HashNET: Beyond Blockchain Technology

Josip Maričević, Tolar.io

February 2018.

OVERVIEW

The arrival of the blockchain technology introduced the world of decentralization, therefore, challenging our preconceived perspectives of the current social, political, and economic systems, most notably, the central banking system.

This technology, however, does have several shortcomings regarding performance, ease of use, and service quality. HashNET consensus uses "redundancy reduced gossip" and "virtual voting" protocols based on a distributed computation and algorithms from theoretical computer science which provides a fair and fast, byzantine fault tolerant consensus algorithm. It is a new consensus substitute platform inspired by the innovative development of Hashgraph methodology, and is designed to run on a non-permissioned (public) network thereby reaching a larger community.

INFORMATION TRANSFER SOLUTION

HashNET provides a novel solution to computational and communicational difficulties of maintaining large-size public distributed ledgers. The key innovation is our efficient asynchronous distributed consensus protocol on an appropriately designed directed acyclic network structure. Our consensus protocol belongs to a class of gossip-based protocols, which provide advantages over structure-based group communication algorithms as they can handle large group sizes, sporadic sources, high user churns, and random network failures (for the details of the theoretical support, see, e.g., [1, 2]).

To ensure history immutability through time, which is an important property for public distributed ledgers, network nodes are connected using hash pointers ([3] provides an introductory technical description). It is well established (e.g., [4,5]) that as long as the selected hash function is secure, already agreed upon history cannot be changed retroactively.

One of the primary goals in designing HashNET was a significant reduction of computational and communication resources needed to operate and maintain the system. With this goal in mind, we design a variant of a Redundancy Reduced Gossip (RRG) protocol for information transfer on appropriately designed network. Such RRG protocols achieve considerably lower traffic load than conventional push-based gossip protocols and conventional push-pull gossip protocols, while maintaining the same probability of successful delivery [6].

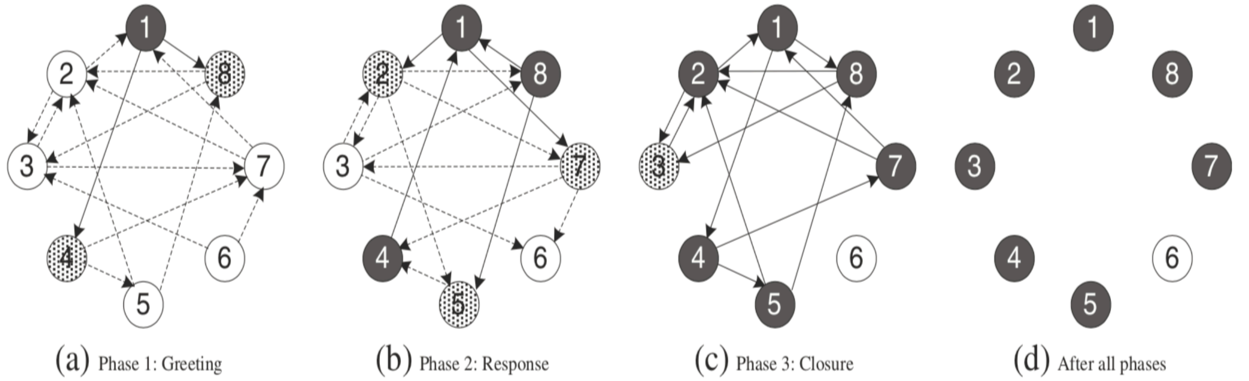

Figure 1: Information frame from peer 1 are diffused via gossip that are 3-phase message exchanges in one cycle: (a) Phase 1: Greeting, (b) Phase 2: Response, (c) Phase 3: Closure, and (d) After all phases; Fanout=2; Peer(s) shaded black is infected at the beginning of the phase; Peer(s) shaded grey is infected at the end of the phase; Peers shaded white remain uninfected at the end of the phase; A solid line refers to a message containing a frame; A dotted line refers to an empty message. [6]

Traffic load in Figure 6 is measured in terms of average number of copies of an information frame received by each peer. But each message contains protocol headers, and the resulting overheads for the two compared protocols are different. When n=100 and the average number of active peers is less than 3, with c=2, the overhead in RRG is around 20% of total traffic. Most of the overhead is contributed by the APL, where membership information is carried and requires 6 bytes per peer. In the same settings, the overhead of the conventional push gossip and the conventional push-pull gossip are around 40% of total traffic. Most of the overhead is contributed by the buffer-map, which is at least 12 bytes per gossip message.

It is important to note that our protocol generates a smaller number of messages than the fully connected peer-to-peer overlay approach in N-to-N communication. The number of messages in our protocol is only around 24% that of the fully connected overlay approach.

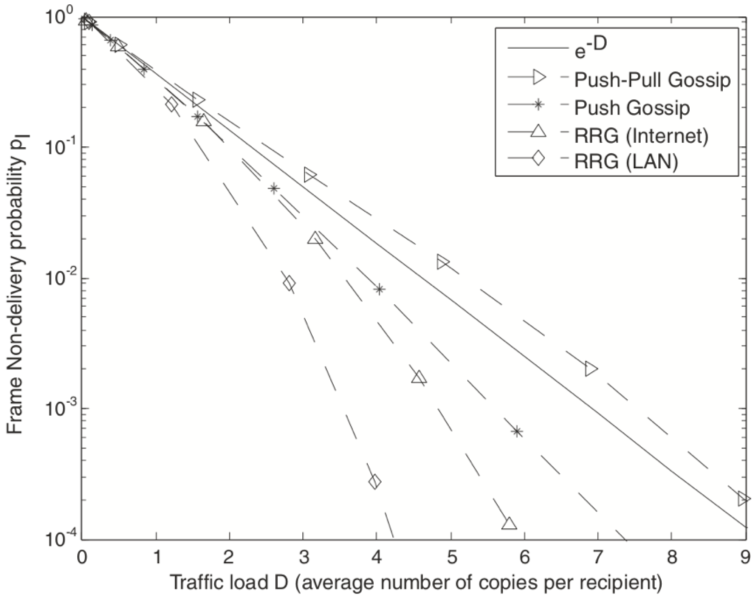

Figure 2: Performance comparison of redundancy reduced gossip (RRG), the conventional push gossip and the conventional push-pull gossip. [6]

COMPUTATIONAL AND COMMUNICATIONAL EFFICIENCY

While RRG and other asynchronous distributed consensus protocols provide communicational and computational efficiencies, additional implementation improvements are necessary to handle large fast growing systems. A direct implementation of such protocols could require exchanging as much as O(n^3) messages for reaching a consensus on a single binary outcome (e.g., see [7]), which would make them not practical and unsustainable for systems where the number of nodes, n, is large. Thus, it is imperative to implement the consensus protocol in a way that minimizes communicational load due to information transfer among nodes.

However, we leverage the fact that every node has a sufficient information on the entire HashNET structure, including information about events and their propagation through the network. We use this information to compute content of the vast majority of messages required by our RRG protocol, thereby eliminating the need for sending them and, consequently, significantly reducing communication requirements. (A somewhat similar approach towards reducing communication requirements in an implementation of a different consensus protocol has been proposed in [8]. Unlike our system, a critical requirement in [8] is that the number of nodes is constant and must remain fixed constant throughout.)

An important prerequisite for an efficient computation of consensus is that the total number of nodes ("voters") needs to be known. This provides an inherent difficulty for an implementation involving public ledger, as the number of nodes can vary greatly. We overcome this difficulty by assigning to every node the vote weight that is equal to their stake at a given point of time. Since, at any given point of time, the current supply of coins in the network is known and fixed, this approach ensures proper consensus computations. Thus, by assigning node weight to be its stake in network, we achieve the ability to calculate votes instead of waiting for and/or sending actual votes over the network. As the protocol provides a Proof-of-Stake blockchain discipline, it offers qualitative efficiency advantages over blockchains based on proof of physical resources (e.g., proof of work).

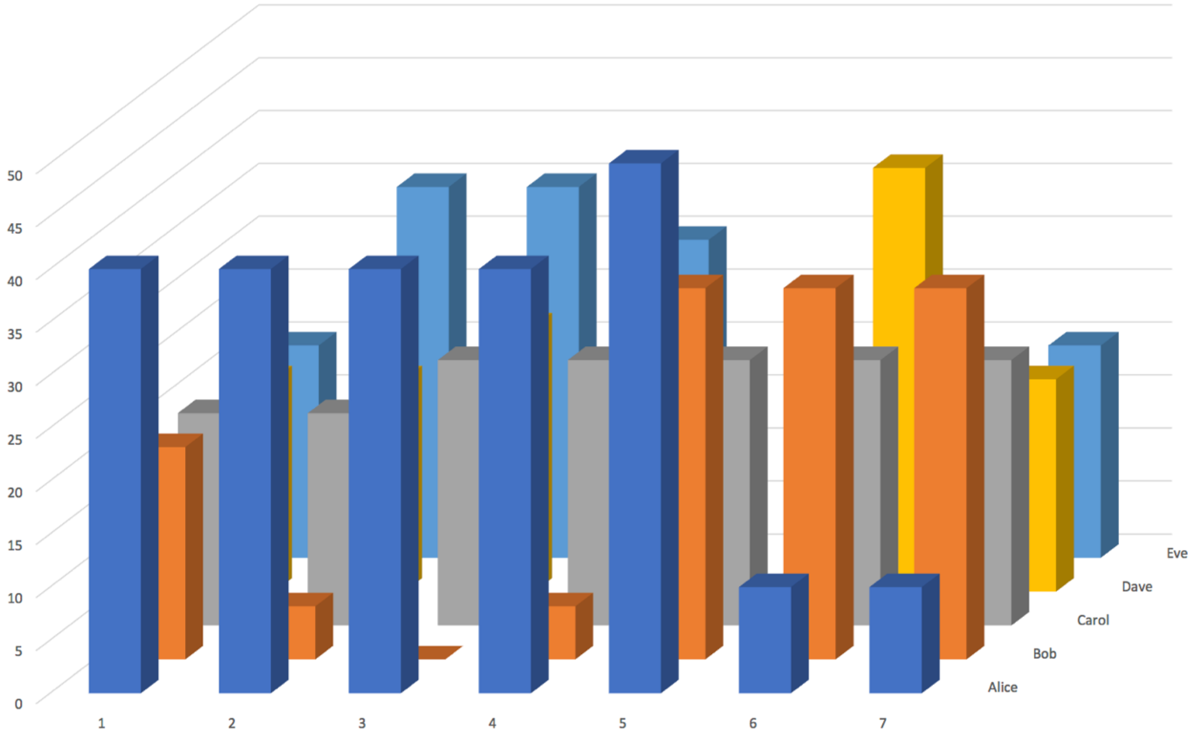

Figure 3: Illustration of HashNET voting weight and stake changes through the voting rounds as a result of transaction events. Do note that in each round some transactions happen that transfer Tolars from one participant to another, however total vote weight through the network stays the same in each round, enabling us to have virtual voting without knowing the number of nodes in advance.

Furthermore, with such weight assignment to nodes, incentives are perfectly aligned with the stakes, suggesting reduced strategy space for obstructive and malicious behavior. Indeed, it can be shown that the reward mechanism for incentivizing Proof-of-Stake can be constructed in such a way that truthful behavior is an approximate Nash equilibrium, thus neutralizing selfish-mining attacks (see, e.g., [9, 10, 11]).

REPUTATION BASED SYSTEM

In addition to the Proof-of-Stake based discipline, a reputation based system is introduced as an additional control and verification mechanism. This allows us to measure 'severity' of network protocol violations: some violations could simply be a consequence of sporadic unfortunate network conditions (small negative impact on reputation), while others could be attributed to a specific malicious intent (large negative impact on reputation). Similarly, for nodes which are consistently "fault-free", i.e., consistently communicate correct information and locally compute consensus correctly, a positive reputation is slowly built-up over time. All nodes whose reputation falls behind a certain negative reputation threshold are banned from the network for the amount of time decided by an exponential backoff algorithm. (Such reputation based systems have already been proven useful for self-regulating in many P2P applications [12].)

Information about node reputation is realized as part of events distribution algorithms.

The reputation score is computed iteratively and cumulatively and involves both pattern analysis for a behavior over time of a single node, as well as randomized verification checks.

ACHIEVING CONSENSUS

Potentially faulty or malicious nodes provide an additional challenge of implementing a consensus protocol in an asynchronous environment. In fact, it is well-known that it is theoretically impossible for any deterministic protocol to reach an agreement in such environments [16]. Thus, our implementation resorts to a class of so-called iterative randomized approximate consensus algorithms. The goal is to allow fault-free nodes to agree on values, overcoming the obstacle posed by (possibly incorrect or unreliable) information disseminated by faulty nodes.

Theoretically, the most challenging case are leader-based consensus algorithms, which rely on a small number of nodes that cannot be faulty (e.g., such as [8]). Such protocols are prone to failure and usually exhibit several unresolved issues in the case of a malicious node becoming a leader [13]. In contrast, fully distributed protocols (in which every node in the network could be decisive and in which no node is always decisive) allow for design of algorithms that overcome faulty nodes as long as fault-free nodes form a supermajority at any point of time. More precisely, the approximate consensus algorithm can be constructed even on fully connected graphs provided that the total number of nodes n > 3f, where f is the number of faulty nodes [17]. Our implementation leverages this result: if a consensus is not reached after a number of rounds of our RRG protocol, we initiate a sequence of "random rounds". In a random round, any non-faulty nodes will chose their votes at random, and will have a non-zero probability of all choosing the same vote. The randomization is such that guarantees the correct agreement among non-faulty nodes with high probability. Thus, implementing a small number of random rounds results ensures convergence to consensus (i.e., the probability of failure is converging to zero at an exponential rate). Finally, we note that randomization does not need to pose a significant computational burden and can be manipulation-free, as bits provided by event hash (that need to be computed anyway) can be used as a source of pseudorandom data [18].

HASHNET UTXO STORAGE REQUIREMENTS

In addition to aforementioned speed performance guarantees made possible by HashNET design, we are also able to guarantee improvements in the global data storage size. Specifically, HashNET data storage is designed by generalizing the approach laid out in MimbleWimble whitepaper [14]. The main benefit of this approach is that it manages to simultaneously handle security and versatility.

We utilize concepts such as Confidential Transactions and "One-Way Aggregate Signatures" (OWAS), which are shown to provide private exchanges and better adaptability. The main idea behind OWAS is that when the outputs are created and destroyed, it is the same as they never existed. Consequently, to approve the entire chain, a client only needs to know when coins were inputted into the framework and what are last unspent yields. We then utilize Confidential Transactions to conceal the sums and OWAS, thereby obscuring the exchange diagram. This approach utilizes less space than, e.g., Bitcoin to enable clients to verify the blockchain. For instance, the Bitcoin blockchain is currently about 160GB in size, while HashNET would require a small fraction of that amount for the same amount of transactions, thus allowing even today's standard smartphones to act as nodes.

In our approach (similar to MimbleWimble), the beneficiary of transaction creates the blinding element which is utilized to demonstrate responsibility for coins. This is done through the "excess value", which is the contrast between the information sources and yields. This overabundance esteem is an arrangement of arbitrary numbers that guarantee that only the individual who created the blinding factor (the collector/receiver) can spend the coins. Thus, the blinding variables don't indicate zero any longer, but instead another number, resembling a private key. The important feature of our approach is that it is not interactive, which in turn creates efficiencies by eliminating the need for storing any redundant information on history of individual transactions. Rather than containing complete history of all transactions, the blocks (analogous to MimbleWimble) only have a list of new inputs, a list of new outputs, and a list of signatures which are created from the aforementioned excess value (Note that the latter list provides sufficient record of all historical transactions).

In summary, generalizing, adapting and implementing MimbleWimble approach for use on a HashNET provides a chain format that has excellent scalability, privacy and fungibility properties.

DEMOCRATIC USER-GOVERNED SYSTEM

One of the central features of HashNET design is a democratic governance system which allows for involvement the entire community and is open to everyone.

Specifically, any node in HashNET network can propose tenders, and then Magnus Consilium decides and votes on each such proposal. Magnus Consilium is comprised of all the masternodes with positive reputation in HashNET. Voting is decided by simple majority.

Next we provide examples of eligible tenders:

- social impact
- distributed governance

- contribution
- outreach
- extensiveness

FUTURE OF HASHNET

Decentralized applications

Once desired throughput is achieved, Ethereum Virtual Machine (EVM) will be deployed on top of network. The EVM is a virtual machine specifically designed to run untrusted code on a network of computers. Every transaction applied to the EVM modifies the State which is persisted in a Merkle Patricia tree. This data structure allows to simply check if a given transaction was actually applied to the VM and can reduce the entire State to a single hash (merkle root) rather analogous to a fingerprint.

The EVM is meant be used in conjunction with a system that broadcasts transactions accross network participants and ensures that everyone executes the same transactions in the same order. Ethereum uses a Blockchain and a Proof of Work consensus algorithm. Here, we will use HashNET.

The combination of EVM and HashNET makes for a fast and secure decentralized applications platform.

Quantum resistance

The elliptic curve signature scheme used by Bitcoin is well-known to be broken by Shor's algorithm [17] for computing discrete logarithms. That's why in the next 5 years it will be imperative to switch to alternative signature schemes that are believed to be quantum safe. Exact scheme that will be implemented in HashNET is still being decided on.

ROADMAP

Q2 2017

TOLAR project founded and funding received

Executive group established

Q3 2017

Blockchain Conference Rovinj

Initial network development.

First cryptocurrency experiments conducted.

Q1 2018

Launch of the Blockchain academy.

Recruiting process of key team members has started.

Core Tolar team created with 30 members.

Website published.

Tolar will introduce project publicly.

Start of the private sale phase and social media presence.

Q2 2018:

Initial version of HashNET whitepaper

Launch of initial mainnet chain with masternodes and staking system

Pre-ICO

Q3 2018:

Finalized version of HashNET whitepaper

Developers demonstration of HashNET

ICO

Q4 2018:

Exchange listing

Chain explorer

Mobile wallets

Q2 2019:

Public release of HashNET chain

Switch mainnet to HashNET protocol

Future development:

Smart contracts

Quantum resistant cryptography

REFERENCES

[1] Jelasity M., Montresor A. and Babaoglu O. 2005. **Gossip-based aggregation in large dynamic networks.** ACM Trans. Comput. Syst. 23, 3 (August 2005), 219-252. DOI=http://dx.doi.org/10.1145/1082469.1082470

[2] Allavena A., Demers A., and Hopcroft J. 2005. **Correctness of a gossip based membership protocol.** In Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing (PODC '05). ACM, New York, NY, USA, 292-301.
DOI: https://doi.org/10.1145/1073814.1073871

[3] **Hash pointers and data structures**, http://learningspot.altervista.org/hash-pointers-and-data-structures/, retrieved 23.03.2018.

[4] Rogaway P., Shrimpton T. (2004) **Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance**. In: Roy B., Meier W. (eds) Fast Software Encryption. FSE 2004. Lecture Notes in Computer Science, vol 3017. Springer, Berlin, Heidelberg

[5] Coron JS., Dodis Y., Malinaud C., Puniya P. (2005) **Merkle-Damgård Revisited: How to Construct a Hash Function.** In: Shoup V. (eds) Advances in Cryptology – CRYPTO 2005. CRYPTO 2005. Lecture Notes in Computer Science, vol 3621. Springer, Berlin, Heidelberg

[6] Wing-Hei Luk V., Kai-Sun Wong A., Chin-Tau Lea, Wentao Ouyang R., **RRG: redundancy reduced gossip protocol for real-time N-to-N dynamic group communication**, *Journal of Internet Services and Applications* 2013 4:14

[7] Correia M. M., Veronese G. S., Neves N. F., and Verissimo P. **Byzantine consensus in asynchronous message-passing systems: a survey. International Journal of Critical Computer-**

Based Systems, 2(2):141–161, 2011.

[8] Leemon B. **The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance**, *SWIRLDS TECHREPORT SWIRLDS-TR-2016-01, 2016.*

[9] **PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake**, https://pdfs.semanticscholar.org/0db3/8d32069f3341d34c35085dc009a85ba13c13.pdf

[10] Bentov I., Gabizon A., Mizrahi A.: **Cryptocurrencies without proof of work**. CoRR, abs/1406.5694 (2014)

[11] Kiayias A., Russell A., David B., Oliynykov R. (2017) **Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol**. In: Katz J., Shacham H. (eds) Advances in Cryptology – CRYPTO 2017. CRYPTO 2017. Lecture Notes in Computer Science, vol 10401. Springer, Cham

[12] Damiani E., De Capitani di Vimercati, Paraboschi S., Samarati P., and Violante F. 2002. **A reputation-based approach for choosing reliable resources in peer-to-peer networks.** In Proceedings of the 9th ACM conference on Computer and communications security (CCS '02), Vijay Atluri (Ed.). ACM, New York, NY, USA, 207-216. DOI=http://dx.doi.org/10.1145/586110.586138

[13] Cachin C., Vukolić M., **Blockchain Consensus Protocols in the Wild,** https://arxiv.org/abs/1707.01873v2

[14] Tom Elvis Jedusor, **MimbleWimble**, July 2016, https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt

[15] Mouton Y. H., **Increasing Anonymity in Bitcoin**, retrieved on 18.02.2018. https://download.wpsoftware.net/bitcoin/wizardry/horasyuanmouton-owas.pdf

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

[17] Shor P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Review, 41(2):303–332, jan 1999.