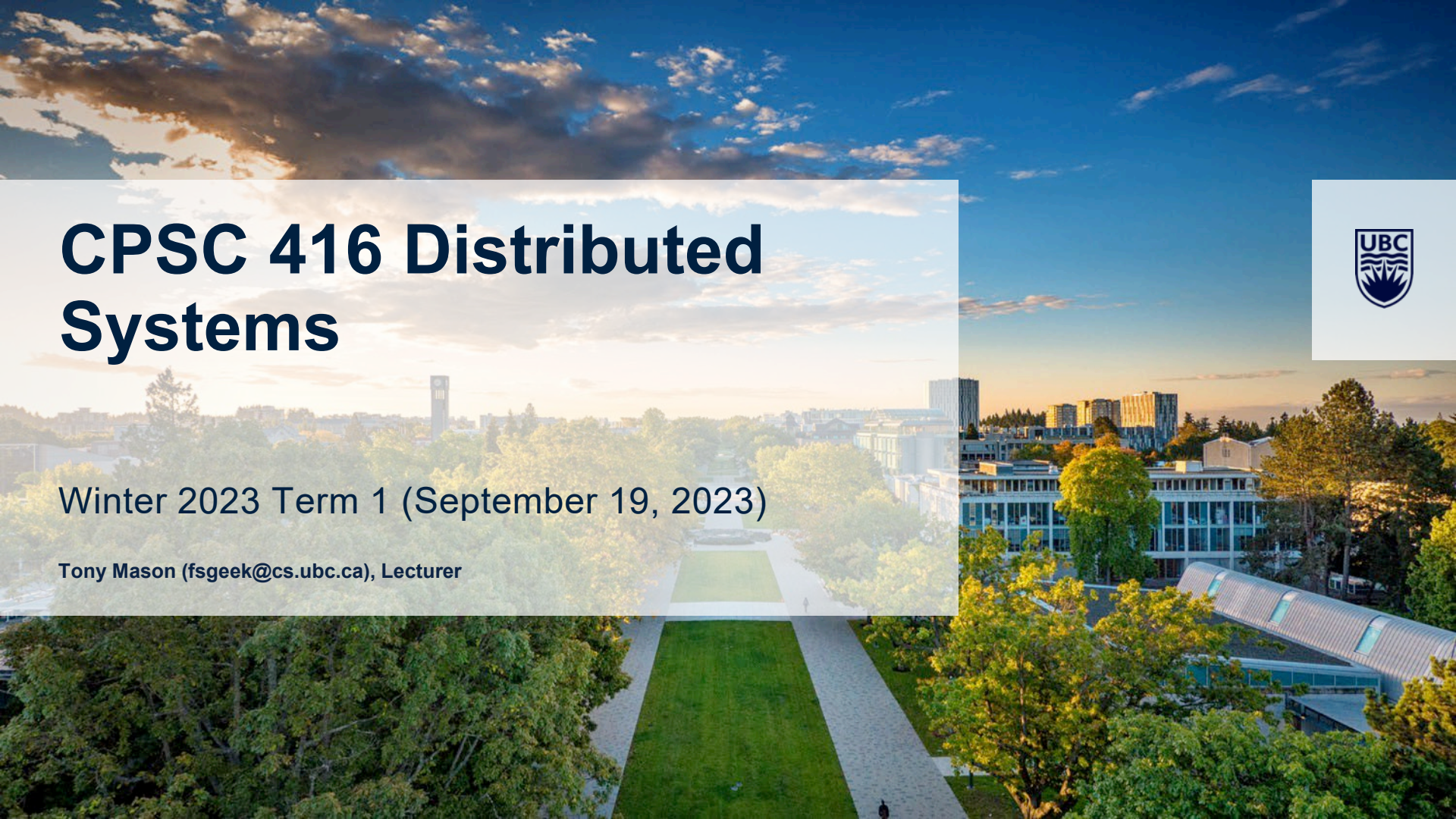


CPSC 416 Distributed Systems

Winter 2023 Term 1 (September 19, 2023)

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

Logistics

Teaching Assistants

Andy Hsu (andy.hsu@alumni.ubc.ca)

Hamidreza Ramezanikebrya (hamid@ece.ubc.ca)

Jonas Tai (jonastai@student.ubc.ca)

Cathy Yang (kaiqiany@student.ubc.ca)

Office Hours

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

- Not on Piazza? Not official.
- Canvas “comments/messages” **are not monitored**

Office Hours:

Who	When	Where
Tony	Monday 14:00-15:00 Wednesday 16:00-17:00	Discord
Andy	Thursday 19:00-20:30	Discord
Hamid	Friday 16:30-18:00	Kaiser 4075
Jonas	Thursday 11:00-12:30	X150, Table 1&2
Cathy	Friday 09:00-10:30 (Starting Sep. 22)	X237

Self-Assessment

This week

- Post-lecture self-assessment activity – Due Tuesday (Sep 26 @ 17:00)
  - Structure change: fewer reflection questions
  - Structure change: more content questions (T/F, MC)

Note:

- You are strongly encouraged to collaborate with others on this
- You should use tools at your disposal to answer these questions
- As previously noted, you get full credit if you submit. **Do not forget to submit it.**

Today’s Failure

Cloudfare

[A Byzantine failure in the real world]

November 2, 2020 14:43 UTC

- *Partial* switch failure
  - Link Aggegation Control Protocol working
  - Border Gateway Protocol working
  - Virtual Port control (vPC) *not* working
  - Data Plane dropping packets

This failure scenario is completely invisible to the connected nodes, as each server only sees an issue for some of its traffic due to the load-balancing nature of LACP. Had the switch failed fully, all traffic would have failed over to the peer switch, as the connected links would've simply gone down, and the ports would've dropped out of the forwarding LACP bundles.

Six minutes later, the switch recovered without human intervention. But this odd failure mode led to further problems that lasted long after the switch had returned to normal operation.

Cloudflare failure

November 2, 2020 14:40 UTC

etcd begins exhibiting errors

- Uses the *Raft* consensus protocol (coming soon to a lecture near you!)
- Experiences a *Byzantine* fault

> In the event that the cluster leader fails, etcd uses the RAFT protocol to maintain consistency and establish consensus to promote a new leader. In the RAFT protocol, cluster members are assumed to be either available or unavailable, and to provide accurate information or none at all. This works fine when a machine crashes, but is not always able to handle situations where different members of the cluster have conflicting information.

Cloudflare Failure

November 2, 2020 14:45 UTC

Database system promotes a new primary database

> When etcd became read-only, two clusters were unable to communicate that they had a healthy primary database. This triggered the automatic promotion of a synchronous database replica to become the new primary. This process happened automatically and without error or data loss.

> For the other cluster, however, performant operation of that database *required* a replica to be online. Because this database handles authentication for API calls and dashboard activities, it takes a lot of reads, and one replica was heavily utilized to spare the primary the load. When this failover happened and no replicas were available, the primary was overloaded, as it had to take all of the load. This is when the main impact started.

Cloudflare Failure

November 2, 2020 21:20 UTC

Database Replica Rebuilt

End of service disruption

The cascade of failures in this incident was interesting because each system, on its face, had redundancy. Moreover, no system fully failed—each entered a degraded state. That combination meant the chain of events that transpired was considerably harder to model and anticipate. It was frustrating yet reassuring that some of the possible failure modes were already being addressed.

The distributed systems community has pointed out that the failure we've encountered would be better characterized as an omission fault rather than a Byzantine fault. Omission faults are much more specific and can be tolerated without BFT protocols.

Design Project 1

Learning Goals

Change focus from *coding* to *designing*

Learn how to design

- Learn by “doing”
- Most people will struggle with this and turn out a poor-quality design

Learn what works and what doesn’t work

- Peer feedback – expect it to be a struggle for the first time
  - Likely see some poor designs
  - Likely won’t know what a *good* design needs to be
- Implementation – expect to be unhappy with the design
  - **This is the real purpose – reflect on *what would have made it better.***
- Implementation Report: how to make the design *better*.

Design Process

Step 1: **Identify the problem**

- State your understanding of the problem
  - Identify key components of the system
  - How do the components interact?
- Explain the challenges
  - What are the potential failure scenarios?
  - What are the guarantees to clients:
    - Consistency
    - Availability
- Identify the goals
  - What are the desired outcomes?
  - What are the performance and scalability expectations
- Identify the *non-goals*
  - Establish expectations: what problems are you not trying to solve

Design Process

Step 2: **Propose a solution**

- Provide “enough detail” someone else will understand your proposed solution
  - Explain component roles
  - Explain data and control flow
- Use graphics and text.
  - Flowcharts
  - State diagrams
  - Architectural diagrams
- Use modular decomposition
  - Use smaller modules and components to simplify
  - Explain what each module/component does (responsibilities)

Design Process

Step 3: Describe how to evaluate the solution

- How do you determine if your goals are achieved?
  - What are the test scenarios or use cases to validate the solution?
  - How will you simulate failures and measure recovery time?
- What metrics validate your solution?
  - Latency how long does it take to process a request:
    - Average time
    - Tail latency (95%, 99%, 99.9%, etc.)
  - Throughput: how many requests can the system handle per unit time?
  - Availability: what is the uptime of the system?
  - Consistency: how to evaluate data consistency of the system?

Design Project 1

The *problem* is laid out in the DS Labs Primary/Backup Replication project

- Project 3 in our Github: https://github.students.cs.ubc.ca/CPSC416-2023W-T1/project3
- Note: we didn't do Projects 1 or 2, so there's no design to use for them
  - Project 2: Client/Server involved the key-value store.
  - Keep it simple: it's an in-memory hash map
- The design is stand-alone, even if the DSLabs project is not.

You are not implementing this. You can code it if you think that's the most effective way to design a solution, but your *report* isn't code you write, it is your model for the problem.

Learning Goals

Learning Goals (Petrov Chapter 8, Part 1)

Goals for this conversation is understanding:

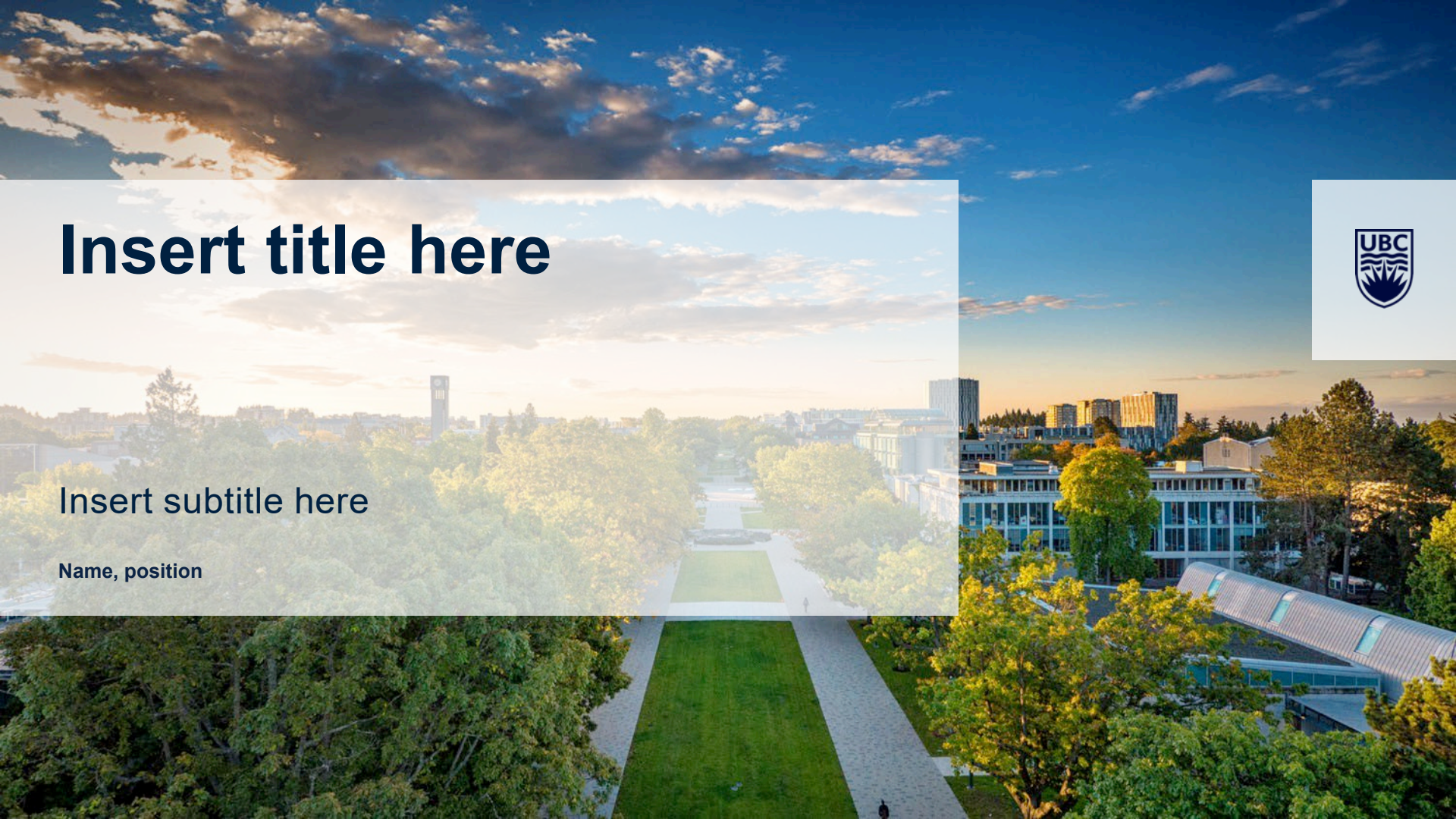
- Concurrent execution
- Shared State
- Time's relative nature
- Consistency
- Failure types
- Handling failures
- Distributed systems abstractions

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

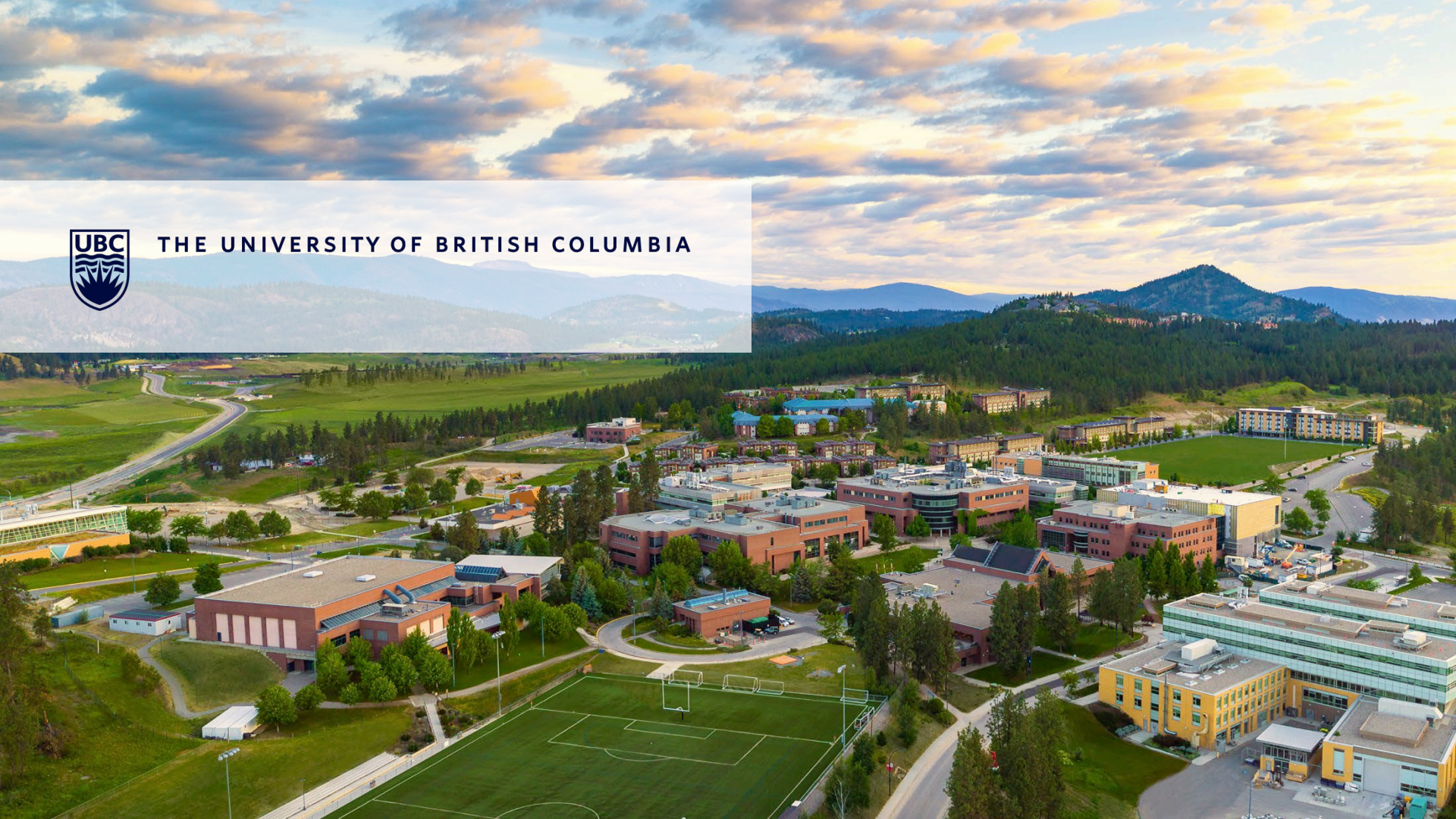
Insert subtitle here

Name, position

Insert title here

Insert subtitle here

Name, position


UBC
THE UNIVERSITY OF BRITISH COLUMBIA