# Trust, but Verify

## AI Interventions in Computer Networking Education

Tanya Shreedhar
TU Delft
The Netherlands
t.shreedhar@tudelft.nl

Christoph Lofi
TU Delft
The Netherlands
c.lofi@tudelft.nl

## Abstract

Generative AI assistants are now part of how undergraduates write and debug code, and computing education has moved from asking whether to allow them to asking how to teach with them. On networking tasks, those whose answers turn on the behavior of a running network the student can measure, these assistants are often fluent and wrong, and a beginner has no reliable way to tell. We call this failure mode *vibe networking*, accepting an AI artifact because it looks plausible, without checking whether it behaves as claimed. The networking course already holds the antidote, because its core method is measurement, the discipline's way of settling a claim against the system itself. We propose that the course give a generative model three roles across the term, shifting as the student's competence grows. At the start it is an adversary the student learns to catch by redoing its computation by hand, in the middle an assistant whose proposed experiments the student runs and measures on the live network, and at the end an examinee whose diagnosis the student confirms or refutes from a capture they take themselves. Measurement is the constant across the three roles: an observation the course produces in its normal run decides whether each AI answer holds, and the grade rewards the student's check, never the model itself. We work out the three roles and design one classroom intervention for each, sized for a 9-week BSc course of around 500 students at TU Delft.

## Keywords

networking education, generative AI, AI literacy, pedagogy, EU AI Act

## 1 Introduction

How do you teach a networking course to students who can get a fluent answer to almost any question from an AI assistant, yet have no reliable way to tell when that answer is wrong? Introductory programming has gone furthest on this question, moving from whether to allow these assistants to how to teach with them [9, 26]. The EU AI Act now requires universities to build a level of AI literacy in their students [11, 12], ACM and IEEE curricular guidance has begun to respond [1, 4], and a large literature on generative AI in education has grown alongside. That attention has split between courses on how AI systems are built and literacy frameworks on what AI is and how to use it day to day. The gap this paper takes up sits between them: how a core technical course should teach its students to use AI on the discipline's own work. It is both a pedagogical question and an open engineering one, since AI's use on networking tasks is itself unsettled. For computer networks the discipline-native check is measurement, and we are not aware of a published curriculum that builds on it.

The curriculum we propose gives the generative model a role that changes across the term as the student's competence grows. In the first weeks the student catches it as an adversary, by re-deriving the answer themselves; Overklift et al. open an introductory programming course with exactly this kind of catch, staged by the lecturer [22]. At the mid-term the student directs it as an assistant, asking it to propose experiments, then runs each one and checks the model's prediction against what the system does. In the end-of-term assessment the student judges it as an examinee, confirming or refuting its diagnosis from their own capture, and the grade attaches to the student's verification. Measurement makes each step safe: it exposes the adversary, adjudicates the assistant's predictions, and backs the student's judgment of the examinee.

Computer networks is a natural home for this proposal. Every lab in the standard layered syllabus produces an observation a claim can be checked against, from a packet trace to a throughput curve or a routing convergence [30]. When an AI artifact, a generated filter, a configuration, or a diagnosis, disagrees with the measurement, the measurement decides, and the student need not already know the answer to see that the artifact is wrong. A course built this way also meets the AI-literacy obligation on its own terms, since checking a claim against an observation is already what it teaches.

The failure mode the proposal guards against is what we call *vibe networking*, by analogy to Karpathy's "vibe coding":[1] a learner takes a fluent AI artifact at face value and never confirms it against the running system. The risk is sharp on networking tasks: Davie argues that command of a protocol's vocabulary is not command of its behavior [7]. The few available benchmarks point the same way, as the best of six models reaches only 79.3% on network topologies and falls off as they grow [10], while the errors of the more capable models are the ones a novice is least likely to catch [2]. Representative undergraduate prompts show the pattern (§A), such as a Wireshark display filter that over-matches the conversation it was meant to scope, or a BGP configuration that compiles cleanly yet leaves the network open to a route leak. Such answers are valid in form and wrong in behavior. A beginner has little to question them with, and a measurement exposes them directly. Better prompting does not close the gap, because specifying the intent in full is itself the hard part of the task [19].

**Contributions.** We contribute (i) the proposal that a computer networks course give AI three roles across the term, each kept honest by measurement; (ii) a detailed design of one classroom intervention per role, fitted to a 9-week BSc networking course of around 500 students at TU Delft, with no tooling beyond what the course has; and (iii) a set of illustrative probes showing where one

[1] A. Karpathy, post on X, Feb. 2025: https://x.com/karpathy/status/1886192184808149383.

current frontier model can fail on representative undergraduate prompts, layer by layer across the syllabus. §2 develops the proposal and works out the three roles, §3 locates the argument in the LLMs-on-networks and CS-education literatures, and §4 sketches a curriculum spine and places the proposal within the EU AI Act and a national public-value AI platform.[2]

## 2 Intervening AI in Three Roles

Our proposal turns on a division of labor between the AI and the instructor. AI works only within the course's empirical layer, the labs and measurements where a claim can be checked against the running network. The conceptual material, how each protocol works and why it is designed that way, stays with the instructor's lecture. A student may still quiz an AI tutor on lecture material at their own pace, and the design does not forbid that use. It intervenes where an AI answer can be wrong in ways a beginner cannot detect, and where a measurement exists to catch it. Foundational skill still comes from the lecture and from hand computation. This boundary keeps the tool from standing in for the mechanism the student is meant to learn. Within that empirical layer the role we give AI changes over the term, tracking the student's growing competence and the kind of task the course reaches, which runs from hand computation at the start to the diagnosis of a live network at the end (the model itself stays the same).

The standard computer networks course is built for this division of labor. Every protocol it teaches has a physical counterpart the model never saw, a frame on the wire or a routing event the student can measure. An AI artifact can therefore always be put to a test that no amount of well-worded description passes. Concretely, a student asks the model for a capture filter, runs it on the course trace, and compares the match count against the expected one. The filter is the model's artifact and the match count is the measurement that judges it. Our 9-week BSc course, "Introduction to Computer Networks", with around 500 students at TU Delft, has exactly this shape. Because the course is fundamental to CS education and is usually mandatory for the undergraduate cohort, the habit it builds percolates across other CS courses as well.

The labs already grade this kind of test, since their outputs are ones a machine can check and an LLM cannot easily fake: the match count above, or a configuration judged against route propagation in a virtual topology. A student whose filter misses the expected count sees at once that something is wrong. Catching a plausible mistake here is mechanical and asks for no judgment the student does not yet have. This is where Anwar et al. locate the danger of advanced models, since the fluent errors are the ones a novice is least able to catch [2]. The same habit runs through networking research, where a measurement decides live-Internet questions [3, 28], and it scales down to a teaching lab once the lab is read as a test of a claim.

How reliably a model answers depends on the kind of task the material sets. A student therefore meets confident wrong output in the normal flow of the course, with no contrived setup. We expect a frontier model to answer the mechanical, well-documented prompts correctly and to fail the ones that turn on observed behavior. The failure is rarely a refusal: asked about a system it cannot see, a model tends to answer for the textbook configuration, and to do so with confidence. Table 3 in the appendix sets out six such prompts spanning the syllabus, each one a student can put to a model, with the outcome we expect.

The course can therefore give AI a different role at each stage. At the *link* and *network-addressing* layers the material is deterministic algorithms the student can re-derive by hand, such as a cyclic-redundancy-check (CRC) remainder or a subnet mask. Here the algorithm itself exposes a wrong answer, with no tool and no diagnostic vocabulary, so AI's role is the one the student learns to catch. The *transport* layer is different: its behavior is valid in form but contingent on configuration, how a `netem`-shaped link or a congestion-control variant like CUBIC or BBR performs under load. The model is quick to propose such variations, yet only a measurement says which prediction holds, and its natural role becomes generating the experiments the student then tests. Routing at the network layer sets a diagnosis task on a topology the model never saw, an OSPF reconvergence after a link failure or a BGP path the student captures. On these the model is often right on the textbook case and wrong on the case typical in practice, and its role shifts again, to a diagnosis the student must judge against their own measurement. Each stage hands the AI's claim to a measurement the course was going to run anyway, and the outcome of that measurement grades the exchange.

The nearest alternative to our proposal places the remedy in understanding rather than measurement: Davie would teach students enough networking to recognize when a model's answer is nonsense [7]. We share that aim, and read measurement as how networking already performs that recognition. Judging an answer from understanding alone presumes the understanding a beginner has yet to acquire. A lab that produces an observation the answer must match supplies the check from the outside, and a student who cannot yet explain why an answer is wrong can still watch it fail. One might expect this design to weaken as models improve and hallucinate less. We argue that it does not. What the middle and final stages test is the gap between the textbook the model correctly recites and the system the student runs. A browser may carry HTTPS over QUIC rather than the TCP a textbook describes, and only a measurement reveals the textbook-correct answer to be wrong about the deployed protocol. Even a model that never hallucinated would meet this gap, because translating an intent into a working configuration is underdetermined by the specification itself [19]. The remedy is therefore a pedagogy that checks the specification against the system as deployed.

We place one intervention in each role, following the student from the start of the term to its end: a demonstration the student disproves by hand (§2.1), a lab in which AI proposes experiments the student measures (§2.2), and an assessment graded on the student's own capture (§2.3). Each intervention climbs a Bloom level [13] on the course's bottom-up syllabus [30] and fits one 90-minute slot. None needs infrastructure beyond what a standard networks course runs, which is what makes the designs reproducible in any large BSc offering.

[2] eduGenAI, the Npuls/SURF generative-AI platform for Dutch education: https://npuls.nl/en/edugenai.

**Intervention 1: CRC remainder (link layer)**

**PROMPT**

```
Compute the CRC remainder for message
D = 1101011011 with generator G = 10011,
and give the transmitted frame.
```

**OUTPUT** *confident, wrong*

```
Remainder = 1011.
Transmitted frame = 1101011011 1011.
```

**CHECK** *long division mod 2, on the board*

```
D*2^4 / G = 10011  ->  R = 1110  (not 1011)
frame / G = 10011  ->  0  (remainder zero: valid)
```

**Figure 1: Intervention 1. The student catches the model's confident wrong answer with the long division the lecture already teaches, needing no tool beyond the algorithm itself.**

## 2.1 Intervention 1: AI as an Adversary

The first intervention runs at the start of term and gives AI the role suited to a beginner, a confident answer the student disproves by hand. The earliest algorithm a student can re-derive unaided on the bottom-up syllabus is CRC error detection, so the demonstration runs in that lecture. The lecturer poses a CRC instance to a frontier LLM live in class and asks for the remainder and the transmitted frame. On an instance large enough to need several rounds of polynomial division, the model often returns a confident but wrong remainder. The long division on the board returns a different remainder, and the frame built from it divides to zero while the model's does not (Figure 1). Which instances trip which model shifts with every release, and a model that executes code while it answers may divide correctly. The demonstration needs one confidently wrong answer, not a reliably failing model, and the lecturer can rehearse candidate prompts before class.

What the demonstration teaches is the habit of checking. The particular error the model made is incidental. A convincing wrong answer is the kind a novice is least able to question [2], and re-deriving the result is how a beginner exposes one. The later material on framing, error correction, and medium access then arrives with a reason to learn it. We expect, though we cannot yet show, that students leave the lecture treating LLM output as a hypothesis in need of a check. Each layer adds a deterministic computation the student can re-derive: a CIDR route aggregation at the network layer, sequence-number arithmetic at the transport layer, and a Dijkstra or distance-vector run over a given topology. These are settled by hand, before the student has met the live measurements the later interventions turn on, and the demonstration can therefore recur in later lectures and assignments well past day one. Past the hand-derived cases, the same role extends to any artifact a single course-standard command can check against a known result. Each probe in the appendix pairs a prompt with an observation that surfaces its failure (Table 3). P1 is the transport-layer case: a Wireshark display filter the model writes confidently, exposed by a match count against a known trace.

| Baseline lab | What-if knob | The check |
|---|---|---|
| `tc netem` shaping | loss, delay, MTU | ping, `iperf`, `iRTT` |
| TCP congestion control | Reno/CUBIC/BBR | throughput, RTT |
| Competing TCP flows | differing RTT | per-flow throughput |
| Queue management | FIFO vs `fq_codel` | ping under `iperf` |
| Path MTU, IPv4 vs IPv6 | MTU, DF bit | capture, `ping` |
| NAT translation | inbound vs outbound | `conntrack` |
| DNS resolution | resolver, TTL | `dig` over time |
| Intradomain routing | a link cost | `traceroute` |

**Table 1: Candidate hosts for the what-if intervention. In each, the student asks the model to predict the effect of a change, then measures it.**

## 2.2 Intervention 2: AI as an Assistant

By the mid-term the student can do more than distrust the model, and the second intervention gives them a role to direct. Students run a baseline lab and then ask a frontier LLM to propose what-if variations on it, one for each parameter the lab exposes. The model can offer more of these than a single session would reach, each with its own prediction. The student runs each variation on the lab network and compares the model's prediction with what the interface reports. Where the measurement matches the prediction the student gains a mechanism they can now explain. A divergence is more interesting: it marks a gap between the protocol the model reasoned about in the abstract and the system as the lab configured it.

The traffic shaping and management lab is the worked example. A student adds 100 ms of delay and 1% loss to an interface with `tc netem`. They then ask the model what happens if the loss rises to 5%, if the link's MTU is halved, or if the round-trip time doubles. Each variation is measured with a ping and a short `iperf` run. We expect the model to write a correct `netem` command but to misread its effect, predicting that the round-trip time rises by about 200 ms because the delay applies to both directions (P5, Table 3). An egress qdisc shapes outbound traffic only, so a single ping shows the round trip rising by about 100 ms. The student therefore checks the model's reasoning, not only its syntax. The student also stays in a generative mode throughout, predicting and then testing, which is where active learning does its work [6]. The risk is that an assistant a novice trusts can carry them past their own understanding [15], and it is the least confident novices who reach for the assistant earliest and most [17]. The grading is what guards against this: the object graded is the student's measured result, never the model's prediction. Table 1 sketches further labs the same move fits, each pairing a knob the student turns with the observation that checks the model's prediction.

## 2.3 Intervention 3: AI as an Examinee

The third intervention runs at the assessment stage, late in the course. It gives AI the role it will most often play in practice, a diagnosis that is usually plausible and must still be checked. The examinee here is the model and the examiner is the student: the instructor grades the student on the quality of that examination, and the AI grades no one. By this point a frontier model is often

| Artifact | Model's diagnosis | Student's check |
|---|---|---|
| OSPF triangle, link down | ≈40 s convergence | re-run, watch the event |
| BGP stub configuration | follows the prompt | route-leak in a topology |
| Slow TCP capture | blames congestion | zero-window stalls in the trace |
| Load-balanced path | a routing loop | re-measure, Paris mode |

**Table 2: Candidate diagnoses for the assessment intervention. The student grades the model's verdict against their own measurement.**

right. The canonical traceroute reading (P4, Table 3) is one we expect it to answer correctly and clearly. A plausible answer is tempting to accept on sight, but a student who has met the first two interventions arrives expecting to check it. Students receive a packet capture or a small topology together with an LLM's diagnosis of it. They either confirm or refute that diagnosis from the capture itself and from a re-measurement in the lab network. The graded artifact is the measurement-backed defense, which an LLM cannot supply because it never saw the student's capture.

Where the model still errs, the error tends to sit in the case an operator meets in practice. The OSPF triangle is the worked case (P6, Table 3). The student is given a three-router topology and the model's answer to one question, how long the network takes to reconverge after a single link fails. A frontier model typically answers with the dead interval, about 40 seconds, the time OSPF waits through four missed 10-second hello packets before it declares a silent neighbor down. That is the right figure for a neighbor that has gone quiet, but not for a link that has failed. A failed link drops carrier, which signals the interface down at once. The adjacency therefore tears down without waiting for the timer, and the network reconverges in a small fraction of that time. The student who reruns the topology and fails the link watches the adjacency drop and measures the reconvergence directly, well inside the 40 seconds the model gave. The examinee role recurs at the inter-domain layer too. We expect a frontier model to produce a BGP stub that follows the prompt cleanly yet leaks routes in a small topology (P3, Table 3). Table 2 lists further diagnoses a course could set, each a case where the model is plausible and requires checking.

## 3 Related Work

The CS-education GenAI corpus [5, 8, 9, 17, 26] is mature. Its findings transfer only partly, because networking has distributed state, layered abstraction, and measurement as a first-class method. Networking-specific LLM evaluation is still small but growing. Anwar et al. [2] and Donadel et al. [10] together provide the two multi-model benchmarks the field has, and Protogeros et al. [27] outline and prototype a continual benchmarking suite for LLM-driven incident management on the operations side. All three evaluate the LLM rather than the student. The discipline's own pedagogy forum has returned to the question in two multi-voice panels fifteen months apart.[3] The operator-facing discourse converges from practice: Pepelnjak uses "vibe coding" verbatim in a netlab context [24], characterizes mid-2025 frontier models as stuck at the "sloppy bullshitting overconfident intern" stage on a Cisco IS-IS task [23], and warns that LLMs are dangerous for beginners who lack the experience to tell when the LLM is hallucinating [25]. In a non-CS-education engineering journal, Stanojević et al. [29] pilot an LLM-backed chatbot for an undergraduate networks course, evidence that the question is being addressed across venue traditions. Guarded AI-tutor design is best represented by CS50.ai [15], whose designers name "instruction dilution," a long guardrail prompt that stops being followed, as a core failure mode [16]. We have found no networking analogue in the published literature. ACM curricular guidance offers a discipline-specific practices volume for introductory programming [4] on top of the CS2023 baseline [1], but no equivalent treatment of computer networks. McDonald et al. [18] analyze the policies of 116 US R1 universities, documenting the swing from regulation toward integration and noting that STEM and code activities are under-served relative to writing, while Lau et al. [14] find programming instructors across nine countries split between banning the tools and integrating them. The EU regulatory context [11, 12, 21] and the eduGenAI platform have not, to our knowledge, been combined with an AI-literacy framework built into an undergraduate computer networks course.

## 4 Discussion and Conclusion

The proposal we have argued for stays within one course and rests on a general mechanism. The networking course can teach its students to work with generative AI by holding every AI artifact to the measurements its labs grade anyway, with no separate compliance module and no blanket policy. The observation that grades the lab doubles as the check on the model, so a student who cannot yet judge an AI answer from understanding alone can run the measurement and find out. The move carries to any course whose research practice is built on measuring a live system.

The EU AI Act's Article 4 [12] obliges every deployer of an AI system, universities included, to ensure "a sufficient level of AI literacy," which Article 3(56) defines as the understanding needed to make informed use of AI and to grasp its risks. The European Commission's guidance sets minimum factors for such a program: a general understanding of AI, the organization's role, the risk profile of the systems used, and calibration to the audience [11]. The Commission's November 2025 Digital Omnibus proposal would soften this duty, shifting it from a binding obligation on deployers toward a duty on the Commission and Member States to promote AI literacy.[4] Whichever way the Digital Omnibus resolves the duty, the pedagogical case stands.

No guidance yet tells the course how to deliver this obligation in terms of an artifact a student can run. The Commission's Repository of practices[5] collects organization-level programs, and general frameworks such as Npuls's AI-GO! [20] scaffold staff and students without entering any one discipline. ACM's discipline-specific guidance stops at introductory programming (§3), and national accreditation guidance explicitly defers the course-level question [21]. A

[3] The Networking Channel: "LLMs: Teaching and Learning (Networking)", Sep. 2024, https://networkingchannel.eu/llms-teaching-and-learning-networking-speakers/; and "Teaching Networking in the Age of LLMs: Challenges, Opportunities, and the Road Ahead", Dec. 2025, https://www.youtube.com/watch?v=glxlE8inLRA.

[4] Bird & Bird, "Digital Omnibus on AI: Trilogue Stalls Ahead of the AI Act Deadline", May 2026: https://www.twobirds.com/en/insights/2026/germany/digital-omnibus-on-ai-trilogue-stalls-ahead-of-the-ai-act-deadline.

[5] https://digital-strategy.ec.europa.eu/en/policies/repository-ai-literacy-practices

networking course that asks students to check every AI-generated artifact against an observation meets all four factors as a by-product of its ordinary teaching, and the interventions in §2 give that answer its concrete form.

How generative AI is incorporated across the CS curriculum as a whole, beyond any one course, is a pressing problem in its own right. One shape a curriculum-wide answer could take is a spine that builds AI literacy through the discipline rather than alongside it. The networking course could serve as the year-one encounter with AI as an object of inspection, with interventions following the decline-and-guide guardrails of CS50.ai [15] and the start-middle-end sequence of interventions reported by Overklift et al. [22]. Concurrent and later courses on programmability, operating systems, and similar topics could treat AI assistance as a default tool, with disclosure rubrics and measurement-validated grading. A final-year course could have students run small AI-assisted studies of their own, replicating the discipline's research practice [3, 28] across the cohort.

Where such a spine runs is a further open question. A national public-value AI platform such as Npuls and SURF's eduGenAI could give it data sovereignty and one uniform place to run the AI. The AI a course teaches students to check is best run on infrastructure the institution governs. A third-party service can change model version and content policy between assignments. A shared platform also equalizes access, so participation does not turn on a personal subscription or prior familiarity with a given tool. Because the graded object is always the student's measurement, a student new to these tools is not disadvantaged.

What this paper offers is a design, and the evaluation remains to be done. A fuller study would run several models, including an open-weight one, with independent expert grading across the syllabus. It would also track whether students who meet the interventions verify AI output more often, catch more of the plausible errors, and transfer the habit to networks they diagnose without AI. It would ask whether those outcomes vary with prior experience. Such a study is more than one course can carry. We bring the design to this workshop to find collaborators: educators who run large core networking courses and whose cohorts can put the three roles to the test.

## References


[1] ACM/IEEE-CS/AAAI Joint Task Force on Computer Science Curricula. 2024. *Computer Science Curricula 2023 (CS2023).* ACM, IEEE Computer Society, and AAAI. doi:10.1145/3664191

[2] Mubashir Anwar and Matthew Caesar. 2024. Understanding Misunderstandings: Evaluating LLMs on Networking Questions. *ACM SIGCOMM Computer Communication Review* 54, 4 (Oct. 2024), 14–24. doi:10.1145/3717512.3717515 Selected for Best of CCR at SIGCOMM 2025.

[3] Florian Aschenbrenner, Tanya Shreedhar, Oliver Gasser, Nitinder Mohan, and Jörg Ott. 2021. From Single Lane to Highways: Analyzing the Adoption of Multipath TCP in the Internet. In *Proc. IFIP Networking Conference*. 1–9. doi:10.23919/IFIPNetworking52078.2021.9472785

[4] Brett A. Becker, Michelle Craig, Paul Denny, Hieke Keuning, Natalie Kiesler, Juho Leinonen, Andrew Luxton-Reilly, Lauri Malmi, James Prather, and Keith Quille. 2023. Generative AI in Introductory Programming. ACM CS2023 Curricular Practices Volume. https://csed.acm.org/large-language-models-in-introductory-programming/

[5] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard—Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proc. 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE).* 500–506. doi:10.1145/3545945.3569759

[6] Michelene T. H. Chi and Ruth Wylie. 2014. The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes. *Educational Psychologist* 49, 4 (2014), 219–243. doi:10.1080/00461520.2014.965823

[7] Bruce Davie. 2024. Can LLMs Be Used in Networking Education? Systems Approach newsletter. https://systemsapproach.org/2024/09/16/can-llms-be-used-in-networking-education/

[8] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Proc. 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE).* 296–302. doi:10.1145/3626252.3630909

[9] Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N. Reeves, Eddie Antonio Santos, and Sami Sarsa. 2024. Computing Education in the Era of Generative AI. *Commun. ACM* 67, 2 (2024), 56–67. doi:10.1145/3624720

[10] Denis Donadel, Francesco Marchiori, Luca Pajola, and Mauro Conti. 2024. Can LLMs Understand Computer Networks? Towards a Virtual System Administrator. arXiv:2404.12689 [cs.NI] https://arxiv.org/abs/2404.12689

[11] European Commission. 2025. AI Literacy — Questions & Answers. Digital Strategy webpage. https://digital-strategy.ec.europa.eu/en/faqs/ai-literacy-questions-answers

[12] European Parliament and Council. 2024. Regulation (EU) 2024/1689 (Artificial Intelligence Act). Official Journal of the European Union. https://eur-lex.europa.eu/eli/reg/2024/1689/oj

[13] David R. Krathwohl. 2002. A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice* 41, 4 (2002), 212–218. doi:10.1207/s15430421tip4104_2

[14] Sam Lau and Philip J. Guo. 2023. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *Proc. 2023 ACM Conf. on International Computing Education Research V. 1 (ICER).* 106–121. doi:10.1145/3568813.3600138

[15] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J. Malan. 2024. Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In *Proc. 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE).* 750–756. doi:10.1145/3626252.3630938

[16] Rongxin Liu, Julianna Zhao, Benjamin Xu, Christopher Perez, Yuliia Zhukovets, and David J. Malan. 2025. Improving AI in CS50: Leveraging Human Feedback for Better Learning. In *Proc. 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE).* 715–721. doi:10.1145/3641554.3701945

[17] Lauren E. Margulieux, James Prather, Brent N. Reeves, Brett A. Becker, Gozde Cetin Uzun, Dastyni Loksa, Juho Leinonen, and Paul Denny. 2024. Self-Regulation, Self-Efficacy, and Fear of Failure Interactions with How Novices Use LLMs to Solve Programming Problems. In *Proc. 2024 Conf. on Innovation and Technology in Computer Science Education V. 1 (ITiCSE).* 276–282. doi:10.1145/3649217.3653621

[18] Nora McDonald, Aditya Johri, Areej Ali, and Aayushi Hingle. 2024. Generative Artificial Intelligence in Higher Education: Evidence from an Analysis of Institutional Policies and Guidelines. arXiv:2402.01659 [cs.CY] https://arxiv.org/abs/2402.01659

[19] Rajdeep Mondal, Nikolaj Bjorner, Todd Millstein, Alan Tang, and George Varghese. 2025. Tackling Ambiguity in User Intent for LLM-Based Network Configuration Synthesis. In *Proc. 24th ACM Workshop on Hot Topics in Networks (HotNets).* 176–183. doi:10.1145/3772356.3772402

[20] Npuls. 2025. AI-GO! A Framework for AI Literacy in Education. https://npuls.nl/_assets/275862bd-10b1-4ab0-ba9c-e7b21cff5502/AI-GO%21%20Framework%20for%20AI%20literacy%20in%20education.pdf

[21] NVAO. 2026. Initial Exploration of Generative Artificial Intelligence. Accreditation Organisation. https://www.nvao.net/files/attachments/.14069/Initial_Exploration_of_Generative_Artificial_Intelligence.pdf

[22] Thomas Overklift and Andy Zaidman. 2026. Pragmatic Use of Generative AI in an Introductory Programming Course. In *Companion Proc. ACM Int. Conf. on the Foundations of Software Engineering (FSE Companion '26).* To appear. Preprint: https://azaidman.github.io/publications/overkliftFSE-SEET2026.pdf.

[23] Ivan Pepelnjak. 2025. ChatGPT Strikes Again: IS-IS on Unnumbered Interfaces. ipspace.net blog. https://blog.ipspace.net/2025/05/chatgpt-isis-unnumbered/

[24] Ivan Pepelnjak. 2025. Vibe Coding netlab Lab Topology with ChatGPT. ipspace.net blog. https://blog.ipspace.net/2025/05/chatgpt-netlab-topology/

[25] Ivan Pepelnjak. 2026. Opinion: Impact of AI on Networking Engineers. ipspace.net blog. https://blog.ipspace.net/2026/01/ai-impact-networking-engineers/

[26] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proc. 2023 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR).* 108–159. doi:10.1145/3623762.3633499

**Probe P1: Wireshark display filter (transport)**

PROMPT

```
Show only TCP SYNs to destination port 443, plus
any TCP retransmissions in the same conversation.
```

OUTPUT

```
(tcp.flags.syn==1 && tcp.flags.ack==0
   && tcp.dstport==443)
  || tcp.analysis.retransmission
```

**Probe P5: `tc netem` shaping (link layer)**

PROMPT

```
Write a Linux tc command to add 100 ms of one-way
delay and 1% packet loss on outgoing traffic from
eth0. Then explain how to verify it.
```

OUTPUT *command right, gloss wrong*

```
sudo tc qdisc add dev eth0 root netem \
     delay 100ms loss 1%   ← runs as written
Verify with ping: the RTT should increase by
about 200 ms, because the delay applies to
both the outbound and the return path.
```

**Figure 2: In P1 the second clause matches retransmissions in every conversation, not only the target one, so a match count against a known trace exceeds the expected one. In P5 the command runs as written, but the verification gloss is wrong: an egress qdisc shapes outbound traffic only, so a round-trip ping rises by about 100 ms, not 200 ms.**

| Probe | Task (layer) | Verdict | The check |
|---|---|---|---|
| P1† | Wireshark filter (transport) | WRONG | match count |
| P2 | subnet allocation (network) | CORRECT | n/a |
| P3† | BGP stub (inter-domain) | WRONG | route leak |
| P4 | traceroute reading (network) | CORRECT | n/a |
| P5† | `tc netem` shaping (link) | WRONG | one ping |
| P6 | OSPF convergence (routing) | MISLEADING | link-down event |

**Table 3: Six probes across the syllabus, each a prompt a student can put to a frontier model. *The check* is the course-standard observation that reveals each expected failure in one command or one capture. Verdicts: correct, misleading (correct-but-misleading), wrong (plausible-but-wrong). † marks the errors a novice would not catch.**


[27] Ioannis Protogeros and Laurent Vanbever. 2025. Continual Benchmarking of LLM-Based Systems on Networking Operations. In *Proc. ACM SIGCOMM 2025 Posters and Demos*. 70–72. doi:10.1145/3744969.3748425

[28] Tanya Shreedhar, Rohit Panda, Sergey Podanev, and Vaibhav Bajpai. 2022. Evaluating QUIC Performance Over Web, Cloud Storage, and Video Workloads. *IEEE Transactions on Network and Service Management* 19, 2 (2022), 1366–1381. doi:10.1109/TNSM.2021.3134562

[29] Jelica Stanojević, Ivan Milenković, Miroslav Minović, and Milica Maričić. 2026. Educational Platform in Higher Education With LLM-Driven Chatbot for Computer Networks and Telecommunications Course. *Computer Applications in Engineering Education* 34, 1 (2026). doi:10.1002/cae.70105

[30] Andrew S. Tanenbaum, Nick Feamster, and David J. Wetherall. 2021. *Computer Networks* (6 ed.). Pearson.


## A Six Probes Across the Syllabus

These six prompts span the course's bottom-up syllabus [30]. We drew them to illustrate the failure mode, without any claim to a measured study. Mapping such cases across the syllabus is itself groundwork a measurement-first pedagogy needs, and we expect the set to grow as educators run their own variants. Table 3 maps each prompt to the outcome we expect and to the check that surfaces a failure. We expect a model to answer two of the six correctly, the subnet arithmetic and the canonical traceroute diagnosis (P2, P4), both at the model-friendly end of the syllabus. Of the other four, three fail in a way a novice would not catch (P1, P3, P5): a display filter that over-matches its conversation scope, a BGP stub that follows the prompt past the point of safety, and a shaping gloss that miscounts the delayed direction. They share the pattern the interventions turn on, valid surface form paired with wrong operational behavior that one command exposes. Figure 2 reproduces the two tightest cases, P1 and P5, where a match count or one ping is enough to expose the failure.


## Acknowledgments

GenAI was used for grammar checking.