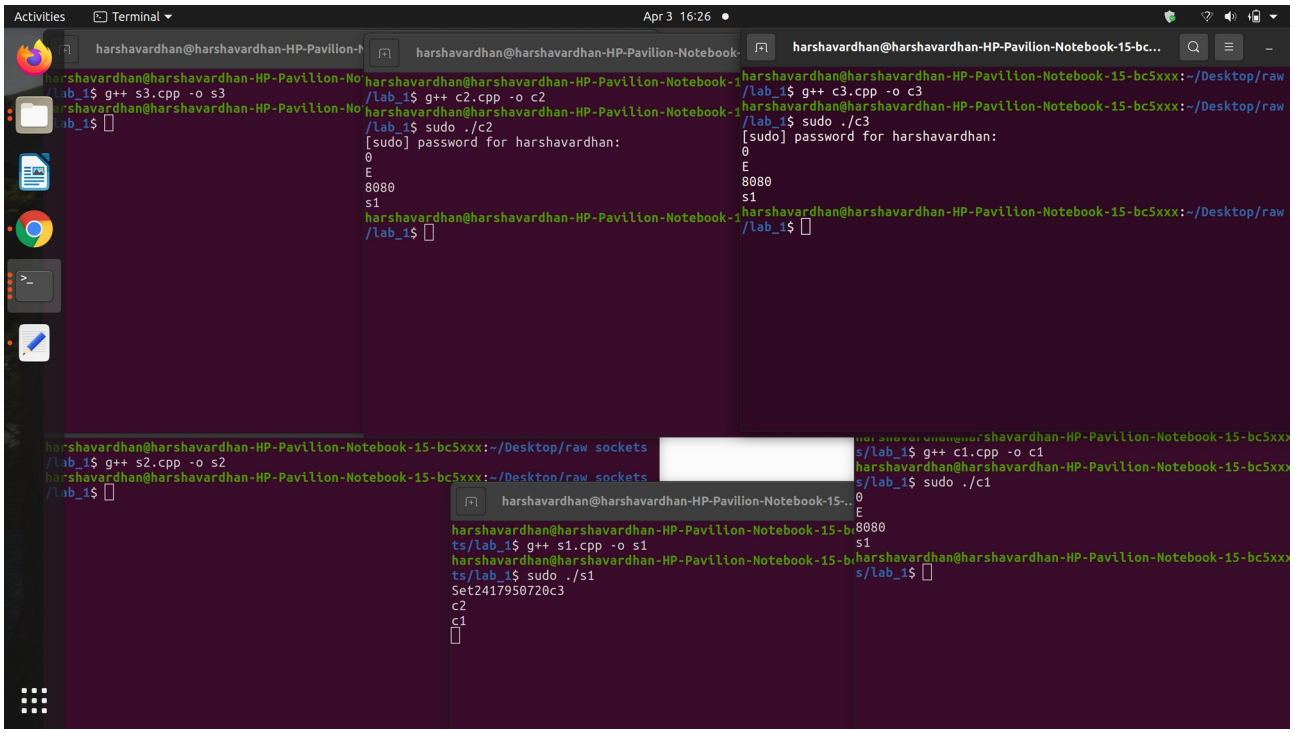
Q1 187241

using ip spoofing, sending port to clients and connecting using sfds (connection oriented)



s1

```
#include<time.h>
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/select.h>
#include<pthread.h>
#include<signal.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/shm.h>
#include<unistd.h>
#include<sys/un.h>
#include<netinet/ip.h>
#include<arpa/inet.h>
#include<errno.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<netinet/ether.h>
```

```
#include<netinet/udp.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<bits/stdc++.h>
using namespace std;
fd_set rfds;
#define BUF_LEN 1024
unsigned short csum(unsigned short *buf, int nwords)
{
        unsigned long sum;
   for (sum = 0; nwords > 0; nwords--)
        sum += *buf++;
   sum = (sum >> 16) + (sum & 0xffff);
   sum += (sum >> 16);
   return ~sum;
}
void print_ipheader(struct iphdr* ip)
{
        cout<<"------------------------\n";
        cout<<"Printing IP header....\n";
        cout<<"IP version:"<<(unsigned int)ip->version<<endl;
        cout<<"IP header length:"<<(unsigned int)ip->ihl<<endl;

        cout<<"Type of service:"<<(unsigned int)ip->tos<<endl;
        cout<<"Total ip packet length:"<<ntohs(ip->tot_len)<<endl;
        cout<<"Packet id:"<<ntohs(ip->id)<<endl;
        cout<<"Time to leave :"<<(unsigned int)ip->ttl<<endl;
        cout<<"Protocol:"<<(unsigned int)ip->protocol<<endl;
        cout<<"Check:"<<ip->check<<endl;
        cout<<"Source ip:"<<inet_ntoa(*(in_addr*)&ip->saddr)<<endl;
        //printf("%pI4\n",&ip->saddr );
        cout<<"Destination ip:"<<inet_ntoa(*(in_addr*)&ip->daddr)<<endl;
        cout<<"End of IP header\n";
        cout<<"------------------------\n";
}




int main()
{
        int s = socket (PF_INET, SOCK_RAW,  14);
        if(s<0)
        cout<<"Hi";
        char buff[4096]="s1";
        struct iphdr *iph = (struct iphdr *) buff;
        struct sockaddr_in sin;
        sin.sin_family = AF_INET;
        sin.sin_port = htons (8081);
        sin.sin_addr.s_addr = inet_addr ("127.0.0.1");
        memset(&buff,0,4096);
        iph->ihl = 5;
```

```
	iph->version = 4;
	iph->tos = 0;
	iph->tot_len = 1024;
	iph->id = htonl (54321);	//Id of this packet
	iph->frag_off = 0;
	iph->ttl = 255;
	iph->protocol =  1;
	iph->check = 0;			//Set to 0 before calculating checksum
	iph->saddr = inet_addr ( "0.0.31.144" );	//Spoof the source ip address
	iph->daddr = sin.sin_addr.s_addr;
	iph->check = csum ((unsigned short *) buff, iph->tot_len);
	int opt=1;
	const int *val = &opt;
	if (setsockopt (s, IPPROTO_IP, IP_HDRINCL, val, sizeof (opt)) < 0)
	cout<<"Not set";
	else
	cout<<"Set";

	struct iphdr *ipp;
	ipp=(struct iphdr*)buff;

//	print_ipheader(ipp);

	cout<<ipp->saddr;


	int sfd1,sfd2,sfd3, nsfd, valread;
	struct sockaddr_in address;
	int addrlen = sizeof(address);

	if ((sfd1 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd1, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8080 );
	if (bind(sfd1, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(sfd1, 3) < 0)
	{
		perror("listen");
```

```
		exit(EXIT_FAILURE);
	}

	if ((sfd2 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd2, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8081 );
	if (bind(sfd2, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(sfd2, 3) < 0)
	{
		perror("listen");
		exit(EXIT_FAILURE);
	}

	if ((sfd3 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd3, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8082 );
	if (bind(sfd3, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(sfd3, 3) < 0)
	{
		perror("listen");
		exit(EXIT_FAILURE);
	}
```

```
for(int i=0;i<3;i++)
{
	sendto(s,buff,iph->tot_len,0,(struct sockaddr *) &sin,sizeof (sin));
	if(i==0)
	iph->saddr = inet_addr ( "0.0.31.145" );
	if(i==1)
	iph->saddr = inet_addr ( "0.0.31.146" );
}

int sfd[3];
sfd[0]=sfd1;
sfd[1]=sfd2;
sfd[2]=sfd3;
while(1)
{
	FD_ZERO(&rfds);
	FD_SET(sfd[0],&rfds);
	FD_SET(sfd[1],&rfds);
	FD_SET(sfd[2],&rfds);
	int ma=-1;
	for(int i=0;i<3;i++)
	{
		if(ma<sfd[i])
		ma=sfd[i];
	}
	int count = select(ma+1,&rfds,NULL,NULL,NULL);

	if(count>0)
	{

		for(int i=0;i<3;i++)
		{
			if(FD_ISSET(sfd[i],&rfds))
			{
				char buffer[1024];
				if ((nsfd = accept(sfd[i], (struct sockaddr *)&address,
(socklen_t*)&addrlen))<0)
				{
					perror("accept");
					exit(EXIT_FAILURE);
				}
				int valread = recv( nsfd , buffer, 1024,0);
				cout<<buffer<<endl;
				string s="s1";

				char *hello = &s[0];
				send(nsfd, hello , s.length() , 0 );
```

```
                        }

                    }

                }

        }



}
```

S2

```
#include<time.h>
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/select.h>
#include<pthread.h>
#include<signal.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/shm.h>
#include<unistd.h>
#include<sys/un.h>
#include<netinet/ip.h>
#include<arpa/inet.h>
#include<errno.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<netinet/ether.h>
#include<netinet/udp.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<bits/stdc++.h>
using namespace std;
fd_set rfds;
#define BUF_LEN 1024
unsigned short csum(unsigned short *buf, int nwords)
{
        unsigned long sum;
```

```
    for (sum = 0; nwords > 0; nwords--)
            sum += *buf++;
    sum = (sum >> 16) + (sum & 0xffff);
    sum += (sum >> 16);
    return ~sum;
}
void print_ipheader(struct iphdr* ip)
{
            cout<<"------------------------\n";
            cout<<"Printing IP header....\n";
            cout<<"IP version:"<<(unsigned int)ip->version<<endl;
            cout<<"IP header length:"<<(unsigned int)ip->ihl<<endl;

            cout<<"Type of service:"<<(unsigned int)ip->tos<<endl;
            cout<<"Total ip packet length:"<<ntohs(ip->tot_len)<<endl;
            cout<<"Packet id:"<<ntohs(ip->id)<<endl;
            cout<<"Time to leave :"<<(unsigned int)ip->ttl<<endl;
            cout<<"Protocol:"<<(unsigned int)ip->protocol<<endl;
            cout<<"Check:"<<ip->check<<endl;
            cout<<"Source ip:"<<inet_ntoa(*(in_addr*)&ip->saddr)<<endl;
            //printf("%pI4\n",&ip->saddr );
            cout<<"Destination ip:"<<inet_ntoa(*(in_addr*)&ip->daddr)<<endl;
            cout<<"End of IP header\n";
            cout<<"------------------------\n";
}




int main()
{
            int s = socket (PF_INET, SOCK_RAW,  15);
            if(s<0)
            cout<<"Hi";
            char buff[4096]="s1";
            struct iphdr *iph = (struct iphdr *) buff;
            struct sockaddr_in sin;
            sin.sin_family = AF_INET;
            sin.sin_port = htons (8081);
            sin.sin_addr.s_addr = inet_addr ("127.0.0.1");
            memset(&buff,0,4096);
            iph->ihl = 5;
            iph->version = 4;
            iph->tos = 0;
            iph->tot_len = 1024;
            iph->id = htonl (54321);        //Id of this packet
            iph->frag_off = 0;
            iph->ttl = 255;
            iph->protocol =  1;
            iph->check = 0;                 //Set to 0 before calculating checksum
            iph->saddr = inet_addr ( "0.0.31.144" );       //Spoof the source ip address
            iph->daddr = sin.sin_addr.s_addr;
```

```
	iph->check = csum ((unsigned short *) buff, iph->tot_len);
	int opt=1;
	const int *val = &opt;
	if (setsockopt (s, IPPROTO_IP, IP_HDRINCL, val, sizeof (opt)) < 0)
	cout<<"Not set";
	else
	cout<<"Set";

	struct iphdr *ipp;
	ipp=(struct iphdr*)buff;

//	print_ipheader(ipp);

  //	cout<<ipp->saddr;


	int sfd1,sfd2,sfd3, nsfd, valread;
	struct sockaddr_in address;
	int addrlen = sizeof(address);

	if ((sfd1 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd1, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8080 );
	if (bind(sfd1, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(sfd1, 3) < 0)
	{
		perror("listen");
		exit(EXIT_FAILURE);
	}

	if ((sfd2 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd2, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
```

```
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8081 );
	if (bind(sfd2, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(sfd2, 3) < 0)
	{
		perror("listen");
		exit(EXIT_FAILURE);
	}

	if ((sfd3 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd3, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8082 );
	if (bind(sfd3, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(sfd3, 3) < 0)
	{
		perror("listen");
		exit(EXIT_FAILURE);
	}



	for(int i=0;i<3;i++)
	{
		sendto(s,buff,iph->tot_len,0,(struct sockaddr *) &sin,sizeof (sin));
		if(i==0)
		iph->saddr = inet_addr ( "0.0.31.145" );
		if(i==1)
		iph->saddr = inet_addr ( "0.0.31.146" );
```

```
	}

	int sfd[3];
	sfd[0]=sfd1;
	sfd[1]=sfd2;
	sfd[2]=sfd3;
	while(1)
	{
		FD_ZERO(&rfds);
		FD_SET(sfd[0],&rfds);
		FD_SET(sfd[1],&rfds);
		FD_SET(sfd[2],&rfds);
		int ma=-1;
		for(int i=0;i<3;i++)
		{
			if(ma<sfd[i])
			ma=sfd[i];
		}
		int count = select(ma+1,&rfds,NULL,NULL,NULL);

		if(count>0)
		{

			for(int i=0;i<3;i++)
			{
				if(FD_ISSET(sfd[i],&rfds))
				{
					char buffer[1024];
					if ((nsfd = accept(sfd[i], (struct sockaddr *)&address,
(socklen_t*)&addrlen))<0)
					{
						perror("accept");
						exit(EXIT_FAILURE);
					}
					int valread = recv( nsfd , buffer, 1024,0);
					cout<<buffer<<endl;
					string s="s2";

					char *hello = &s[0];
					send(nsfd, hello , s.length() , 0 );




				}


			}


		}
```

```
        }



}
```

S3

```
#include<time.h>
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/select.h>
#include<pthread.h>
#include<signal.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/shm.h>
#include<unistd.h>
#include<sys/un.h>
#include<netinet/ip.h>
#include<arpa/inet.h>
#include<errno.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<netinet/ether.h>
#include<netinet/udp.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<bits/stdc++.h>
using namespace std;
fd_set rfds;
#define BUF_LEN 1024
unsigned short csum(unsigned short *buf, int nwords)
{
        unsigned long sum;
   for (sum = 0; nwords > 0; nwords--)
        sum += *buf++;
   sum = (sum >> 16) + (sum & 0xffff);
   sum += (sum >> 16);
   return ~sum;
}
void print_ipheader(struct iphdr* ip)
{
        cout<<"------------------------\n";
        cout<<"Printing IP header....\n";
        cout<<"IP version:"<<(unsigned int)ip->version<<endl;
```

```
	cout<<"IP header length:"<<(unsigned int)ip->ihl<<endl;

	cout<<"Type of service:"<<(unsigned int)ip->tos<<endl;
	cout<<"Total ip packet length:"<<ntohs(ip->tot_len)<<endl;
	cout<<"Packet id:"<<ntohs(ip->id)<<endl;
	cout<<"Time to leave :"<<(unsigned int)ip->ttl<<endl;
	cout<<"Protocol:"<<(unsigned int)ip->protocol<<endl;
	cout<<"Check:"<<ip->check<<endl;
	cout<<"Source ip:"<<inet_ntoa(*(in_addr*)&ip->saddr)<<endl;
	//printf("%pI4\n",&ip->saddr );
	cout<<"Destination ip:"<<inet_ntoa(*(in_addr*)&ip->daddr)<<endl;
	cout<<"End of IP header\n";
	cout<<"------------------------\n";
}




int main()
{
	int s = socket (PF_INET, SOCK_RAW,  16);
	if(s<0)
	cout<<"Hi";
	char buff[4096]="s1";
	struct iphdr *iph = (struct iphdr *) buff;
	struct sockaddr_in sin;
	sin.sin_family = AF_INET;
	sin.sin_port = htons (8081);
	sin.sin_addr.s_addr = inet_addr ("127.0.0.1");
	memset(&buff,0,4096);
	iph->ihl = 5;
	iph->version = 4;
	iph->tos = 0;
	iph->tot_len = 1024;
	iph->id = htonl (54321);	//Id of this packet
	iph->frag_off = 0;
	iph->ttl = 255;
	iph->protocol =  1;
	iph->check = 0;			//Set to 0 before calculating checksum
	iph->saddr = inet_addr ( "0.0.31.144" );	//Spoof the source ip address
	iph->daddr = sin.sin_addr.s_addr;
	iph->check = csum ((unsigned short *) buff, iph->tot_len);
	int opt=1;
	const int *val = &opt;
	if (setsockopt (s, IPPROTO_IP, IP_HDRINCL, val, sizeof (opt)) < 0)
	cout<<"Not set";
	else
	cout<<"Set";

	struct iphdr *ipp;
	ipp=(struct iphdr*)buff;
```

```
//	print_ipheader(ipp);

  //	cout<<ipp->saddr;


	int sfd1,sfd2,sfd3, nsfd, valread;
	struct sockaddr_in address;
	int addrlen = sizeof(address);

	if ((sfd1 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd1, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8080 );
	if (bind(sfd1, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(sfd1, 3) < 0)
	{
		perror("listen");
		exit(EXIT_FAILURE);
	}

	if ((sfd2 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd2, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8081 );
	if (bind(sfd2, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
```

```
	}
	if (listen(sfd2, 3) < 0)
	{
		perror("listen");
		exit(EXIT_FAILURE);
	}

	if ((sfd3 = socket(AF_INET, SOCK_STREAM, 0)) == 0)
	{
		perror("socket failed");
		exit(EXIT_FAILURE);
	}
	if (setsockopt(sfd3, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
	{
		perror("setsockopt");
		exit(EXIT_FAILURE);
	}
	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons( 8082 );
	if (bind(sfd3, (struct sockaddr *)&address, sizeof(address))<0)
	{
		perror("bind failed");
		exit(EXIT_FAILURE);
	}
	if (listen(sfd3, 3) < 0)
	{
		perror("listen");
		exit(EXIT_FAILURE);
	}



	for(int i=0;i<3;i++)
	{
		sendto(s,buff,iph->tot_len,0,(struct sockaddr *) &sin,sizeof (sin));
		if(i==0)
		iph->saddr = inet_addr ( "0.0.31.145" );
		if(i==1)
		iph->saddr = inet_addr ( "0.0.31.146" );
	}

	int sfd[3];
	sfd[0]=sfd1;
	sfd[1]=sfd2;
	sfd[2]=sfd3;
	while(1)
	{
		FD_ZERO(&rfds);
		FD_SET(sfd[0],&rfds);
		FD_SET(sfd[1],&rfds);
```

```
FD_SET(sfd[2],&rfds);
int ma=-1;
for(int i=0;i<3;i++)
{
	if(ma<sfd[i])
	ma=sfd[i];
}
int count = select(ma+1,&rfds,NULL,NULL,NULL);

if(count>0)
{

	for(int i=0;i<3;i++)
	{
		if(FD_ISSET(sfd[i],&rfds))
		{
			char buffer[1024];
			if ((nsfd = accept(sfd[i], (struct sockaddr *)&address,
(socklen_t*)&addrlen))<0)
			{
				perror("accept");
				exit(EXIT_FAILURE);
			}
			int valread = recv( nsfd , buffer, 1024,0);
			cout<<buffer<<endl;
			string s="s3";

			char *hello = &s[0];
			send(nsfd, hello , s.length() , 0 );




		}


	}


}

	}


}
```

C1

```
#include<time.h>
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/select.h>
#include<pthread.h>
#include<signal.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/shm.h>
#include<unistd.h>
#include<sys/un.h>
#include<netinet/ip.h>
#include<arpa/inet.h>
#include<errno.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<netinet/ether.h>
#include<netinet/udp.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<bits/stdc++.h>
using namespace std;

#define BUF_LEN 1024

void print_ipheader(struct iphdr* ip)
{
	cout<<"------------------------\n";
	cout<<"Printing IP header....\n";
	cout<<"IP version:"<<(unsigned int)ip->version<<endl;
	cout<<"IP header length:"<<(unsigned int)ip->ihl<<endl;

	cout<<"Type of service:"<<(unsigned int)ip->tos<<endl;
	cout<<"Total ip packet length:"<<ntohs(ip->tot_len)<<endl;
	cout<<"Packet id:"<<ntohs(ip->id)<<endl;
	cout<<"Time to leave :"<<(unsigned int)ip->ttl<<endl;
	cout<<"Protocol:"<<(unsigned int)ip->protocol<<endl;
	cout<<"Check:"<<ip->check<<endl;
	cout<<"Source ip:"<<inet_ntoa(*(in_addr*)&ip->saddr)<<endl;
	cout<<ip->saddr<<endl;

	//printf("%pI4\n",&ip->saddr );
	cout<<"Destination ip:"<<inet_ntoa(*(in_addr*)&ip->daddr)<<endl;
	cout<<"End of IP header\n";
	cout<<"------------------------\n";
}

void reverse(char* str)
```

```
{
    // l for swap with index 2
    int l = 2;
    int r = strlen(str) - 2;

    // swap with in two-2 pair
    while (l < r) {
        swap(str[l++], str[r++]);
        swap(str[l++], str[r]);
        r = r - 3;
    }
}

// function to conversion and print
// the hexadecimal value
void ipToHexa(int addr)
{
    char str[15];

    // convert integer to string for reverse
    sprintf(str, "0x%08x", addr);

    // reverse for get actual hexadecimal
    // number without reverse it will
    // print 0x0100007f for 127.0.0.1
    reverse(str);

    // print string
    cout << str << "\n";
}


int main()
{
        int rsfd = socket (PF_INET, SOCK_RAW, 1);
        int n = 0 ;
        int one = 1;
        const int *val = &one;
        if (setsockopt (rsfd, IPPROTO_IP, IP_HDRINCL, val, sizeof (one)) < 0)
        cout<<"Not set";
        char buff[4096];
        struct iphdr *iph = (struct iphdr *)buff;
        struct sockaddr_in sin,sin2;
        socklen_t len = sizeof(sin2);
        memset(&sin,0,sizeof(sin));
        sin.sin_family = AF_INET;
        sin.sin_port = htons (6000);
        sin.sin_addr.s_addr = htonl(INADDR_ANY);
        bind(rsfd,(struct sockaddr *) &sin,sizeof(sin));
        vector<int>v;
        while(1)
        {
```

```
        recvfrom(rsfd,buff,4096,0,(struct sockaddr *) &sin,&len);
        cout<<sin.sin_port<<endl;
        cout<<(char*)buff<<endl;
        struct iphdr *ipp;
        ipp=(struct iphdr*)buff;
        int p=0;
        string s=inet_ntoa(*(in_addr*)&ipp->saddr);
        string str="";
        int a=0;
        for(int i=0;s[i];i++)
        {


                if(s[i]!='.')
                {
                        str+=s[i];
                }
                else
                {
                        p=p*256+stoi(str);
                        str="";
                }

        }
        p=p*256+stoi(str);
        cout<<p<<endl;
        v.push_back(p);

        break;

}
int sfd = 0, valread;
struct sockaddr_in address;

char buffer[1024] = {0};
if ((sfd= socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
        printf("\n Socket creation error \n");
        return -1;
}

address.sin_family = AF_INET;
address.sin_port = htons(v[0]);

// Convert IPv4 and IPv6 addresses from text to binary form
if(inet_pton(AF_INET, "127.0.0.1", &address.sin_addr)<=0)
{
        printf("\nInvalid address/ Address not supported \n");
        return -1;
}

if (connect(sfd, (struct sockaddr *)&address, sizeof(address)) < 0)
```

```
        {
                printf("\nConnection Failed \n");
                return -1;
        }
//      while(1){
        string s="c1";
        char *hello = &s[0];
        send(sfd, hello , s.length() , 0 );
        valread = recv( sfd , buffer, 1024,0);
        cout<<buffer<<endl;


//      }


}
```

C2

```
#include<time.h>
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/select.h>
#include<pthread.h>
#include<signal.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/shm.h>
#include<unistd.h>
#include<sys/un.h>
#include<netinet/ip.h>
#include<arpa/inet.h>
#include<errno.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<netinet/ether.h>
#include<netinet/udp.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<bits/stdc++.h>
using namespace std;

#define BUF_LEN 1024

void print_ipheader(struct iphdr* ip)
{
        cout<<"------------------------\n";
```

```
        cout<<"Printing IP header....\n";
        cout<<"IP version:"<<(unsigned int)ip->version<<endl;
        cout<<"IP header length:"<<(unsigned int)ip->ihl<<endl;

        cout<<"Type of service:"<<(unsigned int)ip->tos<<endl;
        cout<<"Total ip packet length:"<<ntohs(ip->tot_len)<<endl;
        cout<<"Packet id:"<<ntohs(ip->id)<<endl;
        cout<<"Time to leave :"<<(unsigned int)ip->ttl<<endl;
        cout<<"Protocol:"<<(unsigned int)ip->protocol<<endl;
        cout<<"Check:"<<ip->check<<endl;
        cout<<"Source ip:"<<inet_ntoa(*(in_addr*)&ip->saddr)<<endl;
        cout<<ip->saddr<<endl;

        //printf("%pI4\n",&ip->saddr );
        cout<<"Destination ip:"<<inet_ntoa(*(in_addr*)&ip->daddr)<<endl;
        cout<<"End of IP header\n";
        cout<<"------------------------\n";
}

void reverse(char* str)
{
    // l for swap with index 2
    int l = 2;
    int r = strlen(str) - 2;

    // swap with in two-2 pair
    while (l < r) {
        swap(str[l++], str[r++]);
        swap(str[l++], str[r]);
        r = r - 3;
    }
}

// function to conversion and print
// the hexadecimal value
void ipToHexa(int addr)
{
    char str[15];

    // convert integer to string for reverse
    sprintf(str, "0x%08x", addr);

    // reverse for get actual hexadecimal
    // number without reverse it will
    // print 0x0100007f for 127.0.0.1
    reverse(str);

    // print string
    cout << str << "\n";
}
```

```
int main()
{
	int rsfd = socket (PF_INET, SOCK_RAW, 1);
	int n = 0 ;
	int one = 1;
	const int *val = &one;
	if (setsockopt (rsfd, IPPROTO_IP, IP_HDRINCL, val, sizeof (one)) < 0)
	cout<<"Not set";
	char buff[4096];
	struct iphdr *iph = (struct iphdr *)buff;
	struct sockaddr_in sin,sin2;
	socklen_t len = sizeof(sin2);
	memset(&sin,0,sizeof(sin));
	sin.sin_family = AF_INET;
	sin.sin_port = htons (6000);
	sin.sin_addr.s_addr = htonl(INADDR_ANY);
	bind(rsfd,(struct sockaddr *) &sin,sizeof(sin));
	vector<int>v;
	while(1)
	{
		recvfrom(rsfd,buff,4096,0,(struct sockaddr *) &sin,&len);
		cout<<sin.sin_port<<endl;
		cout<<(char*)buff<<endl;
		struct iphdr *ipp;
		ipp=(struct iphdr*)buff;
		int p=0;
		string s=inet_ntoa(*(in_addr*)&ipp->saddr);
		string str="";
		int a=0;
		for(int i=0;s[i];i++)
		{


			if(s[i]!='.')
			{
				str+=s[i];
			}
			else
			{
				p=p*256+stoi(str);
				str="";
			}

		}
		p=p*256+stoi(str);
		cout<<p<<endl;
		v.push_back(p);

		break;

	}
	int sfd = 0, valread;
```

```
        struct sockaddr_in address;

        char buffer[1024] = {0};
        if ((sfd= socket(AF_INET, SOCK_STREAM, 0)) < 0)
        {
                printf("\n Socket creation error \n");
                return -1;
        }

        address.sin_family = AF_INET;
        address.sin_port = htons(v[0]);

        // Convert IPv4 and IPv6 addresses from text to binary form
        if(inet_pton(AF_INET, "127.0.0.1", &address.sin_addr)<=0)
        {
                printf("\nInvalid address/ Address not supported \n");
                return -1;
        }

        if (connect(sfd, (struct sockaddr *)&address, sizeof(address)) < 0)
        {
                printf("\nConnection Failed \n");
                return -1;
        }
//      while(1){
        string s="c2";
        char *hello = &s[0];
        send(sfd, hello , s.length() , 0 );
        valread = recv( sfd , buffer, 1024,0);
        cout<<buffer<<endl;


//      }


}
```

C3

```
#include<time.h>
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<sys/select.h>
#include<pthread.h>
#include<signal.h>
#include<stdlib.h>
#include<fcntl.h>
```

```
#include<sys/shm.h>
#include<unistd.h>
#include<sys/un.h>
#include<netinet/ip.h>
#include<arpa/inet.h>
#include<errno.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<netinet/ether.h>
#include<netinet/udp.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<bits/stdc++.h>
using namespace std;

#define BUF_LEN 1024

void print_ipheader(struct iphdr* ip)
{
	cout<<"------------------------\n";
	cout<<"Printing IP header....\n";
	cout<<"IP version:"<<(unsigned int)ip->version<<endl;
	cout<<"IP header length:"<<(unsigned int)ip->ihl<<endl;

	cout<<"Type of service:"<<(unsigned int)ip->tos<<endl;
	cout<<"Total ip packet length:"<<ntohs(ip->tot_len)<<endl;
	cout<<"Packet id:"<<ntohs(ip->id)<<endl;
	cout<<"Time to leave :"<<(unsigned int)ip->ttl<<endl;
	cout<<"Protocol:"<<(unsigned int)ip->protocol<<endl;
	cout<<"Check:"<<ip->check<<endl;
	cout<<"Source ip:"<<inet_ntoa(*(in_addr*)&ip->saddr)<<endl;
	cout<<ip->saddr<<endl;

	//printf("%pI4\n",&ip->saddr );
	cout<<"Destination ip:"<<inet_ntoa(*(in_addr*)&ip->daddr)<<endl;
	cout<<"End of IP header\n";
	cout<<"------------------------\n";
}

void reverse(char* str)
{
   // l for swap with index 2
   int l = 2;
   int r = strlen(str) - 2;

   // swap with in two-2 pair
   while (l < r) {
      swap(str[l++], str[r++]);
      swap(str[l++], str[r]);
      r = r - 3;
   }
}
```

```
// function to conversion and print
// the hexadecimal value
void ipToHexa(int addr)
{
    char str[15];

    // convert integer to string for reverse
    sprintf(str, "0x%08x", addr);

    // reverse for get actual hexadecimal
    // number without reverse it will
    // print 0x0100007f for 127.0.0.1
    reverse(str);

    // print string
    cout << str << "\n";
}


int main()
{
        int rsfd = socket (PF_INET, SOCK_RAW, 1);
        int n = 0 ;
        int one = 1;
        const int *val = &one;
        if (setsockopt (rsfd, IPPROTO_IP, IP_HDRINCL, val, sizeof (one)) < 0)
        cout<<"Not set";
        char buff[4096];
        struct iphdr *iph = (struct iphdr *)buff;
        struct sockaddr_in sin,sin2;
        socklen_t len = sizeof(sin2);
        memset(&sin,0,sizeof(sin));
        sin.sin_family = AF_INET;
        sin.sin_port = htons (6000);
        sin.sin_addr.s_addr = htonl(INADDR_ANY);
        bind(rsfd,(struct sockaddr *) &sin,sizeof(sin));
        vector<int>v;
        while(1)
        {
                recvfrom(rsfd,buff,4096,0,(struct sockaddr *) &sin,&len);
                cout<<sin.sin_port<<endl;
                cout<<(char*)buff<<endl;
                struct iphdr *ipp;
                ipp=(struct iphdr*)buff;
                int p=0;
                string s=inet_ntoa(*(in_addr*)&ipp->saddr);
                string str="";
                int a=0;
                for(int i=0;s[i];i++)
                {
```

```
                if(s[i]!='.')
                {
                        str+=s[i];
                }
                else
                {
                        p=p*256+stoi(str);
                        str="";
                }

        }
        p=p*256+stoi(str);
        cout<<p<<endl;
        v.push_back(p);

        break;

    }
    int sfd = 0, valread;
    struct sockaddr_in address;

    char buffer[1024] = {0};
    if ((sfd= socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Socket creation error \n");
        return -1;
    }

    address.sin_family = AF_INET;
    address.sin_port = htons(v[0]);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, "127.0.0.1", &address.sin_addr)<=0)
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(sfd, (struct sockaddr *)&address, sizeof(address)) < 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
//  while(1){
    string s="c3";
    char *hello = &s[0];
    send(sfd, hello , s.length() , 0 );
    valread = recv( sfd , buffer, 1024,0);
    cout<<buffer<<endl;
```

```
//        }

}
```