Paxos Algorithm Use in Smart Meter Technologies

Introduction

* Why your topic is important (convince us!)
* Where is it used? Applications
* What you will talk about / do
* Overview of the rest of your paper

Background and related work

* Any relevant and specific info, e.g. software / hardware statistics, equipment used
* What other people had to say on this topic
* What other people did on this topic
* Problems and shortcomings of their work
* How your work is different and better

Proposed methodology

* Your approach to the problem
* What you did
* Code / Algorithms
* What did / didn’t work
* Results – include graphs, equations, pictures, as appropriate

Conclusions

* What was accomplished / learned
* What you would have done differently
* Future work

References

Appendix

* **Main point:**

Technology Discussion

1. **Research on Paxos Algorithms**
   1. More on the theoretical side
   2. Used in big applications like Apache Zookeeper
   3. One field it could’ve helped was distributive computing
   4. Lots of explanations
2. **Why Paxos?**
   1. Should be looked at again
   2. Blockchain example
   3. Mission impossible falloutIm
   4. Remove the mystery, treat it as any algorithm
   5. For this paper, we will look at Paxos from an undergraduate level and propose a sample application
3. **Paxos Explanation** 
   1. Mention RTT
4. **Implementation of Paxos**
   1. We look at Paxos through a privacy lens
5. **Results**
6. **Conclusion**
   1. Mention Chen paper

Problem Approach

1. Build Deterfox, setup necessary Windows applications to emulate IEC standard and one machine, See if I can find a target machine time

Conclusion

* Write something lel

**Introduction**

As one of the oldest distributed systems algorithms, Paxos is rarely implemented in production software yet still leads a healthy presence in academic research. Unfortunately, Paxos holds a reputation for being complex and hard to understand since it was made during a time of experimentation. However, the future of cloud computing requires candidates for new problems and Paxos is one algorithm that deserves to be on the list. In this paper, we will explain Paxos at the complexity of an undergraduate networks class and also produce a Paxos sample from the research provided.

**Research on Paxos Algorithms**

* + 1. *History* *of Paxos*

A researcher in distributed systems and the creator of LaTeX, Leslie Lamport first came up with the idea of Paxos in 1990 in his paper “The Part-Time Parliament” [Parttime Parliament]. Lamport had seen a presentation on another built fault-tolerant system named Echo, but the system required many states to handle any potential errors for consensus. Instead, Lamport’s paper tried to generalize the problem of consensus and improve fault-tolerance by having previously connected nodes in the system reconnect with the consensus protocol.

According the synopsis of Lamport’s paper, Paxos was the first time a “clearly stated correctness condition and a proof of correctness” [Parttime Parliament]. By comparison, the Echo system [1] did not provide any mathematical proof to guarantee its consensus protocol. It did provide a stated setup that takes a new look beyond primary and secondary databases and identify key terms that continue to be prevalent in material about distributed systems today.

Unfortunately, the key idea of the Paxos algorithm is lost upon the reception and critic of Lamport’s original paper. While it is sound mathematically in its ideas and principles, Lamport chose a more creative approach to the paper by enclosing it in a story about the Paxon society where a quorum of part-time parliaments, faulty politicians, must concede to pass ballots for the governing of society. The paper also contains mathematics that engineers at the time tended to stay away from. Lamport tried to explain his proof more clearly in “Paxos Made Simple” [Paxos Made Simple], but Paxos would not see a resurgence till the 21st century.

* + 1. *Paxos Research and Development*

Since the initial presentation of Paxos, there have been many papers published on modifications to the Paxos algorithm. The Paxos specification itself is not defined explicitly, so researchers mix the concepts with other algorithms. As with any algorithm, optimization was the first step towards adoption, so Lamport wrote another paper with “Fast Paxos” [Fast Paxos]. Paxos by itself has to have two round trips to establish a single ballot proposal (as we will explain in Section II), so by decreasing these messages to only a part of the network will make progress in consensus faster for the majority of nodes deemed non-faulty and call for other nodes to make consensus or proposals later.

Other implementations of Paxos include Cheap Paxos, Multi-Paxos, Stoppable Paxos, and many more [cornell]. All of these consider optimizations on speed and performance. However, there is one version of Paxos that was addressed but not detailed in the original specification—Byzantine Paxos.

During the development of the Echo distributed filesystem, the question of failure was followed by the question of failure with a variable of sabotage. The algorithm that recovers a fault-tolerant system from malicious processors was one of the key features desired in early distributed systems development. While redundancy was always a key factor, recovering consensus from potential malicious processors benefitted an algorithm greatly to its favor and adoption. Echo states that while failing processors and malicious processors operate in the network clients will just see other clients “making strange but valid operations” but might fall int o a denial of service to other clients [echo].

In Paxos’s case, the paper “Byzantizing Paxos by Refinement” [Byzantine Paxos] delivers proofs to satisfy that a Byzantine Paxos algorithm, named *BPCon*, can accept *f* faulty acceptors with 2­*f* + 1 nonfaulty processors and maintain actual consensus. The way BPCon does this is a big development upon regular Paxos, but it does serve to bring the topic considering the implementation scenario of this paper—smart meter privacy reporting.

Other notable of Paxos development can be attributed to its spread in distributed system products. Nearly all the major cloud providers such as Google, Microsoft, and Amazon implement Paxos in their backend systems to provide maximum uptime. Almost every single implementation had Leslie Lamport on the development team. Every other database system typically has Paxos as feature for developers to use and implement such as Apache Zookeeper and Cassandra.

**Paxos Explanation**

In this section, we will explain the fundamentals of the Paxos algorithm in the complexity of a computer science undergraduate networks class. By learning the fundamentals in this way, readers can move onto the material referenced by this paper. Although this explanation is not substantial enough for a course in distributed systems, it can provide an gateway example into the field.

* + 1. Concept

Paxos is a distributed, peer-to-peer algorithm that tries to achieve consensus with three basic roles. Paxos attempts to provide consensus to clients, *processors*, knowing that these processors are faulty and may stop broadcasting at any time. Consensus is a problem of distributed systems to make sure every processor has the correct data value recorded in memory.

Paxos uses a ledger of *ballots* (also called suggestions, proposals, or messages) to keep track what ballots have been voted on and which ballots have been rejected. Ballots in the ledger come with a sequential, unique id that can be verified to be checked by all processors.

**Implementation of Paxos**

**Results**

**Conclusion**

1. Birrell, A., Hisgen, A., Jerian C., et al. “The Echo Distributed File System”. 1993.
2. Chandra, Tushar D., Robert Griesemer, and Joshua Redstone. "Paxos made live: an engineering perspective." Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing. ACM, 2007.
3. Chen, P., Yang, S., McCann J., et al. (2018). Detection of false data injection attacks in smart-grid systems - IEEE Journals & Magazine.
4. Leslie Lamport, "Paxos Made Simple", ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001), December 2001
5. Leslie Lamport, “Fast Paxos” *Distributed Computing.* 14 April 2006.
6. Leslie Lamport, “The Part-Time Parliament”, ***ACM Transactions on Computer Systems 16, 2 (May 1998), 133-169.***
7. Renesse, R. Altinbuken D., “Paxos Made Moderately Complex”, *ACM Computing Surveys*, Vol. 47, No. 3, Article 42, Publication date: February 2015.
8. T. F. Rezende, P. Sutra, R. Q. Saramago, L. Camargos, "On Making Generalized Paxos Practical", 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), March 2017
9. Tom Cocagne, Essential Paxos, January 2013, Github Repository, <https://github.com/cocagne/paxos>