Building Bridges in Computer Networks: A Nifty Assignment for Cross-Language Learning and Code Refactoring

Ildar Akhmetov
i.akhmetov@northeastern.edu
Northeastern University
Vancouver, BC, Canada

Logan W. Schmidt
l.schmidt@northeastern.edu
Northeastern University
Vancouver, BC, Canada

ABSTRACT

This nifty assignment is designed to introduce students to fundamental networking concepts, such as the client—server model, sockets, and network protocols, through hands-on experience with cross-language programming and code refactoring. The assignment targets students without a prior background in computer science. By engaging students with starter code in C, Python, and Java, the assignment facilitates the understanding of protocols across different programming languages and emphasizes the importance of code reusability and refactoring. Students are tasked with extending server functionality to include custom commands and are encouraged to use AI tools for code development. This approach aims to prepare students for the evolving pedagogical landscape where AI-assisted development plays a significant role in software engineering practices.

CCS CONCEPTS

• **Social and professional topics → Computing education**; • **Networks → Network protocols**.

KEYWORDS

networks, sockets, client—server, assignment

ACM Reference Format:
Ildar Akhmetov and Logan W. Schmidt. 2024. Building Bridges in Computer Networks: A Nifty Assignment for Cross-Language Learning and Code Refactoring. In *The 26th Western Canadian Conference on Computing Education (WCCCE '24), May 2–3, 2024, Kelowna, BC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3660650.3660665

1 INTRODUCTION

Teaching computer networks is a challenging educational task [5]. The topic is complex, and it is usually taught in a dedicated course. However, in many circumstances, it is beneficial to introduce the foundational concepts of computer networks early, and CS2 or a similar course is a good place to do so, as demonstrated by some excellent nifty assignments [2–4]. Assessing students on something that they just briefly touched upon is challenging, so our main goal for this assignment was to build awareness of the most fundamental principles of computer networks. We want to introduce the client—server model, sockets, and network protocols, and to do so in a way that would be engaging and relevant to students. We also aim to build on the programming skills students have acquired in the current and prior courses. This is achieved by using multiple programming languages (C, Python, and Java), refactoring code, and reusing code from previous assignments.

In this assignment, students begin by examining starter code that establishes a basic network communication setup using sockets. The server is written in C, and there are three clients with similar functionality written in C, Python, and Java. Students are tasked with refactoring both the server and client components to enable parameterized ports (passed as command line arguments). Next, students are asked to extend the functionality of the server by implementing several commands, such as arithmetic calculations and list sorting, along with the help command. Throughout the assignment, the use of AI tools (ChatGPT or GitHub Copilot Chat) is encouraged to help students understand complex concepts and facilitate the implementation of new features.

2 AUDIENCE AND DIFFICULTY

The assignment was initially designed for the CS 5008 (Data Structures, Algorithms, and Their Applications within Computer Systems) course at the Khoury College of Computer Sciences at Northeastern University in Vancouver. The course is part of the Align bridge program, which is designed for students with no prior computer science background [1]. The students in the course have diverse backgrounds, and many of them have experience in other fields.

We aimed to keep the assignment engaging, especially for students with no prior CS background, yet reasonably challenging. The assignment is designed to be easy to medium in difficulty, and it can be solved in one week. Students report spending between 5 and 15 hours on the assignment. The assignment is scaffolded in order to move the focus from writing code to code comprehension, refactoring, and code reusability.

In our setup, students use the C language in the course that includes the assignment, and they have already learned Python in the previous course (similar to CS1). Java is introduced in the course that most students take in parallel. This setup informed the selection of programming languages for the assignment. However, the assignment is designed to be language-agnostic, and it can be adapted to other languages or limited to only one or two languages.

3 STRENGTHS

The assignment has several strengths:

WCCCE '24, May 2–3, 2024, Kelowna, BC, Canada
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0997-5/24/05.
https://doi.org/10.1145/3660650.3660665

- By using multiple languages, students understand that protocols are language-agnostic, and how protocols are important for different languages to "talk" to each other.
- The starter code that we provide is already working, so it's a starting point for students to study and refactor. This is a good way to introduce students to the concept of refactoring.
- The assignment encourages code reusability. Students choose a sorting algorithm that they have implemented in a previous assignment (there are a few to choose from) and integrate it into the server code.
- The assignment is real-world: the server and clients can really connect to each other.
- The assignment is autograded (except the extra command), which makes it well suited for large classes.

4 WEAKNESSES

In our experience with the assignment, we have identified several weaknesses:

- If students use a shared server (our setup), port conflicts can happen. We encouraged students to use a wide range of ports and introduced the *ss* command to check for used ports to mitigate this issue, but this steepens the learning curve for beginners.
- In addition to the required commands (*calc*, *sort*, and *help*), we wanted students to implement an extra command of their choice. This is interesting, but it requires manual grading.
- String handling in some languages, including C, can have small complications that lead to difficulties for students.
- The students are only asked to connect a client and server running on the same machine; they would require additional system setup instructions and networking knowledge to connect across machines.

5 DEPENDENCIES

The dependencies for this assignment are minimal: a POSIX system that can run C, Python, and Java. The assignment was tested on Rocky Linux 8.8 and Ubuntu 22.04. It should be possible to run the assignment on Windows, but we have not tested it.

The autograded part of the assignment is implemented as a custom Docker container and relies on Gradescope. However, the autograding scripts are written in Python (using the *unittest* module to run the test suite, and the *subprocess* module to run the server and clients).

6 VARIANTS

The assignment can be easily modified to suit different courses and student backgrounds:

- Based on the languages students know, the assignment can be modified to use different languages. The core idea of the assignment is to show that network protocols are language-agnostic, so adding more languages can be beneficial.
- The required commands can be easily changed, making it very easy to have a new version of the assignment every semester. In addition, if students have implemented a specific algorithm in a previous assignment, they can be asked to include it in the server as a command (promoting code reusability).
- Students can work in pairs where one student runs a server and another student uses their client to connect to it. This would emphasize the principles of client—server architecture even more. However, such a setup is more difficult to manage in larger classes.
- The first point we mentioned in the weaknesses section (port conflicts) could actually lead to an interesting variation that we would like to test in the future. Each student could have a unique flag number, and students would be tasked with connecting to N other students' sockets and getting their flag number with a simple handshake function. This could also mean implementing basic authentication and even encrypting the handshake logs, which would be an interesting way to introduce cybersecurity in CS1/CS2.
- Another option would be to run a "black box" server that students have to connect to, where each student is given a unique flag number that they have to get in response to certain commands sent to the server (e.g., a hash of some ID number unique to the student), with the autograder set up to check if a given flag is valid.

7 SUMMARY

In the new pedagogical reality where AI-assisted development is becoming more common, it is becoming more important to shift the focus to reading, debugging, and refactoring code, while constantly switching between different programming languages. This assignment is designed to help students develop these skills. While it does not go deep into the theory of computer networks, it provides a good starting point for students to understand the basics of network communication and to build on their programming skills. The assignment is designed to be engaging and relevant to students, and it is well suited for large classes.

We tested this assignment with 125 students in the Align bridge program at Northeastern University in Vancouver. The assignment was well received by the students and we plan to use it in the future, experimenting with the variations listed above.

8 ACKNOWLEDGMENTS

We would like to thank Xiaolai (Sean) Chen for developing autograding scripts for this assignment.

REFERENCES

[1] Carla Brodley, Megan Barry, Aidan Connell, Catherine Gill, Ian Gorton, Benjamin Hescott, Bryan Lackaye, Cynthia LuBien, Leena Razzaq, Amit Shesh, et al. 2020. An MS in CS for non-CS Majors: Moving to Increase Diversity of Thought and Demographics in CS. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 1248–1254.

[2] Samuel Hsieh and Sean Wolfe. 2021. Creating a server as a nifty assignment: nifty assignment. *Journal of Computing Sciences in Colleges* 36, 6 (2021), 71–73.

[3] John M Hunt. 2010. Nifty assignment: concurrent multi-user battleship. *Journal of Computing Sciences in Colleges* 26, 2 (2010), 215–219.

[4] Arnold L Patton. 2006. Hack this! reverse engineering a network server: nifty course assignments. *Journal of Computing Sciences in Colleges* 21, 4 (2006), 69–72.

[5] Marina Prvan and Julije Ožegović. 2020. Methods in teaching computer networks: a literature review. *ACM Transactions on Computing Education (TOCE)* 20, 3 (2020), 1–35.