

# Constellation Light Paper

<b>Introduction</b>	<b>1</b>
<b>Evolution from synchronous to asynchronous consensus</b>	<b>1</b>
<b>Reputation Based Consensus</b>	<b>4</b>
<b>Partial/Full Node Scaling</b>	<b>4</b>
<b>Hylochain - Channel Based Application Support</b>	<b>5</b>
<b>Token Model and Relation to Hylochain</b>	<b>7</b>

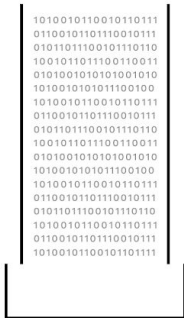
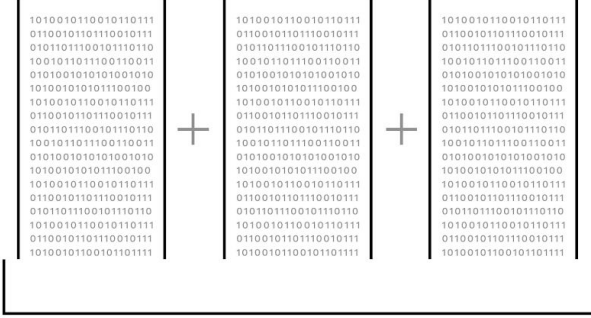
**By: Wyatt Meldman-Foch**

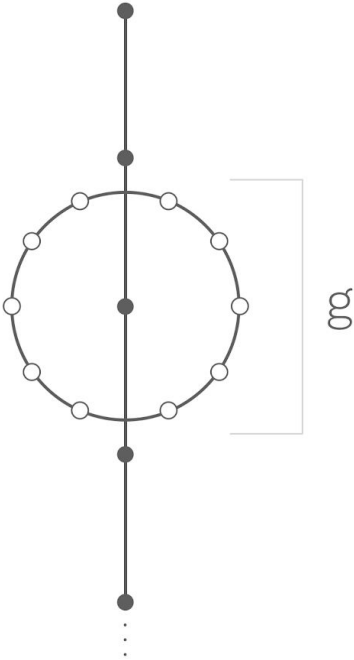
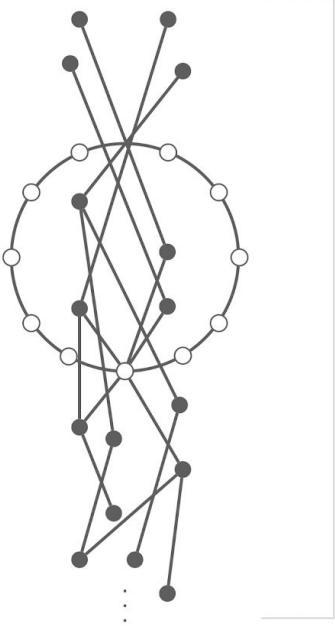
## Introduction

The following is meant to describe fundamental technological components of the Constellation protocol and illustrate the mechanics visually. A full breakdown of architecture and algorithms will follow with our technical white paper which will be released closer to our public network rollout.

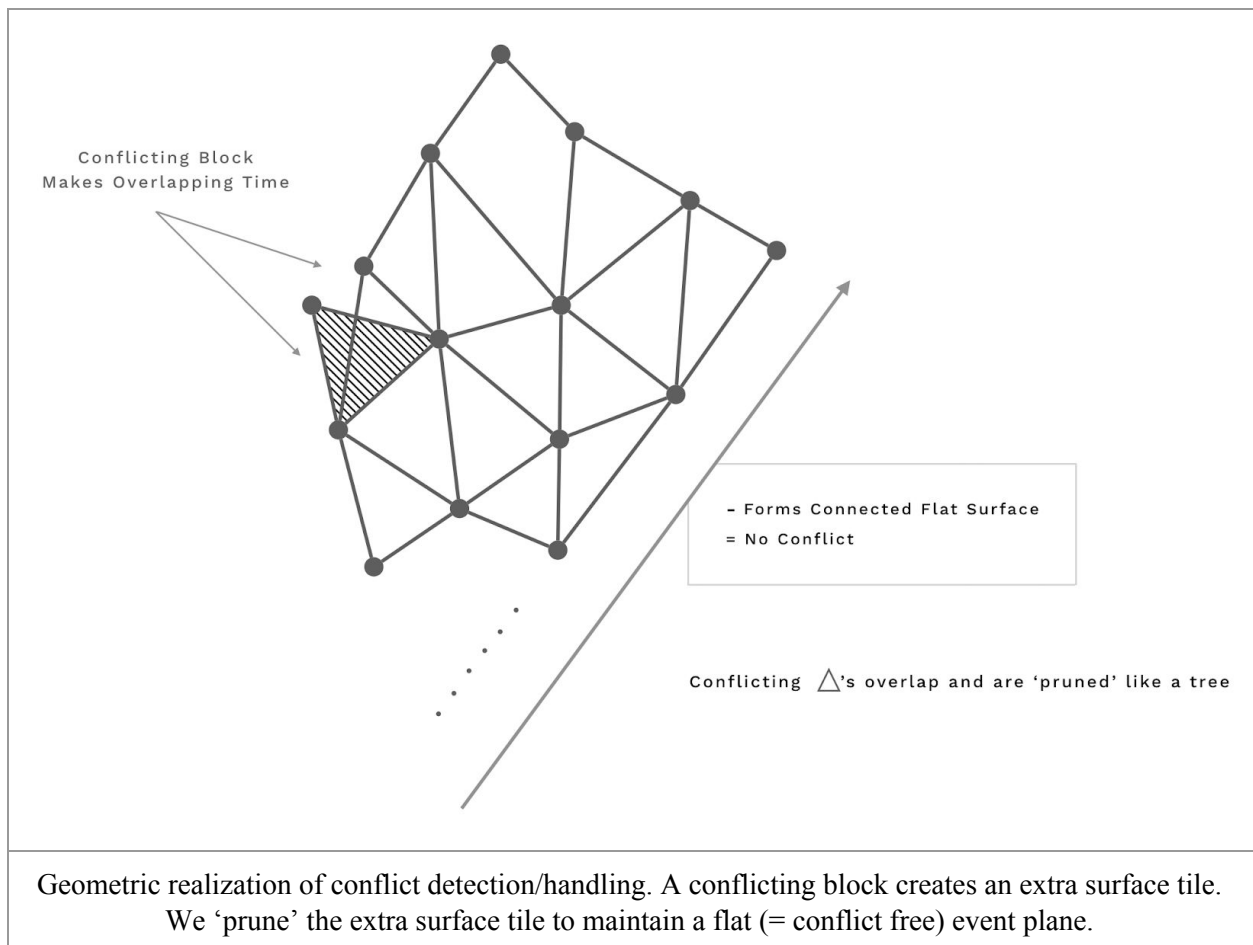
## Evolution from synchronous to asynchronous consensus

Nodes are selected using a deterministic process (the same used in DHTs like bittorrent), which dynamically adjusts the responsibilities of nodes for ‘facilitating’ validation, or more plainly - performing consensus. We choose groups of 3 nodes, and execute consensus rounds in parallel, so one node could be a facilitator in multiple blocks. This allows us to process transactions asynchronously, which essentially means we have multiple blockchains forming at once. The process is like a web being formed with many strands as opposed to a single string forming knots over time. Asynchronous or concurrent processed are the backbone of scalable programming, because it allows all of the resources of a computer to be used, speeding up overall computation. This web is called a directed acyclic graph or DAG in computer science.

 <p><math>g = \text{AREA OF PIPE}</math></p>	 <p><math>= M \cdot g = M \cdot \text{AREA OF PIPE}</math></p>
<p><b>BLOCKCHAIN</b></p> <p>Pipe width of a linear blockchain</p>	<p><b>DIRECTED ACYCLIC GRAPH</b></p> <p>Multiplicative effect of a DAG allowing for multiple concurrent blockchains</p>
<p>The pipe-width of a linear blockchain vs the multiplicative effect of a DAG, where we have multiple concurrent blockchains.</p>	

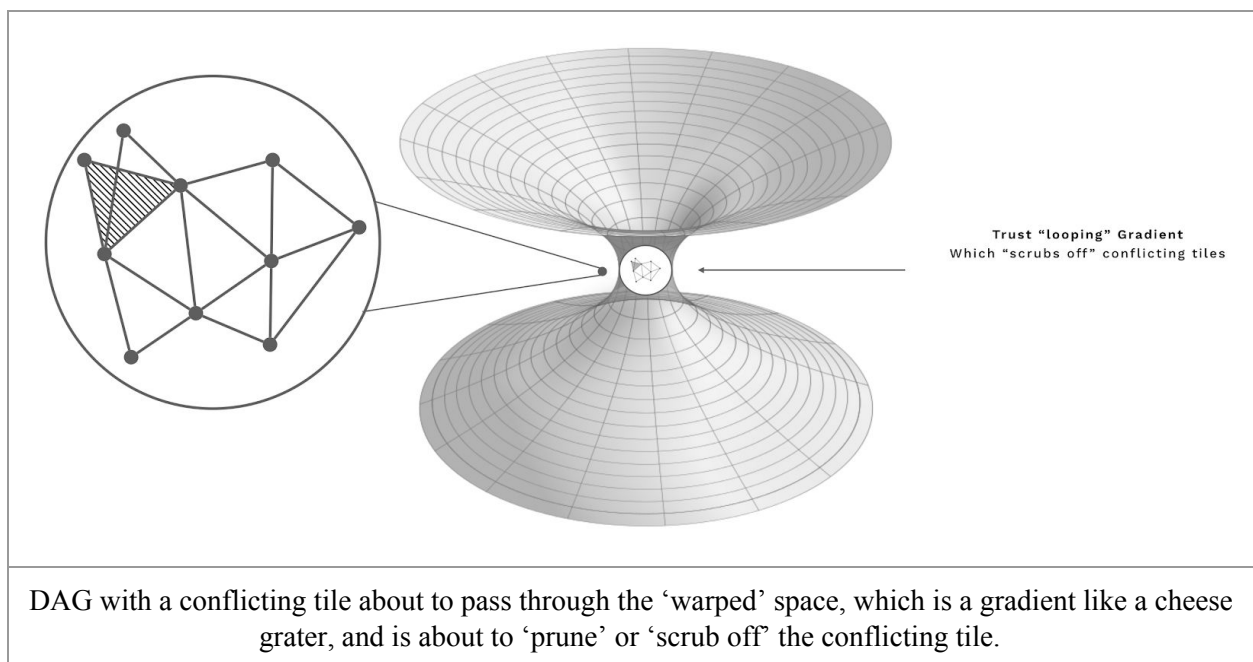
 <p><b>Linear Blockchain</b> Geometric realization of a linear Blockchain</p>	 <p><b>Hierarchically Partitioned DAG</b> Geometric realization of a Directed Acyclic Graph</p>
<p>Geometric realization of a linear blockchain VS a DAG. Black dots are Blocks, white dots are nodes</p>	

We use 3 nodes in each consensus round because it gives us some interesting mathematical processes for reasoning about state, by forming a 'surface plane' across data in the form of tiled triangles. The protocol then uses the triangles to 'stitch together' an optimal surface that is free of redundant or conflicting data and has the minimal triangles possible. Algorithmically this is analogous to the 'min-cut' of a graph and mathematically to the derivative or an optimization function (of which the function finds the shortest path that it can traverse across the surface). This shortest path is equivalent to the optimal storage of data (transactions) within the DAG. Conflicting triangle 'tiles' are 'pruned' to keep the event surface flat and conflict free.



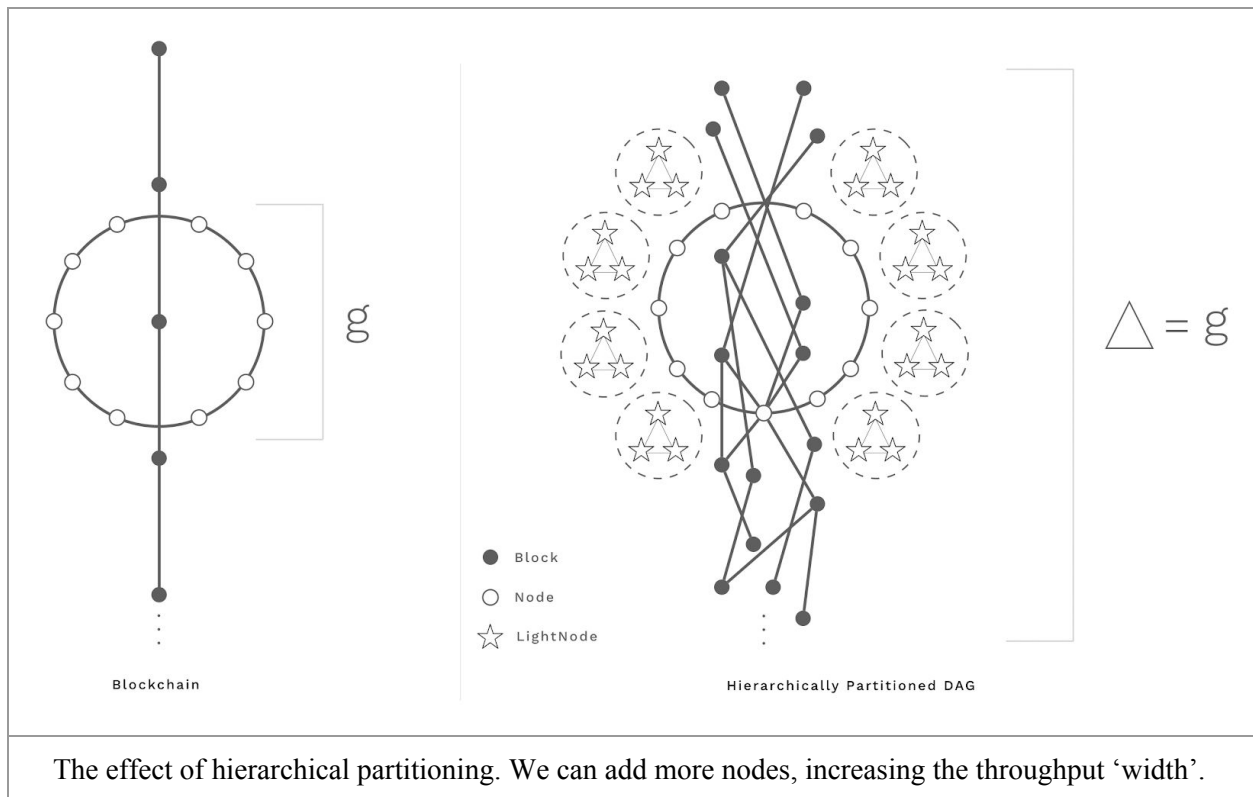
## Reputation Based Consensus

In an optimal decentralized p2p reputation system, every node should be able to independently determine its trust of other nodes. Our system uses a special model which incorporates transitive relationships, or the relationship that a node has with other nodes, in assigning a global score. “You’re only as good as the company you keep”. The end result is a “warping” or gradient based upon transitive trust or reputation across all nodes within \$DAG or a state channel. This can be thought of as a brush or a cheese grater that scrapes on top of the “surface plane” and chooses which “triangle tiles” to scrape off vs which to keep. This is how conflict logic actually prunes “triangle tiles”.



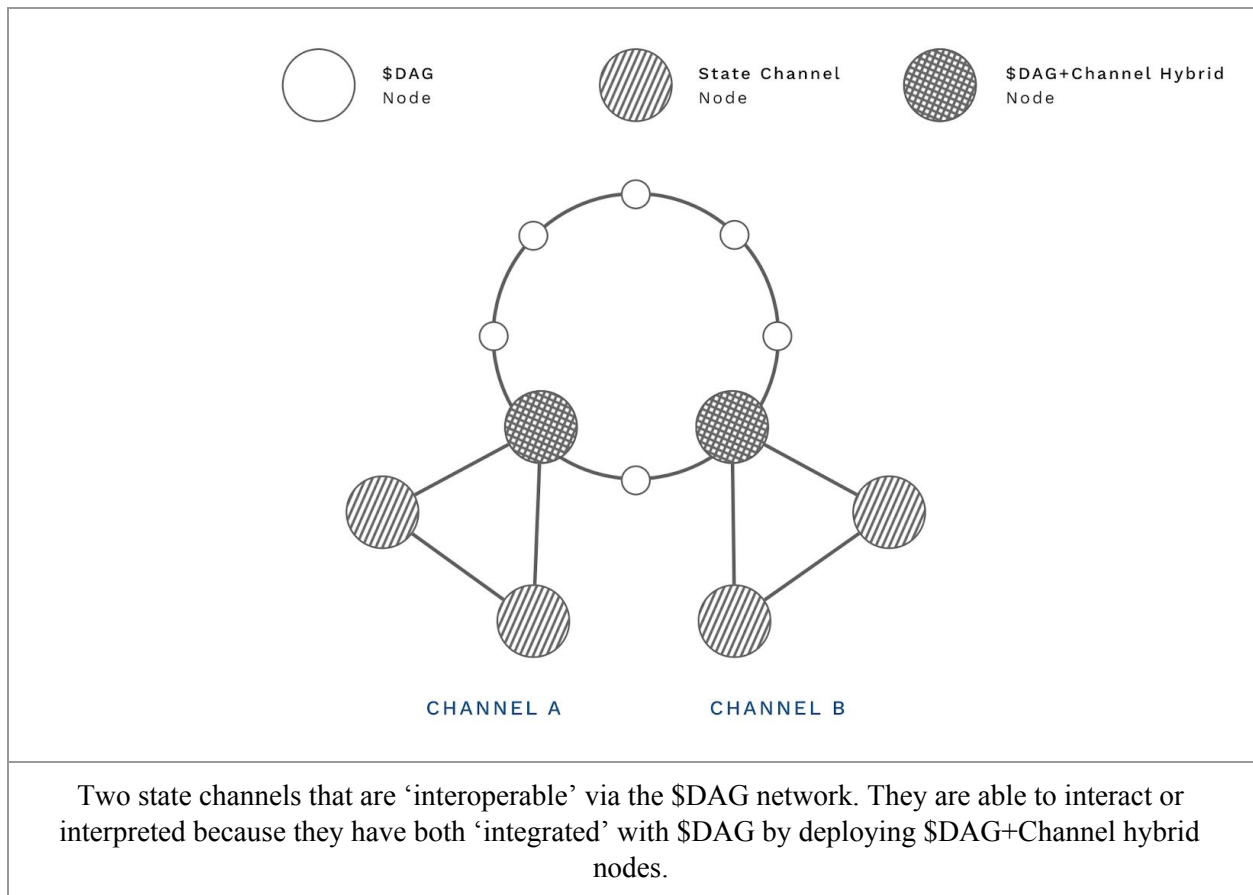
## Partial/Full Node Scaling

In network theory, generally, the optimal arrangement is known as 'scale free' which can be described as a hierarchical arrangement with large central nodes orchestrating many smaller peripheral nodes. This arrangement is seen in nature and most notably the internet. Constellation uses this architecture to 'scale out', or increase throughput, or the width of our DAG.

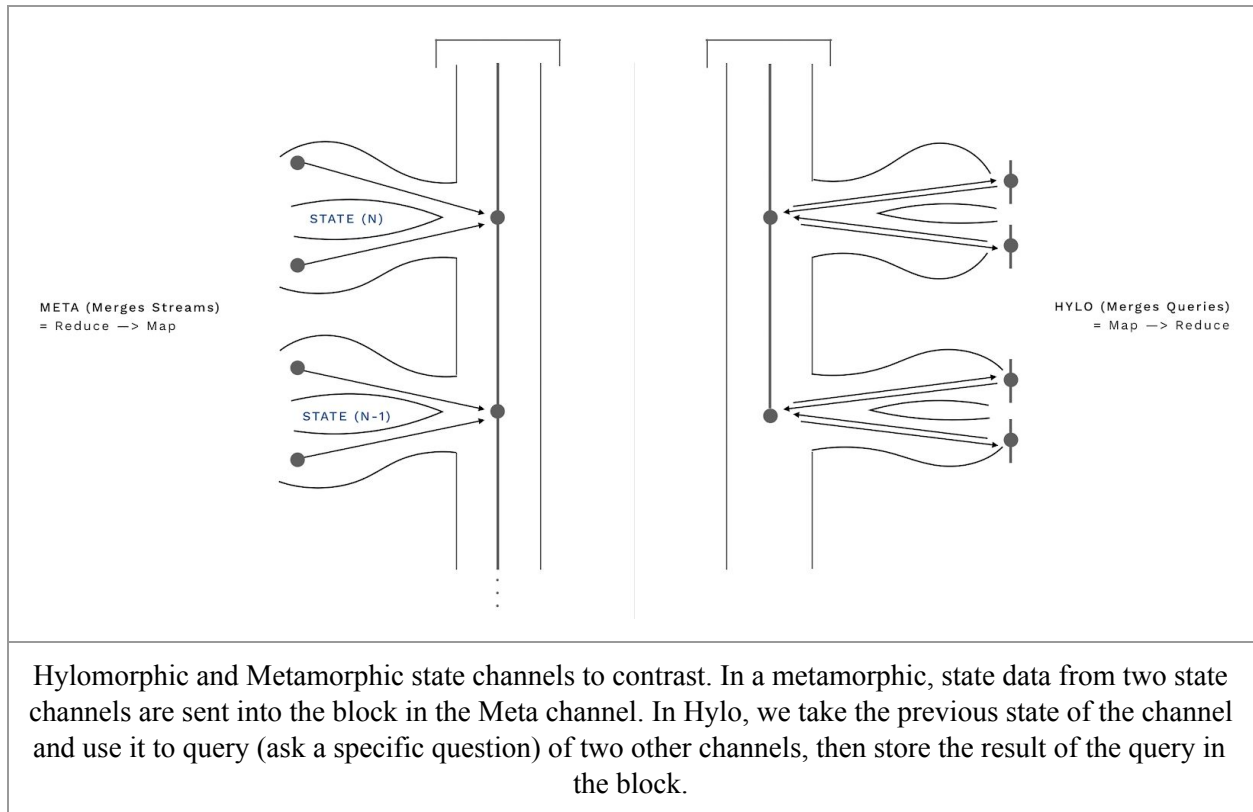


## Hylochain - Channel Based Application Support

Our approach to application support can be thought of as a “decentralized smart contract platform”. Instead of a central network executing all logic and processing all application data, constellation orchestrates application data with “state channels” which can be thought of like a television station broadcasting all data from a stateful system. Each state channel can implement its own validation logic, allowing for a solution to the Oraclization problem by allowing end to end authentication of data producers and transitively validation of complex stateful systems. State channel networks allow concurrent application support, speeding up acceptance times which, in a smart contract network are bottlenecked by traditionally synchronous consensus.



The reason it's called Hylochain is because our approach to application support used the functional programming model of Recursion Schemes to create a MapReduce interface. Specifically, the Hylomorphism and Metamorphism recursion schemes can be integrated to create verifiable queries and streaming joins across state channels by verifying algebraic data types in the same way that op-codes are verified for smart contracts. The end result is a functional MapReduce interface that is familiar to data engineers and interoperable with existing big data technology.



## Token Model and Relation to Hylochain

When a state channel is created, it can be integrated into the \$DAG channel but deploying an ACI or Application Chain Interface. This interface is just a JSON object with configuration information and a public key associated to the channel itself. The reason we associate a public key with the state channel is to create a brokerage mechanism for state channel data. When a state channel is deployed, the developers configure how payments from the \$DAG network are distributed amongst the nodes and operators.

