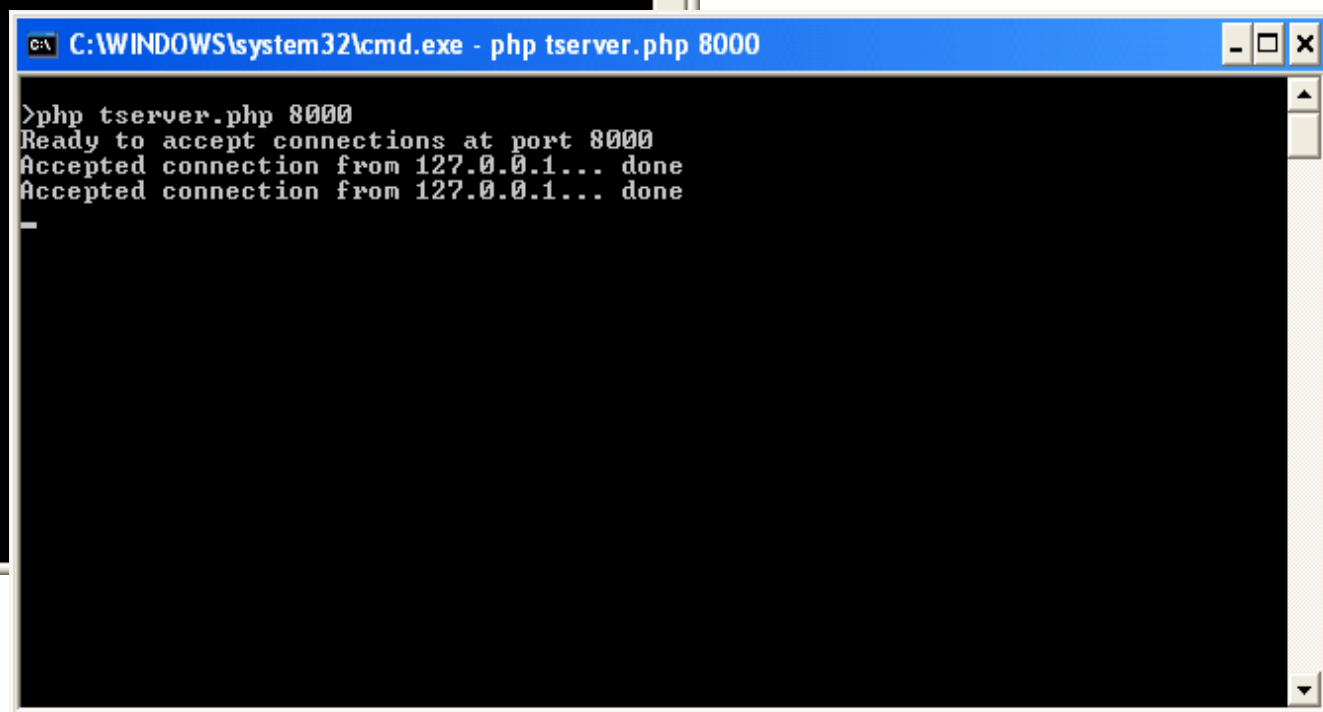
- Poslužitelj ima samo jednu nit izvršavanja
 - može prihvatiti samo jednu vezu istovremeno
 - ostale veze na čekanju ili se odbijaju
- Problem vezan uz *telnet* kao klijent program
 - *telnet* šalje znak po znak, ne retke!
 - na poslužitelju funkcija čitanja iz utičnice vraća jedan po jedan znak (potreban je redak terminiran s \n)
 - korištenje međuspremnik za sastavljanje naredbe i detekciju kraja retka (pritiska na Enter u telnet klijentu)
 - brisanje kontrolnih znakova iz niza znakova naredbe

Neblokirajuća komunikacija

- Poziv funkcije *socket_read()* je blokirajući – vraća se tek nakon prispjeća podataka od poslužitelja
 - da li se to vrijeme umjesto na čekanju može bolje utrošiti?



```
C:\WINDOWS\system32\cmd.exe
>php telnet.php localhost 8000 block
1/27/2008 9:19 pm
>php telnet.php localhost 8000 nonblock
1/27/2008 9:19 pm
>
```



```
C:\WINDOWS\system32\cmd.exe - php tserver.php 8000
>php tserver.php 8000
Ready to accept connections at port 8000
Accepted connection from 127.0.0.1... done
Accepted connection from 127.0.0.1... done
-
```

Neblokirajuće čitanje

```
...
if( strcasecmp($argv[3], "nonblock") == 0 )
    socket_set_nonblock($socket);
elseif( strcasecmp($argv[3], "block") == 0 )
    socket_set_block($socket);
...

socket_write($socket, "TIME\n");
$timeout = 10;

do {
    usleep(500000);        // 0.5 second sleep
    $data = socket_read($socket, 1500);
    echo "-";
    if( --$timeout == 0 ) {
        echo "response timeout!\n";
        reportError();
    }
} while( $data == FALSE );
echo "\n" . $data . "\n";
...
```



tclient.php



Nespojna komunikacija

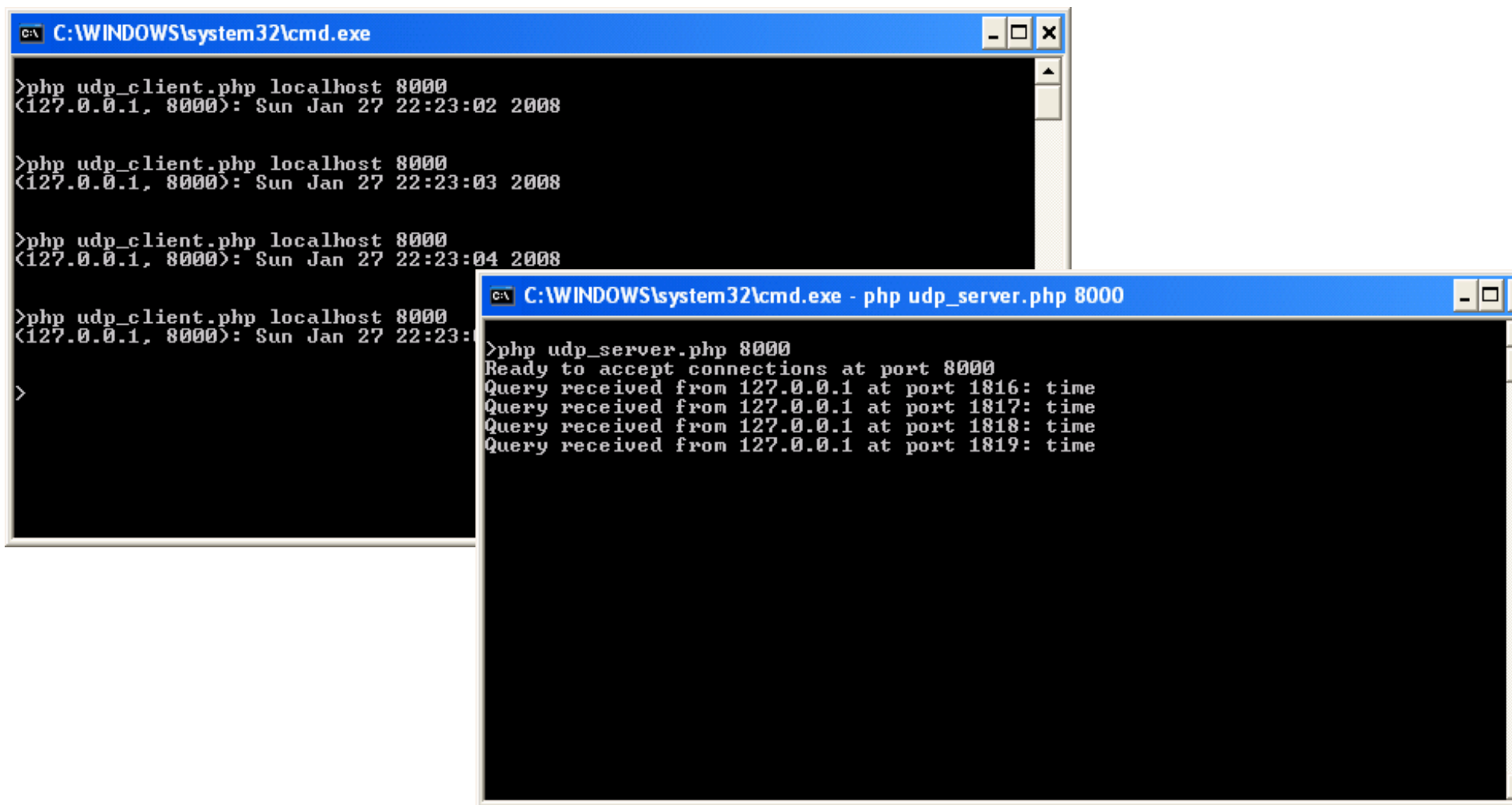
Nespojna komunikacija (UDP)



- Prednosti
 - jednostavnost
 - učinkovita za prenošenje manjih količina podataka
 - nema trajnog zauzeća poslužitelja od jednog klijenta
- Nedostaci:
 - nepouzdana, moguć gubitak ili uvišestručenje paketa
 - dodatna logika unutar aplikacije za detekciju i ispravljanje grešaka
 - složena izvedba u slučaju praćenja stanja konverzacije više klijenata

UDP klijent i poslužitelj

- Klijent šalje upit, poslužitelj vraća vrijeme



The image shows two overlapping Windows command prompt windows. The top window, titled 'C:\WINDOWS\system32\cmd.exe', shows the execution of a PHP UDP client script. It runs 'php udp_client.php localhost 8000' four times, each time receiving a response from 127.0.0.1 at port 8000 with a timestamp. The bottom window, titled 'C:\WINDOWS\system32\cmd.exe - php udp_server.php 8000', shows the execution of a PHP UDP server script. It runs 'php udp_server.php 8000', which then displays 'Ready to accept connections at port 8000' and four lines of received queries from 127.0.0.1 at ports 1816, 1817, 1818, and 1819, each with a timestamp.

```
C:\WINDOWS\system32\cmd.exe
>php udp_client.php localhost 8000
<127.0.0.1, 8000>: Sun Jan 27 22:23:02 2008

>php udp_client.php localhost 8000
<127.0.0.1, 8000>: Sun Jan 27 22:23:03 2008

>php udp_client.php localhost 8000
<127.0.0.1, 8000>: Sun Jan 27 22:23:04 2008

>php udp_client.php localhost 8000
<127.0.0.1, 8000>: Sun Jan 27 22:23:05 2008

>

C:\WINDOWS\system32\cmd.exe - php udp_server.php 8000
>php udp_server.php 8000
Ready to accept connections at port 8000
Query received from 127.0.0.1 at port 1816: time
Query received from 127.0.0.1 at port 1817: time
Query received from 127.0.0.1 at port 1818: time
Query received from 127.0.0.1 at port 1819: time
```

Jednostavan UDP klijent

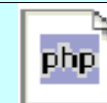
```
...
$socket = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
if( $socket == FALSE ) reportError();

$req = "time";
if( socket_sendto($socket, $req, strlen($req), 0, $argv[1], $argv[2])==-1 )
    reportError();

$s_addr = "";
$s_port = 0;

$len = @socket_recvfrom($socket, $message, 1500, 0, $s_addr, $s_port);
if($len > 1)
    echo "($s_addr, $s_port): $message\n";
else
    echo "Error!\n";

socket_close($socket);
...
```



udp_client.php

Jednostavan UDP poslužitelj



```
$socket = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
if( $socket == FALSE ) reportError();
if( socket_bind($socket, '127.0.0.1', $argv[1]) == FALSE ) reportError();
echo "Ready to accept connections at port $argv[1]\n";

do {
    ...
    $len = @socket_recvfrom($socket, $msg, 1500, 0, $c_addr, $c_port);
    if($len == -1) reportError();
    else if($len == 0) continue;

    echo "Query received from ". $c_addr ." at port ". $c_port .": $msg\n";

    if( strcasecmp($msg, "time") == 0 ) $rsp = date("D M j H:i:s Y\r\n");
    else $rsp = date("not understood");

    if( socket_sendto($socket, $rsp, strlen($rsp), 0, $c_addr, $c_port) == -1)
        reportError();

} while( TRUE );
```



udp_server.php

Slanje datagrama

```
socket_sendto($socket, $msg, $msglen, $flags, $addr, $port);
```

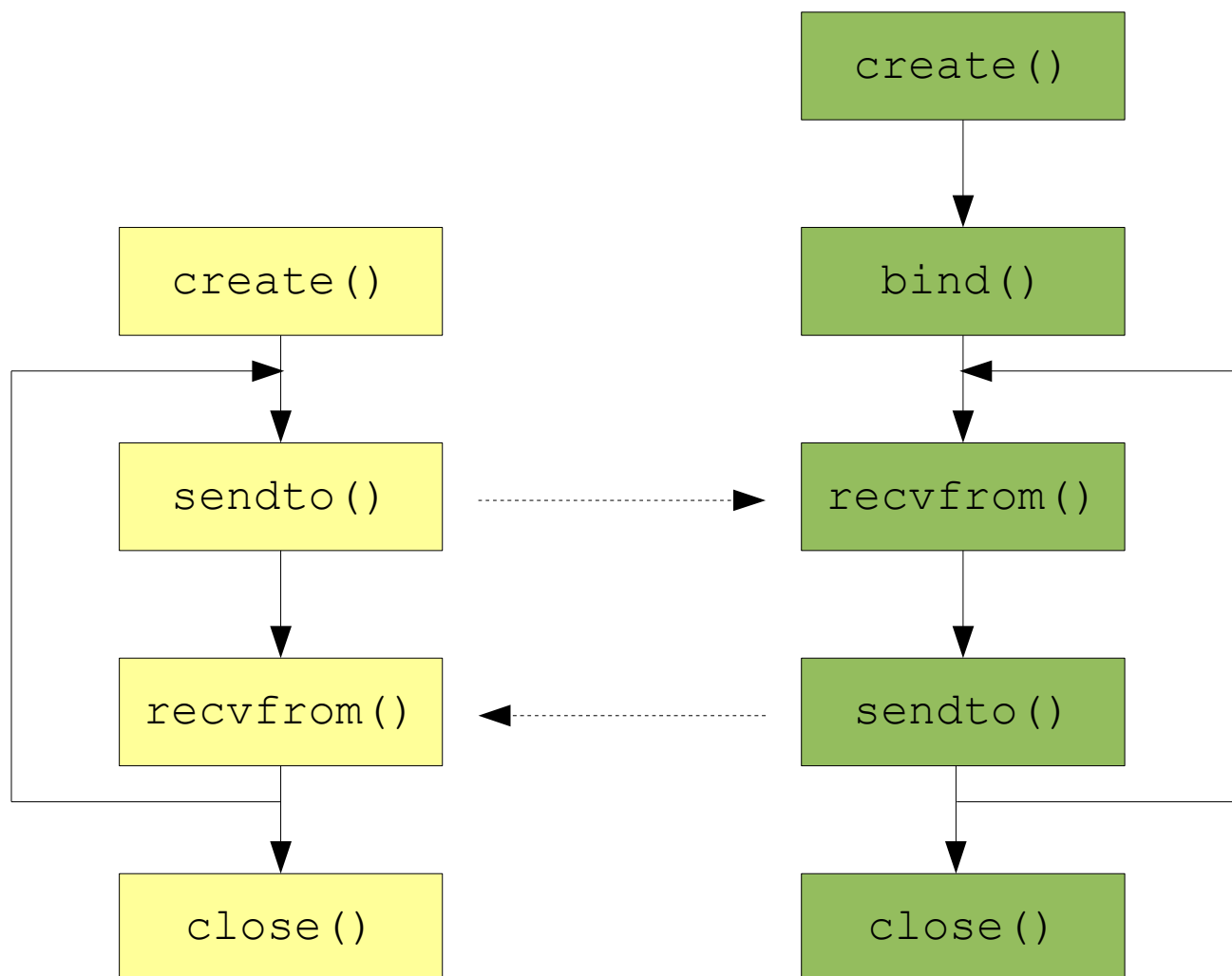
- Šalje podatke u datagramu
 - \$socket – utičnica
 - \$msg - poruka
 - \$msgLen – duljina poruke
 - \$flags – zastavice (vidjeti dokumentaciju)
 - \$addr – IPv4 adresa računala na koji se šalje datagram
 - \$port – vrata na koje se šalje datagram
- Vraća broj stvarno poslanih podataka ili -1

Primanje datagrama

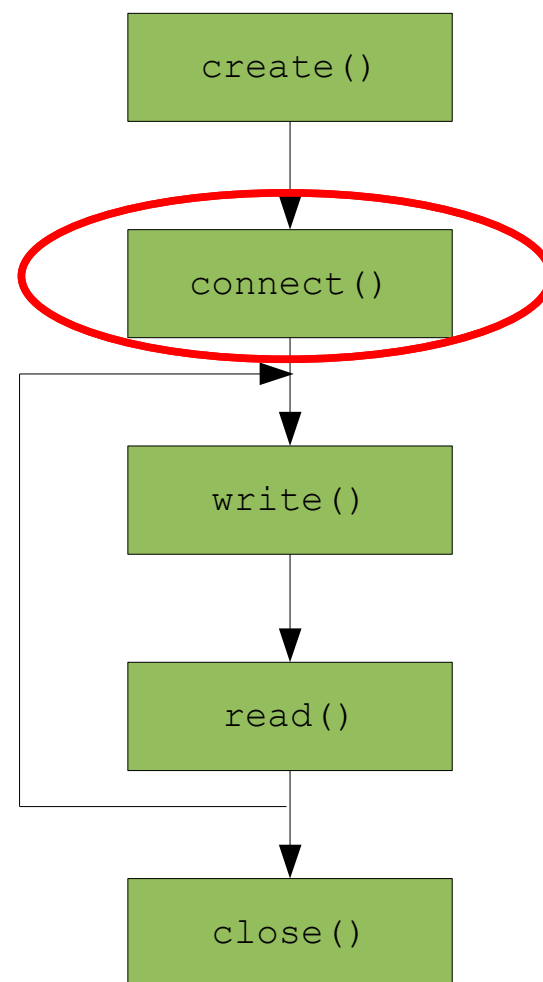
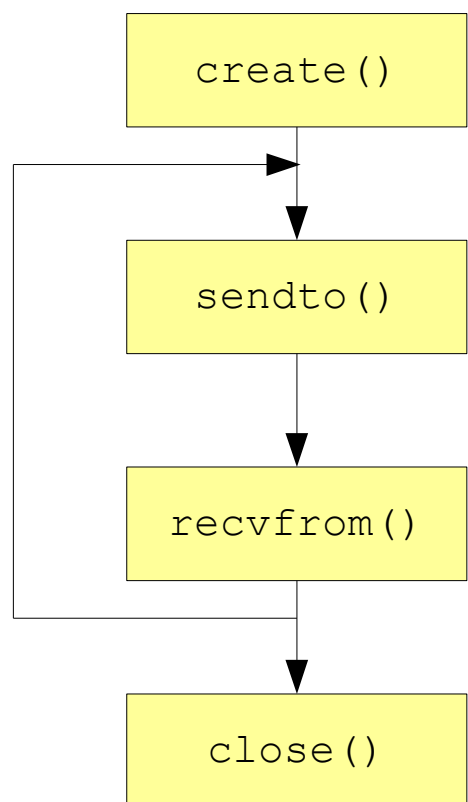
```
socket_recvfrom($socket, $msg, $maxLen, $flags, $addr, $port)
```

- Čita raspoložive podatke iz datagrama
 - \$socket – utičnica
 - \$msg - poruka
 - \$maxLen – najveća duljina poruke
 - \$flags – zastavice (vidjeti dokumentaciju)
 - \$addr – IPv4 adresa računala s kojeg je poslan datagram
 - \$port – vrata s kojih je poslan datagram
- Vraća veličinu primljenih podataka ili -1

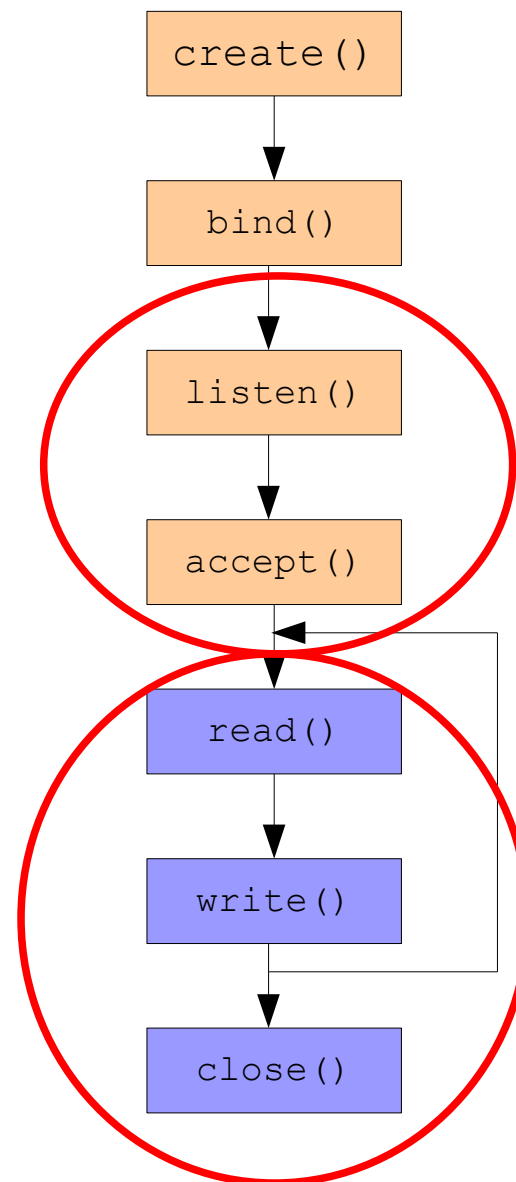
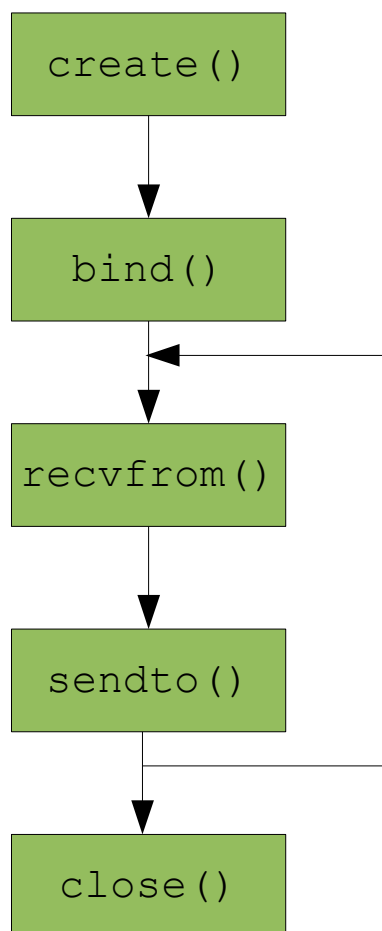
Redoslijed korištenja naredaba



Usporedba TCP i UDP klijenta



Usporedba TCP i UDP poslužitelja

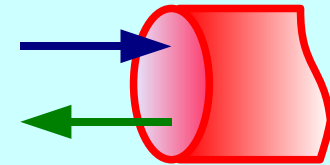




Utičnice u Javi

Klijent; TCP utičnice

```
...
Socket socket = null;
PrintWriter out = null;
BufferedReader in = null;
try {
    socket = new Socket(address, port);
    out = new PrintWriter(socket.getOutputStream());
    in=new BufferedReader(new InputStreamReader
                          (socket.getInputStream()));
} catch ( ...
)
String request = "GET " + args[2] + " HTTP/1.0\r\n\r\n";
out.print(request); out.flush();
String inLine = null;
try {
    while( (inLine = in.readLine()) != null)
        System.out.println(inLine);
} catch (IOException e) { ...
} ...
out.close(); in.close(); socket.close();
```



webclient.java

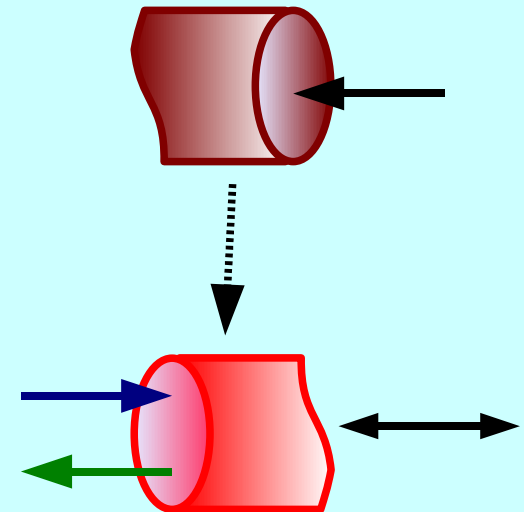
Poslužitelj; TCP utičnice



```
...
ServerSocket ssocket = null;
try {
    ssocket = new ServerSocket(port, backlog);
} catch (IOException e) {
    ...
while(true) {

    Socket cSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;

    try {
        cSocket = ssocket.accept();
        out = new PrintWriter(cSocket.getOutputStream());
        in = new BufferedReader(new InputStreamReader(cSocket.getInputStream()));
    } catch (IOException e) {
        ...
    }
}
```



tserver.java



tnserver.java

Višenitni poslužitelj

```
...
ServerSocket ssocket = null;
try {
    ssocket = new ServerSocket(port, backlog);
} catch (IOException e) {
    ...
while(true) {
    Socket cSocket = null;
    try {
        cSocket = ssocket.accept();
        tnserverThread serverThread = new tnserverThread(cSocket);
        new Thread(serverThread).start();
    } catch (IOException e) {
        ...
    }
}
```



tserverMT.java

- Nakon prihvaćanja veze i stvaranja utičnice:
 - stvara se novi objekt i predaje novostvorena utičnica
 - pokreće se nova nit izvršavanja za obradu zahtjeva
 - originalna nit nastavlja s prihvaćanjem novih klijenata

Nit posluživanja klijenta

- Razred `TserverThread` implementira sučelje `Runnable`, nit izvršavanja u Javi, izvođenje unutar metode `run()`
- Novostvorena utičnica prema klijentu prosljeđuje se kao parametar konstruktora
- Implementacija komunikacije s klijentom ista kao i u iterativnoj izvedbi poslužitelja
- Kraj komunikacije s klijentom: zatvaranje tokova, utičnice i izlazak iz metode `run()` (prekida se izvođenje niti)



`tserverThread.java`

Klijent; UDP utičnice

```
...  
DatagramSocket csocket = new DatagramSocket();  
  
byte[] buff = "time".getBytes();  
  
...  
DatagramPacket msg = new DatagramPacket(buff, buff.length, addr, port);  
csocket.send(msg);  
  
byte[] rbuff = new byte[100];  
DatagramPacket rPacket = new DatagramPacket(rbuff, rbuff.length);  
csocket.receive(rPacket);  
System.out.println(new String(rbuff));  
csocket.close();  
  
...
```



udp_client.java

Poslužitelj; UDP utičnice

```
DatagramSocket ssocket = new DatagramSocket(port);  
while(true) {  
    byte[] buffer = new byte[100];  
    DatagramPacket dPacket = new DatagramPacket(buffer, buffer.length);  
    String message = null;  
    ssocket.receive(dPacket);  
    message = new String(buffer);  
    System.out.println("Query received from " + dPacket.getAddress() + " at port "  
        + dPacket.getPort() + ": " + message);  
  
    String response = null;  
    if( message.trim().equalsIgnoreCase("time") )  
        response = new Date().toString();  
    else  
        response = "not understood";  
  
    byte[] rBuffer = response.getBytes();  
  
    DatagramPacket rPacket = new DatagramPacket(rBuffer, rBuffer.length,  
        dPacket.getAddress(), dPacket.getPort());  
    ssocket.send(rPacket);  
}
```



udp_server.java

Struktura mrežnih aplikacija

Funkcionalne cjeline

- Programski kôd aplikacije s izravnom povezanošću (korištenje utičnica) mora sadržavati:
 - upravljanje mehanizmom komunikacije
 - identificiranje i lociranje druge strane u kom.
 - upravljanje utičnicama
 - upravljanje razmjenjivanim podacima
 - upravljanje grješcima u komunikaciji
 - upravljanje protokolom komunikacije
 - praćene stanja protokola
 - oporavak od pogrešaka u protokolu
 - logiku aplikacije



Razvoj raspodijeljenih aplikacija



- Ciljevi u razvoju raspodijeljenih aplikacija:
 - smanjiti količinu potrebnog koda (vrijeme)
 - svojstva skalabilnosti, interoperabilnosti, ponovne iskoristivosti koda, pouzdanosti ...
- Načini postizanja cilja:
 - korištenje mehanizama komunikacije više razine apstrakcije
 - skrivanje protokola komunikacije
 - skrivanje komunikacijskih mehanizama
 - izmještanje dijelova funkcionalnosti u gotove infrastrukturne komponente (usluge)



Skrivanje protokola komunikacije

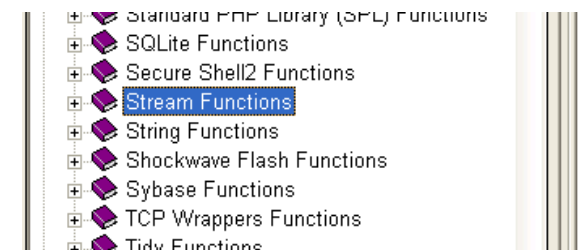


- Programski kôd i dalje upravlja kom. mehanizmom
 - stvaranje utičnica, slanje i primanje podataka, upravljanje grješkama, oslobađanje resursa ...
 - upravljanje komunikacijom poduzimanjem akcija nad mehanizmom (npr. akcije slanja ili primanja podataka)
- Upravljanje komunikacijskim protokolom skriveno u izvedbi komunikacijskih funkcija
 - prilikom stvaranja komunikacijskog mehanizma naznačuje se komunikacijski protokol koji treba koristiti
 - omotač oko komunikacijskog mehanizma brine se za prevođenje općenite akcije u oblik specifičan protokolu

Tokovi podataka u jeziku PHP (I)

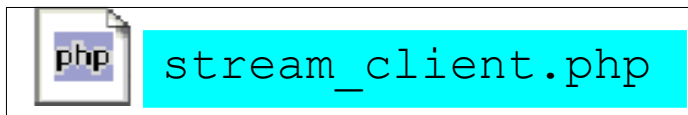


- Tok (*stream*) u PHP-u:
 - općenit mehanizam komunikacije sa slijednim izvorima podataka (datoteke, standardni ulaz i izlaz, mreža ...) koji dijele isti skup funkcija
 - Omotači oko tokova implementiraju pojedine komunikacijske protokole: za pristup datotečnom sustavu, Utičnicama (IP, UDP, TCP), webu (HTTP, HTTPS), udaljenim datotekama (FTP, FTPS) ...



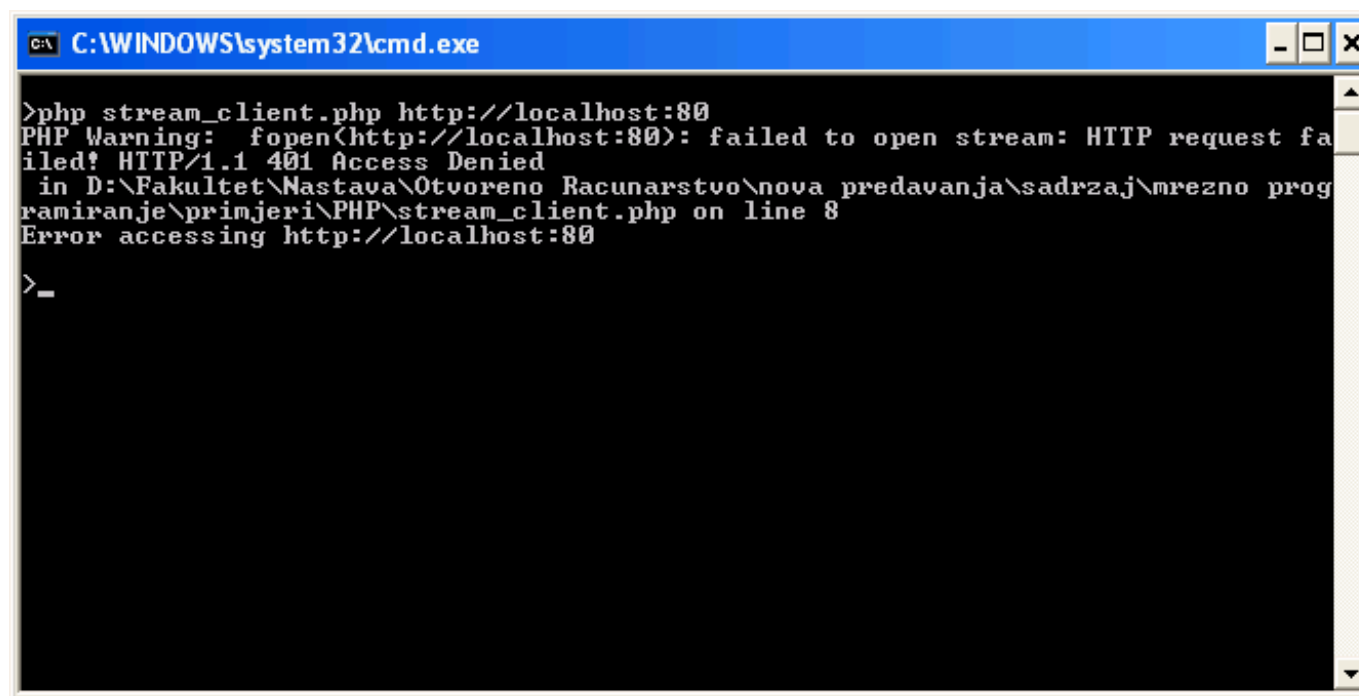
Tokovi podataka u jeziku PHP (II)

- U primjeru koda nigdje nema implementacije aplikacijskog protokola !
- Komunikacijski protokol definiran u trenutku pristupa udaljenom resursu na osnovu URL-a
- Implementacija protokola skrivena od programera



```
...  
$stream = fopen($argv[1], 'r');  
  
if(! $stream ) {  
    echo "Error accessing  
$argv[1]\n";  
    exit(1);  
}  
$content =  
stream_get_contents($stream);  
echo "\n$content\n";  
...
```

Tok podataka i URL



```
C:\WINDOWS\system32\cmd.exe

>php stream_client.php http://localhost:80
PHP Warning:  fopen(http://localhost:80): failed to open stream: HTTP request fa
iled! HTTP/1.1 401 Access Denied
in D:\Fakultet\Nastava\Otvoreno Racunarstvo\nova predavanja\sadrzaj\mrežno prog
ramiranje\primjeri\PHP\stream_client.php on line 8
Error accessing http://localhost:80

>_
```

Nije loš primjer ;) PHP klijent je samostalno obradio zahtjev za redirekciju na novi URL dan od strane web poslužitelja, te je kod drugog pristupa odbijen (kod greške 401) !

Java URL, *URLConnection*

- *URLConnection* je apstraktni razred
- *connection* je referenca na objekt konkretnog razreda koji mora nasljeđivati od razreda *URLConnection*
- Razred implementacije protokola (scheme) određuje se u trenutku stvaranja veze i dinamički učitava u VM – moguće dodati implementacije novih protokola
- Za HTTP shemu objekt implementacije protokola je razreda *HTTPURLConnection*

```
URL targetURL = new URL(args[0]);
URLConnection connection = targetURL.openConnection();
BufferedReader in = new BufferedReader(new InputStreamReader
    (connection.getInputStream()));
String lineIn = null;
while( (lineIn = in.readLine()) != null )
    System.out.println(lineIn);
```



url_client.java



Pitanja?