Design & Implementation Document

EventTokenizer VM for Avalanche

version : initial draft (06 Nov'2021)

1. Introduction

This document describes the design and implementation done by our team, as part of the Hackathon event hosted by Gitcoin and sponsored by Ava-Labs titled : "Subnet and Virtual Machine". Post the hackathon some parts of the architecture was fine tuned and some added (Indexer), and this document gives the updated design and implementation details for the EventTokenizerVM. For the design deatils of the reward system, refer to reward-design-document.pdf in the github repo.

2. Challenge Description

Avalanche subnets allow anyone anywhere to spin up a taylor-made network w/ custom virtual machines and complex validator rulesets. They enable permissionless and permissioned networks to launch with optional privacy and regulatory compliance.

Virtual machines are the application-level logic of your blockchain. ...

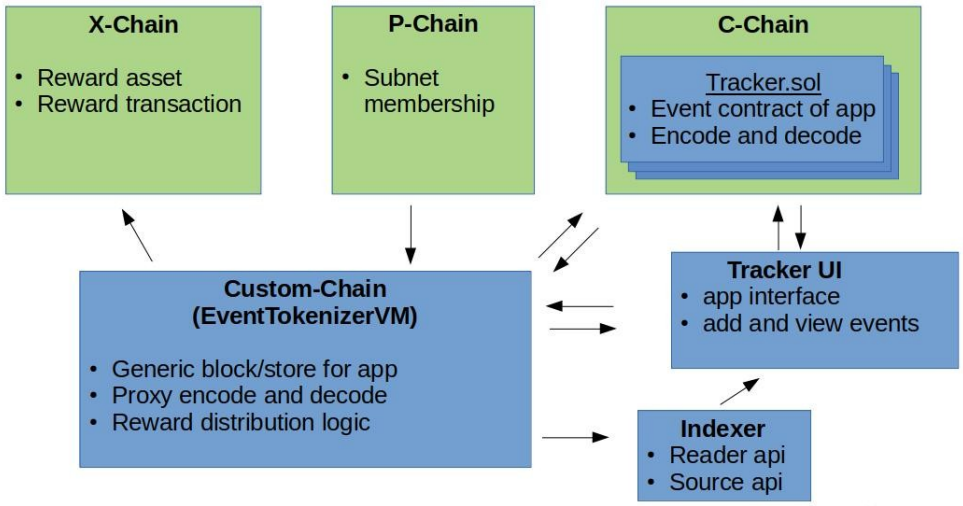
3. Our understanding of the Challenge Description

Currently on Avalanche, the validators have the ability to add Nodes to the Primary subnet and are rewarded as per the Avalanche staking reward system. Validators can also add their nodes to other subnets, however we donot know of any mechanism of getting rewards if they validate additional subnets other than the Primary subnet (viz., P, X & C). There is also a requirement to develop a real application use case which can be hosted as a custom VM and blockchain.

4. Our Proposed Design

We have based our work on the template provided by Avalanche - timetampvm and extended it for defining the reward system as well as the EventTokenizer VM.

The high level block diagram of the EventTokenizerVM architecture and components is as below.



The components viz., Custom-Chain, Tracker Contract on C-Chain, TrackerUI and Indexer were developed as part of the EventTokenizerVM.

Details are given below.

Custom-Chain (EventTokenizerVM) : This is a plugin which has the logic to store the generic blocks of the application. The reward distribution logic is defined in a separate file rewardmgr.go

Tracker.sol Contract : This is the Contract code written in solidity and is deployed on the Avalanche-C chain. It has the encode and decode functions for the data stored on the custom blockchain.

Tracker-UI : This is the dApp developed as demo for the Shipment Tracking use case. It interacts with the Custom-VM and the Indexer componets. The dApp is developed on Next.js framework, and uses Metamask for basic identity and signing during submission of events to the blockchain.

Event-Indexer : This is a bare bones implementation of an Indexer using Nodejs and Leveldb. When the block is written to the custom blockchain, the node will also source the basic blockid to the EventIndexer to index into the database. The Indexer exposes some api's for the Tracker-UI, for getting recent shipments added to the blockchain, and also for getting the sequence of events related to any specific shipmentID.

5. Customization to other Use cases

The EventTokenizerVM can be used for multiple use cases, by having a generic data block structure. The fields are given below

{
- timestamp : time when the block was proposed to write on the blockchain
- eventID : unique id of the block
- parentID : parent block's eventID (used by blockchain infra)
- sequenceID : used as ID to group events specific to a tracking id
- codecAddress : c-chain contract address hosting the encode and decode function
- encodedData : encoded data specific to the application
- refID : future use
- refTime : future use
- creatorAddress: address which identifies the person who added the event/shipment

}

Here the eventID is unique for each block that is written. The sequenceID is the one which is used to link corelated events. For the shipment tracking use case, whenever a new shipment is added, this first eventID is used as the sequenceID for any subsequent related events.

The encodedData structure is defined in the Tracker.sol and for the Shipment tracking use case we have

{
- status : init, in-progress, delivered, etc., based on shipping status
- location : location of the shipment while the status is added

}

The above component design is also generic enough to cater to various other use cases apart from Shipment tracking. The changes required to customize is minimal and restricted to few components. The data structure that is written into the block is generic. So ideally the only change should be in the decode and encode funtions in the Solidity Tracker.sol contract.

Few other example use cases are given below.

A. Helpdesk Ticketing

In the blockchain, the eventID can be the Helpdesk Ticket number, and the sequenceID will be used to track status of subsequent helpdesk events.
The encodedData structure to be changed in Tracker.sol can be

```
{
    status      : opened, assigned, updated, resolved
    owner       : person who is currently incharge of this ticket
}
```

B. Student progress in e-learning

In the blockchain, the eventID can be the enrollmentID for a particular course by a student, and the sequenceID to be used will be this initial eventID in subsequent progress events.
The encodedData structure to be changed in Tracker.sol can be

```
{
    status      : enrolled, in-progress, finished
    module      : week1, week2, assignment1, project
}
```

6. Other Design Considerations

The above design is developed for the submission of the hackathon event. However there are other alternatives that can be thought of or developed based on various pros and cons. We are listing some of them.

Connection to Wallet : For the purpose of identity we are using wallet (Metamask). Once a user connects to the wallet, the dApp has infomation about the C-Chain Address, which is used as identity while writing into event block. The wallet is also used for securely signing the event information, so that no one can maliciously fake entries in the blockchian. For every entry in the blockchain, users has to sign using Metamask. This can become annoying in case a lot of entries are to be made or during production. The alternative is to implement an authentication mechanism so that users login into the dApp initially, and further use the same identity for all new events created. The developer has to take care of storing the credentials, wallet address, etc., securely in the client App.

VM and Blockchain : The same blockchain can host multiple applications, since the block structure is generic. The encoded event data will be different for different applications. Alternately we can have independant custom blockchain for each and every application based on the customer’s needs, scale, etc.,

Indexer sourcing : Currently the indexer is configured to be sourcing the data from a single node as source of truth. We can have an alternate design where the indexer sources from multiple nodes and the aggregate truth value is used while indexing any event.

7. User Workflow

Tracker-UI : This is the dApp for the Shipment tracking use case. On visiting the home page, user can get the overall statistics about the events and shipments. Also a listing of recent 10 shipments is given. Further clicking on any of the trackID, users can see various sequence of events related to a particular trackID.

If a user wants to add any event to existing trackID or add new shipment tracking, they have to connect to wallet (Metamask). The wallet is only used to get the identity of the user. Also there is an option for submission with and without signing. This option is given for the demo purpose to showcase the feature of securely signing the data submitted.

Subnet Validation only : Both the TrackerUI and Indexer are optional components in the architecture. If anyone just wants to add a node as subnet validator for rewards, they can copy the plugin into their node and add their node to the custom subnet by following the standard avalanche documentation.

8. References

a. Hackathon event link : https://gitcoin.co/issue/ava-labs/open-defi-hackathon/3/100026354
b. Avalanche Developer Documentation : https://docs.avax.network/
c. Github link of our hackathon submission : https://github.com/shri4net/ava-labs-reward-system

9. Further Contacts

1. ksk12345@yahoo.com
2. shri4net@gmail.com