Byzantine fault tolerance – protection layer in pairwise messaging system from dishonest (failing) components. BFT makes sure that every node in the system sending the same information to all other nodes. It does not check if the information is correct or not. We can say that BFT is consistency check test. To reach the consensus 3*n(number of components in the system)+1 messaging needed. One of the early solutions to Byzantine tolerance is "Commander and Lieutenants" where we check only generals if they are not traitor, lieutenants obey them and receive message. As long as $n > 3t$*(traitors)* BTF works. Second, using digital signatures to detect forged messages. In bitcoin BFT ensures that bitcoin network works parallel to blockchain using proof of work.

One of the papers discuss new replication method that tolerates Byzantine faults. The algorithm is practical because it works in asynchronous environment, which is main drawback of traditional BFT. Basically, algorithm replicates the main state (node) and replicas work together by gossiping (view system). Each time replica receives request and sends request across the distributed system and waits for certain number of (identical) answers with valid signatures, which will be request result. Each message/request is signed by each replica when viewed which ensures system to stay alive and come to one outcome.

There are two consensus used in blockchain technologies: PoW (permissionless) and BFT (permissioned). BFT is more preferred than PoW because, BFT participants know each other which is main concern of governments but it is not well scalable. BFT is better in a way that reaches to final consensus permanently, no pruned blocks. And also in terms of performance BFT dominates PoW. However, both type of consensuses are not efficient enough. In recent years different proposals have been presented to optimize both consensuses. One of the most appealing proposal is mixing PoW with BFT.