COMS 6998-006 (Foundations of Blockchains): Homework #7

Due by 11:59 PM on Tuesday, November 30, 2021

Instructions:

(1) Solutions are to be completed and submitted in pairs.

(2) We are using Gradescope for homework submissions. See the course home page for instructions, the late day policy, and the School of Engineering honor code.

(3) Please type your solutions if possible and we encourage you to use the LaTeX template provided on the Courseworks page.

(4) Write convincingly but not excessively. (We reserve the right to deduct points for egregiously bad or excessive writing.)

(5) Except where otherwise noted, you may refer to your lecture notes and the specific supplementary readings listed on the course Web page *only*.

(6) You are not permitted to look up solutions to these problems on the Web. You should cite any outside sources that you used. All words should be your own. Submissions that violate these guidelines will (at best) be given zero credit, and may be treated as honor code violations.

(7) You can discuss the problems verbally at a high level with other pairs. And of course, you are encouraged to contact the course staff (via the discussion forum or office hours) for additional help.

(8) If you discuss solution approaches with anyone outside of your pair, you must list their names on the front page of your write-up.

(9) Some of these problems are difficult, so your group may not solve them all to completion. In this case, you can write up what you've got (subject to (4), above): partial proofs, lemmas, high-level ideas, counterexamples, and so on.

Problem 1

(35 points) This problem develops the basic intuition behind DAG-based protocols in an idealized, best-case environment. Consider the starting point of a proof-of-work longest-chain consensus protocol such as Bitcoin or Ethereum. Now consider setting the cryptopuzzle difficulty low enough that the rate of block creation becomes very fast (e.g., 10 blocks per second), and so there will be lots of forks. Assume also that we change the consensus protocol (relative to longest-chain) so that, when a miner mines a block, it can include any number of predecessor blocks that it knows about (as opposed to only one).

(a) (5 points) Suppose we make no assumptions other than that SHA-256 isn't broken. (I.e., all miners can be adversarial, there can be arbitrary propagation delays, etc.) Prove that the announced blocks and their predecessor pointers always form a directed acyclic graph (DAG).

(b) (7 points) In a DAG-based protocol, honest mining means including in your block every leaf block that you know about (i.e., every block that, as far as you know, hasn't been extended yet). Consider a simplified model in which all miners are honest and, other than ties in block creation, there are no propagation delays. Precisely:

- G_0 is the DAG with a single vertex (the genesis block).
- For $t = 1, 2, \ldots$, derive G_t from G_{t-1} as follows:
  - G_t has X_t more vertices than G_{t-1}, where X_t is a positive random variable. (The distribution won't matter for us, it could be e.g. uniform over $\{1, 2, \ldots, 10\}$.) Let V_t denote these new vertices.
  - For each new vertex $v \in V_t$, there is an edge directed from v back to each vertex that is a leaf in G_{t-1}.

An *ordering* σ of a DAG G is a way to sequence its vertices $v_1, v_2, \ldots, v_n$ so that all its edges are directed forward (i.e., every edge (v_i, v_j) satisfies $i < j$). Ordering a DAG imposes an ordering on the transactions contained in its blocks (which is needed to resolve conflicting transactions). We are interested in orderings $\sigma(G)$ computable by a casual observer, meaning that the ordering should depend only on: (i) the graph G; and possibly (ii) the vertex IDs (e.g., the SHA-256 hash of the block contents). In particular, σ should not depend directly on the time steps at which various vertices were created.

For the simplified model above, propose an ordering $\sigma(G_t)$ of the blocks of each graph G_t such that $\sigma(G_t)$ extends $\sigma(G_{t-1})$ for each t (i.e., $\sigma(G_{t-1})$ is a prefix of $\sigma(G_t)$). Prove that your orderings have the desired property.

(c) (8 points) Now suppose we relax the model to reflect the fact that a miner may not be aware of all the latest blocks: at each time step t, each new vertex $v \in V_t$ has an edge to at least one (but not necessarily all) of the leaves in G_{t-1}.

Is there a way to define an ordering $\sigma(G_t)$ of the blocks of each graph G_t that could arise in this relaxed model satisfying the property in (b)? Provide either a proof that there is, or a proof that there isn't.

(d) (9 points) Consider a different (weaker) consistency property that we might want: whenever two blocks appear in both G_s and G_t, their relative order in $\sigma(G_s)$ and $\sigma(G_t)$ should be the same. Is there a way to define an ordering $\sigma(G_t)$ of the blocks of each graph G_t that could arise in the relaxed model in (c) that satisfies this weaker consistency property? Provide either a proof that there is, or a proof that there isn't.

(e) (6 points) Is the weaker consistency property in (d) sufficient for a trustworthy blockchain, or at least one with restricted functionality? (Assume everyone is honest, with forks occurring only inadvertently as in the model in (c).) Explain your answer (and feel free to argue both sides).

Problem 2

(45 points) In this problem we'll look at scaling up Nakamoto (i.e., Bitcoin) consensus in a different way, using m parallel blockchains, each initialized with its own genesis block. (If convenient, you can assume that m is a power of 2, e.g. 256 or 1024. All miners know all genesis blocks in advance.) The rate of block creation within a single blockchain will be as in Bitcoin now (i.e., on average one every ten minutes), so overall there will be a factor-m increase in throughput (i.e., the cryptopuzzle difficulty will be set so that there are m new blocks per 10 minutes, on average).

(a) (5 points) Suppose each of the m blockchains runs Nakamoto consensus in parallel (with no interaction between them). Thus, the authorized transactions in the blockchain B_i are those on the longest chain in B_i. The ordering among authorized transactions within one blockchain is clear; we'll worry about the overall transaction ordering (across blockchains) in parts (c)–(e) below.

Suppose a miner chooses which block to extend (from any of the blockchains) as in Bitcoin, by including the name (hash) of its (unique) predecessor. In this case, explain why it is a bad idea to use m parallel blockchains with m large.

[Hint: it has to do with security. I.e., explain what an attacker could do.]

(b) (10 points) Intuitively, we'd like to force miners to mine simultaneously across all m blockchains, rather than choosing their favorite a priori. Propose modifications to (i) the block metadata; (ii) the interpretation of the block hash (under SHA-256); and/or (iii) the definition of block validity such that (under the usual random oracle assumption for SHA-256):

(i) whenever an honest miner successfully mines a valid block, it is equally likely to belong to any of the m blockchains;

(ii) no matter what an attacker does, for each of the m blockchains B_i, she can only add new blocks to B_i at the usual rate (on average α blocks per 10 minutes, where α is her fraction of the overall hashrate).

Prove that your construction satisfies the desired properties.

[Hints: The amount of new block metadata can scale linearly with m. Use some leftover bits from the block hash.]

(c) (10 points) Using the construction in (b), prove that the liveness property for Nakamoto consensus holds (with high probability) for each of the m blockchains. (I.e., Theorem 3 from Lecture 8; see also Homework #3, Problem 4.)[1]

[Hints: Do not prove from scratch. Given a hypothetical attacker who can break this property, construct another attacker who could break the original liveness property for the standard longest-consensus protocol (Theorem 3 from Lecture 8). Your proof does not have to be 100% rigorous, but it should be extremely convincing.]

(d) (10 points) In this and the next part, assume that all miners are honest and that everybody learns instantly about every new block. (Hence, there is never a fork, and each of the m blockchains is just a chain.) Next we explore how to impose a total ordering across the blocks in all of the blockchains.

Let M denote the union of the m parallel blockchains. An *ordering* $\sigma(M)$ imposes a total ordering on the blocks of M, and can depend only on the metadata in the blocks and on the sequence of blocks in each blockchain. (The ordering should of course respect the orderings within each of the blockchains.) For a sequence $M_0, M_1, \ldots$, call a block b *stable at time* t if the order of the blocks up to and including b is identical in $\sigma(M_t), \sigma(M_{t+1}), \ldots$ (i.e., the linear order is fixed forevermore through block b). The *stabilizing time* of a block b is the difference between the first time at which b is stable and the time at which b was created.

Our first definition of an ordering σ is: sequence blocks in order of height (i.e., how deep they are in their own blockchain), breaking ties by the chain number. I.e., first are the m genesis blocks (starting with that of B_1 and ending with than of B_m), then the blocks whose immediate predecessor is a genesis block, and so on.

Prove that, with this ordering, the expected stabilizing time of a block created at time t cannot in general be bounded by any function that is independent of t.

[Hint: use that the standard deviation of a binomial random variable with n trials scales with $\sqrt{n}$.]

(e) (10 points) Suppose we include in the block metadata two new fields, *rank* and *s-rank*, which will be defined inductively (the "s" stands for "successor"). Every genesis block has rank 0 and s-rank 1. When a new block b_1 is created, extending the block b_0, b_1's rank is set to b_0's s-rank. Meanwhile, b_1's s-rank it set either to its rank plus 1, or to the maximum s-rank of any other block (whichever is larger).

Now suppose we sequence blocks in order of *rank*, breaking ties by the chain number. Prove that, for all time steps t, the expected stabilizing time of the block created at time t can be bounded above by a function that is independent of t.

[Hint: this is related to the coupon collector problem.]

[1] The model is the same as in Lecture 10. Winning hashes are found one by one. The next winning hash has an α chance to be found by the attacker Alice, and a $1 - \alpha$ chance of being found by an honest miner. You've defined what honest miners do as part of your solution to (b). Alice can try to extend any blocks she wants, and can withhold winning hashes until she sees fit.