Maidenlane Protocols (draft)

Sam Bacha[1], Advisor [1], Editor[2], Advisor[2,3]
[1] Freight Trust Clearing, Corporation
[2] v0.4.0
[3] Santa Monica, California, U.S.Al
contact@freighttrust.com, admin@freighttrust.com, PGP
github: /freight-trust, /protocol

ABSTRACT

Many Supply Chain solutions purporte. While the literature has been extensively contributing towards more reliability, robustness and resilience of consensus systems, few works addressed large scale industrial cases focused on State Machine Replication with one block finality. This paper,

Keywords: Byzantine Fault Tolerance, Supply Chain, Logistics, Forward Contracts, Electronic Data Interchange, Commodities, Combinatorial System

1. Introduction

Various studies in the literature dealt with partially synchronous and fully asynchronous Byzantine Fault Tolerant systems **???**, but few of them were really applied in a live Smart Contract (SC) Scenario with plenty of distinct decentralized applications. It is noteworthy that append storage applications posses different level of challenges compared to the current need of SC transactions persisting, which involve State Machine Replication (SMR) **?**. In addition, a second important fact to be considered is related to the finality in appending information to the ledger. Final users, merchants and exchanges want to precisely known if their transaction was definitively processed or still could be reverted.

- One block finality to the end-users and seed nodes;
- Use of cryptographic signatures during different phases of the procedures in order to avoid exposure of nodes commitment to the current block;
- Ability of proposing blocks based information sharing of block headers;
- Disable change views after commitment phase;
- Regeneration mechanism able to recover failed nodes both in hardware and consensus layer.

Furthermore, it introduces a novel mathematical model able to verify specific consensus behavior by means of a discrete model with can simulate real cases operation. While highlighting the positive aspects of the current consensus system, the authors do not measure their efforts in criticizing its faults and negative aspects. On the other hand, solutions, inspired by well-known studies from the literature, are proposed to be incorporated in the current procedure in order to achieve an even more robust and reliable mechanism.

The remainder of this paper is organized as follows. Section 2 details the current mProtocol implementation and proposed specifications.

2. mProtocol detailed description

2.1. Flowchart

2.2. Pseudocode

2.3. Signatures sharing

We consider that nodes can share any valid signature 3.2).

In this sense, we highlight here a decentralized P2P network (fully distributed), in which nodes try to inform each other as much as they can about any useful state. The trade-off that borders the discussions is the use bandwidth while the notorious advantage is the creation and optimization of the best communication routes between entities.

2.4. Block finality

Block finality in the Consensus layer level imposes the following condition presented at Eq (1), which defines that there should not exist two different blocks for a given height h, in any time interval t.

$$\forall h \in \{0, 1, \cdots, t\} \Rightarrow b_t^i = b_t^j \tag{1}$$

In summary, the block finality provides that clients do not need to verify the majority of Consensus for SMR. In this sense, seed nodes can just append all blocks that posses the number of authentic signatures defined by the protocol.

For the current mProtocol, the minimum number of required signatures is $M = 2f$, where $f = \frac{1}{3} \times N$ is the maximum number of Byzantine nodes allowed by the network protocol.

2.5. Blocking change views

It is noteworthy that even in an Asynchronous Consensus without timeout mechanism this case could lead to problems if the Nonce was not yet defined as well as the transactions to be inserted inside a Block.

In this sense, the possibility that naturally came was:

- Lock view changing (currently implemented in the mProtocol) after sending your signature. This means that those who commit with that block will not sign any other proposed Block.

2.6. Regeneration

The Recover/Regeneration event is designed for responding to a given failed node that it lost part of the history. In this sense, if the node had failed and recovered its healthy (sending a change_view) it might receive a payload that provides it the ability to check agreements of the majority and come back to real operation, helping them to sign the current block being processed.

3. Possible faults

3.1. Pure network faults

Possible scenarios:

- ‘f‘ nodes will delays messages;
- At maximum, ‘f‘ will crash both in terms of hardware fault or software problems;

3.2. Pure byzantine faults

First of all, Byzantine attacks should be designed in order that nodes will never be able to prove that it was an attack.

Obviously, nodes that join a given collaborative network posses an identity or stake. In this sense, if anyone can detect this kind of behavior then that node will automatically be removed from the network.

- at maximum, f, nodes will delays messages;
- at maximum, f, nodes will store messages;
- at maximum, f, nodes will send wrong information;
- at maximum, f, nodes will try to keep correct information for strategic occasions;

3.3. Mixed faults

4. A MILP Model for Failures and Attacks on a BFT Blockchain Protocol

We present a MILP model for failures and attacks on a BFT blockchain protocol.

Parameters:

$i \in R$**:** consensus replica i from set of replicas R. R^{BYZ} is byzantine set. R^{OK} is non-byzantine set. $R = R^{OK} \cup R^{BYZ}$, such that $R^{OK} \cap R^{BYZ} = \emptyset$.

f**:** number of faulty/Byzantine replicas. $f = |R^{BYZ}|$.

N**:** total number of replicas. $N = |R| = |R^{OK}| + |R^{BYZ}| = 3f + 1$.

M**:** safety level. $M = 2f + 1$.

$b \in B$**:** block b from set of possible proposed blocks B (may be understood as block hash). $B = \{b_0, b_1, b_2, \cdots\}$.

$h \in H$**:** height h from set of possible heights H (tests may only require two or three heights). $H = \{h_0, h_1, h_2\}$.

$v \in V$**:** view v from set of possible views V (number of views may be limited to the number of consensus nodes N). $V = \{v_0, v_1, \cdots, v_{N-1}\}$

$t \in T$**:** time unit t from set of discrete time units T. $T = \{t_0, t_1, t_2, \cdots\}$.

Variables:

$primary_{i,h,v}$**:** binary variable that indicates if Consensus Node i is primary at height h view v.

$initialized^t_{i,h,v}$**:** binary variable that indicates if replica $i \in R$ is at height h and view v, on time t

$SendPrepReq^t_{i,h,b,v}$**:** binary variable that indicates if replica $i \in R$ is sending Prepare Request message (to all nodes) at height h and view v, on time t, for proposed block b. ACTION VARIABLE MUST BE SET ONLY ONCE FOR EVERY REPLICA, HEIGHT AND BLOCK.

$SendPrepResp^t_{i,h,b,v}$**:** binary variable that indicates if replica $i \in R$ is sending Prepare Response message (to all nodes) at height h and view v, on time t, for proposed block b. ACTION VARIABLE MUST BE SET ONLY ONCE FOR EVERY REPLICA, HEIGHT AND BLOCK.

$RecvPrepReq^t_{i,j,h,b,v}$**:** binary variable that indicates if replica $i \in R$ received a Prepare Request message from replica j at height h and view v, on time t, for proposed block b. ACTION VARIABLE MUST BE SET ONLY ONCE FOR EVERY REPLICA, HEIGHT AND BLOCK.

$RecvPrepResp^t_{i,j,h,b,v}$**:** binary variable that indicates if replica $i \in R$ received a Prepare Response from replica j at height h and view v, on time t, for proposed block b. ACTION VARIABLE MUST BE SET ONLY ONCE FOR EVERY REPLICA, HEIGHT AND BLOCK.

$BlockRelay^{t}_{i,h,b}$**:** binary variable that indicates if replica i has relayed block b at height h, on time t. ACTION VARIABLE MUST BE SET ONLY ONCE FOR EVERY REPLICA, HEIGHT AND BLOCK.

$RecvBlkPersist^{t}_{i,j,h,b}$**:** binary variable that indicates if replica $i \in R$ received a Block Relay message from replica j at height h on time t, for proposed block b. ACTION VARIABLE MUST BE SET ONLY ONCE FOR EVERY REPLICA, HEIGHT AND BLOCK.

$sentPrepReq^{t}_{i,h,b,v}$**:** binary variable that indicates if replica $i \in R$ has sent (in past) to all replicas a Prepare Request message at height h and view v, on time t, for proposed block b. Once set to ONE this is carried forever as ONE.

$sentPrepResp^{t}_{i,h,b,v}$**:** binary variable that indicates if replica $i \in R$ has sent (in past) to all replicas a Prepare Response message at height h and view v, on time t, for proposed block b. Once set to ONE this is carried forever as ONE.

$recvdPrepReq^{t}_{i,j,h,b,v}$**:** binary variable that indicates if replica $i \in R$ has received (in past) from replica j a Prepare Request message at height h and view v, on time t, for proposed block b. Once set to ONE this is carried forever as ONE.

$recvdPrepResp^{t}_{i,j,h,b,v}$**:** binary variable that indicates if replica $i \in R$ has received (in past) from replica j a Prepare Response message at height h and view v, on time t, for proposed block b. Once set to ONE this is carried forever as ONE.

$sentBlkPersist^{t}_{i,h,b}$**:** binary variable that indicates if replica $i \in R$ has sent (in past) to all replicas a Block Relay message at height h, on time t, for proposed block b. Once set to ONE this is carried forever as ONE.

$recvdBlkPersist^{t}_{i,j,h,b}$**:** binary variable that indicates if replica $i \in R$ has received (in past) from replica j a Block Relay message at height h, on time t, for proposed block b. Once set to ONE this is carried forever as ONE.

$blockRelayed_{b}$**:** binary variable that indicates if block b was relayed (on any time, height or view).

5. Network Routing and Search

5.1. Uniform Cost Search

Uniform cost search expands the node with the lowest cost-so-far $=\sum$i j$_{ij}$c$_{ij}$ on the fringe. Hence, the relative ordering between two nodes is determined by the value of $\sum_{ij} c_{ij}$ for a given node. So only for $d_{ij} = \alpha c_{ij}, \alpha > 0$, can we conclude, $\sum_{ijin \text{ path } (n)} d_{ij} \geq \sum_{ijin \text{ path } (m)} d_{ij} \iff \sum_{ijin\,\text{path}(n)} c_{ij} \geq \sum_{ijin\,\text{path}(m)} c_{ij}$, for some nodes n and m

5.2. Iterative Search Parameters

Cost of $X + Y$ is decreased by n, $n > 0$, the edge is not on the optimal path, and was not explored by the first search.

5.3. Proof of Violation

Objective function:

$$maximize \sum_{b \in B} blockRelayed_{b} \tag{2}$$

The adversary can control f replicas, but the other M replicas must follow mProtocol algorithm. The adversary can choose any delay for any message (up to maximum simulation time $|T|$). If it wants to shutdown the whole network, no blocks will be ever produced and objective will be zero (minimum possible). So, adversary will try to maximize blocks produced by manipulating delays in a clever way. Objective function is bounded to [0, $|B|$].

Constraints:
Initialization constraints

$$initialized^{t_0}_{i,h_0,v_0} = 1 \qquad \forall i \in R^{OK} \tag{3}$$

$$initialized^{t_0}_{i,h,v} = 0 \qquad \forall i \in R^{OK}, h \in H \setminus \{h_0\}, v \in V \setminus \{v_0\} \tag{4}$$

$$\sum_{v \in V} initialized^{t}_{i,h,v} = 1 \qquad \forall i \in R, t \in T \setminus \{t_0\}, h \in H \tag{5}$$

$$\sum_{h \in H} initialized^{t}_{i,h,v} = 1 \qquad \forall i \in R, t \in T \setminus \{t_0\}, v \in V \tag{6}$$

Time zero constraints:

$$SendPrepReq^{t_0}_{i,h,b,v} = 0 \qquad \forall i \in R, \forall h, b, v \tag{7}$$

$$sentPrReq^{t_0}_{i,h,b,v} = 0 \qquad \forall h, b, i, v \tag{8}$$

$$RecvPrepReq^{t_0}_{i,j,h,b,v} = 0 \qquad \forall i, j \in R, \forall h, b, v \tag{9}$$

$$recvdPrReq^{t_0}_{i,j,h,b,v} = 0 \qquad \forall j, h, b, i, v \tag{10}$$

$$SendPrepResp^{t_0}_{i,h,b,v} = 0 \qquad \forall i \in R, \forall h, b, v \tag{11}$$

$$sentPrResp^{t_0}_{i,h,b,v} = 0 \qquad \forall h, b, i, v \tag{12}$$

$$RecvPrepResp^{t_0}_{i,j,h,b,v} = 0 \qquad \forall i, j \in R, \forall h, b, v \tag{13}$$

$$recvdPrResp^{t_0}_{i,j,h,b,v} = 0 \qquad \forall j, h, b, i, v \tag{14}$$

$$BlockRelay^{t_0}_{i,h,b} = 0 \qquad \forall i \in R, \forall h, b \tag{15}$$

$$sentBlkPersist^{t_0}_{i,h,b} = 0 \qquad \forall i \in R, \forall h, b \tag{16}$$

$$RecvBlkPersist^{t_0}_{i,j,h,b} = 0 \qquad \forall i, j \in R, \forall h, b \tag{17}$$

$$recvdBlkPersist^{t_0}_{i,j,h,b} = 0 \qquad \forall i, j \in R, \forall h, b \tag{18}$$

(19)

Prepare request constraints:

$$SendPrepReq^{t}_{i,h,b,v} \leq initialized^{t}_{i,h,v} \qquad \forall i, h, b, v, t \tag{20}$$

$$SendPrepReq^{t}_{i,h,b,v} \leq primary_{i,h,v} \qquad \forall i, h, b, v, t \tag{21}$$

$$sentPrReq^{t}_{i,h,b,v} = sentPrReq^{t-1}_{i,h,b,v} + SendPrepReq^{t-1}_{i,h,b,v} \qquad \forall h, b, i, v, t \in T \setminus \{t_0\} \tag{22}$$

$$RecvPrReq^{t}_{i,j,h,b,v} \leq sentPrReq^{t}_{j,h,b,v} \qquad \forall h, b, i \neq j, v, t \tag{23}$$

$$RecvPrReq^{t}_{i,i,h,b,v} = SendPrepReq^{t}_{i,h,b,v} \qquad \forall h, b, i, v, t \tag{24}$$

$$recvdPrReq^{t}_{i,j,h,b,v} = recvdPrReq^{t-1}_{i,j,h,b,v} + RecvPrReq^{t-1}_{i,j,h,b,v} \qquad \forall h, b, i, j, v, t \in T \setminus \{t_0\} \tag{25}$$

Prepare response constraints:

$$SendPrepResp^t_{i,h,b,v} \leq initialized^t_{i,h,v} \qquad \forall i, h, b, v, t \tag{26}$$

$$SendPrepResp^t_{i,h,b,v} \geq \frac{1}{N} \sum_{j \in R} recvdPrReq^{t-1}_{i,j,h,b,v} \qquad \forall i \in R^{OK}, h, b, v, t \tag{27}$$

$$SendPrepResp^t_{i,h,b,v} \leq \sum_{j \in R} recvdPrReq^{t-1}_{i,j,h,b,v} \qquad \forall i \in R, h, b, v, t \tag{28}$$

$$sentPrResp^t_{i,h,b,v} = sentPrResp^{t-1}_{i,h,b,v} + SendPrepResp^{t-1}_{i,h,b,v} \qquad \forall h, b, i, v, t \in T \setminus \{t_0\} \tag{29}$$

$$RecvPrResp^t_{i,j,h,b,v} \leq sentPrResp^t_{j,h,b,v} \qquad \forall h, b, i \neq j, v, t \tag{30}$$

$$RecvPrResp^t_{i,i,h,b,v} = SendPrepResp^t_{i,h,b,v} \qquad \forall h, b, i, v, t \tag{31}$$

$$recvdPrResp^t_{i,j,h,b,v} = recvdPrResp^{t-1}_{i,j,h,b,v} + RecvPrResp^{t-1}_{i,j,h,b,v} \qquad \forall h, b, i, j, v, t \in T \setminus \{t_0\} \tag{32}$$

Block persist constraints:

$$sentBlkPersist^t_{i,h,b} = sentBlkPersist^{t-1}_{i,h,b} + BlockRelay^{t-1}_{i,h,b} \qquad \forall i \in R, h, b, t \tag{33}$$

$$RecvBlkPersist^t_{i,j,h,b} \leq sentBlkPersist^t_{j,h,b} \qquad \forall h, b, i \neq j, v, t \tag{34}$$

$$RecvBlkPersist^t_{i,i,h,b} = BlockRelay^t_{i,h,b} \qquad \forall h, b, i, t \tag{35}$$

$$recvdBlkPersist^t_{i,j,h,b} = recvdBlkPersist^{t-1}_{i,j,h,b} + RecvBlkPersist^{t-1}_{i,j,h,b} \qquad \forall h, b, i, j, t \in T \setminus \{t_0\} \tag{36}$$

Block relay constraints:

$$\sum_{t \in T} BlockRelay^t_{i,h,b} \leq 1 \qquad \forall i \in R, \forall h, b \tag{37}$$

$$blockRelayed_b \geq \frac{1}{N|H|} \sum_{t \in T} \sum_{i \in R} \sum_{h \in H} BlockRelay^t_{i,h,b} \qquad \forall b \in B \tag{38}$$

$$BlockRelay^t_{i,h,b} \leq \frac{1}{M} \sum_{j \in R} recvdPrResp^{t-1}_{i,j,h,b,v} + \sum_{j \in R} recvdBlkPersist^t_{i,j,h,b} \qquad \forall i \in R, h, b, v, t \tag{39}$$

5.4. Example

Fixed values presented in bold.

$initialized^t_{i,h,v}$, for $i \in R^{OK}$, $h = 0$, $v = 0$:

i=0	**1**	1	1	1	1	...
t	0	1	2	3	4	...

$primary_{i,h,v}$, $h = 0$:

i=0	**1**	**0**	**0**	...
i=1	**0**	**1**	**0**	...
i=2	**0**	**0**	**1**	...
v	0	1	2	...

$primary_{i,h,v}$, $h = 1$:

i=0	**0**	**1**	**0**	...
i=1	**0**	**0**	**1**	...
i=2	**0**	**0**	**0**	...
v	0	1	2	...

$SendPrepReq^{t}_{i,h,b,v}$, for $i = 0$, $h = 0$, $b = 0$, $v = 0$:

SendPrepReq(i=0)	**0**	0	1	0	0	0	0	0	...
t	0	1	2	3	4	5	6	7	...

$sentPrepReq^{t}_{i,h,b,v}$, i=0, $h, b, v = 0$:

(i=0)	**0**	0	0	1	1	1	1	1	...
t	0	1	2	3	4	5	6	7	...

$recvdPrepReq^{t}_{i,j,h,b,v}$, for i=0,j=0, $h, b, v = 0$:

-	**0**	0	0	1	1	1	1	1	...
t	0	1	2	3	4	5	6	7	...

$recvdPrepReq^{t}_{i,j,h,b,v}$, i=0,j=1, $h, b, v = 0$:

-	**0**	0	0	1	1	1	1	1	...
t	0	1	2	3	4	5	6	7	...

6. Future works and research directions

6.1. Final considerations

6.2. Possible improvements and extensions

Timeout are extensively criticized in the literature. In this sense, removing the main whole of the Primary/Speaker in proposing specific sets of transactions is surely a step to be considered. In particular, the idea of Redundant rounds of agreements, as proposed in RBFT: Redundant Byzantine Fault Tolerance **?** is a mechanism to be considered.

Acknowledgements

POA Network ,PegaSys, et all.