CPSC 416 Distributed Systems

Winter 2022 Term 2 (January 10, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer

Welcome!

Welcome to CPSC 416 Winter 2022 Term 2 (January 2023)

Background

My first in-person lecture since 2016

Why you should take this course:

- Learn core technology
- Challenge yourself
- Worry about *failure*

Why you should not take this course:

- Challenging subject
- Substantial work
- Don’t like *failure*

Course Instructor

Tony Mason, Sessional Lecturer

At UBC since 2017

- Previously teaching at Georgia Tech (OMSCS)
- Industry teaching (public/private tech seminars) since 1992
- Research in usable storage (PhD candidate)
  - Where is that file?

Distributed Systems Background:

- Stanford: Cheriton's Distributed Systems Group
- Transarc: AFS, DCE/DFS
- FORE Systems: ATM Networking
- Consulting

Teaching Assistants

Yennis Ye (x.ye.99@alumni.ubc.ca)

Eric Yan (ericy676@student.ubc.ca)

Sandhu Japraj (japraj.sandhu@alumni.ubc.ca)

Waitlist

Current waitlist has 86 people!

- Come to class
- Work on assignments
- People will drop

Current 80 students:

- Consider workload

Resources

Canvas - https://canvas.ubc.ca/courses/106384

My website - https://fsgeek.ca/cpsc-416-winter-22-term-2/cpsc-416-syllabus/

Gradescope - https://www.gradescope.ca/courses/9399

Piazza - https://piazza.com/class/lci4krn99vj7gr

Discord - https://discord.gg/5vB56PCd

Twitch - https://www.twitch.tv/fsgeek2

Communications

Use Piazza for **all** official course-related communications

- Not on Piazza? Not official.

TA Office Hours: TBA

Instructor:

- Private Meeting: by appointment
  - In person (must book a private space)
  - Online (TBA)
- Open Hours Online: Zoom (Meeting 860 4678 8629, Passcode 521130)
  - January 16, 2022 15:00-16:00

Networking

Does anyone remember networking (CPSC 317, for example)?

What is TCP? What is UDP?

What is Remote Procedure Call (RPC)?

Course Overview

Learning Goals

Schedule

- Lab 1 due January 16, 2023

Exam: Final Only

Advice:

- Plan before you code
- Choose optional projects carefully
- Use Piazza

Learning Goals

Understand fundamental concepts:

- Failure
- Resilience
- Consensus

Understand problems

- Partial failure
- Consistency
- Performance

Understand solutions

- Visibility
- Verification
- Usability

Workload

Ivan's CPSC 416:

- *The workload for this course is easily double that of any other course I had this term.*
- *Ivan has very high expectations of his students.*
- *I love and hate the fact that this class was a "sink or swim" approach to learning.*

My distributed systems course (GT CS 7210):

- *[I]t truly gave me an appreciation for how difficult it is to write correct distributed applications/protocols.*
- *If you are interested in distributed systems, I highly recommend it. Just set your expectations right - The projects are not some trivial ones that you can sit back, relax, and work out little by little. It requires a high level of focus and dedicated debugging effort.*

Projects

Based on DSLabs: https://github.com/emichael/dslabs

- Some changes to allow using Gradscope
- Report also required

Lab 1: Intro

Lab 2: Client-Server

Lab 3: Primary-Backup

Lab 4: Paxos

Lab 5: Sharded Key-Value Store

These go from easy (1/2) to challenging (3) to difficult (4/5).

Distributed System Examples

Blockchains (including Bitcoin and Ethereum)

Hadoop File System (HDFS)

Cyber-physical systems

Folding@home

Kafka

Domain Name System (DNS)

Cloud Services: AWS, Azure, GCP

Distributed Databases

System versus Application

Abstraction

- API
- Protocol
- Semantics

Resilience

- Fault Tolerant (*failure handling)*
- Byzantine behavior

Scalability

More Examples?

What can you think of that is a distributed system?

What does Distributed Mean?

Physical separation

- Eliminate centralized physical components
- *Remove "fate sharing"*

Availability

- Continued progress despite failure
- Scalability (*slowness* mimics unavailability)

Communications

- Messages between components
- Networks = common source of *failures*

Characteristics

Networking

Asynchronous

Decentralized

Failure resistant

Task Parallelism

Scalability (Performance)

Example: YouTube

Replicated Videos

- Multiple copies of the same video in disparate locations

Scalable

- Spread client load across copies
- Route clients to “nearby” resources
  - Geo-distribution
  - Network sensitive tuning

Distributed Storage

Dropbox, Google Drive, OneDrive, etc.

- Replication across personal devices
- Disconnected operation
- Download on demand
- Consistent data access

Permits sharing

Makes searching challenging (*my* research area)

Distributed Transactions

National Association of Securities Dealers Automated Quotations (NASDAQ)

Distributed Systems started with *databases*

- Transactions – help deal with *failure*
- Atomicity
- Consistency
- Isolation
- Durability

Trusted because it guarantees:

- Strong consistency
- Security

Distributed Systems Challenges

Protocol Complexity (synchronizing machines)

Measuring Performance

Consistency: strong (linearizable) versus weak (eventual) models

Failures: machines, networks, bad actors, partial

Security

Failure

Distributed Systems must handle *failure*

Failure occurs *all the time*.

Exhaustive failure testing *is not possible*

Each lecture I will start with an example of a failure

- Real systems
- Real failures
- Real explanations

Failure: World of Warcraft

November 28, 2022: Blizzard Entertainment Released “World of Warcraft: Dragonflight”

- Simultaneous release all over the world at 15:00 PT

Everything melted down: https://us.forums.blizzard.com/en/wow/t/an-engineering-update-on-the-dragonflight-launch/1437657

Distinct changes interacted negatively:

- Timed Event Trigger (3 pm PT release)
- Encrypted Data

Failure (2)

We now know that the lag and instability we saw last week was caused by the way these two systems interacted. The result was: they forced the simulation server (that moves your characters around the world and performs their spells and abilities) to recalculate which records should be hidden more than one hundred times a second, per simulation. As a great deal of CPU power was spent doing these calculations, the simulations became bogged down, and requests from other services to those simulation servers backed up. Players see this as lag and error messages like "World Server Down".

Failure (3)

As we discovered, records encrypted until a timed event unlocked them exposed a small logic error in the code: a misplaced line of code signaled to the server that it needed to recalculate which records to hide, even though nothing had changed.

Failure (4)

Mitigation makes it worse!

Boats have been a problem in the past, so we turn on portals while we continue investigating. Our NFS is clearly overloaded. There's a large network queue on the service responsible for coordinating the simulation servers, making it think simulations aren't starting, so it launches more and starts to overwhelm our hardware. Soon we discover that adding the portals has made the overload worse, because players can click the portals as many times as they want, so we turn the portals off.

Failure (5)

Recovery after failure can cause more problems

Pushing a fix to code used across so many services isn't like flipping a switch, and new binaries must be pushed out and turned on. We must slowly move players from the old simulations to new ones for the correction to be picked up. In fact, at one point we try to move players too quickly and cause another part of the service to suffer. Some of the affected binaries cannot be corrected without a service restart, which we delay until the fewest players are online to not disrupt players who were in the game. By Wednesday, the fix was completely out and service stability dramatically improved.

Reality Check

Big systems are hard to test

Most big systems *are* distributed

Failures happen *all the time*

If you can imagine a failure it *will* happen

Your imagination is not as good as reality.

An alternate path through CPSC 416

This is an experiment

It may very well be a bad idea.

I’m going to try it anyway.

Expected effort: ~50 hours of *productive* development work, ~10 hours for a *useful* development report.

The instructional team will evaluate and determine the value of what you have contributed.

Alternative 1: Team Project

A team of three people work together to build a useful distributed system.

- You decide what the system is
- You explain what makes it useful

Each team member has a different role:

- Everyone works to define the project
- One person implements the system
- One person implements the client that uses the system
- One person validates the system *and* the client meet the project requirements

Alternative 1: Say what?

Note: I'm not defining the language you use.

- Go
- Rust
- Java
- C#
- ARM assembly

Note: I don't define the service

- You *do* have to get me to approve your project plan

You deliver a project: joint design/requirements, individual implementation report

We will grade it: how useful is it, how well executed is it, how well did each member of the team in fulfilling their requirements.

Alternative 2: Pick an OSS Project

There are a *lot* of distributed systems projects

- Pick one
- Find someone involved in the project willing to mentor you
- Define what you will contribute to the project
- Get me to sign off on your proposal
- Contribute to it

Grading is done based upon:

- Feedback from your mentor
- Your experience report
- Impact of your contribution to the OSS project
- Evaluation by the teaching team as to the merit of your work

Observation: Extra Credit

I have a large and varied t-shirt collection.

Each day I will wear a different t-shirt.

Except once.

Once I will wear the same t-shirt.

On the final exam, you will be given a selection of t-shirts from the semester.
Extra credit if you pick the one that was the duplicate.

Readings

Required:

Collaboration versus cheating

Recommended:

What good are models and what models are good?

Fallacies of Distributed Computing Explained

Introduction to Distributed System Design

The Rise of Cloud Computing Systems (Video)

Questions?

How to use this template

Please note: This template has a variety of slides for your use. To select what slide you would like, click on the drop down menu beside “new slide” button in the top left corner, and pick the corresponding slide. To insert text, simply double click on the text box and start typing. Please be aware that copying and pasting text may change how the font looks. It is better to type directly onto the slide. Also note that larger fonts (size 14+) work better for presentations than smaller sizes. This template uses the font Arial, as PowerPoint users will experience technical difficulties if using UBC’s official fonts. If desired, images can replaced by going into the “Master” view and applying your own image. Please ensure you have the rights to an image before using it.

The following slides are here for visual reference only. Please delete or edit as needed for your own presentation. If you have any questions about how to use this template, please contact UBC Communications and Marketing at comm.marketing@ubc.ca

Insert title here

Insert subtitle here

Name, position

Insert title here

Insert subtitle here

Name, position

Insert title here

Insert subtitle here

Name, position

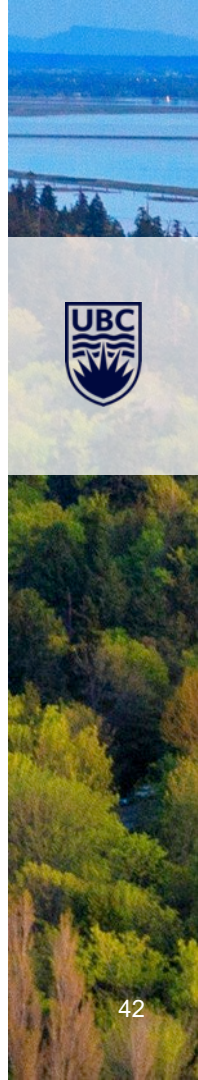
Insert title here

Insert subtitle here

Name, position

Page title

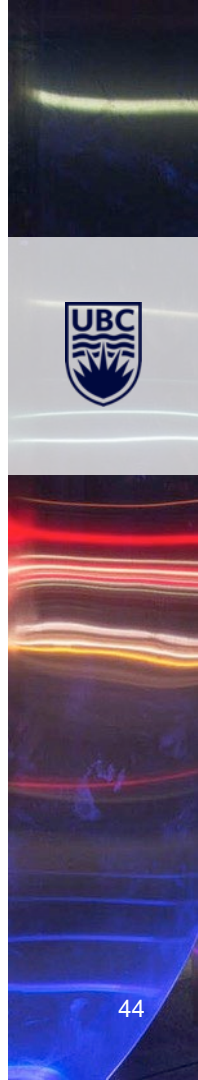
- **Bullet point** list
- **Bullet point** list
- **Bullet point** list
- **Bullet point** list



Insert chapter title

Page title

- **Bullet point** list
- **Bullet point** list
- **Bullet point** list
- **Bullet point** list



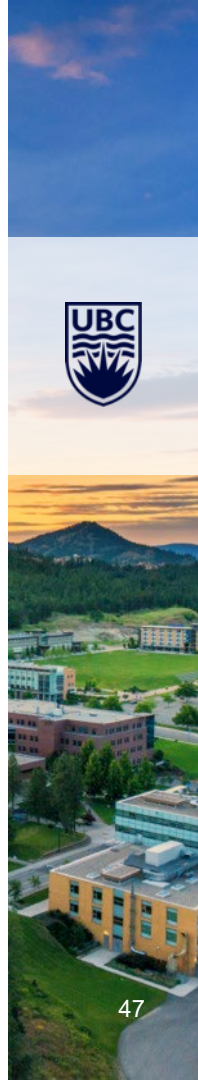
Insert chapter title

Page title

- Bullet point list
- Bullet point list
- Bullet point list
- Bullet point list

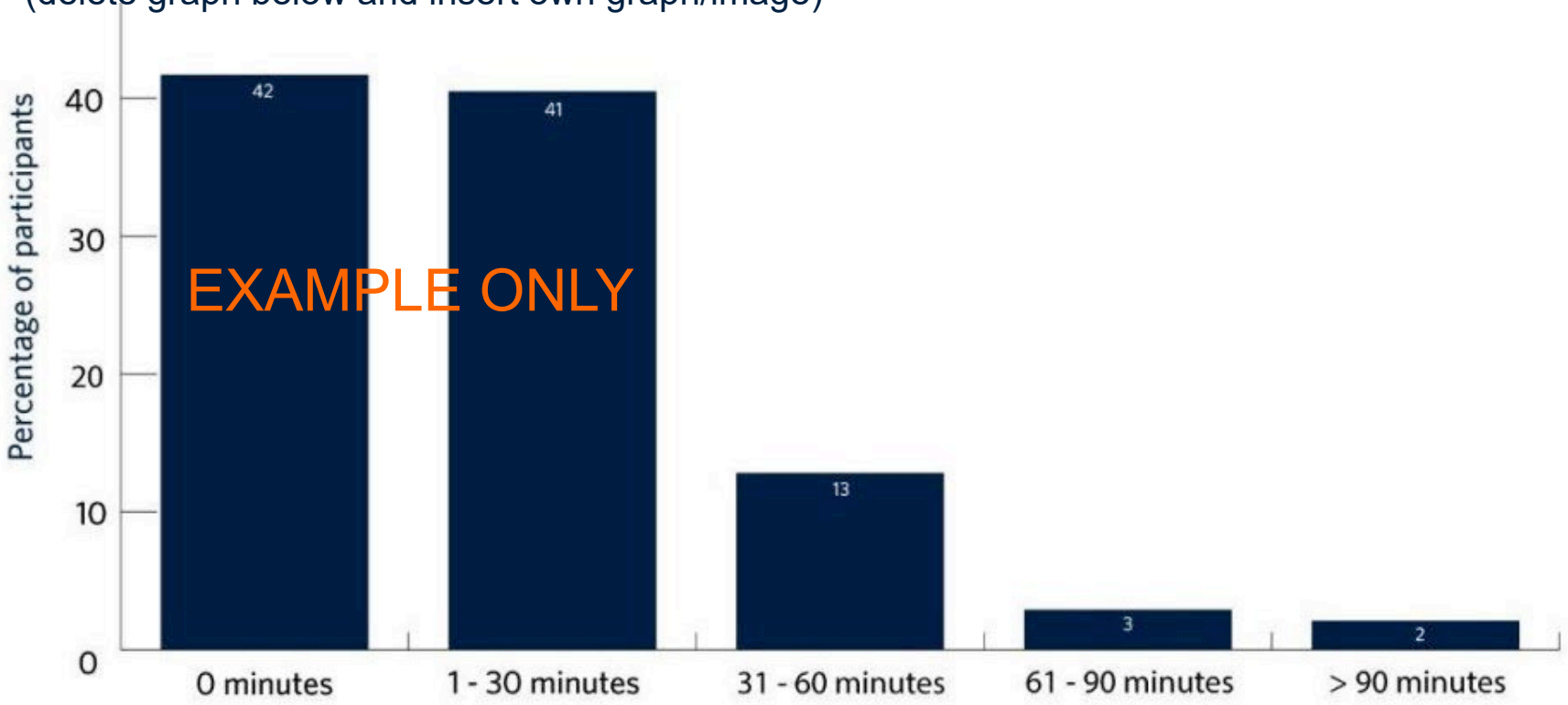
Page title

- **Bullet point** list
- **Bullet point** list
- **Bullet point** list
- **Bullet point** list



Insert title

(delete graph below and insert own graph/image)




UBC
THE UNIVERSITY OF BRITISH COLUMBIA


THE UNIVERSITY OF BRITISH COLUMBIA
THE UNIVERSITY OF BRITISH