

4.3 Tendermint

Tendermint Core (<https://github.com/tendermint/tendermint>) is a BFT protocol that can be best described as a variant of PBFT [22], as its common-case messaging pattern is a variant of Bracha’s Byzantine reliable broadcast [11]. In contrast to PBFT, where the client sends a new transaction directly to all nodes, the clients in Tendermint disseminate their transactions to the validating nodes (or, simply, validators) using a gossip protocol. The external validity condition, evaluated within the Bracha-broadcast pattern, requires that a validator receives the transactions by gossip before it can vote for inclusion of the transaction in a block, much like in PBFT.

Tendermint’s most significant departure from PBFT is the continuous rotation of the leader. Namely, the leader is changed after every block, a technique first used in BFT consensus space by the *Spinning* protocol [62]. Much like Spinning, Tendermint embeds aspects of PBFT’s view-change mechanism into the common-case pattern. This is reflected in the following: while a validator expects the first message in the Bracha broadcast pattern from the leader, it also waits for a timeout, which resembles the view-change timer in PBFT. However, if the timer expires, a validator continues participating in the Bracha-broadcast message pattern, but votes for a *nil* block.

Tendermint as originally described by Buchman [13] suffers from a livelock bug, pertaining to locking and unlocking votes by validators in the protocol. However, the protocol contains additional mechanisms not described in the cited report that prevent the livelock from occurring [14]. While it appears to be sound, the Tendermint protocol and its implementation are still subject to a thorough, peer-reviewed correctness analysis.

	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/3$	$f < n/3$
Liveness	n	$t < n/3$	$f < n/3$

Table 4: Resilience of Tendermint.

4.4 Symbiont – BFT-SMaRt

Symbiont Assembly (<https://symbiont.io/technology/assembly>) is a proprietary distributed ledger platform. The company that stands behind it, Symbiont, focuses on applications of distributed ledgers in the financial industry, providing automation for modeling and executing complex instruments among institutional market participants.

Assembly implements resilient consensus in its platform based on the open-source BFT-SMaRt toolkit (Sec. 3.3). Symbiont uses its own reimplementation of BFT-SMaRt in a different programming language; it reports performance numbers of 80’000 transactions per second (tps) using a 4-node cluster on a LAN. This matches the throughput expected from BFT-SMaRt [9] and similar results in the research literature on BFT protocols [5].

Assembly uses the standard resilience assumptions for BFT consensus in the eventually-synchronous model considered here.

	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/3$	$f < n/3$
Liveness	n	$t < n/3$	$f < n/3$

Table 5: Resilience of BFT-SMaRt (reimplemented inside Symbiont Assembly).