Project proposal: Implementing the HotStuff consensus algorithm

Marc Harvey-Hill

12 10 2022

1 Description

This project aims to implement the HotStuff algorithm as outlined by Yin et al. in 2019 [**?**]. HotStuff is a Byzantine fault-tolerant algorithm that allows state to be replicated across nodes under a partially synchronous model. This has applications in blockchain, and can be used as the foundation for the development of cryptocurrencies and decentralised applications.
A leader node will drive consensus, ensuring that non-faulty replicas run identical commands in the same order so that state is consistent across replicas. The algorithm has a two-phase process for reaching consensus on a given proposal - the first forms a quorum certificate to guarantee a unique proposal and the second guarantees that the next leader will be able to convince replicas to vote for a safe proposal. Additionally the algorithm has a three-phase process for a view-change, which involves selecting a new leader, allowing it to collect information, and make a proposal to replicas.
The project will be implemented in OCaml on top of existing RPC and cryptographic libraries such as AsyncRPC and tezos-crypto. It can be divided into three main sections:

- Core algorithm - a non-chained implementation without Byzantine fault-tolerance
- Cryptography - integration of cryptographic signing to ensure Byzantine fault-tolerance
- Chained - pipelined version of HotStuff allowing a quorum certificate to serve in multiple phases simultaneously.

2 Starting point

I have some experience using OCaml from the IA course.

3 Success Criteria

- Correctness - The consensus algorithm is implemented as it is described in the paper. This can be established by comparison of the program trace to a known correct implementation or mapping to a verified TLA+ model.
- Evaluation - Analysis of system throughput and latency carried out on a simulated network of 32 replicas.

Evaluation will be carried out by testing the program locally, analysing the trace, and testing in an emulator.

4 Extensions

- Improve transaction throughput and reduce latency. This can be achieved through architectural decisions, tuning the scheduler, and ensuring cryptographic libraries are being used efficiently.
- Support reconfiguration of the network.

5 Timetable

Start	End	Work
17/10/22	30/10/22	Preparatory work: Set up local environment and repository. Learn OCaml and RPC library
31/10/22	13/11/22	Study HotStuff paper in depth. Begin implementing core algorithm. *Cloud Computing 1 deadline (9th Nov)*
14/10/22	27/11/22	Continue implementing core algorithm. *Milestone: core algorithm implementation completed*
28/11/22	11/12/22	Integrate cryptographic libraries. *Cloud Computing 2 deadline (28th Nov)*
12/12/22	25/12/22	Begin implementing chained algorithm.
26/12/22	08/01/23	*Christmas break*
09/01/23	22/01/23	Complete implementation of chained algorithm. Test implementation locally. *Milestone: success criteria met*
23/01/23	05/02/23	Write progress report, prepare presentation. *Cybercrime 1 deadline (3rd Feb)* *Milestone: Progress report submitted (3rd Feb)*
06/02/23	19/02/23	*Slack time / work on extensions* *Cybercrime 2 deadline (17th Feb)* *Milestone: Presentation delivered (8th - 15th Feb)*
20/02/23	05/03/23	Write dissertation outline. *Cybercrime 3 deadline (3rd March)*
06/03/23	19/03/23	Write preparation and implementation chapters. *Cybercrime 4 deadline (17th March)*
20/03/23	02/04/23	Write evaluation chapter.
03/04/23	16/04/23	Write introduction and conclusion chapters. *Milestone: Dissertation draft completed and sent to supervisors and DoS*
17/04/23	30/04/23	Respond to feedback. *Milestone: Dissertation completed*
01/05/23	12/05/23	*Slack time* *Milestone: Dissertation submitted (12th May)*

6 Resources

Laptop (Macbook Air 2020 with M1 chip, 16GB RAM, 512GB SSD)
Sofia server (2x Xeon Gold 6230R chips, 768GB RAM)
git for version control and backups to Github. TeX for typesetting.
If there is a problem with my machine, I will clone the repository and continue on another one of my machines or the MCS.