Security Analysis on dBFT protocol of NEO

Q. Wang , J. Yu , Z. Peng , V. Bui , S. Chen , Y. Ding , and Y. Xiang

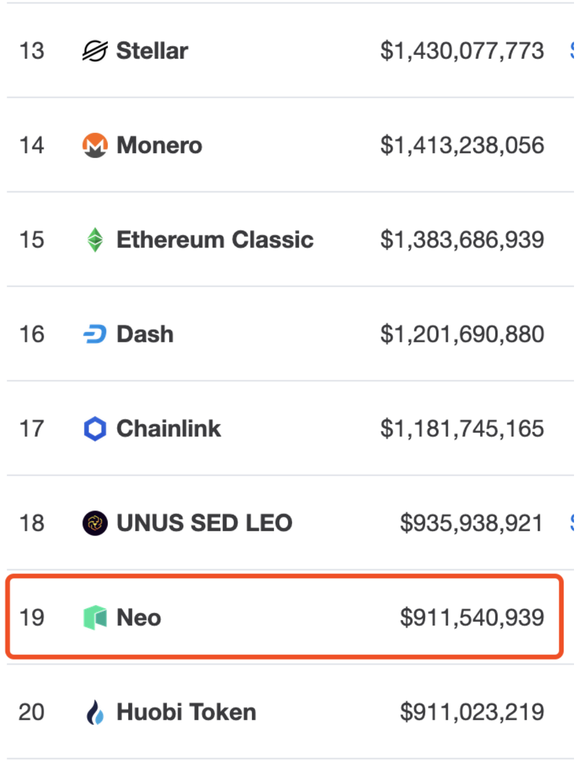
FINANCIAL CRYPTOGRAPHY AND DATA SECURITY 2020

QIN WANG

Contents

- Introduction
  - *NEO project*
  - *dBFT Consensus Mechanism*
  - *Contributions*
- Overview of PBFT
- Detailed dBFT
- Two Attack Cases on dBFT
- Recommendations
- Summary

Introduction — NEO project



13	Stellar	$1,430,077,773
14	Monero	$1,413,238,056
15	Ethereum Classic	$1,383,686,939
16	Dash	$1,201,690,880
17	Chainlink	$1,181,745,165
18	UNUS SED LEO	$935,938,921
19	Neo	$911,540,939
20	Huobi Token	$911,023,219

- Rebranding from the Antshares
- Top-ranked blockchain platforms[1] by its market capitalization[2] in the world.
- Earliest and the longest-running public chain in China.
- Matured ecosystem with DApps

[1] NEO webpage: https://neo.org/ [2] https://coinmarketcap.com/currencies/neo/. Data fetched on 21st Sept. 2019.

Introduction — dBFT protocol

Core component

dBFT (delegated Byzantine Fault Tolerance) consensus mechanism

Widely adopted

Adopted by the Ontology platform

Variant from PBFT

A variant of PBFT, with the modifications on

- procedure of commit (from 3-phase to 2-phase)
- network model (from Client/Server to P2P)
- Leader election (change rules)

Research Question

Is there any security problems of dBFT caused by these modifications?

(especially from *3-phase to 2-phase*)

Introduction — contributions

- The overview of PBFT protocol.
- Clear presentation of dBFT based on its source code [1] comparison towards PBFT.
- Vulnerbilities with no more than $\left\lfloor \frac{n}{3} \right\rfloor$ nodes,
  a) Primary to be Byzantine,
  b) Network delay to make times out.
- Recommendations to fix the identified problems.

Communication with NEO team.

[1] Git commit: 5df6c2f05220e57f4e3180dd23e58bb2f675457d

Overview of PBFT

- Practical Byzantine Fault Tolerance (PBFT)

- The most prevailing BFT protocols in permissioned blockchains.
  E.g. *Hyperledger Fabric v0.5/v0.6*
  Hyperledger Sawtooth v1.0

- Three entities contained in PBFT:
  Client , Primary, Replica

- Three phases involved in the protocol:
  Pre-Prepare , Prepare , Commit

[1] Castro, M., Liskov, B. : Practical byzantine fault tolerance. OSDI'99

Overview of PBFT

2f + 1
PREPARE

2f + 1
COMMIT

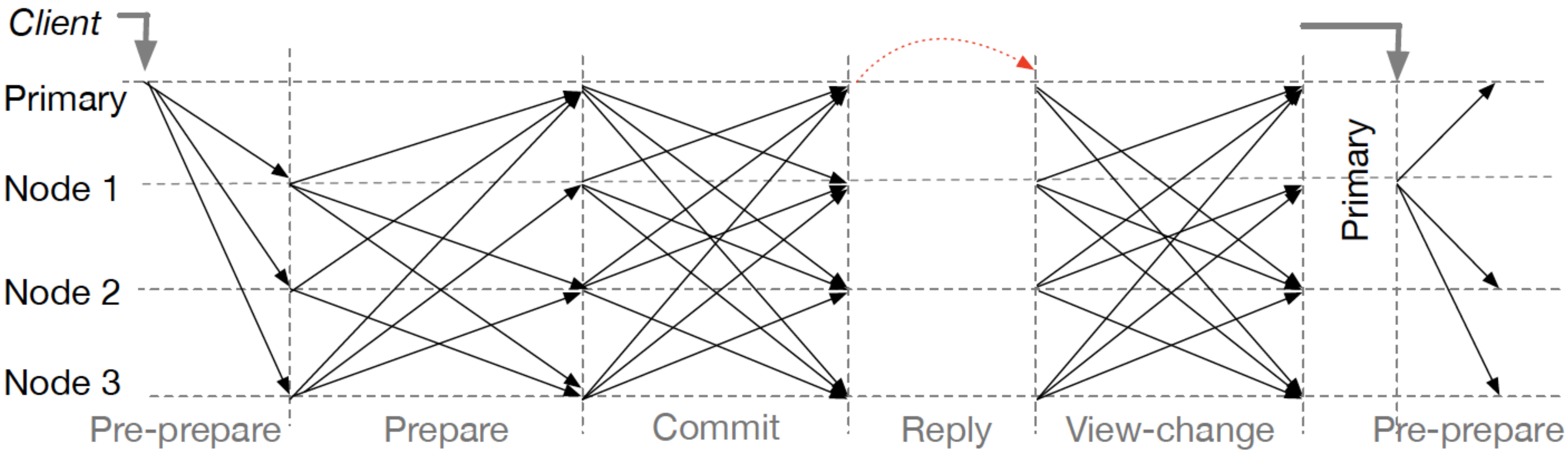


Fig. 1. PBFT Protocol

Detailed dBFT

Network Assumption

partially synchronous network [1]

A message sent from an honest node will eventually arrive within a fixed time-bound, but the bound is unknown.

Safety

It means that the system behaves like a centralized implementation to maintain a total order sequence of decisions.

Liveness

It means that clients eventually receive replies to their requests.

[1] Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of the ACM (JACM) 35(2), 288{323 (1988)

Detailed dBFT

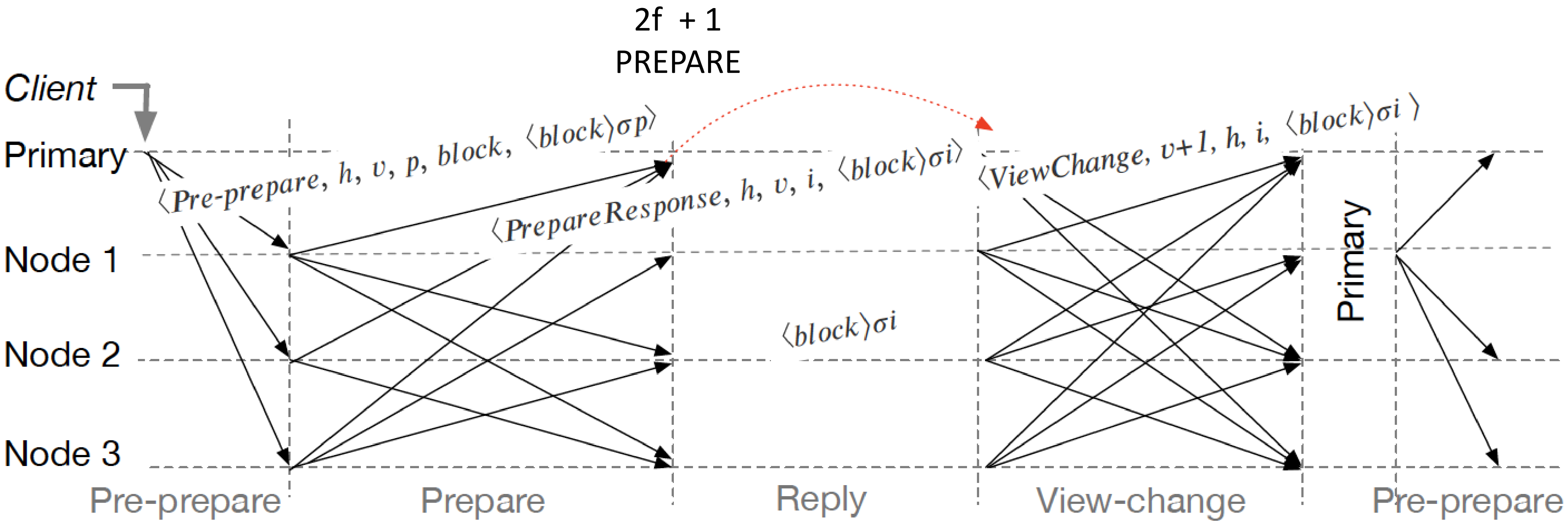
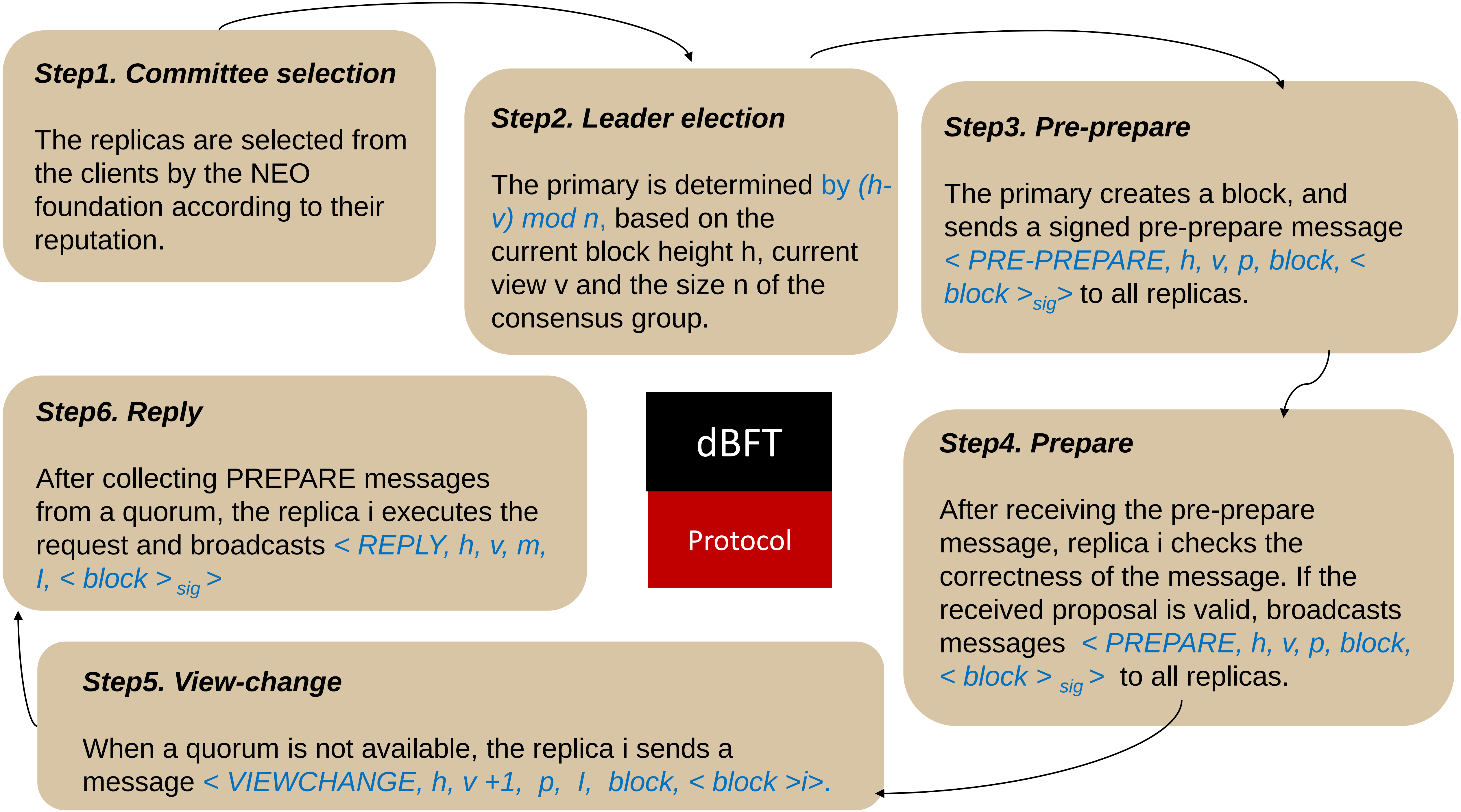

2f + 1
PREPARE
Client
Primary
Node 1
Node 2
Node 3
⟨Pre-prepare, h, v, p, block, ⟨block⟩σp⟩
⟨PrepareResponse, h, v, i, ⟨block⟩σi⟩
⟨block⟩σi
⟨ViewChange, v+1, h, i, ⟨block⟩σi ⟩
Primary
Pre-prepare
Prepare
Reply
View-change
Pre-prepare

Fig. 2. dBFT Protocol



dBFT

Protocol

Step1. Committee selection

The replicas are selected from the clients by the NEO foundation according to their reputation.

Step2. Leader election

The primary is determined by *(h-v) mod n*, based on the current block height h, current view v and the size n of the consensus group.

Step3. Pre-prepare

The primary creates a block, and sends a signed pre-prepare message *< PRE-PREPARE, h, v, p, block, < block >*$_{sig}$*>* to all replicas.

Step4. Prepare

After receiving the pre-prepare message, replica i checks the correctness of the message. If the received proposal is valid, broadcasts messages *< PREPARE, h, v, p, block, < block >* $_{sig}$ *>* to all replicas.

Step5. View-change

When a quorum is not available, the replica i sends a message *< VIEWCHANGE, h, v +1, p, I, block, < block >i>.*

Step6. Reply

After collecting PREPARE messages from a quorum, the replica i executes the request and broadcasts *< REPLY, h, v, m, I, < block >* $_{sig}$ *>*

Comparison with PBFT

Protocols phases	removes the core *Commit* phase from the PBFT, removes the auxiliary protocols including *GarbegeCollection* and *Checkpoint*
Communication model	Y: peer-to-peer network topology N: client-server communication model
Message authentication	Y: digital signatures N: MAC as in PBFT
Consensus committee	Y: (h-v) mod n N: v mod n as in PBFT

Identified Attacks

Same

Both attacks only require no more than f malicious replica

Both attacks has a Byzantine node

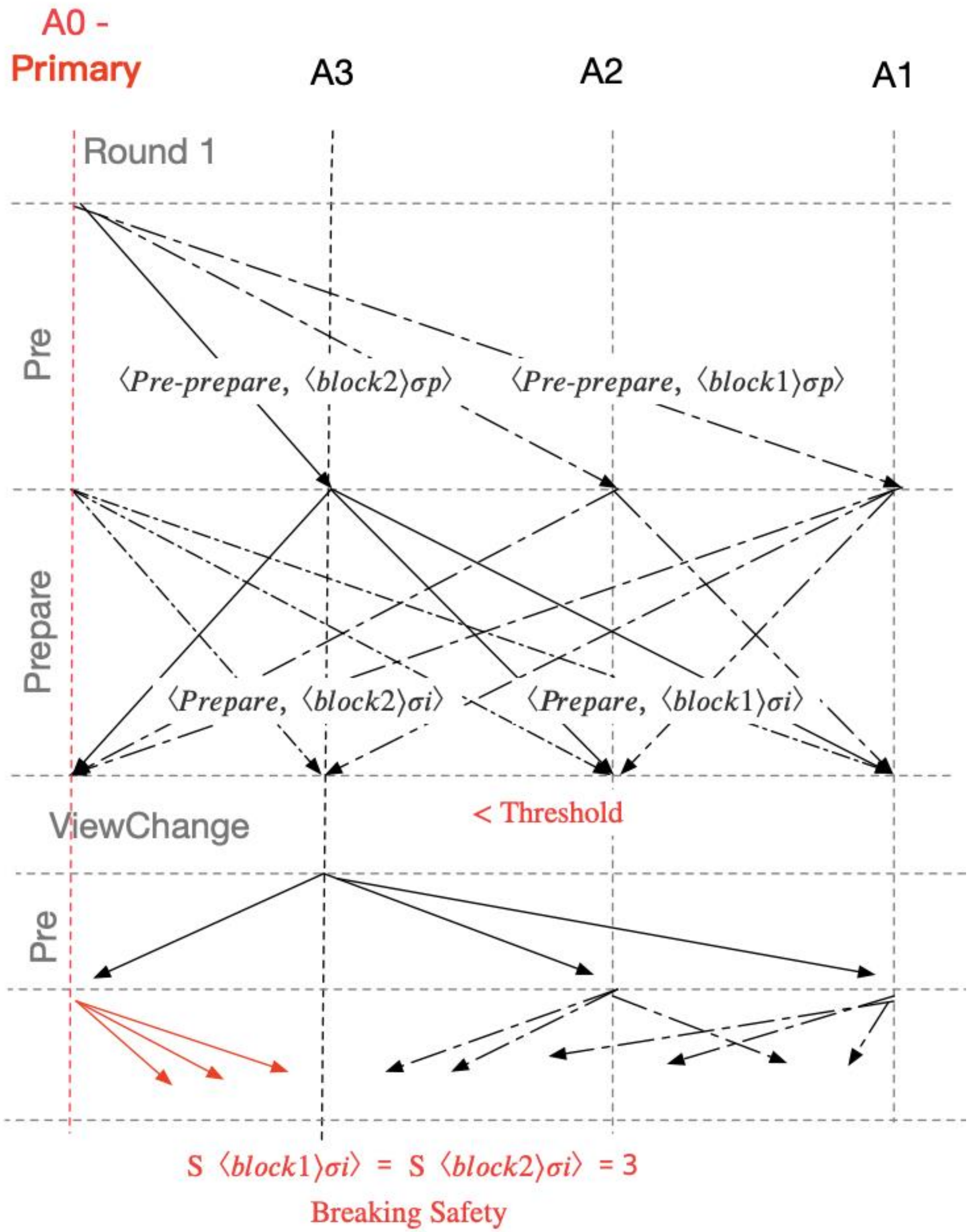
Both attacks need to enforce a view change.

Difference

The first attack requires a Byzantine node to be the primary and fake the states

The second attack requires network delay to trigger the view change, while the Byzantine node only postpone the collected responses

Case 1


A0 - Primary
A3
A2
A1
Round 1
Pre
⟨Pre-prepare, ⟨block2⟩σp⟩
⟨Pre-prepare, ⟨block1⟩σp⟩
Prepare
⟨Prepare, ⟨block2⟩σi⟩
⟨Prepare, ⟨block1⟩σi⟩
< Threshold
ViewChange
Pre
S ⟨block1⟩σi⟩ = S ⟨block2⟩σi⟩ = 3
Breaking Safety

Identified Attacks — case1

Step1

Byzantine primary A0 creates two blocks, block1 and block2. A0 then sends <Pre-prepare> on block1 to A1 and A2, and sends <Pre-prepare> on block2 to A3.

Step2

A1 and A2 will broadcast a <Prepare> message on the block1, A3 will broadcast a <Prepare> message on block2.

Step3

Since no replica receives enough valid <Prepare> message $(2f + 1)$ from a quorum, the current round will timeout, triggering the *ViewChange*.

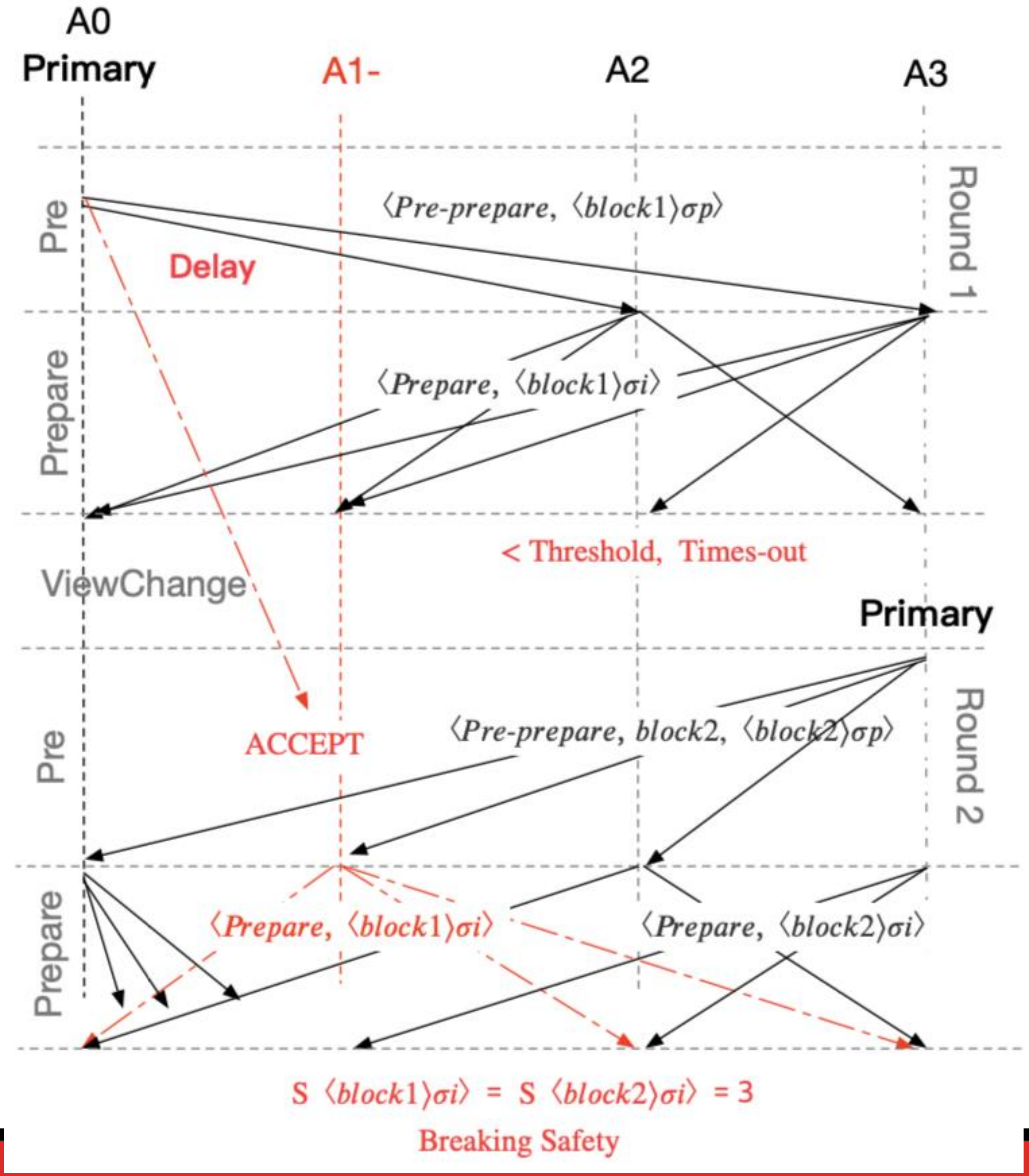
Step4

Following $(h - 1) \bmod 4 = 3$, A3 will be elected as the primary.

Step5

Run the consensus on block2 with $v = 1$. When a decision is reached, A0 can create a conflict decision by releasing $2f + 1 = 3$ valid <Prepare> messages on block1. This breaks the consensus safety.

Case 2


A0
Primary
A1-
A2
A3
Pre
Delay
⟨Pre-prepare, ⟨block1⟩σp⟩
Round 1
Prepare
⟨Prepare, ⟨block1⟩σi⟩
< Threshold, Times-out
ViewChange
Primary
Pre
ACCEPT
⟨Pre-prepare, block2, ⟨block2⟩σp⟩
Round 2
Prepare
⟨Prepare, ⟨block1⟩σi⟩
⟨Prepare, ⟨block2⟩σi⟩
S ⟨block1⟩σi⟩ = S ⟨block2⟩σi⟩ = 3
Breaking Safety

Identified Attacks — case2

Step1

The honest leader sends a valid proposal <Pre-prepare> on block1 .

Step2

If it only receiving two signed messages due to the network delay, Byzantine replica does not react.

Step3

Since no replica receives enough valid <Prepare> message (2f + 1) from a quorum, the current round will timeout, triggering the *ViewChange*.

Step4

Following (h - 1) mod 4 = 3, The normal replica A3 will be elected as the primary.

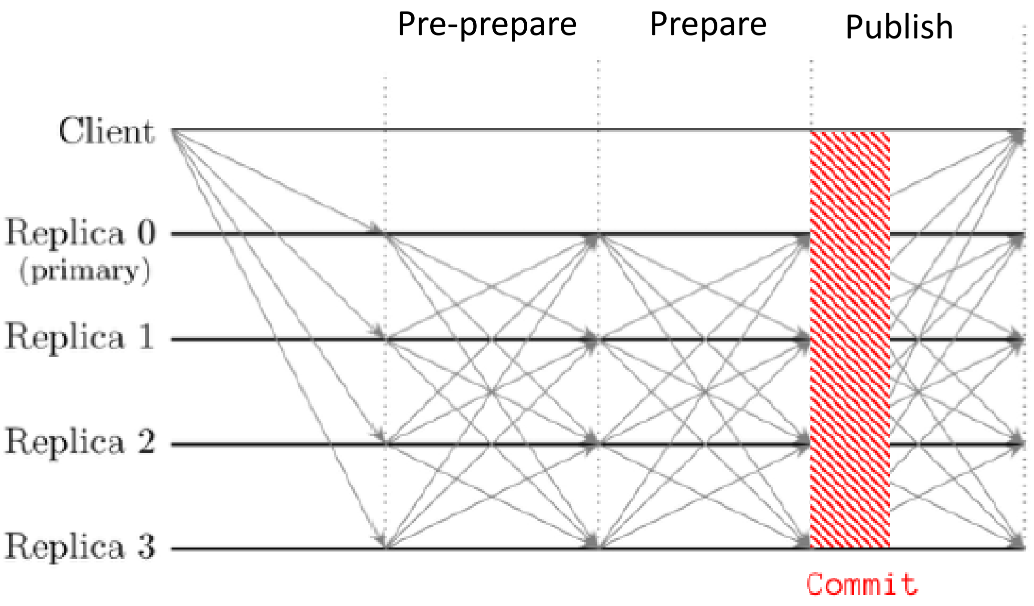
Step5

Byzantine replica A3 releases the two signed <Prepare> messages on block1 collected in the previous view, together with its signed <Prepare> message also on block1 (2f + 1 = 3 valid <Prepare> on block1).
This breaks the consensus safety.

Recommend Fixes

Commit : 2f+1 replicas
have responded to the prepare, or
are ready to move on / roll back decisions.

If at least 2f +1 valid commits messages are collected, then the replica updates the local state of the blockchain by including the block into it, and broadcasts the result.

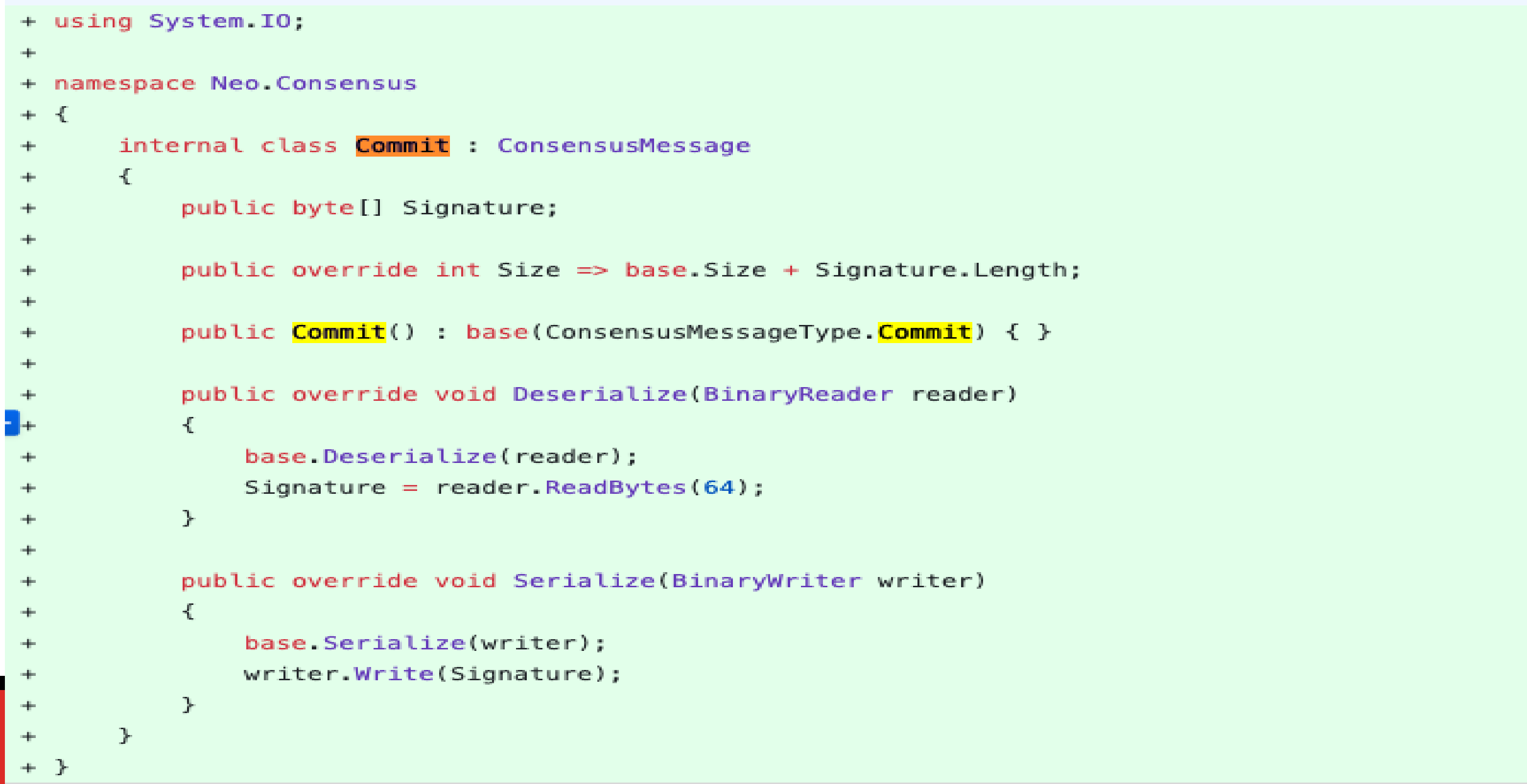


Recommend Fixes

[1] https://github.com/neo-project/neo/tree/master/neo

[2] https://github.com/neo-project/neo/pull/547/files

Fixed has been accepted and applied to NEO project.



```
+ using System.IO;
+
+ namespace Neo.Consensus
+ {
+     internal class Commit : ConsensusMessage
+     {
+         public byte[] Signature;
+
+         public override int Size => base.Size + Signature.Length;
+
+         public Commit() : base(ConsensusMessageType.Commit) { }
+
+         public override void Deserialize(BinaryReader reader)
+         {
+             base.Deserialize(reader);
+             Signature = reader.ReadBytes(64);
+         }
+
+         public override void Serialize(BinaryWriter writer)
+         {
+             base.Serialize(writer);
+             writer.Write(Signature);
+         }
+     }
+ }
```

Summary

Protocol

- We provide the first clear presentation of the widely adopted dBFT consensus mechanism, based on its source code [1]
  git commit 5df6c2f05220e57f4e3180dd23e58bb2f675457d

Vulnerabilities

- We identify two attacks on dBFT. Both attacks are feasible with no more than $\left\lfloor \frac{n}{3} \right\rfloor$ nodes.

Fix

- We provide recommendations to fix the identified problems.

[1] Neo source code: https://github.com/neo-project/neo/tree/master/neo

References

1.Discussion and improvement on dbft (2019), https://github.com/neo-project/neo/pull/320
2. Discussion and improvement on dbft (2019), https://github.com/neo-project/neo/pull/547
3. Hyperledger fabric (2019), https://cn.hyperledger.org/projects/fabric
4. Hyperledger sawtooth (2019), https://cn.hyperledger.org/projects/sawtooth
5. Neo source code on github (2019), https://github.com/neo-project/neo/tree/master/neo
6. Akkoyunlu, E.A., Ekanadham, K., Huber, R.V.: Some constraints and tradeos in the design of network communications. SIGOPS Oper. Syst. Rev. 9(5), 67{74 (Nov 1975). https://doi.org/10.1145/1067629.806523
7. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman:Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference. p. 30. ACM (2018)
8. Cachin, C., Vukolic, M.: Blockchain consensus protocols in the wild. arXiv preprint arXiv:1707.01873 (2017)
9. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999. pp. 173{186 (1999).
10. Decker, C., Seidel, J., Wattenhofer, R.: Bitcoin meets strong consistency. In: Proceedings of the 17th International Conference on Distributed Computing and Networking. p. 13. ACM (2016)
11. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of the ACM (JACM) 35(2), 288{323 (1988)
12. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. Communications of the ACM 61(7), 95{102 (2018)
13. Fischer, M.J., Lynch, N.A., Paterson, M.: Impossibility of distributed consensus with one faulty process. J. ACM 32(2), 374{382 (1985). https://doi.org/10.1145/3149.214121, https://doi.org/10.1145/3149.214121
14. Gilbert, S., Lynch, N.A.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 33(2), 51 59 (2002). https://doi.org/10.1145/564585.564601, https://doi.org/10.1145/564585.564601
15.Ittai Abraham, Guy Gueta, D.M.J.P.M.: Revisiting fast practical byzantine fault tolerance: Thelma, velma, and zelma (2018), https://arxiv.org/abs/1801.10022
16. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: speculative byzantine fault tolerance. In: ACM SIGOPS Operating Systems Review. vol. 41, pp. 45{58. ACM (2007)
17. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008), https:// bitcoin.org/bitcoin
18. Natoli, C., Yu, J., Gramoli, V.,P.: Deconstructing blockchains: A comprehensive survey on consensus, membership and structure (2019)
19. NEO: Neo github (2018), https://github.com/neo-project
20. NEO: Neo whiteopaper (2018), http://docs.neo.org/zh-cn/whitepaper.html

Thanks!