MonadBFT

Pipelined two-phase HotStuff Consensus MonadBFT is a high-performance consensus mechanism for achieving agreement about the transaction ordering under partially synchronous conditions in the presence of Byzantine actors. It is a derivative ofHotStuff with the improvement proposed inJolteon /DiemBFT /Fast-HotStuff which is the reduction from three rounds to two rounds by utilizing quadratic communication complexity in the event of leader timeout.

MonadBFT is a pipelined two-phase BFT algorithm with optimistic responsiveness, and with linear communication overhead in the common case and quadratic communication in the case of a timeout. As in most BFT algorithms, communication proceeds in phases; at each phase, the leader sends a signed message to the voters, who send signed responses to the following leader. Pipelining allows the quorum certificate (QC) or timeout certificate (TC) for blockk to piggyback on the proposal for blockk+1.

Quick facts

Sybil resistance mechanism Proof-of-Stake (PoS) Block time 1 second Finality Single-slot Delegation allowed Yes

Mempool

SeeShared Mempool .

Consensus Protocol

MonadBFT is a pipelined consensus mechanism that proceeds in rounds. The below description gives a high-level intuitive understanding of the protocol.

As is customary, let there ben = 3f+1 nodes, wheref is the max number of Byzantine nodes, i.e.2f+1 (i.e. 2/3) of the nodes are non-Byzantine. In the discussion below, let us also treat all nodes as having equal stake weight; in practice all thresholds can be expressed in terms of stake weight rather than in node count.

- In each round, the leader sends out a new block as well as either a QC or a TC (more on this shortly) for the previous round.
- Each validator reviews that block for adherence to protocol and, if they agree, send signed YES votes to the leader of the next round. That leader derives a QC (quorum certificate) by aggregating (via threshold signatures) YES votes from2f+1
- validators. Note that communication in this case islinear
- : leader sends block to validators, validators send votes directly to next leader.
-
  - Alternatively, if the validator does not receive a valid block within a pre-specified amount of time, it multicasts a signed timeout message toall
-
  - of its peers. This timeout message also includes the highest QC that the validator has observed. If any validator accumulates2f+1
-
  - timeout messages, it assembles these (again via threshold signatures) into a TC (timeout certificate) which it then sends directly to the next leader.
- *
- Each validator finalizes the block proposed in roundk
- upon receiving a QC for roundk+1
- (i.e. in the communication from the leader of roundk+2
-). Specifically:
-
  - Alice, the leader of roundk
-
  - , sends a new block to everyone (along with either a QC or TC for roundk-1
-
  - ; let's ignore that as it's not important).
-
  - If2f+1
-
  - validators vote YES on that block by sending their votes to Bob (leader of roundk+1
-
  -), then the block ink+1
-
  - will include a QC for roundk
-
  - . However, seeing the QC for roundk
-

  - at this point
-
  - is not enough for Valerie the validator to know that the block in roundk
-
  - has been enshrined, since (for example) Bob could have been malicious and only sent the block to Valerie. All that Valerie can do is vote on blockk+1
-
  - , sending her votes to Charlie (leader of roundk+2
-
  -).
-
  - If2f+1
-
  - validators vote YES on blockk+1
-
  - , then Charlie publishes a QC for roundk+1
-
  - as well as a block proposal for roundk+2
-
  - . As soon as Valerie receives this block, she knows that the block from roundk
-
  - (Alice's block) is finalized.
-
  - Say that Bob had acted maliciously, either by sending an invalid block proposal at roundk+1
-
  - , or by sending it to fewer than2f+1
-
  - validators. Then at leastf+1
-
  - validators will timeout, then triggering the other non-Byzantine validators to timeout, leading to at least one of the validators to produce a TC for roundk+1
-
  - . Then Charlie will publish the TC for roundk+1
-
  - in his proposal (he will have to in order to make a proposal, because no QC will be available for roundk+1
-
  -).
-
  - We refer to this commitment procedure as a 2-chain commit rule, because as soon as a validator sees 2 adjacent certified blocksB
-
  - andB'
-
  - , they can commitB
-
  - and all of its ancestors.
- *
-

References:

- Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai AbrahamHotStuff: BFT Consensus in the Lens of Blockchain
- , 2018.
- Mohammad M. Jalalzai, Jianyu Niu, Chen Feng, Fangyu GaiFast-HotStuff: A Fast and Resilient HotStuff Protocol
- , 2020.
- Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun XiangJolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback
- . arXiv preprint arXiv:2106.10362, 2021.
- The Diem Team.DiemBFT v4: State machine replication in the diem blockchain
- . 2021.
-

BLS Multi-Signatures

Certificates (QCs and TCs) can be naively implemented as a vector of ECDSA signatures on the secp256k1 curve. These certificates are explicit and easy to construct and verify. However, the size of the certificate is linear with the number of signers. It poses a limit to scaling because the certificate is included in almost every consensus message, except vote message.

Pairing-based BLS signature on the BLS12-381 curve helps with solving the scaling issue. The signatures can be incrementally aggregated into one signature. Verifying the single valid aggregated signature provides proof that the stakes associated with the public keys have all signed on the message.

BLS signature is much slower than ECDSA signature. So for performance reasons, MonadBFT adopts a blended signature scheme where BLS signature is only used on aggregatable message types (votes and timeouts). Message integrity and authenticity is still provided by ECDSA signatures.

Previous Consensus Next Shared Mempool Last updated5 months ago