# A Blockchain Based on Gossip?
# – a Position Paper

Robbert van Renesse

Cornell University

A blockchain is an append-only sequence of blocks of arbitrary data. The two most popular approaches to blockchains are *permissionless blockchains* based on *Proof of Work* (PoW) and *permissioned blockchains* based on *Byzantine consensus* or *Byzantine Fault Tolerance* (BFT). The first is based on competitions between anonymous participants solving cryptopuzzles, while the latter is a cooperative approach based on mutual trust between participants. Major problems with PoW approaches include that the energy per transaction is enormous, the transaction rate is very low, and the latency is very high. A major problem with BFT is that membership is closed. Various other approaches to blockchains have been proposed to address these problems.

In this paper we propose yet another approach, based on *gossip* (*aka* epidemiological protocols) [1]. Gossip is an approach to agreement in so-called *eventually consistent systems*, and is particularly popular in NoSQL Key-Value Stores such as Dynamo, Cassandra, and so on. In a basic gossip protocol, there is a fixed group of participants. Periodically, each participant randomly selects a peer and exchanges state. This state is reconciled in a way so that all non-faulty participants eventually converge to the same state. It can be shown that this approach is efficient, converging in $O(\log N)$ gossip rounds where $N$ is the number of participants, even in the face of participants failures and message loss [1]. Moreover, gossip protocols are amenable to open and dynamic membership whereby the membership itself is gossiped along with other state [4].

To make the application of gossip to blockchains more concrete, we propose, for simplicity, a setting similar to the Bitcoin blockchain. Blocks are 1 MByte. We assume participants have access to a good source of time, such as a GPS clock. We divide time into 10 minute epochs. Each hour would have six such epochs starting at the top of the hour. Each epoch is further subdivided into 10-second gossip rounds, allowing for 60 rounds of gossip. The intention is for the participants in an epoch to use gossip to agree upon a block. Note that the transaction rate would be the same as for the Bitcoin blockchain—here we only try to address energy consumption.

Each participant, in the background, collects transactions from clients, and at the top of an epoch fills a block with new transactions. The block also contains a *history hash*: a hash of the content of the previous block on the chain, as determined by the participant. The participant then starts gossiping its block and learning about the blocks of other participants. Before we go into further specifics about how the participants may end up agreeing on a particular block, we point out that gossip is prone to Byzantine attack. We will introduce and address various attacks as we go.

The first question is how participants would end up agreeing on a block. The straightforward approach is to order blocks according to some agreed-upon metric. Each participant would simply keep track of the maximum block that it has heard from and gossip this block, possibly along with membership information. A participant would only consider blocks that contains a history hash matching its concept of the block agreed upon in the last epoch. If all participants were honest, then this approach would lead to a very high likelihood that all participants agree on the maximum among the proposed blocks at the end of the epoch.

Unfortunately, a Byzantine participant could introduce one or more new blocks that are higher than the previous maximum in the last few rounds of the epoch and cause the participants to end up disagreeing with one another. This is similar to the concept of *forks* in PoW chains, where miners advertise new blocks at approximately the same time, either accidentally or with selfish intention. In Bitcoin chains, this is resolved by the *longest-chain-wins* policy: when one branch in the fork has become longer than the other(s), the longest branch becomes part of the main chain. Because it is highly unlikely that branches both keep adding a block at exactly the same times, forks are *metastable.*

Can we create a gossip protocol that is also metastable in a way that once agreement has been reached it would be very hard to destabilize it? The answer is yes. Instead of ordering the blocks a priori based on some metric, the participants use a randomized approach that converges with high probability. In this approach, each participant keeps track of a table with a row for each peer (including self). Each row in the table contains

- a unique identifier for the peer;
- a gossip round number for the peer;
- the peer's *favorite* block in that round;
- a public key signature to prevent forging and tampering.

When two peers gossip, they reconcile their tables by adopting, for each participant, the row with the highest round number. Once reconciled, a participant counts for each block in the table how many participants favor that block. The participant then chooses the most favored block as its favorite block and increments its round number in its own row. If there is a tie among several blocks, the participant selects one of them uniformly at random.

In order to prevent an attack whereby a participant tries to gossip with every other participant in every gossip round, we use a *pull gossip*: a correct participant explicitly requests the state of $k$ randomly chosen peers in each round, rather than sending its state to the peers. This way each correct participant updates its state based on that of at most $k$ participants in each round. In the rest of this paper, we use $k = 1$.

Figure 1 shows results of simulations of this protocol for various numbers of participants up to 4096, slightly fewer than the number of miners in Bitcoin today (which is now close to 6000). Each experiment was run 250 times. The x-axis shows the number of participants on a log scale and the y-axis shows the number of rounds before all participants agree upon a block. With 4096 members it takes rarely more than 20 rounds to converge, and certainly within 60 rounds convergence is all but certain. Note that, like Bitcoin, we assume that the participants somehow form a connected network—without it, the blockchain could diverge until connectivity is established. BFT does not suffer from this problem.

With open membership, a Sybil attack is possible on the membership of the protocol described thus far. We leverage that, although it is easy to spoof an IP return address in an IP packet, it is

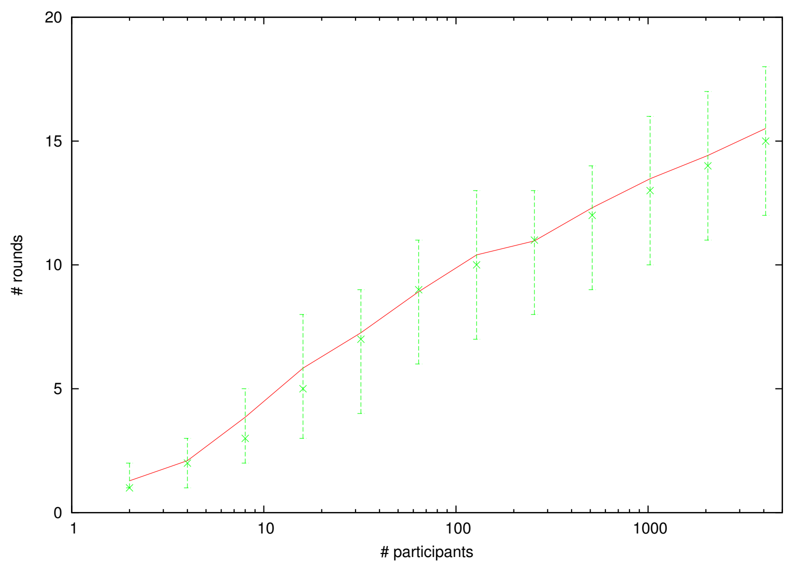

Figure 1: Gossip convergence time (in rounds) as a function of the number of participants. The line shows the average, and the error bars show the median as well as the 5th and 95th percentiles.

not easy to subvert routing and receive messages on a spoofed IP address. An adversary is thus limited by the number of public IP addresses that it actually has control over. More specifically, we propose that peers use TLS connections with endpoints identified by IP address.

In addition, we propose to use an intrusion-tolerant overlay network such as S-Fireflies [3, 2]. S-Fireflies is a self-stabilizing and Byzantine tolerant overlay network that provides its members with a view (approximation) of the current membership as well as each member with a small subset of the view. Those subsets induce a pseudo-random graph of members in which the correct members are connected and the diameter is logarithmic in the number of correct members, both with high probability. We have repeated the experiments above on a simulation of S-Fireflies and found slightly increased convergence times, but no more than one round of gossip.

S-Fireflies requires that each member has a public key certificate issued by a certification authority, further increasing the difficulty for an adversary to launch a Sybil attack. In addition, S-Fireflies would allow permissioned deployment by only accepting a certain class of public key certificates.

Another type of collusion attack against our approach is for a group of rational or Byzantine participants to agree before beginning of an epoch on a block to propose in order to significantly increase the chances of that block emerging as the agreed-upon block. Such collusion is of course possible, if not common, in PoW and BFT approaches as well. However, we will see that there is little incentive to do so as the proceeds of agreeing on a block are evenly divided among the participants.

We will now address is why somebody would deploy a node to participate in the proposed blockchain. In other words, what incentivizes participants? Some incentive comes from transaction fees. We would like, however, to pay participants for their work just like miners get paid for their work, if probabilistically.

In general, it is hard to prove for participants that they have actively participated in the gossip protocol. We propose to use platforms such as Intel SGX to provide such proofs. We envision that certified gossip code is released. When deployed in an SGX enclave, the code will acquire a timestamp from a certified time source, gossip for 24 hours, and provide a proof of correct execution. This proof can then be used to obtain renumeration in the next 24 hours. We envision that a certain amount of funds is distributed daily to all participants who can prove (using SGX or otherwise) that they have run the code during the previous 24 hour period. Note that unlike proposed blockchain protocols based on SGX such as PoET / Sawtooth Lake (intelledger.github.io), we do not rely on SGX for the correctness of the protocol—we only rely on SGX to renumerate participants.

At this point, a gossip-based blockchain is just a preliminary proposal, another point in the design space of blockchains, combining low energy with relatively open membership. Although it can most likely be sped up, in its proposed form it has the same (low) throughput as the Bitcoin blockchain but uses significantly less energy. Latency is at most 10 minutes, whereas for Bitcoin the latency until a block can be trusted to persist is closer to an hour. However, particularly in a permissionless environment, much analysis and refinement will need to happen to ensure that gossip-based blockchains can be trusted for critical applications.

## References

[1] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, 1987.

[2] E. Hoch, D. Dolev, and R. van Renesse. Self-stabilizing and Byzantine-tolerant overlay network. In *Proceedings of the 11th International Conference On Principles Of Distributed Systems (OPODIS'07)*, volume LNCS 4878. Springer, December 2007.

[3] H. Johansen, R. van Renesse, Y. Vigfusson, and D. Johansen. Fireflies: Secure and scalable membership and gossip service. *ACM Transactions on Computer Systems*, 33(5), June 2015.

[4] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-based failure detection service. In *Middleware'98, IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, September 1998.