Introduction to Socket Programming

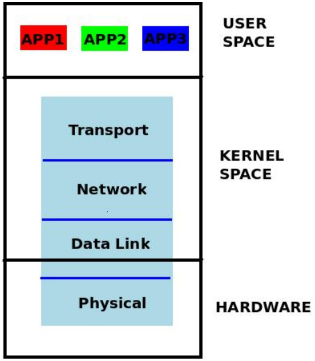
Sandip Chakraborty, K S Rao

Department of Computer Science and Engineering,

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

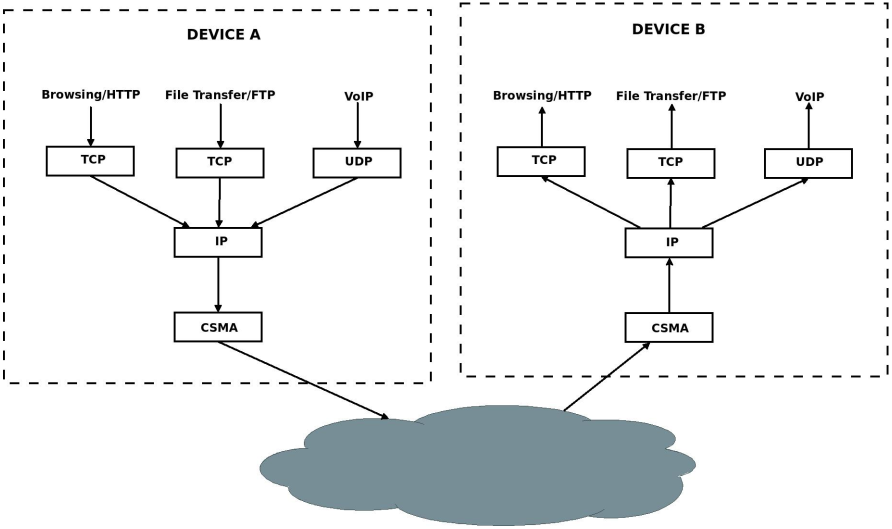
January 24, 2018

Connecting Network with Operating System

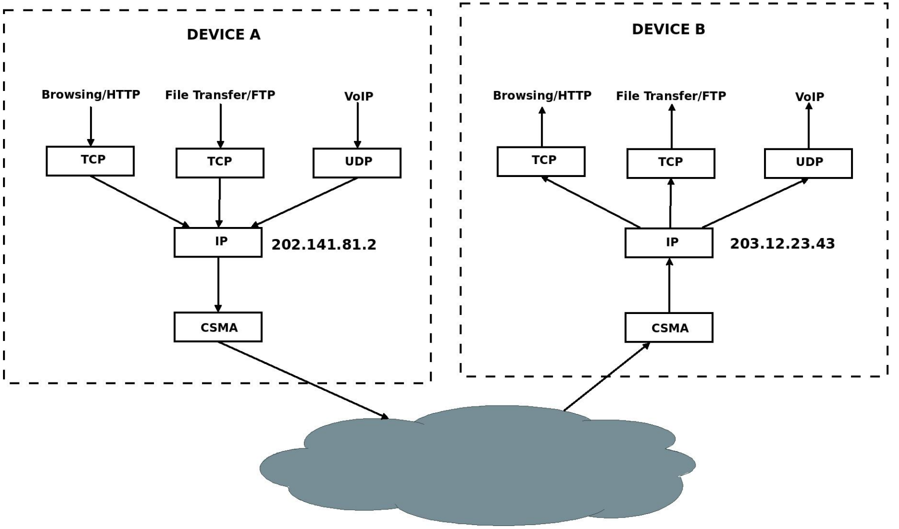


Check the `net` module (download Kernel source and check `/usr/src/linux/net`)!

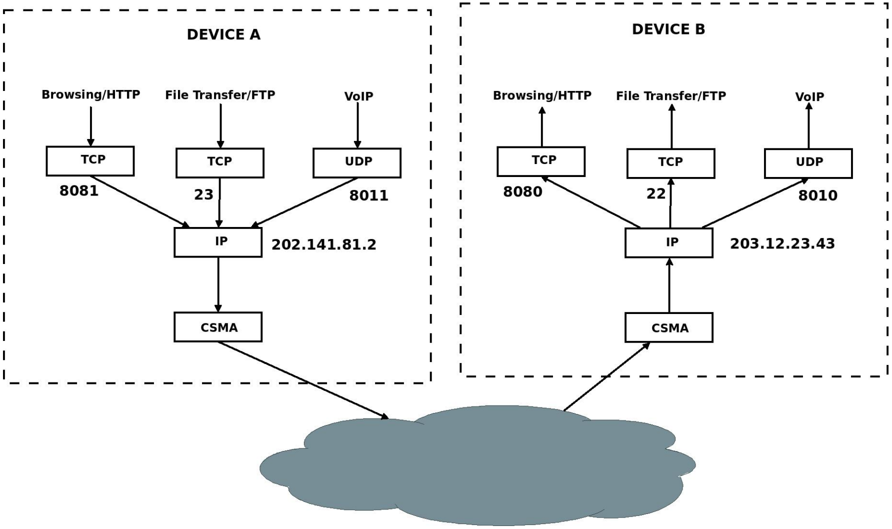
Application Multiplexing in TCP/IP



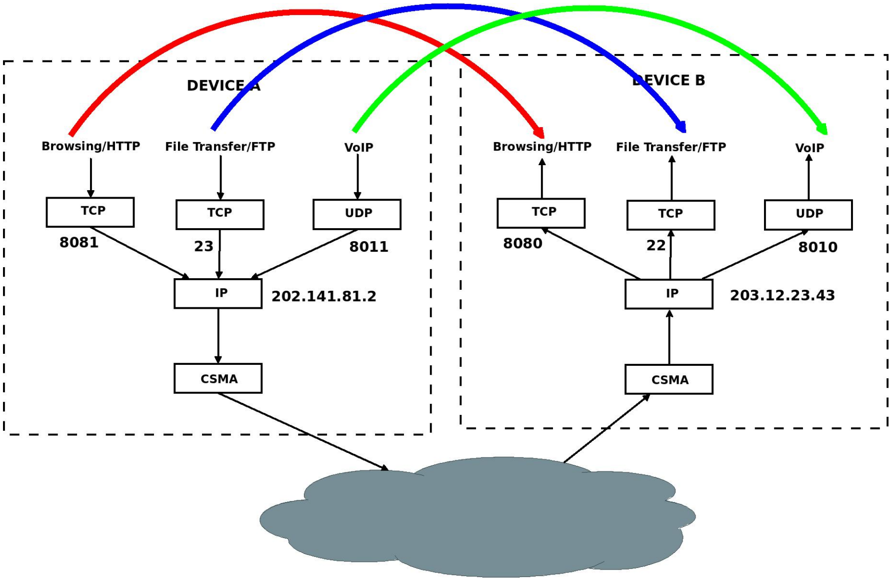
Application Multiplexing in TCP/IP



Application Multiplexing in TCP/IP

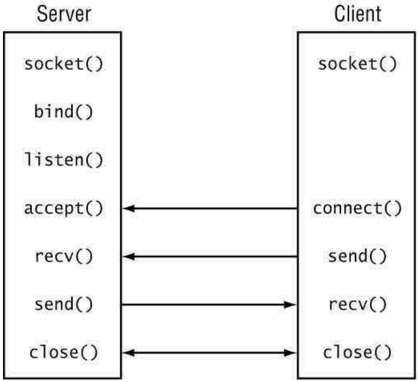


What are Sockets?



Socket Programming Framework/API

A set of **system calls** to get the service from TCP/IP protocol stack (`net` module in the OS kernel).



Socket Types

- The Internet is a trade-off between performance and reliability - **Can you say why?**
- Some application requires fine grained performance (example - multimedia applications), while others require reliability (example - file transfer)
- Transport layer supports two services - Reliable (TCP), and Unreliable (UDP)
- Two types of sockets:
  1. **Stream Socket (SOCK_STREAM)**: Reliable, connection oriented (TCP based)
  2. **Datagram Socket (SOCK_DGRAM)**: Unreliable, connection less (UDP based)

Socket API

- `int s = socket(domain, type, protocol);` - Create a socket
    - `domain`: Communication domain, typically used `AF_INET` (IPv4 Protocol)
    - `type`: Type of the socket - `SOCK_STREAM` or `SOCK_DGRAM`
    - `protocol`: Specifies protocols - usually set to 0 – **Explore!**
- `int status = bind(sockid, &addrport, size);` - Reserves a port for the socket.
    - `sockid`: Socket identifier
    - `addrport`: `struct sockaddr_in` - the (IP) address and port of the machine (address usually set to `INADDR_ANY` chooses a local address)
    - `size`: Size of the `sockaddr` structure

struct sockaddr_in

- sin_family : Address family, AF_INET for IPv4 Protocol
- sin_addr.s_addr: Source address, INADDR_ANY to choose the local address
- sin_port: The port number
- We need to use htons() function to convert the port number from **host byte order** to **network byte order**.

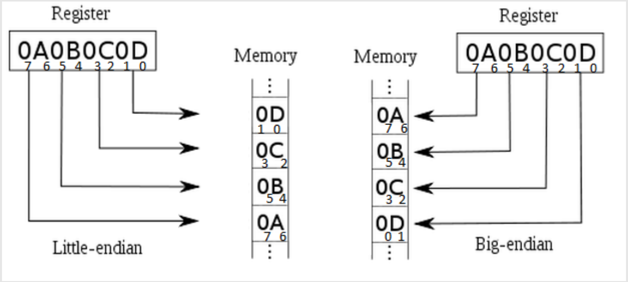
```
 struct sockaddr_in serveraddr;
int port = 3028;
serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = INADDR_ANY;
serveraddr.sin_port = htons(port);
```

Host Byte Order to Network Byte Order - Why?

- Little Endian and Big Endian System

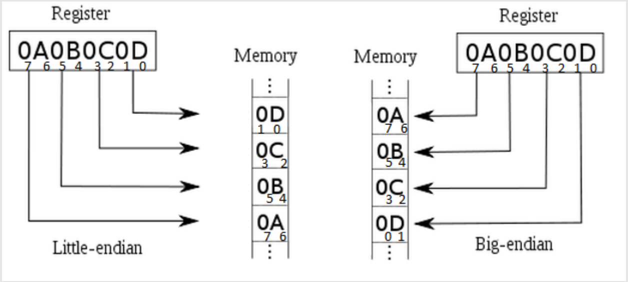
Host Byte Order to Network Byte Order - Why?

- Little Endian and Big Endian System



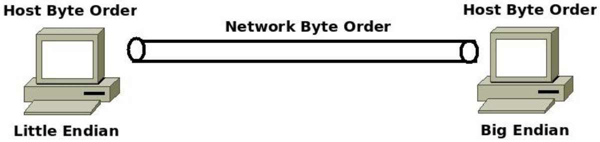
Host Byte Order to Network Byte Order - Why?

- Little Endian and Big Endian System



- **Assume a communication from a Little Endian to a Big Endian System or vice-versa!**

Host Byte Order to Network Byte Order - Why?



Listen and Accept a Socket Connection

```
 struct sockaddr_in cli_addr;
listen(sockfd,5);
clilen = sizeof(cli_addr);
newsockfd = accept(sockfd,(struct sockaddr *) &cli_addr,
&clilen);
```

- **Active Open** and **Passive Open**
  - The server needs to announce its address, remains in the open state and waits for any incoming connections - Passive Open
  - The client only opens a connection when there is a need for data transfer - Active Open
  - Connection is initiated by the client

Data Transfer through Sockets

1. For `SOCK_STREAM`:
   - `read(newsockfd,buffer,255);`
   - `write(newsockfd,‘‘I got your message’’,18);`
2. For `SOCK_DGRAM`:
   - `recvfrom(sock,buf,1024,0,(struct sockaddr *)&from,&fromlen);`
   - `sendto(sock,‘‘Got your message’’,17,0,(struct sockaddr *)&from,fromlen);`

Putting it All Together

Check the details and sample codes at
`http://www.linuxhowtos.org/C_C++/socket.htm`.

Socket Programming Tutorials

- Beej's Guide to Network Programming - `http://beej.us/guide/bgnet/`
- `http://cs.baylor.edu/~donahoo/practical/CSockets/textcode.html`
- `http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html`

Thank You