

CPSC 416 – Distributed Systems

CPSC 416 – Distributed Systems

Lecture 01 – Course framing, failure-first, GenAI + Goodhart

Tony Mason (fsgeek@cs.ubc.ca)
UBC Vancouver

2026-01-06

Motivation

Why you should take this course:

- Learn core technology
- Challenge yourself
- Worry about *failure*
- Why you should not take this course:
 - ▶ Not interested in systems
 - ▶ Prefer theory/ML/other areas
- Substantial work
- Don't like *failure*

Resources

- Canvas - <https://canvas.ubc.ca/courses/185076>
- Gradescope - TBA
- Github - <https://github.students.cs.ubc.ca/CPSC416-2025W-T2>
- Discord - <https://discord.gg/cpsc416-w25t2>
- Twitch - <https://www.twitch.tv/fsgeek2>

Learning Goals

- Understand fundamental concepts:
- Failure
- Resilience
- Consensus
- Understand problems
- Partial failure
- Consistency
- Performance
- Understand solutions
- Visibility
- Verification
- Usability

Course Assessment (Engagement: 55%)

- Start of course Design Recipe
- Post-lecture self assessment activities
- Mini-projects
 - ▶ **Design** - Design a *solution* to the project, starting with failure.
 - ▶ **Design Peer Review** - Review designs from three other students, providing constructive feedback *for each one*.
 - ▶ **Implementation** - Choose one of the reviewed designs and implement it (using Gradescope).
 - ▶ **Reflection** - Reflect on your **original feedback**, what you learned from the implementation, and how you would improve your *feedback* next time.
 - ▶ **Reflection Peer Review** - Review reflections from three other students, providing constructive feedback *for each one*.

Course Assessment (Capstone: 45%)

- Weekly Reports
- Team Declaration (Optional)
- Capstone Code Submission
- Capstone Design Reflection (Optional)
- Capstone Video (Optional)

Warmup: Networking

- Does anyone remember networking (CPSC 317, for example)?
- What is TCP? What is UDP?
- What is Remote Procedure Call (RPC)?

Failure-first question

When a system fails, what would it mean to “recover” – and what evidence would convince you that it did?

Course method (repeatable pattern)

- Start from a failure.
- Derive the guarantee (invariant) needed to recover.
- Connect it to a mechanism (protocol sketch).
- Name the evidence that would falsify the claim.

Generative AI

Generative AI uses deep learning models to analyze massive amounts of data and produce original outputs in response to prompts. Unlike traditional AI, which may simply categorize or process existing data, generative AI creates novel content, including human-like text, realistic images, and even music or code.

Source: <https://research.ibm.com/blog/what-is-generative-AI>

GenAI + Goodhart (the core risk)

- GenAI makes polished artifacts cheap.
- “Polish” is a metric that Goodharts easily.
- Countermeasure: require an evidence trail (not volume, minimal trace).

Evidence chain (how we operationalize it)

- Signed git history shows incremental construction.
- AI interaction log captures decisions and deltas (not transcript dumps).
- Protocol evidence: invariants + assumptions + reasoning.
- Execution evidence: experiments, failure injections, logs/traces.

In-class activity (evidence micro-exercise)

- Take a plausible design claim.
- Enumerate the minimal evidence that would falsify it.
- Share one “missing assumption” and one “missing measurement”.

Evidence Card (studio template)

- Failure model (what can go wrong?)
- Safety invariant (what must never happen?)
- Liveness assumption (what must eventually happen, under what assumptions?)
- Mechanism sketch (messages/timeouts/retries; recovery; quorum intersections)
- Observability plan (what to log/measure to debug failures)
- Falsifying evidence (what observation would prove you are wrong?)

Logistics

- Readings + quiz (Canvas → Readings page, Week 01)
- Self-assessment opens at 13:00, due next class 11:30 (Pacific)
- DS Labs scaffold: ping service sanity check
- Upcoming DP/capstone milestones (Canvas → Assignments / Modules)

O'Reilly books are available without charge via the UBC library.