Relatório da Atividade 4: Raft

Isabelle Ferreira de Oliveira *CES-27 - Engenharia da Computação 2020*
Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos, Brasil
isabelle.ferreira3000@gmail.com

Resumo

Esse relatório documenta um trabalho com o protocolo Raft para consenso distribuído. Não será trabalhado com *log replication*.

Index Terms

Raft, Golang, sistema distribuído

I. IMPLEMENTAÇÃO

A. Ao receber um AppendEntry (heartbeat)

Subsection text here.

1) Follower:
2) Candidate:
3) Leader:

B. Ao receber um RequestVote

Subsubsection text here.

1) Follower:
2) Candidate:
3) Leader:

C. Ao receber um Reply

Subsubsection text here.

1) Follower:
2) Candidate:
3) Leader:

II. RESULTADOS E CONCLUSÕES

Na Figura 1, foi apresentado uma captura de tela de um funcionamento com todas instâncias funcionando.

- Server 2 vira *cadidate*
- Server 2 recebe voto de 1
- Server 3 vira *candidate*
- Server 1 nega voto para 3 (já votou em 2)
- Server 2 nega voto para 3 (já é candidate)
- Server 2 recebe voto de 4 e 5 (e vira leader)
- Server 3 não consegue maioria dos votos
- Server 3 faz *Step Down*
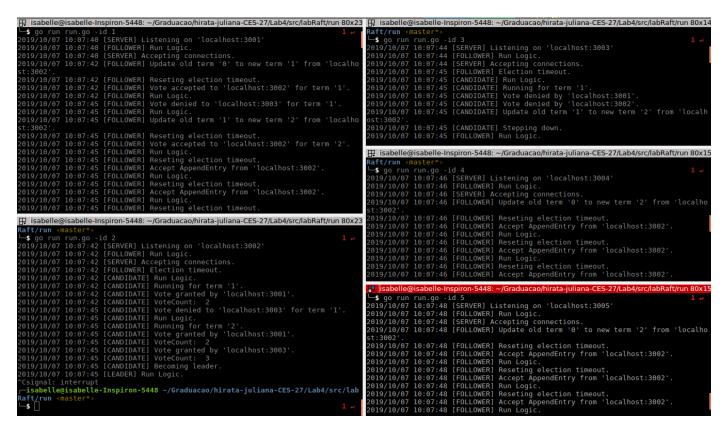- Server 1, 3, 4 e 5 continuam como *follower* indefinidamente, pois 2 é *leader*

Figura 1. Funcionamento normal para os 5 processos.

Na Figura 2, foi apresentado uma captura de tela de um funcionamento com falha do líder (com outro assumindo a liderança) e retomada do servidor falho (com ele reconhecendo a liderança do outro).

- Server 3 (*leader*) falha
- Server 2 vira *candidate*
- Server 2 recebe voto de 1, 4 e 5 (e vira *leader*)
- Server 3 é retomado e recebe *AppendEntry* de 2 (que é o novo *leader*) *leader*

Figura 2. Funcionamento após a falha no *leader*, seguido de retomada do processo.

Figura 3. Funcionamento após a falha no *leader*.

Figura 4. Funcionamento após a falha no próximo *leader*.

.

Figura 5. Funcionamento após a falha no terceiro *leader*. Note que o sistema não consegue mais decidir por um novo *leader*.

.