**Proof of Authority vs PBFT**

Permissioned blockchains are arising as a solution to federate companies prompting accountable interactions. A variety of consensus algorithms for such blockchains have been proposed, each of which has different benefits and drawbacks. Proof-of-Authority (PoA) is a new family of Byzantine fault-tolerant (BFT) consensus algorithms largely used in practice to ensure better performance than traditional Practical Byzantine Fault Tolerance (PBFT). However, the lack of adequate analysis of PoA hinders any cautious evaluation of their effectiveness in real-world permissioned blockchains deployed over the Internet, hence on an eventually synchronous network experimenting Byzantine nodes.

In this paper, analysis of two of the main PoA algorithms, named Aura and Clique, both in terms of provided guarantees and performances. First, functionality is derived including how messages are exchanged, then weight, by relying on the CAP theorem, consistency, availability and partition tolerance guarantees. Reporting of a qualitative latency analysis based on message rounds is also done. The analysis advocates that PoA for permissioned blockchains, deployed over the Internet with Byzantine nodes, do not provide adequate consistency guarantees for scenarios where data integrity is essential but works well for large distributed platforms.

Proof of Authority is a new family of BFT algorithms which has recently drawn attention due to the offered performance and toleration to faults. PoA requires less message exchanges hence provides better performance. PoA algorithms favour availability over consistency, oppositely to what PBFT guarantees. This can sometimes be not suitable for permissioned blockchains. PBFT can be a better choice but it can also be worse than some POA implementations.

PoA algorithms rely on a set of N trusted nodes called the authorities. Each authority is identified by a unique id and a majority of them is assumed honest, namely at least N/2 + 1. The authorities run a consensus to order the transactions issued by clients. Time is divided into steps, each of which has an authority elected as mining leader.The two PoA implementations work quite differently: both have a first round where the new block is proposed by the current leader (block proposal); then Aura requires a further round (block acceptance), while Clique does not.

Aura : If authorities do not agree on the proposed block during the block acceptance, a voting is triggered to decide whether the current leader is malicious and then kick it out. An authority can vote the current leader malicious because (i) it has not proposed any block, (ii) it has proposed more blocks than expected, or (iii) it has proposed different blocks to different authorities.

Clique: Clique computes the current step and related leader using a formula that combines the block number and the number of authorities. Each authority is only allowed to propose a block every N/2+ 1 blocks.

CAP Theorem : The CAP Theorem states that in a distributed data store only two out of the three following properties can be ensured: Consistency (C), Availability (A) and Partition Tolerance (P). Thus any distributed data store can be characterised on the basis of the (at most) two properties it can guarantee, either CA, CP or AP.

\* A blockchain achieves consistency when forks are avoided.

\* A blockchain is available if transactions submitted by clients are served and eventually committed, i.e. permanently added to the chain.

\* When a network partition occurs, authorities are divided into disjoint groups in such a way that nodes in different groups cannot communicate each other.

By applying the CAP Theorem, it is claimed that in this setting PoA algorithms can give up consistency for availability when considering the presence of Byzantine nodes. PBFT keeps the blockchain consistent at the cost of availability, even when the network behaves temporarily asynchronously and Byzantine nodes are present; this behaviours is much more desirable when data integrity is a priority.