Socket and Network Programming

By : Amir Hossein Payberah
payberah@yahoo.com
Contents

- Network Concepts
- Socket
- Related Data Structures
- Related System Calls and Commands
Network Layers

- Application
- Transport
- Network
- Data Link
- Physical

End-to-End Connection
Routing
Framing
Physical topology
Physical Layer

- It sends bits and receives bits.
Data Link Layer

- It ensures that messages are delivered to the proper device.
- It translates messages from the Network layer into bits for physical layer to transmit.
  - It formats the message into data frames.
  - It adds a header containing the hardware destination and source address.
Network Layer

- It is responsible for **routing** through an internetwork and for network addressing.
  - It is responsible for transporting traffic between devices that are not locally attached.
- It uses **software address**.

![Diagram of network address structure: netid and hostid, total 32 bits]
IP Addresses

Class A
- netid: 8
- hostid: 24

Class B
- netid: 16
- hostid: 16

Class C
- netid: 24
- hostid: 8
Transport Layer

- Flow control
  - It prevents a sending host on one side of connection from overflowing the buffers in the receiving host.

- Acknowledgment
  - It guarantees the data won’t be duplicated or lost.

- Windowing
  - It controls how much information is transferred from one end to the other.
Network Connections

- The Transport layer provide two types of connection:
  - Connection-less (UDP)
    - It is an unreliable connection.
  - Connection-oriented (TCP)
    - It handshakes before transfers information.
Connection-oriented

Create endpoint

Bind addr.

Specify queue

Wait for queue

read

write

Create endpoint

Connect to server

write

read
Connection-less

Create endpoint
Bind addr.
Read
Write

Create endpoint
Bind addr
Write
Read
Port Numbers

- It is possible for more than one user process at a time to be using either TCP or UDP.
- This requires some method for identifying the data associated with each user process.
5-Tuple Association

{ protocol, src port, src addr, dst port, dst addr }
Contents

- Network Concepts
- Socket
- Related Data Structures
- Related System Calls and Commands
Socket

- It is an interface between the application layer and other layers.

```c
main()
{
    FILE *fd;
    fd = fopen (…);
    process (fd);
    fclose (fd);
}

main()
{
    int sockfd;
    sockfd = socket (…);
    process (sockfd);
    close (sockfd);
}
```
Type of Sockets

- Stream Socket
  - Provide a reliable, sequenced, two-way connection.
  - This is use TCP Socket.

- Datagram Socket
  - A connection-less and unreliable connection.
  - This is use UDP Socket.

- Raw Socket
  - Used for internal network protocols.
Contents

- Network Concepts
- Socket
- Related Data Structures
- Related System Calls and Commands
```c
struct sockaddr {
    unsigned short     sa_family;  // address family, AF_xxx
    char                sa_data[14]; // 14 bytes of protocol address
};

struct sockaddr_in {
    short int          sin_family; // Address family
    unsigned short int sin_port;  // Port number
    struct in_addr     sin_addr;  // Internet address
    unsigned char      sin_zero[8]; // Same size as struct sockaddr
};

struct in_addr {
    unsigned long s_addr; // that's a 32-bit long, or 4 bytes
};
```
Contents

- Network Concepts
- Socket
- Related Data Structures
- Related System Calls and Commands
Byte Ordering Routines

- `htons()` // "Host to Network Short"
- `htonl()` // "Host to Network Long"
- `ntohs()` // "Network to Host Short"
- `ntohl()` // "Network to Host Long"
Address Conversion Routines

- `inet_addr_t inet_addr (char *cp);`
  - Converts the Internet host address cp from numbers-and-dots notation into binary data in network byte order.

- `int inet_aton (char *cp, struct in_addr *inp);`
  - Converts the Internet host address cp from numbers-and-dots notation into binary data.

- `char *inet_ntoa (struct in_addr in);`
  - Converts the Internet host address given in network byte order to a string in standard numbers-and-dots notation.
socket System Call

- int socket (int family, int type, int protocol);
- It creates the end point.
- Family:
  - AF_INET, AF_UNIX, ...
- Type:
  - SOCK_STREAM
  - SOCK_DGRAM
  - SOCK_RAW
**bind System Call**

- `int bind (int sockfd, struct sockaddr *addr, int addrlen);`
- It assigns a name to an unnamed socket.
connect System Call

- int connect (int sockfd, struct sockaddr *addr, int addrlen);
- A client use it to establish a connection with a server.
listen System Call

- `int listen (int sockfd, int backlog);`
- This system call is used by connection-oriented to indicate that it is willing to receive connections.
accept System Call

- `int accept (int sockfd, struct sockaddr *addr, int *len);`
- An incoming calls arrive at a listening socket, they will be queued until the server program ready to process them.
send and recv System Calls

- int send (int sockfd, char *buff, int len., int flag);
- int sendto (int sockfd, char *buff, int len., int flag, struct sockaddr *to, int addrlen);
- int recv (int sockfd, char *buff, int len., int flag);
- int recvfrom (int sockfd, char *buff, int len., int flag, struct sockaddr *from, int *addrlen);
Connection-oriented

socket() → bind() → listen() → accept() → recv() → send()

socket() → connect() → send() → recv()
Connection-less

- `socket()`
- `bind()`
- `recvfrom()`
- `sendto()`

Tehran Polytechnic University
Question?