Christopher Jones

Achieving distributed consensus with Paxos

Part II Project

Trinity Hall

March 1, 2018

Proforma

Name:	**Christopher Jones**
College:	**Trinity Hall**
Project Title:	**Achieving distributed consensus with Paxos**
Examination:	**Part II Project**
Word Count:	**...** [1]
Project Originator:	Christopher Jones
Supervisor:	Dr Richard Mortier

Original Aims of the Project

...

Work Completed

...

Special Difficulties

...

[1] This word was computed by `detex diss.tex | tr -cd '0-9A-Za-z \n' | wc -w`

Declaration

I, [Name] of [College], being a candidate for Part II of the Computer Science Tripos [or the Diploma in Computer Science], hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed [signature]

Date [date]

Contents

List of Figures

Chapter 1

Introduction

- Quick sentence or two introducing the project as a whole.

1.1 Background

- Describe at a high-level the problem of distributed consensus
- Describe the environment we're operating in (again at a high level)
- Paxos as a means of getting around these problems.
- Why Paxos is important
- Where Paxos is used in real large-scale systems (e.g. Google Chubby)
- How Paxos provides us with primitives to use state machine replication techniques and atomic broadcast etc.

1.2 Aims

- Describe the purpose of this project - to develop an implementation of Multi-Paxos.
- How the project will be evaluated - such as metrics and the comparison to popular existing library (Libpaxos), using a network simulator.

Chapter 2

Preparation

Short introduction of what we're going to cover in preparation. All the stuff that was considered before development began.

Do I need to re-produce proofs here?

2.1 Theotrical background

2.1.1 Aims of Consensus

- A more formal treatment of distributed consensus?
- The formal definition of what we mean by consensus / agreement

2.1.2 Assumptions of the environment

- State the assumptions of the networks in which we operate and their failure modes.
- State the assumptions about how nodes can fail in Paxos.
- The guarantees we can rely on when sending messages, in paticular non-Byzantine faults for all of the above

2.1.3 Single-decree Paxos

- Start with a description of how the algorithm works in the case of proposing a single value.

- Use the parlance in Lamport's Paxos Made Simple - proposers, learners and leaders?
- Discuss the phases and invariants of the algorithm in this simple case to introduce how these systems work
- Include an explanation and a timing diagram

2.1.4 Multi Paxos

- Move now to the parlance used in Paxos Made Moderately Complex and also that with which the project uses.
- Explain how we do not need to perform a whole instance of Paxos for each proposed value in a sequence if we assume the leader does not crash with high regularity.
- Introduce ballots as an indirection between proposals and the voting protocol.
- Describe the roles in Multi Paxos under this scheme. High level overview of clients, replicas, leaders and acceptors (note we can co-locate them if we wish to return to the same node types we have in Paxos above)
- Describe the failures we can tolerate of each of these nodes
- Communication patterns between nodes.

2.1.5 Replicating a State Machine

- Discuss the technique used to take an state machine and replicate it with Multi-Paxos

2.2 Requirements

2.2.1 Application

- Describe the application to build (strongly consistent replicated KV-store). Link back to state machine replication.
- Explain we need to convert RSM transitions into commands clients can issue to replicas.

- A table containing the commands, their parameters, their response types and their semantics (i.e. updating a key that does not exists returns a failure) for the KV-store.
- Describe the need to separate out the messaging subsystem from the Paxos module.
- A diagram showing how the appliction sits atop a Paxos module that sits atop a messaging module that is connected to the network.

2.2.2 System requirements

-

2.2.3 Evaluation strategy

-

2.3 Software Engineering

Brief introduction and describe the structure of this section here.

2.3.1 Choice of language

- Explain why OCaml was used, what advantages it offers.

2.3.2 Methodology

- Discuss the software development methodology (something along the lines of Agile I imagine)
- Describe the testing strategy. Talk about OUnit for unit testing functions as simple units and then more thorough and complete tests for correct funtionality at the cluster-level - (these can either be Mininet scripts or something else entirely...)

2.3.3 Libraries

- Mention OPAM package manager and how it is used for installing / managing libraries and dependencies

- Discuss the licenses of these packages (professional practice)
- Give particular explanation to Capn'Proto and Lwt as these are the most interesting and pervasive libraries used in the project. Save technical details for implementation.
- Remark briefly about using Core as standard library replacement, Yojson for some serialization, OUnit for unit testing, some lib for command line parsing.

2.3.4 Tools

- Vim with Merlin for type inference whilst editing.
- Git for version control, Github (and Google Drive) for backups.
- Mininet (with VirtualBox) for evaluation.

2.4 Summary

Chapter 3

Implementation

3.1 High level structure of program

3.2 Data structures

Include a description of the data structures used in the implementation. Talk about them first so we can discuss them in the messaging section and the protocol implementation sections.

3.2.1 Key value store

...

3.2.2 Proposals

...

3.2.3 Ballots

...

3.3 Messages

- Talk about the Cap'n Proto schema format. Include snippets of the schema file and how it relates to messages we are required to send (could easily fit the schema into a two page appendix as well).
- Include mention of serialization of certain parts of the application as JSON.

- Now talk about implementation of the Cap'n Proto server. How functions-as-values were used to allow callbacks to be performed by the server.
- Recall / reference the message semantics required from the preparation chapter. Explain how we carefully handle exceptions to ensure we get the desired semantics of messaging from Cap'n Proto.

3.4 Clients and replicas - better title?

3.4.1 Clients

- Talk about how clients operate outside the system
- How they send messages to replicas.
- Include .mli interface for replicas, the record (with mutable fields) for storing replicas.

3.4.2 Replicas

- Leading on from clients, explain how messages can be re-ordered / lost / delayed from clients and how broadcasts from replicas allow progress in the event of replica failure.
- Describe the two-fold operation of replicas. They receive and perform de-duplication of broadcast client messages. They then go on to propose to commit commands to given slots. They also handle when leaders reject their proposals so they can be re-proposed later. They also maintain the application state (in this case the replicated key-value store.)
- Interesting points to discuss include looking at a snippet of the de-duplication function and discussing it. Also looking at how mutexes are used to hold locks on the sets of commands, proposals, decisions etc.
- Include a diagram of the three different sets of commands, proposals and decisions and how each moves from one set to another in different sets of circumstances.

3.5 Synod protocol

3.5.1 Quorums

- Describe how the quorum system was implemented. Show the .mli interface as a snippet and describe how this can be used to achieve majority quorums.
- Show briefly how the majority checking function was implemented.

3.5.2 Acceptors

- Discuss the operation of acceptors - how they form the fault tolerant memory of Paxos.
- Discuss the functions (along with snippets) by which acceptors adopt and accept ballots.

3.5.3 Leaders

- Describe how the operation of leaders is split into two sub-processes - scouts and commanders.
- Operational description of how scouts and commanders communicate concurrently. Diagram displaying how there is a message queue that these sub-processes use to send "virtual" messages.
- Snippets of the signatures and structs used to construct these sub-processes.

Scouts

- Describe how scouts secure ballots with acceptors. Relate this to operation of acceptors above.
- State how scouts enqueue virtual adopted messages after securing adoption from quorum of acceptors.
- Describe how pre-emption can occur when a commander is waiting for adoption.

Commanders

- Explain how commanders attempt to commit their set of proposals.
- Give detail on the pmax and "arrow" function as used in the paper.
- Describe how pre-emption can occur when a commander is waiting for acceptance.

3.6 Summary

Chapter 4

Evaluation

4.1 Experimental Setup

4.1.1 Mininet

- Describe the simulation framework used with Mininet - how the simulation scripts are structured.
- Describe the system by which a simulation is performed - include a diagram of how simulation scripts and executable binaries are transferred to the simulation, simulations are performed and resulting log files are returned to the host-machine for analysis and plotting.
- Describe the system by which tracing is performed.

4.1.2 Libpaxos

- Short description of Libpaxos library.
- Explain how application was written / any modifications necessary to run the same sample application.

4.2 Steady state behaviour

- Characterise the steady state behaviour of the system - capturing the latency and throughput in a given configuration. Plot against each other to show characteristics.
- Comparison in the steady state behaviour between the project implementation and Libpaxos.

4.3 Configuration sizes

- Treatment of experiments that describe how latency and throughput vary as a function of the number of different nodes in the configuration varies.
- Include description of the exact experiment performed and how confidence intervals were calculated.
- Include graphs for: latency against different cluster sizes and throughput against different cluster sizes. Include Libpaxos version of this as well.

4.4 Failure traces

- Experiment pertaining to crashing a replica and observing system behaviour over time.
- Describe how data was averaged - including the sample mean and EWMA.
- Graph of latency and throughput as a function of time with a specific point a replica is crashed noted.
- Similar graph with a replica restoration.
- Graphs / on the same plot with behaviour of Libpaxos.

4.5 Quorum sizes

- ...
- ...

4.6 Summary

Chapter 5

Conclusion

...

Bibliography

Appendix A

Projct Proposal