The Graph Protocol — Initial Review

May 2021
Nicholas Ward, Sergii Kravchenko, Heiko Fisch

General Information

These are the notes from our initial review of **The Graph Protocol**. The goal of this review was, first and foremost, to get familiar with the system; an official report was not required or delivered. Any potential issues or points for discussion we found during our review are listed below and have been discussed with the Graph team.

- Client: The Graph. PoC: Ariel Barmat, David Kajpust
- Repository: https://github.com/graphprotocol/contracts/tree/42a6f882fc289bc418f6ee6c457c277845ccc3e6

Review Notes

General Observations

- Very good documentation at a good mix of abstraction levels
- High-quality, well commented code
- Fairly big and complex system both on a technical and a conceptual level
- Allotted time was sufficient to gain a basic understanding of the system and some familiarity with the codebase

System-Level Comments

Disputes

- If Alice wants to create a dispute (query or indexing), what keeps Bob from front-running her tx? In the case of query disputes, they would both successfully create a dispute, and the arbitrator would have to decide which of the two is the “right” fisherman. In the case of indexing disputes, only Bob’s dispute creation would succeed (assuming his front-running is successful).
  We don’t know how exactly the arbitration process works, but for query disputes it seems possible that there is some off-chain communication between the fisherman and the arbitrator involved (for example, the fisherman have to provide the full request and the full response, i.e., preimages for `requestCID` and `responseCID`, assuming these are hashes); in this case, Bob’s attempt at front-running would be futile if he doesn’t have access to this information and can’t guess it. Effectively, this would work like a commit-reveal scheme, where the revelation happens off-chain and helps the arbitrator decide which dispute is legitimate.
  However, for indexing disputes, it looks like no additional information (that has to be shared off-chain) is necessary, so Bob could just let others find the cheating indexers and then take the reward for himself…

Staking

- Partial undelegations/unstaking are not working as we would normally expect. When you undelegate a part of the tokens, they are locked for some period. Then if you undelegate some more tokens before that period has passed, the new locking period is applied to all the locked tokens.
- When undelegate is requested, the tokens are still locked but not receiving rewards anymore.
- `_setDelegationParameters` can only be done after at least `cooldownBlocks` number of blocks after the previous update. We think it makes more sense to have this restriction in a `timelock` manner, when the changes are announced beforehand, and delegators can decide to undelegate beforehand. If the idea behind that logic is not to make any surprising changes to the delegators, that seems like a better solution.

Miscellaneous

- The `DisputeManager` and the `GraphToken` contracts employ EIP-712-style signatures. At deployment time, the chain ID is retrieved and “baked into” the `DOMAIN_SEPARATOR`.
  However, the chain ID is not necessarily constant over the lifetime of a deployed contract. In the event of a chain split, only one of the resulting chains gets to keep the original chain ID and the other will have to use a new one. With the current pattern, a signature will then be valid on both chains — which is probably not the intended behavior.