Flipped Team-Based Learning and Labs for a Course on Computer Networking

Thomas Zinner
thomas.zinner@ntnu.no
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway

Katrien De Moor
katrien.demoor@ntnu.no
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway

Stanislav Lange
stanislav.lange@ntnu.no
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway

David Palma
david.palma@takinobori.com
Takinobori
Coimbra, Portugal

Trond Vatten
trond.vatten@ntnu.no
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway

Abstract

Introductory computer networking courses must help students move beyond memorizing protocols toward reasoning about packet-level behavior, configuration, and debugging. This experience report presents a flipped Team-Based Learning design that integrates weekly preparation, readiness assurance tests, team activities, and reproducible hands-on labs in a 13-week undergraduate networking course. We describe the course workflow, assessment model, and support structure, including an oral exam for individual accountability. Drawing on student feedback and teaching observations, we identify perceived strengths, operational trade-offs, and lessons for scaling active, lab-centered networking education in mid-sized cohorts.

Keywords

networking education, team-based learning, flipped classroom, open educational resources

1 Introduction

Computer networking is a foundational topic in many computing curricula, yet it remains challenging to teach it effectively: students must connect layered abstractions to operational reality, practice tool-based skills, and develop debugging intuition. These challenges have motivated work on tailored pedagogical approaches and infrastructures that increase engagement and support learning. Examples include flipped classrooms, hands-on labs and testbeds, assessment innovations, and open educational resources.

In this context, a major strength of the field is the availability of openly reusable materials. For example, Kurose and Ross provide extensive instructor-facing slide decks explicitly released for broad reuse and adaptation [2], and Bonaventure et al. have developed an open-source eBook and a set of open educational resources for both introductory and advanced networking courses, including an automated grading platform that helps scale hands-on assignments to large cohorts with limited staff [7]. In parallel, a range of freely available lab platforms enable experiential learning, from lightweight emulation (e.g., Mininet [1]) and per-team isolated environments (e.g., iLab@Home [18]) to larger course-wide operational environments such as ETH Zurich's containerized mini-Internet project [14].

However, while these resources lower important barriers for educators, they do not make course-design "plug-and-play". Much of the difficult work lies in the integration between resources: e.g., deciding what students should prepare before class, how instructors can detect and respond to misconceptions, how class time should connect conceptual discussions to later lab work, how practical environments can be kept reliable, and how meaningful feedback can be provided without overburdening the teaching staff. Without such integration, students may experience preparatory readings, class activities, and labs as disconnected tasks, while educators face high coordination and support costs.

The goal of this paper is to make this integration problem concrete and to share transferable experiences. We report on the design and operation of an introductory networking course in the second semester of a five-year integrated master's program in Cyber Security and Data Communication. The course uses a Team-Based Learning (TBL) approach [4] to combine preparation, in-class activity, and labs into a single recurring weekly cycle, rather than leaving them as disconnected tasks.

The course follows the topic arc of Kurose and Ross [2], runs for 13 weeks, and typically serves 50–70 students. Our design combines three main elements: First, we use TBL to structure weekly preparation and in-class activity. Students

complete individual readiness checks before discussing related questions in teams. Second, the class time is organized around team discussion, application-oriented activities, and targeted clarification of concepts instead of repeated presentation of all the preparation material. Third, compulsory labs, formative checkpoints, and periodic deliverables connect students' conceptual understanding with the practical configuration, observation, and debugging of running networks. The intended result is a rhythm in which students prepare before class, expose and resolve misconceptions during class, and apply concepts in labs after class.

We evaluate the design as an experience report, drawing on course surveys, reference-group dialogue reports, and educator and teaching-assistant observations. We use these sources to identify student perceptions, tradeoffs, and lessons for educators who want to combine active learning with practical networking labs under realistic staffing constraints.

This paper makes four contributions:

(1) An integrated course workflow that aligns preparation, in-class, and lab elements into a single reusable 13-week cycle, rather than a set of separately adopted resources.
(2) Networking-specific active-learning materials: We describe the design of readiness-assurance questions and team activities that target networking-specific reasoning, including protocol layering, packet-level behavior, configuration decisions, and common misconceptions.
(3) A reproducible lab and feedback model: We present a lab model based on per-team emulation environments, notebook-based formative checkpoints, and periodic integrative deliverables, designed to support operational practice while keeping feedback demands manageable.
(4) Operational lessons and reusable artifacts[1]: We report lessons from running the course with cohorts of 50–70 students and release the supporting materials, including iRAT and tRAT sources, self-designed team activities, lab specifications, VM build scripts, topology files, and notebook checkpoints.

Section 2 introduces the course context and goals, Sections 3 and 4 present the teaching methods and assessment design, Section 5 summarizes student feedback, and Section 6 concludes.

2 Course Context and Learning Goals

This section summarizes the course context in Section 2.1, learning goals in Section 2.2, and the GenAI policy in Section 2.3.

[1]https://git.ntnu.no/ie-iik/computer-networks, archived at https://doi.org/10.5281/zenodo.21218218

2.1 Course context and constraints

The course is an undergraduate "Computer Networks" course, offered in the second semester of a five-year program in communication technology and cyber security. As the first networking principles-course in the program, it lays the groundwork for the subsequent curriculum in networking and distributed systems. It is a 7.5 European Credit Transfer and Accumulation System (ECTS) course taught in English, which runs over a 13-week semester [17]. The typical cohort size is 50–70 students, motivating a design that supports active learning and hands-on practice.

Four further conditions shape the design. Firstly, students enter this early course with heterogeneous backgrounds in programming and operating systems. Secondly, the lab work is a major graded component—half of the portfolio grade as described in Section 4—which motivates students to work on the labs in a timely manner, use the formative feedback channels, and clear the checkpoints before submitting the reports for grading. Thirdly, the course is staffed by four student teaching assistants (TA) and a PhD teaching assistant; each lab group keeps the same student TA mentor throughout the semester—rather than pooling assistants across sessions—to provide every group with a stable and safe environment in which to work and ask questions. The way this staffing budget is allocated is described in Section 3.5. Fourthly, the in-class sessions take place in an active-learning room with group tables rather than a tiered lecture hall: teams sit together throughout, which makes the weekly tRAT discussions and application activities frictionless and lets the instructor circulate between groups.

Throughout the semester, the course is accompanied by a *reference group*: a small group of students drawn from the cohort who meet with the teaching staff at least three times during the term and summarize the cohort's feedback in a written report. This provides a continuous, formal feedback channel between students and instructors during the semester, rather than only an end-of-term evaluation.

2.2 Learning goals and topic coverage

The course aims to provide students with (i) foundational knowledge of layered architectures, services, and protocols used in computer networks, (ii) practical skills in configuring and operating networks and common Internet services, and (iii) competence to collaborate responsibly and effectively in teams [17]. The formal learning outcomes explicitly include core reference models and protocols across the stack (from physical/link to application), practical deployment and use of widely used Internet services, and fundamentals of network and communication security.

The topic arc follows a standard networking progression aligned with widely used introductory materials such as

Kurose and Ross: introduction and "big picture", application-layer protocols, transport-layer principles, network-layer data and control plane concepts, link-layer/LAN technologies, and network security [2]. In our offering, security is treated as a dedicated thread late in the semester (two weeks), including basic cryptographic concepts followed by applied mechanisms and configuration tasks (e.g., TLS, VPNs, and firewalling) in the context of the earlier layers.

2.3 Generative AI policy

Our institution provides a general policy on the use of generative AI. It is, however, too generic to answer the practical question students face in this course: what may AI be used for, and what must remain their own work? This question is particularly acute here because the learning outcomes go beyond producing correct written answers: students prepare individually, reason with peers, configure and debug running network services, and must explain their own operational choices.

We therefore introduced a course-level "AI contract" with one core principle: *AI may be used as a coach to support learning, but not as a shortcut to produce work the student cannot explain and defend* [10]. Rather than relying on detection or disclosure rules, the course design itself makes shortcuts unattractive: the goal is to make the *assessment* independent of whether and how students use generative AI, without restricting its role in *learning*. Readiness tests are written on paper, lab checkpoints verify behavior in the team's actual environment, and the oral examination requires each student to defend their understanding without AI access. Section 3 describes how this plays out in the individual course activities.

3 Teaching Methods

We now describe the different methods and pedagogical elements used in the implementation of the course, and to achieve the learning outcomes, cf. Section 2.2.

3.1 Weekly workflow

The weekly design follows Biggs' notion of *constructive alignment*: learning outcomes, learning activities, and assessment are aligned so that what students *do* to succeed is the same as what the course intends them to learn [6]. In an introductory networking course this is a deliberate design choice rather than a slogan, because networking behavior is context-dependent and plausible-sounding answers may be wrong without the specific environment and traces to verify them against. The course therefore structures its weekly readiness assurance, in-class team discussion, and labs to reward evidence-based reasoning and debugging practice rather than recall alone.

Concretely, the course follows a flipped structure: students prepare before class using curated materials, and class time is used for readiness assurance, structured discussion, and targeted clarification. The in-class sequence instantiates the standard TBL cycle of (1) preparation, (2) individual readiness assurance (iRAT), (3) team readiness assurance (tRAT) with immediate feedback, and (4) application-oriented activities [4].

Figure 1 shows how these elements recur across the 13-week term: each week pairs preparation and in-class readiness assurance with hands-on lab work on the same topic, the topic progression follows the top-down arc introduced in Section 2.2, and the individual oral examination at the end of the term assesses the same reasoning that the weekly activities practice. The weekly workflow goes as follows:

- **Before the 1st lecture (preparation):** Students complete assigned reading and preparatory prompts aligned with the week's topics.
- **1st session (readiness assurance + team activities):** Students complete an iRAT individually and hand it in, then a tRAT in small discussion groups with immediate feedback (scratch-card style), followed by an appeal opportunity and structured team application activities (Section 3.3). These weekly readiness checks are intended to level out differences due to students' heterogeneous prior knowledge and backgrounds.
- **2nd session (clarification, preview + team activities):** The instructor revisits difficult concepts revealed by the readiness assurance process, previews the next week's material, and runs further team application activities.
- **Labs (after lectures):** Students work in small lab teams in reproducible emulation environments with formative checkpoints; lab work spans multiple weeks and culminates in three lab deliverables.

The split of in-class time between clarification, preview, and team activities across the two lectures is kept flexible and adjusted from week to week, depending on the specific topic and what the readiness assurance reveals. Student discussion groups are formed based on a brief self-assessment to support complementary team composition with respect to skills, prior knowledge, and interests.

3.2 Readiness Assurance Tests (RATs)

The weekly Readiness Assurance Tests (RATs), explained above, serve as the backbone of the adopted Team-Based Learning (TBL) workflow. They follow the canonical TBL readiness assurance process (preparation → iRAT → tRAT → application activities) [3]. Both the iRAT and tRAT contribute to the portfolio assessment (Section 4).

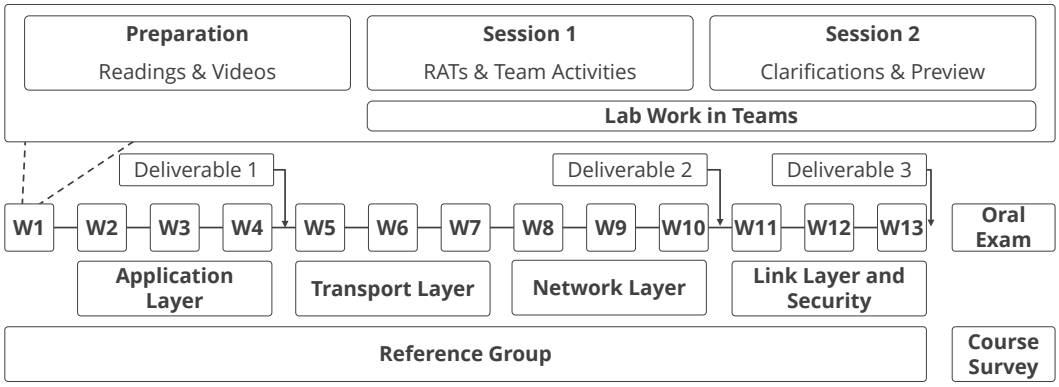


Figure 1: Semester structure with weekly workflow in the course: flipped preparation, iRAT/tRAT, in-class team activities, clarification, and labs.

iRAT and tRAT roles. The iRAT serves as an individual readiness check that incentivizes pre-class preparation and reveals misconceptions early, while the subsequent tRAT externalizes reasoning through peer discussion, requiring teams to reach consensus and immediately confront incorrect reasoning through the feedback mechanism [3]. We deliberately differentiate the two using the revised Bloom taxonomy [5, 16]: iRAT items primarily target lower cognitive levels (remembering and understanding) needed for subsequent work, whereas the tRAT exchanges some questions to trigger richer explanation, application, and analysis in context, thereby improving the quality of team discussion.

Why we add a small number of new tRAT questions. A potential weakness of using identical questions in the iRAT and tRAT (as in the original TBL-approach) is that the team phase can drift towards answer recall ("what did I pick earlier?") rather than explanation and concept application. Farmer et al. [13] evaluated a modified readiness assurance process in which the in-class tRAT used different, but conceptually related questions compared to the iRAT. While this approach did not significantly improve the unit exam performance, the majority of students (69%) preferred the modified approach, which was perceived to improve content interaction and to increase the use of higher-order thinking. Motivated by this evidence and our own classroom observations, we include a small number of new, contextualized questions in each tRAT to reliably trigger discussion beyond simple answer recall.

Tooling: Teampy for scalable paper-based RATs. To keep weekly paper-based RATs manageable in a mid-sized cohort, we use *Teampy*, an open-source toolkit for managing readiness assurance tests. Teampy supports writing RATs in a simple text format and generating individualized PDFs by shuffling questions and answer options per student and per team, reducing copying and streamlining RAT distribution [15]. It also supports rapid grading and reporting by collecting answers in a compact textual format and producing per-question summaries that can guide immediate clarification.

Figure 2 shows a representative pair from the application-layer week. The upper item appears in both the iRAT and the tRAT: it probes a common misconception about HTTP statelessness and can be answered from careful preparation alone. The lower item is added only in the tRAT: it embeds the same conceptual material in an operational scenario and asks teams to weigh configuration options against explicit constraints, so that reaching consensus requires justifying a decision rather than recalling an answer. The complete question sets for all weeks are available in the accompanying repository.

Use of generative AI. Students are allowed a single-page cheat sheet during the RATs. In line with the course AI contract (Section 2.3), generative AI is positioned as a preparation coach for this phase: students may use it for explanations, self-quizzing, and to help condense the week's material onto the permitted one-page sheet. The iRAT and tRAT themselves are written on paper in class without live AI access, which keeps the readiness checks dependent on each student's own recall and reasoning and continues to surface genuine misconceptions.

3.3 Team application activities: discussions and short exercises

After the readiness assurance phase, class time is used for structured team activities that require students to apply concepts, make decisions, and justify their reasoning. This corresponds to the TBL principle that readiness assurance should

Shared iRAT/tRAT item (understanding level)

HTTP is:

(a) a stateless protocol even though mechanisms can be used for tracking users.

(b) a stateful protocol because certain mechanisms allow web sites to track users.

(c) a stateful protocol because it uses TCP, which is a connection-oriented protocol.

(d) a stateless protocol because it uses UDP, which is a connectionless protocol.

Additional tRAT-only item (application/analysis level)

A small departmental website serves one HTML page plus 18 additional objects (images/CSS), all from the same host over HTTP/1.1. Persistent connections are disabled, static files carry no Cache-Control, clients are not configured to use any proxy, and the RTT to the server is about 100 ms with negligible loss. Which change will most reduce the total completion time of the initial page load for typical users under these constraints?

(a) Enable HTTP persistent connections (keep-alive) so the browser can reuse one TCP connection for multiple object requests.

(b) Add Cache-Control (e.g., max-age) and validators (ETag/Last-Modified) to all static objects.

(c) Stand up a forward web cache (proxy) near the server, but keep client browsers unchanged.

(d) Enable cookie-based sessions so the server can maintain state across object requests.

Figure 2: Example RAT questions from the application-layer week. The shared item targets conceptual understanding with distractors from common misconceptions; the tRAT-only item requires reasoning about a configuration decision under operational constraints.

feed into application-focused work, with inter-team discussion facilitated by the instructor [3]. The prompts are either drawn from standard introductory networking materials (e.g., Kurose and Ross [2]) or designed in-house. They follow the topical progression of the course (application-layer protocols, transport behaviors, routing and forwarding concepts, and security mechanisms). A typical activity is presented briefly together with its motivation; then, teams have about ten minutes to work through it before we walk through the proposed solution path and discuss the results together.

Use of generative AI. Aligned with the course AI contract (Section 2.3), these in-class activities are carried out within the team rather than with AI tools. The point of the application phase is to externalize and contest reasoning through peer discussion, so disagreements are meant to be resolved in the group rather than outsourced to a chatbot.

3.4 Labs: reproducible environments, formative feedback, and deliverables

Labs provide the hands-on counterpart to the in-class workflow: students configure services, observe protocol behavior, and practice debugging with realistic tools. The lab design emphasizes (i) reproducible environments to reduce setup friction and ensure comparable observations across teams, (ii) formative feedback during lab work, and (iii) multi-session tasks that culminate in lab deliverables.

The lab infrastructure is organized around a "template-and-update-workflow". Before the course, the staff prepare a Virtual Machine (VM) template containing the required dependencies, including GNS3, supporting system packages, lab tooling, and the notebook environment. For each lab team, this template is cloned into a separate team VM, and access is restricted to the students in that team. New lab material is released through a course Git repository, so students update their environment by pulling the latest notebooks and support files, rather than by installing dependencies or reconfiguring the platform each week.

Lab environment (reproducible emulation). Students work in small lab teams (2–4 students) within reproducible emulation environments that provide each team with an isolated networked setup. This reduces time spent on installation and environment maintenance, and increases time-on-task for experimentation and debugging. Each team is assigned its own lab VM and the associated GNS3 topology, which serves as the team's working environment for the emulated network.

Figure 3 shows a representative topology from the DNS lab. The central part is an internal lab network, here shown as `_net` with subnet `10.20.30.0/24`, where clients, a DNS server, and a web server are connected through a switch. The topology also includes NAT access and Docker network access, allowing the emulated services to interact with the surrounding lab infrastructure while remaining isolated from other teams. Because every team works from an identical template and topology, observations are comparable across teams while each exercise keeps its operational character: in the DNS lab, for instance, students do not only answer questions about DNS conceptually, but configure an authoritative DNS service, connect it to the topology, and verify that name resolution works from the relevant hosts in their own environment.

Formative feedback via interactive notebooks. To provide frequent formative feedback without excessive staff overhead, lab tasks are accompanied by interactive notebooks

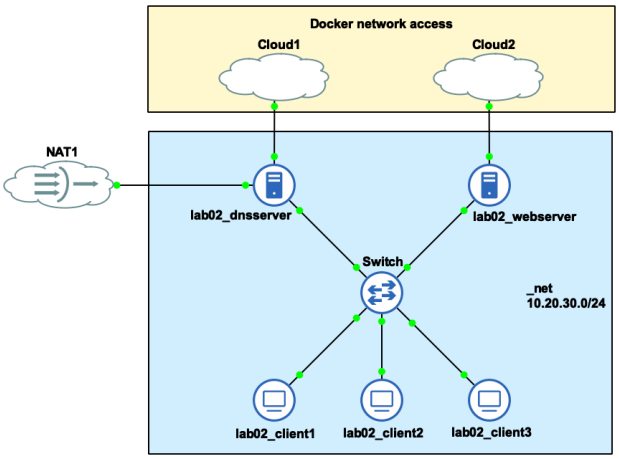


Figure 3: GNS3 topology for a team-specific DNS lab environment. The internal network contains clients, a DNS server, and a web server connected via a switch.

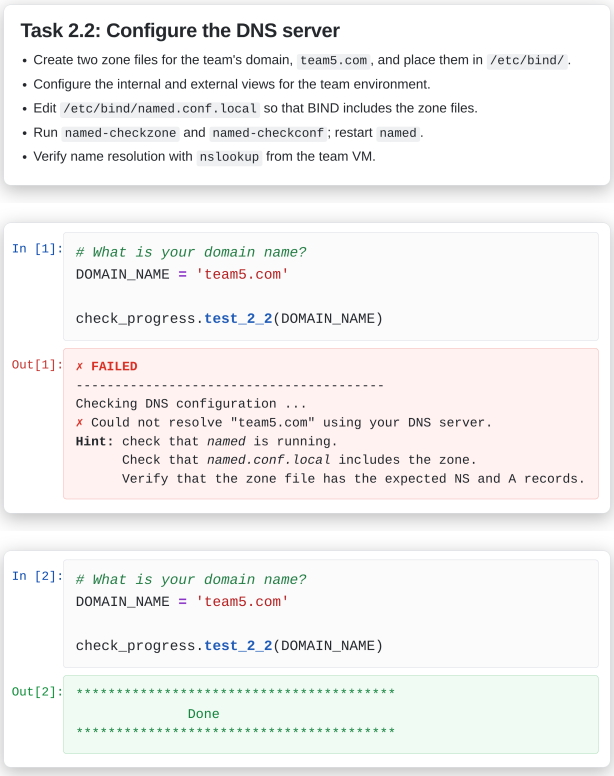


Figure 4: Representative notebook workflow for the DNS lab. Students first receive the task description, then run an automated checkpoint that provides feedback when the configuration is incomplete or incorrect, and finally re-run the same checkpoint after correcting the configuration.

with checkpoints and structured prompts. Checkpoints let students verify intermediate states (e.g., service reachability, configuration properties, or trace-based observations) before progressing, and they enable staff to focus support on teams that are blocked.

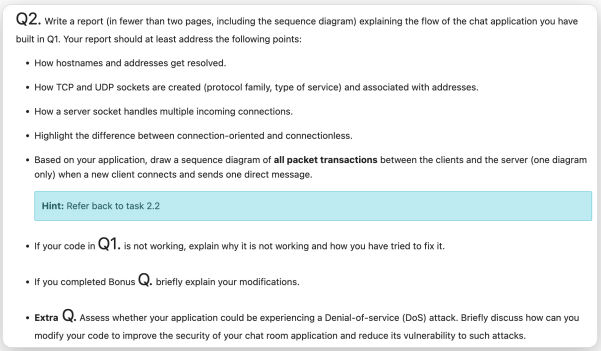
Because the notebooks run from the corresponding team VM, the checkpoints can test the operational state rather than only checking static answers. A DNS checkpoint can, for example, query the team's DNS server, verify that the expected name resolves, and return feedback that points students toward likely configuration errors. This makes the notebook a bridge between the written lab instructions and the emulated network shown in Figure 3.

Figure 4 illustrates this workflow. The task description presents the configuration goal and the files or commands involved. The checkpoint shows the formative role of the notebook: when the test fails, the output identifies the failed condition and gives a concrete hint about likely causes. After students correct the configuration, the same checkpoint can be re-run, producing the successful result.

Multi-session deliverables. Rather than weekly reports, lab work culminates in larger lab deliverables. This approach encourages synthesis across multiple lab topics while keeping the grading workload manageable. Lab deliverables require students to document configurations, interpret measurements and traces, and justify decisions.

The multi-session lab deliverables complement the notebook checkpoints. Although the checkpoints provide immediate feedback on local progress during a lab session, the lab deliverables require students to explain what they configured, why it worked, and how they verified it. For example, students may be asked to include selected configuration excerpts, command output, packet traces, or topology observations together with a short interpretation of the results. This separates formative feedback from summative assessment: the notebook helps students make progress during the lab, whereas the deliverable evaluates their ability to synthesize and communicate the completed work.

Figure 5 shows a representative example. The task description in Figure 5a asks students to explain the flow of a chat application, including hostname resolution, socket creation, connection-oriented and connectionless communication, and packet-level interactions. The corresponding lab deliverable excerpt in Figure 5b illustrates a student response: students connect implementation details, such as DNS resolution and socket setup, to the networking concepts practiced during the labs. This structure supports multi-session work: students can use one session to establish a working configuration, a later session to extend or observe it, and the



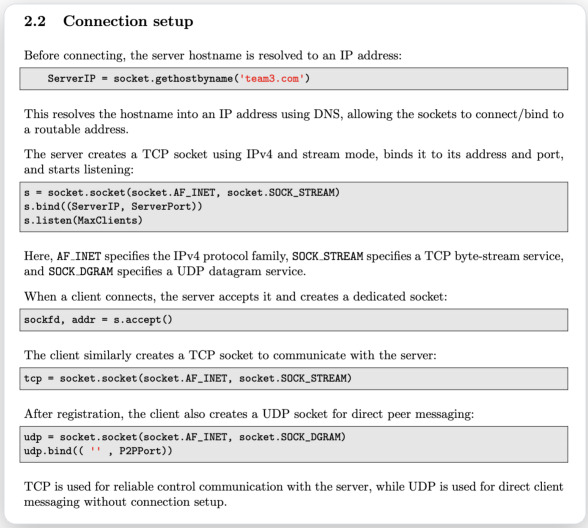
Q2. Write a report (in fewer than two pages, including the sequence diagram) explaining the flow of the chat application you have built in Q1. Your report should at least address the following points:

- How hostnames and addresses get resolved.
- How TCP and UDP sockets are created (protocol family, type of service) and associated with addresses.
- How a server socket handles multiple incoming connections.
- Highlight the difference between connection-oriented and connectionless.
- Based on your application, draw a sequence diagram of **all packet transactions** between the clients and the server (one diagram only) when a new client connects and sends one direct message.

Hint: Refer back to task 2.2

- If your code in **Q1.** is not working, explain why it is not working and how you have tried to fix it.
- If you completed Bonus **Q.** briefly explain your modifications.
- **Extra Q.** Assess whether your application could be experiencing a Denial-of-service (DoS) attack. Briefly discuss how can you modify your code to improve the security of your chat room application and reduce its vulnerability to such attacks.

(a) Excerpt from a deliverable description.



2.2 Connection setup

Before connecting, the server hostname is resolved to an IP address:

```
ServerIP = socket.gethostbyname('team3.com')
```

This resolves the hostname into an IP address using DNS, allowing the sockets to connect/bind to a routable address.

The server creates a TCP socket using IPv4 and stream mode, binds it to its address and port, and starts listening:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((ServerIP, ServerPort))
s.listen(MaxClients)
```

Here, `AF_INET` specifies the IPv4 protocol family, `SOCK_STREAM` specifies a TCP byte-stream service, and `SOCK_DGRAM` specifies a UDP datagram service.

When a client connects, the server accepts it and creates a dedicated socket:

```
sockfd, addr = s.accept()
```

The client similarly creates a TCP socket to communicate with the server:

```
tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

After registration, the client also creates a UDP socket for direct peer messaging:

```
udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
udp.bind(( '' , P2PPort))
```

TCP is used for reliable control communication with the server, while UDP is used for direct client messaging without connection setup.

(b) Excerpt from a deliverable connecting implementation details to protocol concepts.

Figure 5: Excerpts from a lab deliverable.

deliverable to connect the individual steps into a coherent technical explanation.

Use of generative AI. The lab workflow applies the course AI contract (Section 2.3) through the same evidence-based standard. During lab work, students may use generative AI for debugging plans and to interpret observations, but because the checkpoints verify the result of a configuration, observation, or computation in the actual environment rather than checking static answers, a plausible-sounding suggestion passes only once the behavior has actually been configured and verified in the team's environment. For the lab deliverables, AI may assist by providing clarity and structure, but students must cite primary technical sources (textbook, RFCs, documentation) rather than the chatbot and must explain what they configured, why it worked, and how they verified it—an expectation reinforced by the individual oral examination (Section 4).

3.5 Teaching resources and support structure

A key practical consideration for scaling flipped TBL and hands-on labs is staffing: frequent feedback is central to the design but does not come for free. In the 2026 offering, a single instructor was responsible for course teaching and overall grading, including the weekly operational tasks of preparing and running the readiness assurance process, reviewing iRAT/tRAT outcomes to identify misconceptions, delivering short targeted clarifications when needed, and engaging in ongoing dialogue with students during class and via course communication channels.

Lab teaching assistants. Lab activities were supported by four student teaching assistants and a PhD teaching assistant (TA). We deliberately separated responsibilities: the student TAs provided *formative* support during lab sessions (guidance, troubleshooting, and milestone checks), while the PhD TA coordinated the lab program, handled the *summative* assessment of the lab deliverables, and ensured consistent feedback practices across assistants.

Near-peer mentoring. The cohort consisted of 50 students organized into 17 lab groups of 2–4 students, supported by the four student TAs, each mentoring a cluster of groups. To strengthen the learning environment early in the program, and as mentioned, every lab group has a fixed student TA mentor as its first point of contact during lab sessions; this lowers the threshold for asking questions and keeps support consistent across the semester rather than rotating. We intentionally invest in near-peer support in this second-semester course: students build working relationships within and across groups and form early connections to senior students through the mentor structure, helping establish a trusted community.

Scalability and planned staffing reductions. Even though mentors already bundle several lab groups each, near-peer mentoring remains resource-intensive in TA time. For the next iteration, we plan to scale the same pedagogical workflow further with fewer TAs—by enlarging the cluster each mentor covers and formalizing practices for peak demand (e.g., queueing and prioritization during busy lab sessions). Combined with the structured checkpoints and more self-service verification steps described above, we expect this to support larger cohorts without a proportional increase in staffing; we treat this as a design hypothesis to be validated in the next offering.

4 Assessment Design

The course uses a portfolio-based assessment model that combines continuous formative assessment with an individual oral examination. The overall goal is to emphasize learning and engagement, rather than fine-grained point accumulation, in line with the principles described by Edström [12]. The final course grade combines the portfolio (40%) with the individual oral examination (60%).

4.1 Portfolio Assessment

The portfolio comprises three components: the iRATs, the tRATs, and the lab work including its written deliverables. Within the portfolio, the RATs and the multi-session lab deliverables are weighted equally at 50% each.

Both components are graded with a deliberately *coarse-grained* scheme rather than detailed numerical scores: 2 points when the learning outcomes are clearly achieved, 1 point for partial achievement, and 0 points otherwise. For the RATs, the thresholds are based on the number of correct answers (e.g., 2 points from 7 of 10 correct, 1 point from 3 of 10); the lab deliverables use the same three-level scheme, intended to give students a realistic opportunity to reach the higher level.

The thresholds are discussed openly with students and may be raised during the term when performance is consistently strong (e.g., from 7 to 8 correct answers for 2 points); any such change applies only prospectively. The reference group introduced in Section 2.1 is the formal channel for this calibration: it gives students a structured way to signal during the semester whether they feel adequately challenged, so the level can be adjusted mid-course rather than only between offerings. Anonymous tools such as Mentimeter can complement this channel. This reinforces the focus on learning rather than point maximization, and in our experience the model supported sustained engagement.

4.2 Individual Oral Examination

Because most portfolio activities are team-based, the individual oral examination is the primary instrument for assessing each student's personal attainment of the intended learning outcomes, and it represents the largest share (60%) of the final grade. The oral exam follows the assessment principles outlined by Edström [12], particularly the reversal of the burden of proof: the student, not the examiner, is responsible for demonstrating that the learning outcomes have been met.

Each oral examination proceeds in three stages:

(1) **Student summary (10 minutes):** The student presents and summarizes their understanding of selected course topics, using the whiteboard to structure the explanation and to demonstrate achievement of the learning outcomes.
(2) **Discussion and probing (15 minutes):** The examiner asks follow-up questions based on the summary, addressing both what was presented and relevant aspects that were omitted, while focusing on conceptual understanding, reasoning, and transfer to new situations.
(3) **Grading and feedback:** After the student leaves the room, the examiners agree on the grade; the student is then invited back for immediate feedback and grade disclosure.

This format closes the constructive-alignment loop from Section 3.1: by requiring students to justify their reasoning and transfer it to new situations, the oral examination assesses exactly the evidence-based reasoning that the weekly readiness assurance, team discussion, and labs are designed to practice. It thereby provides a valid, individual measure of attainment while preserving the collaborative benefits of the team-based coursework.

5 Student Perspective

This section summarizes student feedback based on internal course evaluation material collected as part of the institution's quality assurance processes. We report only aggregated responses and anonymized excerpts to safeguard students' privacy; the underlying institutional reports are not publicly released.

5.1 Data sources

We primarily draw on the Spring 2026 course survey [11]. It received 32 responses from the cohort of 50 students (64% response rate) and combines Likert-scale items with two free-text questions on the best aspects of the course and on suggested improvements, answered by 24 and 17 respondents, respectively. Where helpful, we cross-check that recurring themes are consistent with internal course and reference-group reports from earlier iterations [8, 9].

We report this material *descriptively*: the aim is to convey the main lines of student feedback that informed our own iteration of the course—what students valued and where they experienced friction—not to provide quantitative evidence of learning gains. The responses may be subject to self-selection and response bias; we therefore treat them as formative input rather than as an evaluation of the design's effectiveness.

5.2 What students valued

On the Likert items, perceptions of the learning design were uniformly positive: all 32 respondents agreed or strongly agreed that the course involved many good learning activities requiring active work rather than just listening (16 agree / 16 strongly agree) and that the learning activities were well prepared and clearly structured (17 / 15); 31 of 32 respondents

agreed or strongly agreed that the course required preparation for the lessons (9 / 22); and all respondents agreed or strongly agreed that, all in all, they were happy with the course (10 / 22) [11].

In the free-text responses, three value drivers recur:

- **Hands-on labs as the learning anchor.** The students most frequently highlighted the labs as the strongest aspect of the course, emphasizing that labs made the theory concrete and relevant. One student described this succinctly as: "The material and practical work/labs showing the relevancy of what we learn in theory".
- **Weekly readiness assurance as pacing and accountability.** The students repeatedly noted that weekly RATs created a productive study rhythm by requiring continuous preparation. For example: "The RATs are really important to keep up with the content of the course, because it requires us to read and understand each week's pages".
- **Team discussion and clarification.** The students valued the team phase (tRAT discussion) and subsequent clarifications as mechanisms to resolve misconceptions and deepen understanding. Illustratively: "Team RATs are also great" and "The discussions and taking the time to make things that are unclear understandable".

5.3 Friction points

Improvement suggestions in the Spring 2026 survey primarily concern operational fine-tuning rather than structural issues [11]:

- **Alignment between weekly readings/RATs and labs.** Some students reported that labs and lessons were not always sufficiently aligned, which could lead to uncertainty during lab work (e.g., "Some labs and lessons were not always aligned, so sometimes I felt a little lost in the labs") [11].
- **RAT clarity and calibration.** Several students requested clearer wording and better contextualization of difficult RAT items, noting that some questions felt unexpectedly difficult or not clearly enough connected to the preparation material [11].
- **Lab support capacity and waiting time.** A recurring concern was waiting time for assistance during labs (e.g., "we had to wait a long time when we needed help") [11]. This is mirrored in the weakest Likert item: only 15 of 32 respondents agreed or strongly agreed that they received many valuable comments on their submissions, indicating that individualized feedback—in contrast to the automated checkpoints—remains the resource-constrained part of the feedback model.
- **Pacing and expectation management.** Individual responses point to inherent trade-offs of the design rather than defects: the weekly preparation rhythm compresses reading time (e.g., before Monday sessions), and some students wished for a clearer picture of what to expect in the oral examination. The former is the cost of the pacing that students elsewhere credit for their continuous engagement; the latter we address through better communication of the exam format rather than structural change.

We further discuss the implications of this feedback in light of the overall course design and implementation in Section 6.

6 Discussion and Conclusions

Taken together, the student feedback is consistent with the intended flipped TBL design: what students reported doing to succeed, largely coincides with what the course intends them to learn. They singled out the labs for making theory concrete—the practical configuration-and-operation competence among the learning goals; they credited the weekly RATs with enforcing continuous preparation, one respondent explicitly noting that this rhythm prepared them for the oral examination; and they valued team discussion and clarification, the collaborative reasoning the TBL phase is meant to develop. This is the constructive-alignment loop showing up in practice.

The design also clarifies the role generative AI can play. Because every activity is built around evidence the student must own—RATs taken on paper, reasoning contested in the team, checkpoints that verify what students actually configured, observed, or computed in their own environment, and deliverables and an oral examination that demand explanation and verification—generative AI fits naturally as a coach for understanding and for suggesting verification steps, but not as a substitute for justifying decisions with evidence. The evidence-based, context-dependent character of networking, where plausible-sounding answers can be wrong without the specific environment and traces, is what makes this distinction hold in practice.

The friction points are better read as a scaling boundary than as structural problems. The recurring concerns—occasional misalignment between weekly RATs and labs, the clarity and calibration of some RAT items, and waiting time for help during labs—are operational fine-tuning rather than objections to the overall design. The waiting-time concern in particular reflects a familiar asymmetry: immediate feedback during the tRATs scales naturally because the whole cohort

is served at once, whereas deep, individualized support during labs is inherently resource-intensive. Our design already absorbs part of this load through the notebook checkpoints, which give students self-service formative feedback and let staff concentrate on genuinely blocked teams. The planned next steps target this remaining pressure directly: tighter alignment between the weekly RAT focus and lab milestones, continued iteration on RAT-item clarity, and more scalable lab-support workflows—bundling more lab groups under each mentor, formalized queueing at peak demand, and more self-service verification.

For educators considering a similar course, the transferable lesson is not any single component but their integration into one recurring weekly cycle. Preparation, in-class readiness assurance, and labs reinforce one another, and a few deliberate choices make the combination workable under realistic staffing: reproducible per-team environments to keep observations comparable and setup cheap; automated checkpoints to scale formative feedback; coarse-grained portfolio grading paired with an individual oral examination to preserve individual validity without discarding the benefits of team-based work; and a standing reference group for in-semester calibration. We release the supporting materials so that others can adopt and adapt the cycle rather than reassemble it from scratch.

None of this comes for free. Running the weekly readiness assurance, maintaining reproducible lab environments, and sustaining frequent formative feedback is demanding and time-intensive for the teaching team. Our experience is nonetheless that the integration repays this cost: it produces a course in which preparation, in-class reasoning, and operational practice reinforce one another, and in which what students value lines up with what they are meant to learn.

Acknowledgments

We thank the PhD teaching assistants and student teaching assistants of all course iterations, whose dedicated support is a cornerstone of the course, the student reference groups of this and previous years for shaping the course design, and all students whose engagement and survey responses inform this experience report. Generative AI (Anthropic's Claude) was used throughout the preparation and revision of this paper as a writing and editing aid, in line with the AI-as-coach principle of Section 2.3; all content and claims are the authors' own.

References

[1] 2022. Mininet: An Instant Virtual Network on Your Laptop (or Other PC). https://mininet.org/. Accessed 2026-05-07.

[2] 2025. Computer Networking: A Top-Down Approach – PowerPoint Slides (8E/9E). https://gaia.cs.umass.edu/kurose_ross/ppt.php. Accessed 2026-05-07.

[3] 2025. TBL Process (Preparation, iRAT, tRAT, Application Exercises). https://docs.lamsfoundation.org/tbl/tbl-process. Accessed 2026-05-07.

[4] 2026. Team-Based Learning Collaborative: Definition and Core Elements. http://www.teambasedlearning.org/definition/. Accessed 2026-05-07.

[5] Lorin W. Anderson and David R. Krathwohl (Eds.). 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, New York.

[6] John Biggs. 1996. Enhancing Teaching through Constructive Alignment. *Higher Education* 32, 3 (1996), 347–364. doi:10.1007/BF00138871

[7] Olivier Bonaventure, Quentin De Coninck, Fabien Duchêne, Anthony Gégo, Mathieu Jadin, François Michel, Maxime Piraux, Chantal Poncin, and Olivier Tilmans. 2020. Open Educational Resources for Computer Networking. *ACM SIGCOMM Computer Communication Review* 50, 3 (July 2020), 38–45. doi:10.1145/3411740.3411746

[8] Course Coordinator TTM4200. 2024. *Course Quality Report TTM4200 Computer Networks, Spring 2024*. Internal quality-assurance report. Norwegian University of Science and Technology, Department of Information Security and Communication Technology. Not publicly available.

[9] Course Coordinator TTM4200. 2025. *Course Quality Report TTM4200 Computer Networks, Spring 2025*. Internal quality-assurance report. Norwegian University of Science and Technology, Department of Information Security and Communication Technology. Not publicly available.

[10] Course Coordinator TTM4200. 2026. TTM4200 Course-Specific Generative AI Guidance and Usage Policy. Course material, Norwegian University of Science and Technology. Available in the accompanying repository, doi:10.5281/zenodo.21218218.

[11] Department of Information Security and Communication Technology, NTNU. 2026. Course Survey TTM4200 Computer Networks, Spring 2026. Not publicly available; aggregated results summarized in this paper.

[12] Kristina Edström. 2016. The Teaching Trick: How to Improve Student Learning without Spending More Time Teaching. Presentation slides. https://www.janleenkloosterman.nl/reports/icab_edstrom_20161109.pdf Slide deck dated 2016-11-09 (PDF copy of PPTX).

[13] Tadd Farmer, Michael C. Johnson, Jorin D. Larsen, and Lance E. Davidson. 2025. Exploring a modification to the readiness assurance process in team-based learning. *Advances in Physiology Education* 49, 2 (2025), 366–373. doi:10.1152/advan.00062.2024 First published Feb 17, 2025.

[14] Thomas Holterbach, Tobias Bühler, Tino Rellstab, and Laurent Vanbever. 2020. An Open Platform to Teach How the Internet Practically Works. *ACM SIGCOMM Computer Communication Review* 50, 2 (April 2020), 45–52. doi:10.1145/3402413.3402420

[15] Frank Alexander Kraemer. 2025. Teampy: Tools for Team-Based Learning (source code). https://github.com/falkr/teampy. Accessed 2026-05-07.

[16] David R. Krathwohl. 2002. A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice* 41, 4 (2002), 212–218. ERIC record: https://eric.ed.gov/?id=EJ667155.

[17] Norwegian University of Science and Technology. 2026. TTM4200 Computer Networks – Course Description. https://www.ntnu.edu/studies/courses/TTM4200. Accessed 2026-07-02.

[18] Marc-Oliver Pahl. 2020. iLab@Home: Hands-On Networking Classes without Lab Access. In *2020 ACM SIGCOMM Education Workshop: Teaching and Learning Computer Networking During the Pandemic and Beyond*. https://s2labs.org/download/publications/2020_SigComm_iLab_at_home.pdf White paper; informal proceedings.