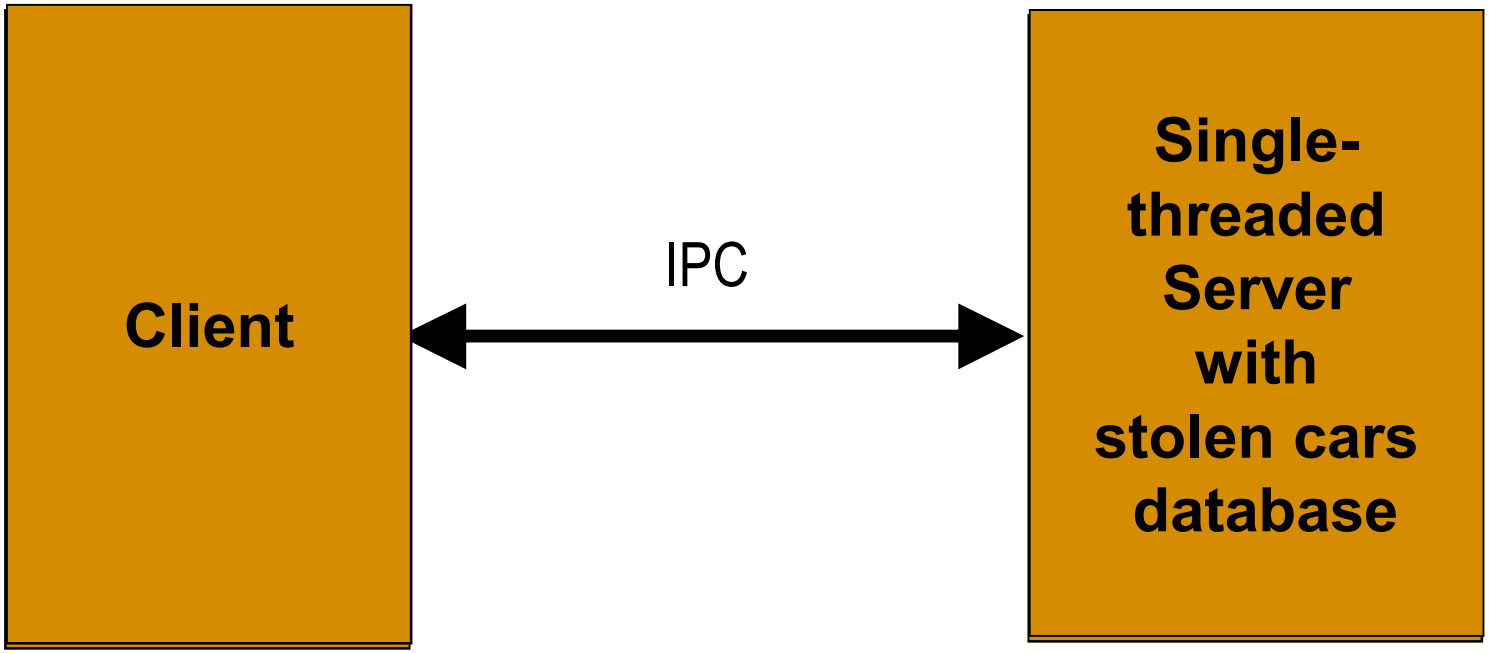
THE SECOND SPRING 2023 COSC 3360/6310 ASSIGNMENT

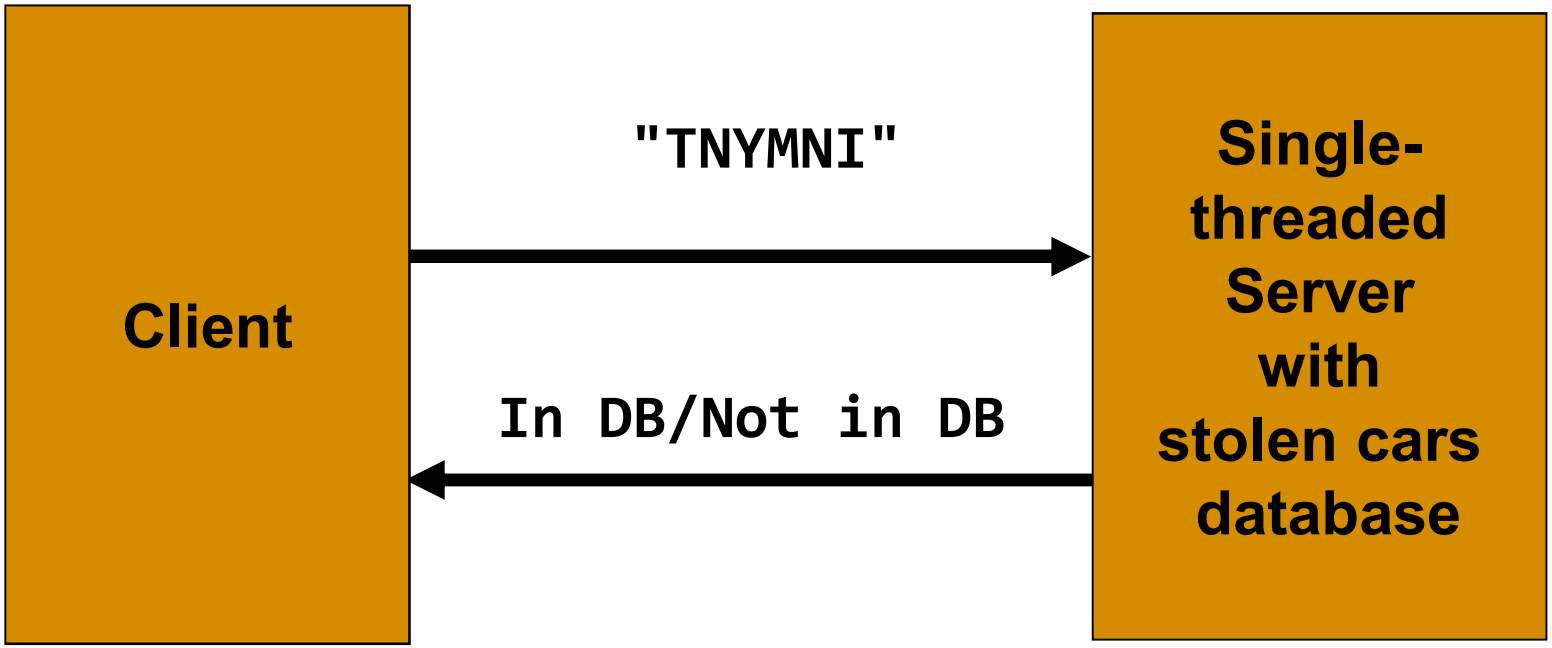
THE PROBLEM

- Build a client server pair that tells police officers whether a given car has been reported as stolen.

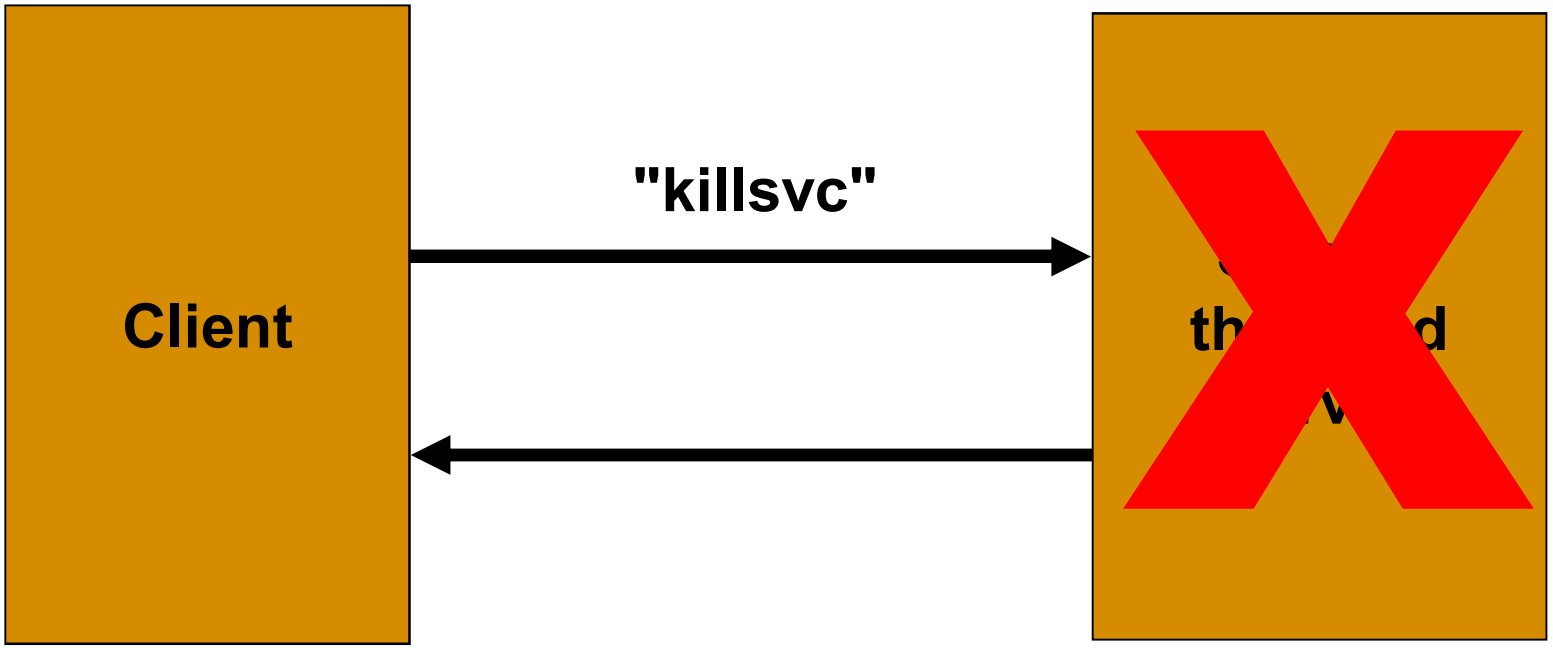
YOUR PROGRAMS



The messages being exchanged



The messages being exchanged



Overview

- Will have to implement a very basic talk pair
- Will have to write
  - A ***client*** that will send one request, wait for one reply then terminate.
  - A single-threaded ***server*** that will wait for requests from multiple clients

Client side

- Client will
  1. Prompt the user for server host name and port number
  2. Create a socket
  3. Prompt the user for a car license number
  4. Send the number to the server
  5. Receive the server’s reply and print it out
  6. Terminate

Server side

- Server will
  1. Read in the day's stolen car database
  2. Create a socket
  3. Bind an address to that socket
  4. Wait for incoming messages
  7. Receive and print out a message from client
  8. Consult its DB and send a Yes/No reply to the client
  9. Wait for next client message request

Server side

- Server will
  1. Read in the day's stolen car database
  2. Create a socket
  3. Bind an address to that socket
  4. Wait for incoming messages
  7. Receive and print out a message from client
  8. Consult its DB and send a Yes/No reply to the client
  9. Wait for next client message request

Loop will end when the server gets a **KILLSVC** message from one of the clients


Communicating through
sockets

UDP socket calls (I)

- **`socket(…)`**
  creates a new socket of a given socket type
  (*both client and server sid*es)
- **`bind(…)`**
  binds a socket to a socket address structure (*server side*)
- **`close(…)`**
  close socket(*server side*)

UDP socket calls (II)

- **`sendto(…)`**
  sends a message (*both sides*)
- **`recvfrom(…)`**
  receive a message and learn the address of its sender (*both sides*)

Summary

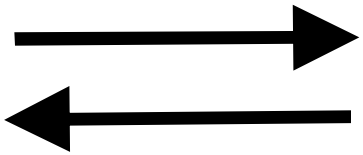
- **Client side:**

```
csd = socket(…)

sendto(csd, …)
recvfrom(csd, …)
```

- **Server side:**

```
ssd = socket(…)
bind(…)
recvfrom(ssd, …)
sendto(ssd, …)
```



Bad news and good news

- The bad news is that socket calls are somewhat esoteric
  - Might feel you are not fully understanding what you are writing
- The good news is most of these mysterious options are fairly standard

Some examples (I)

```
// create socket
if ((s = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
   return(-1);
```

- So far so good

Some examples (II)

```
gethostname(myname, MAXHOSTNAME); // skip for WSL!
// get host address structure
hp = gethostbyname(myname);
sa.sin_family = hp->h_addrtype; // host address
sa.sin_port = htons(portnum);   // set port number
//bind address to an existing socket
if (bind(s, &sa, sizeof(struct sockaddr_in)) < 0) {
   close(s);
   return(-1);
} // if
```

Picking a port number

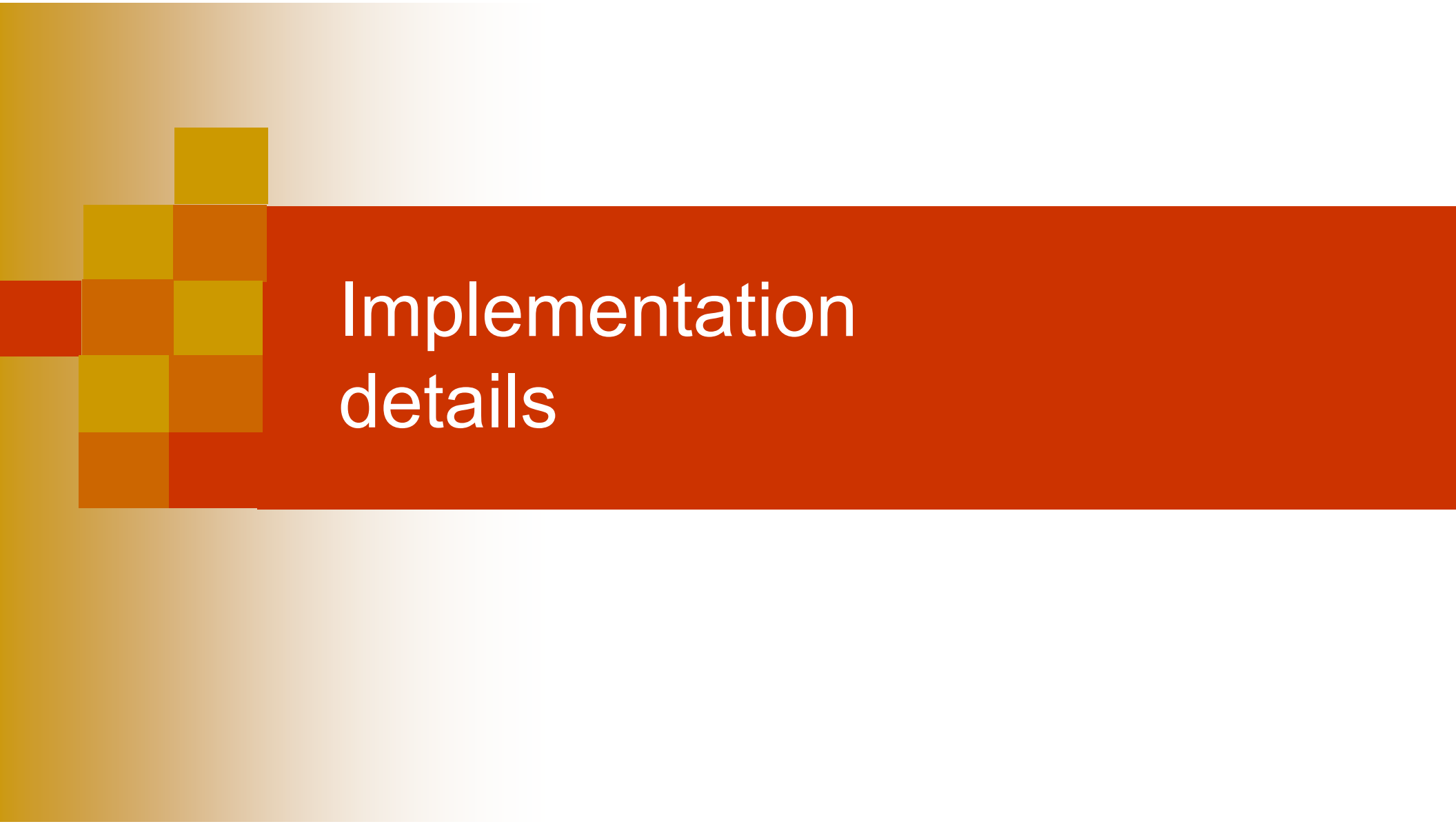
- Your port number should be
  - ***Unique***
    - *Should not interfere with other students' programs*
  - ***Greater than or equal to* 1024**
    - *Lower numbers are reserved for privileged applications*

Some examples (III)

```
// send a message and return number of bytes sent
n = sendto(
            sd,                  // sender's socket
            buffer,              // the message
            strlen(buffer),      // its length in bytes
            0,                   // default flag values
            (const struct sockaddr *)&server, // receiver
            length               // length of struct sockaddr
           );
```

Some examples (IV)

```
// receive a message, return number of bytes sent and
learn addresss of sender
n = recvfrom(
              sd,                    // receiver's socket
              buffer,                // the receiving buffer
              BUFSIZE,               // its length in bytes
              0,                     // default flag values
              (const struct sockaddr *)&from, // sender
              &length              //length of struct sockaddr
             );
```


Implementation
details

Doing networking assignments on your PC

- The host name of a Windows PC does not include its domain
  - **`jfparis@Odeon:~$ hostname`**
    `Odeon`
- **`hp = gethostbyname("Odeon");`**
  does ***not always*** work
- Use instead **`localhost`**
  `hp = gethostbyname(localhost);`

The stolen car database

- Very short file
- Read in by the server
  - Each line will contain one license number
    - **HIOFCR**
      SHKSPR
      2DIE4
      …
  - No spaces and no more than 8 characters

Some good tutorials

- https://www.geeksforgeeks.org/udp-server-client-implementation-c/
- https://www.programminglogic.com/sockets-programming-in-c-using-udp-datagrams/
- https://www.softprayog.in/programming/network-socket-programming-using-udp-in-c

- Can lift from them all the code you need.