Multiple Client Server

Server

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

void main(){
   int sockfd, connfd1, connfd2, len;
   struct sockaddr_in servaddr, cli;
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd == -1){
      printf("Socket creation failed.\n");
      exit(0);}
   else
      printf("Socket successfully created.\n");
   bzero(&servaddr, sizeof(servaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
   servaddr.sin_port = htons(PORT);
   if ((bind(sockfd, (SA *)&servaddr,
sizeof(servaddr))) != 0) {
      printf("Socket bind failed.\n");
      exit(0);}
   else
      printf("Socket successfully binded.\n");
   if ((listen(sockfd, 5)) != 0){
      printf("Listen failed.\n");
      exit(0);}
   else
      printf("Server listening.\n");
   len = sizeof(cli);
   char buf1[100], buf2[100];
   connfd1 = accept(sockfd, (SA *)&cli, &len);
   connfd2 = accept(sockfd, (SA *)&cli, &len);

   if (connfd1 < 0 || connfd2 < 0){
      printf("Connection with clients failed.\n");
      exit(0);
   }
   else
      printf("Connection created with clients
successfully.\n");
   while (1){
      read(connfd1, buf1, sizeof(buf1));
      read(connfd2, buf2, sizeof(buf2));
      printf("Message from client 1: %s\n", buf1);
      printf("Message from client 2: %s\n", buf2);
      write(connfd1, buf2, sizeof(buf2));
      write(connfd2, buf1, sizeof(buf1));}
   close(sockfd);
}
```

Client

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

void main(){
   int sockfd, connfd;
   struct sockaddr_in servaddr, cli;
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd == -1)  {
      printf("Socket creation failed.\n");
      exit(0);}
   else
      printf("Socket successfully created.\n");
   bzero(&servaddr, sizeof(servaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
   servaddr.sin_port = htons(PORT);
   char buf1[100], buf2[100];
   if (connect(sockfd, (SA *)&servaddr,
sizeof(servaddr)) < 0) {
      printf("Connection failed.\n");
   }else
      printf("Connected successfully\n");
   while (1) {
      printf("Enter message to send to client 1: ");
      scanf("%s", buf1);
      write(sockfd, buf1, sizeof(buf1));
      read(sockfd, buf2, sizeof(buf2));
      printf("Message from client 1: %s\n", buf2); }
   close(sockfd);
}
```

Output

Server

```
Socket successfully created.
Socket successfully binded.
Server listening.
Connection created with clients successfully.
Message from client 1: Hello
Message from client 2: hai
```

Client1

```
Socket successfully created.
Connected successfully
Enter message to send to client 1: Hello
Message from client 1: hai
```

Client2

```
Socket successfully created.
Connected successfully
Enter message to send to client 1: hai
Message from client 1: Hello
```

Concurrent Time Server Application using UDP

Server

```
#include<stdio.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<stdio.h>
#include<time.h>

#define S_PORT 43454
#define C_PORT 43455
#define ERROR -1
#define IP_STR "127.0.0.1"

int main(int argc,char const* argv[]){
        int sfd,num; time_t current_time;
        struct sockaddr_in servaddr,clientaddr;
sfd=socket(AF_INET,SOCK_DGRAM,IPPROTO_U
DP);
        if(sfd==ERROR){
                perror("could not open a socket");
                return 1;}
        memset((char*)&servaddr,0,sizeof(servaddr));
        servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
        servaddr.sin_port=htons(S_PORT);
memset((char*)&clientaddr,0,sizeof(clientaddr));
        clientaddr.sin_family=AF_INET;
clientaddr.sin_addr.s_addr=inet_addr(IP_STR);
        clientaddr.sin_port=htons(C_PORT);
        if((bind(sfd,(struct
sockaddr*)&servaddr,sizeof(servaddr)))!=0){
                perror("could not bind socket");
                return 2;}
        printf("server is running on %s:
%d\n",IP_STR,S_PORT);
        while(1){
                recvfrom(sfd,&num,sizeof(num),0,
(struct sockaddr*)&clientaddr,
(socklen_t*)&clientaddr);
                current_time=time(NULL);
                printf("client at %s:%d asked for time:
%s\n",inet_ntoa(clientaddr.sin_addr),ntohs(clientaddr.
sin_port),ctime(&current_time));
sendto(sfd,&current_time,sizeof(current_time),0,
(struct sockaddr *)&clientaddr,sizeof(clientaddr));}
        return 0;}
```

Client

```
#include<stdio.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<stdio.h>
#include<time.h>

#define S_PORT 43454
#define C_PORT 43455
#define ERROR -1
#define IP_STR "127.0.0.1"
 int main(int argc,char const* argv[]){
        int sfd; int num=1;
        time_t start_time,rtt,current_time;
        struct sockaddr_in servaddr,clientaddr;
        socklen_t addrlen;
sfd=socket(AF_INET,SOCK_DGRAM,IPPROTO_U
DP);
        if(sfd==ERROR){
                perror("could not open a socket");
                return 1;}
        memset((char
*)&servaddr,0,sizeof(servaddr));
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=inet_addr(IP_STR);
	servaddr.sin_port=htons(S_PORT);
	memset((char
*)&clientaddr,0,sizeof(clientaddr));
	clientaddr.sin_family=AF_INET;
clientaddr.sin_addr.s_addr=inet_addr(IP_STR);
	clientaddr.sin_port=htons(C_PORT);
	if((bind(sfd,(struct sockaddr
*)&clientaddr,sizeof(clientaddr)))!=0){
		perror("could not bind socket");
		return 2;}
	printf("client is running on %s:
%d\n",IP_STR,C_PORT);
	start_time=time(NULL);
	sendto(sfd,&num,sizeof(num),0,(struct
sockaddr*)&servaddr,sizeof(servaddr));
	addrlen=sizeof(clientaddr);
recvfrom(sfd,&current_time,sizeof(current_time),0,
(struct sockaddr*)&clientaddr,&addrlen);
	rtt = time(NULL)-start_time;
	current_time += rtt/2;
	printf("server's time:
%s\n",ctime(&current_time));
	return 0;}
```

Output

Server

```
server is running on 127.0.0.1:43454
client at 127.0.0.1:43455 asked for time:Thu Mar 13
21:15:44 2025
```

Client

```
client is running on 127.0.0.1:43455
server's time:Thu Mar 13 21:15:44 2025
```

File Transfer Protocol

Server

```
 #include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5035
#define MAX 60
int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char * argv) {
	int sockfd, newsockfd, clength;
	struct sockaddr_in serv_addr,cli_addr;
	char t[MAX], str[MAX];
	strcpy(t,"exit");
	sockfd=socket(AF_INET,
SOCK_STREAM,0);
	serv_addr.sin_family=AF_INET;
	serv_addr.sin_addr.s_addr=INADDR_ANY;
	serv_addr.sin_port=htons(SERV_TCP_PORT);
	printf("\nBinded");
	bind(sockfd, (struct sockaddr*)&serv_addr,
sizeof(serv_addr));
	printf("\nListening...");
	listen(sockfd, 5);
	clength=sizeof(cli_addr);
	newsockfd=accept(sockfd, (struct sockaddr*)
&cli_addr,&clength);
	close(sockfd);
	read(newsockfd, &str, MAX);
	printf("\nClient message\n File Name: %s\n",
str);
	f1=fopen(str, "r");
	while(fgets(buff, 4096, f1)!=NULL){
		write(newsockfd, buff, MAX);
		printf("\n"); }
	fclose(f1);
	printf("\nFile Transferred\n");
	return 0; }
```

Client

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include <arpa/inet.h>
#define SERV_TCP_PORT 5035
#define MAX 60

int main(int arg,char*argv[]) {
	int sockfd,n;
	struct sockaddr_in serv_addr;
	struct hostent server;
```

```
        char send[MAX],recvline[MAX], s[MAX],
name[MAX];
sockfd=socket(AF_INET,SOCK_STREAM,0);
        serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
        serv_addr.sin_port=htons
(SERV_TCP_PORT);
        connect(sockfd, (struct
sockaddr*)&serv_addr,sizeof(serv_addr));
        printf("\nEnter the source file name: \n");
        scanf("%s", send);
        write(sockfd, send, MAX);
        while((n=read(sockfd,recvline,MAX))!=0) {
                printf("%s", recvline); }
        close(sockfd);
        return 0; }
```

Output

Server

```
Binded
Listening...
Client message
 File Name: file1


File Transferred
```

Client

```
Enter the source file name:
file1
Hello
```

Leaky Bucket Algorithm

```
#include <stdio.h>

void main(){
   int incoming, outgoing, buck_size, n, store = 0;
   printf("Enter bucket Size , outgoing rate and no. of
i/p: ");
   scanf("%d %d %d", &buck_size, &outgoing, &n);
   while (n != 0){
      printf("Enter the incoming packet size: ");
      scanf("%d", &incoming);
      printf("Incoming packet size %d\n", incoming);
      if (incoming <= (buck_size - store))  {
         store += incoming;
         printf("Bucket buffer size %d out of %d\n",
store, buck_size);}
      else{
         printf("Dropped %d no. of packets\n",
incoming - (buck_size - store));
         printf("Bucket buffer size %d out of %d\n",
store, buck_size);
         store = buck_size;}
      store = store - outgoing;
      if (store < 0)
         store = 0;
      printf("After outgoind %d packets left out of %d
in buffer \n", store, buck_size);
      n--; }}
```

Output

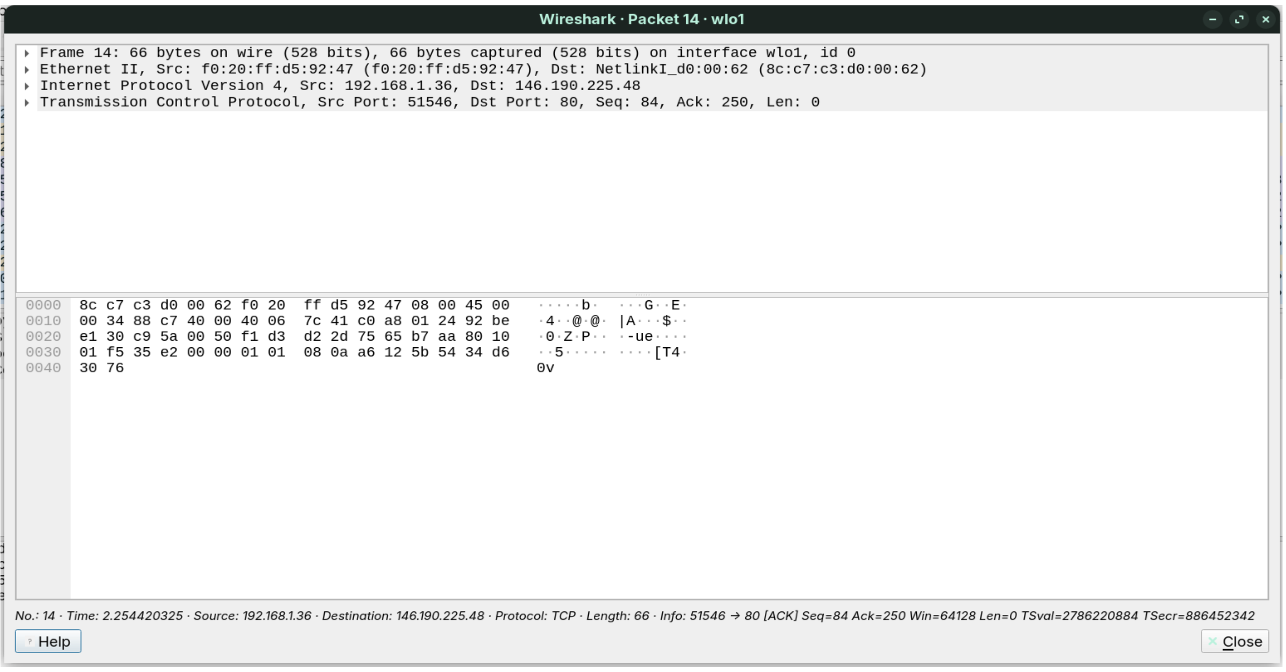
```
Enter bucket Size , outgoing rate and no. of i/p: 100 5
3
Enter the incoming packet size: 25
Incoming packet size 25
Bucket buffer size 25 out of 100
After outgoind 20 packets left out of 100 in buffer
Enter the incoming packet size: 50
Incoming packet size 50
Bucket buffer size 70 out of 100
After outgoind 65 packets left out of 100 in buffer
Enter the incoming packet size: 20
Incoming packet size 20
Bucket buffer size 85 out of 100
After outgoind 80 packets left out of 100 in buffer
```

Distance Vetcor Routing Algorithm

```
#include <stdio.h>

int costMatrix[20][20], n;
struct routers {
   int distance[20];
   int adjNodes[20];
} node[20];
void readCostMatrix() {
   int i, j;
   printf("\n  ENTER COST MATRIX\n");
   for(i = 0; i < n; ++i) {
      for(j = 0; j < n; ++j) {
         scanf("%d", &costMatrix[i][j]);
         node[i].distance[j] = costMatrix[i][j];
         node[i].adjNodes[j] = j;}}}
void calcRoutingTable() {
   int i, j, k;
   for(i = 0; i < n; ++i) {
      for(j = 0; j < n; ++j) {
         for(k = 0; k < n; ++k) {
            if(node[i].distance[j] > node[i].distance[k] +
costMatrix[k][j]) {
               node[i].distance[j] = node[i].distance[k] +
costMatrix[k][j];
               node[i].adjNodes[j] = k;}}}}}
void displayRoutes() {
   int i, j;
   for(i = 0; i < n; ++i) {
      printf("\n Router %d\n", i + 1);
      for(j = 0; j < n; ++j) {
         printf("Node %d via %d : Distance %d \n", j +
1, node[i].adjNodes[j] + 1, node[i].distance[j]);}
      printf("\n");}}
int main() {
   int i, j;
   printf(" Enter Number of nodes: ");
   scanf("%d", &n);
   readCostMatrix();
   calcRoutingTable();
   displayRoutes();
   return 0;}
```

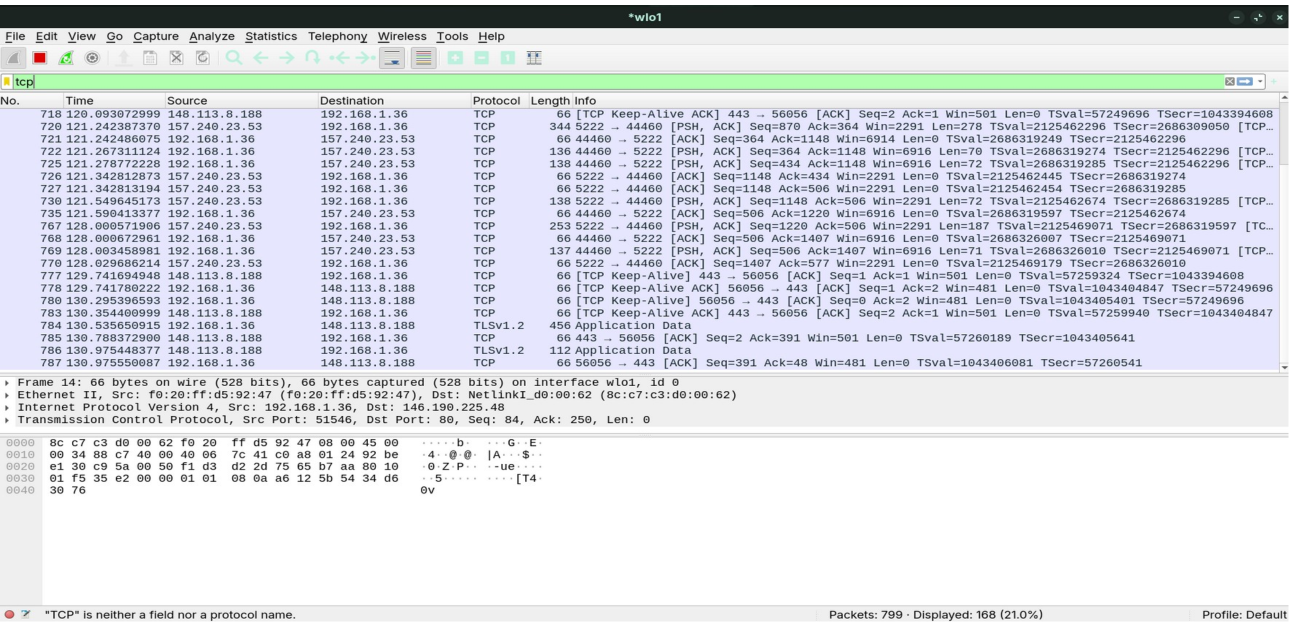
Output

```
 Enter Number of nodes: 3

  ENTER COST MATRIX
0 1 5
1 0 2
5 2 0

 Router 1
Node 1 via 1 : Distance 0
Node 2 via 2 : Distance 1
Node 3 via 2 : Distance 3


 Router 2
Node 1 via 1 : Distance 1
Node 2 via 2 : Distance 0
Node 3 via 3 : Distance 2


 Router 3
Node 1 via 2 : Distance 3
Node 2 via 2 : Distance 2
Node 3 via 3 : Distance 0
```



```
▸ Frame 14: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlo1, id 0
▸ Ethernet II, Src: f0:20:ff:d5:92:47 (f0:20:ff:d5:92:47), Dst: NetlinkI_d0:00:62 (8c:c7:c3:d0:00:62)
▸ Internet Protocol Version 4, Src: 192.168.1.36, Dst: 146.190.225.48
▸ Transmission Control Protocol, Src Port: 51546, Dst Port: 80, Seq: 84, Ack: 250, Len: 0

0000  8c c7 c3 d0 00 62 f0 20  ff d5 92 47 08 00 45 00   ·····b·  ···G··E·
0010  00 34 88 c7 40 00 40 06  7c 41 c0 a8 01 24 92 be   ·4··@·@·  |A···$··
0020  e1 30 c9 5a 00 50 f1 d3  d2 2d 75 65 b7 aa 80 10   ·0·Z·P··  ·-ue····
0030  01 f5 35 e2 00 00 01 01  08 0a a6 12 5b 54 34 d6   ··5·····  ····[T4·
0040  30 76                                              0v
```

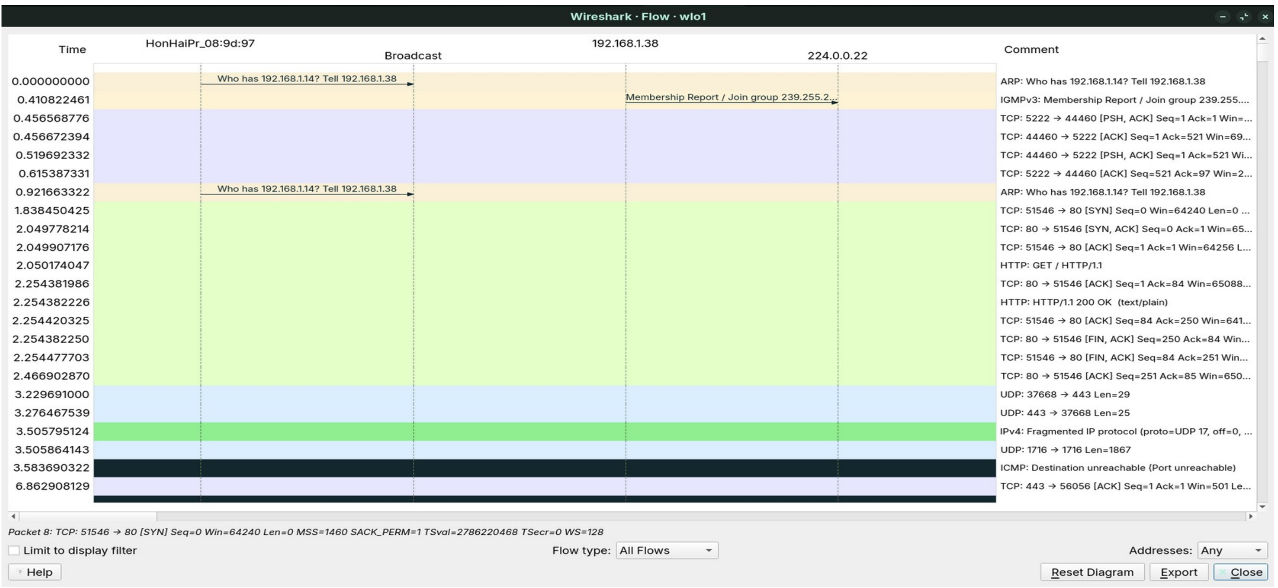
No.: 14 · Time: 2.254420325 · Source: 192.168.1.36 · Destination: 146.190.225.48 · Protocol: TCP · Length: 66 · Info: 51546 → 80 [ACK] Seq=84 Ack=250 Win=64128 Len=0 TSval=2786220884 TSecr=886452342



tcp

```
No.  Time          Source          Destination     Protocol Length Info
 718 120.093072999 148.113.8.188   192.168.1.36    TCP       66 [TCP Keep-Alive ACK] 443 → 56056 [ACK] Seq=2 Ack=1 Win=501 Len=0 TSval=57249696 TSecr=1043394608
 720 121.242387370 157.240.23.53   192.168.1.36    TCP      344 5222 → 44460 [PSH, ACK] Seq=870 Ack=364 Win=2291 Len=278 TSval=2125462296 TSecr=2686309050 [TCP…
 721 121.242486075 192.168.1.36    157.240.23.53   TCP       66 44460 → 5222 [ACK] Seq=364 Ack=1148 Win=6914 Len=0 TSval=2686319249 TSecr=2125462296
 722 121.267311124 192.168.1.36    157.240.23.53   TCP      136 44460 → 5222 [PSH, ACK] Seq=364 Ack=1148 Win=6916 Len=70 TSval=2686319274 TSecr=2125462296 [TCP…
 725 121.278772228 192.168.1.36    157.240.23.53   TCP      138 44460 → 5222 [PSH, ACK] Seq=434 Ack=1148 Win=6916 Len=72 TSval=2686319285 TSecr=2125462296 [TCP…
 726 121.342812873 157.240.23.53   192.168.1.36    TCP       66 5222 → 44460 [ACK] Seq=1148 Ack=434 Win=2291 Len=0 TSval=2125462445 TSecr=2686319274
 727 121.342813194 157.240.23.53   192.168.1.36    TCP       66 5222 → 44460 [ACK] Seq=1148 Ack=506 Win=2291 Len=0 TSval=2125462454 TSecr=2686319285
 730 121.549645173 157.240.23.53   192.168.1.36    TCP      138 5222 → 44460 [PSH, ACK] Seq=1148 Ack=506 Win=2291 Len=72 TSval=2125462674 TSecr=2686319285 [TCP…
 735 121.590413377 192.168.1.36    157.240.23.53   TCP       66 44460 → 5222 [ACK] Seq=506 Ack=1220 Win=6916 Len=0 TSval=2686319597 TSecr=2125462674
 767 128.000571906 157.240.23.53   192.168.1.36    TCP      253 5222 → 44460 [PSH, ACK] Seq=1220 Ack=506 Win=2291 Len=187 TSval=2125469071 TSecr=2686319597 [TC…
 768 128.000672961 192.168.1.36    157.240.23.53   TCP       66 44460 → 5222 [ACK] Seq=506 Ack=1407 Win=6916 Len=0 TSval=2686326007 TSecr=2125469071
 769 128.003458981 192.168.1.36    157.240.23.53   TCP      137 44460 → 5222 [PSH, ACK] Seq=506 Ack=1407 Win=6916 Len=71 TSval=2686326010 TSecr=2125469071 [TCP…
 770 128.029686214 157.240.23.53   192.168.1.36    TCP       66 5222 → 44460 [ACK] Seq=1407 Ack=577 Win=2291 Len=0 TSval=2125469179 TSecr=2686326010
 777 129.741694948 148.113.8.188   192.168.1.36    TCP       66 [TCP Keep-Alive] 443 → 56056 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=57259324 TSecr=1043394608
 778 129.741780222 192.168.1.36    148.113.8.188   TCP       66 [TCP Keep-Alive ACK] 56056 → 443 [ACK] Seq=1 Ack=2 Win=481 Len=0 TSval=1043404847 TSecr=57249696
 780 130.295396593 192.168.1.36    148.113.8.188   TCP       66 [TCP Keep-Alive] 56056 → 443 [ACK] Seq=0 Ack=2 Win=481 Len=0 TSval=1043405401 TSecr=57249696
 783 130.354400999 148.113.8.188   192.168.1.36    TCP       66 [TCP Keep-Alive ACK] 443 → 56056 [ACK] Seq=2 Ack=1 Win=501 Len=0 TSval=57259940 TSecr=1043404847
 784 130.535650915 192.168.1.36    148.113.8.188   TLSv1.2  456 Application Data
 785 130.788372900 148.113.8.188   192.168.1.36    TCP       66 443 → 56056 [ACK] Seq=2 Ack=391 Win=501 Len=0 TSval=57260189 TSecr=1043405641
 786 130.975448377 148.113.8.188   192.168.1.36    TLSv1.2  112 Application Data
 787 130.975550087 192.168.1.36    148.113.8.188   TCP       66 56056 → 443 [ACK] Seq=391 Ack=48 Win=481 Len=0 TSval=1043406081 TSecr=57260541

▸ Frame 14: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlo1, id 0
▸ Ethernet II, Src: f0:20:ff:d5:92:47 (f0:20:ff:d5:92:47), Dst: NetlinkI_d0:00:62 (8c:c7:c3:d0:00:62)
▸ Internet Protocol Version 4, Src: 192.168.1.36, Dst: 146.190.225.48
▸ Transmission Control Protocol, Src Port: 51546, Dst Port: 80, Seq: 84, Ack: 250, Len: 0

0000  8c c7 c3 d0 00 62 f0 20  ff d5 92 47 08 00 45 00   ·····b·  ···G··E·
0010  00 34 88 c7 40 00 40 06  7c 41 c0 a8 01 24 92 be   ·4··@·@·  |A···$··
0020  e1 30 c9 5a 00 50 f1 d3  d2 2d 75 65 b7 aa 80 10   ·0·Z·P··  ·-ue····
0030  01 f5 35 e2 00 00 01 01  08 0a a6 12 5b 54 34 d6   ··5·····  ····[T4·
0040  30 76                                              0v
```

"TCP" is neither a field nor a protocol name.   Packets: 799 · Displayed: 168 (21.0%)   Profile: Default


Wireshark · Flow · wlo1
HonHaiPr_08:9d:97
Broadcast
192.168.1.38
224.0.0.22
Who has 192.168.1.14? Tell 192.168.1.38
Membership Report / Join group 239.255.2...
Who has 192.168.1.14? Tell 192.168.1.38
Help
Reset Diagram
Export
Close

```
Time          Comment
0.000000000   ARP: Who has 192.168.1.14? Tell 192.168.1.38
0.410822461   IGMPv3: Membership Report / Join group 239.255....
0.456568776   TCP: 5222 → 44460 [PSH, ACK] Seq=1 Ack=1 Win=...
0.456672394   TCP: 44460 → 5222 [ACK] Seq=1 Ack=521 Win=69...
0.519692332   TCP: 44460 → 5222 [PSH, ACK] Seq=1 Ack=521 Wi...
0.615387331   TCP: 5222 → 44460 [ACK] Seq=521 Ack=97 Win=2...
0.921663322   ARP: Who has 192.168.1.14? Tell 192.168.1.38
1.838450425   TCP: 51546 → 80 [SYN] Seq=0 Win=64240 Len=0 ...
2.049778214   TCP: 80 → 51546 [SYN, ACK] Seq=0 Ack=1 Win=65...
2.049907176   TCP: 51546 → 80 [ACK] Seq=1 Ack=1 Win=64256 L...
2.050174047   HTTP: GET / HTTP/1.1
2.254381986   TCP: 80 → 51546 [ACK] Seq=1 Ack=84 Win=65088...
2.254382226   HTTP: HTTP/1.1 200 OK  (text/plain)
2.254420325   TCP: 51546 → 80 [ACK] Seq=84 Ack=250 Win=641...
2.254382250   TCP: 80 → 51546 [FIN, ACK] Seq=250 Ack=84 Win...
2.254477703   TCP: 51546 → 80 [FIN, ACK] Seq=84 Ack=251 Win...
2.466902870   TCP: 80 → 51546 [ACK] Seq=251 Ack=85 Win=650...
3.229691000   UDP: 37668 → 443 Len=29
3.276467539   UDP: 443 → 37668 Len=25
3.505795124   IPv4: Fragmented IP protocol (proto=UDP 17, off=0, ...
3.505864143   UDP: 1716 → 1716 Len=1867
3.583690322   ICMP: Destination unreachable (Port unreachable)
6.862908129   TCP: 443 → 56056 [ACK] Seq=1 Ack=1 Win=501 Le...
```

Packet 8: TCP: 51546 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2786220468 TSecr=0 WS=128

☐ Limit to display filter   Flow type: All Flows   Addresses: Any