The Not-So-Short ZILLIQA Technical FAQ

[Version 0.1]

The ZILLIQA Team & The ZILLIQA Community
www.zilliqa.com enquiry@zilliqa.com @zilliqa

December 28, 2017

Abstract

This document is a compilation of questions that have been asked by the ZILLIQA community members across different channels including but not limited to Slack, Telegram and email. This FAQ is not intended to be an introduction to ZILLIQA and hence assumes familiarity with the basics of ZILLIQA's design architecture. We assume that the reader has read one of the following documents: the technical whitepaper, the position paper or the blogposts to be found on the ZILLIQA webpage.

1 Let's talk about ZILLIQA

1. What is ZILLIQA?

ZILLIQA is a high-throughput public blockchain platform designed to scale to thousands of transactions per second.

2. What is the origin of the name ZILLIQA?

ZILLIQA is a play on silica. Just as silicon powers the computing industry, the team hopes ZILLIQA will power the next generation of high-throughput applications.

3. How do you pronounce ZILLIQA?

ZILLIQA is pronounced ['zilikə], where "Zi" is pronounced as in "Zi[nc]" and "ca" is pronounced as in "[sili]ca".

4. What is so cool about the scalability of ZILLIQA?

ZILLIQA processes more transactions per second as more mining nodes join the network. This may sound like a natural thing at first, but it is actually technically challenging to accomplish and certainly hasn't been accomplished by existing blockchains.

5. Please explain linear scalability

Linear scalability in the case of ZILLIQA means that the more miner nodes participate, the more transactions the network can process, and that happens at an almost linear rate.

Most blockchains do not yield higher throughput when the network expands. Rather, the rate of return decreases the larger the network gets. For instance, when new miners join the Ethereum network, transactions have to be broadcast to them as well before being confirmed. This means as more and more miners join the network, each data about a block has to travel a lot more before being part of the blockchain. This makes the network slower just as distributing information to many people is slower than to a small number of people. This is the reason why many solutions to accelerate transaction throughput rely on restricting the number of nodes (e.g., permissioned blockchains). None of those solutions allow linear scaling as ZILLIQA does.

6. Is ZILLIQA PoW-based?

Yes and no. ZILLIQA only leverages PoW to establish mining identities. PoW defends against Sybil attacks and is also used to perform network sharding. Unlike other PoW-based blockchains, ZILLIQA does not use PoW for consensus. PoW is performed only at larger intervals, not by every miner on every block. Thus, ZILLIQA has a much smaller energy footprint.

7. Which consensus protocol is used in ZILLIQA?

ZILLIQA uses an optimized practical Byzantine Fault Tolerant (pBFT) protocol for consensus.

8. Is ZILLIQA the first to use BFT for consensus?

No, NEO, Tendermint and Hyperledger all use variations of BFT.

9. What are the advantages of pBFT over PoW-based consensus?

There are several advantages of using pBFT as a consensus protocol:

1. It is not computationally intensive and hence less energy-consuming than PoW.
2. It can leverage a small consensus group for efficiency.
3. It gives finality to transactions. In other words, unlike PoW-based consensus, where usually 6 confirmations are required, pBFT guarantees that no temporary fork will happen due to the consensus protocol and hence, no confirmation is required.

10. How does pBFT in ZILLIQA give finality to transactions?

The definition of finality of a consensus protocol is that at the end of the protocol, all the honest nodes agree on the proposed block. This property is not achieved in Bitcoin. The reason is the following: a node may do PoW and propose a block. At the same time, another node may also do a PoW on another block and propose it. Now, it is very possible that a subset of honest nodes in the network sees only the first block, while another subset of honest nodes sees only the second. They hence may not agree on the next block. The finality property is not guaranteed here.

In pBFT (the consensus protocol used in ZILLIQA), the last step of the protocol requires nodes in the network to sign that they have all seen and agreed on the block. Malicious nodes may decide not to sign though. The block is committed to the blockchain only if all the honest nodes have signed the block. This guarantees finality as all nodes agree by signing that they think the signed block should be the next block.

11. Why does ZILLIQA not use PoS?

PoS is still relatively new compared to PoW which has survived the test of time. We strongly believe that PoS is still a very nascent idea. It was only recently that a provable PoS protocol (Ouroboros) was proposed. The proposed protocol however has some assumptions that may not be true in the real world setting. On the contrary, PoW is a very well-understood mechanism.

That said, the use of PoW in ZILLIQA is very different from the way it is used in say Bitcoin, which ensures that PoW does not become a bottleneck for scalability. ZILLIQA shows that even with PoW, it is possible to hit a throughput of thousands of transactions per second.

12. Would there be plans to move from PoW to PoS in the future?

We may replace PoW with a stake-based mechanism for Sybil resistance. However, we would still continue using pBFT as the underlying consensus protocol.

PoS is also not the ideal solution to every problem. To see this, one must consider the threat model under which a PoS-like consensus protocol will work. PoS by definition works under a set of assumptions which are often referred to as the "cryptoeconomic assumptions" or "rational behavior". This means that since participants have stake in the system, they would not want to behave maliciously and if they ever do so, then they will probably get detected and possibly penalized.

On the other hand, the (p)BFT consensus protocol works under a more hostile assumption, where nodes are Byzantine. This means that nodes do not care about losing stake and are only motivated to subvert the system. Since the adversary models are different, the two consensus proposals cannot be directly compared and cannot be replaced by one or the other.

2 Sharding in ZILLIQA

13. What is sharding and how does it make ZILLIQA scalable?

Sharding is a concept that has existed in distributed databases for a long time but hasn't been demonstrated in open, permissionless blockchains (where anyone can join and participate) at scale. The idea is to automatically split up a large network of machines processing transactions into parallel sub-committees or "shards". Each shard processes its own microblock in parallel with other shards, and resulting micro-blocks are merged into a final one. This automatic network parallelization has now been made practical for open blockchains through ZILLIQA!

14. Is sharding similar to MapReduce?

Yes, sharding and MapReduce are similar, but in MapReduce, mappers do not need to communicate. In non-state sharding they may need to send some messages at the end of the epoch.

15. What is the size of a shard?

The shard size is not fixed and is chosen in a dynamic manner. But, there will be a lower and upper bound. The lower bound is 600. We chose this shard size because at this size, the probability that 1/3 of the shard members are malicious is significantly low. We will also have an upper bound so as to make sure that broadcast does not become a bottleneck. The upper bound is still under test as it requires experimental testing and stress tests. We will inform when we have an exact number for this.

16. Does the shard count get adjusted dynamically?

Yes, if more nodes join the network after the end of epoch, the shard count will increase. If nodes leave, the shard count will decrease. However, the number of nodes in a shard should be at least 600 for security reasons.

17. What stops a shard from becoming fully Byzantine?

Shard creation using PoW is equivalent to sampling a subset of nodes uniformly at random. Hence, roughly speaking, in order to take over a shard, one needs to control 1/3 of the network. For a shard size of 600 nodes, the probability of 1/3 nodes being malicious is 1 in a million.

18. Is it possible to coordinate which shard nodes get into?

No, nodes are assigned to shards at random.

19. Will nodes know if they are in the same shard?

Yes, nodes will know other members of their shards.

20. Is there inter-shard communication?

ZILLIQA does not do state sharding, so, inter-shard communication is minimal. A shard communicates asynchronously with other shards only to transfer the transaction data.

21. Do shards have to trust other shards?

No, a shard does not need to trust another shard. No node has to trust any other node in the network.

22. How is PoW used only for Sybil resistance?

Every new node first does a PoW to join the ZILLIQA network. The network verifies the PoW submission and depending on the submission, it gets assigned to a specific shard. This can be seen as an identity establishment phase. Once the node gets into a shard, it can run multiple rounds of consensus with other nodes in the shard. This consensus is not PoW-based (as in say Bitcoin) but relies on an efficient Byzantine Fault Tolerant consensus protocol. This means that multiple rounds of consensus can be done with only one PoW.

23. Does PoW prevent a malicious miner from joining the network and leading a shard?

No, the protocol cannot prevent a malicious miner from doing a PoW. The purpose of using PoW is to ensure that we do not have too many malicious miners.

24. Do DS nodes and shard nodes have equal voting rights?

DS nodes vote for the consensus protocol that runs in the DS committee. A node in a shard votes for the consensus protocol in the shard which it is assigned to. Each node in a shard or in the DS committee has equal voting rights.

25. What is the expected block time based on the throughput of 2,500tx/s?

We are currently still tuning the block latency which will depend on many factors and is going to be optimised empirically. In the testnet, the block time is within 2 mins.

26. What is the length of a DS epoch?

Let us say that the DS committee size is n and the processing of a final block takes t mins. As such, the DS epoch will be $n \times t$ mins. This is under the assumption that every node in the DS will eventually become a leader and gets to propose a final block.

In the current configuration, $n = 600$ and $t = 2$mins. This means that the DS epoch is $600 \times 2 = 1,200$ mins (20 hours). 20 hours may seem long and nodes may go offline during this period. There are two possible solutions that we are exploring 1) Reduce t as much as possible 2) Instead of doing a round robin and proposing n final blocks during the DS epoch, allow the processing of less than n final blocks say a constant number 60. This would imply that not every node in the DS will become a leader. The leaders will then have to be selected in a randomized manner.

27. Can ZILLIQA be used for real-time transactions?

In ZILLIQA, consensus happens at two layers. First, at the shard layer where consensus is reached on micro-blocks. Second, at the DS layer, where consensus is reached on final blocks. For payments-like transactions (that excludes transactions that call smart contracts), being able to see a micro-block should suffice. In other words, one may skip seeing the final block. Of course, there is a chance that the DS committee may reject that micro-block, but the probability of that happening should not be very high. Such transaction confirmations could be done in seconds.

3 Smart contracts in ZILLIQA

28. Why are you developing a new language for smart contracts?

We believe in moving towards a language or a subset of an existing language grammar on which we can prove safety properties about a contract. This will avoid most of the issues that existing contracts are facing, e.g., DAO and Parity. Without any formal proofs about the contracts, such situations are unavoidable.

29. What is the language going to look like?

Unlike in Ethereum, ZILLIQA's smart contract language won't be Turing complete. The computation model will be based on communicating I/O automata (I/O Automata theory by Lynch and Tuttle'88[1] with CPS style return of values. The front-end language can be close to Solidity.

The rationale behind the choice is the following: not all applications require a Turing complete language. Moreover, Turing complete languages are hard to reason about and hence prone to bugs. A non Turing-complete language becomes amenable to formal methods-based verification because of its simplicity . In more concrete terms, it becomes possible to prove interesting safety and liveness properties about a non-Turing complete program such as the funds never get locked, etc.

30. Will formal verification be obligatory or optional?

[1] http://www.markrtuttle.com/data/papers/lt89-cwi.pdf

In the ideal scenario, we would like to make formal verifications obligatory. This would ensure that any dApp running on the platform is safe to use. Moreover, if one runs a sensitive dApp on the platform, then appending a proof will give dApp users confidence to use the system. However, not all developers are familiar with formal proof assistants. Hence, making it obligatory may hinder adoption of the platform. The other potential issue is that if we make formal verifications obligatory, then miners may have to check the proof while running the consensus protocol.

31. Which dApps would be a good fit for ZILLIQA given its Turing-incomplete language?

dApps that require high throughput and do not have infinite loops are a good fit for ZILLIQA.

32. What are some sample applications that will require Turing-completeness?

Imagine a hypothetical blockchain platform with no gas fees - in other words, it is free to run a dApp. Now consider a dApp that requires to run a loop which may never terminate. A simple example would be a dApp that waits for the user's input, which the user may never provide. Such an application cannot be built in a Turing-incomplete language. One may also imagine a dApp that talks to another dApp non-stop. For instance, a smart meter that communicates with a smart grid non-stop.

That said, we do not foresee severe limitations on dApps in ZILLIQA except that in the contract, one may not be able to write infinite loops and unbounded recursion. Also, any external call has to be the last instruction in the transition function.

33. Is it possible to implement sidechains (like Plasma) with ZILLIQA's smart contracts?

Yes, it is possible to implement sidechains, but the details on how sidechains can be attached to ZILLIQA in terms of APIs, communications, etc. need to be figured out.

34. Does ZILLIQA have plans to scale further to accommodate dApps like Facebook which handle millions of requests per second?

Security requirements for a social network (e.g., preventing forged posts) are less stringent than for cryptocurrencies. There is less incentive trying to attack and forge messages than to directly steal money. Hence, off-chain solutions which may be less secure (sidechains, Raiden, etc.) could be added to ZILLIQA to gain even more throughput for applications with weaker threat models.

4 A little bit about mining

35. How is mining ZILLIQA different from Bitcoin?

Unlike Bitcoin, the mining process in ZILLIQA is not directly based on PoW. Every ZILLIQA node first does a PoW at the start of what is called a DS epoch. Once, a valid PoW solution is submitted to the network, each node will then have to participate in the pBFT consensus protocol. Consider the consensus protocol as a simple voting. If a super-majority of nodes vote for it, then the block will be considered valid and can be committed to the blockchain. Once a node has done PoW, it can vote for a

certain number of blocks. Every block that gets committed to the blockchain will yield some reward. The difference wrt Bitcoin is that in Bitcoin, nodes do a PoW for every new block. In ZILLIQA, a node will do a PoW say every 60 blocks. This also means that the energy footprint associated with PoW in ZILLIQA is low. Benefits of mining ZILLIQA is further detailed in the following blog post: https://blog.zilliqa.com/with-zilliqa-do-more-with-your-gpus-miners-fbe29ac72ede

36. Is mining on ZILLIQA more CPU or GPU-oriented?

In the beginning before difficulty ramps up, a CPU should work. However, a CPU will not be as efficient as GPU mining in the longer run.

37. Is it possible to mine ZILLIQA and other coins at the same time?

As a node in ZILLIQA will not be using GPU to mine every final block, it is perfectly possible to use the GPU to mine on another chain in parallel when it is not in use. However, one must take into account the average block time for the other chain to ensure that mining on the two chains remains separated in time.

38. Is it possible to use specialized mining equipment only at boot time?

Yes, it is possible to use specialized mining equipment only at boot time, i.e., at the start of the DS epoch as the rest of the protocol is not computationally intensive.

39. Between two DS epochs, does a node have to stay connected to the shard to mine?

Yes, mining in ZILLIQA does not end with PoW as the node later has to participate in the consensus protocol.

40. Is an epoch synchronized across the entire network?

Yes, it is weakly synchronized across the netwok. At the start of each epoch, everyone starts doing PoW and submits the solution to join the network. There is only a limited time window to do this. Once the window has passed, no one can join the network (till the next DS-epoch) and sharding begins.

41. What is the difficulty ratio for the DS nodes and nodes in a shard?

There are two ways of managing the difficulty of PoW for shards and the DS Committee. One possible solution is to keep the difficulty the same and wait for say 601 submissions (600 nodes for a shard and 1 node for the DS committee per DS epoch). The PoW submissions can then be sorted and then the node submitting the smallest one is assigned to the DS committee, while the rest of the 600 nodes get assigned to a shard. The other possibility (as mentioned in the whitepaper) is to first perform a PoW for the DS committee and then a second round for shards. In this case, the difficulty will have to be significantly different. We are going the latter route but also exploring the former. The advantage of the latter being that the protocol can decide the frequency of the two PoWs. Since we are still in the private testnet phase, we do not have concrete difficulty parameters decided yet.

42. What determines the difficulty level of PoWs?

The difficulty is determined by estimating the network hashrate. We have assumed this to be an intrinsic parameter in Ethash.

For the DS-level PoW, the difficulty estimation is going to be exactly the same as in traditional PoW-based blockchains such as Bitcoin. For the shard level, we will start with a standard pre-determined difficulty and measure the average time it took for each shard member to solve the puzzle. We then take the average. Similar measurements will be taken over several epochs, then we will take another average. This should allow the network to estimate the effective hashrate and hence adjust the difficulty.

43. Is there Emergency Difficulty Adjustment like in Bitcoin Cash?

ZILLIQA ensures that the more miners join the network, the better the throughput becomes. However, we will not be able to support an extremely large network say of 1 million nodes. In that case, we may have to resort to something like Emergency Difficulty Adjustment.

44. Would it be possible to join multiple shards if you have excess hashpower?

Technically yes, if a node has sufficiently large mining power to join more than one shard. However, each identity is bound to a network address and port. In order to join multiple shards, a miner will need to use a different port for the "other" node.

45. Do nodes have to wait for their turn to receive mining rewards?

Yes, a node will have to wait for his turn to get the reward in a round robin setting. A good feature with this model is that once a node gets into a shard, it is guaranteed to get the reward. As a result, reward variance will be low. Moreover, since a node does a PoW only once before joining the shard, it won't be doing PoW during the round robin cycle. The computations that a node would do after PoW are not very intensive.

46. What are the rewards for DS block and transaction block processors?

The rewards are given only for the transaction blockchain and not for the DS blockchain.

47. What is the reward emission curve going to look like?

The high level plan is to make it heavy in the first few years. As the strength of our protocol is the number of miners, we would like to make it as attractive as we can for miners in the first few years. We will be releasing the exact emission curve soon.

48. What is the energy usage like to run a node on ZILLIQA?

Mining on ZILLIQA is expected to be more energy-efficient than existing PoW-based blockchains. Let us say the energy consumed per node is x units/s. Let n be the number of nodes in the network that

can together process m tx/s. This means that the energy required to process 1 tx for one node is x/m units. The total energy consumption for the entire network to process 1 tx will be $n \times x/m$ units. Now the linear scaling means that when n doubles to $2n$, m doubles to $2m$. This means that the energy consumption to process 1 tx for the entire network remains unchanged to $n \times x/m$ units. This is clearly not the case with Ethereum or Bitcoin as when n doubles to $2n$, m remains unchanged, hence the total energy consumption by the network to process 1 tx will double and not remain constant. For a rough energy consumption we estimate that the cost of running a mining node on ZILLIQA will be roughly 1/10 of the cost of running a mining node on say Ethereum.

49. Does ZILLIQA suffer from verifier's dilemma?

In a Nakamoto consensus-based protocol, where a leader proposes a block and others accept or reject it, the miners have an incentive to ignore transaction verification and start mining on the next block. In a (p)BFT like consensus, where miners together reach consensus on the next block, the incentive to skip transaction verification is less.

5 Let's discuss attacks

50. Has the security/performance of ZILLIQA been analyzed to a sufficient degree?

For the overall architecture, a high-level security analysis is shown in the Elastico paper, which went through rounds of peer review before its publication. The proof is given that less than 1/3 of all the members in each shard are Byzantine with high probability.

As for the security of each consensus run of pBFT, it was analyzed in the original academic paper by Castro and Liskov. It was shown that the pBFT protocol guarantees safety and liveness in a weakly asynchronous model with a fraction of Byzantine nodes in the consensus group.

As for performance, we have implemented ZILLIQA's core layers (including the consensus layer) and have tested it on our internal testnet with nodes running on AWS EC2 instances. With 3,600 nodes (6 shards), we observed a peak throughput of 2,488 transactions per second.

51. Has ZILLIQA's whitepaper been peer reviewed?

ZILLIQA borrows ideas from several peer-reviewed academic papers. The core idea of sharding comes from a CCS'16 paper that was co-authored by some of the ZILLIQA team members. The idea of using two blockchains to decouple the leader election and the block proposals comes from BitcoinNG which was published in NSDI'16. Use of pBFT for consenus in a sharded enviroment comes from the CCS'16 paper and the paper on ByzCoin which was published in Usenix Security'16. We also use EC-Schnorr multisignature to reduce the signature size in pBFT. This comes from the CoSi paper which was published in S&P'16 (Oakland). All these conferences CCS, NSDI, Usenix Security and S&P are top-ranked security and distributed system conferences.

52. Are 51% attacks possible on ZILLIQA?

51% attacks are possible on most PoW-based blockchain platforms. For any blockchain to work, one

must first assume a threat model. For instance, most PoW-based blockchains assume that no single entity will ever get control of 51% of hashing power. Under this threat model, it can be shown that the blockchain protocol is secure. Since many blockchains have gotten very close to violating the 51% assumption in the past, ZILLIQA decided to go with a stronger assumption. In ZILLIQA, the assumption is that less than 1/3 of the hashing power is under the control of the adversary.

Also, 51% attacks by definition make more sense at the DS level than at the shard level. This is because at the DS level, only one node from each shard gets into the DS committee whereas multiple nodes will get into a shard.

53. Can a shard be attacked?

There are two ways one could attack a shard. 1) Attack its leader 2) Attack non-leader nodes. If one attacks the leader, then the shard momentarily stops proposing new blocks. However, this does not mean that the protocol gets stalled forever. The reason is that there will be other shards that continue to submit blocks. Moreover, the protocol will replace the affected leader. Now, if one attacks the non-leader nodes, if the number of victim nodes and the remaining Byzantine nodes is less than 1/3, the protocol will function without any issue.

54. Are attacks from quantum computing possible?

Attacks from quantum computing are feasible, especially on the signature scheme that ZILLIQA uses. The reason we have not gone with quantum-resistant signatures is that their signature size is usually much larger than that of classical signatures. Having said that, we are very keen on moving to quantum signatures if crypto researchers come up with short quantum-resistant signatures. Another point is that signature schemes need to have aggregation properties as in EC-Schnorr and it is unclear if existing quantum signatures have this property.

55. Does dividing the network into smaller shards make it more susceptible to Sybil attacks?

ZILLIQA assumes that less than 1/3 of the hashing power is under the control of the adversary. Under this assumption, a supermajority of the nodes in the network will be honest. This assumption is also necessary for any BFT-based protocol to work. So, in short ZILLIQA is secure as long as the 1/3 assumption holds.

The fact that ZILLIQA implements sharding does not change this 1/3 assumption. Attacks have nothing to do with the number of shards, rather, they depend on the size of each shard. Consider a scenario where one runs a BFT protocol with the entire network without any shard. As long as the network as a whole does not have too many malicious nodes, BFT will work. Now, in ZILLIQA we wish to do parallel processing and hence we need to divide the network into a certain number of shards. If one starts with an initial network with the 1/3 assumption, then one must ensure that the 1/3 assumption is also true for each shard that gets created. This is because each shard will eventually run BFT.

In order to guarantee this, the protocol does random sampling via PoW submissions, i.e, the nodes that go in a specific shard are picked randomly from the initial pool of the entire network. Note that the shard size is crucial. We do not want to create a shard with just a couple of nodes otherwise, the

1/3 assumption will not hold with strong probability. There is a thorough probabilistic analysis in the Elastico paper. For these reasons ZILLIQA requires a minimum of 600 nodes per shard.

56. Is it possible to obtain a majority in one shard by pure luck?

The ZILLIQA threat model follows a classical Byzantine Fault Tolerance model where we assume that at the start of the protocol, at most 1/3 of the nodes in the initial network (before sharding) are malicious. Under this assumption, we ask nodes to perform PoW. PoW gives a way to do random sampling from the initial pool of nodes and allows us to create shards. If the shard size is sufficiently large (say more than 600), then it can be shown through probabilistic analysis that a shard will also have at most 1/3 of its nodes as malicious.

57. What is the maximum aggregate damage that a Byzantine party can impose?

Any Byzantine party by definition can do all it wants to subvert the protocol, but the pBFT protocol ensures that the safety and liveness properties will not be sacrificed within any consensus run if their fraction is limited. Further, view change and rotation of the leader will be triggered to replace the misbehaving leader with a new one to prevent prolonged stalling. A rational attacker who is a leader does not have the incentive to disrupt the execution of the protocol as he will lose rewards.

58. How would (Byzantine) nodes divide their PoW power between PoW1 and PoW2?

PoW1 and PoW2 will not influence the decision of an honest node if their difficulties are the same and the expected reward associated with being in a shard or being in a DS committee is the same. If the difficulties are different, then honest nodes will attempt to join the shard as it is more challenging to get into the DS committee. However, since DS nodes may get slightly higher rewards they have incentive to work on PoW1, Byzantine nodes may attempt to target the DS committee (via PoW1) as getting into a specific shard (via PoW2) is more difficult due to random sampling.

59. Can PoW1 introduce temporary forks?

Once PoW1 is submitted, the DS committee runs a pBFT to agree on the next DS block and the winner of PoW1, which gives finality to the block.

60. Does ZILLIQA require all participants to be available?

No, ZILLIQA does not require all nodes to be available. Unavailable nodes or unresponsive nodes are counted as Byzantine nodes. So, for a shard size of 600 nodes, around 200 nodes can be unavailable or behave maliciously (assuming at most 1/3 are byzantine). As long as the honest nodes, i.e., the remaining 400 nodes are available, the shard can still process transactions. In ZILLIQA, if a shard does not have a sufficient number of online nodes, the shard cannot propose valid microblocks with valid collective signatures. However, this will not halt the execution of the whole protocol, as the other shards can still propose microblocks to accept transactions. Under specific conditions, e.g., a shard does not propose microblocks for multiple consensus rounds to the DS committee, DS committee can decide to re-shard the network. Right now, we are still testing and trying to find the proper parameters for block timing and the number of consensus rounds for each epoch.

In ZILLIQA, the DS committee drives the protocol to move forward. In the extreme case with negligible probability, if over 1/3 of the nodes in the DS committee are offline, the protocol may make no progress for several epochs. To resolve this issue, one possible way that we are exploring is that we will fall back to Nakamoto consensus. When the network elects a sufficient number of DS nodes, the DS committee will inform the network to run the normal protocol. This gives a way to recover from severe outage situations. This part is not present in the whitepaper.

61. If ISP were to throttle the nodes, how much of an impact will it have on rebuilding of final blocks from micro blocks?

Let us assume that ISP throttles a few nodes. Now, as long as the number of victim nodes and the remaining malicious nodes is less than 1/3, the DS committee will still be able to build the final block.

62. Is ZILLIQA self-stabilizing in case of severe attacks?

ZILLIQA runs under the assumption of 1/3 Byzantine nodes. If the assumption does not hold for some reason, then the security guarantees cannot be ensured. Compare this with a PoW-style consensus as in Bitcoin which assumes that no entity can control 51% of the hashing power. Now, consider a really powerful mining pool which violates this assumption, we do not see how the system can self-heal by itself. In other words, any secure system is secure only under a specific threat model.

6 Surely there must be some limitations?

63. What is the maximum throughput capacity of ZILLIQA?

In theory, throughput in ZILLIQA increases with the number of nodes. However, in practice, if the network size grows really large say to 1 million nodes, then broadcast becomes an issue. Hence, in practice, there is a sweet spot until which the throughput can linearly increase. We are still running experiments to know that sweet spot.

It is also possible to make some assumptions on the available bandwidth at each node, take into account the communication complexity of the protocol and the size of each message exchanged and compute a rough upper bound.

64. Is there a trade-off between failure rate (security) of a shard and transaction scaling?

There does exist a tradeoff between throughput and security. The overall throughput of the ZILLIQA network depends on both the number of shards and the size of each shard. Roughly speaking, the larger the number of shards, the more transactions the network can handle thanks to parallelization and the larger the shard, the more secure it is.

As for the shard size, we require that each shard has to have at least 600 nodes. Let us say we have a total of 1,000 nodes to create shards. The ZILLIQA network will then only create a single shard with 1,000 nodes instead of creating two shards with 600 and 400 nodes respectively. This ensures that security is not compromised. However, we also do not want the shard size to be too large either. If the shard size is very large say 10,000 nodes, then broadcast within the shard will become a bottleneck. As

a result, we chose to have a shard that is large enough to ensure 1/3 assumption holds but not too large to create bandwidth issues.

65. Is the transaction speed for a user limited to the processing power of its specific shard?

Let us go through the protocol steps first. When a user creates a transaction, it is sent to a specific shard (depending on the sender's account address). The shard checks if the transaction is valid. A number of similar valid transactions are then grouped together to form a "micro block" and a consensus is reached. The micro block is then sent to the DS committee which aggregates micro blocks sent from several shards and creates what is called a "final block". The DS committee then runs a consensus on the final block which finally gets committed to the blockchain. Hence, the performance will depend on the specific shard that is supposed to process the transaction and the DS committee, but not on the other shards.

66. Is the limitation above serious?

There are two likely reasons why blockchains receive a large volume of transactions. First, it could be that a single user sends a large number of transactions (say 100) in a short time interval. Second, multiple users (say 100 users) could be sending transactions at the same time, with each sending a single transaction. We believe that the latter is more prevalent than the former. Hence, if the number of shards is sufficiently large, the latter case can be well handled by each shard individually. ZILLIQA can handle the scenario where a single user issues say 100 tx/s. However, while the network can handle 2,500 tx/s with 3,600 nodes, it can't handle 2,500 tx/s from a single sender. So there are some limits, even though they are high. Additional testing can help make those limits clearer and additional engineering can focus on improving them.

67. Can ZIL tokens change multiple hands within one DS epoch?

Within one DS epoch, we'll have multiple final blocks, in which transactions can be confirmed. With this, let us consider the following scenario: Alice sends 10 ZILs to Bob, and then Bob sends 5 ZILs to Charlie. If before the first transaction, Bob had a balance of 0, then the second transaction cannot be accepted in the same final block. Any such transaction will be buffered by the nodes in their memory pool for the next final blocks. However, if Bob had an initial balance of 5 ZILs, then both the transaction can be processed with the same final block. So multiple correlated transactions can be processed either within one final block, or a few final blocks, but all should be within one DS epoch. We will complete this version of design first, and explore optimisations for such correlated transaction netting in future versions.

68. Can the ZILLIQA protocol stop, freeze transactions to manage monetary sends and receives?

We have discussed stopping/freezing of transactions but found it difficult to implement it at the protocol level in a decentralized manner. We instead considered implementing it at the wallet level.

69. Why is ZILLIQA not working on state sharding?

We have had several rounds of discussion on state sharding, and the conclusion was that there still

isn't a state sharding scheme that is secure enough (e.g., resilient to attacks) and efficient (e.g., without excessive cross-shard communication). That's why we prefer keeping state sharding to future work for now.

70. How does ZILLIQA address the storage requirements arising from its high throughput?

If we ignore the smart contracts for a moment, then because of the finality, one doesn't necessarily need to store the entire history. You will only need to store the latest state. The use case where one may need to store the entire history is for block explorers. With smart contracts, the storage does become an issue as the smart contracts need to be stored for a later execution. We are currently in touch with distributed storage network platforms such as Bluzelle and Genaro to see how we can leverage their network for storage. One possibility is that when a block gets generated by the ZILLIQA network, the block gets stored on the Bluzelle or Genaro network. When a transaction needs to execute a contract, the block can be fetched by the miners and the computation can then be done. ZILLIQA nodes may also cache the block if the contract gets called very frequently.

71. Does the ZILLIQA architecture prevent it from running a Turing-complete language?

No, the ZILLIQA architecture does not prevent running of Turing-complete languages.

72. Will there be a way to edit existing contracts or publish updates to them?

It will not be easy to implement mutability of smart contracts. The reason is that the underlying blockchain is designed to be immutable. If we allow smart contract developers to change the contract while it is on the blockchain, we fear that malicious contract owners may change the terms of the contract to avoid landing in unfavorable scenarios.

73. Could the cost of gas make the training of neural nets on ZILLIQA impractical?

We believe that the sharding architecture is ideal to run applications that require high throughput/ volume. The sharded architecture along with a data-flow like language can potentially be used for some interesting applications such as training models.

It is clear that such computations will have an impact on gas fees. The advantage that the blockchain can provide here is strong guarantees on the correctness of the computation result (high accuracy of say 99.999%). We are aiming for low fees for basic transactions. If throughput is high, miners will get sufficient incentives in aggregation.

6.1 Governance

74. Does ZILLIQA allow forks?

If you mean permanent forks like Bitcoin and Bitcoin Cash, then ZILLIQA does not prevent that as the source code will be public. Anybody should be able to fork it. If you mean temporary forks as in double spend attacks, then ZILLIQA will not allow that. This is because of the underlying consensus

protocol that gives finality. So, once a block gets committed to the blockchain, it is not possible to have another block that shares the same parent.

75. How does ZILLIQA handle governance?

If you mean fork governance along Tezos-like governance protocol, ZILLIQA currently does not have such a framework in place.

There are trade-offs in either of the governance models: on-chain and off-chain governance. If the ZILLIQA team makes a decision, it becomes very centralized, but the decision making process is fast. However, if the decision is made by the community and its various players, then the decision is very democratic but will be slow. To start with, ZILLIQA Research will certainly take the driving seat. We will release details on ZILLIQA Research and its governance model soon.

7 Competitors

76. Who are ZILLIQA's competitors?

To start with, competing in an open source framework like in the blockchain space does not make much sense. Since, the project is open source, anybody is free to reuse the idea in another project. The biggest problem in blockchains today is scalability. There are several solutions to this problem. The beauty is that each project provides its own solution and this creates a healthy environment of learning from one project and building another one. Each project usually has its own strengths and weaknesses. Attempting to compete with any blockchain whatsoever will take the project nowhere. This will prevent us from adopting smart ideas out there in our so-called competitors. Our philosophy is that one needs to identify the strength of the idea, follow it and be open to learning from other projects.

77. How does ZILLIQA's throughput and scalability stack up vs "competitors"?

There are other platforms which claim much higher throughput than ZILLIQA. But, to the best of our knowledge, ZILLIQA is the only protocol that is truly scalable without sacrificing security and decentralization, i.e., throughput increases with a growing network size.

78. Is sharding available in other blockchains?

To the best of our knowledge, sharding is not currently available in other blockchains. But we are aware of some of the plans and discussions in other channels where other projects are actively looking into it. One of the foremost examples is Ethereum.

Index