Teaching Networking in the Age of Generative AI: An Experience Report on Teaching Socket Programming

Daniel T. Fokum
University of the West Indies
Kingston, Jamaica
daniel.fokum@uwi.edu

Amit N. Ramkissoon
University of the West Indies
Kingston, Jamaica
amit.ramkisoon@uwi.edu

Abstract

The widespread availability of generative AI tools requires a rethinking of various aspects of Computer Science education. In the particular area of computer networking, there are questions on how introduction to networking courses are structured, whether one should be teaching socket programming, or whether the structure of homework should be changed. In this paper we focus on the teaching of socket programming and show how grades have evolved in a socket programming project for an undergraduate computer networking course. We provide some suggestions on how to improve assessment of computer networking projects in, what we believe is, a scalable manner.

CCS Concepts

• **Social and professional topics** → **Computing education**; • **Networks**;

Keywords

Networking Education, Assessment, Socket programming

ACM Reference Format:
Daniel T. Fokum and Amit N. Ramkissoon. 2026. Teaching Networking in the Age of Generative AI: An Experience Report on Teaching Socket Programming. In *SIGCOMM '26: SIGCOMM Education Workshop 2026: Networking Education for the AI Generation (A4NE).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/9876543.1234567

1 Introduction

The understanding of Socket programming is a fundamental part of the Computer Networking curriculum. Socket programming is one of the key topics in an undergraduate computer networking course, with the topic being found in textbooks such as [12, 17]. As explained by [18], this is an essential learning outcome for Computer Science students who appreciate the benefit of understanding how to design and implement their own networking protocols. Socket programming as the name suggests, is the design and implementations of computer programs that involve machine-to-machine communication via the use of computer network sockets and ports [13]. A port can be thought of as a "little window" in a computer via which a message can enter. This message travels to this window via a channel or tunnel known as a socket. Socket programming involves writing and executing computer programs that make use of this knowledge and allowing the programs to send and receive messages over a computer network, inherently building a distributed system.

Understanding this fundamental concept that lies at the core of Computer Networks is essential for a good computer science student. This learning ensures that students grasp how computer networks work, via a practical application. They learn how messages are packaged from transmission, the various protocols involved in the transmission and how addressing works to ensure the transmission of a message. This concept also extends into their understanding of critical methodologies like client-server, peer to peer, centralized, decentralized, infrastructure and infrastructureless networks.

Generative AI (GenAI) is a subfield of computer science that has become one of the greatest disruptors of our time. Generative AI involves the use of Large Language Models to produce answers based upon text analysis and knowledge engineering. With the advent of GenAI, the way in which lessons are taught in the classroom have also become disrupted. Students have embraced GenAI as a tool to assist in their learning.

The widespread availability of generative AI tools such as ChatGPT [16] or Gemini [10] means that students can easily submit many socket programming problems of yesteryear to generative AI and receive fully functional solutions without truly understanding how socket programming actually works. According to [14] AI can be used to develop network configurations, network program code, network documentation and debug outputs. GenAI, however, is also highly used by students as a tool to provide quick and easy answers to assessment that previously required them to spend time reading and then developing answers. This problem of GenAI in education has become a global phenomenon, with universities struggling to develop assessments that can still honestly test student understanding of a given topic.

This ongoing struggle with GenAI is also present in the world of teaching socket programming. Given that GenAI is capable of solving most problems and providing answers in a few seconds, socket programming tasks are easily resolved by the said GenAI. In our experience, we have been dealing with this modern issue of GenAI in socket programming and have also implemented some unique solutions to the said problem.

In late 2025 someone initiated a discussion in a Slack group on teaching computer networking. The initial questions included ideas such as: whether the socket programming assignments had been changed, whether socket programming was still being taught, whether the structure of homework had been changed, or whether

A4NE 2026, Denver, CO

ACM ISBN 978-1-4503-9201-3/22/07
https://doi.org/10.1145/9876543.1234567

the way in which we deliver content had been changed. All of these questions are quite appropriate, and in this paper we focus on our recent experience with teaching socket programming.

We submit that socket programming is a concept that Computing students ought to be able to understand and use comfortably. This position is in line with the ACM's Computer Science Curricula 2023 [11], which lists socket programming as a core part of the undergraduate computer science curriculum.

The following research questions inform this experience report:

RQ1 How have socket programming grades evolved in an undergraduate computer networking course following the widespread availability of generative AI?

RQ2 How should the assessment of socket programming content change given the widespread availability of generative AI?

The rest of this paper is laid out as follows: Section 2 documents our experience assessing a socket programming exercise at an institution in a small island developing state. In Section 3 we discuss some implications for assessment based on our experiences, and in Section 4, we provide concluding remarks.

2 Experience

As previously stated, the understanding of socket programming is essential to every computer science student. In our experience teaching socket programming provides the student with a visualization of how the various layers of the network stack interact with each other. At our institution, students have always had a socket programming project as part of their computer networking course. The socket programming was then assessed via assignments and proctored examinations, testing the students' understanding of how to establish the connection with both protocols and how they both differ. Initially, the projects were essentially extensions of the core ideas of the socket programming concept taught in [12]. Over time, the project evolved such that students were given a simplified text-based protocol to implement. This pedagogical change was made to reinforce the class content on protocols and to hone their socket programming skills. Over time, the students were also required to start their clients and servers from the command line, and to read certain start-up parameters, e.g., host name and port numbers, from the command line.

Historically, students found the socket programming exercise to be challenging, but they also appreciated the connection to the course content. The socket programming exercise also proved to be a good way to introduce some smaller concepts, e.g., showing students how a "toy" Diffie-Hellman key exchange worked. The socket programming exercise also provided a scaffold upon which one could develop another project where a client and server used a simplified version of RSA and simplified AES [15] to simulate having a client and server exchange a session key securely in order to have confidential communications.

Table 1 shows summary statistics for the socket programming project for five academic years where the course has been taught by the same instructor[1]. The project has always been designed by that same instructor. It should be noted that the scores in Table 1 have not been modified by curves. Over the years under review, the number of attempts of the project has ranged from a low of 173 in 2022/23 to a high of 260 in 2024/25. The first quartile (Q_1) of the project scores jumped from 9% in 2020/21 to 71% in 2022/23, and then dropped in 2023/24, with major increases in each of the following years. Similarly, the mean grades jumped from 45% to 88% between 2020/21 and 2025/26, while the median grades jumped from 43% to 97% over the time frame.

[1]The 2021/22 academic year is excluded because the instructor was on sabbatical.

Table 1: Summary statistics for the socket programming project

	Academic Year				
	2020/21	2022/23	2023/24	2024/25	2025/26
n	238	173	200	260	250
Q_1	9	71	51	77	94
Mean	45	77	75	81	88
Median	43	93	94	94	97
Q_3	81	97	100	100	100

The fact that the mean grade was higher than the median in the 2020/21 academic year indicates that there were students with high scores who "pulled the class average up." In this case, the median score was more representative of the class performance. From the 2022/23 academic year onward, we observe that the median is markedly greater than the mean. This time, this suggests that the scores are "pulled down by abnormally low values." It is also worth pointing out in Table 1 that during the 2025/26 academic year, the first quartile score on the socket programming exercise is less than the mean. This is because there are extreme low outliers on the socket programming assignment. It is highly likely that these low scores from 2023/24 onward are from students whose attempts did not make use of generative AI, and who failed to make use of office hours to get assistance. The jump in student performance from 2022/23 to 2023/24 is most likely due to the widespread availability of large language models (LLMs), given that LLMs could already generate code from 2021 [8, 9].

While there has been a marked increase in performance on the socket programming project, the magnitude of this increase is not reflected in final exam performance over this period. In fact, there are students who have had perfect scores on programming assignments who have failed to answer exam questions that are based on the programming projects. Section 3 we discuss the implications of these results for student assessment.

Our institution has a position on AI [2] that states:

> ... generally supports the responsible use of AI tools (Generative and otherwise) in all of its activities, provided that such use is appropriate for its context, and aligned with The University's Core Values.

The institution's AI policy [2] further states that "all courses have a clear statement defining acceptable use of AI for its students that is tailored for the learning outcomes of the course but consistent with the overarching university AI policy framework. Individual course instructors are free to allow or disallow some or all uses of AI tools provided there is a sound pedagogical basis; nevertheless, no programme completely shuns the use of AI across all its courses." Given our university's over-arching position on AI, students were permitted in the 2023/24 academic year to complete the socket

programming exercise using an LLM for extra credit. Anecdotally, it is clear that more and more students are using AI to complete the programming exercises in the networking course, even though they have been asked to complete the exercises themselves. In the past, MOSS [7] was used to test all student submissions for software plagiarism; however, this became harder to do as 1) the course enrollment increased; and 2) as MOSS limited the number of daily submissions per user. In the coming academic year, tools such as compare50 [3] and JPlag [4] will be investigated for their role in detecting plagiarism in student submissions.

The evolution of the project grades shows that students need to be educated and reminded about acceptable uses of AI

3 Discussion

Socket programming is an important part of an undergraduate computer networking course, and we believe that it is important for students to learn how to do it properly, even if Generative AI can carry out the task. In other words, it is important for students to learn the basics first. Thus, it is important to develop good (and accessible) socket programming exercises for students to work on. We believe that it is important to learn from what other academic communities do with respect to artifacts for assessing learning.

The ACM's SIGCSE community has an annual session at its technical symposium called Nifty Assignments [5], in which instructors of the first two tertiary-level computing courses (called CS1 and CS2) in the SIGCSE community share programming project ideas. The stated goal of the Nifty Assignments project is to gather "great CS assignments to make their ideas and materials freely available for the CSEd community" [6]. The Nifty Assignments session is, arguably, one of the most popular at any SIGCSE Technical Symposium.

More recently, the International Symposium on Computer Architecture (ISCA), which is co-sponsored by SIGARCH, has introduced a "Nifty Assignments" session at its associated Workshop on Computer Architecture Education [1]. Unlike SIGCSE, there is no repository of "Nifty Assignments" for WCAE; however, the objective is similar. The goal here is for instructors to share their favorite assignments with attendees [1].

Given what obtains in at least two learned societies, we submit that it will be useful for the SIGCOMM community to have a similar session on Nifty Assignments at its annual conference (or via some other technical forum) for instructors to trade good programming projects in the context of computer networking. Having such a forum will allow for the vetting of projects and it will also lead to better networking education.

In addition to having vetted computer networking programming projects, it is also useful for the networking community to discuss how to assess programming projects and homework in a scalable manner. Zhang *et al.* [20] has noted that the widespread availability of LLMs has changed significantly since 2022. Furthermore, they argue that it has become harder for instructors to assess student learning and effort. There are many emerging ideas on how to handle this assessment. Some might include forcing students to use a cloud-based IDE (integrated development environment) that prevents code from being pasted in, which prevents them from prompting an LLM and then copying and pasting that response in without any learning. Other approaches include using AI-driven oral examinations [19], or oral examinations that are conducted by teaching assistants (TA). In the latter case, the teaching assistants are given a set of questions that should be easily answered by students who have written their code themselves. It will be useful for the SIGCOMM community to discuss best practices around the grading of computer networking projects. It may also be useful for the community to coalesce around similar positions around what kind of AI use is permitted as well as the type of guidance on AI use that instructors should give to their students.

4 Conclusion

Socket programming is arguably an important part of an undergraduate computer networking course. Thus, it is important for students to learn this skill even when they have access to generative AI tools. This paper reviewed how student performance on a socket programming project had evolved over a four year period since the widespread introduction of generative AI. We showed that there has been significant improvement in project scores over that period; however, final examination grades have not improved as significantly over the period, leading one to conclude that the improvement in the project scores is largely due to AI use. We note that at least two ACM SIGs have "Nifty Assignments" sessions at their conferences, and we submit that the SIGCOMM community should consider including a similar session at its annual conference, or potentially a regularly scheduled workshop.

References

[1] 2023. Workshop on Computer Architecture Education: Early decision deadline approaching – TCCA. Web page. https://ieeetcca.org/2023/04/24/workshop-on-computer-architecture-education-early-decision-deadline-approaching/ [Accessed 06-06-2026].

[2] 2025. The UWI Artificial Intelligence Policy Framework. Web page. https://uwi.edu/obus/sites/obus/files/tmp/UWI%20AI%20Policy%20Framework.pdf [Accessed 07-17-2026].

[3] 2026. compare50 – compare50 Docs. Web page. https://cs50.readthedocs.io/projects/compare50/en/latest/index.html [Accessed 07-17-2026].

[4] 2026. GitHub - jplag/JPlag: State-of-the-Art Source Code Plagiarism & Collusion Detection. Check for plagiarism in a set of programs. Web page. https://github.com/jplag/JPlag [Accessed 07-17-2026].

[5] 2026. Nifty Assignments. Web page. https://nifty.stanford.edu/ [Accessed 06-06-2026].

[6] 2026. SIGCSE TS 2026 - Nifty Assignments - SIGCSE TS 2026. Web page. https://sigcse2026.sigcse.org/track/sigcse-ts-2026-nifty-assignments [Accessed 06-06-2026].

[7] Alex Aiken. 2026. MOSS - Plagiarism Detection. Web page. https://theory.stanford.edu/~aiken/moss/ [Accessed 07-17-2026].

[8] Russell Beale. 2025. Computer Science Education in the Age of Generative AI. arXiv preprint arXiv:2507.02183v1. https://arxiv.org/html/2507.02183v1

[9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG]. doi:10.48550/arXiv.2107.03374

[10] Google LLC. 2026. Gemini. Large language model. https://gemini.google.com [Accessed 01-06-2026].

[11] Amruth N. Kumar, Rajendra K. Raj, Sherif G. Aly, Monica D. Anderson, Brett A. Becker, Richard L. Blumenthal, Eric Eaton, Susan L. Epstein, Michael Goldweber, Pankaj Jalote, Douglas Lea, Michael Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, and Qiao Xiang. 2024. *Computer Science Curricula 2023.* Association for Computing Machinery, New York, NY, USA.

[12] James F. Kurose and Keith W. Ross. 2025. *Computer Networking - A Top-Down Approach* (9 ed.). Pearson.

[13] Rolou Lyn R Maata, Ronald Cordova, Balaji Sudramurthy, and Alrence Halibas. 2017. Design and implementation of client-server based application using socket programming in a distributed computing environment. In *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC).* IEEE, 1–4.

[14] Rene Molenaar. 2026. AI and ML in Networking. https://networklessons.com/network-automation/ai-and-ml-in-networking

[15] Mohammad A. Musa, Edward F. Schaefer, and Stephen Wedig. 2003. A Simplified AES Algorithm and its Linear and Differential Cryptanalyses. *Cryptologia* 27, 2 (2003), 148–177. doi:10.1080/0161-110391891838

[16] OpenAI. 2022. Introducing ChatGPT | OpenAI. https://openai.com/index/chatgpt/. [Accessed 01-06-2026].

[17] Larry Peterson and Bruce Davie. 2019. *Computer Networks: A Systems Approach* ("sixth" ed.). Elsevier.

[18] Hugh Smith. 2016. Network Programming-Beyond Sockets. In *2016 Rocky Mountain Section Conference.*

[19] Tim Tregubov and Pape Sow Traore. 2026. AI-Driven Oral Examinations for Code Assessment: Evaluating Understanding Beyond the Commit. In *Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.2* (USA) *(SIGCSE TS 2026).* Association for Computing Machinery, New York, NY, USA, 1728. doi:10.1145/3770761.3777032

[20] Yufan Zhang, Jaromir Savelka, Seth Goldstein, and Michael Conway. 2026. Changes in Coding Behavior and Performance Since the Introduction of LLMs. In *Proceedings of the LAK26: 16th International Learning Analytics and Knowledge Conference (LAK '26).* Association for Computing Machinery, New York, NY, USA, 844–851. doi:10.1145/3785022.3785075

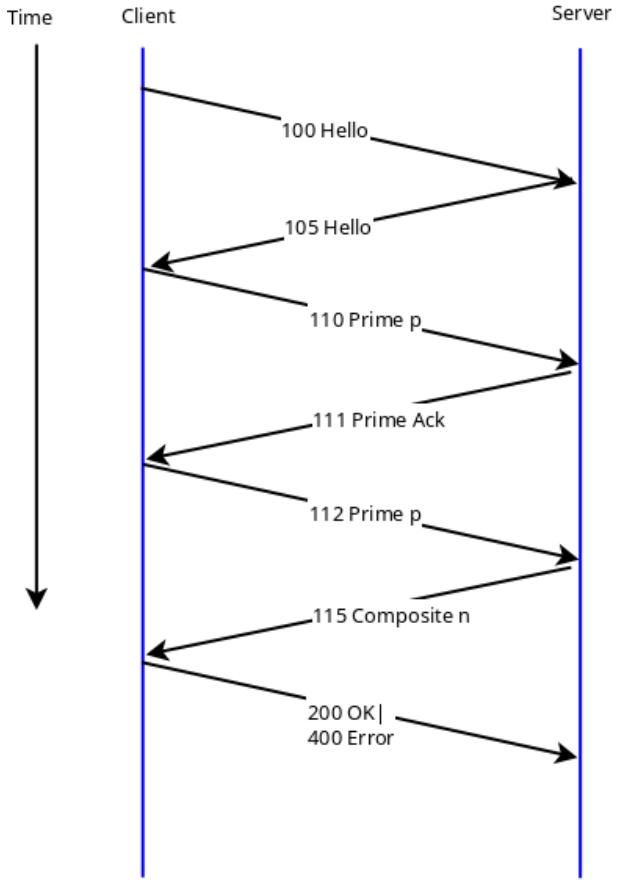

Figure 1: Message sequence exchanged between client and server

A Socket Programming Project from 2023/24

In this assignment, you'll write a client that will use sockets to communicate over TCP with a server that you will also write. Your client and server will exchange the sequence of messages shown in Figure 1

In this assignment, you will "simulate" a client sending prime numbers to a server. You will write a client that will communicate with a server that you will also write. On start-up, each client will print out a string saying: "`Client of Joan A. Smith`" and then it will open a TCP socket to your server and send a message (a string of characters) to your server containing the string "`100 Hello`". Upon receipt of the "Hello" message, the server will send its own hello message to the client as shown above. The server's hello message will contain the string "`105 Hello`". Upon receipt of this message, the client will prompt the user for a prime number, which it will then send to the server in the "`110 Prime` p" message, where p represents the prime that the user entered.

The server will acknowledge this message with a "`111 Prime Ack`" string, and the client will send the second prime with the "`112 Prime` p" message, where p represents another prime from the user. The server will then compute the product of the two prime numbers that it received, and then send that product down to the

client in the "`115 Composite` *n*" message, where *n* represents the product. If the server-computed product matches the product that the client has, then the client will send the "`200 OK`" message to the server, otherwise, it will send a "`400 Error`" message to the server. After sending any one of these messages, the server will close the connection socket that it has created for the client. The client will also close its connection to the server. The server and client will then terminate.

A.1 Server Implementation

Your server will create a string containing its name (e.g., "`Server of Joan A. Smith`"). As indicated above, the server will wait until the client says hello with a "`100 Hello`" message. The rest of the server operation is as described above.

A.2 Client Implementation

Your client will create a string containing its name (e.g., "`Client of Joan A. Smith`"). The client will then send a "`100 Hello`" message to the server. The rest of the client operation is as described above.

A.3 Additional Details

Note that a short design document is required. You should program each process to print an informative statement whenever it takes an action (e.g., sends or receives a message, detects termination of input, etc.), so that you can see that your processes are working correctly (or not!). This also allows the grader to also determine from this output if your processes are working correctly. You should hand in screen shots of these informative messages. Your document should also discuss how you tested the program to ensure that it meets the specification above.

A.3.1 Programming Notes:

- You must choose a server port number greater than 1023 (to be safe, choose a server port number larger than 5000).
- Starter code is provided for you on [the learning management system]
- You must write your programs in Python3. Details about the socket module can be found at http://docs.python.org/3/library/socket.html and Socket Basics file posted on [the learning management system].
- Many of you will be running the clients and senders on the same UNIX machine (e.g., by starting up the receiver and running it in the background, then starting up the relay in the background, and then starting up the sender. This is fine since you are using sockets for communication these processes can run on the same machine or different machines without modification. Recall the use of the ampersand to start a process in the background. If you need to kill a process after you have started it, you can use the UNIX `kill` command. Use the UNIX `ps` command to find the process id of your server.
- Make sure you close every socket that you use in your program. If you abort your program, the socket may still hang around and the next time you try and bind a new socket to the port ID you previously used (but never closed), you may get an error. Also, please be aware that port ID's, when bound to sockets, are system-wide values and thus other students may be using the port number you are trying to use.
- *Do not* open up a separate socket for each message that you send. Doing so is poor programming style, and your grade will be affected negatively for doing so.

A.3.2 Rules:

- The project is designed to be solved independently.
- You may not share submitted code with anyone. You may discuss the assignment requirements or your solutions away from a computer and without sharing code, but you should not discuss the detailed nature of your solution. If you develop any tests for this assignment, you may share your test code with anyone in the class. Please do not put any code from this project in a public repository or in a private repository that you share with other students.