Comp8670 Final Project Proposal

Reliable Pub-Sub Messaging System with Redis Streams

Team Members:
Aaron Barnoff (103177919, barnoffa@uwindsor.ca)
Matthew Belanger (105016182, belangeq@uwindsor.ca)

April 4, 2025

1 Problem Statement

Implementing fault-tolerant publish-subscribe messaging systems is crucial for ensuring reliable communication in distributed systems.

- **Motivation**: Reliable messaging systems are essential for real-time data streaming, critical event handling, and robust backend communication, particularly where message loss or delay can significantly affect system performance.
- **Background context**: Publish-subscribe systems are foundational to modern distributed architectures, prominently used in computer networks and applications requiring real-time messaging. To implement them effectively we need to understand where they are used and where possible errors can occur.
- **Real-world applicability**: Highly applicable in scenarios like IoT telemetry data, financial transaction processing, and social media updates where data reliability and timeliness are crucial. The publish-subscribe pattern can be used for any two machines that desire to communicate. Implementing this pattern reliably has serious real world consequences do to the wide applicability of this pattern.

2 Objectives

Key goals of the project:

- Build a fault-tolerant publish-subscribe messaging system using Redis Streams based on the work of Pallickara et al. [1].
- Compare the system against RabbitMQ and Kafka regarding message processing speed, reliability, and fault tolerance.

3 Methodology

Approach details:

- Tools/technologies: Redis Streams, RabbitMQ, Kafka.
- System architecture/components: Publisher-subscriber model implemented in Redis Streams, RabbitMQ, and Kafka setups individually, with integrated failure scenarios to test fault tolerance.
- Datasets: Synthetic messaging data of varying payload sizes and frequencies
- Steps of implementation/experimentation: Deploy individual messaging systems simulate network conditions and node failures, conduct benchmarking tests, and collect performance metrics.

Diagrams will illustrate system architecture and testing setup.

4 Expected Outcomes

Deliverables by project end:

- A functional, fault-tolerant publish-subscribe messaging system using Redis Streams.
- Comprehensive benchmarking data comparing Redis Streams, RabbitMQ, and Kafka.
- Performance metrics: throughput, latency, reliability under simulated faults.

5 Roles and Responsibilities

- Matthew: Module setup, Redis Streams implementation.
- Aaron: RabbitMQ and Kafka implementation, performance benchmarking.

References

[1] S. Pallickara, H. Bulut, and G. Fox, "Fault-tolerant reliable delivery of messages in distributed publish/subscribe systems," in *Proceedings of the Fourth International Conference on Autonomic Computing (ICAC 2007)*, pp. 19–19, IEEE Computer Society, July 2007.