SLICIFY: FAULT INJECTION TESTING FOR NETWORK PARTITIONS

Seba Khaleel*, **Sreeharsha Udayashankar***, and Samer Al-Kiswany

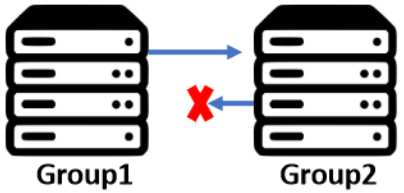
IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2024

* denotes equal contribution

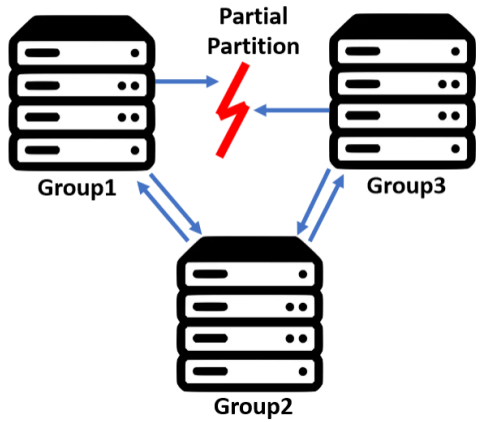
Introduction

- Network Partitions
  - Multiple Flavors [1, 2]
  - Common in production [3]
- Impact on distributed systems
  - 32 failures in 7 production systems [1]
  - Better testing is the solution [2]

a. Complete Partition



b. Simplex Partition



c. Partial Partition

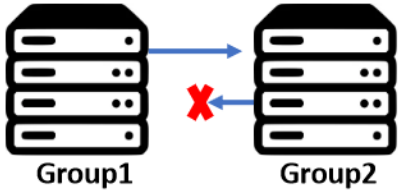
[1] Alquraan, Ahmed, et al. "An analysis of Network Partitioning failures in cloud systems." *USENIX OSDI,* 2018.
[2] Mohammed Alfatafta et al. "Toward a Generic Fault Tolerance Technique for Partial Network Partitioning." *USENIX OSDI,* 2020
[2] Turner, Daniel, et al. "On failure in managed enterprise networks." *HP Labs HPL-2012-101*, 2012.

UNIVERSITY OF WATERLOO

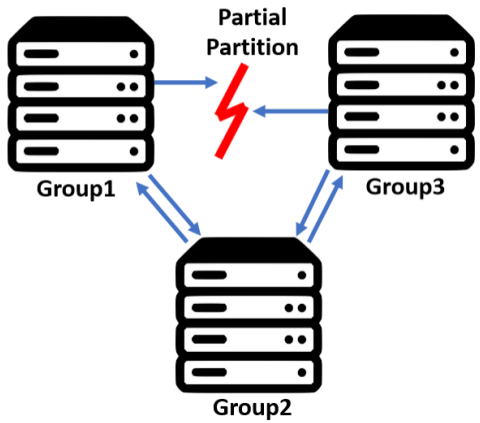
Introduction – Testing Challenges

- Testing distributed systems is complicated
  - Partition characteristics
    - Flavors
    - Can occur between any two components at any time
  - System characteristics
    - Size
    - Knowledge

a. Complete Partition



b. Simplex Partition



c. Partial Partition

Manual testing is insufficient!

UNIVERSITY OF WATERLOO

Slicify

- Slicify
  - Automated testing for partial and complete partitions
- Key Ideas
  - Reduce test space using component connectivity
  - Application-agnostic
- Capabilities
  - Reproduced previously seen failures in Spark and Kafka [3, 4]
  - Discovered new failures in Hazelcast, Flink and ActiveMQ

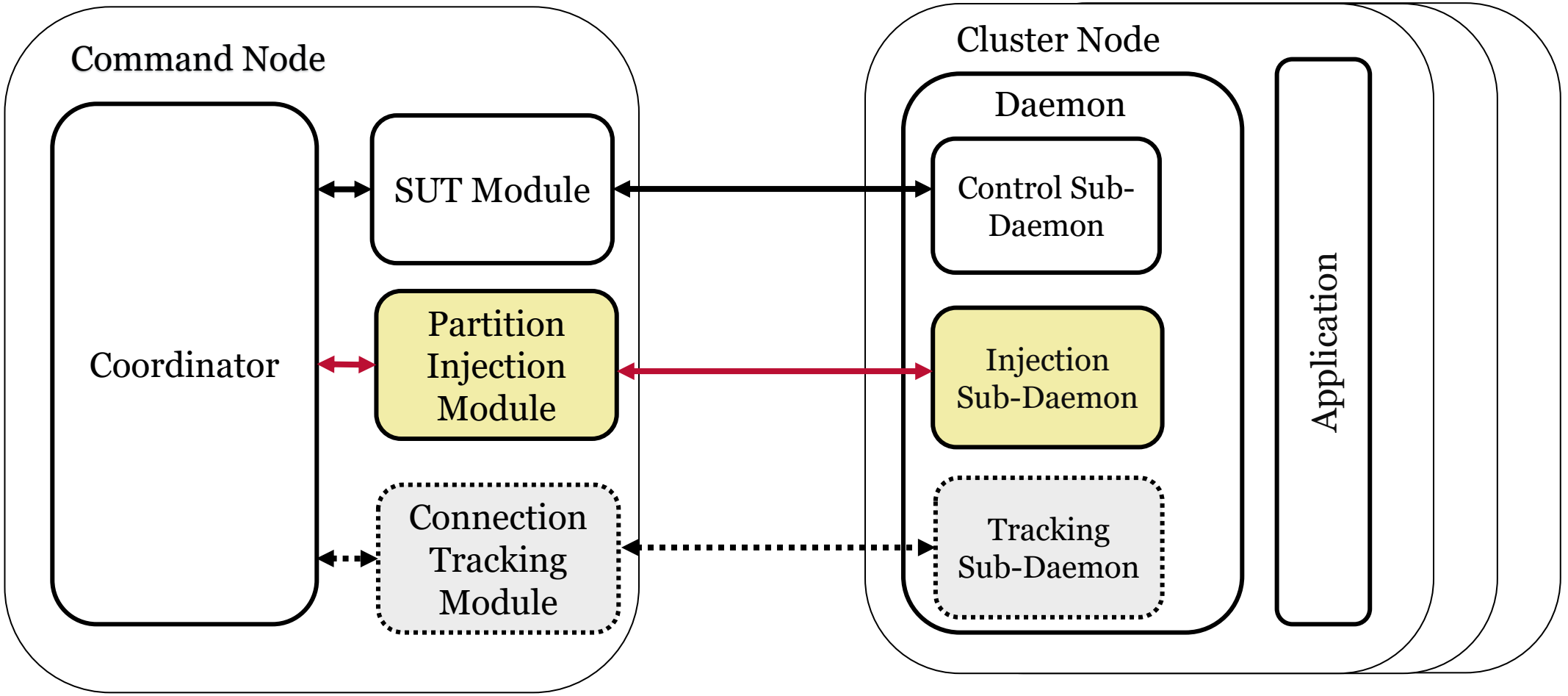
[3] Qunaibi, Sara, et al. “CASPR: Connectivity-Aware Scheduling for Partition Resilience." *IEEE SRDS*, 2023.
[4] Kafka-8702: Kafka leader election doesn’t happen when leader broker port is partitioned off the network. *https://issues.apache.org/jira/browse/KAFKA-8702*

UNIVERSITY OF WATERLOO

Outline

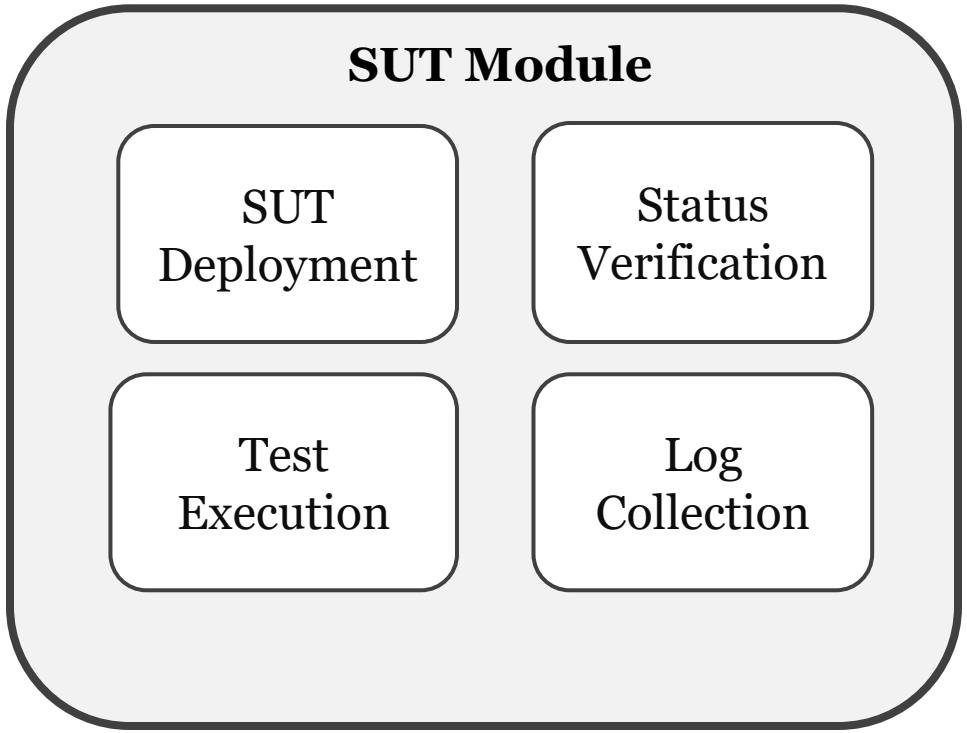
- Introduction
- Design
- Testing Procedure – Partial Partitions
- Capabilities
- Conclusion

Design


Command Node
Coordinator
SUT Module
Partition Injection Module
Connection Tracking Module
Cluster Node
Daemon
Control Sub-Daemon
Injection Sub-Daemon
Tracking Sub-Daemon
Application

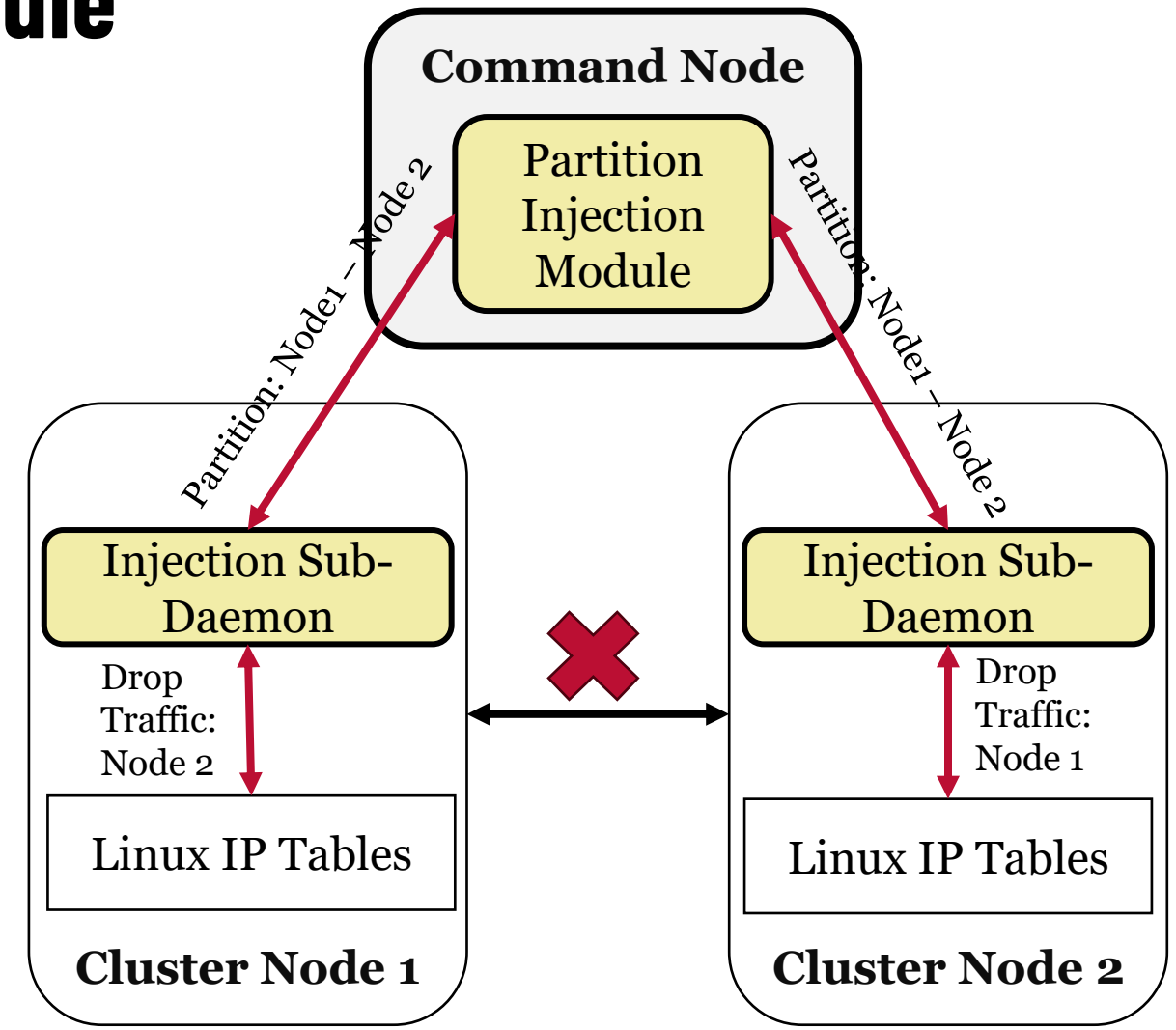
Design – SUT Module

- Handles SUT deployment and test execution
  - Communicates with control sub-daemon
  - Exposes APIs
    - Implemented by developers
    - 50 lines for Python client-server application



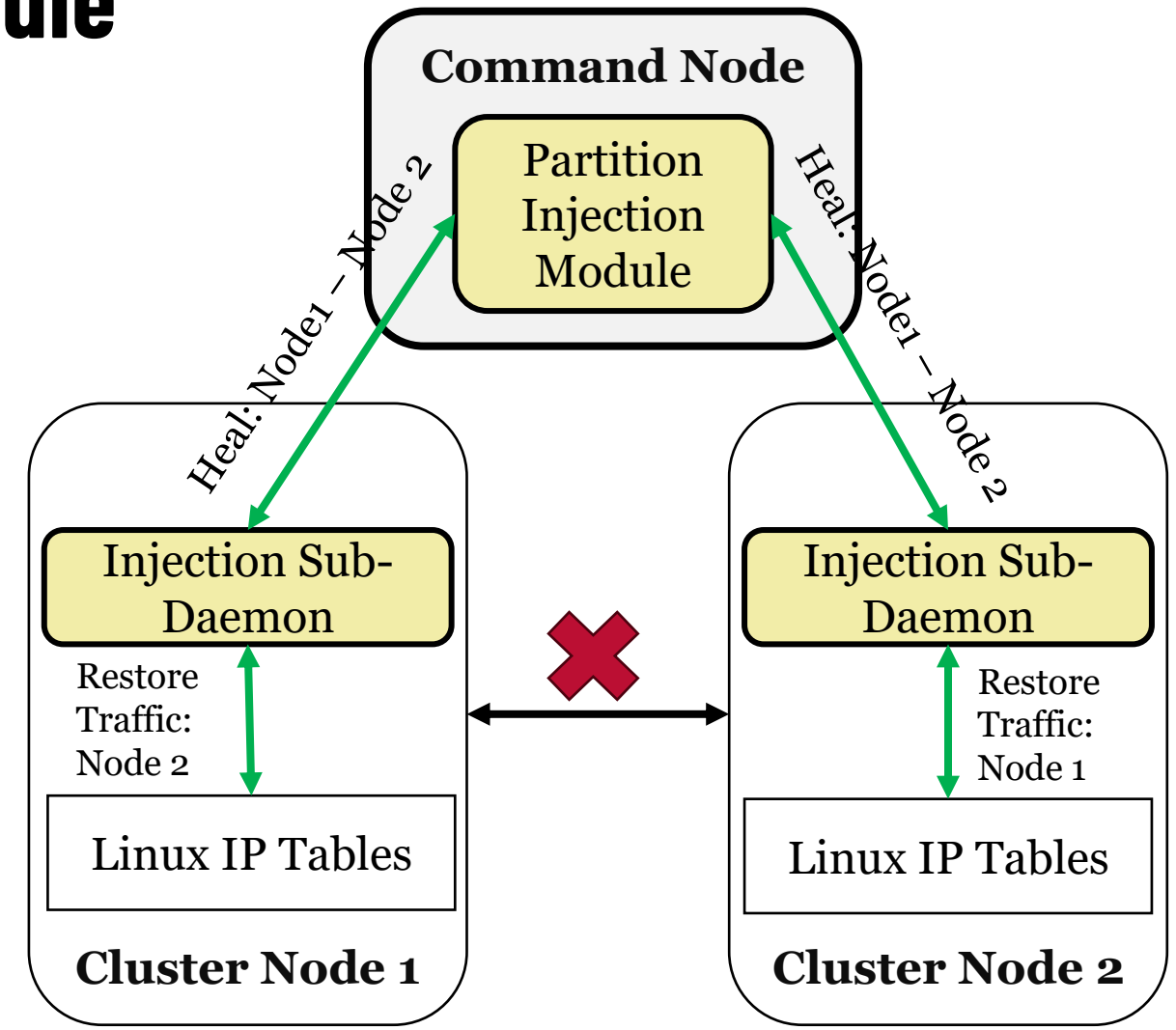
Design – Partition Injection Module

- Injects partitions between cluster nodes



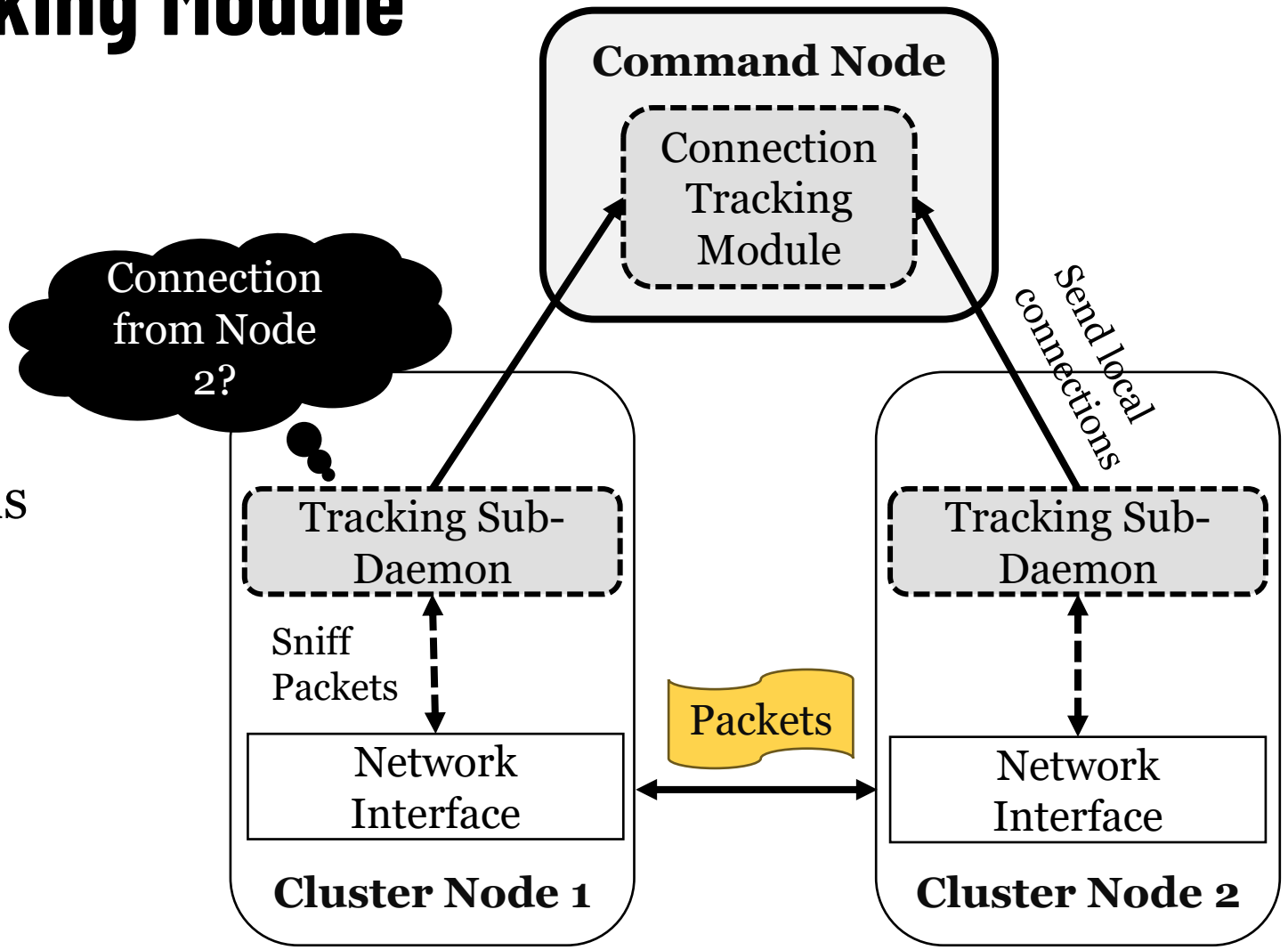
Design – Partition Injection Module

- Injects partitions between cluster nodes
- Heals partitions between cluster nodes



Design – Connection Tracking Module

- Responsible for tracking communication between cluster nodes
- Sub-daemons monitor connections on cluster nodes
- *Global Connection List:* Results combined on command node

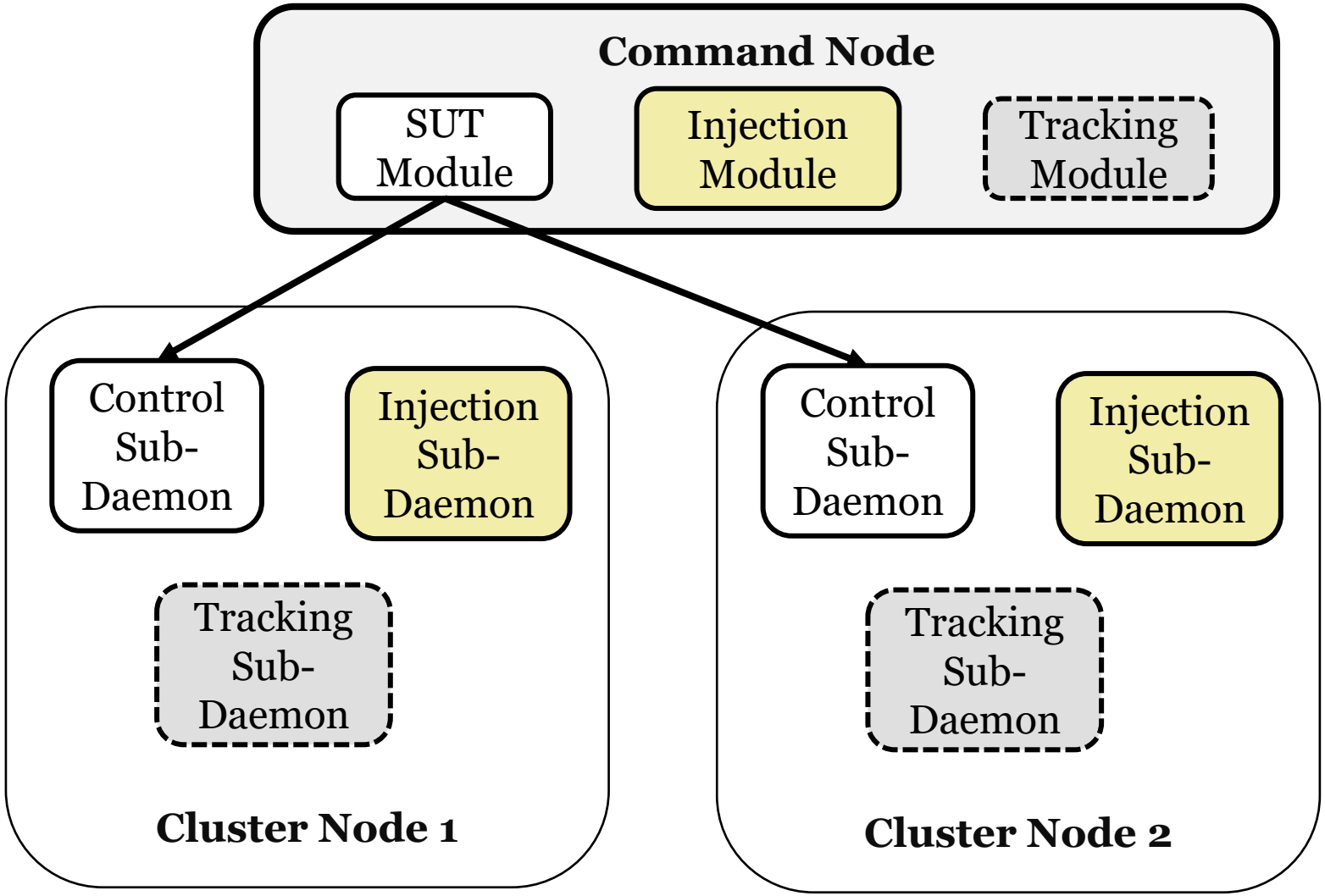


Outline

- Introduction
- Design
- Testing Procedure – Partial Partitions
- Capabilities
- Conclusion

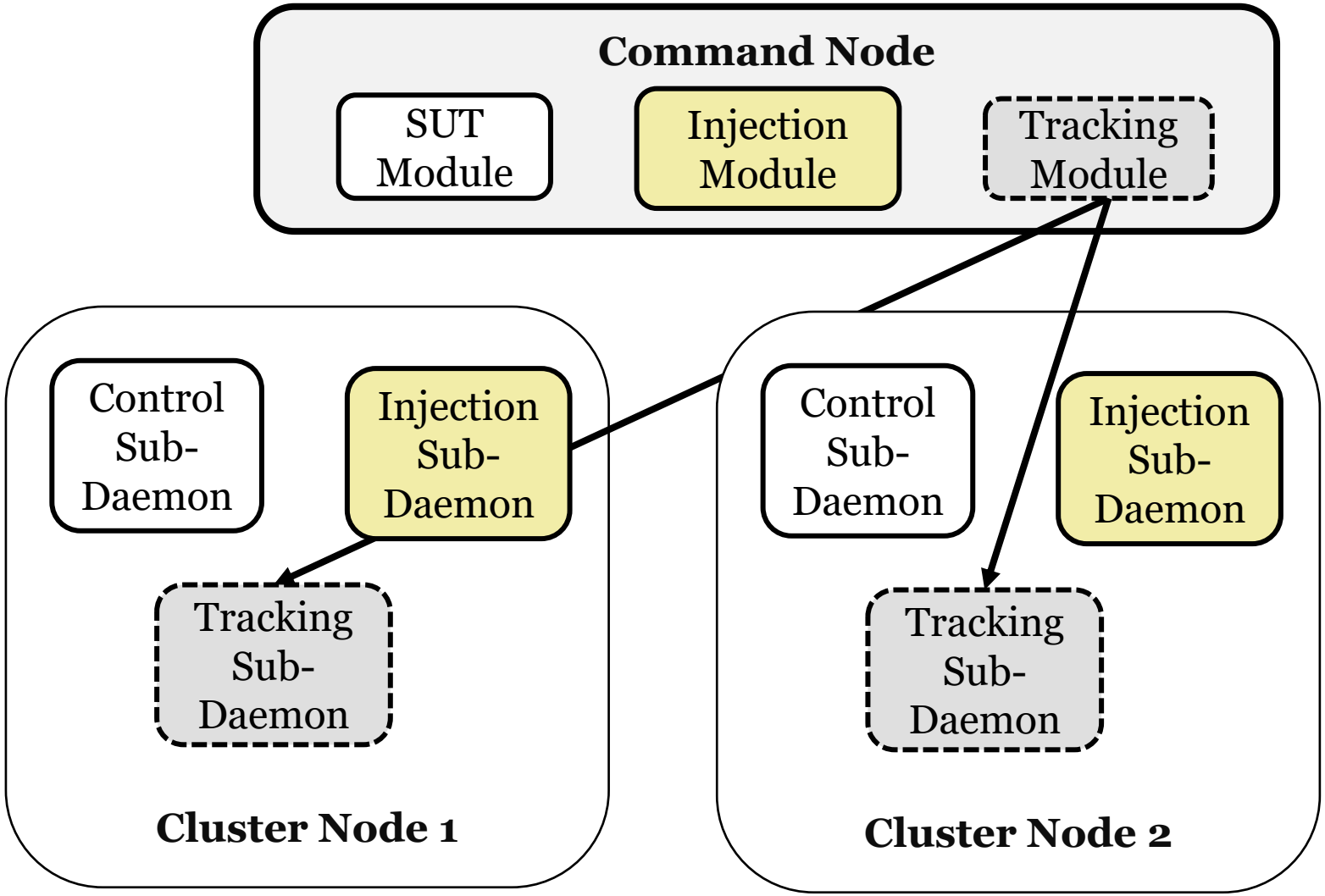
Test Preparation

- Run test to measure *fault-free execution time*



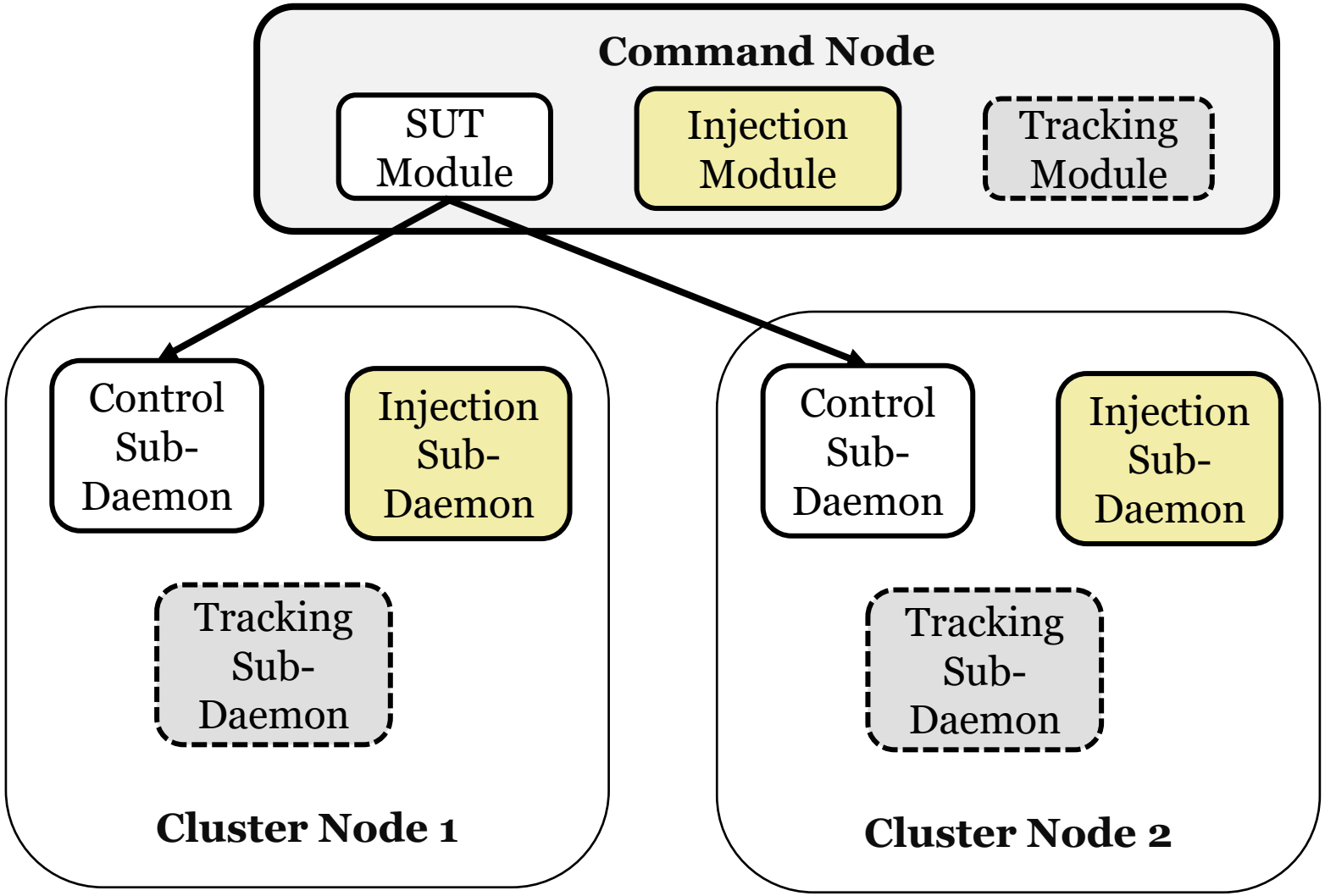
Test Preparation

- Run test to measure *fault-free execution time*
- Sync clocks and begin tracking



Test Preparation

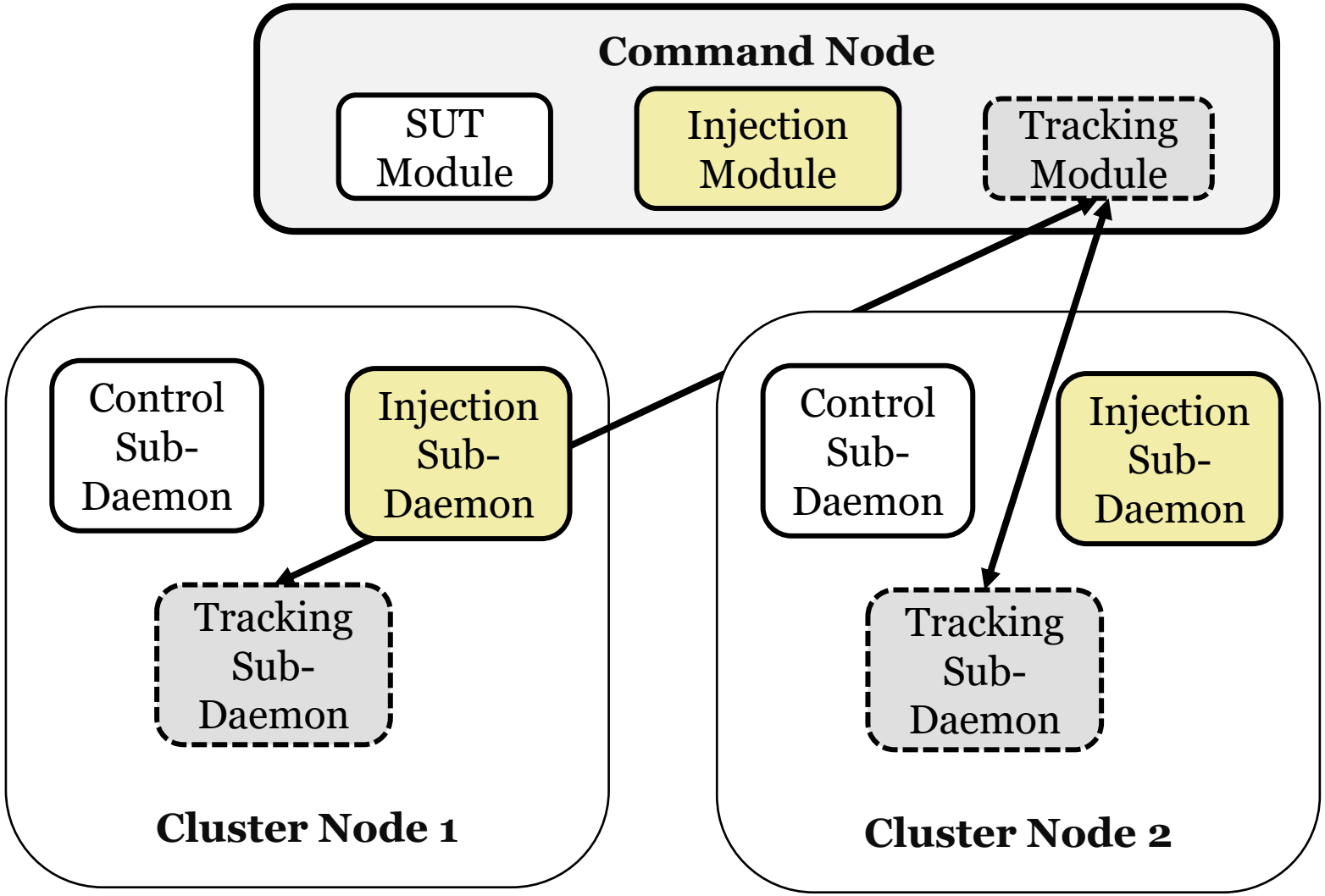
- Run test to measure *fault-free execution time*
- Sync clocks and begin tracking
- Run test with tracking



UNIVERSITY OF WATERLOO

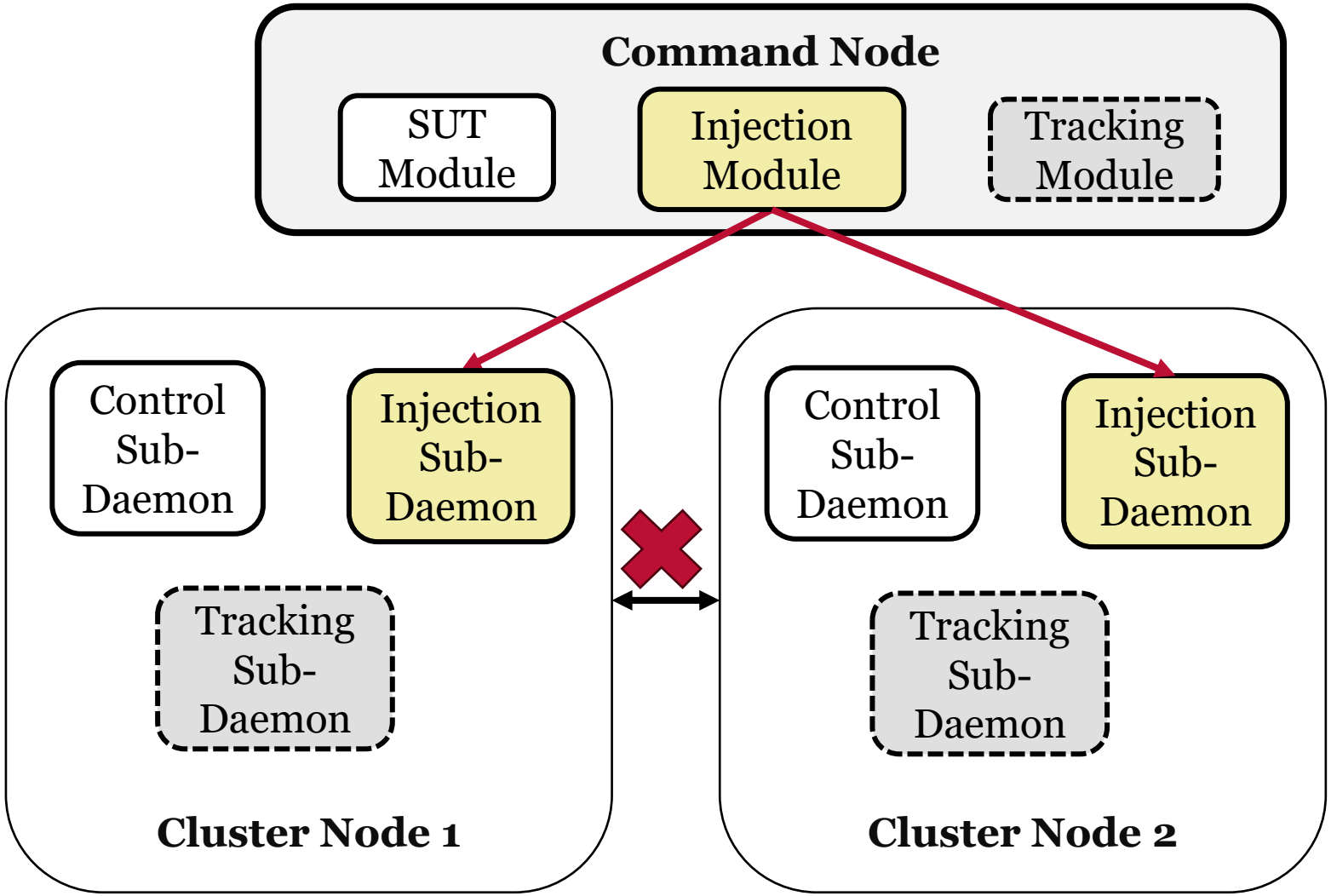
Test Preparation

- Run test to measure *fault-free execution time*
- Sync clocks and begin tracking
- Run test with tracking
- End tracking and build *global connection list*



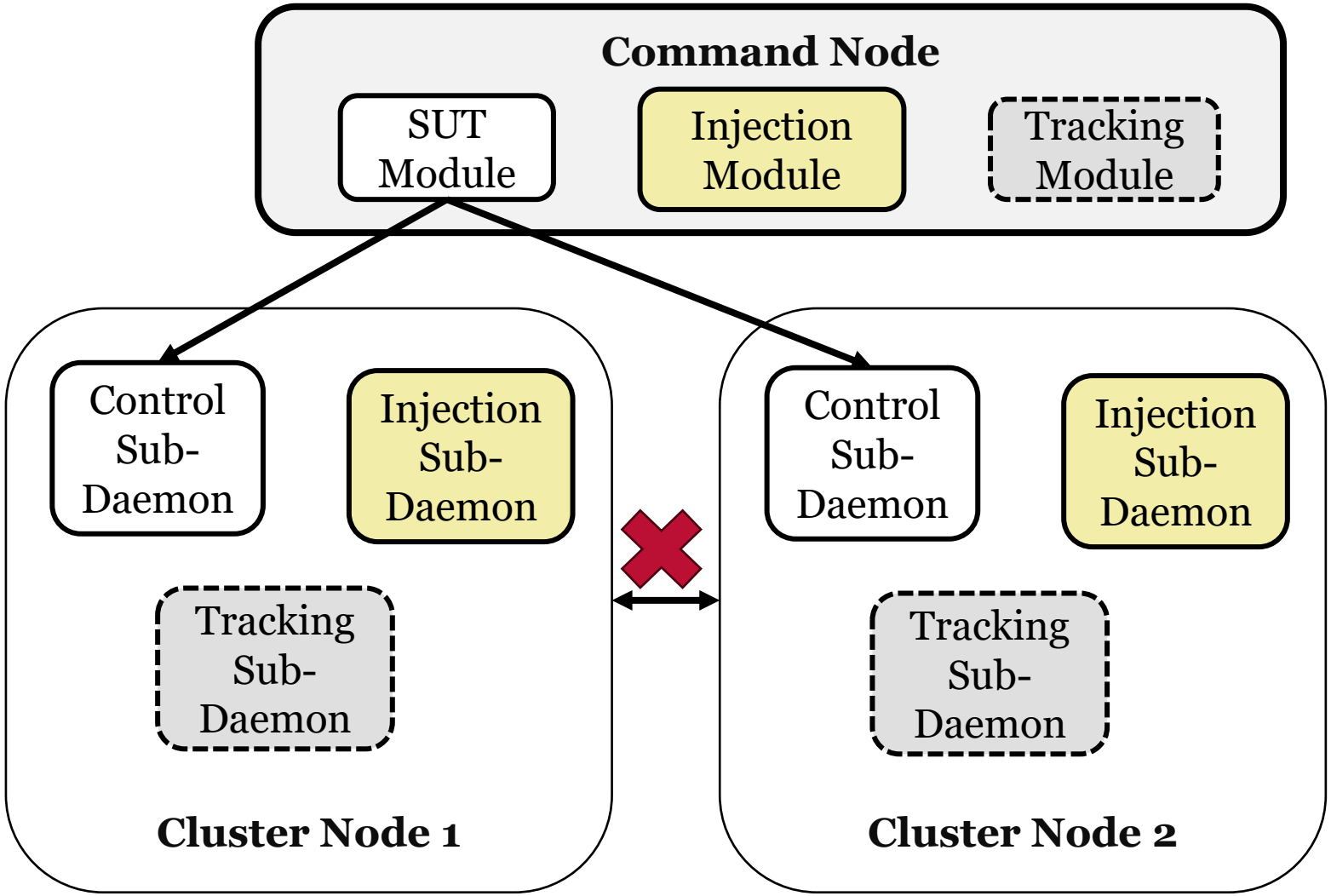
Testing

- For each connected node pair
  - Insert partition



Testing

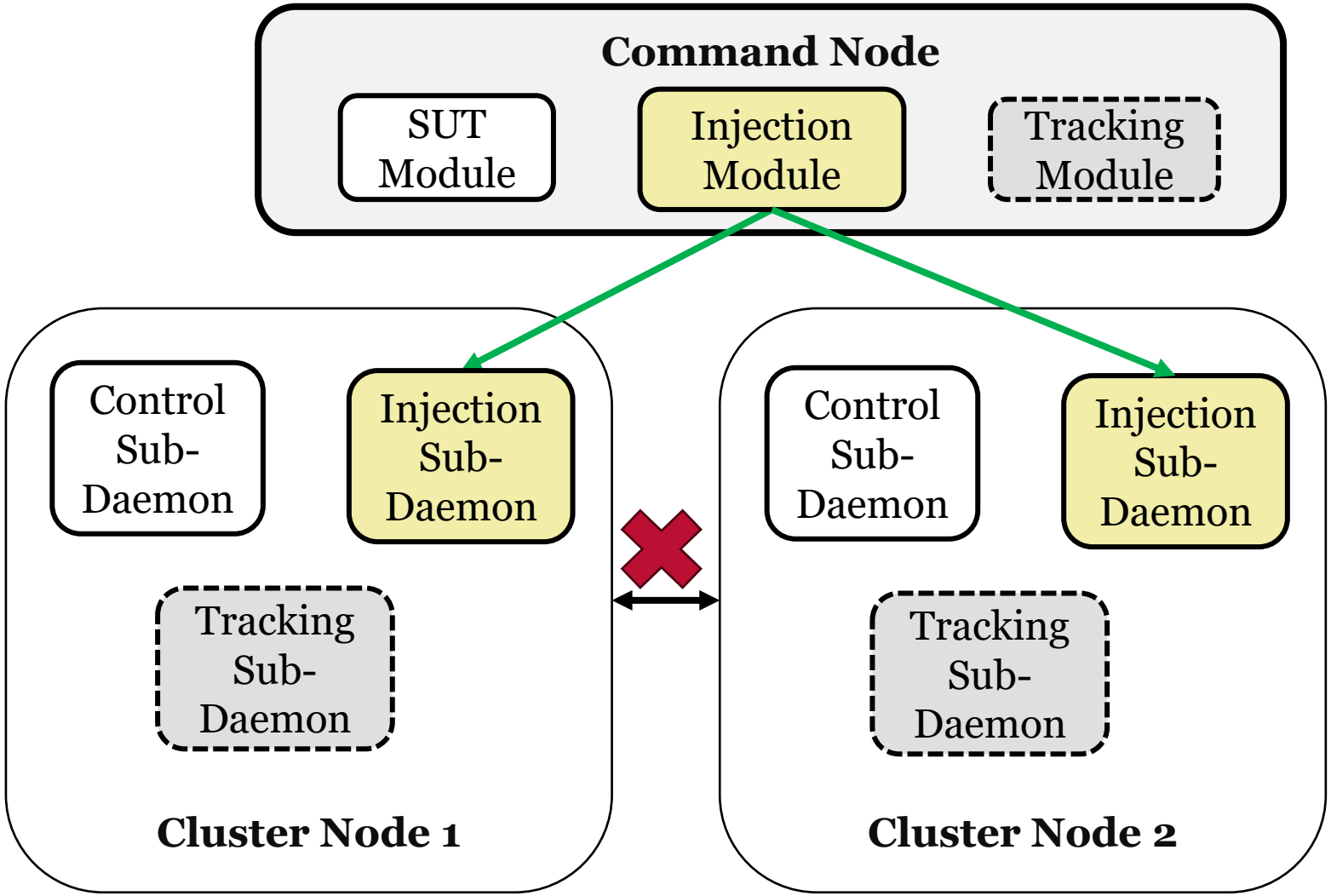
- For each connected node pair
  - Insert partition
  - Run test and record results



UNIVERSITY OF WATERLOO

Testing

- For each connected node pair
  - Insert partition
  - Run test and record results
  - Heal Partition
- Repeat for all node pairs

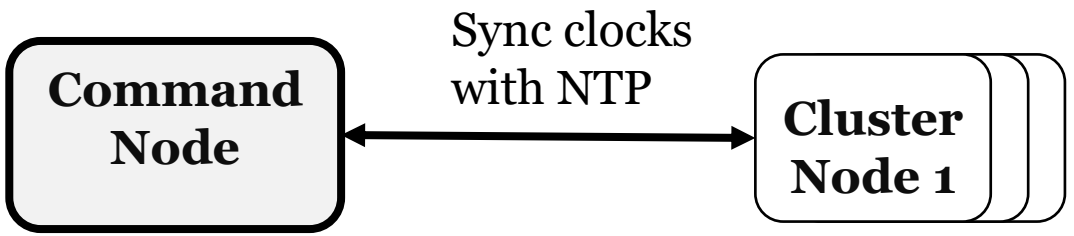


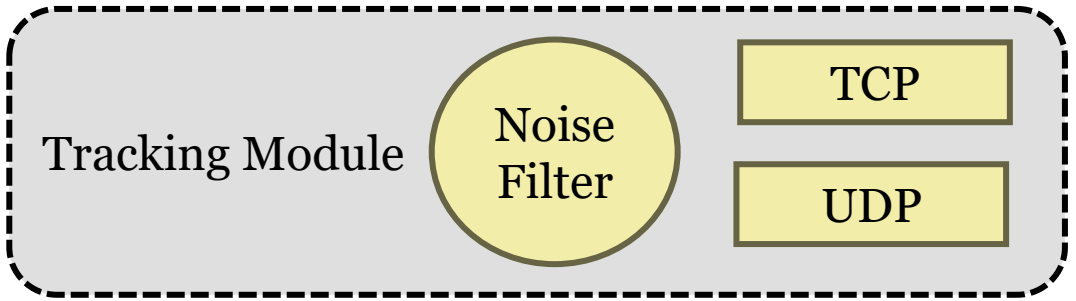
Challenges

- Communication Direction

- Tracking
  - Background noise
  - TCP and UDP differences
    - SYN / FIN only vs all packets

- Capture Performance
  - Distributed capture





UNIVERSITY OF WATERLOO

Outline

- Introduction
- Design
- Testing Procedure – Partial Partitions
- Capabilities
- Conclusion

UNIVERSITY OF WATERLOO

Capabilities

Tested Applications

- Verification:
  - Apache Spark
  - Apache Kafka
- Discovery:
  - Hazelcast – Maps and Locks
  - Apache Flink
  - Apache ActiveMQ

Apache Spark Configuration

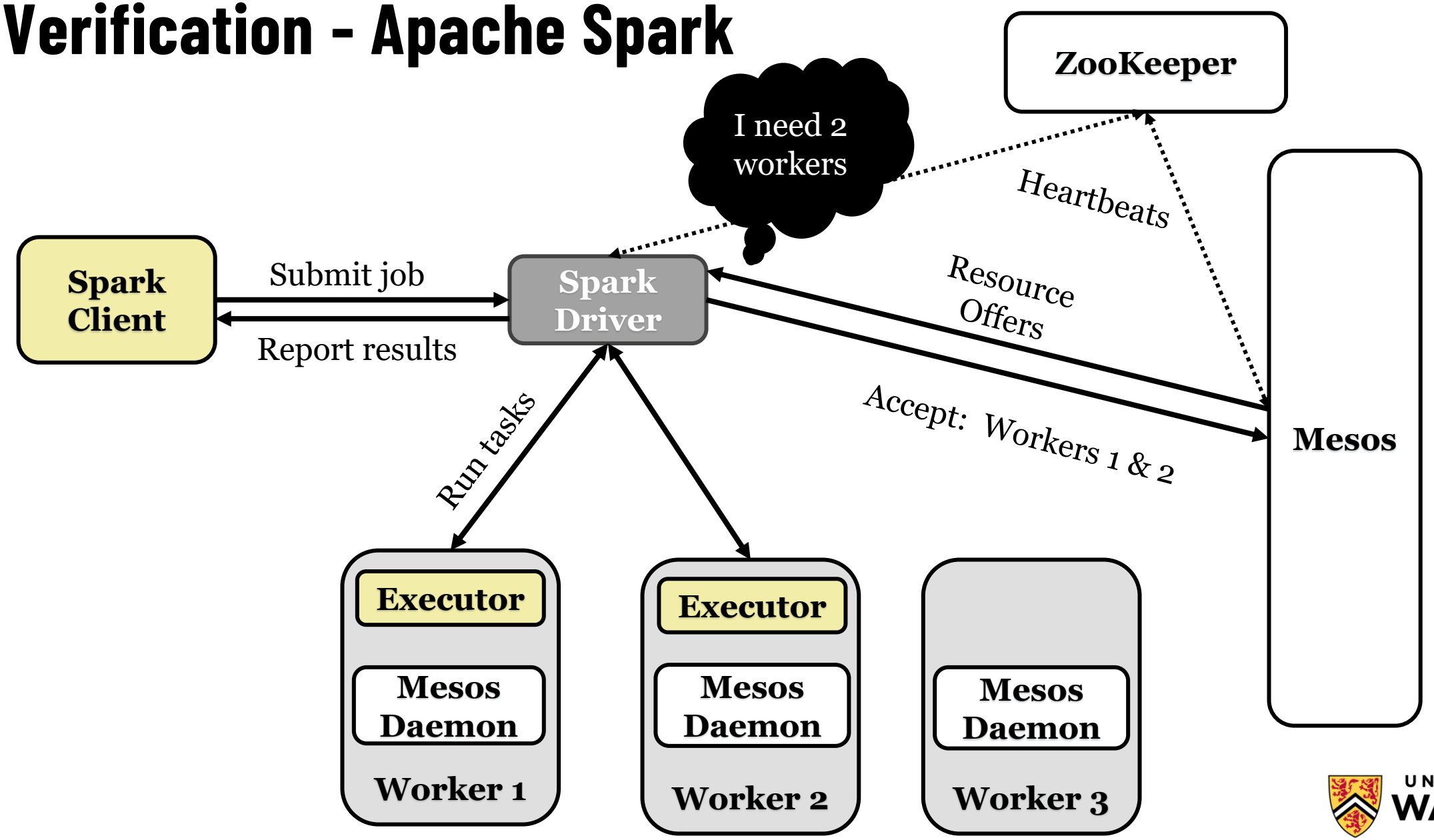
- **Workload:** WordCount
- **Version:** Spark + Mesos
- **Cluster:** 1 Driver and 3 Workers.
  - Slots per Worker: 1.
  - Tasks in Job: 2.

Apache Flink Configuration

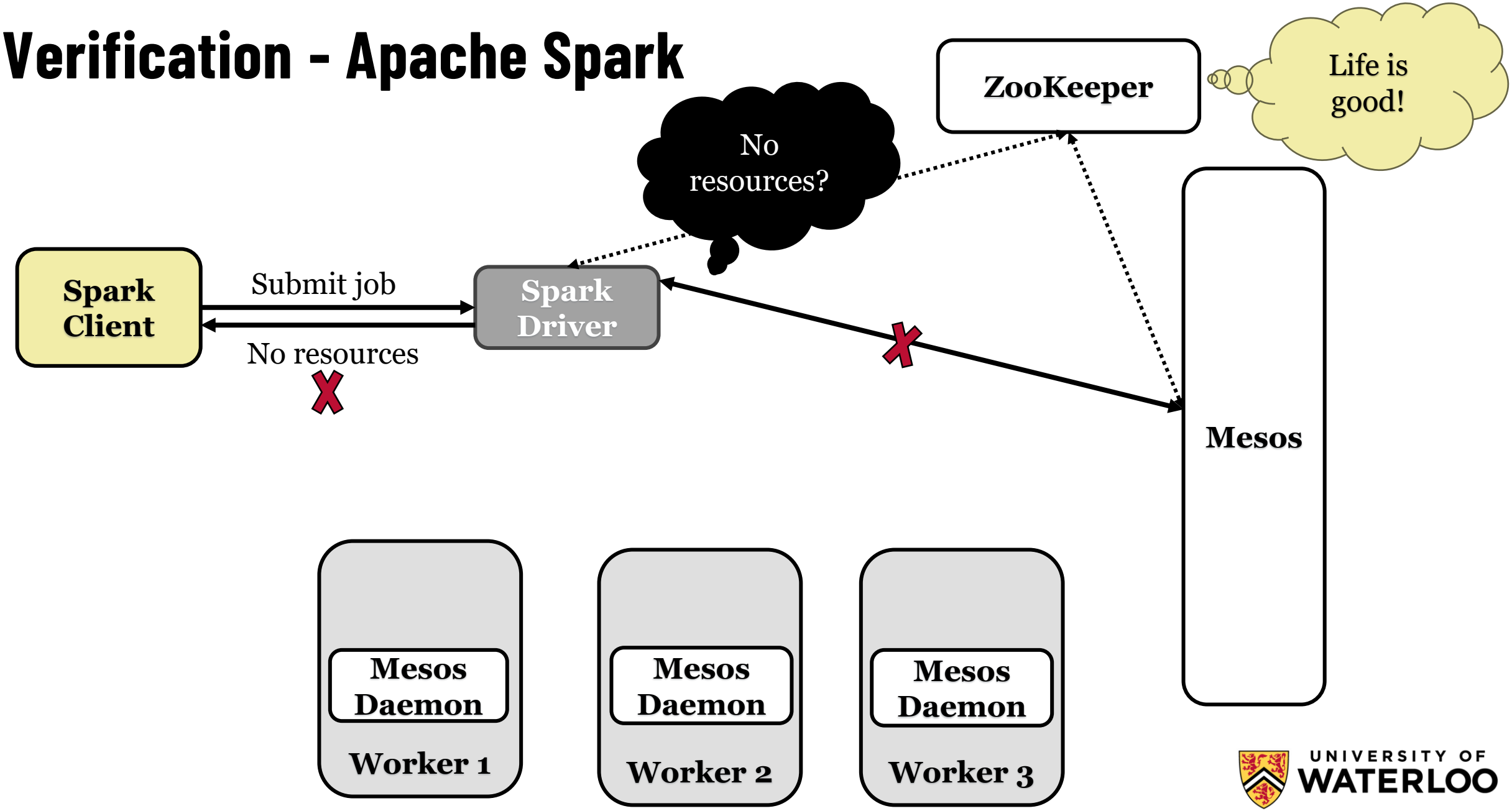
- **Workload:** WordCount
- **Version:** Flink with checkpointing and restart failover
- **Cluster:** 1 Job Manager and 5 Workers.
  - Slots per TaskManager: 1.
  - Level of parallelism: 3.
  - Number of restart attempts: 3.
  - Time between restarts: 10 seconds.

UNIVERSITY OF WATERLOO

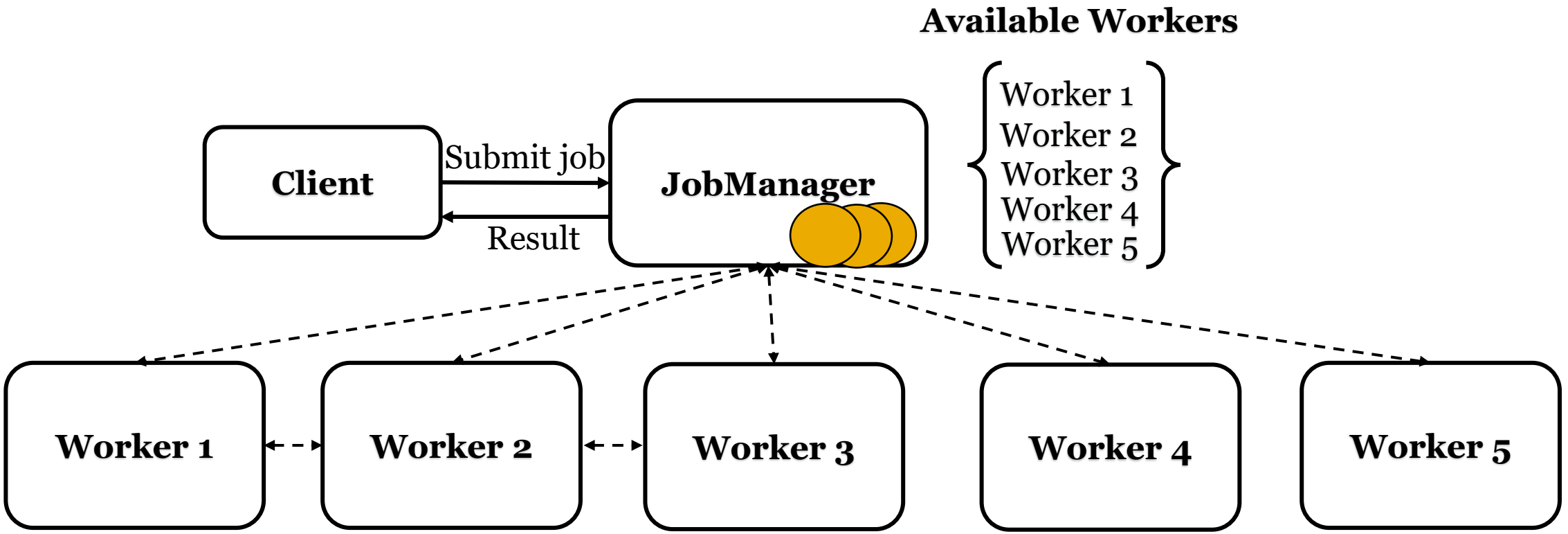
Verification - Apache Spark



Verification - Apache Spark

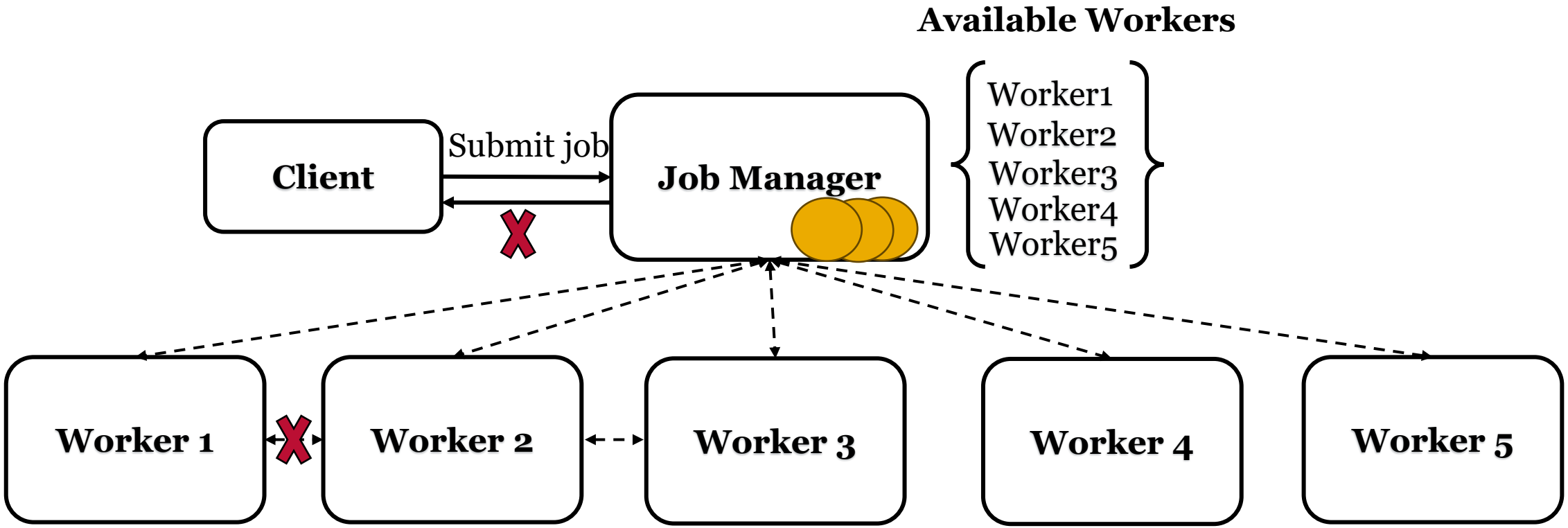


Discovery - Apache Flink



UNIVERSITY OF WATERLOO

Discovery - Apache Flink



Bug Report Filed with Apache [4]

[4] Flink-34006: Flink terminates the execution of an application when there is a network problem between TaskManagers. *https://issues.apache.org/jira/browse/FLINK-34006.*

UNIVERSITY OF WATERLOO

Summary

- Testing for network-partitions is complex
  - Manual testing is insufficient
- Slicify
  - Automated & application-agnostic
  - Uses novel connection-tracking mechanism to reduce test space
- Capabilities
  - Reproduced failures from previous studies in Apache Spark and Kafka
  - Found 3 new bugs in Hazelcast, Apache Flink and Apache ActiveMQ
- Code: https://github.com/UWASL/slicify

UNIVERSITY OF WATERLOO

UNIVERSITY OF
WATERLOO