EE374 Scaling Blockchains **Stanford, Spring 2020**

Homework #3*

Due: Thurs, 4-June-2020, 11:59pm – Gradescope entry code: `92J6Y3`

Please upload your answers timely to Gradescope. Start a new page for every problem. For the programming/simulation questions you can use any reasonable programming language (please no assembly, brainfuck, etc. ☺). Comment your source code and include the code and a brief overall explanation with your answers. A tentative point distribution (in % of the total) is provided in brackets. For most problems there is more than one valid way of solving them!

1. (25%) In this question, we will compute an expression for $H(t)$, the number of effective honest votes on the single public proposer block, which we discussed in lecture notes #6. We will use it to determine the confirmation latency of Prism in this case. Throughout, we assume the network delay $\Delta = 0$.

   a) Let m be the number of voter chains, and λ_h be the rate at which honest nodes mine blocks on *each* voter chain. Suppose the public proposer block b_p arrives at time 0. At time $t > 0$, compute the expected number of voter chains that have voted for b_p. Explain why this expectation is close to the actual (random) number of votes at time t if m is large.

   b) Compute the expected number of voter chains for which the vote is k-deep at time $t > 0$, in terms of $k \geq 1$. (As usual, $k = 1$ means the vote is from the tip of the voter chain, etc.)

   c) Using the earlier parts or otherwise, give an expression for $H(t)$, valid asymptotically for large m, in terms of the reversal probabilities q_k, for $k = 1, 2, ...$

   d) Using the Chernoff approximation for q_k,

   $$q_k \approx \left(\frac{4\lambda_h \lambda_a}{(\lambda_h + \lambda_a)^2} \right)^k,$$

   and assuming $\beta = 0.3$, $\lambda = \frac{1}{15} \frac{\text{block}}{\text{second}}$, find numerically t^* such that $H(t^*) = 0.5$.

2. (25%) In lecture #10, you have heard that the chain quality is lower bounded by $1 - 2\beta$, which means that even if an adversary controls β fraction of the hashing power then still (*at least*) an average of $1 - 2\beta$ fraction of the blocks in the final chain will have been mined by honest nodes. In this question, you will improve this lower bound to $(1 - 2\beta)/(1 - \beta)$ and show that this improved bound is tight. That is, that there is an

*Version: 2 – Last update: 29-May-2020

adversarial strategy with which an average of $(1-2\beta)/(1-\beta)$ fraction of the blocks in the final chain will have been mined by honest nodes.

Assume zero delay $\Delta = 0$, i.e., honest blocks will immediately be known to everyone.

a) Understand why the lower bound $1-2\beta$ we derived in the lecture cannot be tight and improve it to $(1-2\beta)/(1-\beta)$.

(**Hint:** Assume the chain growth rate is g and derive a lower bound in terms of g, and then optimize over g.)

b) It is typically assumed that when there are multiple longest chains (of equal length) the adversary can choose which longest chains honest miners adopt. Why is this a useful and reasonable modelling assumptions?

c) What chain quality is achieved if the adversary mines like an honest node?

d) What is the best attack you can come up with? Full credit will be given if your attack achieves chain quality $(1-2\beta)/(1-\beta)$.

(**Hint:** You have seen all the necessary ingredients for the attack already in this course. For inspiration, revisit problem 3b of homework #1, or check out [1].)

e) Using a mathematical model or a computer simulation, demonstrate the chain quality achieved under the attack you described in c).

f) What does it mean for

i. the mining rewards, and

ii. the transaction throughput,

if the chain quality is reduced to $(1-2\beta)/(1-\beta)$?

3. (25%) In Prism, there are three types of blocks: proposer, voter and transaction blocks. To focus on the impact of using transaction blocks, let us consider a simplified version of Prism, lets call it Prism-Lite. In Prism-Lite, there are only two types of blocks, transaction blocks and proposer blocks. There are no voter chains. The proposer blocks contain hashes of the transaction blocks like in Prism, but the leader proposer blocks are selected simply by the longest chain rule among the proposer blocks themselves, i.e., the proposer blocks on the longest chain in the proposer tree form the ordered sequence of hashes of the transaction blocks. Prism-Lite retains the high throughput of Prism but has Bitcoin confirmation latency (from using the longest chain protocol).

a) Suppose the communication bandwidth per node is 20 Mbits/second and each transaction block is 20 KBytes in size (like the blocks in Ethereum). On the average, how many transaction blocks is each node downloading each second if the system is operating at the physical limit? You can ignore the communication requirement of proposer blocks for now.

b) If one transaction is 500 Bytes in size, what is the total throughput of the system in terms of transactions per second? How does this compare to Ethereum, where the block mining rate is 1 block every 15 seconds?

c) If the proposer blocks in Prism-Lite are mined at one block per 15 seconds, what is the average number of transaction blocks each proposer block refers to? You can assume there is no forking in the proposer tree. If one thinks of the combination of the proposer block (as a header) with the transaction blocks it refers to (as the content) as a superblock, how big is this superblock?

d) Now suppose a fraction of adversary nodes with hashing power β fraction of the total hashing power is trying to do selfish mining on the proposer chain. Using your strategy in problem 2, the adversary is able to lower the chain quality of the honest nodes to $(1-2\beta)/(1-\beta)$ on the proposer chain. Moreover, the adversary's proposer blocks only refer to its own transaction blocks. (It's selfish, after all.)

   i. What fraction of the total *transaction blocks* belong to the adversary?

   ii. On the average, *at most* how many transaction blocks does each adversarial proposer block point to? How big are these adversarial superblocks?

   iii. On the average, *at least* how many transaction blocks does each honest proposer block point to? How big are these honest superblocks?

   iv. Is Prism-Lite fair in terms of the transaction throughputs of each party, despite the unfair chain quality? Discuss any differences with Bitcoin.

   v. Does the same conclusion apply to the full Prism?

4. (25%) We continue with the Prism-Lite system introduced in the previous question but explores the effect of applying sharding to such a system. The communication bandwidth per node, transaction block sizes, transaction sizes and the proposer blocks mining rate are the same as in problem 3.

   a) First let us do a more careful accounting of the total throughput of Prism-Lite by including the proposer blocks in the calculation. If each hash of a transaction block is 256 bits, what is the average size of a proposer block? What is the total throughput of the system in terms of transactions per second? What is the percentage loss due to the overhead in communicating the proposer blocks?

   b) Prism-Lite is now sharded into 10 different shards. Nodes in each shard download all proposer blocks but only transaction blocks in their respective shard. Each proposer block refers to transaction blocks of all shards, and contains the hashes of the transaction blocks it refers to. Each hash is 256 bits. What is the total throughput of the system, in terms of transactions per second? What is the percentage loss due to communicating the proposer blocks?

      Compare these two numbers to those of the unsharded system and discuss. What happens if the number of shards is increased to 100?

   c) The previous analysis ignores another overhead: the need to sample out-of-shard transaction blocks to check data availability. Let's divide each transaction block into 1000 chunks and code it using a rate 1/2 Reed-Solomon code.

i. What is the resulting size of a transaction block and how many chunks does it have?

ii. The data availability check should be correct with probability 0.999. How many samples of chunks does an out-of-shard node need to take per transaction block?

(Ignore the bandwidth requirement for communicating fraud proofs.)

iii. Compute now the total throughput of the system, in terms of transactions per second. Take everything into consideration. What is the percentage loss due to overhead in doing the data availability checks?

d) Please suggest how one can tune the various parameters of the protocol to reduce the proposer block overhead and the data availability check overhead.

References

[1] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. *CoRR (arXiv)*, abs/1311.0243, 2013.