Socket

Maja Štula

ak. god. 2008/2009

Soket

- Soket je apstrakcija krajnje komunikacijske točke (*socket is an abstraction for an end point of communication*).
- Soketi omogućavaju dvosmjernu komunikaciju od točke do točke (point-to-point) između dva programa.
- To je metoda komunikacije između programa klijenta i programa servera na mreži.

2

Soket

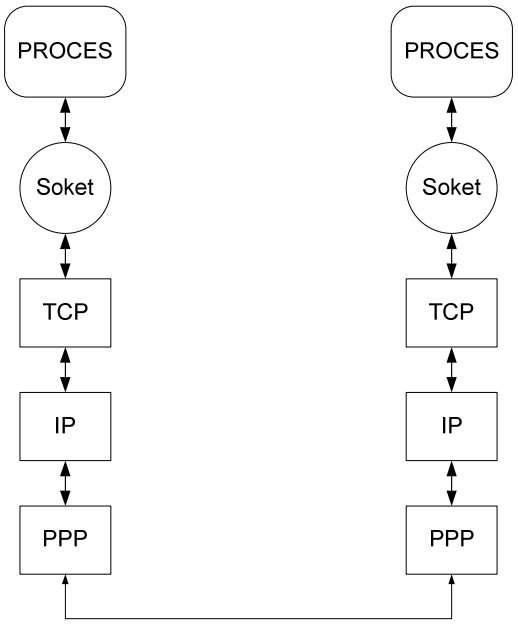
- Soket je TCP/IP API.
- Podržan je od većine OS-ova i programskih jezika.
- Nedostatci su:
  - složenost
  - podložnost koda greškama
  - svaki dio komunikacije treba biti eksplicitno programiran

3

TCP/IP API

- TCP/IP protokol softverski je sastavni dio operacijskog sustava.
- Operacijski sustav programeru pruža API preko kojeg programer koristit TCP/IP protokol.
- TCP/IP standard ne specificira kako se aplikacijski softver povezuje sa softverom TCP/IP protokola već samo predlaže funkcionalnost koju treba imati sučelje između aplikacije i softvera TCP/IP protokola.

4



5

TCP/IP API

- Prednost
  - fleksibilnost i prilagodljivost različitim OS-ovima
- Nedostatak
  - Detalji sučelja prema TCP/IP softveru razlikuju se od jednog do drugog OS-a.
  - Često korišteni API za TCP/IP:
    - Berkeley UNIX socket Interface
    - System V UNIX Transport Layer Interface (TLI)
    - Windows Sockets Interface
    - MacTCP

6

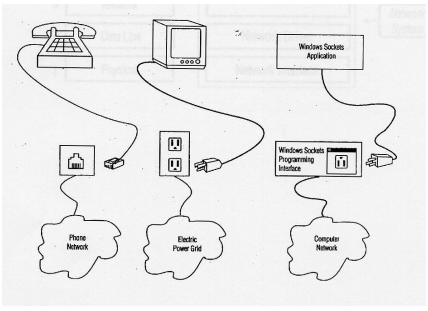
TCP/IP API

- Alokacija resursa za komunikaciju.
- Specificiranje lokalne i udaljenje krajnje točke komunikacije.
- Iniciranje konekcije (na strani klijenta).
- Čekanje na dolaznu konekciju (na strani server).
- Slanje i primanje podataka.
- Utvrđivanje dolaska podataka.
- Generiranje hitnih podataka.
- Obrada dolaznih hitnih podataka.
- Skladan raskid konekcije (FIN *handshake*).
- Obrada raskida konekcije s udaljenog mjesta.
- Prekid komunikacije.
- Obrada grešaka i prekida komunikacije.
- Oslobađanje lokalnih resursa nakon završetka komunikacije.

7

Soket

- Soket sučelje za TCP/IP protokol je razvijeno na Berkeley-u za UNIX.
- Od 1993. se koristi i na Windows operacijskim sustavima pod imenom WinSock. To je samo implementacija Berkeley soketa na Windows OS.



8

Tipovi soketa

- TCP soket (*stream socket*)
- UDP soket (*datagram socket*)
- IP soket (*row socket*) (ne na Winsock-u)

- Soket je posebni tip *handla* na file koji proces koristi da bi zatražio neku mrežnu uslugu od operacijskog sustava.

9

Funkcije soketa

- Da bi aplikacija mogla koristiti soket treba ga kreirati funkcijom:

int socket(int *family*, int *type*, int *proto*); UNIX
SOCKET socket(int *af*, int *type*, int *protocol*); Windows

- Poziv funkcije *socket* vraća handle na soket. Funkcija prima tri argumenta. Prvi argument određuje "obitelj" protokola koji će se koristiti (PF_INET (*protocol family Internet*) ili AF_INET (*adress family Internet*) za TCP/IP, INET-Internet) jer se soketi mogu koristiti i sa drugim protokolima, a ne samo sa TCP, UDP protokolima. Sljedeći argument određuje tip soketa (SOCK_STREAM, SOCK_DGRAM, SOCK_RAW (samo Unix)). Zadnji argument definira protokol i obično se postavlja u 0 koji znači korištenje *defaultnog* protokola.

10

Funkcije soketa

- Sljedeći korak na strani servera (soketi su server/klijent metoda mrežnog IPC-a) je vezivanje (*binding*) soketa na određenu IP adresu i port na kojem će soket biti otvoren. Ovaj korak se može preskočiti (ne mora) na strani klijenta jer će sa taj dio napraviti prilikom spajanja na server.

  int bind(int *sockfd*, const struct sockaddr* *name*, int *localaddrlen*);
  int bind(SOCKET *s*, const struct sockaddr* *name*, int *namelen*);

- Prvi argument funkcije je handle na soket, zatim ide pokazivač na strukturu koja sadrži adresu na koju se soket vezuje i zadnji argument je veličina adresne strukture (obično se samo stavi sizeof(sockaddr)). Funkcija vraća 0 u slučaju uspješnog *binding*, a -1 u slučaju greške.

11

Struktura *sockaddr*

- Struktura *sockaddr* je generička struktura soket adrese definirana ANSI C standardom. ANSI C je poslije preuzela i ISO organizacija, većina C kompajlera radi prema tom standardu i većina koda je temeljena na ANSI C.

```
struct sockaddr {
    uint8_t sa_len;             /*veličina strukture*/
    sa_family_t sa_family;      /*obitelj adresa: AF_xxx*/
    char sa_data[14];           /*adresa određena protokolom*/
} /*deklarirana <sys/socket.h> */

/*Windows*/
struct sockaddr {
    u_short sa_family;          /*obitelj adresa: AF_xxx*/
    char sa_data[14];           /*adresa određena protokolom*/
}; /*deklarirana Winsock.h */
```

12

Soket adresa

- Tip *sockaddr* automatski određuje strukturu adrese ovisno o postavljenoj "obitelji" adresa (AF_INET).
- Detaljnija specifikacija adrese daje se u strukturi *sockaddr_in* koja će se koristiti i na Windowsima i na Unixu ukoliko se radi s IPV4, a struktura *sockaddr_in6* ukoliko se radi s IPV6.

```
struct sockaddr_in
    {short sin_family;
    unsigned short sin_port;
    struct   in_addr sin_addr;
    char sin_zero[8]; };
```

```
struct sockaddr_in6
{ short sin6_family;
u_short sin6_port;
u_long sin6_flowinfo;
struct in6_addr sin6_addr;
u_long sin6_scope_id; };
```

13

Soket adresa

```
/* Windows */
    struct sockaddr_in
    {
    short sin_family;                  /* Obitelj adresa AF_INET, AF_INET6,..*/
    unsigned short sin_port;           /* Port */
    struct   in_addr sin_addr;         /* 32-bitna IPv4 adresa */
    char sin_zero[8];                  /* Popuna do veličine sockaddr strukture*/
    }; /*deklarirana u Winsock2.h za IPv4; ws2tcpip.h za IPv6. */

/*Unix*/
struct sockaddr_in {
    uint8_t sin_len;                   /*duljina strukture (16) */
    sa_family_t sin_family;            /* Obitelj adresa AF_INET, AF_INET6,..*/
    in_port_t sin_port;                /* 16-bitni TCP ili UDP broj porta*/
    struct in_addr  sin_addr;          /* 32-bitna IPv4 adresa */
    char       sin_zero[8];            /* Ne koristi se (padding) */
};  /* deklarirana in <netinet/in.h> */
```

14

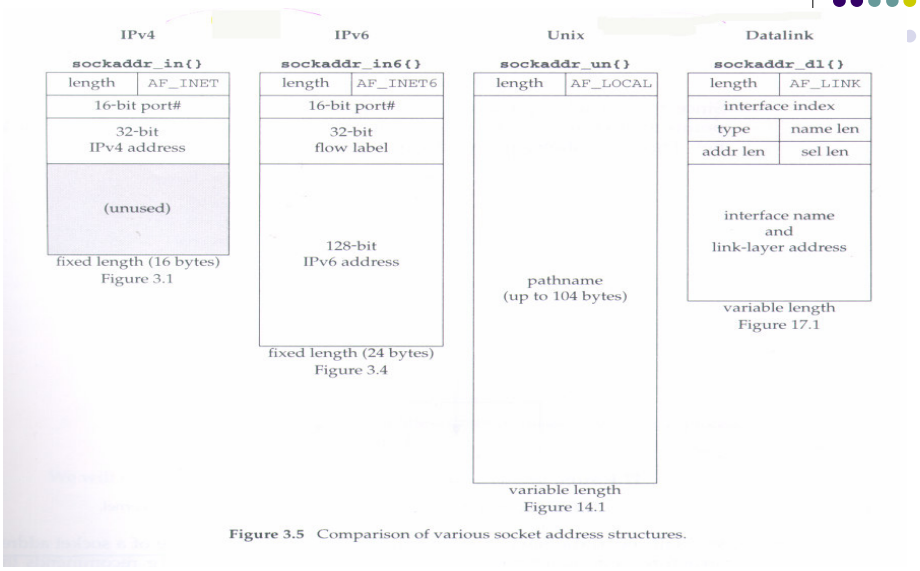


Figure 3.5 Comparison of various socket address structures.

15

Soket adresa

- Zapis 32-bitne IP adrese također je formatiran strukturom:

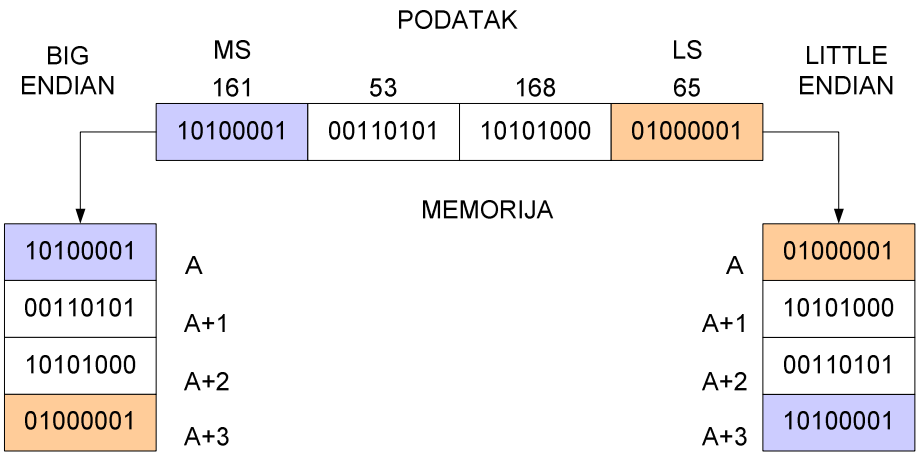
```
struct in_addr {
union {
        struct {  unsigned  char   s_b1, s_b2,   s_b3, s_b4;
                }  S_un_b;  /*Adresa formatirana kao 4 u_char-a*/
        struct { unsigned  short  s_w1, s_w2;
                }  S_un_w;  /*Adresa formatirana kao dva u_short-a*/
        unsigned long  S_addr;     /*Adresa formatirana kao jedan long*/
        } S_un;
    };
```

16

Mrežni redoslijed bajtova

- IP adresa i broj porta u adresnim strukturama soketa trebaju biti zapisani u mrežnom redoslijedu bajtova (*network byte order*).
- Npr. IP adresa 161.53.168.65 ima 32 bita odnosno 4 bajta i može se u memoriji zapisati na dva načina.
- Prvi način je da se najznačajniji bajt (*MS-most significant*) spremi prvi. Taj način se naziva *big-endian* zapis i koristi se kao NBO.
- *Little-endian* zapis sprema najmanje značajan bajt (*LS-least significant*) na prvo mjesto.

17



18

Mrežni redoslijed bajtova

- O čemu ovisi način pohrane 32 bitne vrijednosti u memoriju?

- PC arhitektura s Intelovim procesorom koristi *little-endian* zapis.
- RISC arhitekture i *mainframe* računala koriste *big-endian* zapis.
- Kao standard na mreži je prihvaćen *big-endian* zapis i svi paketi koji idu preko mreže trebaju imati zapisanu IP adresu (i port) u ovom obliku.

19

Mrežni redoslijed bajtova - funkcije

- Programeru su na raspolaganju funkcije koje omogućavaju pretvaranje jednog načina zapisa u drugi.
- Imenovanje funkcija je postavljeno na način da funkcije koje pretvaraju iz formata zapisa na *hostu* u mrežni format zapisa počinju s 'h' (*host byte order*), a funkcije koje pretvaraju iz mrežnog formata u format zapisa na *hostu* počinju s 'n' (*network byte order*).
- Funkcije za pretvaranje 16 bitnih vrijednosti (port) završavaju sa slovom 's' (*short*), a za pretvaranje 32-bitnih vrijednosti završavaju s 'l' (*long*).

u_short (Windows)

uint16_t htons(uint16_t); uint16_t ntohs(uint_16_t);
uint32_t htonl(uint32_t); uint32_t ntohl(uint32_t);

20

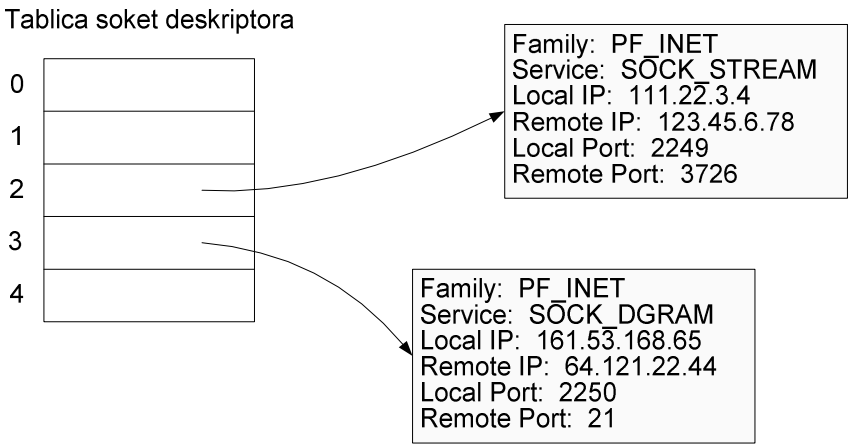
Adresa i port

- Ukoliko niste sigurni koju adresu koristiti (npr. imate više mrežnih sučelja) možete zatražiti od OS da procesu dodijeli adresu.
- Također možete zatražiti i da OS procesu dodijeli slobodni port što klijentske aplikacije u pravilu i rade.
- Za serverske aplikacije obično je definirano na kojem portu "slušaju" (npr. HTTP server – 80) pa trebaju koristiti upravo taj port jer će klijentske aplikacije pokušati otvarati konekciju prema serveru na tom portu.

```
struct sockaddr_in skaddr;
skaddr.sin_family = AF_INET;
skaddr.sin_addr.s_addr = htonl(INADDR_ANY);
skaddr.sin_port = htons(0);
```

21

Soket



22

Soket

- Nakon kreiranja soketa i vezivanja IP adrese i porta uz soket handle ili soket deskriptor, server za TCP konekciju (SOCK_STREAM tip soketa) “osluškuje” dolazne zahtjeve za konekcijama od strane klijenata funkcijom:

SOCKET
int listen(int *sockfd*, int *backlog*);

- Prvi argument funkcije je soket deskriptor, a drugi argument je maksimalni broj konekcija koji se može otvoriti na tom soketu.

23

Soket

- Nakon kreiranja soketa i vezivanja IP adrese i porta uz soket deskriptor, klijent za TCP konekciju (SOCK_STREAM tip soketa) pokušava uspostaviti konekciju prema nekom serveru funkcijom:

SOCKET
int connect(int *sockfd*, const struct sockaddr* *name*, int *namelen*);

- Prvi argument funkcije je soket deskriptor, drugi argument je pokazivač na strukturu s podacima o serveru (IP adresa i port) prema kojem se uspostavlja konekcija, a zadnji je argument veličina strukture.

24

Soket

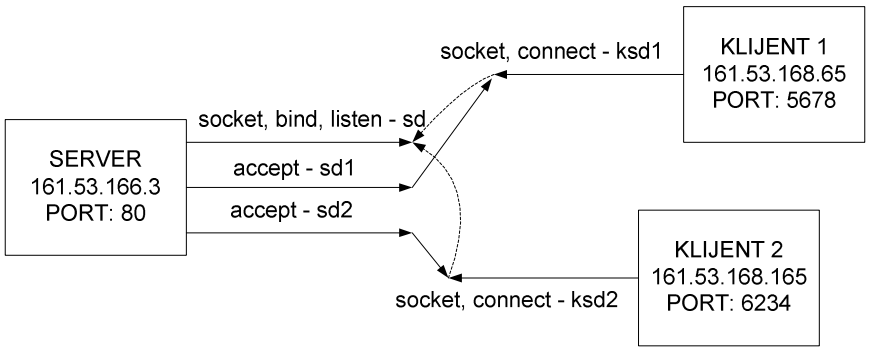
- Nakon što je server preko *listen* funkcije dobio zahtjev od klijenta za otvaranje konekcije ukoliko prihvaća klijentov zahtjev konačno se konekcija uspostavlja funkcijom:

SOCKET SOCKET
int accept(int *sockfd*, struct sockaddr* *addr*, int* *addrlen*);

- Prvi argument funkcije je soket deskriptor, drugi argument je pokazivač na strukturu s podacima o klijentu koji se spojio i treći argument je pokazivač na duljinu strukture koji je opcionalan. U slučaju uspješnog izvršavanja i ova funkcija vraća 0.

25

Višestruke konekcije na istom portu



26

Prijenos podataka

- Za prijenos podatak tipom soketa SOCK_STREAM koriste se sljedeće funkcije:

  /*Unix*/

  int read(int *sockfd*, const void **buf*, int *buflen*);

  int write(int *sockfd*, const void **buf*, int *buflen*);

  /*Windows*/

  int recv(SOCKET *s*, char* *buf*, int *len*, int *flags*);

  int send(SOCKET *s*, const char* *buf*, int *len*, int *flags*);

- Ukoliko su se uspješno izvršile povratne vrijednosti su broj pročitanih ili poslanih bajtova.

27

Prijenos podataka

- Za prijenos podatak tipom soketa SOCK_DGRAM koriste se sljedeće funkcije:

  int sendto(SOCKET *s*, const char* *buf*, int *len*, int *flags*,
  const struct sockaddr* *to*, int *tolen*);

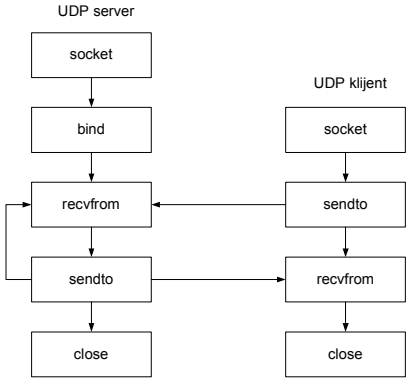
  int recvfrom(SOCKET *s*, const char* *buf*, int *len*, int *flags*,
  const struct sockaddr* *to*, int *tolen*);

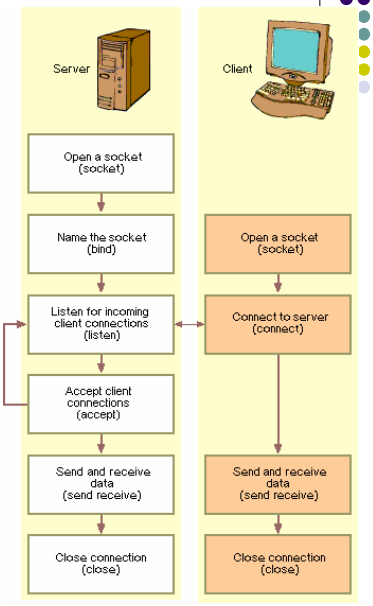
28

Zatvaranje soketa i konverzija string↔adresa

- Za zatvaranje soketa koristi se funkcija:
  int close(int *sockfd*);
- Ukoliko se IP adresa predaje kao argument programu obično je zapisana u string formatu. Funkcije za konverziju u in_addr strukturu:
  /*Unix*/
  int inet_aton(char *, struct in_addr *);
  /*Windows*/
  unsigned long inet_addr(const char* *cp*);
- Funkcija za konverziju iz in_addr formata u string format:
  char *inet_ntoa(struct in_addr);

29





30