Notes on Byzantine Consensus and Reliable Broadcast

Mahdi Zamani
mzamani@visa.com

Byzantine Consensus (aka, Byzantine Agreement [LSP82]). n parties, each starting with an initial value, want to reach agreement on a single value despite the malicious behavior of up to t of them who can deviate arbitrarily from the protocol. The protocol must satisfy the following properties:[1]

- *Agreement:* All honest parties output the same value.
- *Validity*: If all honest parties start with the same initial value v, then all honest parties output v.
- *Termination*: All honest party eventually decide on a value.

An stronger notion of consensus, known as *strong consensus* [Nei94], can be achieved by providing the following guarantee:

- *Strong Validity*: If the honest parties decide on v, then v is the input of some honest party.

When messages can be authenticated (*e.g.*, via digital signatures and a public-key infrastructure), then Byzantine consensus is known as *authenticated* Byzantine consensus.[2]

Blockchain Consensus (aka, State-Machine Replication [Sch90]). In the context of blockchain protocols, consensus typically means consensus on a sequence of values (*i.e.*, a ledger) which was traditionally known as Byzantine fault tolerant (BFT) *state-machine replication (SMR)* [Sch90, CL99] and *atomic broadcast* in the literature [HT93, CV17]. In addition to the Byzantine consensus guarantees stated above, blockchain consensus provides the following guarantee:

- *Total Order:* An honest party outputs m_1 before m_2 if and only if the sender sends m_1 before m_2.

Reliable Broadcast (aka, Byzantine Generals Problem [LSP82]). The parties want to agree on a message sent by a *distinguished* party, referred to as the *leader* (aka, the *general* or the *dealer*), while providing the following guarantees:

- *Validity:* If the leader is honest and proposes v, then all honest parties output v.
- *Integrity:* Every honest party decides on at most one value which is the input of some party.
- *Termination:* Every honest parties eventually outputs on some value.

Unlike Byzantine consensus which is only solvable when $t < n/2$, reliable broadcast is also solvable for $t \geq n/2$ [LSP82] in the authenticated setting.

[1] Byzantine agreement may refer to multiple problems in the literature (*e.g.*, compare [DFF$^+$82, LLR02]). Here, we adopt the definition used more frequently as in [LLR02, KS16, GK18].

[2] This requires that the author of a signed message be determined by anyone holding the message, regardless of where it came from (this is often called the *non-repudiation* property). Therefore, message authentication codes (MACs) cannot be solely used to provide authenticated consensus.

Interactive Consistency (IC) [PSL80]. A variant of Byzantine consensus where all honest parties want to agree on a vector of their inputs, one for each participant. More formally:

- *Agreement*: All honest parties output the same array of values $A[v_1, ..., v_n]$.
- *Validity*: If a party i is honest and its input is v_i, then all honest parties agree on v_i for $A[i]$. If party i is faulty, then honest parties can agree on any value for $A[i]$.
- *Termination*: Each honest party must eventually decide on A.

Equivalence of Consensus Variants. If any of the consensus variants (Byzantine consensus, blockchain consensus, and interactive consistency) has a solution, then all of them have a solution. For example, IC can be achieved from Byzantine consensus by running it n times, one with each participant as the leader. Also, consensus can be achieved from IC by running IC, then outputting the first element of the vector. If a broadcast primitive is available, then Byzantine consensus becomes trivial for $t < n/2$: Each party simply broadcasts its input, and then honest parties decide on the majority value.

Solvability of Consensus and Broadcast. The well-known results are as follow:

1. Byzantine consensus is impossible to achieve when $t \geq n/3$ in the unauthenticated setting [LSP82] and when $t \geq n/2$ in any setting [Fit03]. Authenticated consensus is solvable when $t < n/2$ [LSP82].

2. *FLP Impossibility [FLP85]:* Deterministic consensus in the asynchronous setting is impossible even with one crash failure. This is because it is impossible to distinguish a failed process from a very slow one, so processes remain ambivalent when deciding. To avoid this impossibility and/or to achieve practicality, the protocol may be allowed to provide probabilistic safety or liveness guarantees. In the latter case, the protocol is said to provide *eventual liveness* (or *eventual consistency*) since parties can decide on a value with probability one but with no time bound.

3. *Bracha-Toueg Impossibility [BT83]:* There is no (probabilistic) asynchronous consensus protocol resilient to $t \geq n/3$ malicious parties.

4. *Dolev-Strong Impossibility [DS83]:* $t + 1$ rounds are necessary and sufficient for *deterministic* authenticated broadcast.

5. *Dolev-Reischuk Impossibility [DR82]:* Any (deterministic) authenticated BA requires exchanging $\Omega(nt)$ messages.

Safety and Liveness Guarantees. In the context of state-machine replication, *safety* (aka, *consistency*) means that all honest parties agree on the same outcome. *Liveness* means that the honest parties will make progress during the course of the protocol. Safety can be trivially achieved by making no forward progress at all from an initial state. Liveness can be trivially achieved by making progress unsafely by just letting each party decide on arbitrary outcomes. Neither of these guarantees are useful by themselves. Typically, the agreement property of a consensus protocol provides safety, while the validity and termination guarantees together provide liveness [GK18].

Consensus Termination. A consensus protocol terminates when all honest participants have terminated. If all honest participants terminate in the same round, then we call they have achieved an *immediate agreement*. Otherwise, we call they have achieved an *eventual agreement* [Fis83].

Dolev-Strong Byzantine Agreement [DS83]. They show that this can be achieved by exchanging $O(nt)$ messages using the following algorithm:

1. In each round, each party signs the value and sends it to only the participants who have not signed it yet.
2. After t rounds, each message has t signatures on it. Hence, each correct value has been seen by all the correct participants after $t + 1$ rounds.

Dolev-Strong Reliable Broadcast [DS83]. Lamport *et al.* [LSP82] propose an exponential computation and communication protocol for reliable broadcast in the information-theoretic setting. Dolev and Strong [DS83] propose a polynomial-time broadcast protocol using signatures as follows [GK18]: In the first round, the sender digitally signs and sends its message to all the other parties, while in subsequent rounds parties append their signatures and forward the result. If any party ever observes valid signatures of the sender on two different messages, then that party forwards both signatures to all other parties and disqualifies the sender (and all parties output some default message).

View-Change Protocols. Some consensus protocols such as Paxos [Lam98], Raft [OO14], and PBFT [CL99] proceed in *epochs* (a.k.a., *views*), where a *leader* (a.k.a., the *primary*) is responsible for the progress of the protocol. When a party (a.k.a., a *replicate* or *backup*) suspects that the leader is faulty, the party broadcasts a View-Change message to replace the leader. The new leader has to gather at least $2f + 1$ signed View-Change messages, to broadcast a New-View message. A Byzantine leader cannot violate safety (*i.e.*, cause inconsistencies) but can violate liveness (*i.e.*, prevent progress) by not proposing. In PBFT [CL99], every honest participant monitors the current leader, and if a new message is not proposed after some time, the honest participant requests a new leader.

References

[BT83] Gabriel Bracha and Sam Toueg. Resilient consensus protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, PODC '83, pages 12–26. ACM, 1983.

[CL99] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, 1999.

[CV17] Christian Cachin and Marko Vukolic. Blockchain consensus protocols in the wild. *CoRR*, abs/1707.01873, 2017.

[DFF$^+$82] Danny Dolev, Michael J. Fischer, Rob Fowler, Nancy A. Lynch, and H. Raymond Strong. An efficient algorithm for Byzantine agreement without authentication. *Information and Control*, 1982.

[DR82] Danny Dolev and Ruediger Reischuk. Bounds on information exchange for Byzantine agreement. In *PODC '82: Proceedings of the first ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 132–140. ACM, 1982.

[DS83] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[Fis83] Michael Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *Fundamentals of Computation Theory*, pages 127–140, 1983.

[Fit03] Matthias Fitzi. generalized communication and security models in Byzantine agreement. PhD Thesis, ETH Zurich, ftp://ftp.inf.ethz.ch/pub/crypto/publications/Fitzi03.pdf, 2003.

[FLP85] M.J. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[GK18] Juan Garay and Aggelos Kiayias. Sok: A consensus taxonomy in the blockchain era. Cryptology ePrint Archive, Report 2018/754, 2018. https://eprint.iacr.org/2018/754.

[HT93] Vassos Hadzilacos and Sam Toueg. Distributed systems. chapter Fault-tolerant Broadcasts and Related Problems, pages 97–145. ACM Press/Addison-Wesley Publishing Co., 1993.

[KS16] Valerie King and Jared Saia. Byzantine agreement in expected polynomial time. *J. ACM*, 63(2):13:1–13:21, March 2016.

[Lam98] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[LLR02] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the composition of authenticated byzantine agreement. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 514–523. ACM, 2002.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[Nei94] Gil Neiger. Distributed consensus revisited. *Inf. Process. Lett.*, 49(4):195–201, February 1994.

[OO14] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320. USENIX Association, 2014.

[PSL80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[Sch90] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, December 1990.