CS 856: Programmable Networks

Mina Tahmasbi Arashloo

Winter 2023

Networks when they started (1970s)

- Small and simple

Tens of nodes

Networks today (2020s)

- Large and complex

Thousands, even millions of nodes

Networks when they started (1970s)

- Small and simple

Networks today (2020s)

- Large and complex

Networks when they started (1970s)

- Small and simple
- A scientific experiment

Networks today (2020s)

- Large and complex

Networks when they started (1970s)

- Small and simple
- A scientific experiment

Networks today (2020s)

- Large and complex
- Critical infrastructure/ Public utility

Networks when they started (1970s)

- Small and simple
- A scientific experiment
- Few simple requirements

Get data from A to B
(preferably without loss 🙂)

Networks today (2020s)

- Large and complex
- Critical infrastructure/ Public utility

Networks when they started (1970s)

- Small and simple
- A scientific experiment
- Few simple requirements

**Get data from A to B
(preferably without loss 🙂)**

Networks today (2020s)

- Large and complex
- Critical infrastructure/ Public utility
- Many complex requirements

- Get data from A to B
- Ensure isolation
- Maintain quality of service
- High throughput
- Low latency
- Low jitter
- …

How does this affect network design, operation, and management?

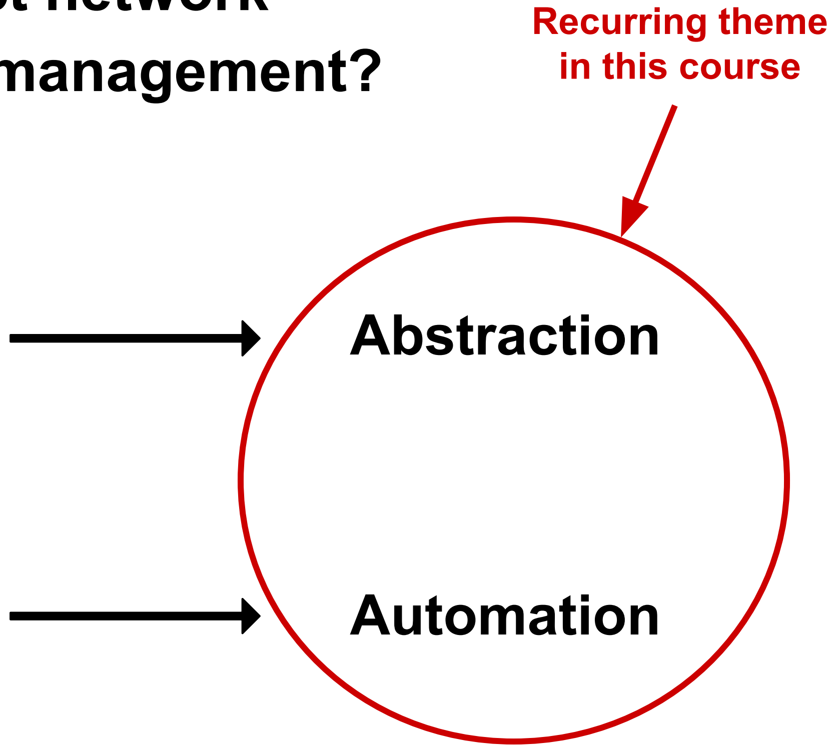
Separate *what* you want the network to do from *how* it is implemented $\longrightarrow$ **Abstraction**

Don't implement in *manually* 🙂 $\longrightarrow$ **Automation**

How does this affect network design, operation, and management?

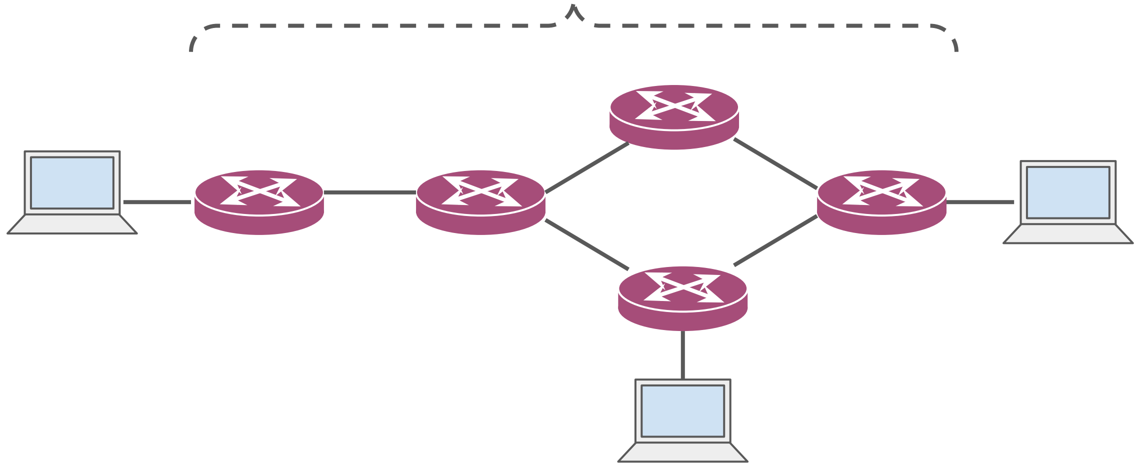
Separate *what* you want the network to do from *how* it is implemented

Don't implement in *manually* 🙂



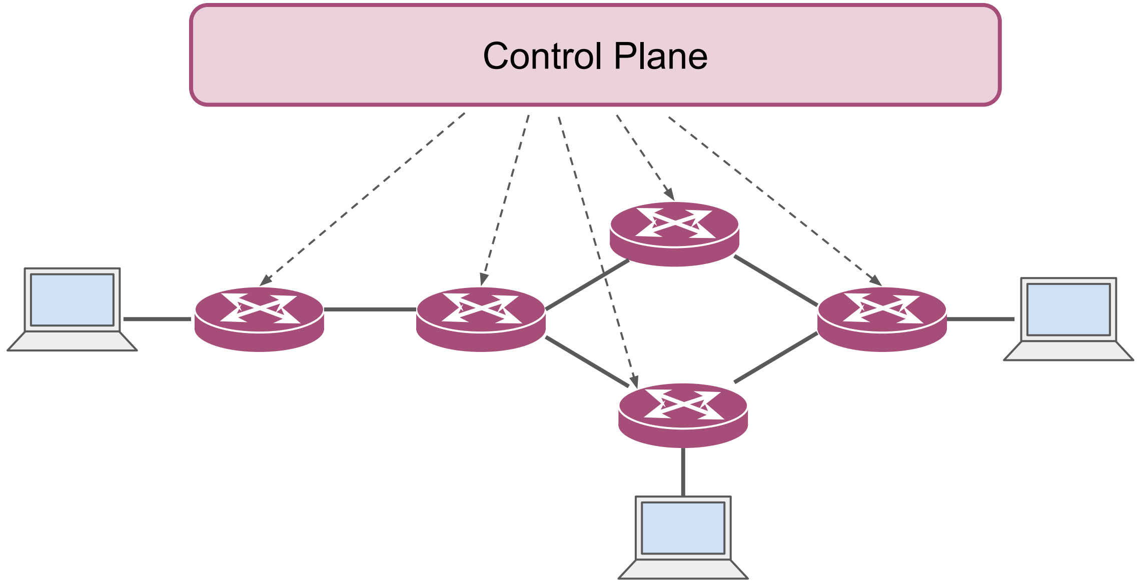
Here are some examples…

Configure a pre-defined set of distributed protocols (e.g., OSPF, BGP, etc.) to pick your desired forwarding paths.



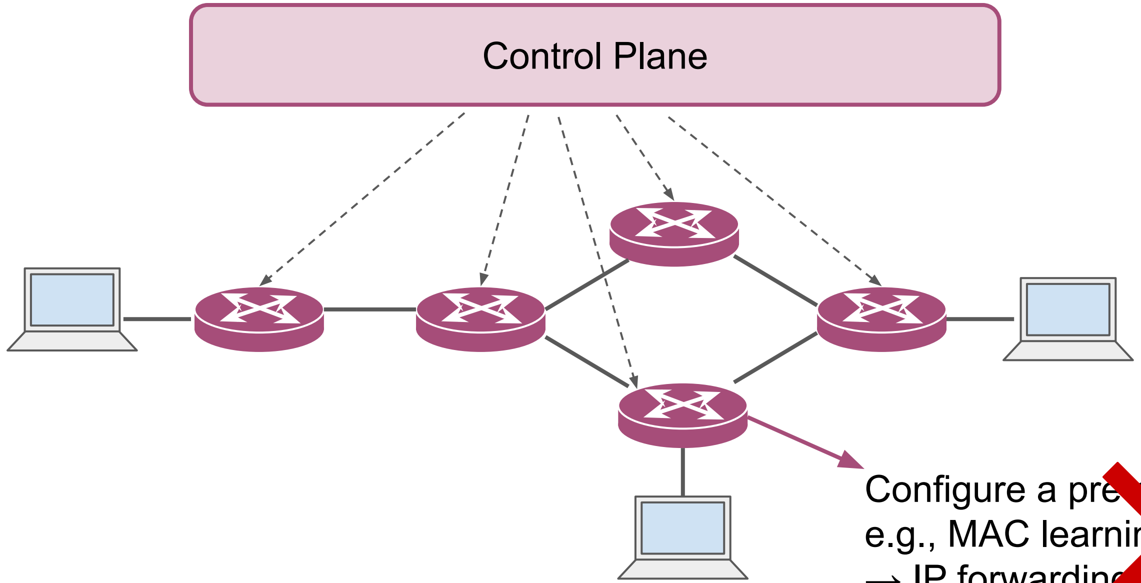
Here are some examples…

- Write a program that decides the forwarding paths.
- Have a runtime configure the underlying protocols or directly communicate forwarding rules to network devices.



Here are some examples…

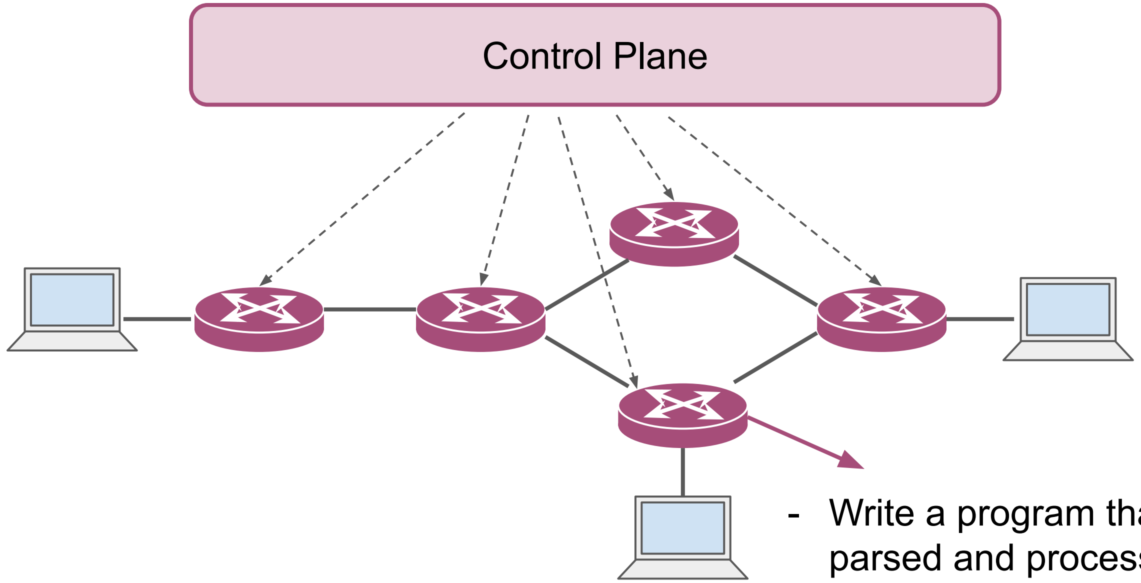
- Write a program that decides the forwarding paths.
- Have a runtime configure the underlying protocols or directly communicate forwarding rules to network devices.



~~Configure a pre-defined forwarding pipeline, e.g., MAC learning → GRE-Tunnel Processing → IP forwarding~~

Here are some examples…

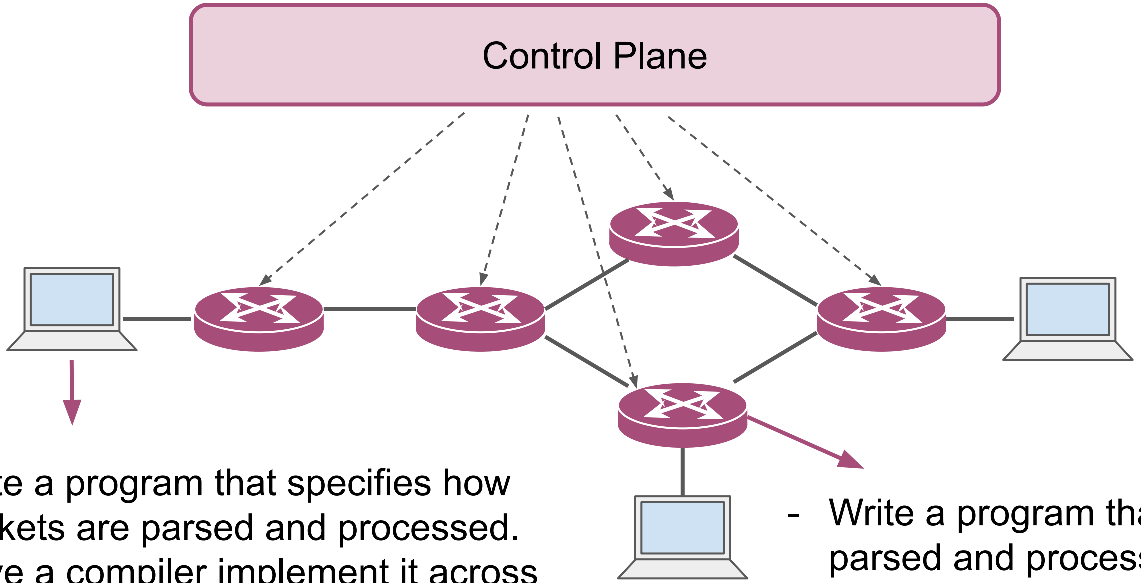
- Write a program that decides the forwarding paths.
- Have a runtime configure the underlying protocols or directly communicate forwarding rules to network devices.



- Write a program that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a program that decides the forwarding paths.
- Have a runtime configure the underlying protocols or directly communicate forwarding rules to network devices.

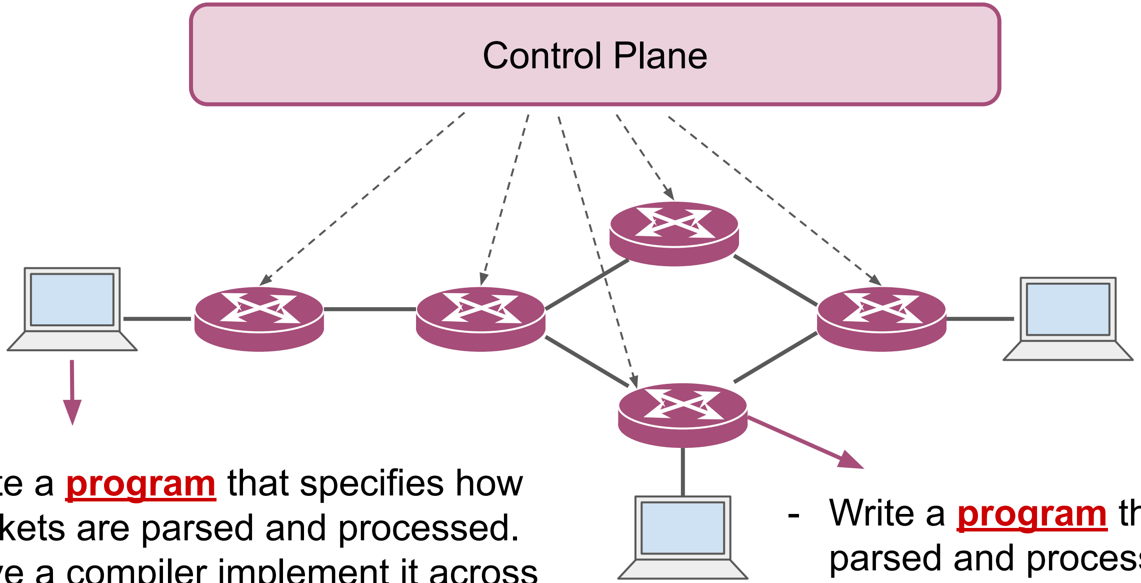


- Write a program that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a program that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a **program** that decides the forwarding paths.
- Have a runtime configure the underlying protocols or directly communicate forwarding rules to network devices.

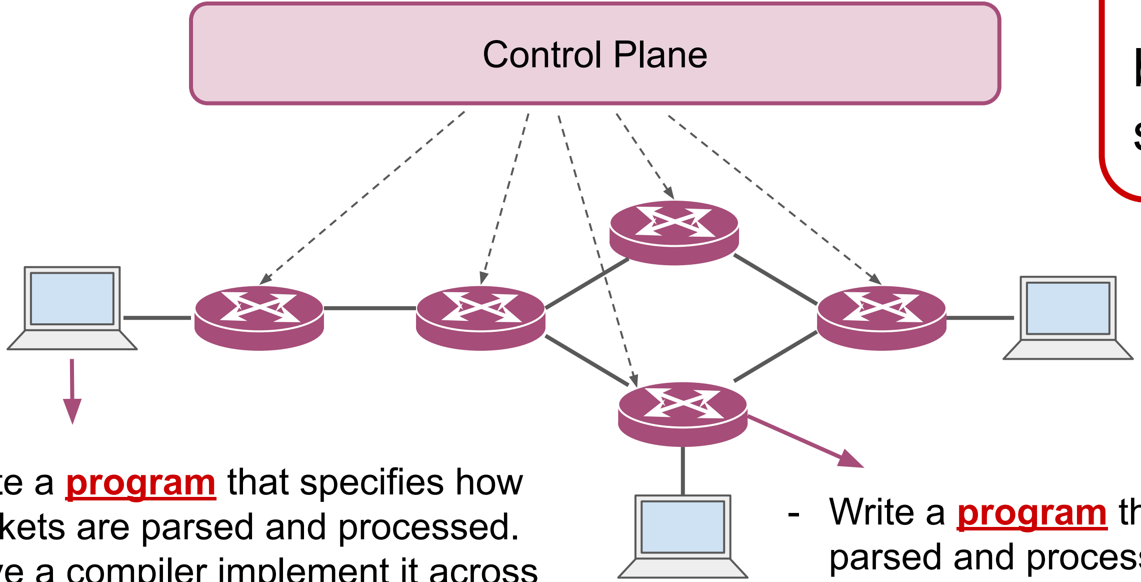


- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a **program** that decides the forwarding paths.
- Have a runtime configure the underlying protocols or directly communicate forwarding rules to network devices.



Treat the network as a big, distributed, and specialized computer

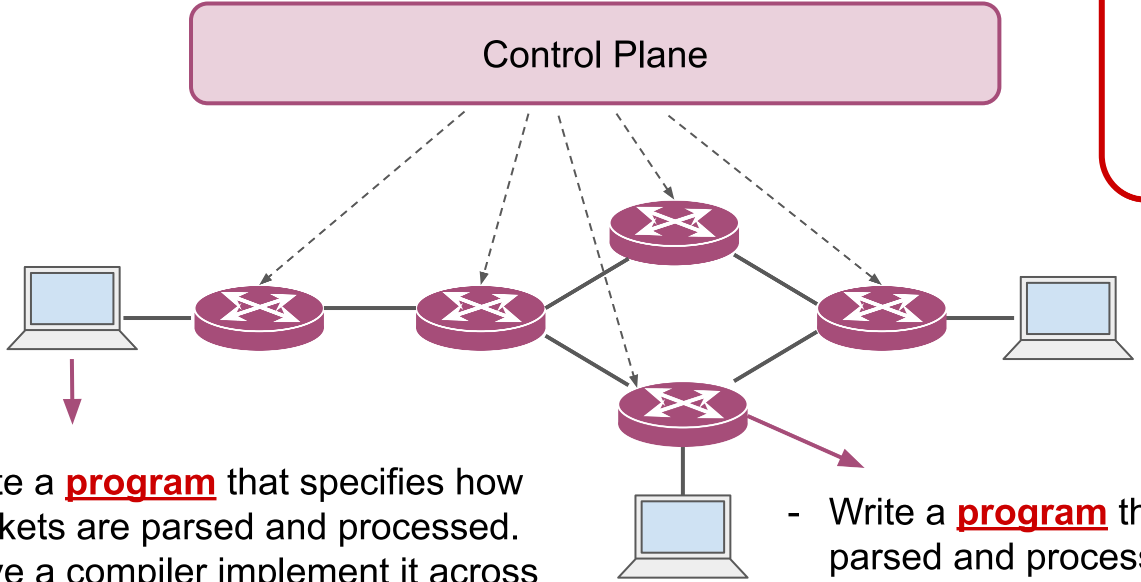
- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a **program** that decides the forwarding paths.
- Have a runtime configure the underlying protocols or directly communicate forwarding rules to network devices.

Programmable Networks



- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

When we can "program" the network…

We can

- Analyze high-level programs to verify network functionality
- Program network devices to
    - measure fine-grained statistics about traffic
    - add a variety of signals about congestion to packets for end-to-end congestion control algorithms
    - implement sophisticated and customized packet scheduling algorithms to provide quality of service (QoS) guarantees
    - accelerate distributed applications (!)
    - …
- …

This Course

(Programming) abstraction and automation applied to different components in networks

what it has enabled

what is left to do

Logistics

- Class is Tuesdays and Thursdays, 1:00pm to 2:20pm.
- Thursdays: lecture followed by discussion
  - Lay of the land for that topic
  - Context about the papers we want to read
- Tuesdays: Paper discussion
- Some classes may be online. The class is in-person by default, unless you receive a **calendar invite** at least 48 hours in advance with zoom information.

Logistics - Continued

- Instructor is me! Email me for any questions and to request office hours
  - prefix the email with [CS856] for a timely reply
- We will use Slack for announcements and other discussions

Course Components

- Reviews (25%)
- Paper Presentation (15%)
- Assignments (10%)
- Project (50%)

Reviews

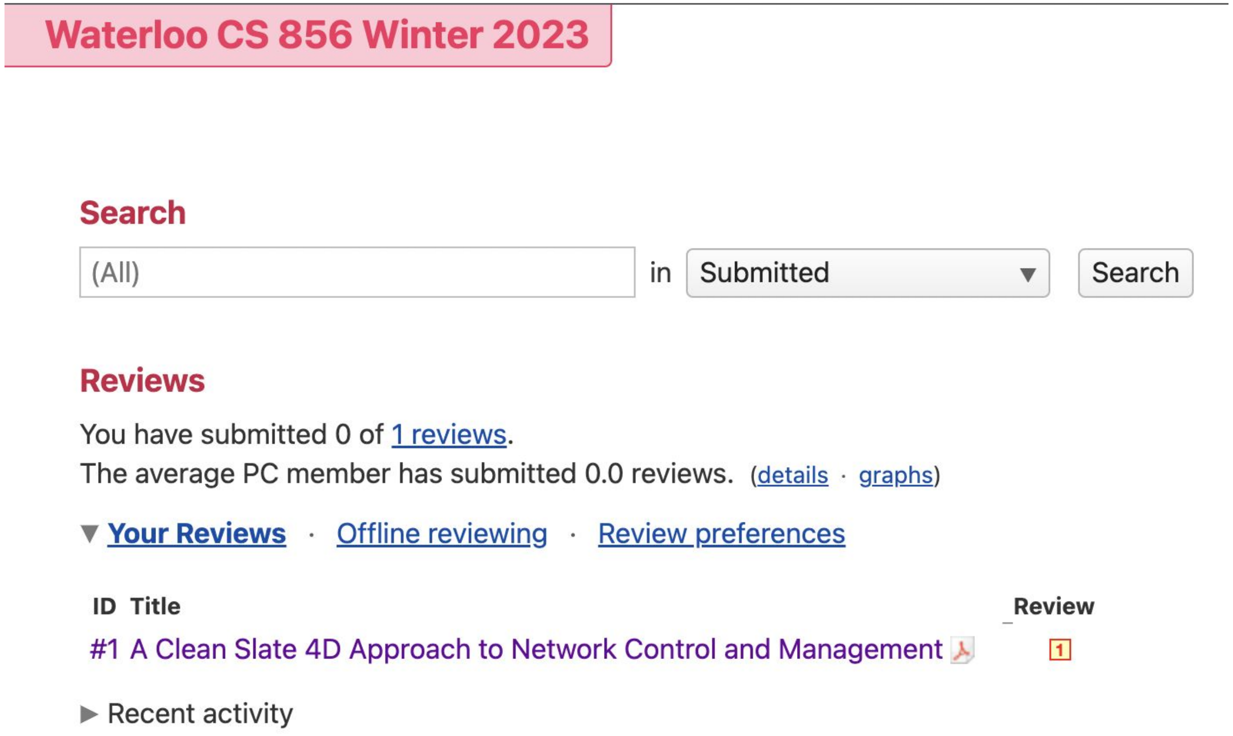
- Two papers each week
- Due on <u>**Mondays at 5pm EST.**</u>
- Will be visible (anonymously) afterwards, so make sure to check them before class on Tuesday.
- Review grading
    - Complete (2 points): adheres to the reviewing guidelines (next slide), clearly demonstrates that the reviewer has read and thought about the paper.
    - Partially Complete (1 point): Misses some but not all the reviewing guidelines, demonstrates that the reviewer has some understanding of the paper.
    - Incomplete (0 points)

Reviewing Guidelines

Each review should be ~500 words and contain the following sections, following the typical format of reviews in networking and systems conferences:

- A concise summary of the paper (1 paragraph)
- A list of the paper's main strengths (at least 2 bullet points)
- A list of opportunities for improvement (at least 2 bullet points)
- Critical analysis and comments (justifying the strengths and improvement opportunities listed in the previous sections)

Reviewing Platform: HotCRP



Waterloo CS 856 Winter 2023

Search

(All) in Submitted ▼ Search

Reviews

You have submitted 0 of 1 reviews.

The average PC member has submitted 0.0 reviews. (details · graphs)

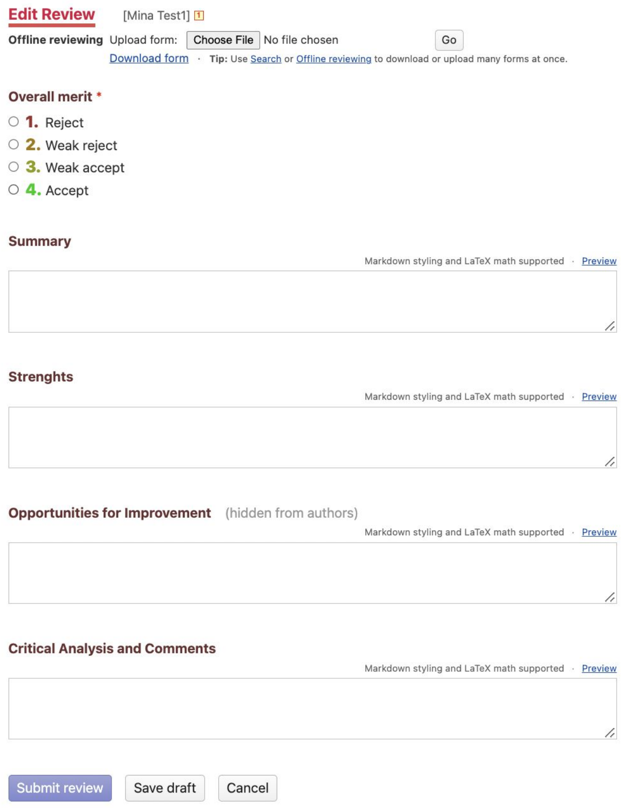
▼ **Your Reviews** · Offline reviewing · Review preferences

ID	Title	Review
#1	A Clean Slate 4D Approach to Network Control and Management	1

▶ Recent activity

Reviewing Platform: HotCRP

- When ready, submit review
- Every Monday at 5pm, the review form is deactivated and you can see all the other reviews submitted for the paper.



Edit Review [Mina Test1]

Offline reviewing Upload form: Choose File No file chosen Go

Download form · **Tip:** Use Search or Offline reviewing to download or upload many forms at once.

Overall merit *

- 1. Reject
- 2. Weak reject
- 3. Weak accept
- 4. Accept

Summary

Markdown styling and LaTeX math supported · Preview

Strenghts

Markdown styling and LaTeX math supported · Preview

Opportunities for Improvement (hidden from authors)

Markdown styling and LaTeX math supported · Preview

Critical Analysis and Comments

Markdown styling and LaTeX math supported · Preview

Paper Presentation

- Each Paper discussion starts by a 10-minute presentation:
  - Describe the context and motivation behind the paper
  - The main problem the paper is trying to solve
  - The main design choices and/or techniques used in the solution
  - A summary of evaluation results
  - 4-5 discussion questions
- Each student is expected to do 1-2 presentations
- Feel free to send me a draft a few days before for feedback

Assignments

- Two programming assignments, each 5% of the final grade
  - Assignment 1: implement a simple network functionality using P4
  - Assignment 2: analyze the correctness of a simple network functionality using existing analysis tools
- The assignments are quite light
- The main purpose is for you to just install and use the tools

Project

- Individually or in groups of two
- Original research projects related to programmable networks
- **One-Page Proposal (Jan 31)**
  - problem statement, context and motivation, and a high-level overview of related work
- **Two-Page Progress Report (March 2)**
- **Presentation (Last week of classes)**
- **Final Project Report (April 10)**
  - 6-page conference-style paper
  - problem statement and motivation, design, evaluation, related work, and future research directions

Final Remarks

- Seminar courses are only as good as the discussions we have.
- Be active, ask questions, and voice your opinion.
- There are no bad ideas, and I mean it 🙂
- If you have a hard time speaking up, let me know and I'll make sure to provide space for you to voice your opinion.
- Be mindful of others in discussions.