Monadring: A lightweight consensus algorithm for organizing subnets upon a blockchain system

Zhang Yu, Gang Tang

June 15, 2024

Abstract

Existing blockchain networks are often large-scale, requiring transactions to be synchronized across the entire network to reach consensus, and on-chain computations can be prohibitively expensive, making many CPU-sensitive computations infeasible. Inspired by the structure of IBM's token ring networks, we propose a lightweight consensus algorithm called Monadring to address these issues. Monadring allows nodes that are already part of a large blockchain network to form a smaller subnetwork to perform computations more quickly and cheaply, while still maintaining the same security guarantees as the main blockchain network. The paper details the design and implementation of the Monadring algorithm, and evaluates its performance and feasibility through simulation experiments. This research contributes to enhancing the practical utility of blockchain technology in large-scale application scenarios.

1 Introduction

Recent blockchain systems have often adopted faster and more energy-efficient consensus protocols like Ouroboros, BABE, and Tendermint. Blockchain networks are typically open and permissionless, and we can define the degree of decentralization of such networks by the number of participating nodes. Leveraging the consensus algorithms mentioned earlier, it is possible to rapidly construct new blockchain networks. However, decentralized applications often cannot flexibly customize their degree of decentralization based on the importance of their underlying data.

For example, decentralized social media applications may require faster response times and lower storage costs compared to decentralized finance applications, and hence may need a lower degree of decentralization. This flexibility to adjust the level of decentralization based on the needs of the application is an important consideration that has not been fully addressed in existing blockchain architectures.

Building a subnetwork over an existing blockchain system is a possible way to address this issue. By drawing inspiration from the token ring architecture, we propose a novel consensus algorithm called Monadring that aims to enable nodes within an existing blockchain network to form smaller, lightweight subnetworks capable of performing computations more efficiently and cost-effectively, while still maintaining the same security guarantees as the main blockchain.

Token ring network operates at the data link layer of the OSI/RM model. It was designed to solve the problem of physical link contention just like Ethernet. In a token ring network, a token is passed sequentially from one node to the next, granting the holder the right to transmit data. This token-based system for managing access to the shared medium bears some similarity to the consensus mechanisms used in blockchain networks for selecting block producer.

Despite the fact that token ring networks have largely fallen out of favor in modern networking due to their limited scalability and other drawbacks, the underlying principles of their decentralized, token-based structure could provide valuable insights for designing a lightweight consensus algorithm for small blockchain systems.

2 Definitions and Model

Hostnet and subnet. Consider a network composed of a set of participants V in which the majority obey a protocol to reach Byzantine Agreement and finality[?] over ledger L. S_i is a subset of V, $V = S_0 \cup S_1 \cup .. \cup S_{n-1}$. We call V is a *hostnet* and S_i is a *subnet* of V.

- A node participant $v \in V$ could be a member of any S_i at meantime.$v \in V, v \in S_m \cap S_n$ is valid.
- We assume the ledger L of hostnet maintains all the subnets information in form of a mapping *subnet id→node list.*
- Any participant of the hostnet could join a specific subnet through proposing a modification over ledger L.
- Since the ledger L is under Byzantine Agreement by all participants, the map can be considered a provable information outside any subnets.

Subnet ledger. For each subnet S_i, all participants $v_j \in S_i$ maintain an independent ledger L_i different from the ledger L of hostnet, while the root state of each subnet ledger will be recorded in the hostnet ledger, $L_i \nsubseteq L, \digamma(L_i) \in L$.

We want to formalise the procedure of reaching consensus on the ledger across all participants as a protocol that can be deployed along with any kind of blockchain network to build subnet. We can assume that the subnet ledger has properties as below:

- The subnet ledger contains all the modification events and each event has an increasemental number as index. The ledger is expected to be in a deterministic state after the n_{th} event being applied, $S_{n+1} = f(S_n, e_n)$.
- Particularly, changing the function f is also a kind of event. The first event e_0 is loading the function.
- The time complexity of looking up the n_{th} event is $O(1)$.
- The time complexity of retrieving the maximum event id is $O(1)$.

Subnet topology and token. The participants of the subnet strictly follows the sequence of the *node list* to form a ring topology. A *token* T of a subnet with n nodes is a special signal circulates around the ring following the sequence of the *node list*.

- The token carries groups of events from its sender and the sender's forehead recursively. Group G_i is composed by the node v_i. It contains a list of modification events originally from its pending request queue, the node's digital signature, a digest of its local ledger after these modification applied, a number q indicates how many times this group should delivered and the nonce of the signer:

  $G_i = (E = [e_k, e_{k+1}..e_{k+n}], S_{k+n}, signature(nonce, E, S_{k+n}), nonce, q)$

  The event list E could be empty.
- Normally, a token circulates in a subnet with n nodes should always include n groups unless there were malicious behaviors or some nodes went offline. Whenever a node receives the token, it ought to check the signature and q for each group. Then applying all the events of each group and compare the digest with the local ledger. The q of executed groups should be decreased by 1.

- If all checks pass, the node should handle transactions from its local queue and compose them as a new group with initial $q = n - 1$ to replace previous one in the token. Then try to deliver the token to its successor.

Malicious behaviours and offline. TODO

Blind challenge. TODO

3 The Monadring Protocol