A Review of 'Practical Byzantine Fault Tolerance'

Prashanth R Duggirala

Summary

Paper by MIGUEL CASTRO and BARBARA LISKOV, 1999

The paper describes a new algorithm that can tolerate Byzantine faults in a distributed system which is more practical when compared to algorithms which assume a synchronous system. Byzantine fault tolerant algorithms are designed to deal with a set of failures that arise from having malicious nodes in the system. pBFT focuses on providing fault tolerance by copying servers and consistent client interaction with server copies. pBFT can ensure this while allowing thousands of operations per second with almost negligible increase in latency. All the nodes in the pBFT system are ordered in a sequence and one among them serves as a primary node and the rest are called backup nodes. All of the nodes in the system are allowed to communicate with each other in order to achieve the goal of all the honest nodes come to an agreement of the state of the system. There is a heavy communication between the nodes and all the messages need to be properly verified of their authenticity. One of the assumptions of this algorithm is that the byzantine nodes act independently, another assumption is that each of the nodes in the system follow a state machine which is deterministic in nature and the main assumption for this algorithm to function properly is to have the number of malicious nodes not to exceed one-third of all the nodes in the system. The algorithm provides liveness and safety when the above condition holds true.

Each round of a pBFT consensus is called a view and contains three phases once the system receives a request from the client. The algorithm follows a primary and backup format than having all generals as equals. Once a client sends a request to the system, one of the nodes acts as a leader and multicasts the request to the backup nodes. This is called the pre-prepare phase. The nodes exchange messages amongst themselves about the current state of the system and enter a prepare phase. The pre-prepare and prepare phases of the algorithm guarantee that non-faulty replicas agree on a total order for the requests within a view. Once they agree, they enter the commit phase and reply back to the client. The request is served successfully when the client receives at least f + 1 replies from nodes in the system with the same message, where f is the maximum number of allowed faulty nodes in the system. The primary node usually keeps changing in a round-robin fashion but a view change protocol is invoked if the honest nodes vote on the legitimacy of the current primary node and replace it with the next node in line. To demonstrate that their algorithm can work in reality, the paper authors implemented it on a network file system and evaluated the performance using benchmarks to demonstrate only a small overhead than standard unreplicated system.

Strong points

1. This is the first paper to propose an algorithm for replicating state-machine in a Byzantine fault prone system and tries to efficiently function in more realistic asynchronous systems and realized in at least 3f+1 nodes.

2. In pBFT, the requests can be agreed upon and finalized without needing multiple confirmations and the system is energy efficient as pBFT can achieve consensus without requiring energy intensive proof-of-work like complex computations.

3. The paper describes some optimizations to improve the performance over the basic algorithm, such as reducing the communication load and using message authentication codes rather than digital signatures in messages to reduce cryptographic overhead.

Weak points

1. The algorithm relies on the assumption that we know the value of f, the maximum number of nodes that can be faulty.
2. Scalability of an algorithm that requires heavy messaging overhead in every phase is questionable.
3. The algorithm's assumption that the byzantine nodes act independently is a weakness that can be exploited by attacks.

Comments expanding upon weak points

It seems to be non trivial that we know the value of f before the runtime. Since the client needs to wait for (f+1) replies from different replicas with the same result the client can know the actual f value for the system. Exposing such system parameters seems to be a dangerous idea. pBFT does not seem to scale well with the increase in the number of nodes in the system because of its heavy communication overhead which increases in the order of O(n^k), where n is the number of messages and k is the number of nodes. With this the time taken to respond to the request also significantly increases. View change protocol also incurs a heavy overload. The assumption that the byzantine nodes fail independently is not a very practical assumption, there a number of attacks where a single node controls multiple nodes in the system and they all launch a coordinated attack on the system and compromises security.

References:

Practical Byzantine Fault Tolerance

https://en.bitcoinwiki.org/wiki/PBFT

https://crushcrypto.com/what-is-practical-byzantine-fault-tolerance/

https://www.comp.nus.edu.sg/~rahul/allfiles/cs6234-16-pbft.pdf

https://blockonomi.com/practical-byzantine-fault-tolerance/