Socket Programming

- Socket programming with TCP
- Socket programming with UDP

Socket programming *with UDP*

UDP: no "connection" between client and server

- no handshaking
- sender explicitly attaches IP address and port of destination to each packet
- server must extract IP address, port of sender from received packet

UDP: transmitted data may be received out of order, or lost

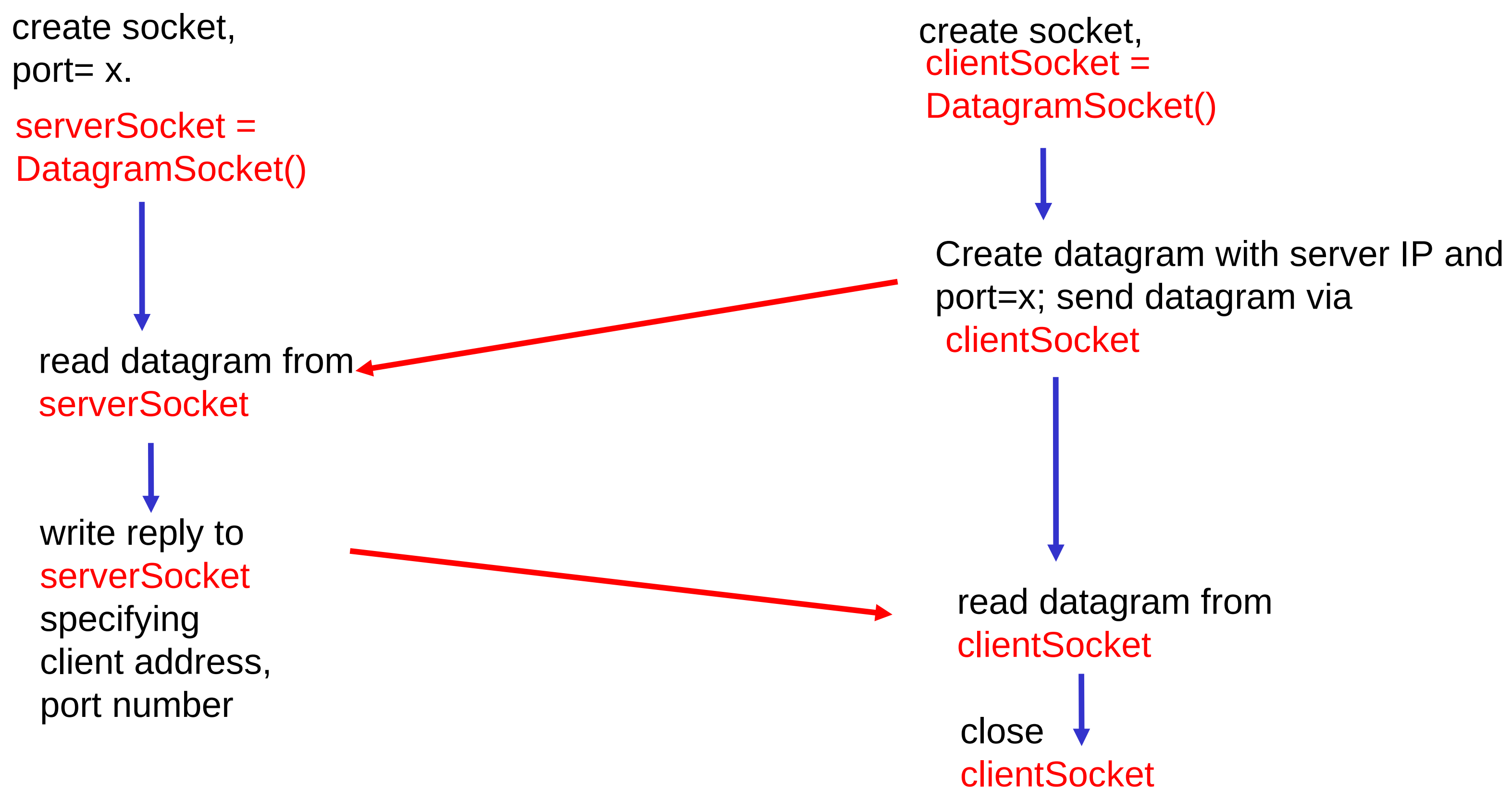
application viewpoint

UDP provides unreliable transfer of groups of bytes ("datagrams") between client and server

Client/server socket interaction: UDP

Server (running on **`hostid`**)

Client



Server:

create socket, port= x.
serverSocket = DatagramSocket()

read datagram from serverSocket

write reply to serverSocket specifying client address, port number

Client:

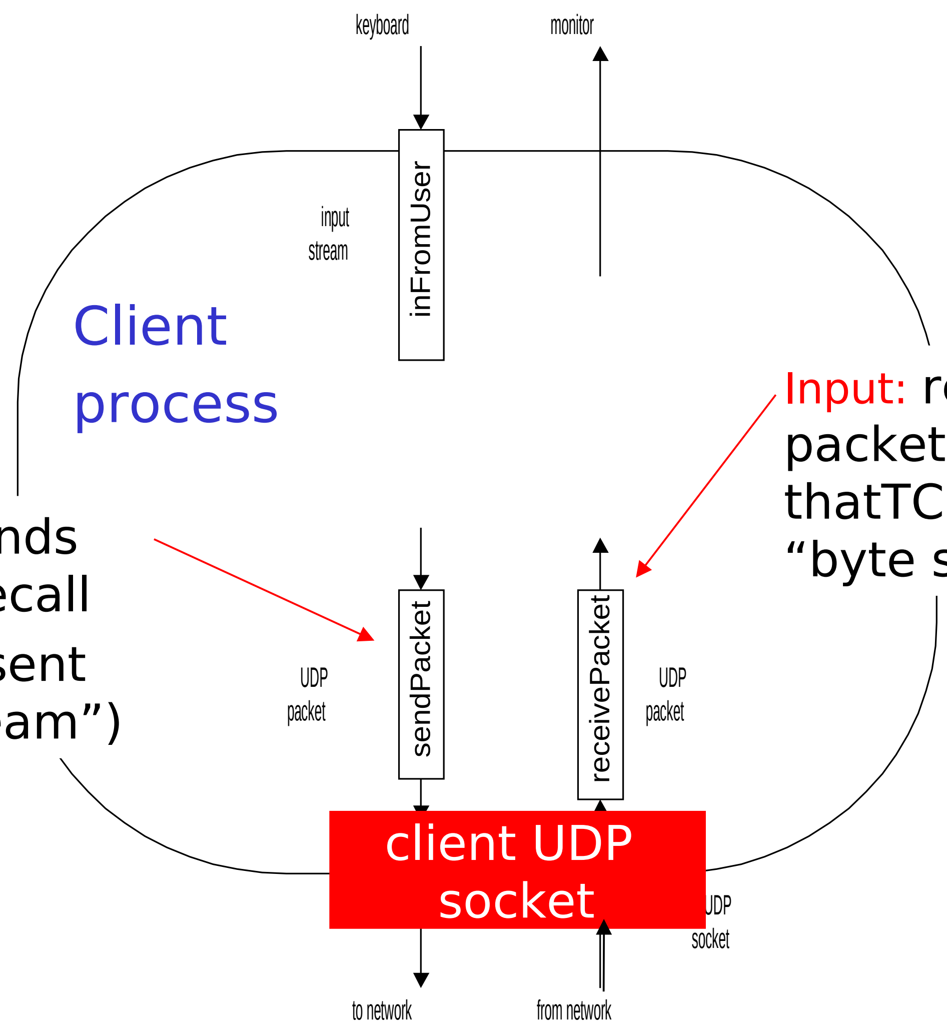
create socket, clientSocket = DatagramSocket()

Create datagram with server IP and port=x; send datagram via clientSocket

read datagram from clientSocket

close clientSocket

Example: (UDP)



Output: sends packet (recall that TCP sent "byte stream")

Input: receives packet (recall thatTCP received "byte stream")

socket()

- Create a socket for the protocol you want:

```
s = socket(PF_INET, SOCK_STREAM, 0);//TCP
s = socket(PF_INET, SOCK_DGRAM, 0);//UDP
```

Now we also need special ways of sending/receiving information via UDP. The functions are sendto() and recvfrom().

No explicit connection is needed, you simply send or wait to receive information...

Still, *IP addresses* and *port numbers* play a role.

socket commands: sendto()

- Send data through a socket:

```
sendto(SOCKET s, char *msg, int msglen, int flags, struct sockaddr *toaddr, int *tolen);
```

s = socket (inside the socket descriptor: *port* and *IP address*...), created with **SOCK_DGRAM** option
msg = a pointer to a buffer (a string)
strlen = the length of the buffer
flags = 0 (forget about them for this exercise...)
toaddr=structure of address with the IP / port #
tolen=length of the structure

Example:

```
sendto(s, sbuffer, strlen(sbuffer),0,(struct sockaddr*) toaddr, &tolen);
```

socket commands: recvfrom()

- Receive data:

int recvfrom(SOCKET s, char *msg, int msglen, int flags,
    struct sockaddr *fromaddr, int *fromlen);

s = socket
msg = pointer to a buffer
msglen = length of the buffer
flags = 0
fromaddr=structure of address with the IP / port #
fromlen=length of the structure
Example:

```
recvfrom(s, &rbuffer[n], 1, 0,(struct sockaddr *)
  fromaddr, &fromlen);
```

Example: C client (UDP)

```
//159.334 - Networks
//        CLIENT
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define BUFFESIZE 80
#define SEGMENTSIZE 78

void get_keyboard(char * send_buffer) {
fgets(send_buffer,SEGMENTSIZE,stdin);
//printf("lenght %d \n", strlen(send_buffer));
   if (send_buffer[strlen(send_buffer)-1]=='\n') {//if the last one is \n, which means there is at least one \0 after it
          send_buffer[strlen(send_buffer)-1]='\r';
          send_buffer[strlen(send_buffer)]='\n';
    }
    else {
        send_buffer[strlen(send_buffer)]='\n';//write \n on the last byte
    }
}

int main(int argc, char *argv[]) {
//*******************************************************************
// Initialization
//*******************************************************************
   struct sockaddr_in localaddr,remoteaddr;
   memset(&localaddr, 0, sizeof(localaddr));//clean up
   memset(&remoteaddr, 0, sizeof(remoteaddr));//clean up
   int s;
   char send_buffer[BUFFESIZE],receive_buffer[BUFFESIZE];
   int n,bytes;
```

Example: C client (UDP) cont...

```
  if (argc == 1) {
    printf("USAGE: client IP-address [port]\n");
    exit(1);
  }
//Port number: get it from argv[2], convert/copy to sin_port
  if (argc == 3) remoteaddr.sin_port = htons((u_short)atoi(argv[2]));
  else remoteaddr.sin_port = htons(1234);
//Family of protocolsbind(s, (struct sockaddr *)&localaddr, sizeof(localaddr));
  remoteaddr.sin_addr.s_addr = inet_addr(argv[1]);
  remoteaddr.sin_family = AF_INET;
//*******************************************************************
//CREATE CLIENT'S SOCKET
//*******************************************************************
  s = socket(AF_INET, SOCK_DGRAM, 0);
  if (s < 0) {
    printf("socket failed\n");
       exit(1);
  }
  memset(send_buffer, 0, sizeof(send_buffer));//clean up
  get_keyboard(send_buffer);
  while (strncmp(send_buffer,".",1) != 0) {
//*******************************************************************
//SEND
//*******************************************************************
    bytes = sendto(s, send_buffer, strlen(send_buffer),0,(struct sockaddr *)(&remoteaddr),sizeof(remoteaddr) );
printf("SENDER --> %s \n",send_buffer);
    if (bytes < 0) {
      printf("send failed\n");
       exit(1);
    }
    memset(send_buffer, 0, sizeof(send_buffer));//clean up
    get_keyboard(send_buffer);
  }
  close(s);
  return 0;
}
```

Example: C server (UDP)

```
#include <errno.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

main(int argc, char *argv[]) {
//*******************************************************************
// INITIALIZATION
//*******************************************************************
  struct sockaddr_in localaddr,remoteaddr;
  int s;
  char send_buffer[80],receive_buffer[80];
  char remoteIP[INET_ADDRSTRLEN]="127.0.0.1";
  int remotePort=1234;
  int localPort;//no need for local IP...
  int n,bytes,addrlen;
  memset(&localaddr,0,sizeof(localaddr));//clean up the structure
  memset(&remoteaddr,0,sizeof(remoteaddr));//clean up the structure
//*******************************************************************
//SOCKET
//*******************************************************************
  s = socket(PF_INET, SOCK_DGRAM, 0);
  if (s <0) {
    printf("socket failed\n");
  }
  localaddr.sin_family = AF_INET;
  if (argc == 2) localaddr.sin_port = htons((u_short)atoi(argv[1]));
  else localaddr.sin_port = htons(1235);//default listening port
  localaddr.sin_addr.s_addr = INADDR_ANY;//server address should be local
//*******************************************************************
//BIND (notice, no listen()...)
//*******************************************************************
  if (bind(s,(struct sockaddr *)(&localaddr),sizeof(localaddr)) != 0) {
    printf("Bind failed!\n");
    return;
  }
```

Example: C server (UDP) cont...

```
//*******************************************************************
//REMOTE HOST IP AND PORT
//*******************************************************************
remoteaddr.sin_family = AF_INET;
remoteaddr.sin_addr.s_addr = inet_addr(remoteIP);
remoteaddr.sin_port = htons(remotePort);
//*******************************************************************
//INFINITE LOOP
//*******************************************************************
  while (1) {
    addrlen = sizeof(remoteaddr);
//*******************************************************************
//RECEIVE
//*******************************************************************
printf("Waiting... \n");
        bytes = recvfrom(s, receive_buffer, 78, 0,(struct sockaddr *)(&remoteaddr),&addrlen);
printf("Received %d bytes\n",bytes);
//*******************************************************************
//PROCESS REQUEST
//*******************************************************************
n=0;
      while (n<bytes){
        n++;
        if ((bytes < 0) || (bytes == 0)) break;
        if (receive_buffer[n] == '\n') { /*end on a LF*/
           receive_buffer[n] = '\0';
           break;
        }
        if (receive_buffer[n] == '\r') /*ignore CRs*/
           receive_buffer[n] = '\0';
      }
      if ((bytes < 0) || (bytes == 0)) break;
        printf("After processing, recvbuffer is %s \n",receive_buffer);
  }
  close(s);
  return 0;
}
```

Blocking
X
Non-blocking

recvfrom()

- Remember: receive functions will put the process in a ***wait*** state.
- This means that you need to synchronize sender/receiver
- How do you do that if you don't know when to receive???
- Dead-lock possible with UDP
  - **no guarantee that the packet arrives...**
- For assignment 2, use **non-blocking** options

Recvfrom() non-blocking

- **Linux:**

```
bytes = recvfrom(s, receive_buffer, 78, MSG_DONTWAIT,
  (struct sockaddr *)(&remoteaddr),&addrlen);//non-blocking
```

- **Windows:**

Windows does not follow the standards. You need to change the socket to get non-blocking recv...

```
s=socket(AF_INET, SOCK_DGRAM, 0);
//nonblocking option
u_long iMode=1;
ioctlsocket(s,FIONBIO,&iMode);
```

Dealing with DNS requests using sockets

Host structure

- How to do DNS requests automatically?
- Old way: Get host (finds an IP address given a name):

```
struct hostent {
   char *h_name;       /*official host name*/
   char **h_aliases;   /*other aliases*/
   short h_addrtype;   /*addr type*/
   short h_length;     /*address length*/
   char **h_addr_list; /*list of addresses*/
}*h;
```

gethostbyname()

Example given URL:

```
if ((h=gethostbyname(host)) != NULL){

  memcpy(&s.sin_addr,h->h_addr_list,h->h_length);}

else { printf("error\n");  exit(1); }
```

“new” function: getaddrinfo()

- Sometimes a better way: use getaddrinfo()

```
int getaddrinfo( char *node, char *service,
  struct addrinfo *hints, struct addrinfo
  **res);
```

Structure addrinfo

- Use addrinfo for the address structures

- e.g.

```
struct addrinfo sin;
```

- Some parameters:

```
  sin.ai_family=AF_INET;
  sin.ai_socktype=SOCK_STREAM;
  sin.ai_flags=0;
  sin.ai_protocol=0;
```

getaddrinfo() example

- Create results and rtemp

```
struct addrinfo *results, *rtemp;
```

- Use getaddrinfo()

```
   if (getaddrinfo(argv[1],
  argv[2],&sin,&results)!=0){
        printf("An error occured while
  attempting to translate the IP address\n");
        exit(1);
   }
```

getaddrinfo() example

- The only drawback is that you have to test different results when connecting...

```
for (rtemp = results; rtemp != NULL; rtemp = rtemp->ai_next) {
  s = socket(rtemp->ai_family, rtemp->ai_socktype,rtemp->ai_protocol);
  if (s == -1)   continue;
  if (connect(s, rtemp->ai_addr, rtemp->ai_addrlen) != -1)    break;
   close(s);
}
if (rtemp == NULL) {                    /* No address succeeded */
  printf("connect failed\n");
  exit(1);
}
```