

# Assignment 2: UDP Socket Programming

Computer Networks (CS-UH 3012) - Spring 2022

## 1 Code of Conduct

All assignments are graded, meaning we expect you to adhere to the academic integrity standards of NYU Abu Dhabi. To avoid any confusion regarding this, we will briefly state what is and isn't allowed when working on an assignment.

1. Any document and program code that you submit must be fully written by yourself.
2. You can discuss your work with fellow students, as long as these discussions are restricted to general solution techniques. In other words, these discussions should not be about concrete code you are writing, nor about specific results you wish to submit.
3. When discussing an assignment with others, this should never lead to you possessing the complete or partial solution of others, regardless of whether the solution is in paper or digital form, and independent of who made the solution.
4. You are not allowed to possess solutions by someone from a different year or section, by someone from another university, or code from the Internet, etc.
5. There is never a valid reason to share your code with fellow students.
6. There is no valid reason to publish your code online in any form.
7. Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g. if the solution matches that of others), we reserve the option to let the student explain why this is the case. In case doubts remain, or we decide to directly escalate the issue, the suspected violations will be reported to the academic administration according to the policies of NYU Abu Dhabi. More details can be found at:

<https://students.nyuad.nyu.edu/academics/registration/academic-policies/academic-integrity/>

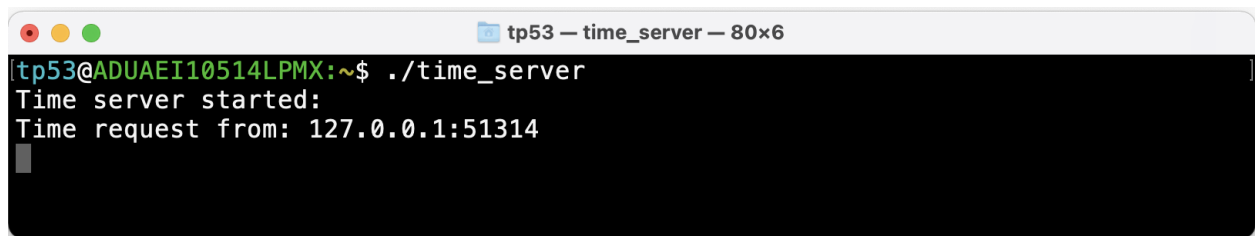
## 2 Assignment Goal

The goal of this assignment is to implement a UDP client and server using socket programming in C. This will give you practice in working with sockets, understanding their concepts and getting familiar with the UDP transport protocol.

### 3 Task Description

The task of this assignment is to write two C programs that can exchange messages using the UDP transport protocol. The task consists of two implementations: a server and a client.

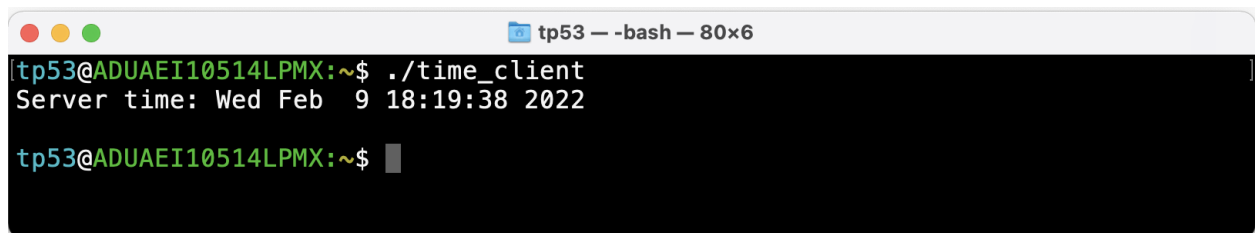
The UDP server program implementation opens a UDP socket (`SOCK_DGRAM`), listens on port 9000 for incoming messages and replies to **every** client message received with the current system time and date. Furthermore, the server implementation should print a message for every incoming message indicating that a message has been received and from which IP address and port number, see Figure 1. Please note that the server should not terminate after replying to a client and should handle further incoming requests in the same fashion as described above.

A terminal window titled "tp53 — time\_server — 80x6" showing the execution of the time\_server program. The prompt is [tp53@ADUAEI10514LPMX:~\$] and the command ./time\_server has been entered. The output is "Time server started:" followed by "Time request from: 127.0.0.1:51314".

```
[tp53@ADUAEI10514LPMX:~$] ./time_server
Time server started:
Time request from: 127.0.0.1:51314
```

Figure 1: Example program output of the UDP server

The UDP client program implementation opens a UDP socket, sends an arbitrary message, e.g. "Time request", to the server on port 9000 and waits for a reply from the server. The reply, i.e. the current time and date of the server, from the server should be printed on the terminal, see Figure 2. After printing the reply, the client should terminate.

A terminal window titled "tp53 — -bash — 80x6" showing the execution of the time\_client program. The prompt is [tp53@ADUAEI10514LPMX:~\$] and the command ./time\_client has been entered. The output is "Server time: Wed Feb 9 18:19:38 2022".

```
[tp53@ADUAEI10514LPMX:~$] ./time_client
Server time: Wed Feb 9 18:19:38 2022
```

Figure 2: Example program output of the UDP client

For the client to server communication, you can use the localhost IP address 127.0.0.1 which allows you to run the client and server in two different terminals on the same computer. When creating the `sockaddr_in` `srvaddr` in the client, you can use the following code snippet to set the destination IP address of the server to 127.0.0.1:

```
srvaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

## 4 Grading

Description	Score (/4)
Successful compiling of the programs using a Makefile	0.5
Successful client-server communication using UDP sockets on port 9000	2.5
Printing of events, as depicted in Figure 1 and 2	0.5
Usage of meaningful comments	0.5

## 5 Submission Details and Policy

**Submission Deadline:** The deadline of this assignment is after 6 days of its release via Brightspace. No extensions will be given.

**Submission Format and System:** You can directly submit your files (C files and Makefile) as a zip file on Brightspace (<https://brightspace.nyu.edu/>). Due to technical limitations, submissions via email are not accepted.

**Late Submissions:** Late submissions will be penalized by 10% per 24 hours, with a maximum of 3 days late. In case of a late submission, please upload your zip file to Brightspace and inform the TA and the professor.