Socket TCP

Reliable connection-oriented transport

Download, compile and run

- The TCP server returns “Hello UPO student!” at each client request
  - Download and extract myFirstNetworkApp.zip archive from the course website

    unzip myFirstNetworkApp.zip
  - Compile the source code

    gcc server.c –o server

    gcc client.c –o client
  - Launch the client and the server

    ./server <listeningPort>

    ./client <serverIP><serverPort>

First steps

- Once the client/server work on your machine
  - change the server message (be creative)
  - add port number as input parameter for both client and server inside the code (not as parameter on the command line)
    - hint: convert string to int
    - If you use the LAB PC, it must be between 10000 and 12000 (other ports are blocked)
  - try to make the client and server interact

Connection problems

- Try to connect to a wrong IP
- Try to connect to a correct IP without a listening server
- Try to connect to a correct IP using a port not allowed (e.g., <10000 or >12000 if you use the Lab PC)

Exercise II: dynamic answer

- Client connects to the server
- Server returns the current date and time

```
#include <time.h>
…
time_t ticks = time(NULL);
snprintf(buff, sizeof(buff), "%.24s\r\n", ctime(&ticks));
```

Exercise III: echo server

- Client
  - For X times, it sends a string to the server and it prints out the answer
    - input X and strings
- Server
  - It receives a strings and it returns the same string back

Exercise III: hints

- Clean the buffer
- To read sentences (not just a word), you can use fgets() instead of scanf()
  - fgets() reads also the'\n'

Exercise III.b: echo server, advanced

- Client
  - It sends the number of interactions, and then it sends each string read from stdin

- Server
  - It receives the number of interactions and then it replies with the same string received
    - The server closes the connection after the last string is sent back

Exercise III.c: echo server, complete

- Client
  - sends the number of interactions
  - sends the strings
  - sends ”bye”
  - waits for server “ack”
- Server
  - receives the number of interactions
  - receives the strings and replies the strings back
  - waits for “bye”
  - sends “ack”

Exercise IV: maxServer

- Client
  - sends a number (from stdin)
  - prints the number received from the server
- Server
  - receives a number
  - returns the highest number received so far