DTU Informatics — 02228 - Fault tolerant systems

FINAL REPORT
RAFT CONSENSUS ALGORITHM

Joachim Kirkegaard Friis s093256
Anders Nielsen s103457

November 11, 2014

References

[1] Michael Bailey, Evan Cooke, Farnam Jahanian, Yunjing Xu, and Manish Karir. A survey of botnet technology and defenses. In *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*, CATCH '09, pages 299–304, Washington, DC, USA, 2009. IEEE Computer Society.

[2] John Black, Martin Cochran, and Trevor Highland. A study of the md5 attacks: Insights and improvements. In Matthew Robshaw, editor, *Fast Software Encryption*, volume 4047 of *Lecture Notes in Computer Science*, pages 262–277. Springer Berlin Heidelberg, 2006.

[3] David Dagon, Guofei Gu, and ChristopherP. Lee. A taxonomy of botnet structures. In Wenke Lee, Cliff Wang, and David Dagon, editors, *Botnet Detection*, volume 36 of *Advances in Information Security*, pages 143–164. Springer US, 2008.

[4] William Rex. The zombies all around you. `http://www.meannnasty.com/zombies-around/`, april 2014.

1 Preface

In this section we will describe and motivate our project idea. We will start out by presenting the Two Generals problem in which two processes try to reach consensus about when to attack the enemy. Then we will relate this problem to modern distributed systems, in which consensus must be achievable in presence of faulty processes.

1.1 Two Generals Problem

1.2 Complexity and understandability of current solutions

2 Analysis

In this section we will come back to the Two Generals problem and further analyse and realize that reaching consensus without a leader/authority, who can issue commands, is impossible. Then we will present Raft as a solution to this problem and compare it to other solutions and discuss why we have chosen it.

2.1 Impossibility of the Two Generals Problem

2.2 Raft as a solution

3 Requirements

The tool we are implementing should serve as an illustration tool, to show how the Raft algorithm works for given scenarios of faulty processes in a distributed system.

a. The tool must illustrate a scenario of a given set of processes in a distributed system, in which a leader is elected.

b. The user must then be able to disconnect the leader.
    a) If a leader is disconnected a new leader must be elected automatically.

c. The user must be able to add new processes to the system.

d. The user should be able to input parameters to vary the scenario, such as the number of processes, and the heart rate of the system.
    a) The system should be able to have the parameters changed at run-time.

e. The tool must be an implementation of the Raft algorithm.
    a) The Raft implementation should have the following properties: Safety, availability, timing independence, and that commands complete as soon as a majority has responded.

f. The tool could be further extended with a visualisation of the given distributed system.

4 Design

In this section we will relate to our requirements of the tool, and from this design the implementation.

5 Implementation

6 Tests

a. Simulation and validation of different scenarios described in the requirements.
   a) Leader election:
      i. The system must choose a new leader if the current leader is disconnected.
      ii. The system must be able to choose a leader if the system is partitioned.
   b) Timing and availability
      i. The system must not depend on timing of new elections, thus not producing incorrect results because of unknown events.

7 Conclusion