CrushCrypto

What is Practical Byzantine Fault Tolerance (pBFT)

Educational Series

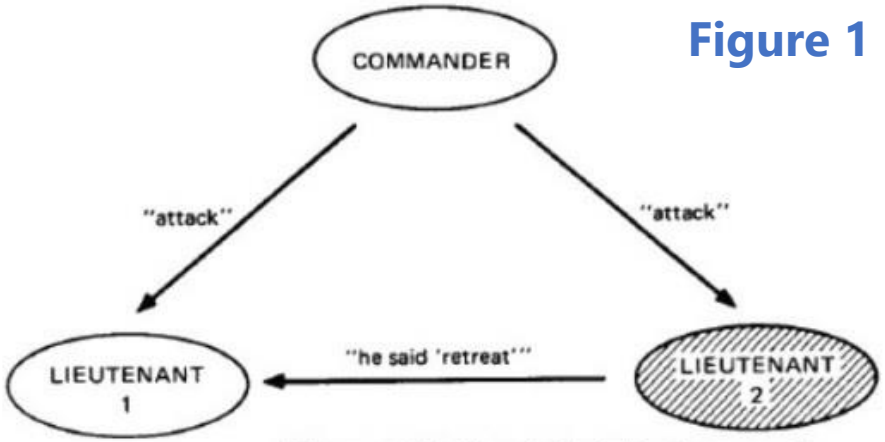
October 21, 2018

Overview

- Byzantine Fault Tolerance is the ability of a distributed computer network to correctly reach a sufficient consensus despite malicious nodes in the system failing or sending out incorrect information. The goal of BFT is to protect against catastrophic system failures by reducing the influence of these malicious nodes.
- BFT is derived from the Byzantine Generals' Problem, a computer science term for a situation where involved parties must agree on a single strategy to avoid a complete failure. However, it assumes some of the involved parties might be corrupt or otherwise unreliable.

Overview (continued)

Figure 1



The one with stripes is the traitor because of which a consensus cannot be reached

Figure 2

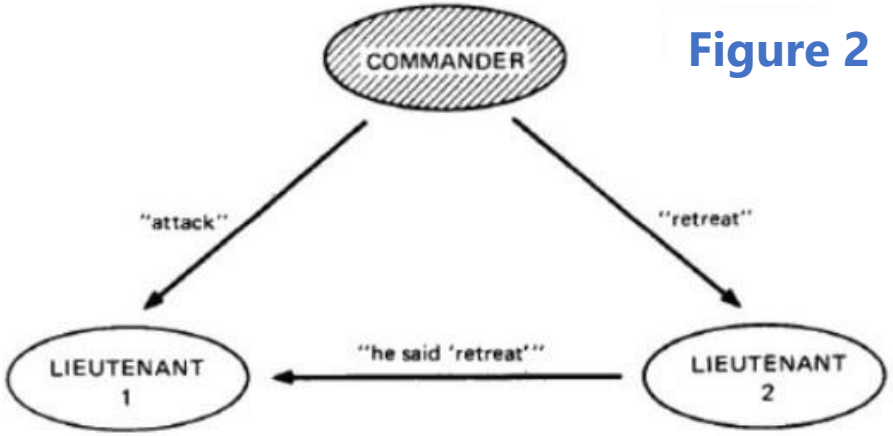
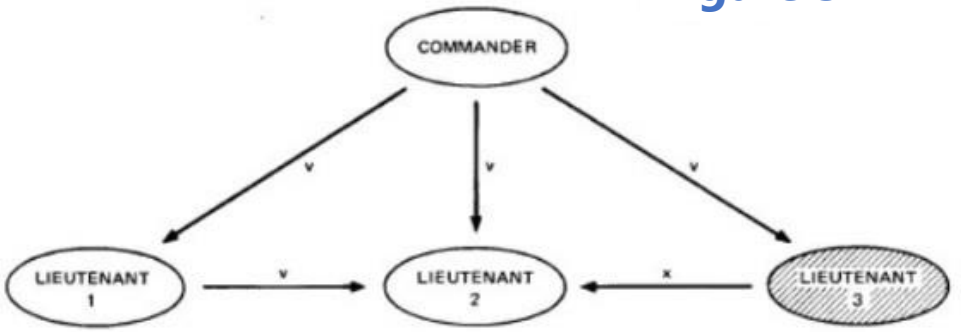
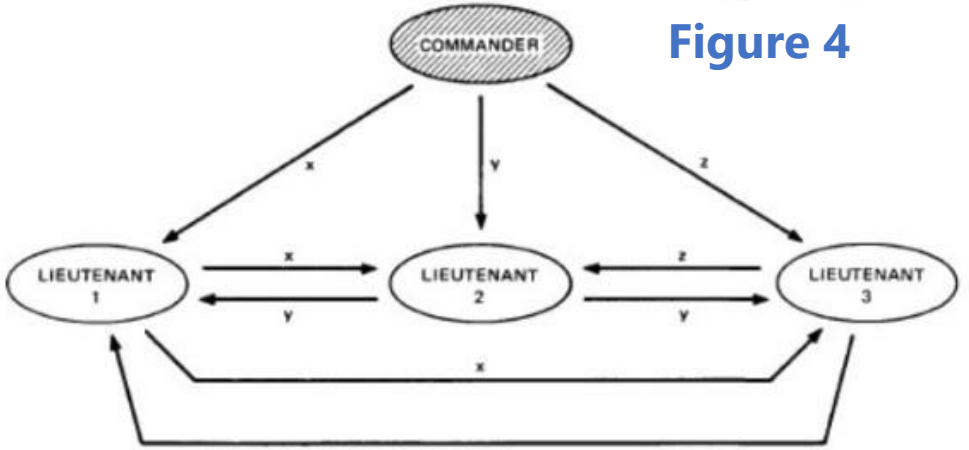


Figure 3



The one with stripes is the traitor because of which a consensus cannot be reached

Figure 4



Overview (continued)

- Byzantine Fault Tolerance is the characteristic that defines a system that tolerates the class of failures belonging to the Byzantine Generals' Problem. BFT has been needed in airplane engine systems, nuclear power plants, and almost any system with actions that depend on many sensors. It also applies to blockchain-based systems, where trust must be established in a distributed network of nodes.
- Practical Byzantine Fault Tolerance (pBFT) is an algorithm that optimizes aspects of Byzantine Fault Tolerance (in other words, protection against Byzantine faults) and has been implemented in several modern distributed computer systems, including some blockchain platforms. These blockchains typically use a combination of pBFT and other consensus mechanisms.

History

- Miguel Castro and Barbara Liskov introduced the Practical Byzantine Fault Tolerance (pBFT) algorithm in a paper released in 1999. It provided high-performance Byzantine state machine replication, processing thousands of requests per second with sub-millisecond increased in latency.
- After the release of pBFT, more BFT protocols were introduced in attempts to improve robustness and performance. Examples of these included Q/U, HQ, Zyzzyva, and ABsTRACTs, which addressed the performance and cost issues, as well as Aardvark and RBFT, which addressed robustness issues.

Appears in the *Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA, February 1999*

Practical Byzantine Fault Tolerance

Miguel Castro and Barbara Liskov
Laboratory for Computer Science,
Massachusetts Institute of Technology,
545 Technology Square, Cambridge, MA 02139
{castro,liskov}@lcs.mit.edu

Abstract

This paper describes a new replication algorithm that is able to tolerate Byzantine faults. We believe that Byzantine-fault-tolerant algorithms will be increasingly important in the future because malicious attacks and software errors are increasingly common and can cause faulty nodes to exhibit arbitrary behavior. Whereas previous algorithms assumed a synchronous system or were too slow to be used in practice, the algorithm described in this paper is practical: it works in asynchronous environments like the Internet and incorporates several important optimizations that improve the response time of previous algorithms by more than an order of magnitude. We implemented a Byzantine-fault-tolerant NFS service using our algorithm and measured its performance. The results show that our service is only 3% slower than a standard unreplicated NFS.

1 Introduction

Malicious attacks and software errors are increasingly common. The growing reliance of industry and government on online information services makes malicious attacks more attractive and makes the consequences of successful attacks more serious. In addition, the number of software errors is increasing due to the growth in size and complexity of software. Since malicious attacks and software errors can cause faulty nodes to exhibit Byzantine (i.e., arbitrary) behavior, Byzantine-fault-tolerant algorithms are increasingly important.

and replication techniques that tolerate Byzantine faults (starting with [19]). However, most earlier work (e.g., [3, 24, 10]) either concerns techniques designed to demonstrate theoretical feasibility that are too inefficient to be used in practice, or assumes synchrony, i.e., relies on known bounds on message delays and process speeds. The systems closest to ours, Rampart [30] and SecureRing [16], were designed to be practical, but they rely on the synchrony assumption for correctness, which is dangerous in the presence of malicious attacks. An attacker may compromise the safety of a service by delaying non-faulty nodes or the communication between them until they are tagged as faulty and excluded from the replica group. Such a denial-of-service attack is generally easier than gaining control over a non-faulty node.

Our algorithm is not vulnerable to this type of attack because it does not rely on synchrony for safety. In addition, it improves the performance of Rampart and SecureRing by more than an order of magnitude as explained in Section 7. It uses only one message round trip to execute read-only operations and two to execute read-write operations. Also, it uses an efficient authentication scheme based on message authentication codes during normal operation; public-key cryptography, which was cited as the major latency [29] and throughput [22] bottleneck in Rampart, is used only when there are faults.

How does it work?

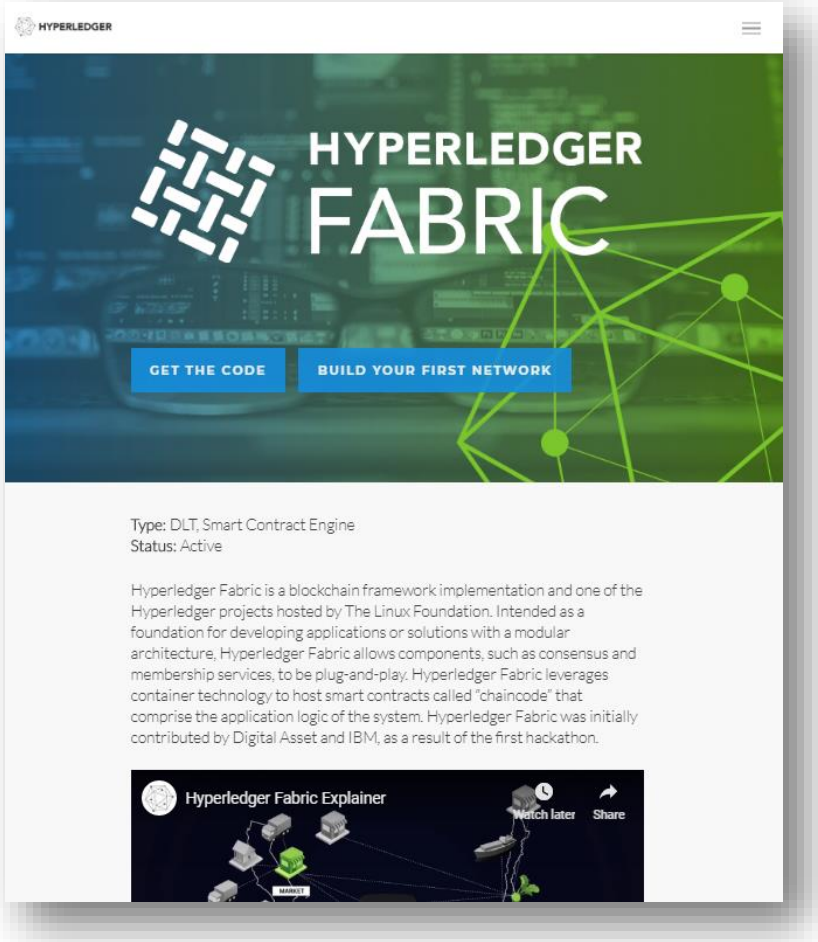
- pBFT focuses on providing a practical Byzantine state machine replication that tolerates Byzantine faults (i.e. malicious nodes) by assuming there are independent node failures and manipulated messages sent through specific nodes.
- Nodes in a pBFT system are sequentially ordered with one node being the leader and others referred to as backup nodes. All nodes in the system communicate with one another with the goal being that all honest nodes will come to an agreement of the state of the system using a majority rule.
- For the pBFT system to function, the number of malicious nodes must not equal or exceed one third of all nodes in the system in a given vulnerability window. Similar to the proof of work consensus mechanism, the more nodes there are in a pBFT network, the more secure it becomes.

How does it work? (continued)

- pBFT consensus rounds are called views and are broken into 4 phases:
  - A client sends a request to the leader node to invoke a service operation.
  - The leading node broadcasts the request to the backup nodes.
  - The nodes execute the request, then send a reply to the client.
  - The client awaits f+1 replies from different nodes with the same result, where f represents the maximum number of potentially faulty nodes.
- The leading node is changed during every view and can be replaced with a protocol called a view change if a certain amount of time has passed without the leading node broadcasting the request. Also, a supermajority of honest nodes can determine when a leader is faulty and replace them with the next leader in line.

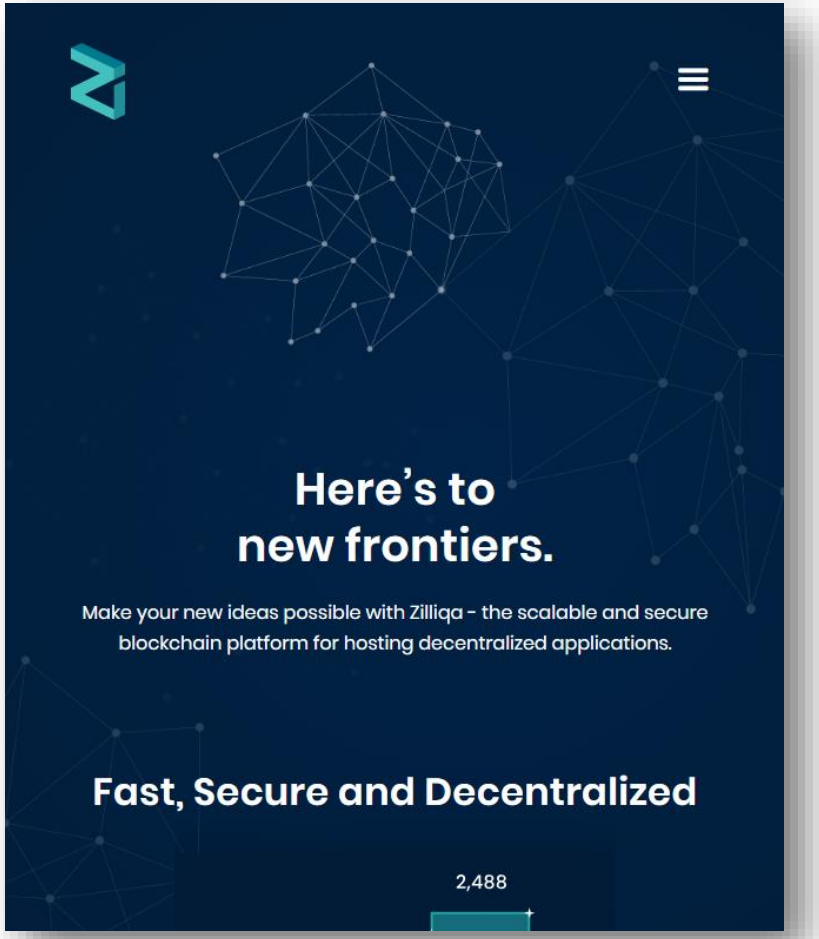
Implementations – Hyperledger Fabric

- An open-source collaborative environment for blockchain projects hosted by the Linux Foundation using a permissioned version of pBFT for the platform. Since permissioned chains use small consensus groups and don't need the same decentralization as public blockchains, pBFT is effective for providing high-throughput transactions.
- Also, permissioned blockchains are private and operated by invite with known identities, meaning there is already an element of trust between parties. This mitigates the need for a trustless environment and allows the network to reap the benefits of pBFT without the downsides.



Implementations – Zilliqa

- Zilliqa uses an optimized version of classical pBFT to achieve consensus about data on the blockchain. Zilliqa also uses a proof of work consensus round every ~100 blocks to perform network sharding, where miners are split into smaller groups referred to as shards. Each shard is capable of processing transactions in parallel, yielding a high throughput for the network.
- The network uses multi-signatures to reduce the communication overhead of classical pBFT and have reached a few thousand transactions per second in their testing environments.



Benefits

- **Transaction finality**: The nature of pBFT means that transactions can be agreed upon and finalized without needing multiple confirmations. There is no waiting period to ensure a transaction is secure after including it in a block.
- **Energy efficiency**: Unlike proof of work consensus mechanisms, pBFT can achieve network consensus without requiring energy intensive computations. Some pBFT systems use proof-of-work to prevent Sybil attack (where a single adversary is controlling multiple nodes on a network, pretending to be multiple parties), but only after a set number of blocks (i.e. 100) and not for every block.
- **Low reward variance**: pBFT requires collective decision through voting on records by signing messages, unlike proof-of-work where only the leader proposes the next block. Thus, every node in a pBFT system can be incentivized, lowering reward variance for miners.

Weaknesses

- **Scaling**: pBFT is a promising consensus solution when the group of nodes is small but becomes inefficient for large networks. This is because each node must talk to every other node to keep the network secure, which can quickly grow into a huge communication cost as the amount of nodes scales upwards.
- **Sybil attacks**: The pBFT models is susceptible to Sybil attacks, where a single party creates or manipulates a large number of nodes in the network and compromises security. This threat is reduced with larger network sizes but considering the scalability issue of pBFT it typically needs to be used in combination with another consensus mechanism.

CrushCrypto