Socket Programming: a Primer

Socket to me!

Why does one need sockets?

application
sockets
network
protocol

Feb. 23, 2001 EE122, UCB

So what exactly does a socket do?

- It is an API between applications and network protocol software
- Functions it provides:
  - Define an “end-point” for communication
  - Initiate and accept a connection
  - Send and receive data
  - Terminate a connection gracefully
- Supports multiple protocol families
  - Examples: Unix inter-process communication, TCP/IP
  - Only Internet sockets will be covered in this lecture

Feb. 23, 2001 EE122, UCB

Types of Sockets

- Two different types of sockets:
  - stream vs. datagram
- Stream socket (a.k.a. connection-oriented socket)
  - It provides reliable, connected networking service
  - Error free; no out-of-order packets (uses TCP)
  - Applications: telnet, http, …
- Datagram socket (a.k.a. connectionless socket)
  - It provides unreliable, best-effort networking service
  - Packets may be lost; may arrive out of order (uses UDP)
  - Applications: streaming audio/video, …

Feb. 23, 2001 EE122, UCB

How should one define a socket?

- To define an end-point of communication, one needs to specify
  - the family of protocol it uses (Internet vs. others)
  - addressing information (IP address + port number)
  - the type of service it provides (stream vs. datagram)
- Done in three steps
  - create a socket
  - define address and port number
  - associate address with the socket

Feb. 23, 2001 EE122, UCB

How to create a socket?

```c
#include <sys/types.h>
#include <sys/socket.h>

int sock;
sock = socket (AF_INET, SOCK_STREAM, 0); /* for stream */
sock = socket (AF_INET, SOCK_DGRAM, 0); /* for datagram */
```

- Notice that the socket descriptor is just a regular int!
- So it has the same usage as a file descriptor in Unix…

Feb. 23, 2001 EE122, UCB
How to define address?

- struct sockaddr {
  u_short sa_family;
  char sa_data[14];
}

- struct sockaddr_in {
  short sin_family;
  u_short sin_port;
  struct in_addr sin_addr;
  char sin_zero[8];
}

struct in_addr {
  u_long s_addr;
}

WARNING: Don’t forget to convert byte orders! htons, htonl, ntohs, ntohl

Bind a Socket

- bind( ): associate a socket descriptor with an address
  int bind (int sockfd, struct sockaddr *addr, int len);
- putting everything together
  int sockfd;
  struct sockaddr_in addr;
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  addr.sin_family = AF_INET;
  addr.sin_port = htons(5000); /* 0: randomly assigned by OS */
  addr.sin_addr.s_addr = htonl(INADDR_ANY); /* local address */
  bzero(&(addr.sin_zero[0]), 8); /* pad zeros */
  bind(sockfd, (struct sockaddr *)&addr, sizeof(struct sockaddr));

How to convert addresses?

- You also need to define address for the other end
- If you know its IP address
  myaddr.sin_addr.s_addr = inet_addr("128.32.138.240");
- If you know its name only:
  - need to perform a DNS lookup
    struct hostent *gethostbyname(char *name)

Using a Datagram Socket

- Sending data
  int sendto(int sockfd, void *msg, int msg_len, u_short flags,
             struct sockaddr *dest, int dest_len);
- Receiving data
  int recvfrom(int sockfd, void *msg, int msg_len, u_short flags,
               struct sockaddr *src, int src_len);

Using a Stream Socket

- establishing a connection
  int listen(int sockfd, int backlog)
  int connect(int sockfd, struct sockaddr *addr, int addr_len)
  int accept(int sockfd, void *addr, int *addrlen);

Using a Stream Socket (cont’d)

- Sending data
  int send(int sockfd, void *msg, int msg_len, u_short flags);
- Receiving data
  int recv(int sockfd, void *msg, int msg_len, u_short flags);
- Notice that no address is required!
A Quick Summary: Datagram

socket() to create socket
bind() to a receiving port
recvfrom() Sendto()

socket() to create socket
bind() to any port
recvfrom() Sendto()

Sample Codes: Datagram Client

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define Bfsize 1024

main(int argc, char *argv[]) {
    int sock;
    struct sockaddr_in client, server;
    struct hostent *host, *gethostbyname();
    sock=socket(AF_INET, SOCK_DGRAM, 0); /* open socket */
    client.sin_family=AF_INET;
    client.sin_addr.s_addr=htonl(INADDR_ANY); /* local addr */
    client.sin_port=htons(0);  /* any port # */
    bind(sock,(struct sockaddr *)&client,sizeof(client));
    host=gethostbyname(argv[1]);   /* get host name */
    memcpy((char *)&server.sin_addr, (char *)host->h_addr,
            host->h_length);
    server.sin_family=AF_INET;
    server.sin_port=htons(atoi(argv[2]));
    sendto(sock,msg,sizeof(msg),0,(struct sockaddr *)&server,
            sizeof(server));
    close(sock);
}
```

Sample Code: Datagram Server

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define  Bfsize 1024

main(int argc, char *argv[]) {
    int sock, length, count;
    struct sockaddr_in server, client;
    char buffer[Bfsize];
    sock=socket(AF_INET, SOCK_DGRAM,0);
    server.sin_family=AF_INET;
    server.sin_port=htons(atoi(argv[1])); /* listening port */
    bind(sock,(struct sockaddr *)&server,sizeof(server));
    count=recvfrom(sock, buffer, Bfsize, 0,
                                        (struct sockaddr *)&client,&length);
    printf("---> %s\n", buffer);
    close(sock);
}
```

Note that we don’t have to define client’s address here, because the server can receive from any one on sock. After return from recvfrom, client contains sender’s address information.
Sample Code: Stream Client

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define msg "hello ee122"

main(int argc, char *argv[
int sock;
struct sockaddr_in client, server;
struct hostent *host, *gethostname();
host=gethostbyname(argv[1]);
memcpy((char *)&server.sin_addr,(char *)host->h_addr,host->h_length);
server.sin_family=AF_INET;
server.sin_port=htons(atoi(argv[2]));
/* no bind is needed! */
sock=socket(AF_INET, SOCK_STREAM,0);
connect(sock,(struct sockaddr *)&server, sizeof(server));
send(sock,msg,sizeof(msg),0);
close(sock);
}
```

Sample Code: Stream Server

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define Bfsize 1024

main(int argc, char *argv[ ]
int new_sock, sock, length, count;
struct sockaddr_in server, client;
struct hostent *host, *gethostbyname();
char buffer[Bfsize];
sock=socket(AF_INET, SOCK_STREAM,0);
server.sin_family=AF_INET;
server.sin_addr.s_addr=htonl(INADDR_ANY);
server.sin_port=htons(atoi(argv[1]));
bind(sock, (struct sockaddr *)&server, sizeof(server));
listen(sock,1);
new_sock = accept(sock, (struct sockaddr *)&client, &length);
count=recv(new_sock,buffer,Bfsize,0);
printf("---> %s\n", buffer);
close(new_sock); close(sock);
```