# Lightweight Distributed Service

Yesheng Ma, Zucheng Wu, Yikai Zou

Shanghai Jiao Tong University

May 31, 2017

# Overview

1. Introduction to distributed services
2. Raft Overview
3. Our Implementation

# What are distributed services?

Some examples are:

- Distributed key-value database.
- Distributed lock service.
- Replicated state machine(RSM).

# Why distributed services?

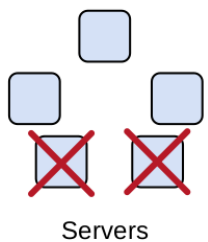
- To make services highly available.
- To make the system fault tolerant.
- The system can recover after machine failure.

# How to build distributed services?

*Consensus!* A system is available if the majority is alive.

Early works date back to Leslie Lamport's 1989 paper on *Paxos*. But Paxos is famous for its difficulty to understand.

*Raft* comes to our rescue, which is easier to understand with an explicit leader.



# What is Raft

Raft has two key components:

1. Leader election:
   - A server is elected leader if most peers vote for it.
   - Only a server with an up-to-date log can become a leader.
2. Log replication:
   - Clients send commands to leader.
   - Leader replicates its logs to other followers.

# Our Implementation: Overview

Our goal:

- KISS: keep it simple, stupid, and easy to reason about.
- Lightweight: about 4000 locs.
- Cross platform.
- Extensible.

# Our Implementation: Language

Written with the Go programming language:

- A statically-typed PL with useful concurrent features.
- Multiple platforms: from PC to Android devices.
- Especially suitable to build network applications like Raft.

# Our Implementation: Network Simulation

Real world network is not reliable.

We simulates three kinds of RPC failures:

- Packet loss.
- Long delays of packet transmission.
- Reordering of packets.

# Our Implementation: Synchronization

Say no to locks! Only one mutex lock used.

Instead use channels for synchronizations between threads:

1. The Raft entity receives timeouts/replies and redirects it to a respective channel.
2. A working thread takes the message from a channel and begins to work on it.
3. Once the working thread is done, send result back to the Raft entity.

# Our Implementation: Performance

Current version of our system can handle about 10 transactions per second.

We have optimized the system to do batch log replication, i.e. replicate multiple log entries in a single remote procedure call.

# Future Work

- Interface to real world network environment.
- Network topology for scalability.
- Network aware election timeout setting.

# Demo

# The End