Shall I compare thee to ZILLIQA?

[Version 0.1]

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

January 23, 2018

Abstract

This document compares and contrasts ZILLIQA with other projects. The goal is to position ZILLIQA wrt several existing blockchain projects. We would like to stress that we appreciate the ideas put forth by each one of these projects and do not undermine any scientific ideas therein. On the contrary, some of the ideas used in ZILLIQA were inspired by these projects. Please report any technical inaccuracies in this document regarding other projects to our team members.

1. ZILLIQA vsEthereum1.0? . . . 2
2. ZILLIQA vsEthereum2.0? . . . 2
3. ZILLIQA vsPlasma/Raiden? . . . 3
4. ZILLIQA vs IOTA? [A comparison given by @Snow on the Slack channel] . . . 3
5. ZILLIQA vsEOS? . . . 4
6. ZILLIQA vsBitshares? . . . 4
7. ZILLIQA vsNEO? . . . 4
8. ZILLIQA vsRChain? . . . 5
9. ZILLIQA vsRadix? . . . 5
10. ZILLIQA vsCardano? . . . 6
11. ZILLIQA vsStratis? . . . 7
12. ZILLIQA vs0Chain? . . . 8
13. ZILLIQA vsORBS? . . . 8
14. ZILLIQA vsOmniLedger? . . . 9
15. ZILLIQA vsWavesNG? . . . 9
16. ZILLIQA vsCypherium? . . . 9
17. ZILLIQA vsIOS? . . . 10
18. ZILLIQA vsCosmosTendermint? . . . 11
19. ZILLIQA vsHashgraph? . . . 12
20. ZILLIQA vsRedBelly? . . . 12
21. ZILLIQA vsUniversa? . . . 13

22. ZILLIQA vsRaiblocks? 13
23. ZILLIQA vsOMG? 13
24. ZILLIQA vsGolem? 13
25. ZILLIQA vsSONM? 13
26. ZILLIQA vsTezos? 13
Index

Page 2

1. ZILLIQA vs Ethereum 1.0?

- **Transaction rate:** The Ethereum blockchain protocol has a transaction rate of around 10 tx/s. ZILLIQA is currently 250 times of that.
- **Consensus:** Ethereum uses PoW for consensus, ZILLIQA uses pBFT for consensus. PoW in ZILLIQA is used only to establish identities to prevent Sybil attacks. Once identities are established, the network can reach consensus on several blocks in a row. This makes PoW more efficient as a new PoW is not required per block as in Ethereum.
- **Smart contracts:** ZILLIQA will also be different from Ethereum in terms of smart contracts. The basic difference comes from the fact that ZILLIQA's underlying smart contract language will not be Turing-complete.
- **Finality:** ZILLIQA's consensus protocol gives finality. This means that no confirmation is required. This is different from PoW-based consensus in Ethereum where temporary forks can occur and hence a certain number of confirmations are required to have a cushion against double spends.
- **Scalability:** Ethereum is exploring how to do PoS securely and effectively in its future versions. ZILLIQA takes a different approach with sharding for an accelerated path to scalability. We will try to be compatible with Ethereum wherever possible wrt mining, smart contract language, etc.
- **Mission:** ZILLIQA has a specific mission to support high-throughput dApps that need to leverage the properties of a blockchain: openness, accountability, and transparency, etc. ZILLIQA will focus on supporting those dApps requiring high throughput or high volume. Some of the existing dApps on Ethereum may benefit from ZILLIQA, but we do not expect those to be the majority.

2. ZILLIQA vs Ethereum 2.0?

ZILLIQA does what is called network or transaction sharding. Imagine a sample network of 1,000 nodes. ZILLIQA will automatically divide the network into 10 shards each with 100 nodes. Each shard can now process transactions in parallel. If each shard is capable of processing 100 transactions per second, then all shards together can process 1000 transactions per second. The ability to process transactions in parallel due to the sharded architecture ensures that the throughput in ZILLIQA increases (roughly) linearly with the size of the network.

Ethereum is currently researching on what is called state sharding, i.e., how to divide the blockchain state so that storage does not become an issue in the long run. ZILLIQA is not doing state sharding in its current immediate plan. Running smart contracts on a sharded network without state sharding is

already a big challenge. Having said that, state sharding is in our future plans.

Ethereum 2.0 is a research project and several details are not yet figured out. Phase 1 of Ethereum 2.0 (a work in progress) is a loosely coupled sidechain-like solution attached to the main Ethereum chain via a validator manager contract (VMC). VMC sits on the main Ethereum chain and maintains the sharding system. In other words, it is responsible for key tasks such as adding a new validator,

assigning a new validator for a shard, etc.

Phase 1 of Ethereum 2.0 is a simple solution that does not solve all sharding-related issues. We expect the Phase 1 solution to have the following issues:

1. **Single point of failure:** It appears that VMC may become a bottleneck and a single point of failure as key steps of the protocol need to pass through VMC. Any bug with VMC can bring the entire system down. ZILLIQA does not have any central entity or contract on which the entire system relies.
2. **Finality:** Each shard will use a PoS-based consensus protocol with the longest-chain-rule as the fork resolution mechanism. As a result, Ethereum 2.0 (Phase 1) does not provide finality to the system state. ZILLIQA on the other hand will provide finality to the state through its pBFT protocol.
3. **Throughput:** It is claimed that Phase 1 will increase the throughput by around 100x. ZILLIQA's throughput with a modest network size of 3600 nodes reported a throughput of 250x the throughput of Ethereum 1.0.
4. **Cross-shard communication:** It appears that Phase 1 of the sharding proposal will not allow cross-shard communication (or it is minimal). This means that a smart contract residing in one shard and that needs to call another smart contract residing in other shard may not run. In fact, cross-shard communication is one of the biggest challenges when one needs to run a Turing-complete language on a sharded architecture.

   The project does have plans to use a UTXO-type model (through receipts) to handle cross-shard communications but that does not seem to be within the scope of Phase 1 of Ethereum 2.0.

3. ZILLIQA vs Plasma/Raiden?

Plasma is a sidechain solution while Raiden is an off-chain solution to the scalability problem. This means that not all data gets committed to the (main) Ethereum chain. This gives scalability.

That said, a side chain solution (or an off-chain) may not provide the same security, resilience and decentralization guarantees as a fully on-chain scalability solution. Moreover, these solutions do not directly make the main chain scalable.

ZILLIQA is an on-chain solution and does not rely on any sidechain (or off-chain) for scalability and hence does not sacrifice on security and decentralization. Any sidechain (or off-chain) solution can potentially be integrated into ZILLIQA to further increase its throughput if necessary.

4. ZILLIQA vs IOTA? [A comparison given by @Snow on the Slack channel]

Consensus in IOTA is reached via a directed acyclic graph aka Tangle. Confirmation in IOTA comes

when two new nodes are created on top of a previous node. This means that IOTA can scale exponentially but the security is dependent on the weight of the nodes. As of now, IOTA compensates the lack of weight with a coordinator, which is controversial as it is centralised.

In addition, as the graph is not linear, timestamp is not determinant but hypothetical and hence may limit IOTA's smart contract capabilities as it is not possible to establish a timeline properly. Also, it is

hard to guarantee that the Directed Acyclic Graph (DAG) will branch in the right direction and that convergence can be reached. ZILLIQA on the other hand gives finality to transactions.

IOTA also has no incentive layer. Without such incentives, one has to assume that nodes will be honest as cheating/double spending/attacking network would be illogical. If the premise is that the nodes are honest (which is a huge assumption), it is unclear who will own the majority of such dedicated devices and if there is true competition to ensure the ledger is truly immutable.

5. ZILLIQA **vs EOS?**

The key feature of EOS is scalability and hence we compare ZILLIQA and EOS wrt the scalability aspect.

The high throughput of EOS comes from its consensus protocol called dPoS (delegated PoS). dPoS in EOS works in the following manner: A dedicated subset of 21 nodes aka block producers is elected by the network to propose the blocks. The probability of a node being elected is based on its stake in EOS.

Each of the 21 nodes proposes exactly one block with a block time of 3s. As you may imagine, this mechanism will certainly yield very high throughout. However, this also leads to some risks. The number 21 is not large enough and hence it should be easy to attack them or at least a majority of them. The obvious solution here would be to increase 21 to a larger number say 1,000 or even more. By doing so, latency will increase and the throughput will drop. This is because each node would need to broadcast the block to a larger network and broadcast is always a bottleneck in any distributed system.

Another point to add is that trusting only 21 nodes also leads to some centralisation risks. Moreover, dPoS does not guarantee finality. This means that a malicious node may produce a competing block. As a result, confirmations are required (as in Bitcoin) to have cushion against double spends.

ZILLIQA uses a different consensus protocol called pBFT. The benefit of pBFT is that it gives finality. Hence, no confirmation is required. The consensus in ZILLIQA is run in parallel within a set of around 600 nodes. The size is large enough to ensure that the centralisation risk is low.

6. ZILLIQA **vs Bitshares?**

Bitshares is very similar to EOS and hence the same comparison given in Question 5 holds.

7. ZILLIQA **vs NEO?**

NEO uses a delegated BFT for consensus aka dBFT. This means that a designated subset of nodes aka bookkeeping nodes are responsible for running the consensus protocol on behalf of the NEO network. Bookkeeping nodes must deposit a sum of money as a collateral and this collateral amount must also be sufficiently large. The problem with the collateral idea is the following: if the number of bookkeeping nodes is large, then the collateral locks away liquidity from the market. If the number is

small, then there is a risk that they can be taken control of by attackers.

In ZILLIQA, we use the standard (practical) BFT protocol and for security reasons, it is required that the protocol be run with a sufficiently large number of nodes. The nodes are not required to put up

Page 5

collateral and hence the market liquidity is not affected. Furthermore, the absence of collateral makes the network more open. ZILLIQA also implements sharding that allows the transaction throughput to scale as network grows.

The other difference is the approach to reach high throughput or scalability. Classical BFT does not scale beyond a hundred nodes. However, BFT protocols have the advantage that they come with finality. There are two ways in which one could use BFT without hitting the underlying bottleneck issues. One is to run BFT among a subset of delegated nodes in the network as NEO does. The other is to shard the network into several sub-networks where BFT can be run in parallel. ZILLIQA takes the latter approach. This has two advantages: 1) security and decentralisation are maintained. ZILLIQA also uses some other primitives such as Schnorr multi-signing to circumvent some issues. This means that 2) throughput can increase linearly with the number of nodes. The second advantage is clearly absent in the first approach. Typically, a larger network means better decentralisation but, in a non-sharded network it unfortunately leads to bottlenecks. A sharded network like in ZILLIQA however, can make the most of a large network while avoiding the bottlenecks.

8. ZILLIQA vs RChain?

RChain and ZILLIQA differ in three aspects:

- **Consensus protocol:** ZILLIQA uses pBFT for consensus while RChain uses Casper PoS for consensus. The advantage of pBFT is that it assumes a Byzantine adversarial model whereas PoS assumes a cryptoeconomic adversarial model whereby the attacker is rational, i.e., there is little incentive for anyone that holds a stake to attack the system due to their vested interest. From this perspective, pBFT is secure in a much more hostile model. Furthermore, pBFT comes with exact finality unlike PoS where finality is only eventual and hence confirmations are required.
- **Concurrency:** RChain employs a multi-threaded VM to run contracts and hence the parallelism is at the node level. The parallelism in ZILLIQA is at the network level, not at the VM level. In other words, RChain leverages language-level parallelism while ZILLIQA leverages network-level parallelism. RChain however does have the notion of namepsaces that can potentially be used for network-level parallelization.
- **Smart contract language**: The smart contract language in ZILLIQA is based on communicating automata. Rholang (the smart contract language in RChain) is based on asynchronous polyadic π-calculus that is best suited to work in a conncurrent setting. Rholang is Turing-complete and hence admits unbounded recursion, while the smart contract language in ZILLIQA will be non Turing-complete as it will only allow bounded recursion. The other difference comes from the fact that the smart contract language in ZILLIQA mandates all external calls to other contracts to be made at the end of the transition. As a result there is no complex interleaving of external calls and local instructions. Hence, analyzing and proving safety properties using formal methods becomes much easier in ZILLIQA than in Rholang.

9. ZILLIQA vs Radix?

Note: This comparison comes from the reading of the Tempo whitepaper which lacks some important details including the threat model under which the system can be called secure. This makes it very hard for us to compare it with ZILLIQA at a deeper level.

- **Sharding:** Ignoring the lack of a well-defined threat model, Radix does what is called state-sharding. The global state of the system is divided into shards. Note that the term shard is not used in the same way as in ZILLIQA. A shard in ZILLIQA means a subset of nodes. In Radix, a shard means a portion of the global state. In ZILLIQA, we do not do state-sharding but instead focus on transaction sharding, i.e., increasing the throughput by parallel processing.

  In Radix, each node in the network is free to pick the shard it wants to keep. Each shard contains what is referred to as an atom that constitutes and modifies the global state of the system. Whenever a user creates a transaction (also called a Transaction Atom), it is first sent to a node which the client is connected to. The node then picks a random node that is linked with the transaction. We call a node to be linked with the transaction if it holds a shard that contains the state of either the sender or the destination. The second node then picks another one and so on. Each node on this path checks whether the transaction is valid and whether it is an attempt to double spend. This is done by maintaining a local counter. The length of the path is decided by the client. Since the nodes may have different views, there is a conflict resolution mechanism.

  There are a few points which are unclear from the whitepaper. For instance, the whitepaper does not discuss the threat model under which the system is secure. The webpage further says that Radix is based on neither PoW nor PoS. This means that any node can freely join the network and hence increase the fraction of malicious nodes in the network. It is unclear how the system will handle an influx of malicious nodes.

  Also, since nodes are free to pick the shard they want, it is possible that a shard never gets picked up by any node. This would mean that a transaction concerning that shard will never get processed. This is also known as the data availability problem.

  Moreover, the whitepaper says that a client submitting a transaction selects a node it is connected to. What happens if that node is malicious? Or the next node randomly chosen by this node is malicious? Then, the required path length can never be achieved and the transaction may never go through. In short, it is hard to appreciate the solution put forth in Radix without a well-defined threat model.

10. ZILLIQA vs Cardano?

There are three aspects wrt which Cardano and ZILLIQA can be compared:

1. **Approach to scalability:** The current approach that Cardano takes towards scalability is PoS. ZILLIQA's approach to scalability is sharding with pBFT as the underlying consensus protocol and PoW as a Sybil resistance mechanism.

   PoS alone does not give linear scalability as sharding does. Cardano in its design philosophy[1] claims that sharding becomes trivial to implement with PoS. We disagree with this statement as sharding is known to introduce multiple challenges including but not limited to: how to merge transactions processed from different shards, how to safely and efficiently transfer transactions

https://whycardano.com/#scalability

Page 7

processed in one shard to the other shards without creating a bottleneck, etc.

With smart contracts, sharding introduces further challenges. To see this, consider a smart contract that runs an ICO, i.e., it sells a finite number of ERC20-type tokens in exchange for some native Cardano tokens. Each contributor sends a transaction specifying the required number of Cardano tokens and in return gets some number of ERC20-type tokens depending on the price of the token.

Now consider the case, where, the contract has only 1 remaining token and two contributors say Alice and Bob wanting to buy the last remaining token. In a sharded environment, Alice and Bob's transactions may get handled in different shards. In the absence of cross-shard communication prior to creating the next block, each shard is oblivious of the fact that the token has been allocated to the other user in some other shard. Such conflicts need to be resolved either synchronously by communicating with other shards or could be settled a posteriori after the two shards have proposed their blocks. Conflict resolution creates challenges that are non-trivial to solve if we want to maintain scalability and avoid computation and bandwidth bottlenecks.

2. **Consensus protocol:** The consensus protocol in Cardano is PoS, while ZILLIQA uses pBFT. The two consensus protocols work under different threat models. PoS assumes a cryptoeconomic model where the adversary is not willing to lose a collateral for malicious interests. On the other hand, pBFT assumes a harsher adversary model where miners are Byzantine and can arbitrarily deviate from the protocol. The other advantage that pBFT has over PoS is finality — this means that ZILLIQA does not need confirmations.

3. **Development principle:** Cardano's most recent release Byron (September, 2017) is an implementation with centralized nodes. Cardano aims to eventually transition towards a decentralized architecture. ZILLIQA's development philosophy is different. We believe that such transitions are prone to issues and may require an enormous change in the codebase that may take a considerable amount of time. With ZILLIQA, we have built a decentralized infrastructure from the ground up and hence can potentially provide an accelerated path to a scalable solution.

11. ZILLIQA vs Stratis?

ZILLIQA and Stratis can be compared wrt their approach to scalability. ZILLIQA solves the scalability problem by implementing the idea of sharding, while Stratis solves the problem by implementing several private chains (sidechains) that are attached to a main Stratis chain. ZILLIQA's approach has the following advantages over a scaling solution `a la sidechain.

1. **On-chain scalability:** ZILLIQA scales the throughput of the main chain, while, a sidechain solution is still somewhat an indirect solution and its performance is still pegged to the scalability of the main chain.

2. **Security:** Security of a sidechain solution relies on the security of each sidechain. If one does not have enough miners on the sidechain, then the system is not very secure and decentralised.

3. **Consensus protocol:** ZILLIQA uses a standard practical Byzantine fault tolerant (pBFT) protocol to reach consensus in each shard. While, Stratis employs PoS for consensus on the main chain. PoS assumes a cryptoeconomic threat model where a block validator is not willing to lose a large

amount of collateral or stake. pBFT assumes a more hostile adversary willing to do anything to subvert the system.

Page 8

4. **Finality:** pBFT in ZILLIQA gives finality to transactions that PoS does not. In other words, no confirmation is required in ZILLIQA.

12. ZILLIQA vs 0Chain?

0Chain is a protocol that uses several parametrized chains to handle different aspects of the blockchain such as data, smart contract, miners, etc. Such a modular architecture then makes it easier to fork and change the protocol parameters in a streamlined manner. The main comparison point between ZILLIQA and 0Chain is their approach to scalability. While, ZILLIQA uses the idea of sharding with pBFT as the underlying consensus protocol, 0Chain uses dPoS for consensus. There are several advantages of the approach that ZILLIQA takes:

- **Transaction finality:** pBFT in ZILLIQA gives finality to transactions that dPoS in 0Chain does not give. This means that ZILLIQA does not require confirmations.
- **Adversary model:** dPoS assumes a crypto-economic adversary model where miners are not willing to lose their deposit to subvert the system. While, pBFT assumes a much harsher Byzantine model. In the Byzantine model, adversaries have no economic hindrance to attack the system.
- **Decentralization:** dPoS in 0Chain (as per the initial parameters given in the whitepaper) works with a set of only 3 block producers and 6 block verifiers (aka secondaries). We believe that such small numbers of block producers and miners can severely impact the decentralization aspect of the protocol. On the other hand, pBFT in ZILLIQA is run on a subset of 600 nodes and hence we do not sacrifice on decentralization.

  The 0Chain whitepaper however does mention that the number of block producers can take other values such as 21, 100, 2048, etc. But, the choice of 3 as an initial parameter is certainly not secure as it becomes easy to attack the block producers and pull them down. Also, it is clear that increasing the number of block producers from 3 to say 2048 will undoubtedly increase the latency and lower the throughput.
- **Linear scalability:** Sharding in ZILLIQA gives linear scalability that dPoS in ZILLIQA does not give.

13. ZILLIQA vs ORBS?

ORBS seems to be using the SPECTRE protocol to achieve high throughput and low latency. SPECTRE is a protocol that uses a DAG-based ledger to record the blocks mined on top of previous blocks. The protocol then uses a virtual voting mechanism to get a partial order on the blocks. However, as discussed in the SPECTRE paper, the protocol does not give a total order on the blocks and hence is not suitable for running smart contracts. Below is an excerpt from the paper:

> "Key to SPECTREs achievements is its willingness to delay the decision regarding visibly double-spent transactions. It thus solves a weaker problem than traditional consensus protocols. This fact also makes it less suitable for systems like Ethereum, where a total order over transactions is required."

ZILLIQA on the other hand relies on a standard byzantine fault tolerant consensus protocol named pBFT that guarantees finality to transactions. Furthermore, as ZILLIQA uses a linear chain of blocks, it also gives a total order to the mined blocks, thereby making it possible to run smart contracts.

14. ZILLIQA **vs OmniLedger?**

ZILLIQA is inspired from Omniledger and its predecessors ByzCoin and CoSi. Omniledger is an academic paper and putting the theory into practice required ZILLIQA to modify some of its underlying building blocks, e.g., modifying the CoSi protocol used in Omniledger which is vulnerable to some attacks.

CoSi implements Schnorr multi-signature. It assumes an honest leader that aggregates contributions from various signers and at the end of the protocol, generates a multi-signature. ZILLIQA does not assume the leader to be honest. In fact, CoSi protocol is vulnerable to the following two attacks that we prevent in ZILLIQA.

1. The first attack assumes that the leader is malicious. At the start of the CoSi protocol, the leader may announce that he wishes to generate a multi-signature on a specific message say “This transaction is a double spend”. However, the protocol does not oblige the leader to stick with this message in the next steps. He may in fact change it to any message of his choice say “This transaction is not a double spend”. There is no possible way for signers to detect this. As a result, the signers may end up signing messages “blindly”.

2. The second attack assumes that the signers are malicious. The goal here is to mount a DoS attack. The leader in CoSi performs no sanity check on the contributions sent from each signer. As a result, it becomes possible for a signer to send an invalid contribution that cannot be detected until the end of the protocol when the signature verification is done. This attack works even when a single signer is malicious.

We prevent these attacks by adding extra steps and message rounds to the CoSi protocol both on the leader’s side and the signer’s side. The steps ensure that malicious behavior gets detected as early as possible.

There are also other differences. For instance, Omniledger also proposes to do state sharding, which ZILLIQA does not do. We believe that state sharding has a lot more challenges. ZILLIQA will also have a non Turing-complete smart contract language which OmniLedger does not have.

15. ZILLIQA **vs Waves NG?**

Waves NG implements a protocol inspired from Bitcoin NG with PoS being used for Sybil resistance. ZILLIQA uses PoW for Sybil resistance. Since ZILLIQA is inspired from ByzCoin which in turn is inspired from Bitcoin NG, there are some design similarities between Waves NG and ZILLIQA. However, the main difference between the two protocols is that Waves NG does not do sharding.

16. ZILLIQA **vs Cypherium?**

Cypherium implements a protocol inspired from ByzCoin with PoW being used for Sybil resistance. Since ZILLIQA is also inspired from ByzCoin, both protocols share some common design. The difference between ZILLIQA and Cypherium is that the latter does not do sharding. Hence, if the size

of the consensus group in Cypherium becomes large, the underlying consensus protocol will end up with a communication bottleneck. This also means that Cypherium does not scale linearly like ZILLIQA.

17. ZILLIQA **vs IOS?**

The IOS project proposes a scalable blockchain platform that employs the idea of sharding to scale. IOS assembles primitives from several academic papers such as the ones that propose Collective Signing, ByzCoin, RandHound-RandHerd and OmniLedger. More precisely, IOS adopts the idea of signature aggregation from Collective Signing, the idea of running pBFT as a consensus protocol with signature aggregation from ByzCoin, the concept of transaction and state sharding from OmniLedger and uses a bias-resistant distributed random number generator from RandHound. In fact, OmniLedger itself combines other works and hence, IOS is very close to OmniLedger. The designers of IOS also mention this in the whitepaper.

ZILLIQA employs some of the ideas from Collective Signing and ByzCoin and hence it bears some similarities with IOS. The two projects however differ in the following aspects:

1. **Sybil resistance:** The IOS whitepaper does not explicitly discuss the way it prevents Sybil attacks. It appears that Sybil attacks are prevented via a stake-based mechanism. ZILLIQA on the other hand uses PoW for Sybil resistance.

2. **Consensus protocol:** IOS uses a combination of pBFT and a proof-of-believability (PoB) protocol. PoB is very similar to a dPoS protocol where a validator proposes a block. Consensus in IOS happens in two phases. In the first phase, a PoB protocol is run among a small set of nodes called believable validators, whereby, they propose a small set of transactions. Later in the second phase, the proposed transactions are verified by a larger set of nodes called normal validators. Normal validators do not verify every transaction proposed by the believable validators. Instead, they first decide which transactions to verify and then they run the pBFT consensus protocol to agree upon the transactions.

   The whitepaper however does not give a security analysis of the proposed PoB protocol, as a result it is difficult to understand its security properties. ZILLIQA on the other hand uses a standard pBFT protocol for consensus. The safety and liveness properties of pBFT have been well established in the academic literature.

3. **Size of the consensus group:** A given shard in IOS will have several groups of believable validators. Each group proposes a different set of transactions. However, the size of each group is 1. This implies that it should be fairly easy to single out all the believable validators in a shard and attack them. Moreover, as believable validators are assigned using known metrics such as reviews, reputation, etc., it is also easy to narrow the set of believable validators beforehand with the available public information.

4. **Threat model:** ZILLIQA assumes a well defined Byzantine adversarial model where less than 1/3 of the total network is byzantine. On the other hand, the threat model in IOS is not-so-well defined. IOS employs two different consensus protocols namely pBFT and PoB. PBFT assumes a byzantine adversary model while PoB assumes a rational adversary not willing to lose large deposits to subvert the consensus protocol. Hence, it is not clear how a node in the IOS network adaptively switches from one model to the other and whether such a hybrid threat model is justifiable.

Page 11

5. **UTXO model:** Since IOS closely follows the OmniLedger protocol, it employs a UTXO-based transaction model to handle inter-shard communications. Roughly speaking, a user creates a transaction with some inputs and outputs. The shards handling the inputs then lock those inputs and issue a receipt certifying that the inputs have been locked and can be consumed as outputs. Once the receipt is issued, the outputs can be validated by the shards handing the outputs. Since, IOS claims to be a blockchain platform capable of running smart contracts, it's unclear how a UTXO model can efficiently handle smart contracts that need to manipulate global state variables unlike UTXOs that only handle certain inputs.

   ZILLIQA on the other hand uses an account-based model and hence eliminates any such restriction on running smart contracts.

6. **Transaction finality:** Since the normal validators do not validate every single transaction, it is possible that a malicious believable validator proposes an invalid (or double spend) transaction that gets detected only after-the-fact. As a result, a committed transaction is not final and one needs to wait for confirmations. Since, ZILLIQA uses pBFT among a large group of miners (the minimum being 600) and verifies every single transaction, transaction finality is guaranteed.

7. **Attacks on collective signing:** Both ZILLIQA and IOS use Collective Signing (CoSi) for signature aggregation. However, the protocol as described in the CoSi paper is vulnerable to two specific attacks.

   CoSi implements Schnorr multi-signature. It assumes an honest leader that aggregates contributions from various signers and at the end of the protocol, generates a multi-signature. ZILLIQA does not assume the leader to be honest. In fact, the CoSi protocol is vulnerable to the following two attacks that we prevent in ZILLIQA.

   (a) The first attack assumes that the leader is malicious. At the start of the CoSi protocol, the leader may announce that he wishes to generate a multi-signature on a specific message say "This transaction is a double spend". However, the protocol does not oblige the leader to stick with this message in the next steps. He may in fact change it to any message of his choice say "This transaction is not a double spend". There is no possible way for signers to detect this. As a result, the signers may end up signing messages blindly.

   (b) The second attack assumes that the signers are malicious. The goal here is to mount a DoS attack. The leader in CoSi performs no sanity check on the contributions sent from each signer. As a result, it becomes possible for a signer to send an invalid contribution that cannot be detected until the end of the protocol when the signature verification is done. This attack works even when a single signer is malicious.

   We prevent these attacks by adding extra steps and message rounds to the CoSi protocol both on the leader's side and the signer's side. The steps ensure that malicious behavior gets detected as early as possible. It is not whether how IOS prevents the afore-described attacks.

18. ZILLIQA vs Cosmos Tendermint?

Cosmos Tendermint is a consensus protocol that is a mix of PoS and a custom BFT protocol, while ZILLIQA is a mix of PoW and pBFT. PoW in ZILLIQA and PoS in Cosmos Tendermint are used mainly for Sybil resistance.

Page 12

Classical BFT protocols including the one used in Tendermint are however, limited in terms of scalabilty. In general, those protocols do not scale beyond 100 nodes. Hence, if the network becomes larger than say a few hundred nodes, a trivial application of the underlying BFT consensus protocol will end up creating a bottleneck in communication.

Tendermint circumvents this problem in the following way: It selects a small set of validators from the entire network. The custom BFT protocol is then run only amongst the set of validators. If the set of validators is say around 100 nodes, then the bottlenecks do not appear. Tendermint currently uses a validator set of 100 nodes but has plans to extend it to 300 nodes.

Since the consensus is run only amongst a subset of nodes from the network, a high throughput is not surprising. However, using a validator set of 100 nodes also introduces centralization risks. The validator nodes are selected wrt the amount of collateral they put into the system.

ZILLIQA takes a different approach. Instead of delegating the consensus to a subset of nodes, it shards the network into smaller networks sufficiently large to ensure that security and decentralization are not compromised. Each sharded sub-network can then process transactions in parallel. ZILLIQA also uses primitives such as Schnorr multi-signature scheme to ensure that pBFT within a shard does not become a bottleneck even when the shard size is around 600 nodes. The other advantage that sharding gives is linear scalability. The throughput in ZILLIQA increases (almost) linearly with the number of shards (or nodes).

It is clear that more nodes in the network yield better decentralization. However, a larger network in Tendermint's consensus group creates communication bottlenecks. On the other hand, ZILLIQA with its sharding technology can leverage a large network to increase the throughput rate.

19. ZILLIQA **vs Hashgraph?**

Hashgraph is a consensus protocol that is currently implemented in a permissioned setting. It does not handle open-membership which means that it is not open to any arbitrary node. In order to become permissionless, the project discusses mechanisms such as employing PoS to elect the consortium and then using the Hashgraph consensus algorithm within the consortium. ZILLIQA on the other hand is a public blockchain platform that uses pBFT for consensus and PoW for Sybil resistance.

It is also not clear how well Hashgraph will scale, as the number of messages is at least linear in the number of nodes. For a small cluster of nodes like a hundred nodes in the permissioned setting, it is still acceptable. However, in the permissionless setting with over thousands of nodes, the protocol will be heavily affected by network congestion of messages. Interested readers are encouraged to read the following extended critique made by one of the ZILLIQA team members: http://bit.ly/2jNOM5c

20. ZILLIQA **vs RedBelly?**

Red Belly has a different target use case where they focus only on the consortium blockchain model whereas ZILLIQA is a public blockchain. In a consortium blockchain model, consensus participants are restricted to the members of the consortium and identities of the consortium members are known by all the participants. These identities provide a natural protection to the consensus protocol against Sybil attacks. Also, Red Belly's experimental results seem to be based on a small network size, e.g., up

to 260 nodes.

21. ZILLIQA **vs Universa?**

Universa is a permissioned blockchain, while ZILLIQA is a public blockchain. Hence, comparison given in Question 20 holds here as well.

22. ZILLIQA **vs Raiblocks?**

The following is an excerpt from their whitepaper:

> RaiBlocks is able to make use of a fixed quantity it controls - the genesis amount, to determine normal participation e.g., quorum, and to limit participation directly to users who have an interest in maintaining the system.

This means that Raiblocks is to some extent a permissioned system with nodes which are partially trusted.

23. ZILLIQA **vs OMG?**

OMG is a decentralized exchange platform that requires an underlying blockchain protocol to work with. OMG uses Ethereum. ZILLIQA on the other hand is a blockchain platform. Simply put, OMG is an application while ZILLIQA is a system on which you may want to run OMG. OMG does not solve the scalability problem on its own. You will still need to have an underlying blockchain that has high throughput.

24. ZILLIQA **vs Golem?**

The Golem infrastructure is designed to run on Ethereum in the form of smart contracts. This means that a Golem application's throughput will be limited by Ethereum's throughput. In other words, if a Golem application becomes really popular and leads to high volume transactions, then the underlying network won't be able to handle that volume.

On the other hand, ZILLIQA's computation platform relies on its own underlying sharded architecture and hence can leverage its high throughput. The other difference is regarding the smart contract language. Golem is based on Ethereum which has a Turing-complete language. In case of ZILLIQA, the team believes that Turing-completeness is not necessary for many interesting applications. The other benefit of Turing-incompleteness is that it makes programs easy to reason about and it also becomes possible to prove interesting properties about such programs.

25. ZILLIQA **vs SONM?**

SONM seems to follow a similar path to Golem and relies on Ethereum for consensus. Hence, similar comparison holds. Refer to Question 24 for a comparison between ZILLIQA and Golem.

26. ZILLIQA **vs Tezos?**

Page 14

Tezos essentially gives a way to handle updates to a blockchain protocol from say Bitcoin to Bitcoin Cash. It will work as follows: Tezos will first be instantiated to run Bitcoin and then stakeholders will propose a protocol change to Bitcoin Cash. They then vote for the change. If a supermajority is up for it, then Bitcoin gets replaced by Bitcoin Cash.

Tezos can be seen as a governance enabler. Tezos code will be written in OCaml and hence will eliminate known issues in languages such as C++. Tezos will be using PoS to reach consensus on updates to the underlying blockchain (Bitcoin in the example provided).

Tezos on its own however, does not invent the new change to the underlying blockchain, i.e., the idea of replacing Bitcoin by Bitcoin Cash or say ZILLIQA must come from the research/developers' community. Hence, ZILLIQA and Tezos are completely orthogonal projects. In fact, the community has to come up with ZILLIQA first and then propose it as an update to Bitcoin.

The only common point that can be compared between ZILLIQA and Tezos is the underlying consensus protocol. Tezos uses PoS. ZILLIQA uses pBFT. pBFT gives you finality, PoS may not, as confirmations are required.

Index

0Chain, 8

BFT, 4, 5, 11
Bitcoin NG, 9
Bitshares, 4
Block finality, 2–7, 14
ByzCoin, 9, 10

Cardano, 6
Collective signing, 10
CoSi, 9
Cosmos Tendermint, 11
Cypherium, 9

dBFT, 4
dPoS, 4

EOS, 4
Ethereum, 2, 3

Golem, 13
Governance, 13

Hashgraph, 12

IOS, 10
IOTA, 3

NEO, 4

Off-chain solution, 3
OMG, 13
OmniLedger, 9, 10
On-chain solution, 3
ORBS, 8

pBFT, 4, 7, 10
Permissioned blockchain, 12, 13
Plasma, 3
PoB, 10
PoS, 2, 5–7, 10, 11, 14
PoW, 2

Radix, 5
Raiblocks, 13
Raiden, 3
RChain, 5
RedBelly, 12

Scalability, 2
Sharding, 8
Side-chain, 7
Side-chain solution, 3
Smart contract, 2, 8
SONM, 13
SPECTRE, 8
State sharding, 5
Stratis, 7
Sybil resistance, 2, 6, 9–12

Tezos, 13
Transaction finality, 10

Universa, 13

Waves NG, 9