A formal definition and statistical model checking of the constellation blockchain

March 6, 2018

Abstract

Introduction

Fundamental Data Structures, Types and Functions

Tier <: Numeric

Numeric type delineating tier within multi-tier hierarchy

BlockData[Tier] <: U

Consider a type $BlockData$ which serves as the parent type (under universal type U) of all data going on chain. All transactions, validator requests, meme data etc. All BlockData is equivalent via covariance and thus all BlockData can be compressed into a Block.

Block[U] <: U

Contains compressed form of Block Data. This is the result of consensus. Each unit of $BlockData$ must reference the previous round of consensus, this is satisfied by containing the $Block$ hash. It is worth noting that $Block[U] \equiv Block[BlockData[Tier]]$ through covariance. Block[U] is a Fix Point Type [1]

Transaction <: BlockData[0]

Transaction is a subtype of BlockData that can only exist on the bottom tier. It is the building block of our currency.

[1]https://jto.github.io/articles/typelevel-fix/

LeftHand <: RightHand <: Transaction

Note: LeftHand and RightHand are symmetric subtypes of Transaction. For a Transaction to be valid it must be 'signed' by the counterparty. This is satisfied by sending a RightHand transaction that references a LeftHand transaction.

Definition: Consensus Simplex, σ

A set of validators, undergoing cryptographic consensus to produce a *Block*. Formally defined as a Simplex.

$$\sigma := collection[node] \tag{1}$$

Definition: Validator n

A node that has 'woken up' by starting a *consensus* process. It now gossips transactions to it's neighbors and is waiting to be selected as a Delegate.

$$n \in \sigma \tag{2}$$

Definition: Delegate

A node who has been selected to perform consensus. Delegates are implicitly chosen, locally on each node via the Generating Function (see below). Once a new *Block* is received, it is passed to the Generator Function, which tells this node if it is a delegate. A delegate is a node $n \in d$ such that

$$d \subset \sigma \mid g(Block, d) \to d \tag{3}$$

where g is the generating function below.

Generating Function g

This function is used to determine the next set of delegates from a given set of validators (consensus cluster) by selectively sampling a subset of nodes, based on reputation score and a probability distribution. See GURU for examples selective sampling via probability distribution.

$$g : (Block, \sigma) \to \phi \mid \phi \subseteq \sigma \tag{4}$$

Consensus Function $c : \sigma_k \to Block_k$

A function that maps a σ_k to $Block_k$. It follows that this is isomorphic to a catamorphism, with the collection being a cluster's mempool and the result the reduce being a Block.

Node ADT

let n be an abstract data type Node, with three attributes, a *protocol* process a *consensus* process and a *chain* class. The *protocol* process handles transaction signing and essentially all functionality to send and receive payments. The *consensus* process which implements the responsibilities of a validator node, is optional. Nodes can be 'sleepy', turning on the *consensus* process at will. The *chain* class is a local blockchain made up of all transactions/interactions with the chain that this node has made. Each link in the chain is a sub type of BlockData.

Formally we define n as the algebraic data type

```
case object Node {
  val chain: Chain = new Chain()
  val protocol: => Seq[BlockData[T]] = new Protocol()
  val consensus: => Block[U] = new Consensus()
}
```

Network Topology

We define the primitives of our network topology. We come to the conclusion that our network topology can be formally defined as a simplicial complex, specifically and ordered set of a special simplex called a Radial, which we define below. We define constellation's topology as an ordered set R_K of Radials R_t, following ordering defined by Tier $t \in T$

$$R_K : \{R_0 \dots R_T\} \tag{5}$$

Simplex

The clique complex of a graph G is a simplicial complex whose simplices are the cliques of G[2]. We define a simplex σ as a completely connected graph (clique) of constellation nodes, with which we can perform cryptographic consensus. Given $\sigma = \{n_0 \dots n_i\}$, σ is a simplex iff:

1: σ is a complete graph such that for a set of edges e, corresponding to nodes n,

$$\forall n \in \sigma, \exists e \in E \mid e \equiv \sigma \setminus \{n\} \tag{6}$$

Corollary: for each n in a a simplex σ_K, the simplicial star[3] of n, $st_K(n)$, for simplex K is equivalent to the σ_K that is:

$$\forall n \in \sigma_k, st_K(n) \equiv \sigma_K \tag{7}$$

[2]https://arxiv.org/pdf/1007.0418.pdf

[3]https://en.wikipedia.org/wiki/Simplicial$_complexClosure,_s tar,_a nd_l ink$

2: There exists a deterministic mapping given by an immutable generating function g

$$\exists g : \sigma \to d, \ \mid d \subseteq \sigma \tag{8}$$

3: There exists the notion of a star cluster[4] Such that star cluster σ forms an independence complex of K.

$$SC(\sigma_k) = \bigcup_k st(n) \in I_G \ \forall k \in K \tag{9}$$

Where I_G is the set of all independence graphs of K[5]. It follows from corollary (7) that

$$\begin{aligned} SC(\sigma_k) &= \bigcup_k st(n) \\ &\equiv st(n) \ \forall n \in k \end{aligned} \tag{10}$$

Radials: Tiers of Hypergraphs

A hypergraph represents an arbitrary set of subsets of it a graph's vertex set, where each subset is called a hyperedge[6]. We define the the Radial abstract data type in terms of a hypergraph and mappings between vertex sets within that hypergraph. Specifically we define a Radial ADT R_t for given Tier $t \in \mathbb{N}$ (see above) such that

```
case class Sigma(nodes: Node*)

type Radial[T <: Tier] {
/*
simplex and starCluster are hyperedges
*/
   val starCluster: Sigma[T-1] \\ we need a stricter definition here
   val simplices: Sigma[T]*
   def hyperPlane[T1, T2] = Sigma[T1] => Sigma[T2]
   def route[T]: hyperPlane[T, T-1]
}
```

The hypergraph for all simplices σ_k within Tier t is given by the two element set

$$H_{t,k} = \{I_{g_{t-1}} \mid g_{t-1} \subset G_{t-1}, \sigma_k\} \tag{11}$$

where $I_{g_{t-1}}$ is a subset of all simplexes in tier $t-1$ (as we know all simplexes are independent) and $\sigma_k \in R_t.simplices$. $hyperPlane$ is a function that maps

[4]https://arxiv.org/pdf/1007.0418.pdf

[5]Definition 2.1 https://arxiv.org/pdf/1007.0418.pdf

[6]Pal S. et. al. http://www.facweb.iitkgp.ernet.in/ spp/geomgraph.pdf

between two hyperedges, potentially across tiers. *route* is a function that connects our two hyperedges with a *hyperPlane* between Tier t injectively to an independence complex in $t-1$. Note that $Sigma[T-1]$ and $I_{g_{t-1}}$ are equivalent via homotopy as shown in (5).

In the degenerate case of our definition of R_T, namely when $R_t = \{R_0\}$, *starCluster* is a mempool as defined below, but of transactions which are isomorphic to all subtypes of BlockData via covariance.

Chain Topology

We define the primitives of our DAG chain. We come to the conclusion that our DAG chain is a directed acyclic graph who's direction is given by simplex graph[7] κ and topological ordering follows the tiered ordering of R_K. The simplex graph $\kappa(G)$ of an undirected graph G is itself a graph, with one node for each clique (a set of mutually adjacent vertices) in G. As all of our simplexes are independent, $\kappa_j \mid j \subseteq t-1$ for tier t is formed as a disjoint subset of simplexes in $t-1$. We define κ as a mapping from an undirected graph G to a new graph $\kappa(G)$ who's vertices are cliques and compositions of cliques of G

$$\kappa : \sigma_j \to \sigma_k \mid j \subset t-1, \sigma_k \in R_t.simplices \tag{12}$$

Remark: there is notable symmetry in our definition of κ and Radial.*route* above. We will see a duality formed from this connection.

Chain Fibering

We formulate notion of chain fibers by defining consensus in terms of a simplex graph κ. A chain fiber is given by

$$f \circ \kappa_j^k : \sigma_k \to Block_k \tag{13}$$

where κ_j^k is a simplex graph made up of chain fibers $Block_j$ in a preceding tier. The chain fibers from the preceding tier, $Block_j$, become the domain of the consensus function for tier k. We can say equivalently

$$c_k : \{Block \ldots Block\}_j \to Block_k \tag{14}$$

We can formally define the mempool *mem* of simplex σ_k in tier t as

$$mem_k \equiv \{Block \ldots Block\}_j \mid j \subset R_t.simplices \tag{15}$$

[7]https://en.wikipedia.org/wiki/Simplex$_{graph}$

Consensus, hyperplanes and delegate selection

Delegate selection is performed by the Generating function, a deterministic function and Consensus, a probabalistic function, both of which are defined by the hyperplane function defined on radials. Specifically we use the framework above to formally define the Generating function and Consensus

The Generating function is an instance of Radial.$hyperPlane$ where $T1 = T2 = T$. Formally

$$\begin{aligned} g : (Block, \sigma) \to \phi \mid \phi \subseteq \sigma \\ g \cong Radial.hyperPlane[T, T] \end{aligned} \tag{16}$$

g is implemented isomorphic to

$$\begin{aligned} g \cong Radial.simplices \\ .fold(\sigma => z.update(\sigma))(z : Histogram) \\ .map(\sigma* => dist(\sigma*)) \\ .flatten \end{aligned} \tag{17}$$

where $dist$ is the generating function of a probability distribution given by precompiled bytecode and notarized by the TGE Block.

Consensus is an instance of Radial.$hyperPlane[T-1, T]$. Formally

$$\begin{aligned} c_k : \{Block \dots Block\}_j \to Block_k \\ \cong Radial.simplices \\ .filter(\sigma_k => Radial.route[t, j](\sigma_k)) \end{aligned} \tag{18}$$

where $Radial.route[t, j]$ is isomorphically defined as

$$Radial.route[t, j](\sigma_k) \cong \exists \kappa_j^k \tag{19}$$

Sound Bytes

Consensus

Consensus is the process of forming a $Block$, it can be thought of as a function $f : \sigma \to Block$. Consensus clusters are formed in a tiered hierarchy. The top most tier forms the global state of the chain, which is made up of $Block$s from preceding tiers. Each tier, from top down until the second to last, has responsibility for routing transactions and validating the blocks from consensus clusters.

What is the lifecycle of a transaction

When a transaction is sent, it is sent from a node to a higher tier which 'routes' the transaction to the consensus cluster(s) that host its 'shard', or the shard of the blockchain that host's it's public key's history. Each transaction has a left and right half. The initiator of the transaction sends the $LeftHalf$ to the network. Its $LeftHalf$ is then referenced by the $RightHalf$ which is sent by the counterparty.

Why do we double sign?

It preserves the notion of ordering. Scenario: I have 5 dollars, I send to two people. I send 5 to Wyatt and then 5 to Preston. I will need to wait until the top tier finishes consensus because both transactions are effectively 'racing' each other. Double signing allows us to preserve ordering (with high probability) without waiting for the total network to update state. Without this, consumer facing point of sale systems (think grocery store) are not possible; we would need to wait for a global state to reach consensus. Double siginig gives users the illusion of instant transaction confirmation.

How is consensus performed

Consensus $f : \sigma \rightarrow Block$ is the act of creating notarized data in the form of $Blocks$. In our case, we are using the HoneyBadgerBFT, which prevents against sybil attacks using encryption [8]. Nodes are rewarded based upon successful completion of consensus, the number of transactions they provide, and metadata about their performance. This is all calculated post facto by proof of meme and rewards are given within a set interval of blocks.

How are delegates selected

Delegates are selected locally, by passing the previous block and current set of validators into our Generating Function. This happens within the Consensus FSM.

How is this secure/fit in with our incentive model

Double spends across asynchronous consensus is prevented by double signing transactions. Consensus clusters are rewarded for processing transactions and sybil attacks are mitigated via encryption in honeybadgerBFT. We also are able theoretically improve typical byzantine fault tolerance over 40% thanks to GURU and our reputation model. Ddos attacks can be mitigated via throttling of accounts with low reputation scores. I propose an incentive for routing where each node that routes a transaction signs the tx, and when a tx is notarized each account that routed the tx is given a reputation increase.

[8] ch. 4.3 https://eprint.iacr.org/2016/199.pdf