ThunderCore

(LitePaper)

Thunder Research

Rafael Pass and Elaine Shi

August 29, 2018

1 Introduction

Today, we are relying on, and trusting, powerful companies (e.g., VISA, Facebook, Uber, Wells Fargo) with our data, our ability to financially transact, and engage in businesses with each other. We don't have the option of not "trusting" them and "verifying" how they operate. The emergence of cryptocurrencies such as Bitcoin and Ethereum bring forth the promise of a new "decentralized" Internet, which is more *transparent*, *fair*, and *secure*.

- *Transparency*: In a decentralized system, the rules of the game are public; anyone can verify the validity of transactions and computations (i.e., computer code), and users/stake-holders are not at the whim of a CEO of some company.
- *Fairness:* There are no entry barriers (e.g., you don't need a bank account to transact), censorship is impossible (e.g., money can't be frozen), and anyone participating gets treated in the same way.
- *Security:* Breaking the security of these protocols requires controlling a large fraction of the participating nodes. This is in contrast to currently standard "trusted-third party" solutions where a single company by either volition, or if hacked, can completely compromise the security of the entire system.

These properties create exciting new opportunities for decentralized applications and mechanisms to incentivize entities and individuals to collaborate and transact together.

The innovation that enables this development is the notion of a *blockchain*—that is, a method for maintaining an *append-only, linearly-ordered, list of data* (e.g., transactions). This notion, together with that of a *smart contract*—and in particular *expressive* (i.e., fully programmable) smart contracts as in Ethereum's systems—are central to the potential of realizing the above-mentioned promise: We want decentralization not only for payment systems, but rather to enable the above features for *general* applications. Indeed, in the last years, there has been an abundance of, so-called, DApps (i.e., *Decentralized Apps*) created that operate on Ethereum's virtual machine (EVM). These include decentralized exchanges and games (such as CryptoKitties).

The Bottlenecks: Scalability + Confirmation Times The emergence of DApps such as Cryptokitties put a heavy toll on the Ethereum network, which resulted in transaction fees and confirmation latencies escalating. The reason for this is the following:

- **Low Throughput/Poor Scalability:** Existing blockchains as employed by major crypto-currencies (e.g., Bitcoin or Ethereum) require flooding the whole network with all transactions. As a consequence, these blockchains do not scale well to handle a large transaction volume. For instance, Bitcoin and Ethereum can handle less than 15 transactions/second. This small throughput severely hinders a wide-spread adoption of such crypto-currencies, and prevents a large-scale use of DApps. As a comparison, VISA handles 2000 transactions/sec (but of course is not decentralized and does not enable DApps).

Although the throughout problem is the most major problem plaguing current blockchains, there is also another issue that severely hinders some smart contract/DApps applications:

- **Slow Confirmation Times:** The security of Nakamoto's blockchain protocol requires the "block-time" (i.e., the average time until new blocks are created) to be significantly larger than the maximum network latency [19]; this is why in Bitcoin, new blocks of transactions get mined every 10 minutes, and a transaction is only considered confirmed once it has been incorporated in a block and there are 5 subsequent blocks following it; thus it takes on average one hour for a transaction to be confirmed (with low confidence). While Ethereum uses a smaller block-time, the average confirmation time still remains on the order of one hour. These long confirmation times hinder/or renders impractical several important smart contract/Dapp applications. Indeed, for most "standard" centralized apps, users expect "instant" responses.

Thunder: "Decentralization with the Benefits of Centralization" As we have seen, *centralized* solutions currently have many benefits over decentralized solutions: a) they handle a large volume of transactions/data, and b) they "almost instantly" confirm transactions and are thus more amenable to apps which require "instant" responses. Yet, they fail to satisfy the above-mentioned transparency, fairness and security desiderata, which we believe are crucial for a next generation Internet.

The Thunder paradigm provides a bridge between the two: it is a fully decentralized method for maintaining a public EVM-compatible blockchain, with the performance of centralized solutions: Our **Thunder** protocol is a new blockchain that:

- achieves both *high throughput* and *fast confirmation time*;
- is robust up to a 50% attack (just as standard blockchains);
- directly supports EVM-smart contracts and EVM-DApps designed for Ethereum with little or no modifications;
- comes with a mathematical proof of its security.

We believe it is the simplest, fastest, most expressive and secure blockchain solution out there.

While we are aware of several other recently suggested approaches for overcoming the above two bottlenecks, these approaches are either restricted to only payments (and do not handle general

smart contracts/DApps), or provide security only against a weaker 1/3-attack. Additionally, most of these approaches are based on heuristic arguments, whereas our protocol is accompanied by a rigorous proof of security; see Section 4 for more details.

The design of Thunder is inspired by a new theoretical paradigm for consensus proposed in our earlier work, Thunderella [20], and provides the first practical instantiation of this general paradigm.

2 The Thunder Protocol in a Nutshell

Our key idea is to combine a centralized approach with a decentralized approach, and thereby achieve the scalability and speed of centralized approaches, while maintaining the decentralized nature of the blockchain.

Following our Thunderella paradigm [20], the Thunder protocol combines any "standard" blockchain—which we will refer to as the *slow chain*—with an *optimistic "fast-path"*. The fast path protocol is executed by a *committee of stake-holders* and coordinated by a central authority called the Accelerator. The Accelerator's job is to linearize transactions and data (which is a task that is hard to do in a decentralized way, yet easy for a central authority).

As long as a) the Accelerator is correctly doings its job, b) network conditions are good, and c)$>$ 3/4 of the committee are functioning correctly, we barely need to use the slow-chain at all; instead, transactions can be confirmed on the fast-path by the committee using an extremely simple and lean fast-path protocol. This protocol offers both high throughput and instant confirmation of transactions.

But the fast-path only confirms new transactions assuming the above-mentioned good conditions are met. When they are not (e.g., the Accelerator misbehaves or is under attack), we leverage the "slow chain" to make sure that no damage can be done to the blockchain, and then to recover in a provably sound manner. This results in a protocol that:

- **Under optimistic conditions** (that is, when the network is operating properly and the systems is not under a major attack) support high throughput and instant confirmation.
- **Under worst-case conditions**, the blockchain remains secure even if the Accelerator is get fully corrupted, as long as a) a majority of the stake-holders are honest, and b) the underlying "slow-chain" remains secure.

To select the committee of stake-holders, we again rely on the slow chain. Anyone having stake in our system, can put down an "escrow"—a so-called "stake-in"—onto the slow chain and that qualifies the entity to become a committee member. Next, we subselect a committee of 500 nodes (using a provably fair mechanism) among the entities that have put down the stake-in. Both committee members and the Accelerator are rewarded for their work, both in terms of transaction fee and from a pre-allocated "participation pool" (analogous to Bitcoin's and Ethereum's mining rewards).

In our initial deployment, we are relying on Ethereum's blockchain as our underlying "slow-chain"; this blockchain currently is based on proof-of-work, but there are ongoing plans to replace it with a proof-of-stake system. In our initial deployment, we will also only rely on a single Accelerator; in later versions, we will extend the system to support multiple shards, each having

their own dedicated Accelerator. Allowing anyone to create their own shards will also help establish a healthy ecosystem where accelerators are incentivized to provide faster and more reliable service (and accelerators providing poor service will die out). We here content ourselves with providing an overview of the single Accelerator system.

As mentioned, our first release will be fully EVM compatible to enable direct migration of existing Ethereum DApps.

3 The Thunder Protocol

We here provide a more detailed high-level description of the Thunder Consensus protocol.

3.1 Consensus and Blockchains

Roughly speaking, a blockchain is simply a permissionless implementation of a primitive known as *state machine replication*, or *consensus* in the distributed systems literature.

Consensus: A consensus protocol enables maintaining a "linearly ordered log" of transactions in a distributed fashion such that the following two properties hold:

- **Consistency:** at any point in the execution, all honest participants have consistent logs of transactions—that is, either their logs are identical, or one participant's log is a prefix of the other's.
- **Liveness**: any honest protocol participant can propose to add a transaction; this transaction is then guaranteed to be incorporated into their logs within some fixed, small amount of time; additionally, whenever a participant sees some transaction in their log, the same transaction will appear in every other participant's log within some fixed, small amount of time.

In essence, these properties mean that from the view of the participants, they are communicating with a trusted third party that maintains a *global*, *ordered*, *append-only* log of transactions that anyone can add to—in essence a *public ledger*. Our goal here is to implement a consensus protocol in the permissionless setting with high throughput and fast confirmation of transactions.

Blockchains: A formal definition of a blockchain can be found in [19]. Strictly speaking, a blockchain does not directly give the above-described consistency property; rather, the consistency property defined for a blockchain is that players have a consistent view of the blockchain *when dropping the last κ blocks*, where κ is a *security parameter*: in other words, the last κ blocks may be "unstable", but everything before that has been stabilized (except with exponentially vanishing probability as a function of the security parameter κ). (For Ethereum's blockchain, think of $\kappa = 300$.)

Other than the above-mentioned main properties of consistency and liveness, [19] also defines a notion of *bounded chain-growth*: the length of the blockchain should grow at a well-defined pace: not to fast and not too slow. We will require the use of a slow-chain that satisfies such bounded chain-growth.

Pass et al. [19] show that Nakamoto's protocol satisfies all the above properties once the mining hardness is appropriately set (as a function of the network delay and total computing power) assuming that a majority of the mining is done by honest users (a.k.a. "computational honest majority"). More precisely, the Nakamoto protocol satisfies consistency and liveness *if and only if* the mining-hardness is set as a function of the maximum network latency so that the block-time (which in the Bitcoin system is set to 10 minutes) significantly exceeds the maximum network latency. Ethereum's protocol is related to Nakamoto's and ought to satisfy them as well.

Security Contexts: We will distinguish between two different conditions on the network participants.

- **Worst-case conditions W** under which the protocol is guaranteed to provide consistency and liveness (i.e., **W** are conditions under which the protocol is guaranteed to be *secure*). In our case, **W** will be the conditions that:

  1. the underlying slow-chain is secure (which in turn follows from computational honest majority in the case of Nakamoto's blockchain); and,
  2. a majority of the committee is honest.

  Under such worst-case conditions, confirmation times may be slow, and the transaction load may be limited (just as is the case for Bitcoin and Ethereum).

- **Optimistic conditions O** under which the transaction volume can be large, and transactions are "instantly" confirmed. In our case, **O** will be the conditions:

  1. the slow-chain is secure;
  2. the Accelerator is online and honest;
  3. $> \frac{3}{4}$ of the committee is online and honest.

Our new Thunder protocol will make use of a simple fast-path protocol in conjunction with an underlying slow-chain to ensure worst-case security under **W**, and instant-confirmation and high-throughput under **O**. As mentioned above, the key idea is to make use of the "fast-path" protocol when things work well, and only use the slow-chain to recover from faults.

3.2 A Simple Fast-Path Protocol

Consider first the following simple fast-path protocol:

- We have a designated entity: the Accelerator.

- All new transactions are sent to the Accelerator; the Accelerator bundles up transactions into micro-blocks, signs each micro-block *with increasing sequence numbers* seq, and sends out the signed micro-block to a "committee" of players.

- The committee members "ack" (acknowledge) all Accelerator-signed micro-blocks by signing them, *but* only ack at most one micro-block per sequence number.

- When a micro-block has received acks (i.e., signatures) from more than 3/4 of the committee members, we refer to the micro-block as being **notarized** (and the notarization of the micro-block is this collection of these signatures)[1]. Participants can *directly output* their longest sequence of consecutive (in terms of their sequence numbers) notarized micro-blocks—all transactions contained in them are considered **confirmed**. In this way, the fast-path instantly finalizes transactions.

It is easy to prove (and well-known) that this protocol is consistent under the condition $\mathbf{W}^{\mathbf{fast}}$ = "$> 1/2$ of the committee members are honest"; additionally, it satisfies liveness with instant confirmation under the conditions $\mathbf{O}^{\mathbf{fast}}$ = "1) the Accelerator is online and honest, 2) $> 3/4$ of the committee members are online and honest". In fact, under these optimistic conditions, we only

[1] In an actual implementation, one would make use of an *aggregate signature scheme* to describe the notarization in a compact form.

need 2 communication rounds to confirm a transaction! This approach is extremely practical and indeed similar protocols are often used in practice—for instance `chain.com` uses a related protocol for their permissioned consensus protocol.

The problem with this approach, however, is that the protocol does not satisfy liveness (even "slow" liveness) under condition $\mathbf{W}^{\mathbf{fast}}$. If the Accelerator is cheating (or is simply taken down from the network), the protocol halts. (Indeed, in this case `chain.com` would have to manually fix the issue, which is impossible in a decentralized setting.)

3.3 How to Recover from Faults

To overcome this problem, we leverage the underlying "slow-chain" which we assume satisfies both consistency and liveness. Roughly speaking, we deal with this as follows.

Heartbeats: For every length ℓ of the slow-chain, the Accelerator takes the hash h of the current log (of the fast path) and sends out the tuple (ℓ, h, seq) (where seq is the current sequence number) to get notarized by the committee members (who all check that h is correct based on the history they have seen); we refer to a notarized version of such a tuple (ℓ, h, seq) as a `heartbeat`. The Accelerator is subsequently required to post the `heartbeat` to the slow-chain.

Cool-down: Next, if some user notices that its transaction is not getting confirmed by the accelerator/committee, some "evidence" of this will become apparent on the underlying slow-chain—as we shall see shortly, the evidence is simply the fact that no `heartbeat` has been observed on the slow-chain for a sufficiently long period of time. (Note that by liveness, and the bounded chain growth properties of the slow-chain, if Accelerator and more than 3/4 of the committee are online and honest, we are guaranteed that a heartbeat for length ℓ gets incorporated close to length ℓ). Whenever such evidence of cheating (i.e., no recent heartbeat are contained on the slow-chain) has been found, we enter a "cool-down" period, where committee members stop signing messages from the Accelerator, yet we allow anyone to post any "new" notarized transactions (which were not included in the most recent heartbeat) to the slow-chain. The length of the cool-down period is counted in blocks of the slow-chain (say κ blocks where κ is a security parameter).

Slow mode: Finally, after the cool-down period ends, we can safely enter a "slow period" where transactions *only* get confirmed in the slow-chain blockchain (and the fast-path is no longer active). We stay in the slow-chain for an appropriate amount of time counted in blocks on the slow-chain (e.g., enough blocks to restore or replace a faulty Accelerator) and can next reboot with a new epoch of the fast-path protocol; more details on the reboot step can be found in Section 3.4.

Let us point out the reason for having a cool-down period: Without it, players may disagree on the set of fast-path transactions that have been confirmed before entering the "slow mode", and thus may end up with inconsistent views. The cool-down period enables honest players to post all notarized transactions they have seen to the slow-chain, and thus (slowly) reach consensus on this set of transactions; once we have reached this consistent view (at the end of the cool-down), we can finally fully switch over to confirming new transactions on the slow-chain.

Collecting evidence of cheating: "yell" transactions It remains to explain how to collect evidence that the Accelerator (and/or committee) is cheating or is simply "offline". If a player notices that his or her transaction is not getting confirmed by the Accelerator or committee, he or she can send a special `yell` transaction (which contains the transaction it wants to see confirmed) to the underlying slow-chain. The Accelerator is additionally instructed to confirm all such `yell` transactions it sees on the slow-chain.

Now, if a committee member sees some transactions on the slow-chain that have not gotten notarized within a sufficiently long amount of time—counted in blocks in the slow-chain (say within κ blocks)—they know that the Accelerator (or a large fraction of the committee members) must be cheating or offline. At this point, the committee member will stop signing `heartbeat` messages.

So, as long as just 1/4 of the committee is honest, a cheating Accelerator will be "choked"—no new `heartbeats` by Accelerator will be notarized. Consequently, everyone that observes the slow-chain will enter the "cool-down" phase and subsequently the slow mode.

3.4 How to Reboot the Fast-path

Once we have entered the slow mode where all transactions are being posted on the slow-chain, we need to have a way to return to the fast path. The easiest method is simply to stay in the slow-mode for a fixed (e.g., 100κ) number of blocks on the underlying slow-chain and switch to "fast-path" confirmation after these blocks; the hope is that the accelerator can be repaired during this time (and if it is not, transactions need to be confirmed on the slow path).

3.5 How to Select the Committee

So far we have deferred the discussion of how the committee is selected. There are many approaches for selecting the committee among the stake-holders. We focus here on one particular method.

We require anyone who wishes to serve on the committee to put down an *escrow* on to the slow-chain. Next, we subselect a group of 500 members from the entities having put down an escrow on the slowchain. We warn the reader that although this committee selection approach seems simple, getting it right is non trivial. More precisely, we rely on a method from our earlier work Snow White [10]:

- The committee that is active when the slow chain has a length ℓ is selected based on the stake-holders who have put up an escrow by the time the slow-chain reaches length $\ell - 2\kappa$, and (depending on the selection method used) some nonce taken from the slow-chain at length $\ell - \kappa$.

As explained in more detail in our earlier work [10], the reasons for the 2κ "look-back" is to prevent against attacks where an attacker adaptively selects its keys to have a higher probability of getting elected to the committee.

This approach ensures consistency and liveness under the conditions $\mathbf{W}$ = "1) the slow-chain is secure, 2) $> 1/2$ of the stake holders with a valid escrow are honest"; and achieves fast confirmation under the conditions $\mathbf{O}$ = "1) The Accelerator is online and honest, 2) $> 3/4$ of the stake holders with a valid escrow are online and honest".

4 Comparison with Related Initiatives

We provide some comparisons with related initiatives.

- Offchain payment channels (e.g., Lightning Networks [1] and Raiden networks [2]) and state channels [12, 17] support fast offchain payments, but existing systems offer little to no smart contract programming support; moreover, parties need to engage in a prior setup.

- Several initiatives rely on "classical-style" consensus methods to build a decentralized cryptocurrency. For example, Zilliqa [3] and Tendermint [15] rely on PBFT [9] or its variants to reach consensus. In comparison with Thunder, which combines an asynchronous fast path and a *synchronous* "fallback", PBFT instead relies on an asynchronous recovery path. This means that 1) PBFT can tolerate at most $< \frac{1}{3}$ fraction corruption due to a well-known lower bound [11] (in contrast, our approach tolerates up to a 50% attack); 2) PBFT requires more rounds of voting in the normal path (which is necessary for an asynchronous recovery) and thus is at least twice as slow as our protocol; and 3) in comparison, our approach is conceptually much simpler than PBFT and thus friendlier to implementation and large-scale deployment [13, 18].

- The beautiful work of Algorand [16] designs a new synchronous state machine replication protocol and builds a cryptocurrency on top. Just like ours, their protocol is provably secure. However, it only tolerates up to $\frac{1}{3}$ corruptions; moreover, their approach requires complete redesign of the consensus layer, whereas our approach allows us to accelerate any existing blockchain using a simple fast-path protocol.

- Several works rely on a DAG-based approach [4, 22] (rather than chain-based) to prevent double-spending. These works do not realize our notion of a public ledger; they also do they support general smart contracts/DApps.

- Dfinity's approach [5] relies on offchain voting on blocks to reach finality in "2 blocks of time". Their approach does not improve throughput w.r.t. to the underlying slowchain.

- Ethereum's Casper [7, 8] also relies on offchain voting on blocks, but they do so to checkpoint blocks and not to improve speed and throughput. The elegant Plasma framework by Poon and Buterin [21] and several sharding initiatives [3, 6, 14] are conceptually related to ours. These approaches are orthogonal and complementary to our effort.

5 Acknowledgments

We are very grateful to Vitalik Buterin and the Thunder Team for helpful discussions and insightful feedback.

References

[1] `https://lightning.network/`.

[2] `https://raiden.network/`.

[3] `https://www.zilliqa.com/`.

[4] http://iota.org/.

[5] https://dfinity.org/.

[6] On Sharding Blockchains. https://github.com/ethereum/wiki/wiki/Sharding-FAQ.

[7] Vitalik Buterin. https://medium.com/@VitalikButerin/minimal-slashing-conditions-20f0b500fc6c, 2017.

[8] Vitalik Buterin and Vlad Zamfir. Casper. https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/, 2015.

[9] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, 1999.

[10] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016.

[11] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 1988.

[12] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment channels over cryptographic currencies. *IACR Cryptology ePrint Archive*, 2017:635, 2017.

[13] Pedro Fonseca, Kaiyuan Zhang, Xi Wang, and Arvind Krishnamurthy. An empirical study on the correctness of formally verified distributed systems. In *Proceedings of the Twelfth European Conference on Computer Systems*, EuroSys '17, pages 328–343, New York, NY, USA, 2017. ACM.

[14] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 19–34, 2018.

[15] Jae Kwon. Tendermint: Consensus without mining. http://tendermint.com/docs/tendermint.pdf, 2014.

[16] Silvio Micali. Algorand: The efficient and democratic ledger. https://arxiv.org/abs/1607.01341, 2016.

[17] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. Sprites: Payment channels that go faster than lightning. *CoRR*, abs/1702.05812, 2017.

[18] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 31–42, New York, NY, USA, 2016. ACM.

[19] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Eurocrypt*, 2017.

[20] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. https://eprint.iacr.org/2017/913.pdf.

[21] Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. `https://plasma.io/`.

[22] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. SPECTRE: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.