A formal definition and statistical model checking of the constellation blockchain

March 5, 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 := $collection[node]$

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

Definition: Validator

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.

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.

Generating Function $g : Block \to collection[node, reputationscore]$

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.

Consensus (function) $f : ConsensusSimplex \to Block$

A function that maps a consensus cluster to a Block. 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

```
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 of Radials, which we define below.

Consensus Simplexes

We define a Consensus Simplexes S as a clique within a Clique Complex. As a Clique Complex can be formulated by a Simplex[2], we adopt this formulation and define a Consensus Simplex as follows: given a simplex $S = \{n_0 \dots n_i\}$ of constellation nodes n, S is a simplex Consensus Simplex iff:

1: S is a complete graph such that

$$\forall n \in S, \exists e \in E : S \setminus \{n\}$$

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

$$\exists g : e \to d, \text{ where } d \subset S$$

3: There exists the notion of a star cluster[3]

Such that star clusters across across Consensus Simplexes are homotopy equivalent and given by an independence complex.

(by Lemma 3.2 of (4) we know that the notion of contractibility ensures that an independence complex is shares homotopy with subsets of the same clique complex)

Corollary:

Star(n) for each n in a Consensus Simplex is equivalent to the Consensus Simplex.

[2]

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

Radials: Tiers of Hypergraphs

We define the the Radial abstract data type in terms of an hypergraph and mappings between vertex sets within that hypergraph. A hypergraph represents an arbitrary set of subsets of it a graph's vertex set, where each subset is called a hyperedge[4]. Specifically we define a Radial R_j for given tier $j \in \mathbb{N}$ such that

```
Type Radial[Tier[ {
/*
simplex and starCluster are hyperedges
*/
val starCluster: independenceComplex[Tier-1]
val simplex: S[Tier]
val route[Tier]: (independenceComplex[Tier-1]) => S[Tier]
val Hyperplane = route[Tier](starCluster) => simplex
}
```

$$R_j = \{SC(S_{j-1}), S_j \equiv \{n_{j,0} \ldots, n_{j,i}\}\}$$

where $SC(S_{j-1})$ is an independence complex (star complex) of simplex tier $j-1$, and *route* is a function that maps an independence complex from tier $j-1$ to the current tier's consensus simplex. As a radial is isomorphic to a Consensus Simplex (can be shown by homotopy), our network topology is Consensus Simplex, specifically a clique complex, of Radials.

Chain Topology

We define the primitives of our DAG chain. We come to the conclusion that our network topology can be formally defined as a simplex graph with formal definition given by [5].

DAG of Chain Fibers

The constellation blockchain is a DAG, where each node in the DAG is a simplex graph, composed of all nodes in the topological order preceding the current node. The nodes correspond to an undirected graph of chain fibers who will form a independence complex (mempool) upon which consensus can be performed.

Chain Fibering

We define the notion of chain fibers using a simplex graph. A chain fiber $C_k = \{\kappa(S) \ldots \kappa(S_i)\}$ is a simplex graph made up of chain fibers in a preceding tier.

[4] Pal S. et. al. http://www.facweb.iitkgp.ernet.in/ spp/geomgraph.pdf
[5] def 3.1https://arxiv.org/pdf/1007.0418.pdf

Mempool of independent simplexes

The Mempool of a consensus cluster is given by an independent set of all child simplexes routed by the

Consensus and Underlying Protocol

Consensus

Consensus is the process of forming a $Block$, it can be thought of as a function $f : collection[node] \rightarrow Block$. 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 $f : collection[node] \rightarrow Block$ 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 [6]. 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.

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

How are delegates selected

Delegates are selected locally, by passing the previous block 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.

Edges and delegate selection