Transport It Your Way: Rethinking A Networking Project to Emphasize Design

Alexander Krentsel*
UC Berkeley
Berkeley, CA, USA

Tess Despres*
UC Berkeley
Berkeley, CA, USA

Sylvia Ratnasamy
UC Berkeley
Berkeley, CA, USA

Abstract

AI is quickly shifting what it means to be a computer scientist in networking. As coding agents become more capable, the day-to-day practices of building systems are quickly evolving. These shifts force us to reexamine how we teach networking, raising many open questions including: should syntax still be emphasized, how should we scaffold student coding assignments, and how should we assess what students have learned from artifacts? With these questions in mind, we focus on revamping an upper division networking and systems class project in this proposal. The previous project, which centers implementation, can now be completed end-to-end by a coding agent. To mitigate the impact of this reality, we propose a redesigned project that shifts effort towards system design and tradeoff analysis, while granting students the autonomy to use modern coding tools for their implementations. We also discuss how this approach could generalize to other projects in upper division systems classes.

1 Introduction

AI, and coding agents in particular, are reshaping both how computer scientists spend their days and how we teach computer science. Developers are reporting widespread adoption of AI coding tools, increasing the amount of code we produce [6, 12]. The impact on students in computer science is troubling. Students who use AI coding tools to complete assignments, originally designed to be completed independently, skip over the productive struggle that comes from iteratively debugging their own code [8, 13]. Additionally, uneven access to paid models threatens to create a new digital divide among students. At the same time, there is some hope that AI models can serve as effective interactive tutors, enabling increased access to personalized education [11].

A core emerging challenge, is that these tools (*i.e.* Claude Code, Codex, etc.) can now complete many class assignments without human help. Many class assignments are structured in a way that helps students along a path, but agents can use the same project scaffolding as context, increasing the likelihood of reaching the correct answer. Additionally, the CS education community has invested heavily in autograders and test suites to scale up class sizes, but these can also provide an iterative feedback signal to an agent. The success of AI agents, at least when it comes to completing projects, is a core challenge for educators because now it is difficult to assess if students have truly internalized the material.

Systems classes are particularly impacted by AI coding tools since many of our assignments concentrate on the applied. This is because many of the core insights in systems classes only become apparent through exploring an implementation. For example, an operating system class implements a scheduler, a security class carries out penetration testing, and a networking class implements routing algorithms. By construction, these assignments couple learning about *system design* with the act of *implementation*.

In the presence of capable coding agents, this coupling fails both students who use AI, and less intuitively those who do not. A student who hands the assignment to an agent receives a working artifact in a single shot, forgoing the implementation practice that was meant to reinforce the underlying systems principles. A student who refuses to use AI completes the assignment as originally specified, but graduates without exposure to the tools (*i.e.* coding agents and the workflows that surround them) that increasingly define real-world systems work. The second may be acceptable and even desirable in an introductory course. However, our institution's upper division networking class is often one of the last systems courses students take before entering industry or a graduate degree, and therefore we feel it is important for students to learn to use modern AI coding tools.

In light of these challenges, we ask, how do we educate the next generation of systems thinkers, while preparing them for real-world practices of computer science? We argue that to do this, systems education must now decouple *implementation* from *design*. Assignments that emphasize implementation, or that weight implementation and design equally, will continue to lose pedagogical value as implementation becomes fully automatable. Thus, we believe a more durable curriculum must center on design choices.

Centering design choices is not a new concept in CS education. For example, the popular textbook "How to Design Programs" focuses on design recipes and explicit guidelines rather than programming languages and domain knowledge [5]. Departments have even reorganized CS education to have students tackle open-ended design problems each

*These authors contributed equally to this work.

semester, borrowing from architecture studio practices, with promising results [4, 7]. Our thinking builds on this line of work, similarly focusing on system design and tradeoffs, but aiming to push it further by leaning on the very AI tools that we recognize as a challenge for current curricula.

Our perspective is to view AI's ability to automate implementation assignments as a silver lining: when implementation work becomes automated, a larger percent of project time can be spent on design, which is ultimately the goal of the class. Building a complete new system was previously beyond the reach of most undergraduate assignments. In fact, the fun of building new systems is often still reserved for PhD-level work. Therefore, courses reasonably settle for implementing a single algorithm or a partial existing system in isolation. With implementation cost reduced, a larger implementation scope is now feasible, and the higher-level questions of why a design exists, for whom it is built, and what consequences it carries can move to the center of the assignment. With this larger scope, we believe students can now both design their own networked systems and implement them with the help of modern tools.

As a step forward in this direction, we propose a project redesign for the undergraduate networking class at our institution. The proposal draws on our experiences as part of the teaching team over 4 semesters, spanning 2018 to 2024, before, during, and after the rise of capable coding agents. Our goal with the project redesign is to teach students to build systems in an environment with AI, and perhaps even more importantly to consider the design tradeoffs such as, which constraints the deployment context imposes, and what the protocol does, for whom. We are not aiming to make the assignment AI-proof, but rather to enable students to work with AI to build new and creative networked systems.

Concretely, our proposal centers on having students design a transport layer protocol for one of three given workloads and environments. The project is intentionally open ended: students can create their own protocol. To provide some structure, we plan to give students a common IP datagram abstraction and test environment to visualize how their protocol is working. In the rest of this short paper, we give background on ourselves and the class in §2, report on our past experiences running the class in §3, outline our project design in §4, and close with broader implications in §5.

2 Background

In this section, we discuss our positionality as instructors, provide context on the course, and outline our project goals.

2.1 Our Positionality

This paper and project is shaped by who we are and our backgrounds. The two co-first authors are both PhD students in computer science at UC Berkeley, a large US-based public university. We are drawing heavily from both our experience taking this or similar courses, our experiences on the teaching team, and industry engineering experience.

2.2 Class Context

The networking class at UC Berkeley is an upper-division computer science course on Internet architecture and protocols, offered as a regular semester or a condensed 8-week summer session. Prerequisites in coding and computer systems are expected, but not strictly enforced. Enrollment ranges from the low to mid hundreds. The teaching team is typically one instructor (responsible for lectures and exams), a head TA (responsible for schedules and staff), and 6–8 additional TAs and tutors. Each week during the semester the class has two lectures, two or more TA-led discussion sections, and office hours. The course has three projects (traceroute, a routing algorithm, and TCP) and two exams.

2.3 Goals

The stated goal of the class is for students to learn how to reason through complex networking design problems. Information about how the Internet works is the framework for considering design tradeoffs in the class. With this goal in mind, we want to take a step beyond introducing students to existing architectures and protocols and encourage them to consider the implications of their choices when they design networked systems. We considered focusing on other verbs such as building, evaluating, measuring, maintaining, securing, and recovering and believe that each of these could be an interesting project (or even full module) in a computer systems class. Ultimately, we decided to focus on design since it aligns well with the class and because we believe it gives the students flexibility to be creative and consider alternative worlds and futures. In empowering students to design alternative networked systems, we also want to prompt them to consider the implications of their design, such as who (what demographics, companies, governments, etc.) might use their system and what the result (*e.g.* providing education access or enabling drone controls) of that use might be on society.

3 Past Experiences

We report on our experience running assignments for the networking course both before and after the rise of LLMs.

3.1 Case Study: The Transport Project

We have run a multi-week Transport project for over 10 years, that aims to give students a hands-on understanding of TCP's mechanisms. Students implement a user-space TCP socket against a Python network simulator, covering the three-way handshake, in-order and out-of-order receive,

	2018	2022	2026
Avg # multiple submissions*	4.4	4.0	3.3
Median first→final span (hours)*	15.1	2.2	1.4

Table 1: Submission behavior across three Transport offerings, spanning the pre-LLM (2018), early-GPT (2022), and coding-agent (2026) eras. *Multi-submitters only.

sending, advertised-window handling, active and passive close, retransmission, and RTO/RTT estimation.

Project setup. Our project is intentionally highly-scaffolded, in order to allow students to focus their time on the practical details of TCP rather than boilerplate implementation. We provide both skeleton code (excerpt shown in Fig. 1), and a detailed spec, totaling 21 pages across 9 numbered implementation stages, each in turn broken into numbered sub-steps. Many sub-steps include explicit hints of the form "use the raw `tx()` function" or "`RetxQueue` has a helpful method."

Grading. To scale to 100s of students, we provide a deterministic, public, and complete autograder. The final grade comes from 30 public unit tests (~2-6 tests per stage). Students can iterate against the autograder.

Challenges. We have observed a marked shift in students' submission behavior as LLM coding tools have become available (Table 1). Table 1 shows aggregate submission statistics across pre-LLM, early-LLM, and post-agent offerings of the course. The average number of submissions has dropped by 30%, with the fraction of students finishing on a single submission rising by 18%. Most strikingly, median first-to-final submission span among iterating students has fallen from over 15 hours to under 90 minutes. From conversations with students in office hours, we attribute these shifts to students delegating the project to LLM coding tools and completing it in a single short session.

We posit this is because the same artifacts that help students also help agents. The spec's enumeration of named methods and ordered hints, the localized fill-in markers in the skeleton, and the deterministic public test suite together reduce the project to a sequence of well-specified, well-localized fill-in the blank tasks, precisely the prompt structure modern coding agents thrive on.[1] In informal trials, we have observed a coding agent complete the full project end-to-end in a single session with no human guidance beyond the spec PDF and the skeleton repository.

3.2 Case Study: Research Synthesis Reading

In a recent offering, we piloted a purely design-oriented reading-response assignment. Students were asked to read Clark's seminal paper, "The Design Philosophy of the DARPA Internet Protocols" [3], then answer questions spanning factual recall on the paper's content, short-answer questions mapping Clark's internet design goals onto the Internet's architecture, and an open-ended design challenge to ponder how Internet architecture would need to change to operate as an interplanetary network between Earth, the Moon, and Mars, given low-bandwidth, high-latency, and intermittently-connected satellite links.

Students' experience with the assignment was mixed. Students generally did thoughtfully engage with the interplanetary design challenge and frequently cited specific design goals from Clark when arguing about which layers to modify. However, reading an academic research paper proved to be a significant barrier for many undergrads, and grading was hard to standardize and scale to many hundreds of students. These observations directly motivate the project structure we propose in §4, which preserves the design-tradeoff focus, but removes the research reading load.

```
def connect(self, ip, port):
    """Begins the TCP handshake by initializing
    the socket and sends a SYN to the peer."""
    assert self.state is CLOSED
    assert not self.is_bound

    self.snd = TXControlBlock()
    self.rcv = RXControlBlock()
    self.rcv.wnd = self.RX_DATA_MAX
    # ... device lookup and bind setup elided ...
    self.peer = IPAddr(ip), port
    self.bind(dev.ip_addr, 0)

    ## Start of Stage 1.1 ##
    # Write your code here
    ## End of Stage 1.1 ##
```

Figure 1: Stage 1.1 of the Transport project: the student fills three lines between the markers; the rest of `connect()` is given. The skeleton has 25 such blocks total, one per numbered sub-step of the spec.

4 Proposal

The goal of our project is to help instill systems design thinking, such as tradeoffs recognition and explicit decision making around these tradeoffs. We want to empower the students to have agency in making these decisions, and also hold them to justifying the decisions that they make. Because we are proposing replacing the third and final project, students have been introduced to all the layers of the internet including a few application layer protocols (DNS and HTTP). We expect this project to cover the span of 4 weeks, from Spring Break to the second to last week of class.

[1] Indeed, systems researchers report following exactly such a structure to enable full, end-to-end autonomous system synthesis [1, 2, 9, 10, 14].

4.1 Project Specifics

We propose an open-ended transport layer design project. In this section, we give details about our project.

Assignment. The assignment is to design a transport layer protocol for a defined workload in teams of 2-3 students. The protocol can take heavy inspiration from TCP, but ultimately will be judged based on performance against the workloads. Students will need to share why particular design decisions were made in a final report.

Workloads. We will provide three workloads for student teams to pick from. The three workloads that we provide intentionally vary across three dimensions: data volume (throughput), latency, and reliability.

Workload A: Remote Sensor Network

Description: Marine biologists deploy low-power, intermittent sensors to monitor water temperature.
Constraints: low-volume (100 bps), relaxed latency, and high-reliability.

Workload B: Data Center Backbone

Description: A hyperscaler is designing transport between two data centers. The network must be high-speed and respond to dynamic demand.
Constraints: extremely high volume, tail latency must be minimized, and high reliability requirement.

Workload C: Live Streaming Video

Description: A video conferencing company is designing a custom transport layer for streaming.
Constraints: extremely low-latency and high-volume, but packet level reliability is not critical

Student Presentation and Stages. We plan to present the project to students in three stages. We will provide an evaluation environment for students to test their code, but intentionally hide actual workload flows until after they submit an implementation.

The first stage, Stage 0, will retain the spirit of the paper reading assignment from previous semesters, with more scaffolding. In Stage 0, students will reflect on how goals and constraints shape protocol design, while reading select excerpts from "The Design Philosophy of the DARPA Internet Protocols" [3]. The core deliverable of this stage will be answering free response questions on Gradescope which prompt the students to consider the choice of workload as a commitment to a major design decision.

Stage 1 will include team formation and planning. We plan to let students form their own teams of 2-3, and will help facilitate this over our digital class forum. Teams will select one workload and scope their design in a design document. This design document will be the core deliverable for this stage along with a 10 minute discussion with a TA about their plan. These discussions can occur during dedicated discussion section time or in office hours.

In Stage 2, students will implement their protocols and workload traces based on their interpretation of our constraints, in our provided environment, with coding agents. We will provide guidance on programming with AI agents, but for the most part students will be free to explore what works for them. The deliverable for this stage is a code artifact submitted to Gradescope.

The final stage, Stage 3, will be focused on evaluation and reporting results. Students will evaluate their protocol by running it against our revealed workload and generating graphs to measure how well their protocol meets the goals they set out to meet. They will also have a chance in this stage to change their design based on the true workloads. The deliverable for this stage is a report which describes the protocol they implemented, the workload they picked, design tradeoffs, and results. If resources allow (sufficient TA hours available), this stage can also include an oral component where teams walk through and defend their results in conversation with a TA.

Grading. Students will be evaluated on two major aspects. The first aspect is how well their code performs on the workload they picked in terms of the key constraints. The second aspect is less defined, the report. We plan to generate a rubric which maps to the key components (*i.e.* protocol, workload, tradeoffs, and results) of the report.

5 Discussion

While we are excited about the project described in this proposal, there are still many open considerations. We discuss some below, and hope to engage further with the community's efforts on project design.

Equalizing access. As briefly mentioned in §1, uneven access to tools is a major concern. In our academic context, all students have access to Gemini which helps create a baseline, but some teams might want to use other models. To support this we are exploring if existing class funds could be used.

Rebuilding teamwork skills. Compared to our previous project, an at home individual coding task, this project emphasizes group work with required TA discussions. This approach has the added benefit of exercising collaboration and teamwork skills. We believe this is relevant not only for preparing students for post-grad careers, but also for building community and strong social ties among students.

Scaling. Another always present challenge of large classes is maintaining efficiency when it comes to grading and TA time. Our project adds a 10 minute discussion time with a TA which could add up quickly for a large class. For larger classes, we plan to pair groups with other groups who picked the same workload to discuss in peer groups as a TA walks around to check in with each cluster. TAs will undergo a short training session before these discussions.

Enabling pedagogical variety. We believe the pedagogically modular design of the project, *i.e.* the evaluation infrastructure (simulation), grading approach (oral reports and code), and problem properties (workloads) being separable components, will enable this project to adapt and evolve more easily over time. If solutions to previous semester workloads appear online, instructors can easily swap in new workloads with different design characteristics, without needing to re-write the entire project. Furthermore, the project could be adapted to integrate new workloads as new relevant technologies emerge.

Generalizing. We focus on redesigning one project as a starting point in this paper. From there, if successful, we believe the concept of decoupling implementation from design by leaning on AI coding tools could be expanded to many other open-ended projects across systems curriculum. In general, one way or another, networking and more broadly systems will have to adapt to this new world with advanced coding agents. We are hopeful that this project can be a small part of figuring out the path forward.

Acknowledgments

We thank our pedagogy-oriented colleagues at UC Berkeley for valuable discussions that shaped this work, including Lisa Yan, Scott Shenker, Rishabh Iyer, Dev Bali, and Jonah Bedouch. We also thank the anonymous reviewers for their insightful feedback.

References

[1] Shubham Agarwal, Alexander Krentsel, Shu Liu, Mert Cemri, Audrey Cheng, Rui Meng, Tomas Pfister, Chun-Liang Li, Sylvia Ratnasamy, Aditya Parameswaran, Matei Zaharia, Ion Stoica, and Mohsen Lesani. 2026. Inductive Deductive Synthesis: Enabling AI to Generate Formally Verified Systems. arXiv:2605.23109 [cs.AI] https://arxiv.org/abs/2605.23109

[2] Nicholas Carlini. 2026. Building a C compiler with a team of parallel Claudes. https://www.anthropic.com/engineering/building-c-compiler. Accessed: 2026-2-17.

[3] David D. Clark. 1988. The Design Philosophy of the DARPA Internet Protocols. In *Proceedings of the ACM Symposium on Communications Architectures and Protocols (SIGCOMM '88)*. ACM, New York, NY, USA, 106–114.

[4] Michael Docherty, Peter Sutton, Margot Brereton, and Simon Kaplan. 2001. An Innovative Design and Studio-based CS Degree. In *Proceedings of the 32nd ACM Technical Symposium on Computer Science Education (SIGCSE '01) (SIGCSE Bulletin, Vol. 33)*. ACM, 233–237.

[5] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2018. *How to Design Programs: An Introduction to Programming and Computing* (2nd ed.). MIT Press. https://htdp.org/

[6] GitHub. 2025. Octoverse 2025: A New Developer Joins GitHub Every Second as AI Leads TypeScript to #1. https://github.blog/news-insights/octoverse/octoverse-a-new-developer-joins-github-every-second-as-ai-leads-typescript-to-1/. Accessed: 2026-05-26.

[7] Christopher Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. 2010. Does studio-based instruction work in CS 1?: an empirical comparison with a traditional approach. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)* (Milwaukee, Wisconsin, USA). ACM, New York, NY, USA.

[8] Yasmin B. Kafai, David DeLiema, Deborah A. Fields, Gary Lewandowski, and Colleen Lewis. 2019. Rethinking Debugging as Productive Failure for CS Education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)* (Minneapolis, MN, USA). ACM, New York, NY, USA, 169–170.

[9] Keisuke Kamahori, Shihang Li, Simon Peter, and Baris Kasikci. 2026. VibeServe: Can AI Agents Build Bespoke LLM Serving Systems? arXiv:2605.06068 [cs.AI] https://arxiv.org/abs/2605.06068

[10] Shu Liu, Alexander Krentsel, Shubham Agarwal, Mert Cemri, Ziming Mao, Soujanya Ponnapalli, Alexandros G. Dimakis, Sylvia Ratnasamy, Matei Zaharia, Aditya Parameswaran, and Ion Stoica. 2026. The Time is Here for Just-in-Time Systems: Challenges and Opportunities. arXiv:2605.24096 [cs.DB] https://arxiv.org/abs/2605.24096

[11] Zachary A. Pardos and Shreya Bhandari. 2023. OATutor: An Open-source Adaptive Tutoring System and Curated Content Library for Learning Sciences Research. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM.

[12] Stack Overflow. 2025. 2025 Developer Survey: AI — Impacts of AI Agents. https://survey.stackoverflow.co/2025/ai#3-impacts-of-ai-agents. Accessed: 2026-05-26.

[13] Hussel Suriyaarachchi, Paul Denny, and Suranga Nanayakkara. 2025. Investigating the Use of Productive Failure as a Design Paradigm for Learning Introductory Python Programming. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025)*. ACM.

[14] Johannes Wehrstein, Timo Eckmann, Matthias Jasny, and Carsten Binnig. 2026. Bespoke OLAP: Synthesizing Workload-Specific One-size-fits-one Database Engines. arXiv:2603.02001 [cs.DB] https://arxiv.org/abs/2603.02001