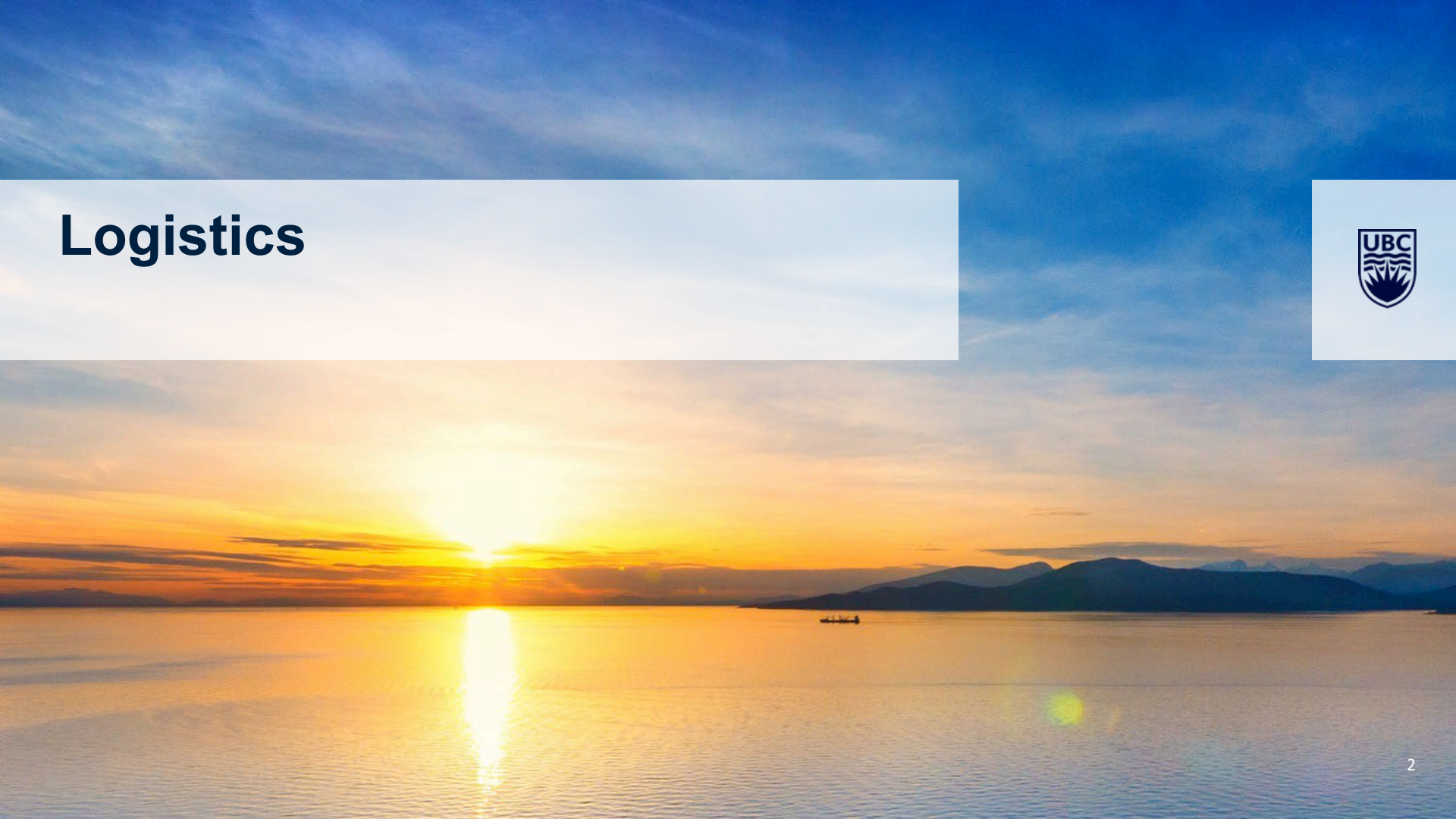
# CPSC 416 Distributed Systems

Winter 2022 Term 2 (January 24, 2023)

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

# Logistics



# Deadlines

**Project 3 Released.** Initially Due: February 13, 2023

**Project 4 Released.** Initially Due: March 13, 2023

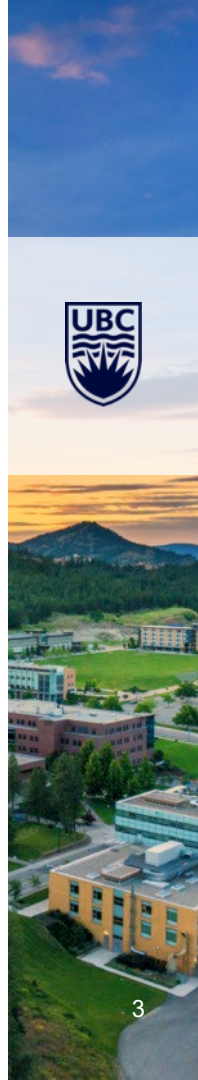
**Project 5 Released** Due: April 13, 2023

Note: all project work is due April 13, 2023. Late projects have a 75% score cap.

**Alternate Path 1 & 2:** Initial Proposal due January 30, 2023.

Instructor Office Hours:

- Zoom Office Hours (Tuesday) @ 13:00-14:00
- Discord (Casual) Office Hours (Thursday) @ 14:00-15:00



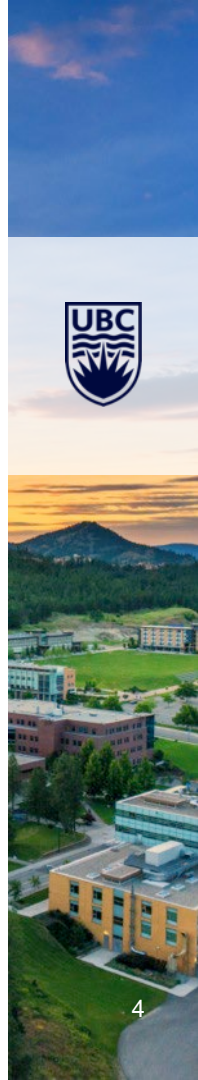
# Readings

Required:

Impossibility of Distributed Consensus with One Faulty Process

Recommended:

Two General's Problem

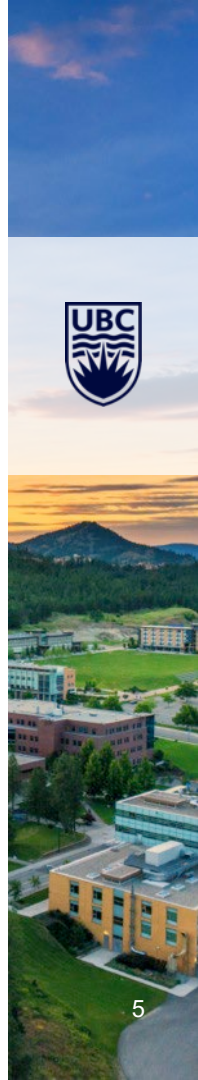


# Questions?

Questions about the class?

Questions about the previous lecture?

Funny stories to share?



# Today’s Failure

# Types of Failures

Robert Vitillo's Blog (How distributed systems fail)


Single point of failure

- Non-replicated configuration database
- HTTPS
  - Manually renewed certificate = nobody can connect

Slow networks

- How long should we wait?
- What happens if we *don't* wait?

Slow Processes

- TCP connection exhaustion

# Types of Failures

Demand spikes

- Failover = load spike
- Increased load = slow/no response
  - How long does a client wait?

Cascading failures

- Load spike
- Failed node resumes operation
- Working node collapses
- Repeat cycle

# Lesson Goals

# Achieving Consensus

Define *consensus*

Distributed systems limits on reaching consensus

Fischer-Lynch-Patterson (FLP) Theorem

Practical consensus

# What is consensus?

**Agreement** between distributed processes on shared state

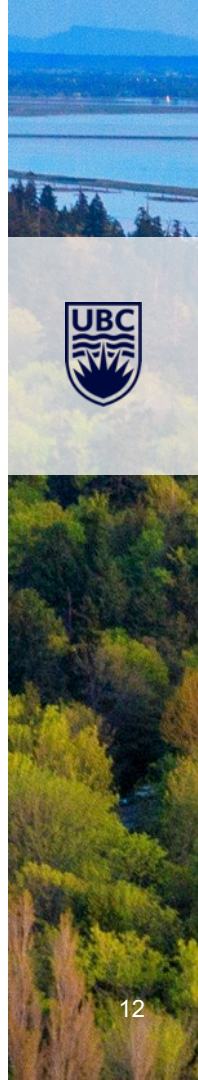
- ***Value***
- ***Action***
- ***Timestamp***
- ***Transaction outcome***

**Consensus** allows a system to be correct

# Challenges achieving consensus

Classic distributed systems problems

- Network behaviour
- No global clock
- Bad actors
  - Malicious
  - Broken behaviour
- Non-determinism



# System Properties

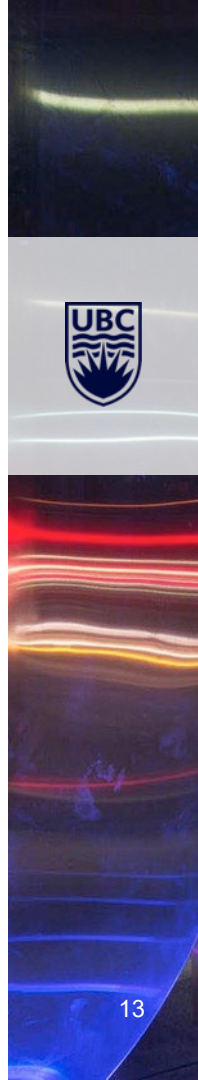
All non-faulty processes *eventually* determine the value

All processes determine the *same* value

The value is one that was proposed by *at least one* process

- Not externally provided

Question: how can (or if) we achieve this?



# System Model

Asynchronous

- Messages may be reordered
- Messages may be delayed
- Messages are *not* corrupted

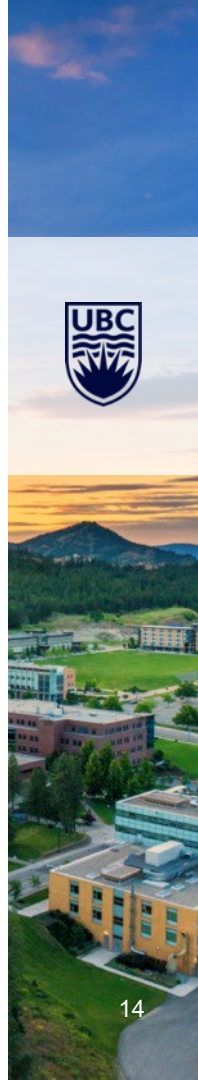
Bad behaviour

- At most *one* faulty process

Fail-stop model:

- Same as message delay

Note: the real world is *more complex*.



# Is consensus reachable?

Given our simple model:

- If not possible in *this* simple model, it won't work with:
  - Corrupted messages
  - Multiple bad actors
  - Byzantine failures
- If *possible*: try for more complex models

# Terminology

Admissible run: run of our model system

- At most one faulty process
- Messages *eventually* delivered

Deciding run = admissible run where *some* non-faulty processes reach a decision

Consensus = all admissible runs are deciding runs

- A “totally correct” consensus protocol

Decisions:

- Uni-valent – single value result
- Bi-valent – two *or more* values result (“non-deciding”)

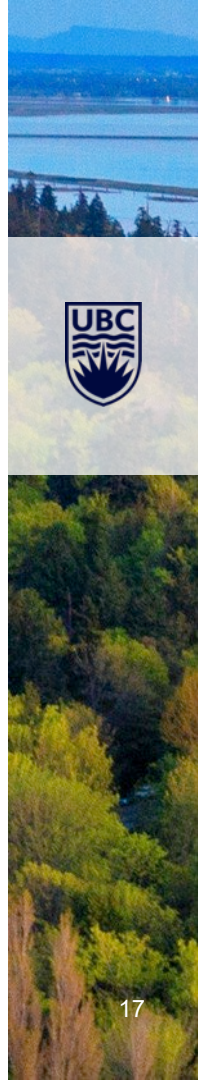
# FLP Theorem

In a system with one faulting process **there is no correct consensus protocol.**

This result is important because:

- True for *this* system

Question: Can we find a *different* (but viable) system that can have consensus?



# Proof Sketch

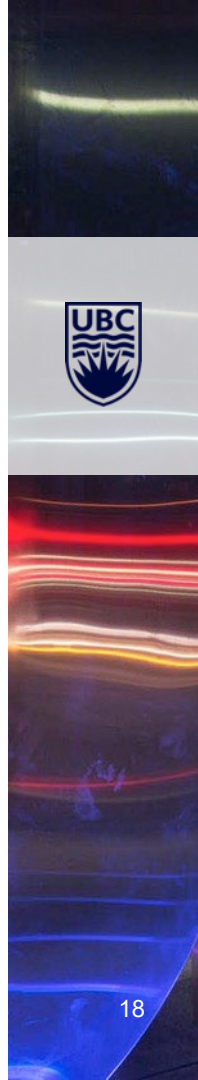
Model System:

- Asynchronous communication
- One faulty process
- Fail-stop model

Question 1: Can we identify a configuration and run that do not reach a deciding state?

Question 2: Can we find at least one admissible schedule that is not a deciding schedule?

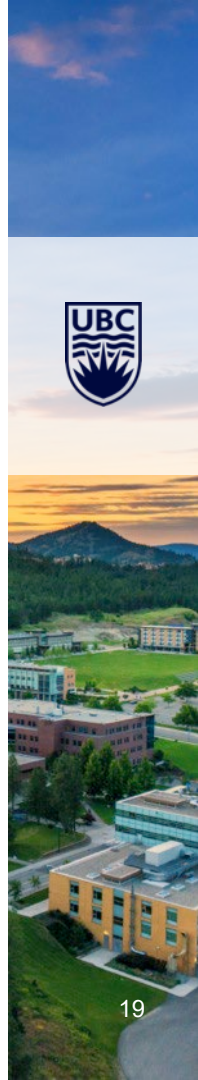
- Admissible schedule = “1 faulty process, all messages delivered”
- Deciding schedule = “system is in a bivalent configuration”



# Proof walk-through

Start

- Nodes make a binary decision (true/false)
- One faulty node is *possible*
- Messages may be delayed or reordered (but *not* lost)



# Lemma

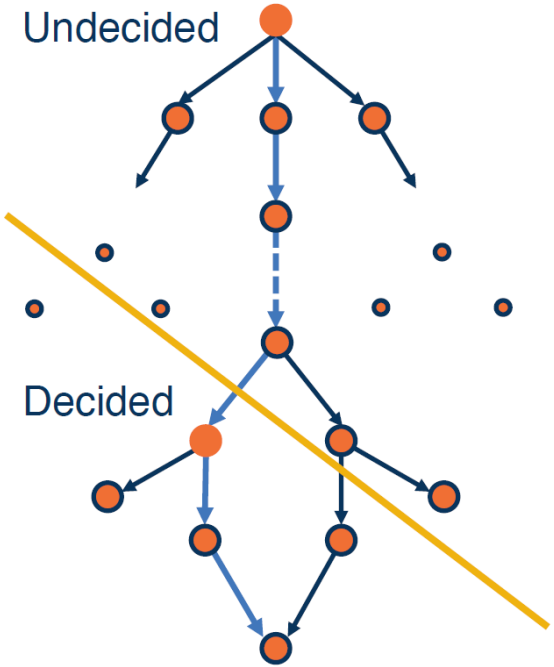
Assume:

- There is an *initial configuration*
- The *final decision is not pre-determined*
- Consensus is based upon *proposed* (not pre-determined) value
- The final decision depends on the event schedule

Conclusion:

- There *must exist* an initial bivalent configuration

# FLP Proof (Lemma)




There must be:

- A bivalent state
- A step (message)
- That message must change the system to a *univalent* system

Why?

To be deciding, the system must reach a single decision = univalent.

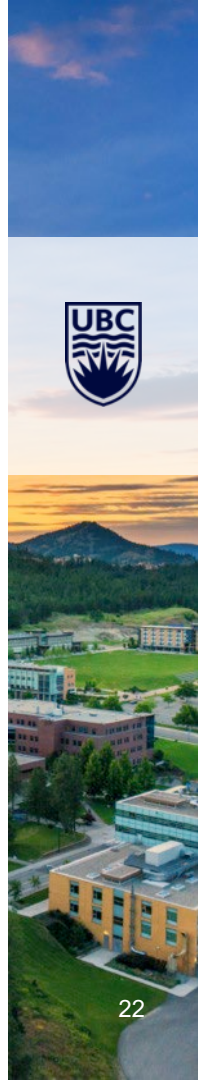
Could be the *last* message.

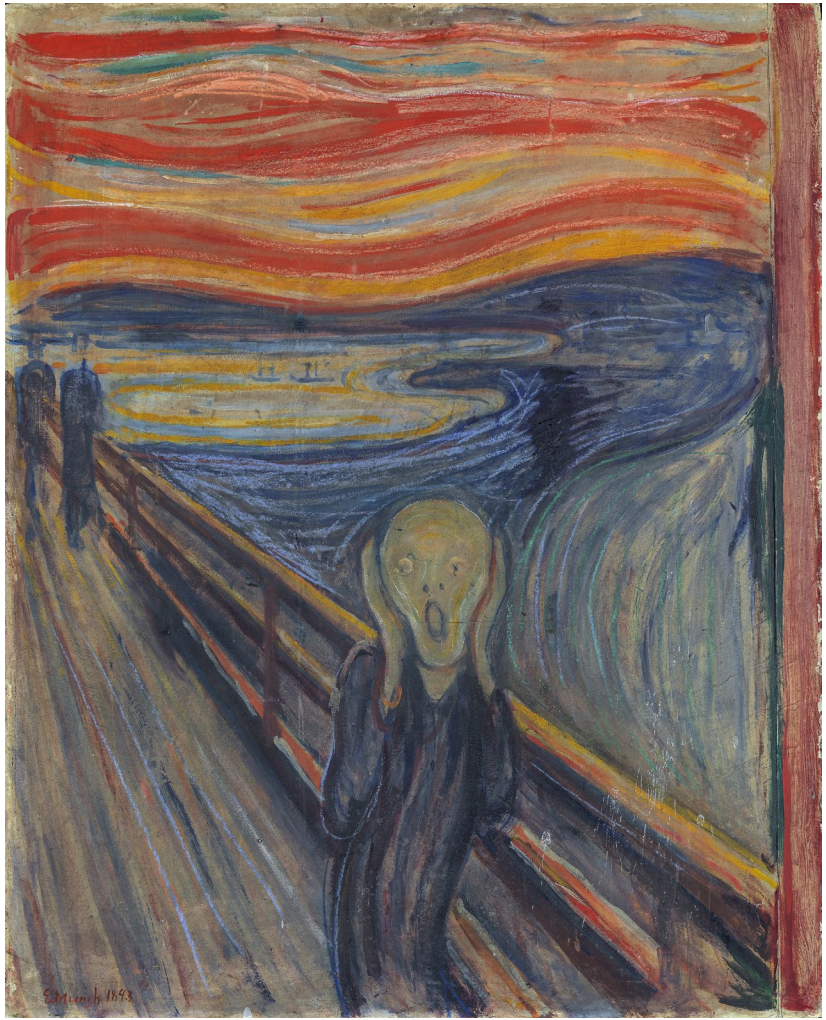
# Lemma (FLP Theorem)

Recall:

- System allows delayed/reordered messages
- Permits at least one schedule that never becomes univalent

This means an admissible non-deciding schedule can exist




E. Munch 1893

# FLP Theorem: Putting it all together

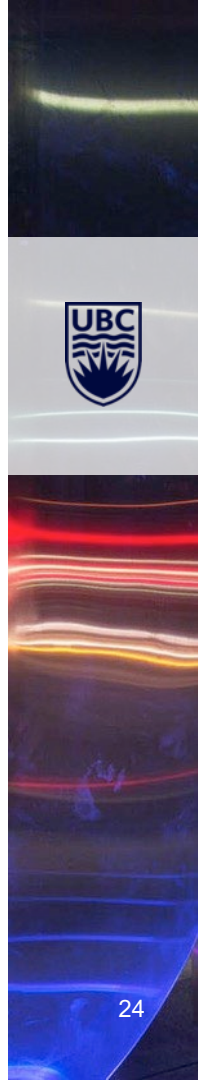
System consists of nodes that decide true/false

Lemma:

- There is an initial configuration
- A non-predetermined final decision
- Final decision depends on the schedule of events
  - Must have an initial bivalent configuration
- Some message must cause system to become univalent
- Messages can be delayed/reordered to avoid that specific message

Recall: one faulty node is *possible.* Combine this…

**An admissible non-deciding schedule can exist**.



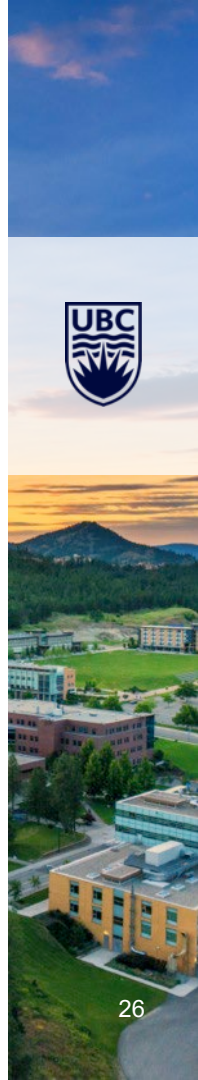
# So… does that mean we can't reach consensus?

Real world:

- Faults are *inevitable*
- Network delays *will happen*
- Can't make a *stronger* useful model

Question:

*How can we reach consensus?*



# Change the model

Change our assumptions

Change the system properties

Find situations where the protocol *does* decide.

- What are the conditions where it *will* provide consensus?

# Consensus is possible


2 Phase Commit (2PC)

3 Phase Commit (asynchronous 2PC)

Paxos

Raft/Viewstamped Replication

# Lesson Summary

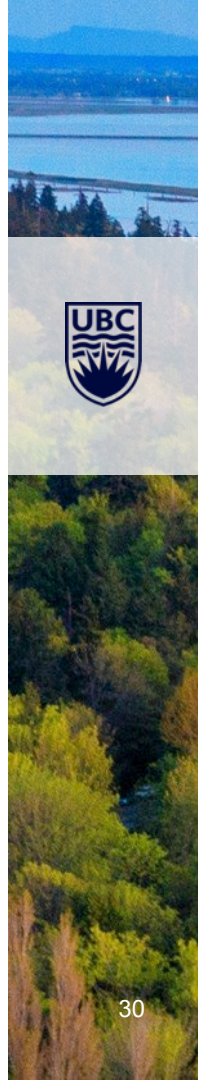
# Consensus: harder than we thought

FLP Theorem proves

- Given a simple model
  - One faulting process
  - Reordering/delaying messages
- Cannot guarantee consensus

We *need* consensus

- We can change the “simple model”
- We have more work to do!



# 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**

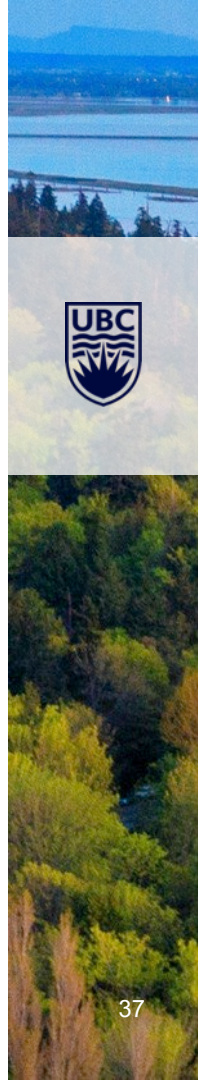
# Insert title here

Insert subtitle here

**Name, position**

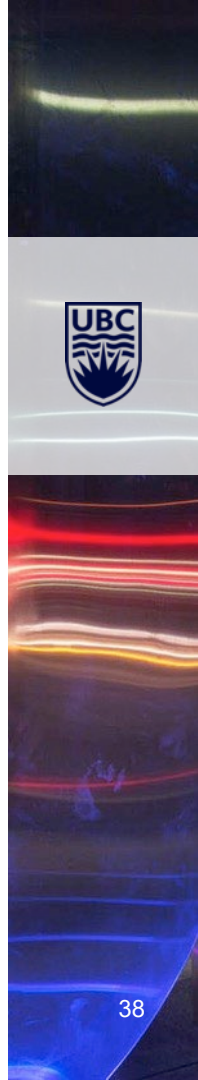
# Page title

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



# Page title

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



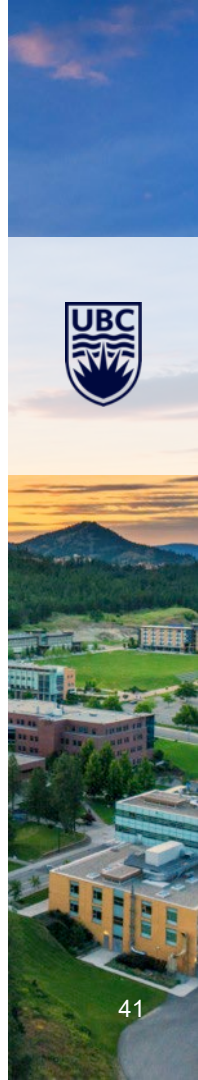
# Insert chapter title

# Page title

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

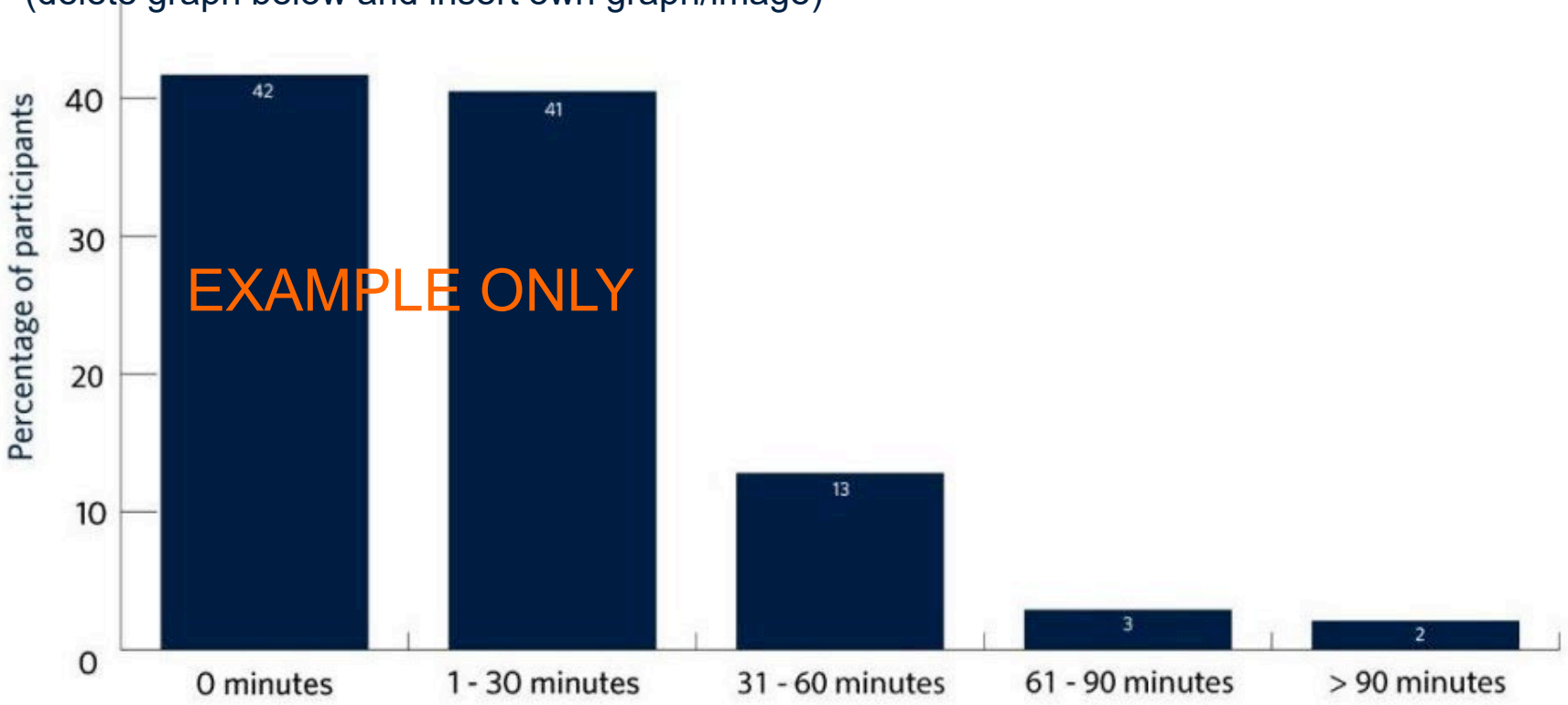
# Page title

- **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