Byzantine Fault Tolerance (BFT)

The object of Byzantine fault tolerance (BFT) is to be able to defend against failures, in which components of a system fail in arbitrary ways, i.e., not just by stopping or crashing but by processing requests incorrectly, corrupting their local state, and/or producing incorrect or inconsistent outputs, producing different symptoms to different observers. The other components of the system, in order to declare it failed this component and shut it out of the network, they need to reach a consensus regarding which component is failed in the first place. The objective of Byzantine fault tolerance is to be able to defend against Byzantine failures in systems that require consensus. Correctly functioning components of a Byzantine fault tolerant system, will be able to provide the system's service, assuming that there are not too many faulty components. It's important to remark that a Byzantine failure is not necessarily a security problem involving hostile human. Real-world examples of Byzantine failures happened during the testing for the Virgina class submarine, or in honeybee swarms, where they have to find a new home, and the many scouts and wider participants have to reach consensus about which of perhaps several candidate homes to fly to. There are different solutions to this problem like: (i) scenarios in which messages may be forged, but which will be Byzantine-fault-tolerant as long as the number of failure nodes does not equal or exceed one third of the nodes that are working as they should, (ii) a scenario that requires unforgeable message signatures, and (iii) scenarios that allow a Byzantine-fault-tolerant behavior in some situations where not all nodes can communicate directly with each other. Another solution was introduced by Miguel Castro and Barbara Liskov called the "Practical Byzantine Fault Tolerance" (PBFT) algorithm for state machine replication that is able to tolerate Byzantine faults. The PBFT algorithm, imposes two requirements on replicas: (i) they must be deterministic, and (ii) they must start in the same state. Given these requirements, PBFT ensures the safety property by guaranteeing that all non-faulty replicas agree on a total order for the execution of requests despite failures. The PBFT algorithm works in asynchronous scenarios like the internet, whereas previous algorithms assumed a synchronous system. Also, this algorithm offers both liveness and safety provided at most (n-1)/3 out of a total of n replicas are simultaneous faulty. Authors allow an attacker that can coordinate faulty nodes, delay communications, or delay correct nodes in order to cause the most damage to the replicated service. Also they assume that the adversary cannot delay correct nodes indefinitely. They limit the amount of damage that an attacker can do by providing access control. The PBFT algorithm does not address the problem of fault tolerant privacy, in which a faulty replica may leak information to an attacker.

Consensus on BFT replication and PoW can be compare in different aspects. PoW consensus depends on computational power of a node, whereas BFT approach to consensus requiring that every node know the entire set of its peer nodes participating in consensus. An interesting property called "consensus finality" says that a valid block appended to the blockchain at some point in time, be never removed from the block chain. This property is not satisfied by PoW-based blockchains like Bitcoin, whereas is satisfied by BFT and state-machine replication protocols.