EE 374: Blockchain Foundations **Stanford, Winter 2023**

Lecture 18: Streamlet

May 16, 2023

Lecturer: Prof. David Tse

1 Introduction and Recap

Topics discussed in this lecture is Streamlet, more specifically:

- Safety and Liveness
- Latency
- Partition tolerance and dynamic availability
- Accountability

Recap of Streamlet Protocol As was shown in the previous lecture, Streamlet[1] is a simple example of Byzentine Fault Tolerant (BFT) protocols. In Streamlet, time is divided into epoch of 2Δ, where Δ is the network delay bound. In each epoch a random leader L is chosen, L proposes a block to extend the longest notarized chain. Each party votes on the first valid block it receives. (A valid block is one that comes from the current epoch and also extends the longest notarized chain in the view of the party.) If a block gets more than $\frac{2}{3}n$ of n votes, it gets notarized. Life cycle of a Streamlet block:

- First, a valid block is proposed by the epoch leader, it does not have any votes yet.
- Second, if the valid block gets at least $\frac{2}{3}n$ votes, it gets notarized. Note that it is not yet confirmed.
- Third, notarized block gets confirmed (also known as finalized in the Streamlet paper). Confirmed block is a part of the ledger.

Important point here to note is that notarization does not mean confirmation! The question we are now interested in: what is a confirmation rule that will guarantee safety and liveness?

2 A Strawman Confirmation Rule

In the last lecture we proposed a Strawman confirmation rule - every notarized block gets confirmed. We showed that adversary, that controls less than $\frac{1}{3}n$ partys, cannot produce two notarized blocks in the same epoch by the quorum intersection argument. Is there any other way to cause safety violation? Yes! For instance, say adversary manages to propose a block in epoch 5 with no more than $2n/3 - 1$ votes, which is not enough to get the block notarized. This is possible if adversary releases a block in the second half of the epoch 5. In this case, there is an honest party that received this block in the beginning of the epoch 6 and won't vote for the block because it is from the previous epoch.

Then, the adversary can vote on this block in later epochs, so this block can be notarized at any time the adversary chooses. Hence, such protocol is not safe and notarization must not be equivalent to confirmation. Typically, in all BFT protocols multiple phases are passed before a block is confirmed.

What is the Streamlet confirmation rule?

3 Confirmation Rule

The confirmation rule that Streamlet uses is: if three blocks are notarized in consecutive epochs along a notarized chain, then all blocks until the second to last block are confirmed. For example, if blocks B_3, B_4, B_5 get notarized, then all blocks up to B_4 are confirmed.

4 Safety

Theorem 4.1 (Streamlet Safety Theorem)**.** *As long as $t < \frac{n}{3}$, under the confirmation rule "If three blocks are notarized in consecutive epochs, confirm until the second block" Streamlet protocol is safe.*

Proof. Suppose that there is a safety attack, this means that there is another chain which goes until the height of B_6. First consider the possibility that the block at the same height as B_6 is from the same epoch 6.

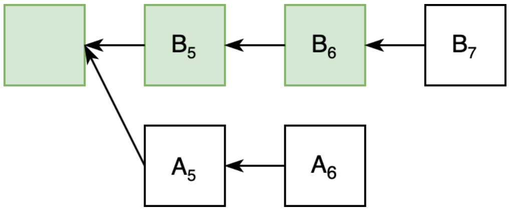

Figure 1: Safety violation scenario in Streamlet. Green block are confirmed. White blocks are notarized.

Can A_6 and B_6 coexist? No. If $t < \frac{n}{3}$, then it is impossible for both blocks to be notarized in the same epoch. In fact, the block at the same height as B_6 cannot be from epochs $5, 6$ or 7 by the same reasoning.

So, the adversarial blocks must be from epochs either before 5 or after 7. For example, as illustrated in the Figure 2. Let us see if this is possible. (You should consider the other cases yourself.)

Suppose an honest party P votes on B_5 in epoch 5. From the view of this party, B_5 was supposed to extend the longest notarized chain. Hence, A_3 cannot be notarized at this time in the view of P.

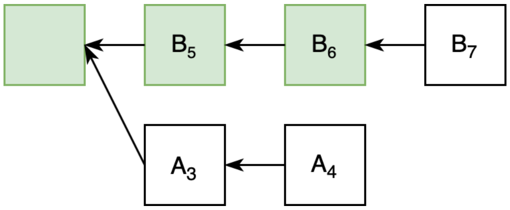

Figure 2: Safety violation scenario in Streamlet. Green blocks are confirmed. White blocks are notarized. The block at the same height as B_6 is from epoch 4.

So could P have voted on A_4 in epoch 4? No, because if A_3 is not notarized in its view in epoch 5, then A_3 is surely not notarized in epoch 4 in its view. So A_4 cannot be a valid block in its view in epoch 4, because it is not even extending a notarized chain.

Therefore, no honest party can vote on both B_5 and A_4. By the quorum intersection argument, it cannot be that both B_5 and A_4 are notarized. This is a contradiction. □

5 Liveness

By using the **confirmation rule** in the section above, it will be easy to give an intuition on how to prove **liveness** property for Streamlet. In particular, recall that the confirmation rule states that if there are three blocks notarized in consecutive epochs, the chain up to the second block will be finalized.

Intuitively speaking, one may think that this confirmation rule may imply that as long as three consecutive epochs all have honest leaders, then the ledger will have grown.This is because since there are more than $2/3n$ honest partys, they will all vote for each of these honest blocks in the three epochs. Hence they will all get notarized along a chain, and by the confirmation rule, the ledger would grow by 2 blocks.

While this is true when extending from Genesis block, in the general case, we cannot assume that the block the three honest partys are extending from is honest. In the liveness proof in the Streamlet paper, which is not trivial and would not be described here, guarantees that when there are 5 consecutive honest epochs, then a liveness event will occur.

Theorem 5.1 (Streamlet Liveness Theorem)**.** *If there are five consecutive epochs with honest leaders, then at least one more block is confirmed at the end of the five epochs compared to the beginning.*

6 Latency

To analyze Streamlet confirmation latency, our approach will be to use the liveness theorem. The high-level idea is that if we have five sequential epochs with honest leaders, then at least one block will have been added to the ledger. Thus, by calculating the probability of this event, we can derive an upper bound on how many epochs on average it will take for a transaction to get confirmed.

Let us split epochs into superepochs. In particular, let epochs 1-5 be the first super epoch, epochs 6-10 be the second super epoch, and so on. To prove something about latency, we will now calculate the probability that any superepoch contains entirely honest leaders; this is simply $(\frac{2}{3})^5 \approx \frac{1}{8}$. This means that the average time for a transaction to be confirmed is at most $1/(1/8) = 8$ super-epoches , which is 80Δ seconds! Note that to get an even tighter upper bound, one should compute the chance that any sliding window of five epochs is honest. However, we will not cover these details in our lectures. The important thing is that the average latency is a constant, independent of the security parameter κ. In contrast, the latency for longest chain protocols is linear in κ. Hence, we say that Streamlet has fast confirmation, while longest chain protocols has slow confirmation.

7 Partition Tolerance and Dynamic Availability

Recall that every epoch in the Streamlet protocol is of length 2Δ. Interestingly, however, our analysis of safety in Section 4 makes no assumption that the communication delay is less than Δ. Hence, the safety of the Streamlet protocol does not rely on the synchronicity assumption. In other words, the Streamlet protocol achieves safety no matter how long the message delays are or how badly the network may be partitioned. That said, the protocol may not be live, as message delays or partitions of the network may cause blocks to not be notarized. On the other hand, the liveness analysis does assume that the network is synchronous with delay bound Δ. Hence, we can say that while the protocol is always safe, the ledger is guaranteed to makes progress under *periods of synchronicity*. Longest chain protocols, in turn, are not safe under network partition. Indeed, two chains can be built in parallel on both sides of a network partition, and transactions are confirmed in both chains, violating safety.

Protocols which satisfy such properties are known as *partition tolerant*. It can be proven that partition tolerant protocols cannot function without a supermajority honest parties,i.e. $t < \frac{1}{3}$, which means that Streamlet is optimal in this regard [2]

Recall that in the previous lectures we covered the Proof-of-Stake Longest Chain (PoSLC) protocol. One difference to observe between PoSLC and Streamlet is that Streamlet is safe under asynchrony but only live under synchrony. This is in contrast to PoSLC which ensures safety only under synchrony. On the other hand, PoSLC provides a stronger liveness guarantee than Streamlet. First, a honest majority of the parties is sufficient to guarantee liveness in PoSLC, while a honest supermajority is needed in Stramlet. Second, PoSLC satisfies a *dynamic availability* property: even when the number of parties online is much less than the total number n, the protocol remains live if the majority of the parties *online* is honest, a stronger property than the majority of *all* the n parties. In contrast, Streamlet cannot be live once the number of parties online drops to less than $2n/3$, because there will not be enough parties to meet the voting quorum of $2n/3$.

8 Accountability

The security theorems we stated so far in this course makes assumptions on how many adversary partys there are. But what do we do when these assumptions fail?

The nice thing about Streamlet is you can provably show that many partys did not honestly follow the protocol when there is a safety violation.

Theorem 8.1. *In the Streamlet protocol, whenever there is a safety violation, at least $\frac{1}{3}$ of the partys can be provable held accountable for violating the protocol.*

Proof. A successful attack on Streamlet involves confirming two blocks at the same height in the chain. The safety analysis says that in all cases, at least $n/3$ partys have double-signed two blocks which they would not have double-signed if they were following the protocol. Thus, these 1/3 partys cn be held accountable.

□

This ability to provably determine who has violated the protocol brings us to the concept of **slashing**. When you start running a validator party, you will stake or deposit assets that will get slashed if you violate the protocol. Hence, there are economic incentives to not violate the protocol. Thus, this gives a new form of security: *cryptoeconomic security.* In this context, the stake of a party not only represents a *share* of the total network; it also represents

In contrast, longest chain protocols are not accountable. Adversary can violate safety of the longest chain protocol if she manages to displace a k-deep block. Is there a way to provably identify the adversary? Recall that safety of such protocols requires honest majority assumption and network synchronicity. Therefore, adversary can always claim that the network was partitioned and she simply didn't see the blocks in longest chain that she displaced. Moreover, because the partys who propose blocks are randomly sampled from a potentially very large population of partys, there can be not a single common party who double-sign on the two chains.

References

[1] B. Y. Chan and E. Shi. Streamlet: Textbook streamlined blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 1–11, 2020.

[2] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, apr 1988.