BLOCKSCIENCE

On Engineering Economic Systems

Warren Center for Network and Data Sciences
University of Pennsylvania
April 19, 2018

Michael Zargham, Phd
Founder & CEO, BlockScience
@mzargham

Speaker Bio

- Actively Researching Blockchain enabled Coordination and Decision Systems with BlockScience Team
- Contributor & Advisor to Several Blockchain Projects
- Former Director of Data Science and Architect of Data & Decision Systems at Cross MediaWorks
- PhD in Engineering, UPENN; studied Decentralized Optimization and Decision Science
- Academic background in optimal and stochastic Control Theory, Convex Optimization, Game Theory & Applied Math
- Quantitative Analyst and Decision Systems Engineer since 2005.

BlockScience Research Team

- **Matthew Barlin,** Research Engineer

  System Architecture, Model Testing, and Empirical Evaluation for Economic Networks

- **Markus Koch,** Research Engineer

  Blockchain Engineer with a focus on Data Engineering, Data and Decision Sciences

- **Joshua Jodesty**, Data Engineer

  Data Platform Architecture and Distributed Computing Engineer

- **Charlie Rice,** Data Scientist

  Numerical Experiments using Stochastic Processes and Exploratory Empirical Analysis

- **Awesome Operations Staff:**
  Charlie, Nick, Nelly, Kirstin, Antonio

- **Coming Soon, New Researchers:**
  Fernando and Katie

What is Blockchain?

Blockchain: a Data Technology

Also called **Decentralized Ledger Technology** (DLT)

In general, a blockchain network is defined by:

A data structure: the "ledger" but can be generalized, i.e. for the Ethereum network the data structure is the state of the EVM

A set of methods which operate on the data structure; in the simplest case this is a transaction sending tokens from one address to another

A consensus protocol: set of rules for agreeing on the true state of the data structure, based on verifying the validity of transactions

A community: the set of agents (human or machine) which are participating in the network; lightweight clients may broadcast transactions but full nodes are required to participate in validation process (consensus protocol)

Consensus and Coordination

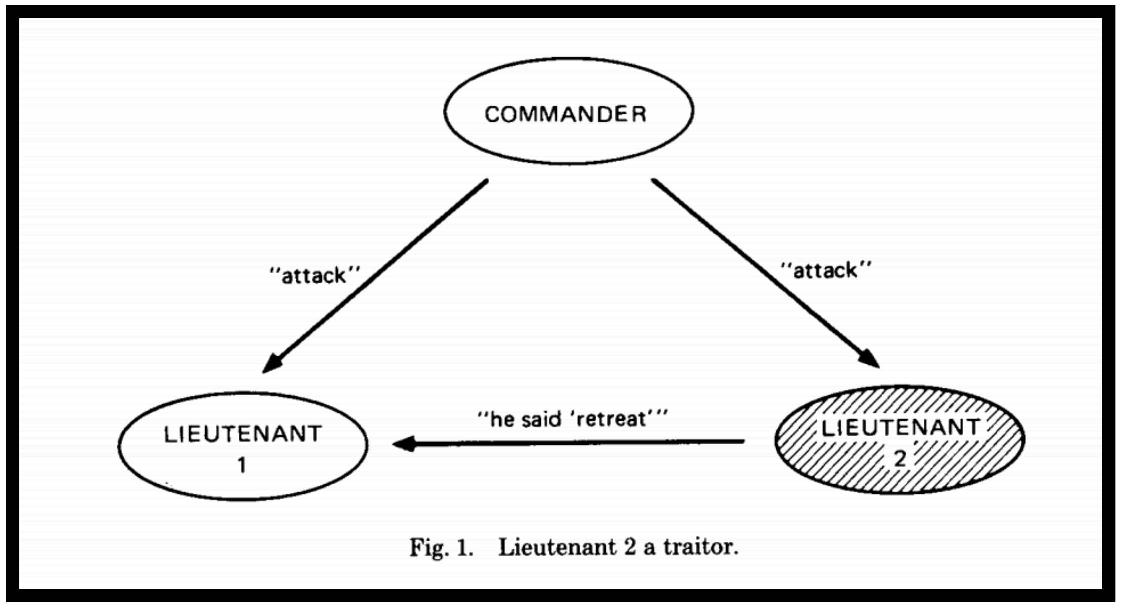


Fig. 1. Lieutenant 2 a traitor.

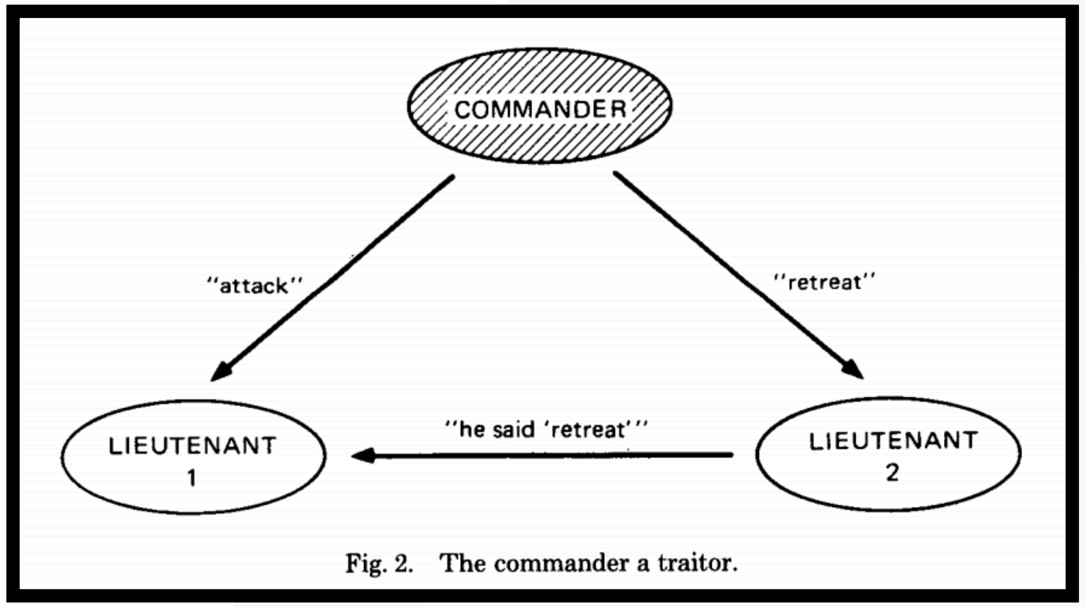
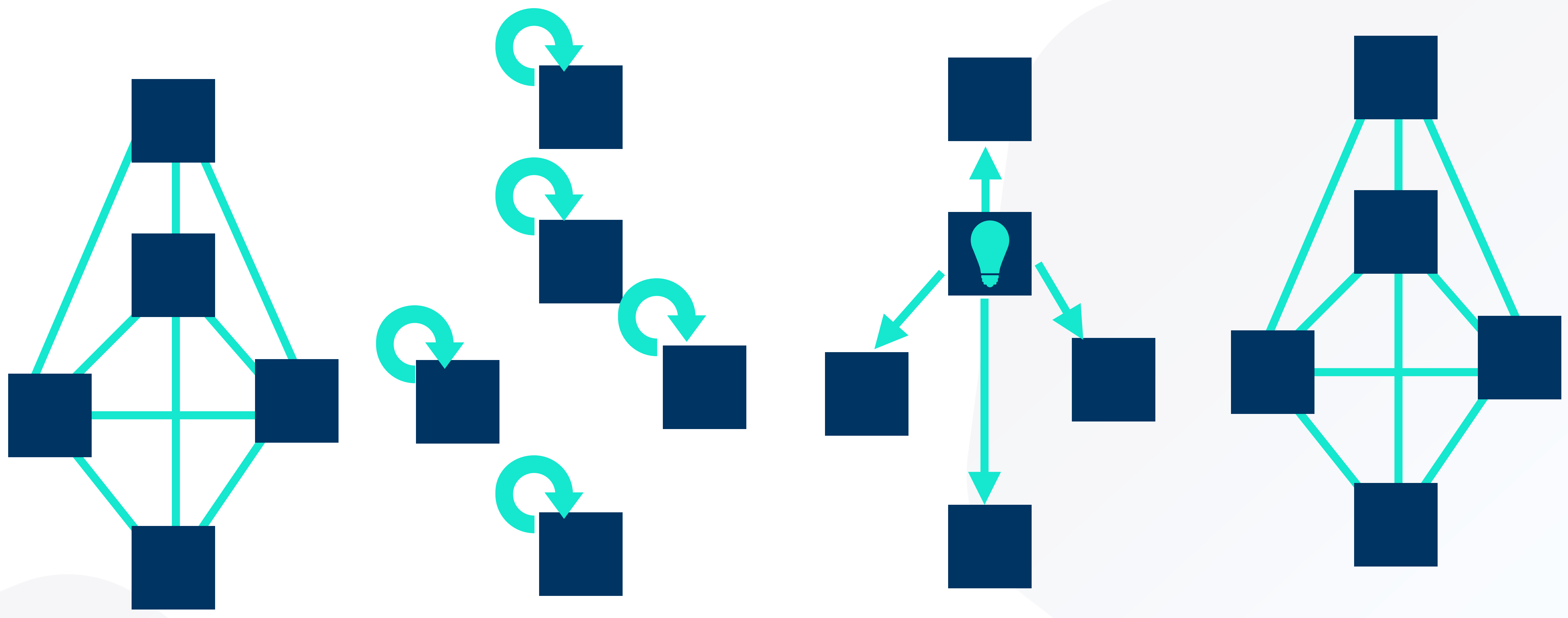


Fig. 2. The commander a traitor.

Byzantine Generals Problem (1982)
LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
http://bnrg.cs.berkeley.edu/~adj/cs16x/hand-outs/Original_Byzantine.pdf

https://en.wikipedia.org/wiki/Byzantine_fault_tolerance

Blockchain-Enabled Coordination



Agree On Prior State

Compute locally
by operating on the same priors

One Agent Declares
'solution'
candidate posterior state

Network Validates
Solution, adopting the
Posterior state

Coordinated Computation: Smart Contracts

- A Smart contract is "on-chain" code which is executed by every node in the network.
- The Ethereum network is a Turing complete virtual machine but computational "gas" limits are imposed on each block functionally limiting the computational complexity
- Ethereum is a public Network for shared computation but private or "permissioned" networks can be built based on Quorum or Hyperledger and others
- Contracts are Passive — they are acted upon but cannot act autonomously
- Trusted "micro-services" whose computation is verifiable
- Computation can be internal state and external context dependent
- Due to passivity, reference data from the outside world is delegated to oracles which provide the contextual data when the contract is called
- Due to computational limits complex computations still happen off-chain

Blockchain Technology Status

Blockchain Menagerie

bitcoin

ripple

QTUM

NEO smart economy

ethereum

CARDANO

EOS™

Quorum™

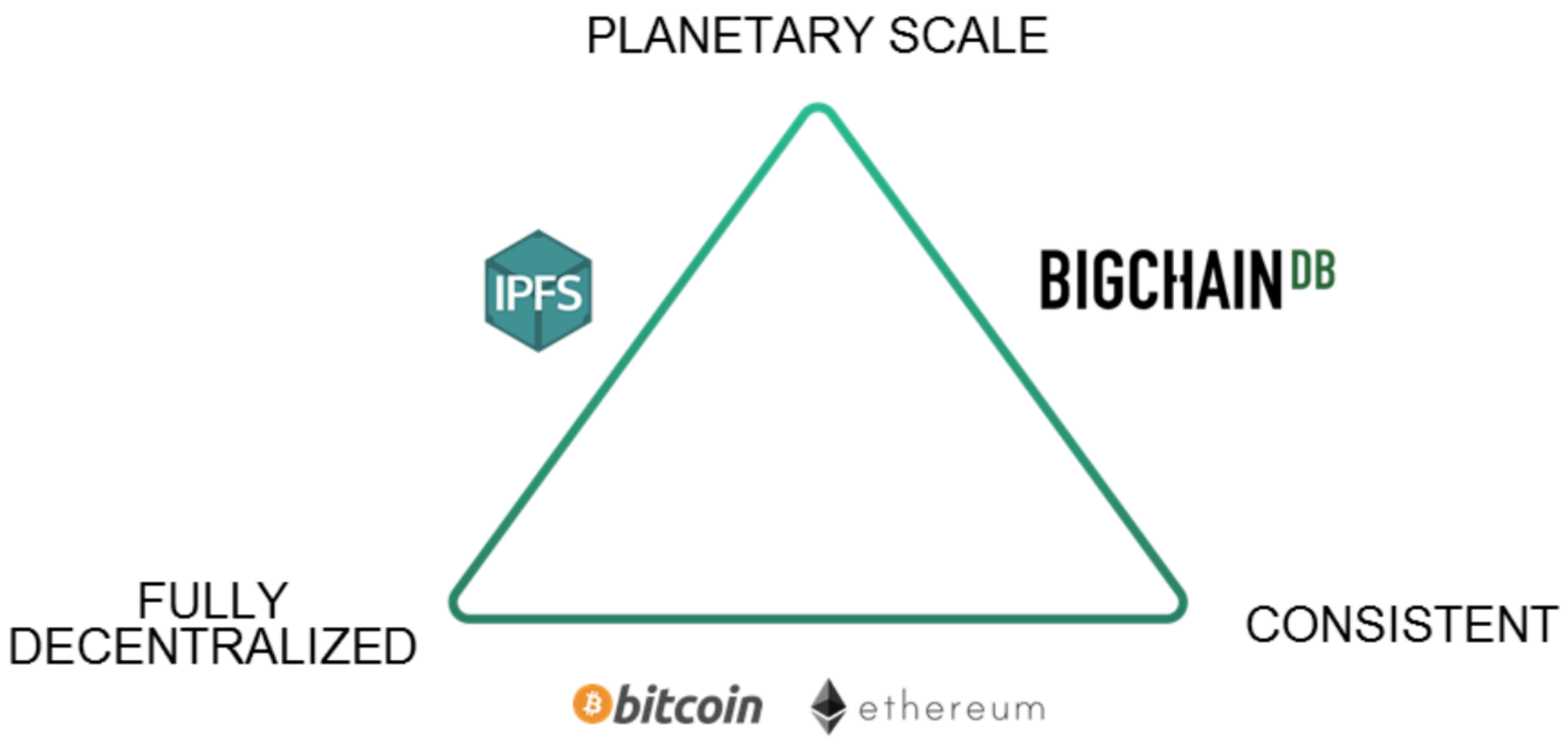
STELLAR

HYPERLEDGER PROJECT

RCHAIN COOPERATIVE

corda

Scalability and Performance



Credit: Trent McConaghy, PhD
CTO BigChainDB
https://blog.bigchaindb.com/the-dcs-triangle-5ce0e9e0f1dc

Generalizing Crypto-Assets

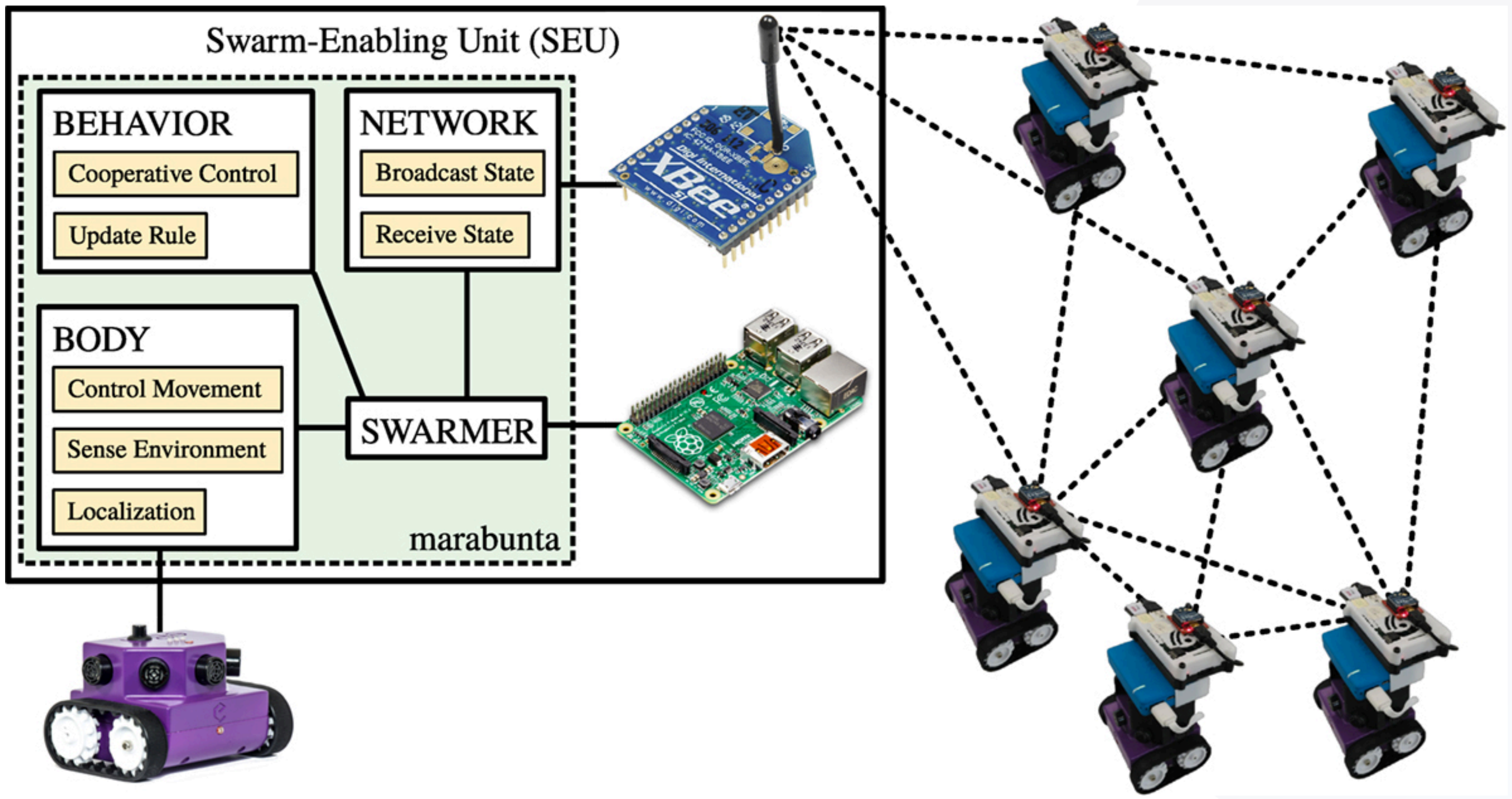
	Fungible	Unique
Divisible	Cryptocurrencies	Equity or Shares in specific Assets
Discrete	Tokens representing physical Commodities	Tokens representing Titles or Deeds

- Tokens are digital representations of Assets which may or many not be tangible (goods vs rights)
- Ownership in the asset is governed by a decentralized ledger
- Crypto-Assets allow ownership to be transferred via smart contract
- Smart Contract simply automate the execution of workflows defined in legal contracts

Engineering Economic Networks

Engineered Multi-agent Network

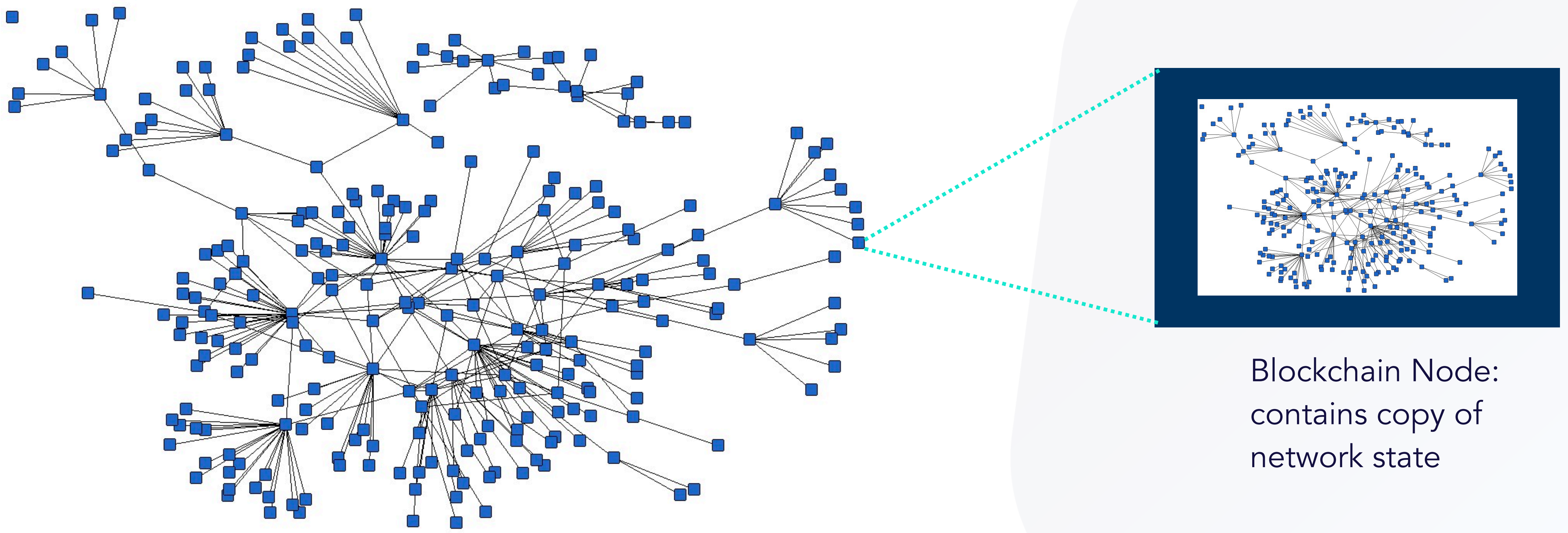
Consider an analogy to a robotic network:



https://www.frontiersin.org/articles/10.3389/frobt.2017.00012/full

Blockchain as Data Infrastructure

Blockchain Network State



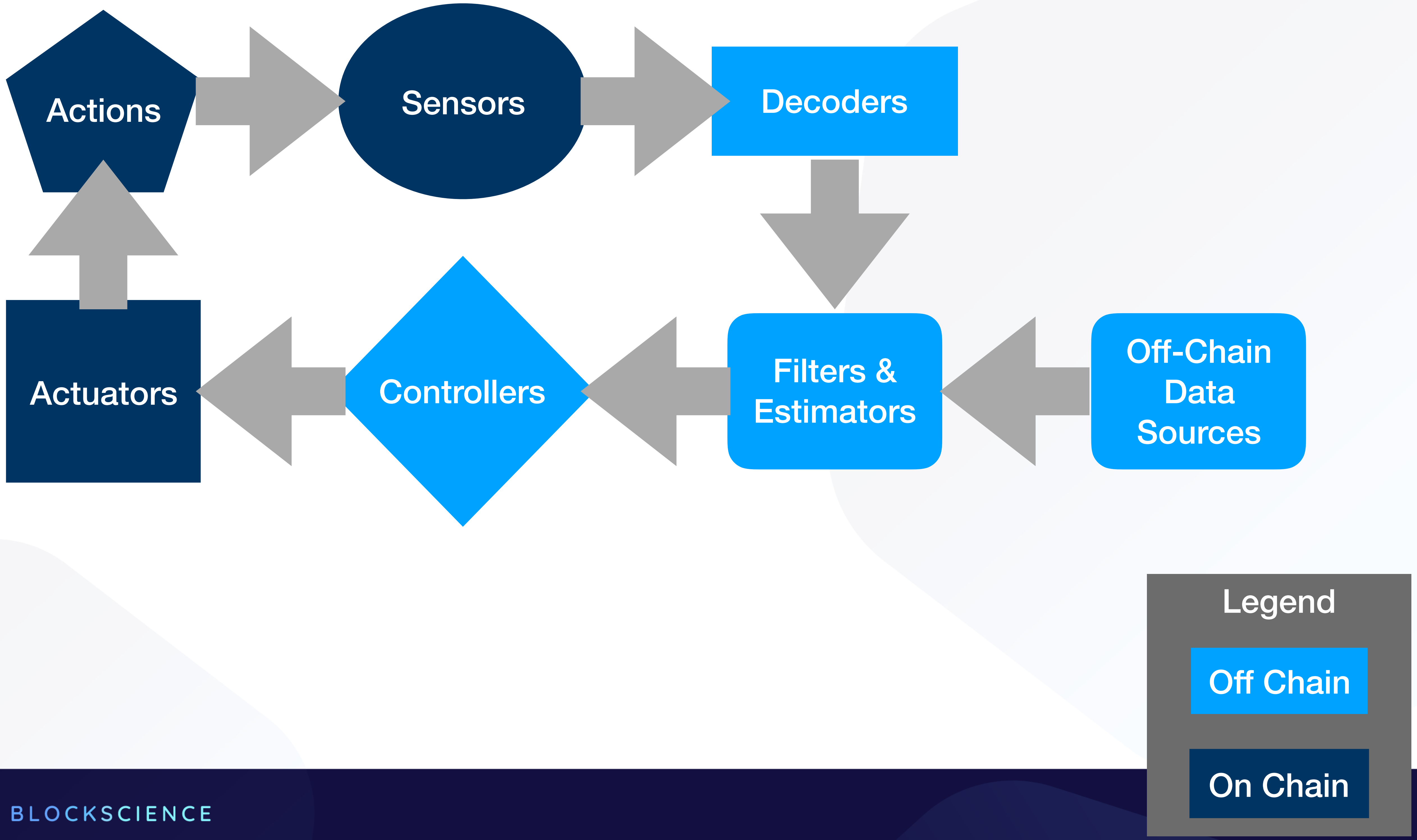
Blockchain Node: contains copy of network state

In addition to being the **Plant** of the system,
Each blockchain node is effectively a **Sensor** for the entire network

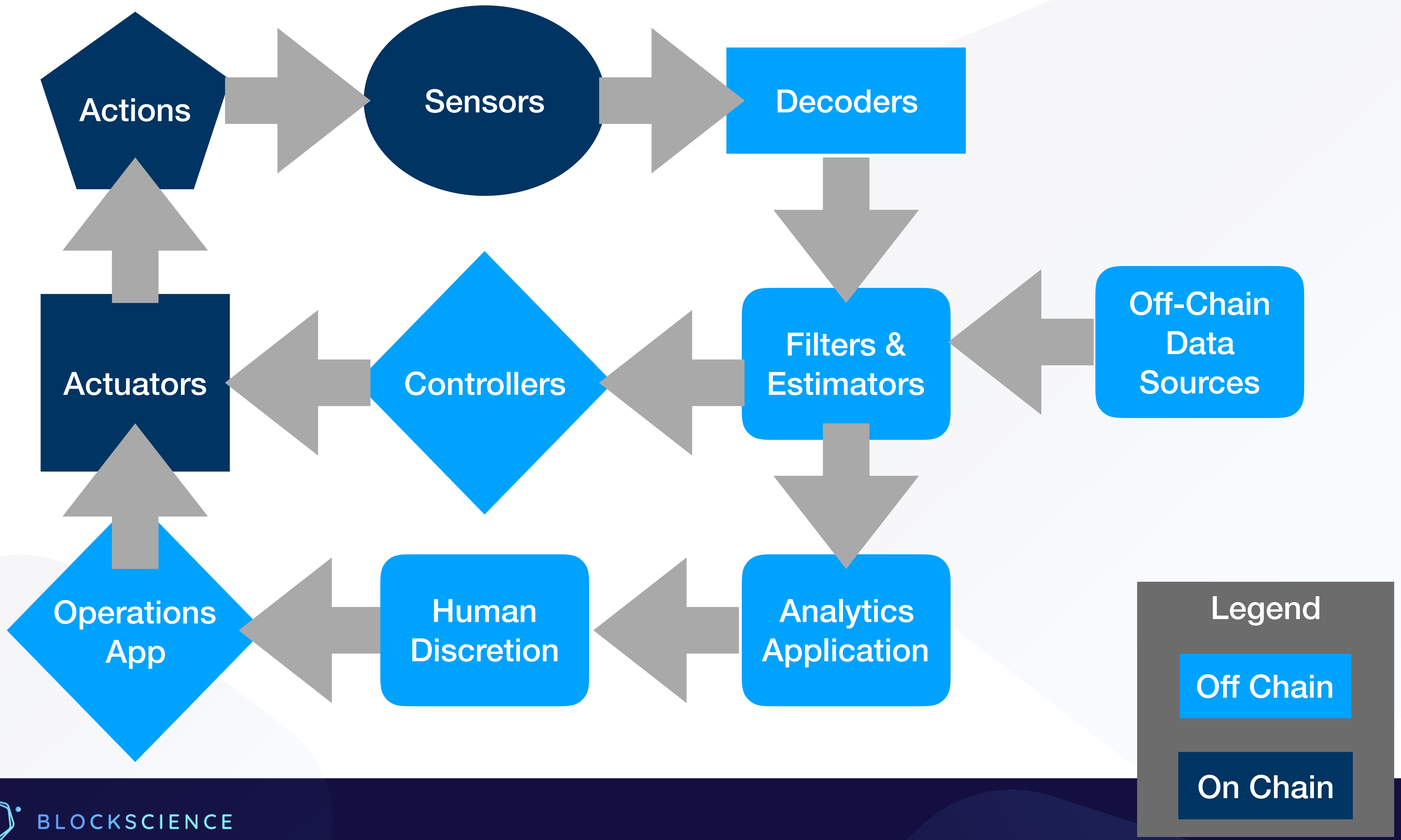
Data Driven Decision Making Agents

- **Sensors** — the blockchain data structure, APIs to market data feeds and other sources
- **Decoder** — ETL software that accesses, sub-selects, and restructures blockchain data for use
- **Filters & Estimators** —processes that remove noise and fuse signals to create useful signals
- **Controllers** — software that computes decision variables from signals available
- **Actuators** — smart contract code which defines and carries out decisions made by agents in the network
- **Actions** — the actions that are taken by agents; the loop is closed as these actions appear in the Blockchain data structure making them observable

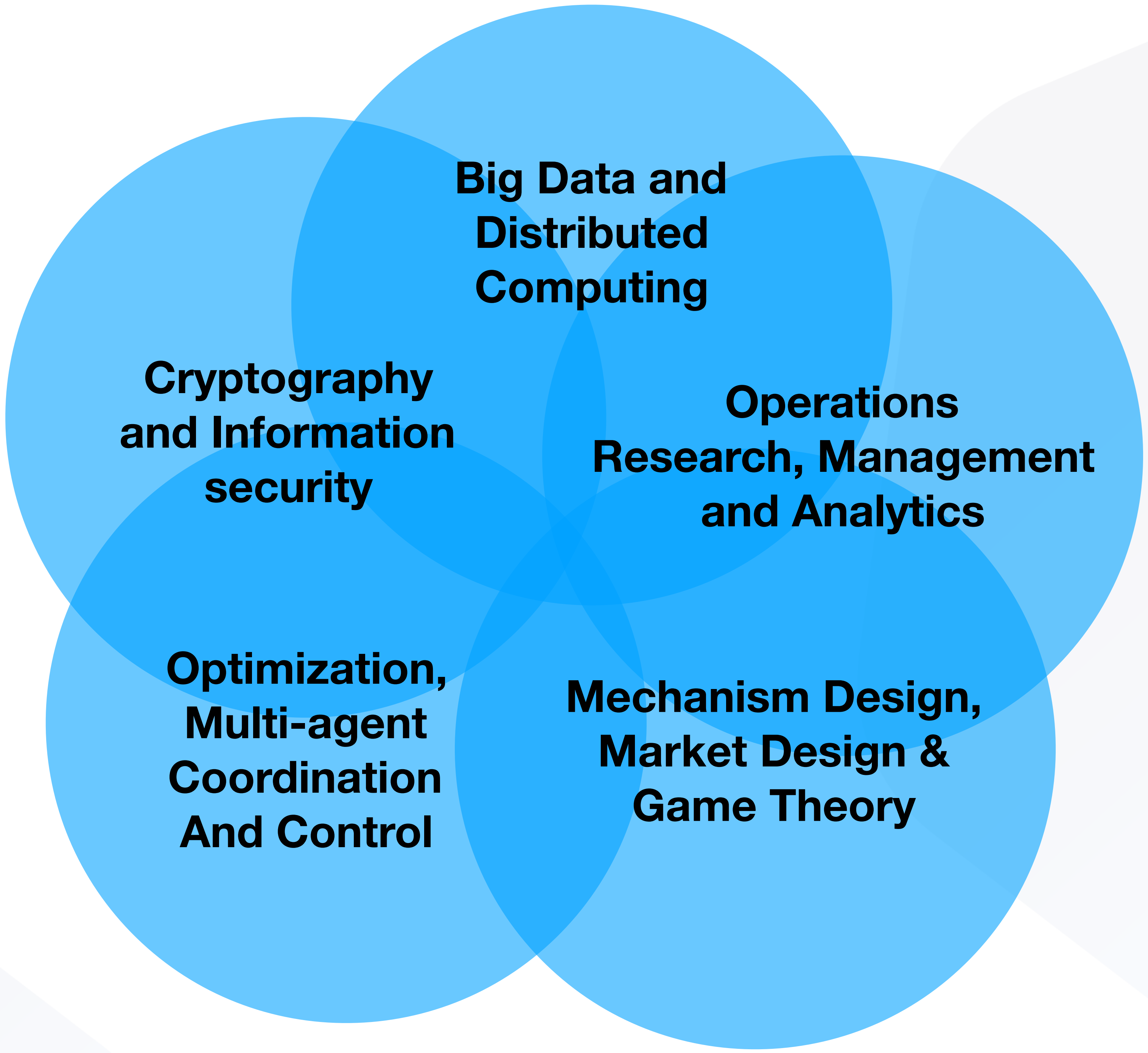
Closed Loop Decision Systems



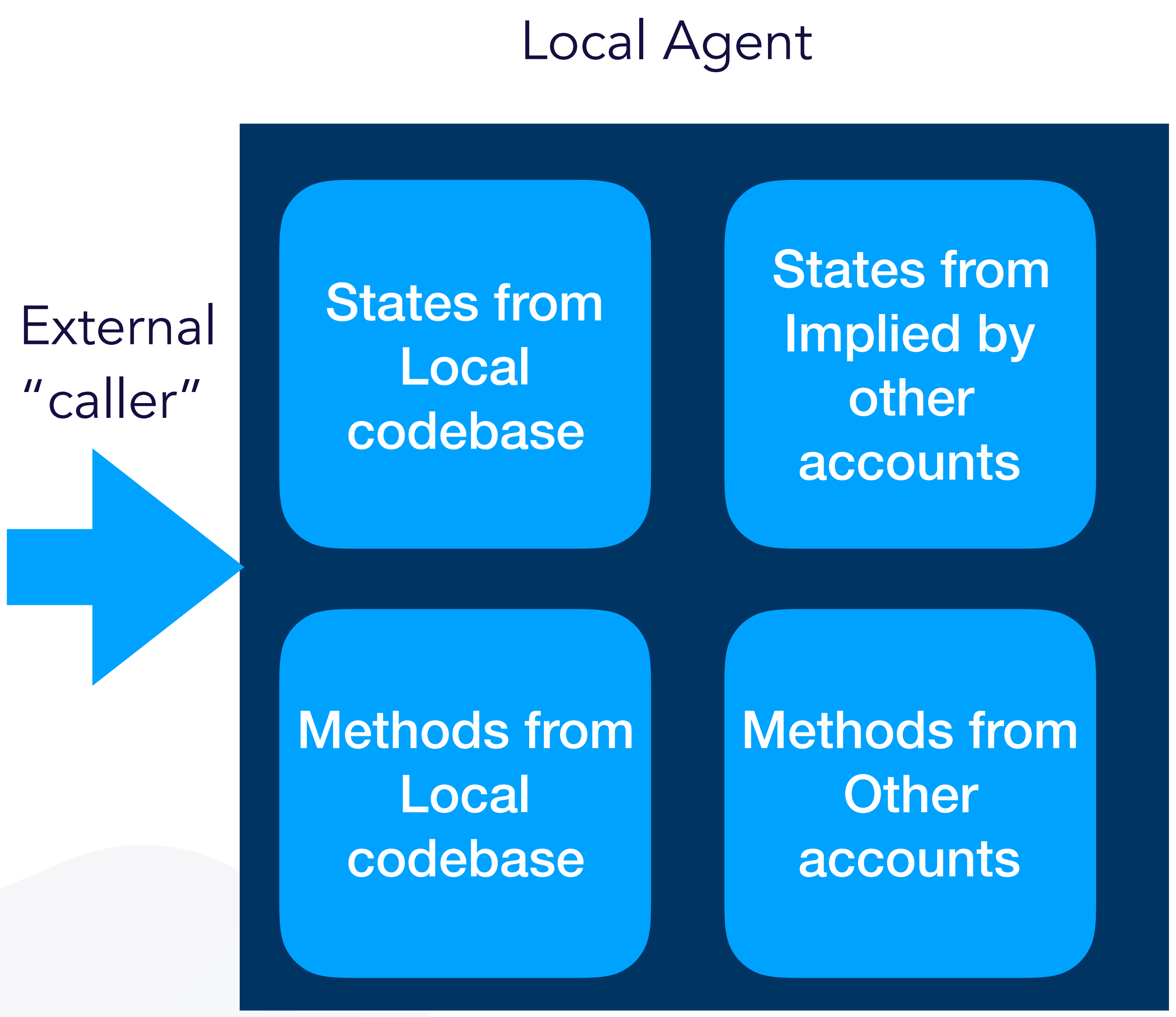
Human In the Loop Decision Systems



Automated, Secure, Data Driven Decision Making


Big Data and Distributed Computing
Cryptography and Information security
Operations Research, Management and Analytics
Optimization, Multi-agent Coordination And Control
Mechanism Design, Market Design & Game Theory

Smart Contracts as Agents



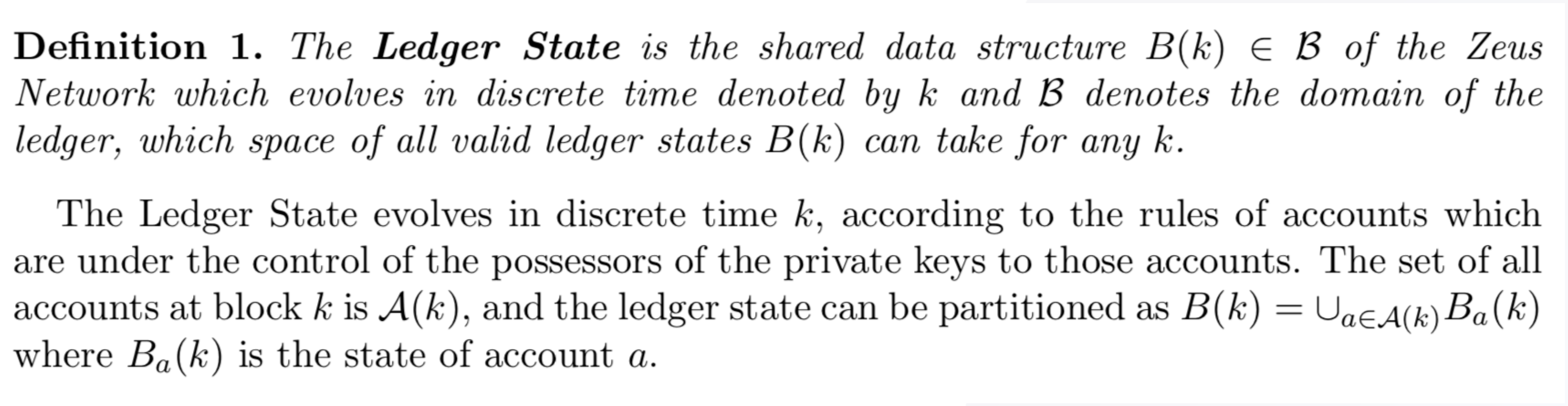
Local Agent
External "caller"
States from Local codebase
States from Implied by other accounts
Methods from Local codebase
Methods from Other accounts

State Changes

- Caller with Account sk uses a local method to change the local state and possibly others (e.g. send native funds)
- Caller with Account sk uses another account method to change local state and possibly others (e.g. ERC20 send)
- Caller different Account sk uses a local method to change local state and possibly others (e.g. ERC20 send from the perspective of the ERC contract)
- Caller with different account sk uses a method from another agent to change the local agent's state (e.g. proportional airdrop of tokens)

Formal System Model

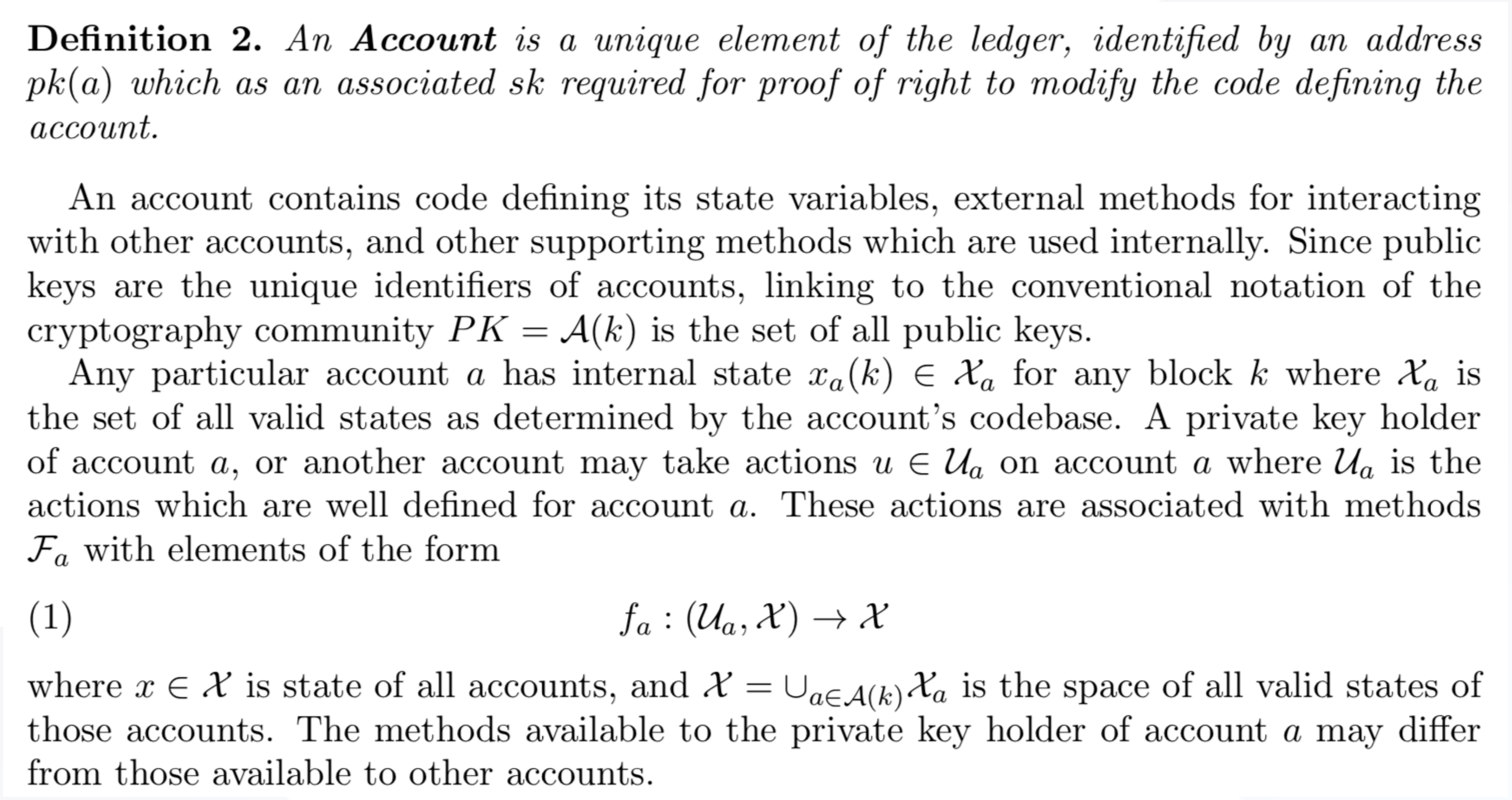
Partitioning the Global State into Local Accounts



Definition 1. *The* ***Ledger State*** *is the shared data structure $B(k) \in \mathcal{B}$ of the Zeus Network which evolves in discrete time denoted by k and $\mathcal{B}$ denotes the domain of the ledger, which space of all valid ledger states $B(k)$ can take for any k.*

The Ledger State evolves in discrete time k, according to the rules of accounts which are under the control of the possessors of the private keys to those accounts. The set of all accounts at block k is $\mathcal{A}(k)$, and the ledger state can be partitioned as $B(k) = \cup_{a \in \mathcal{A}(k)} B_a(k)$ where $B_a(k)$ is the state of account a.

Accounts Define State Variables and Action Spaces



Definition 2. *An* ***Account*** *is a unique element of the ledger, identified by an address $pk(a)$ which as an associated sk required for proof of right to modify the code defining the account.*

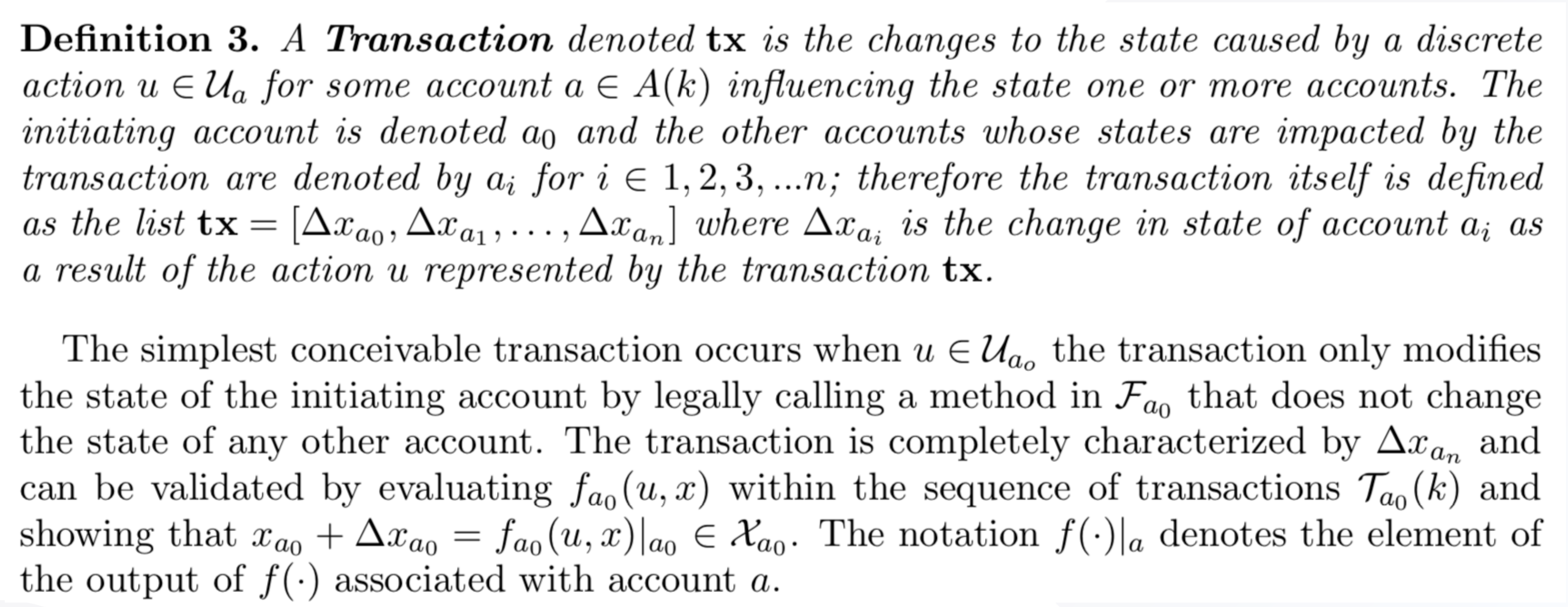
An account contains code defining its state variables, external methods for interacting with other accounts, and other supporting methods which are used internally. Since public keys are the unique identifiers of accounts, linking to the conventional notation of the cryptography community $PK = \mathcal{A}(k)$ is the set of all public keys.

Any particular account a has internal state $x_a(k) \in \mathcal{X}_a$ for any block k where $\mathcal{X}_a$ is the set of all valid states as determined by the account's codebase. A private key holder of account a, or another account may take actions $u \in \mathcal{U}_a$ on account a where $\mathcal{U}_a$ is the actions which are well defined for account a. These actions are associated with methods $\mathcal{F}_a$ with elements of the form

$$f_a : (\mathcal{U}_a, \mathcal{X}) \to \mathcal{X} \quad (1)$$

where $x \in \mathcal{X}$ is state of all accounts, and $\mathcal{X} = \cup_{a \in \mathcal{A}(k)} \mathcal{X}_a$ is the space of all valid states of those accounts. The methods available to the private key holder of account a may differ from those available to other accounts.

Transactions: The State Change as the Result of Action



Definition 3. *A* ***Transaction*** *denoted* $\mathbf{tx}$ *is the changes to the state caused by a discrete action* $u \in \mathcal{U}_a$ *for some account* $a \in A(k)$ *influencing the state one or more accounts. The initiating account is denoted* a_0 *and the other accounts whose states are impacted by the transaction are denoted by* a_i *for* $i \in 1, 2, 3, \ldots n$*; therefore the transaction itself is defined as the list* $\mathbf{tx} = [\Delta x_{a_0}, \Delta x_{a_1}, \ldots, \Delta x_{a_n}]$ *where* Δx_{a_i} *is the change in state of account* a_i *as a result of the action* u *represented by the transaction* $\mathbf{tx}$.

The simplest conceivable transaction occurs when $u \in \mathcal{U}_{a_o}$ the transaction only modifies the state of the initiating account by legally calling a method in $\mathcal{F}_{a_0}$ that does not change the state of any other account. The transaction is completely characterized by Δx_{a_n} and can be validated by evaluating $f_{a_0}(u, x)$ within the sequence of transactions $\mathcal{T}_{a_0}(k)$ and showing that $x_{a_0} + \Delta x_{a_0} = f_{a_0}(u, x)|_{a_0} \in \mathcal{X}_{a_0}$. The notation $f(\cdot)|_a$ denotes the element of the output of $f(\cdot)$ associated with account a.

Local Transactions Encode the Global State Update

Definition 4. *A* ***Transaction Block*** *is an ordered list of transactions* $\mathbf{TX}(k) = [\mathbf{tx}_0, \mathbf{tx}_1, \ldots, \mathbf{tx}_m]$. *Since transactions which are ordered lists of account state changes, it can interpreted as a flat list, the block is valid if each individual transaction is computed correctly and the state change for each account after each sequential transaction is valid.*

$$\begin{aligned} x_{a_0} + \Delta x_{a_0} = f_a(u,x)|_{a_0} &\quad\in\quad \mathcal{X}_{a_0}, \\ x_{a_1} + \Delta x_{a_1} = f_a(u,x)|_{a_1} &\quad\in\quad \mathcal{X}_{a_1}, \\ &\ldots \\ x_{a_n} + \Delta x_{a_n} = f_a(u,x)|_{a_n} &\quad\in\quad \mathcal{X}_{a_n}, \end{aligned}$$

$$\Delta x(k) = x(k) - x(k-1) \tag{5}$$

is completely characterized by $\mathbf{TX}(k)$ and can be evaluated account-wise according to equation (4).

Networked Second Order Discrete Dynamical System

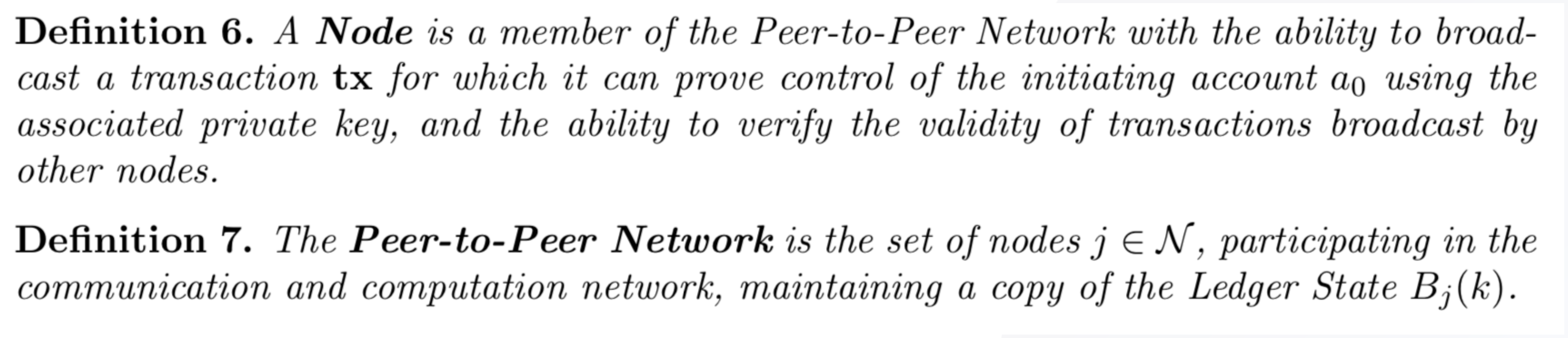
Definition 5. *A **Block** is the ledger state $B(k)$at k which given the definitions above can be formally characterized as the pair*

$$B(k) = (x(k), \mathbf{TX}(k)). \tag{6}$$

Property 1. *The evolution of the Ledger State $B(k)$ for $k = 0, 1, 2, \ldots$ is a discrete second order networked system.*

It is a networked system comprised of accounts a with locally defined internal dynamics and rules for interacting according to the account Definition, 2. It is a discrete second order system because the ledger state $B(k)$ contains precisely the states $x_a(k)$ and the backward discrete derivatives $\Delta x_a(k) = x_a(k) - x_a(k-1)$ for all accounts $a \in \mathcal{A}(k)$.

Disambiguating the Term 'Network'



Definition 6. *A **Node** is a member of the Peer-to-Peer Network with the ability to broadcast a transaction* **tx** *for which it can prove control of the initiating account a_0 using the associated private key, and the ability to verify the validity of transactions broadcast by other nodes.*

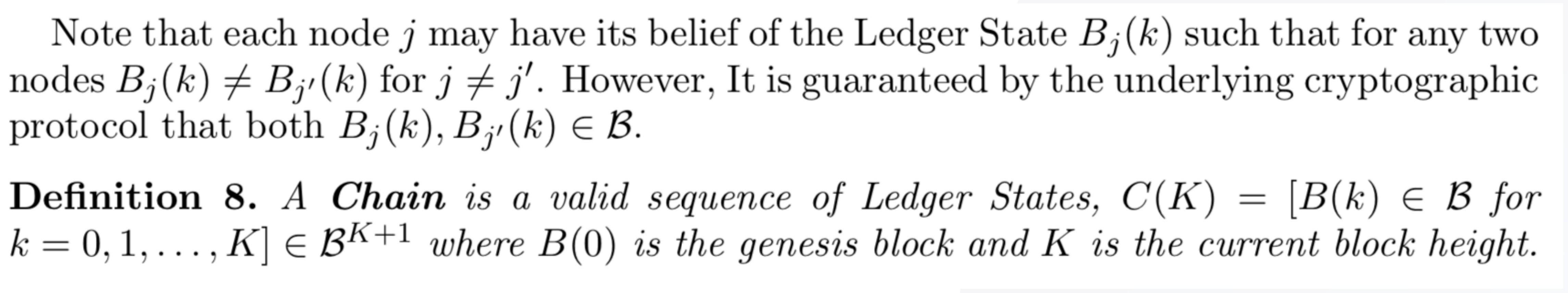
Definition 7. *The **Peer-to-Peer Network** is the set of nodes $j \in \mathcal{N}$, participating in the communication and computation network, maintaining a copy of the Ledger State $B_j(k)$.*

Nodes are Not the Agents, the Accounts are

Relationships between nodes and accounts:

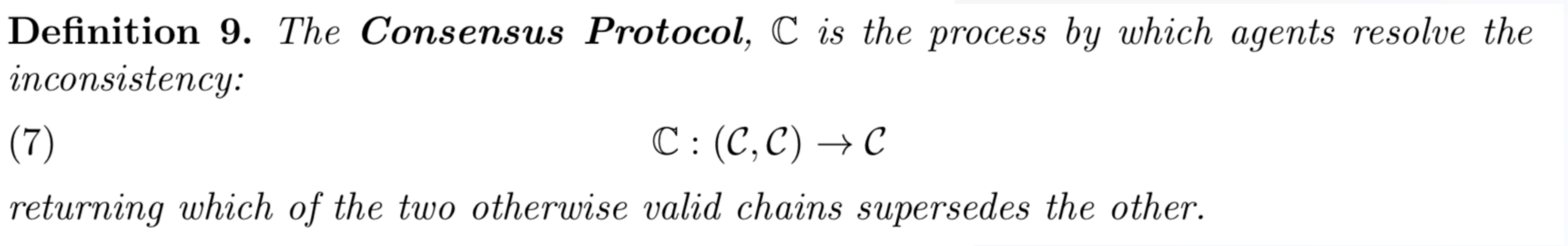
- When a node finds a block, an Account receives the block reward
- Any Node i can use Account a's private key to initiate actions as account a

Agreeing on the State Requires the whole Chain



Note that each node j may have its belief of the Ledger State $B_j(k)$ such that for any two nodes $B_j(k) \neq B_{j'}(k)$ for $j \neq j'$. However, It is guaranteed by the underlying cryptographic protocol that both $B_j(k), B_{j'}(k) \in \mathcal{B}$.

Definition 8. *A* ***Chain*** *is a valid sequence of Ledger States,* $C(K) = [B(k) \in \mathcal{B}$ *for* $k = 0, 1, \ldots, K] \in \mathcal{B}^{K+1}$ *where* $B(0)$ *is the genesis block and* K *is the current block height.*

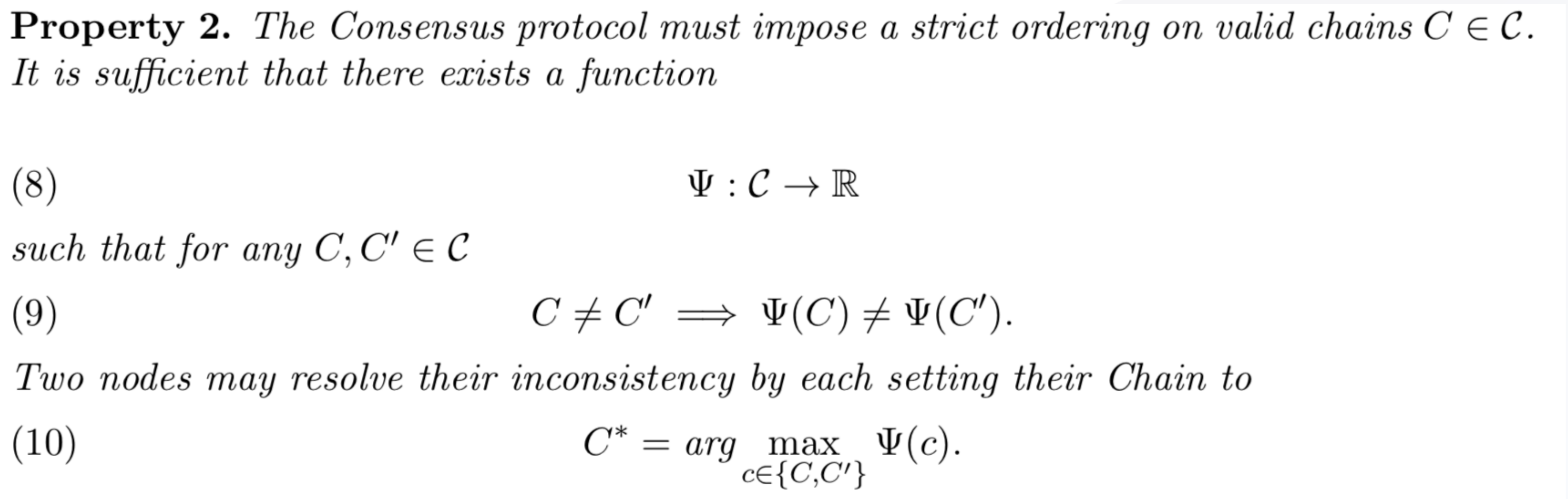


Definition 9. *The* ***Consensus Protocol***, $\mathbb{C}$ *is the process by which agents resolve the inconsistency:*

$$\mathbb{C} : (\mathcal{C}, \mathcal{C}) \rightarrow \mathcal{C} \tag{7}$$

returning which of the two otherwise valid chains supersedes the other.

Conjecture: Sufficient Condition for Consensus



Property 2. *The Consensus protocol must impose a strict ordering on valid chains $C \in \mathcal{C}$. It is sufficient that there exists a function*

$$\Psi : \mathcal{C} \to \mathbb{R} \tag{8}$$

such that for any $C, C' \in \mathcal{C}$

$$C \neq C' \implies \Psi(C) \neq \Psi(C'). \tag{9}$$

Two nodes may resolve their inconsistency by each setting their Chain to

$$C^* = arg \max_{c \in \{C, C'\}} \Psi(c). \tag{10}$$

Theoretically Speaking — this argument should be constructed with Pr(.)=1 and provide an always eventually consistent Chain from after some fixed number of confirmations

Practically Speaking — one should only need to look back to the last place the chains disagree in order to compute Psi

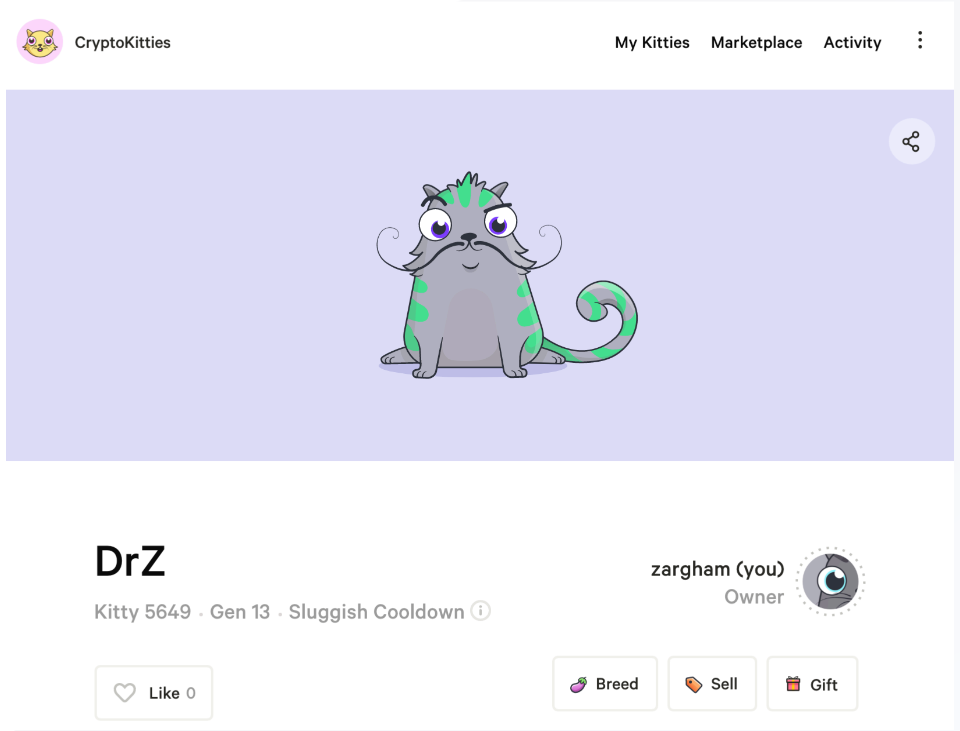
Show me the Data!

Cryptokitties: NFTs, Functions and States

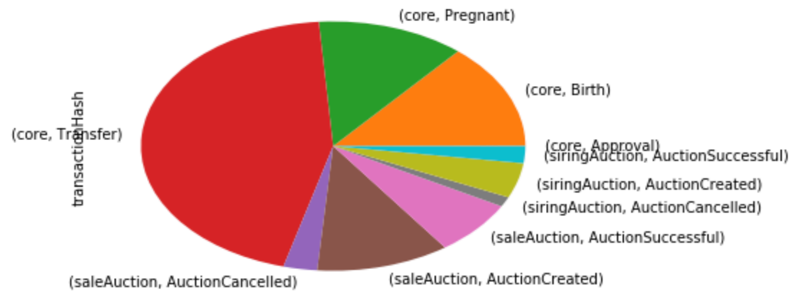
```python
# relevant Events signatures
events_signatures = {
    'AuctionCreated' : 'AuctionCreated(uint256,uint256,uint256,uint256)', # AuctionCreated(uint256
 tokenId, uint256 startingPrice, uint256 endingPrice, uint256 duration);
    'AuctionSuccessful' : 'AuctionSuccessful(uint256,uint256,address)', # AuctionSuccessful(uint25
6 tokenId, uint256 totalPrice, address winner);
    'AuctionCancelled' : 'AuctionCancelled(uint256)', # AuctionCancelled(uint256 tokenId);
    'Pause' : 'Pause()',
    'Unpause' : 'Unpause()',
    'Transfer' : 'Transfer(address,address,uint256)', # Transfer(address from, address to, uint256
 tokenId);
    'Approval' : 'Approval(address,address,uint256)', # Approval(address owner, address approved,
 uint256 tokenId);
    'ContractUpgrade' : 'ContractUpgrade(address)',
    'Birth' : 'Birth(address,uint256,uint256,uint256,uint256)', # Birth(address owner, uint256 kit
tyId, uint256 matronId, uint256 sireId, uint256 genes);
    'Pregnant' : 'Pregnant(address,uint256,uint256,uint256)' # Pregnant(address owner, uint256 mat
ronId, uint256 sireId, uint256 cooldownEndBlock);
}

from Crypto.Hash import keccak
def keccak256(string):
    return keccak.new(digest_bits=256, data=bytes(string, 'utf-8')).hexdigest()

events_hashes = {'0x'+keccak256(v): k for k, v in events_signatures.items()}
for event in events_signatures:
    print(event + ': ' + keccak256(events_signatures[event]))
```

```
AuctionCreated: a9c8dfcda5664a5a124c713e386da27de87432d5b668e79458501eb296389ba7
AuctionSuccessful: 4fcc30d90a842164dd58501ab874a101a3749c3d4747139cefe7c876f4ccebd2
AuctionCancelled: 2809c7e17bf978fbc7194c0a694b638c4215e9140cacc6c38ca36010b45697df
Pause: 6985a02210a168e66602d3235cb6db0e70f92b3ba4d376a33c0f3d9434bff625
Unpause: 7805862f689e2f13df9f062ff482ad3ad112aca9e0847911ed832e158c525b33
Transfer: ddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef
Approval: 8c5be1e5ebec7d5bd14f71427d1e84f3dd0314c0f7b2291e5b200ac8c7c3b925
ContractUpgrade: 450db8da6efbe9c22f2347f7c2021231df1fc58d3ae9a2fa75d39fa446199305
Birth: 0a5311bd2a6608f08a180df2ee7c5946819a649b204b554bb8e39825b2c50ad5
Pregnant: 241ea03ca20251805084d27d4440371c34a0b85ff108f6bb5611248f73818b80
```

Because we know the events signatures, we can query the blockchain for all the transactions that caused those events to happen



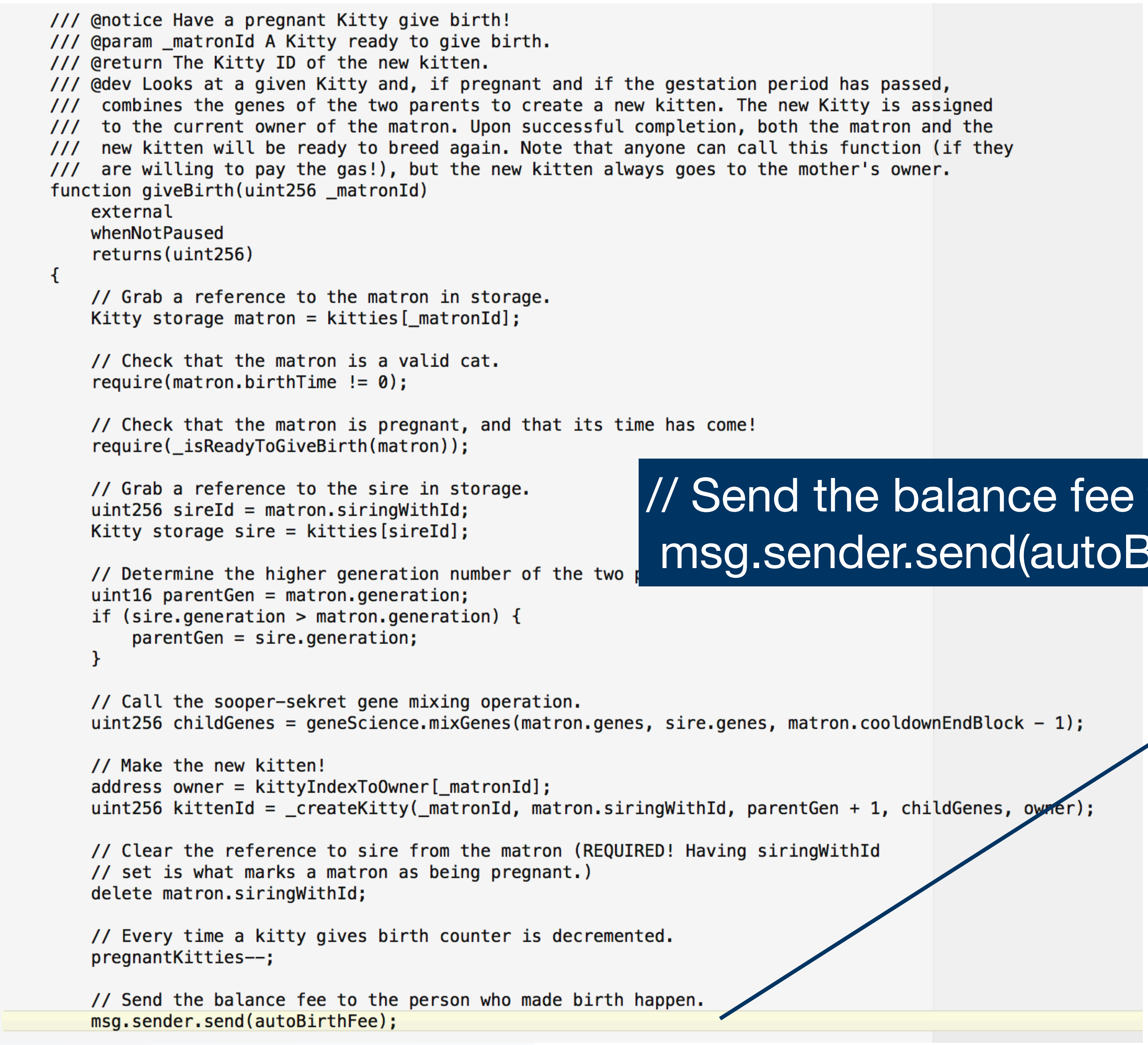
```
In [28]: events.groupby(['contract','event']).transactionHash.count().plot(kind='pie')
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x109783c50>
```



Bar chart gives us an idea of volume: roughly one pregancy and one birth per block; 2-3 transfers per block

https://github.com/BlockScience/EthereumStats/blob/master/KittyExplorer.ipynb

GiveBirth() and the Midwife Role



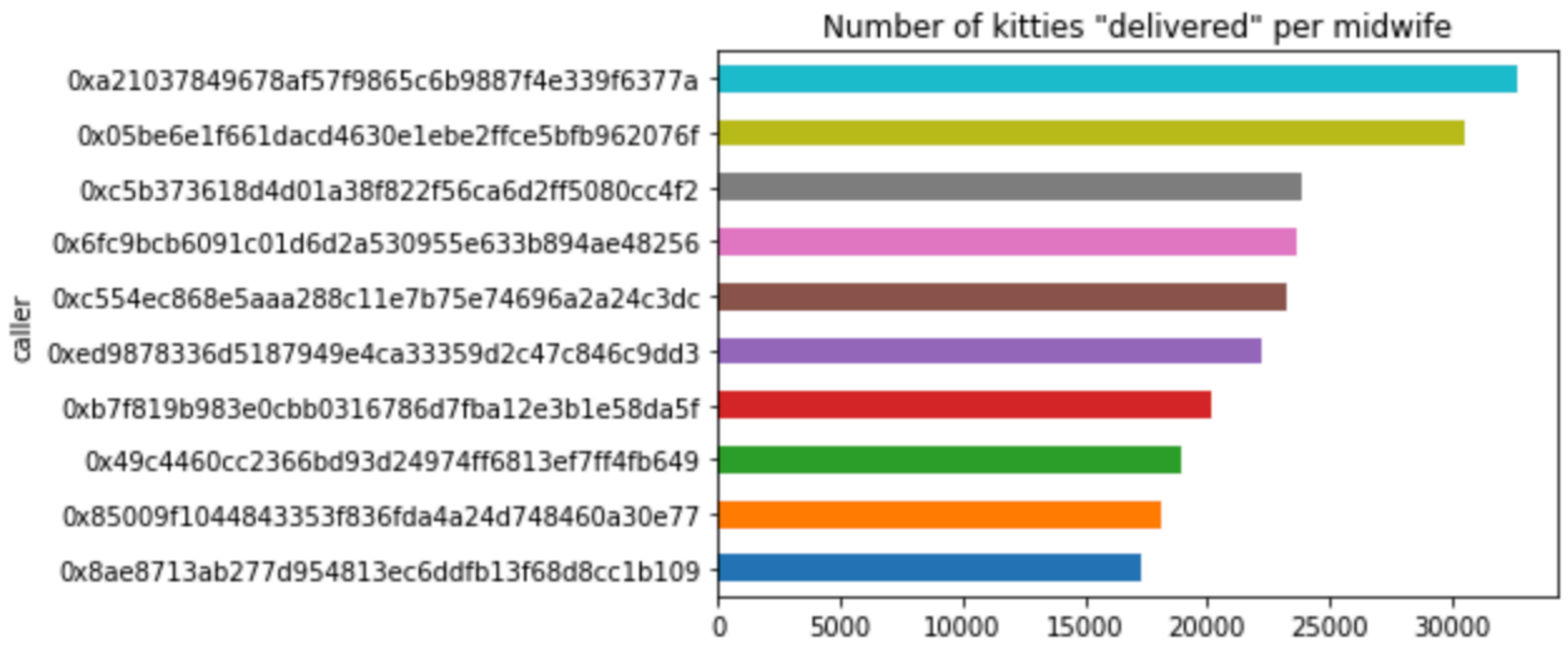
```
/// @notice Have a pregnant Kitty give birth!
/// @param _matronId A Kitty ready to give birth.
/// @return The Kitty ID of the new kitten.
/// @dev Looks at a given Kitty and, if pregnant and if the gestation period has passed,
///  combines the genes of the two parents to create a new kitten. The new Kitty is assigned
///  to the current owner of the matron. Upon successful completion, both the matron and the
///  new kitten will be ready to breed again. Note that anyone can call this function (if they
///  are willing to pay the gas!), but the new kitten always goes to the mother's owner.
function giveBirth(uint256 _matronId)
    external
    whenNotPaused
    returns(uint256)
{
    // Grab a reference to the matron in storage.
    Kitty storage matron = kitties[_matronId];

    // Check that the matron is a valid cat.
    require(matron.birthTime != 0);

    // Check that the matron is pregnant, and that its time has come!
    require(_isReadyToGiveBirth(matron));

    // Grab a reference to the sire in storage.
    uint256 sireId = matron.siringWithId;
    Kitty storage sire = kitties[sireId];

    // Determine the higher generation number of the two
    uint16 parentGen = matron.generation;
    if (sire.generation > matron.generation) {
        parentGen = sire.generation;
    }

    // Call the sooper-sekret gene mixing operation.
    uint256 childGenes = geneScience.mixGenes(matron.genes, sire.genes, matron.cooldownEndBlock - 1);

    // Make the new kitten!
    address owner = kittyIndexToOwner[_matronId];
    uint256 kittenId = _createKitty(_matronId, matron.siringWithId, parentGen + 1, childGenes, owner);

    // Clear the reference to sire from the matron (REQUIRED! Having siringWithId
    // set is what marks a matron as being pregnant.)
    delete matron.siringWithId;

    // Every time a kitty gives birth counter is decremented.
    pregnantKitties--;

    // Send the balance fee to the person who made birth happen.
    msg.sender.send(autoBirthFee);
```

```
// Send the balance fee to the person who made birth happen.
 msg.sender.send(autoBirthFee);
```

Births By Midwife

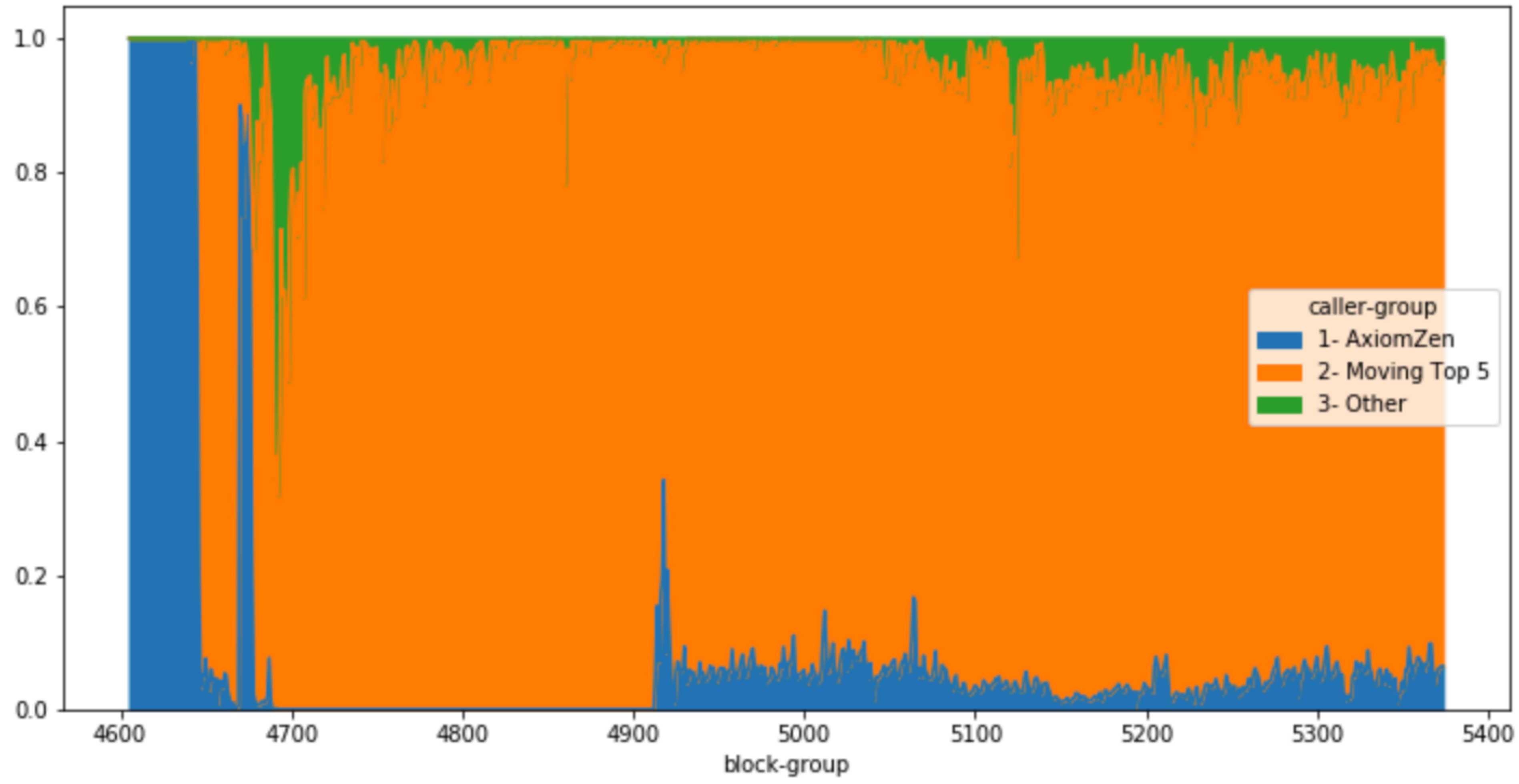
The all time top 10 Midwives

```
In [59]:  births[births['caller'].isin(allTimeTopMidwives)].groupby(['caller']).data.count().sort_valu

Out[59]:  <matplotlib.axes._subplots.AxesSubplot at 0x1f75c9240>
```



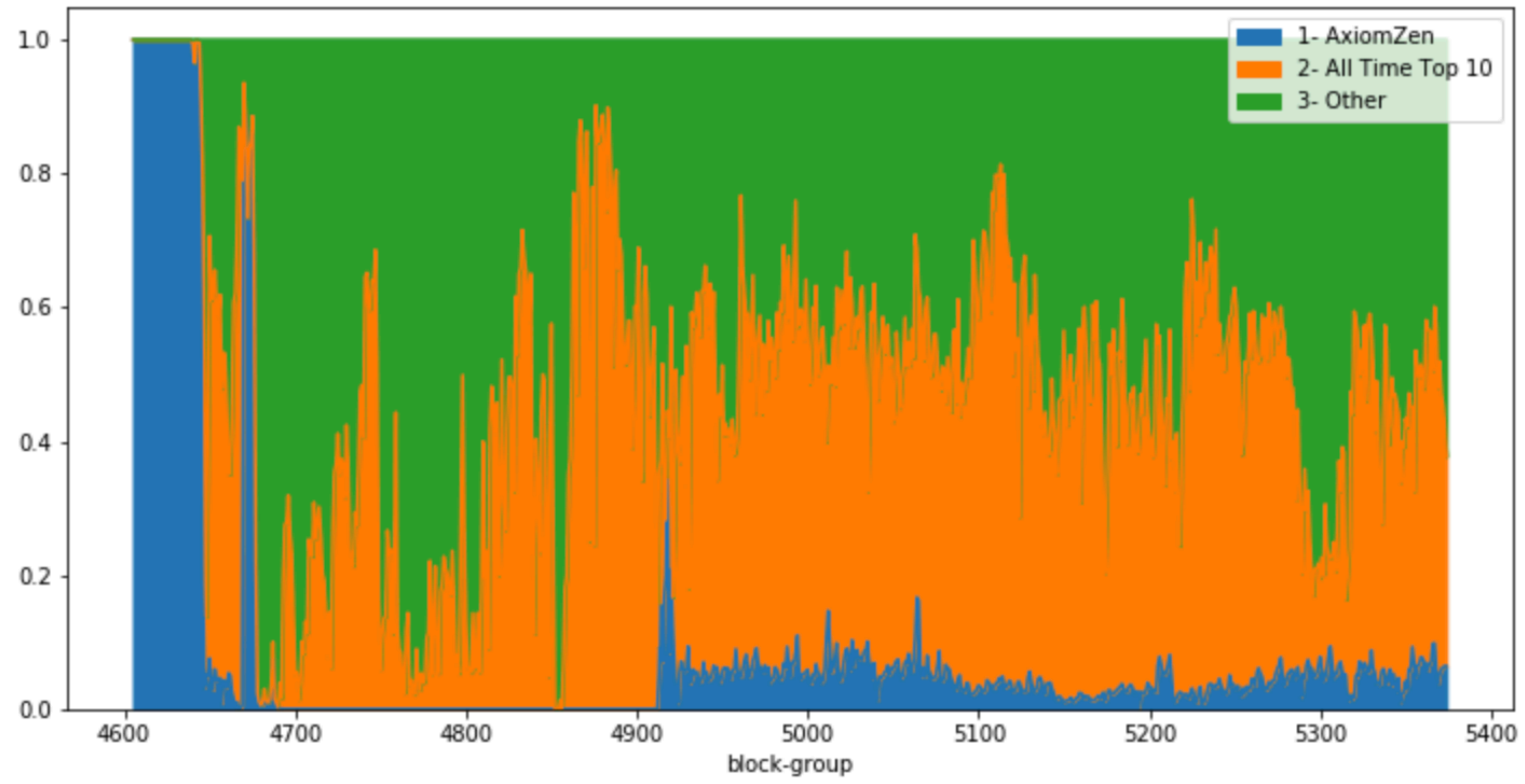
May the Best Midwife win...

```
In [63]: births['caller-group'] = births['caller'].apply(lambda x: '1- AxiomZen' \
                                                   if x in AxiomZenAccounts \
                                                   else '2- Moving Top 5' if x in movingTopFiveMidwives \
                                                   else '3- Other')
         areaplot = births.groupby(['block-group','caller-group']).transactionHash.count().reset_index().pivot(index='b
         areaplot.divide(areaplot.sum(axis=1), axis=0).plot.area(figsize=(12, 6))

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x1fea9ea58>
```



Though Not as Dominant as it appears…

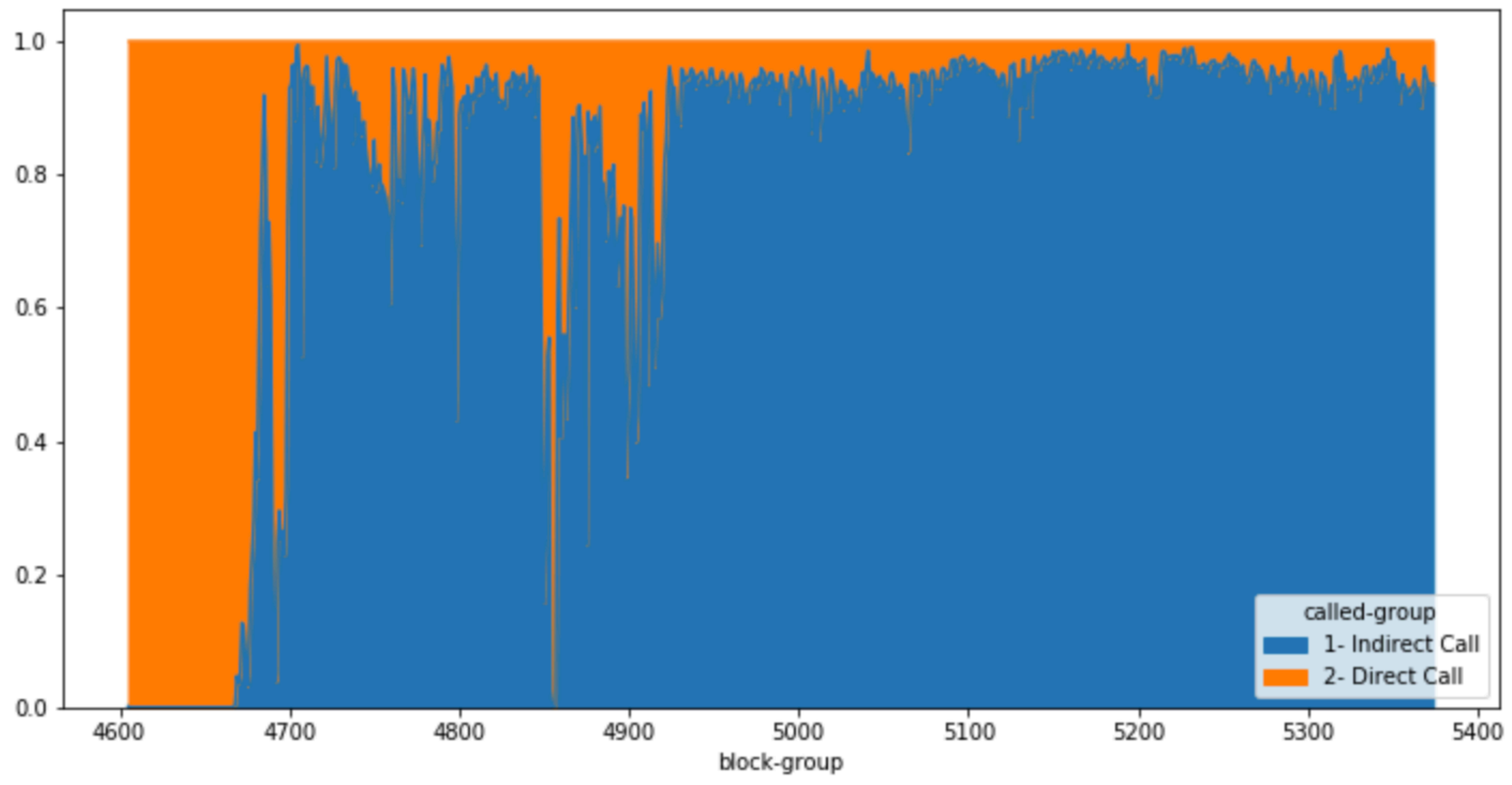
```
In [62]: births['caller-group'] = births['caller'].apply(lambda x: '1- AxiomZen' \
                                                    if x in AxiomZenAccounts \
                                                    else '2- All Time Top 10' if x in allTimeTopMidwives \
                                                    else '3- Other')
         areaplot = births.groupby(['block-group','caller-group']).transactionHash.count().reset_index().pivot(index='block-gro
         areaplot.divide(areaplot.sum(axis=1), axis=0).plot.area(figsize=(12, 6))
         plt.legend(loc=1)

Out[62]: <matplotlib.legend.Legend at 0x1ee33e080>
```



Midwives Go Professional: Smart Contracts

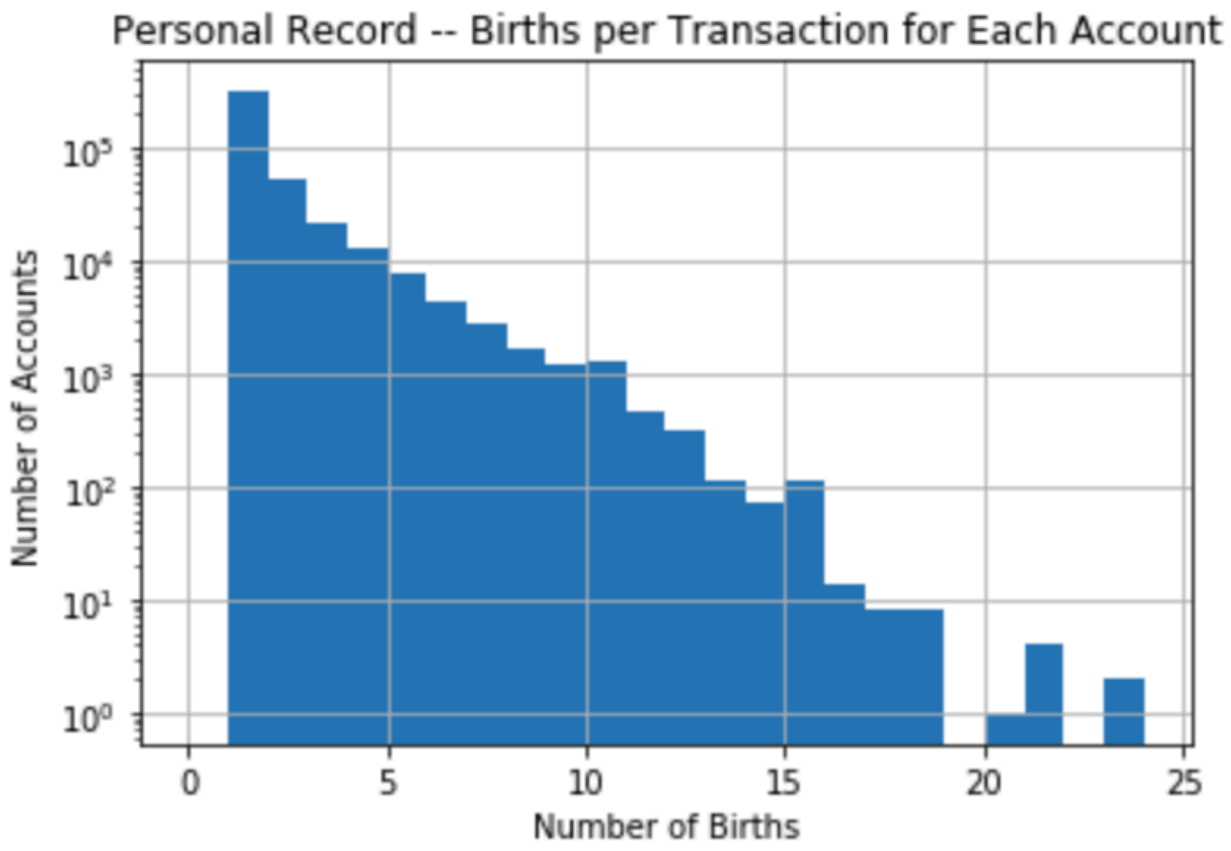
Instead of "external accounts" — Smart contracts specialized to call give Birth Emerged a few weeks in

```
In [55]: areaplot.divide(areaplot.sum(axis=1), axis=0).plot.area(figsize=(12, 6))

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x1fd8ea9b0>
```



Smart Contracts and Serious Multi-tasking

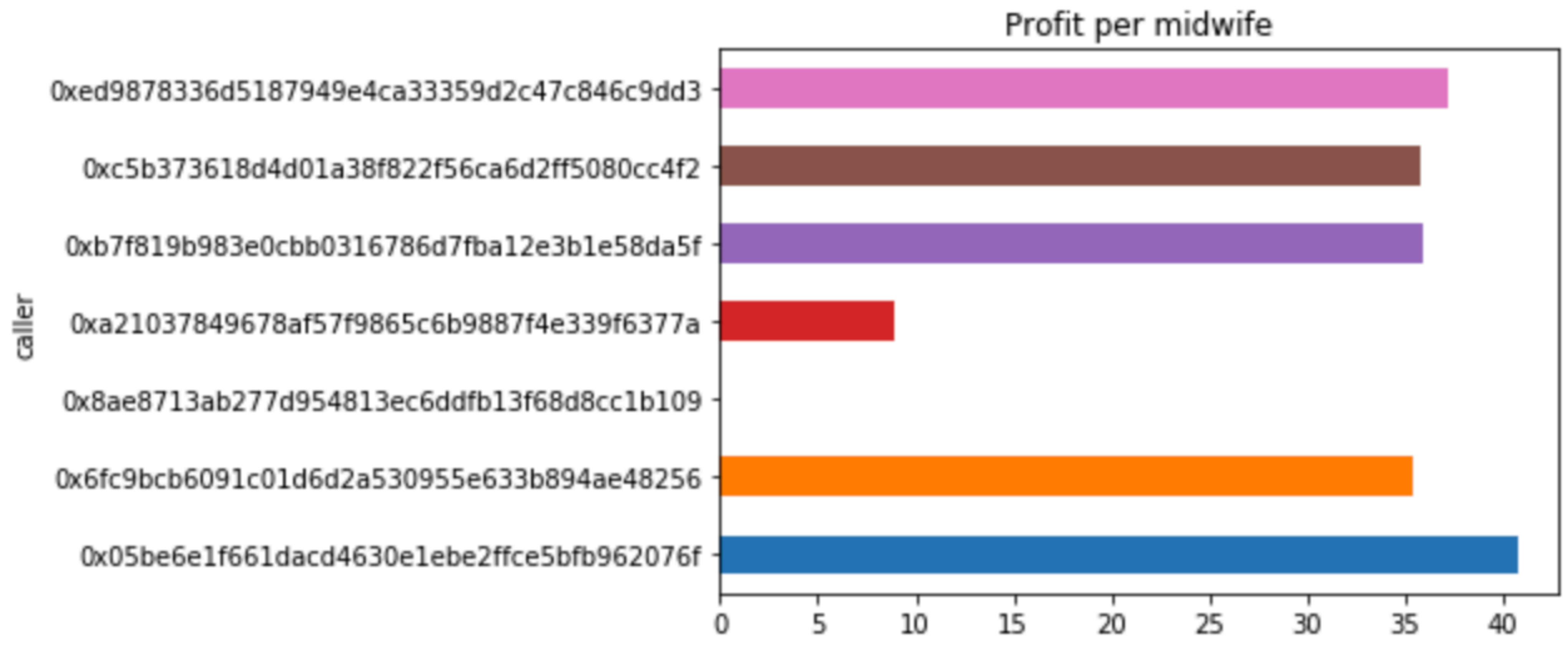
```
In [65]: maxBirths = births.groupby(['transactionHash']).transactionHash.count().max()
         births.groupby(['transactionHash']).transactionHash.count().hist(bins=range(maxBirths+2))
         plt.yscale("log")
         plt.title("Personal Record -- Births per Transaction for Each Account")
         plt.xlabel("Number of Births")
         plt.ylabel("Number of Accounts")

Out[65]: Text(0,0.5,'Number of Accounts')
```



Birthing Smart Contacts: Profit by Caller

Restricted to top Midwife Accounts

```
In [35]: df_profits.groupby(['caller']).profit.sum().plot(kind='barh', title='Profit per midwife')
Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x11237d208>
```

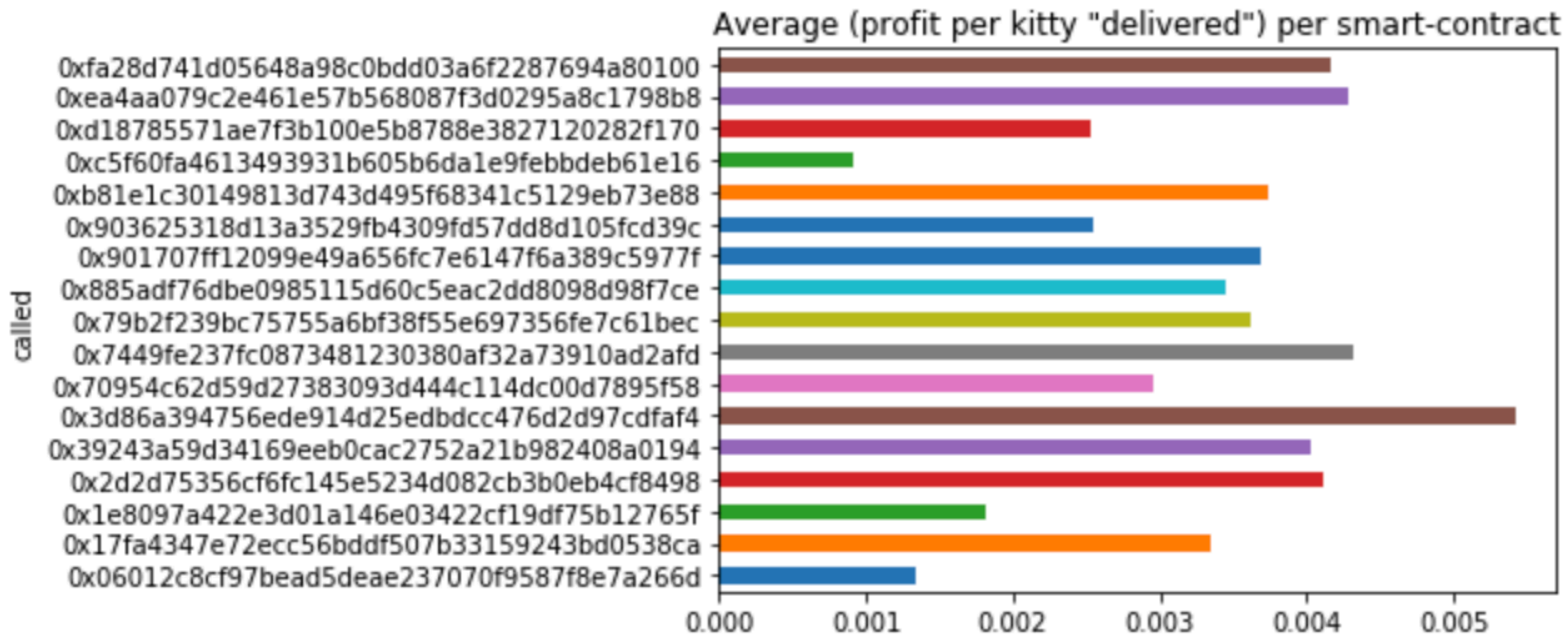


Birthing Smart Contacts: Profit by Called

Serious differences in performance despite a flat reward function

```
In [37]: df_profits.groupby(['called']).efficiency.mean().plot(kind='barh', title='Average (prof
```

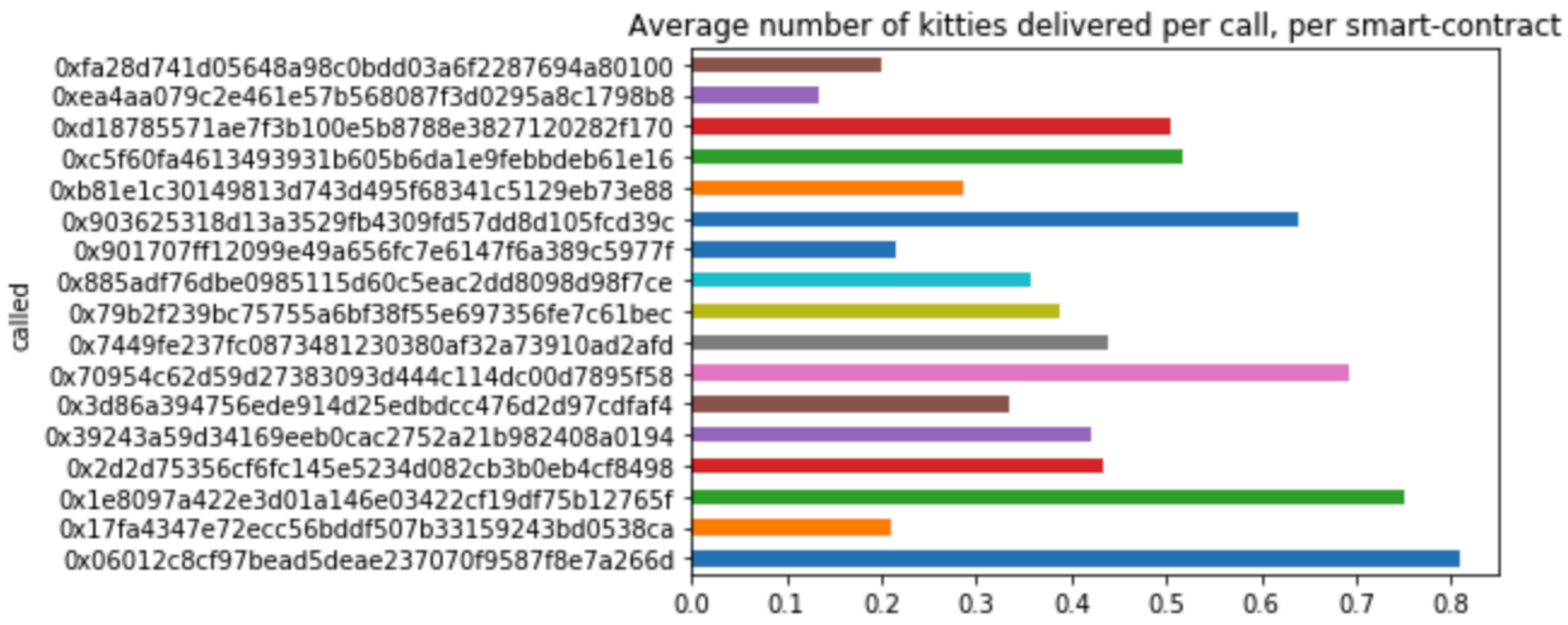
```
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x11243deb8>
```



There can be only one... you lose a lot

Each kitty can only be born once so there is a trade off between gas fee on the call and chance of winning

```
In [38]: df_profits.groupby(['called']).kitties_delivered.mean().plot(kind='barh', title='Average nu
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x1ace01be0>
```



Opportunities and Challenges

Current and Former Clients & Partners

DECENTURE

MAD

STREETWIRE
NETWORK SYSTEM

ADLEDGER
BLOCKCHAIN CONSORTIUM

Sweetbridge

RE Ledger
blockchain consortium

APERIO

ODEM

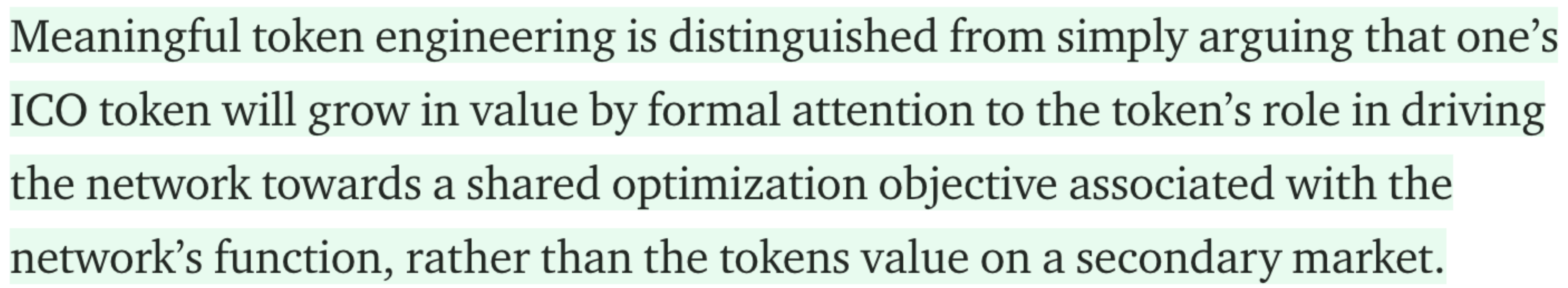
fr8 network

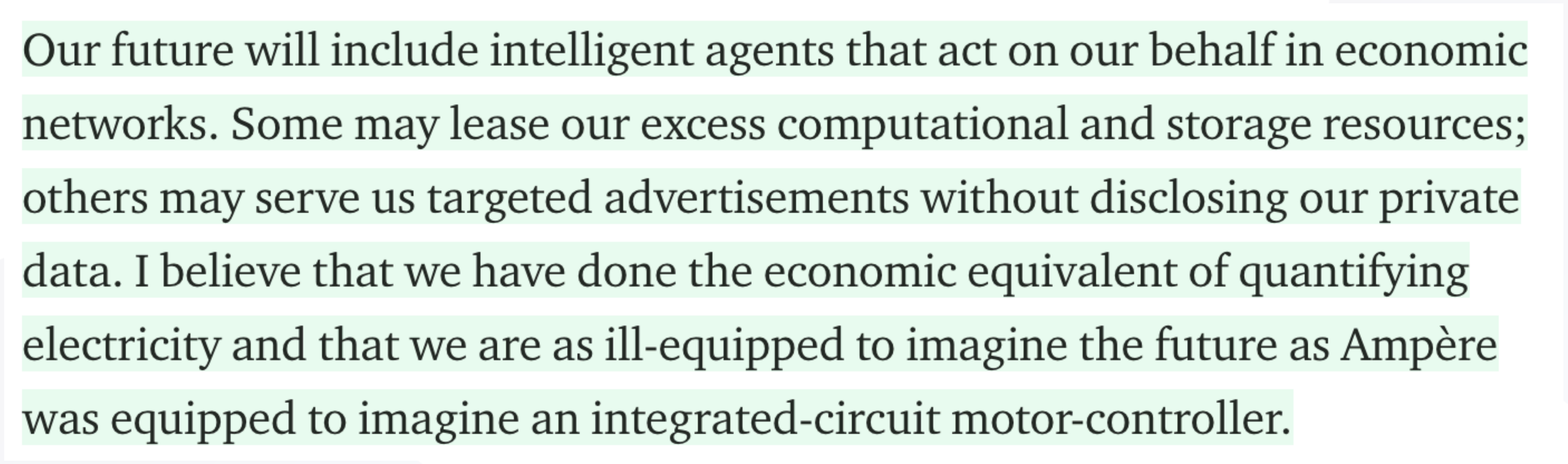
AxiomZen
THE INNOVATION STUDIO

Our Blockchain + Economics Understanding is Naive

At best…

I put a light draft of these thoughts on Medium last week and the points that got Highlighted articulated two aspects of the same thought:

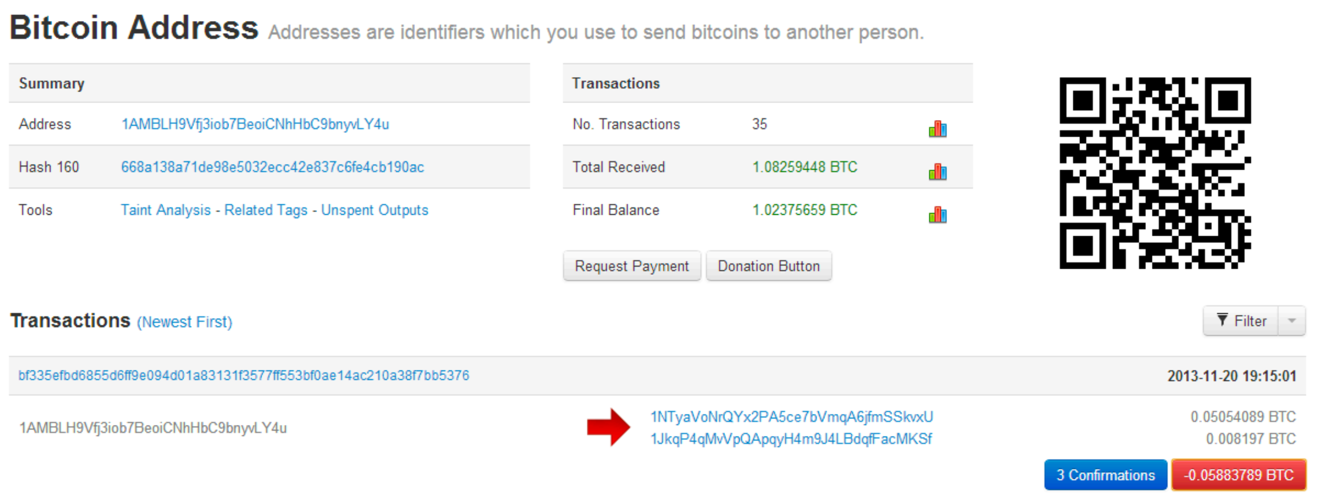

Meaningful token engineering is distinguished from simply arguing that one's ICO token will grow in value by formal attention to the token's role in driving the network towards a shared optimization objective associated with the network's function, rather than the tokens value on a secondary market.


Our future will include intelligent agents that act on our behalf in economic networks. Some may lease our excess computational and storage resources; others may serve us targeted advertisements without disclosing our private data. I believe that we have done the economic equivalent of quantifying electricity and that we are as ill-equipped to imagine the future as Ampère was equipped to imagine an integrated-circuit motor-controller.

https://medium.com/block-science/on-engineering-economic-systems-1cff055d3a5f

Identity and Privacy

All blockchain ‘accounts’ have public key addresses



Bitcoin Address Addresses are identifiers which you use to send bitcoins to another person.

Summary	
Address	1AMBLH9Vfj3iob7BeoiCNhHbC9bnyvLY4u
Hash 160	668a138a71de98e5032ecc42e837c6fe4cb190ac
Tools	Taint Analysis - Related Tags - Unspent Outputs

Transactions	
No. Transactions	35
Total Received	1.08259448 BTC
Final Balance	1.02375659 BTC

Request Payment | Donation Button

Transactions (Newest First)

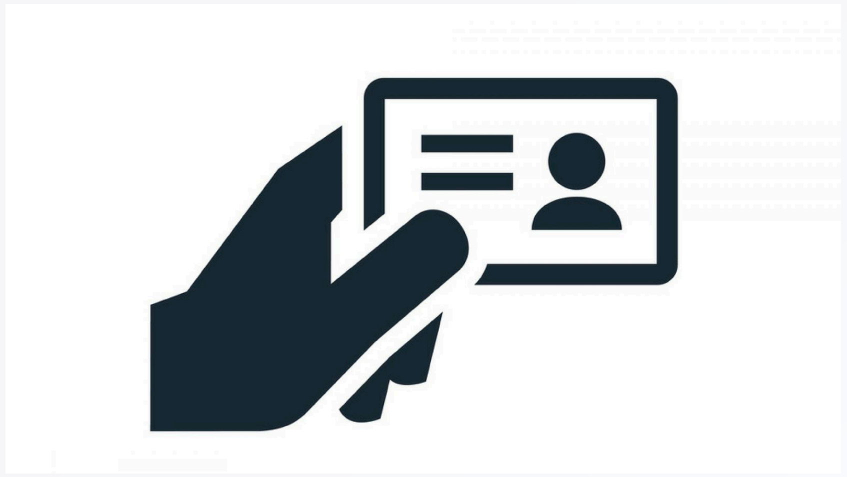
Filter

bf335efbd6855d6ff9e094d01a83131f3577ff553bf0ae14ac210a38f7bb5376 — 2013-11-20 19:15:01

1AMBLH9Vfj3iob7BeoiCNhHbC9bnyvLY4u → 1NTyaVoNrQYx2PA5ce7bVmqA6jfmSSkvxU 0.05054089 BTC
1JkqP4qMvVpQApqyH4m9J4LBdqfFacMKSf 0.008197 BTC

3 Confirmations | -0.05883789 BTC

Government Issued



“Legal” Identity

Networks are Psuedo-nonymous

Public Ledgers are a double-edged sword not a Panacea
Layered Architectures and Nuanced thinking around Policy and Regulation are required build and regulate systems that store, manage or make decisions using Personally Identifying Information (PII)

Regulation and Governance

Technology is *what can be done*
but Society needs to think about
What the technology is used for.

When economic 'robotic agents' are part of the economy….

Who decides what their objective functions are allowed to be?

Who is responsible for monitoring the health of the economy?

… for tuning network or agent parameters?
… for decommissioning them?

Do we need professional engineering licenses for economic agents?

How do we make sure they aren't taking advantage of people?
… or incentivizing or enabling nefarious behavior?

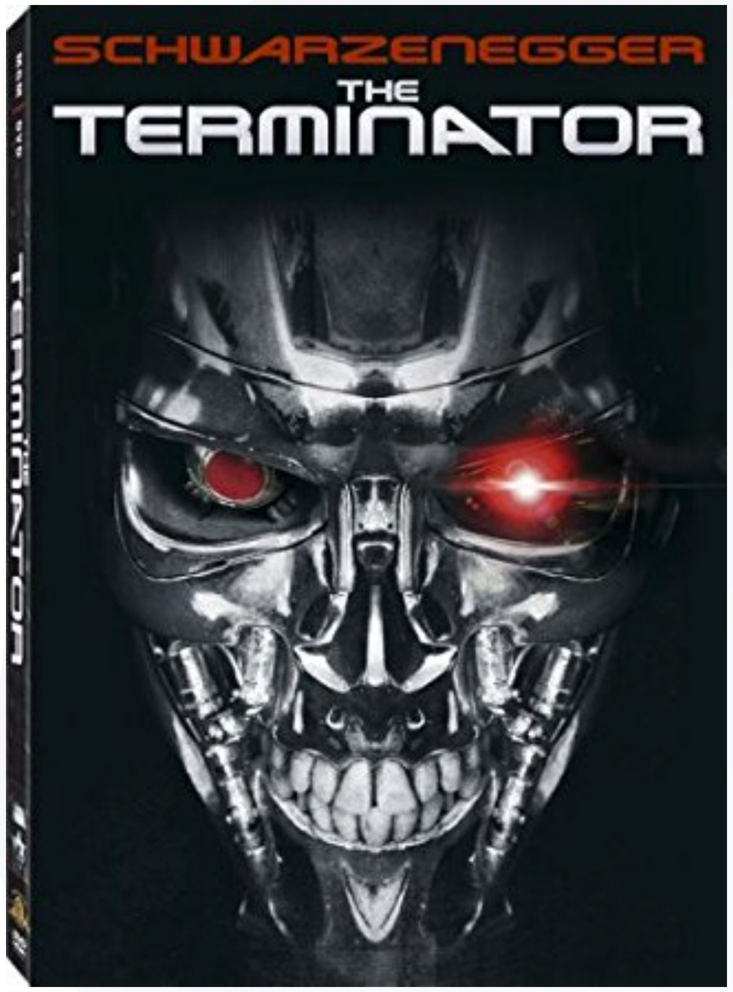
Scifi Dystopian Future?

Trending up

THE
MATRIX
MUSIC FROM THE MOTION PICTURE
PARENTAL ADVISORY EXPLICIT CONTENT

A synthetic world interwoven with the physical one in which autonomous software agents interact with each other, their environment and humans.

Trending down



Emergence of a hyper-intelligent AI that has central command of legions of robotic agents; coordination issues: Byzantine General's Problem

BLOCKSCIENCE

Thank You!

Michael Zargham, Phd
Founder & CEO, BlockScience
@mzargham

Bonus Round
Client Showcase
Sweetbridge

Sweetbridge, Inc

Sweetbridge is a blockchain-based economic framework

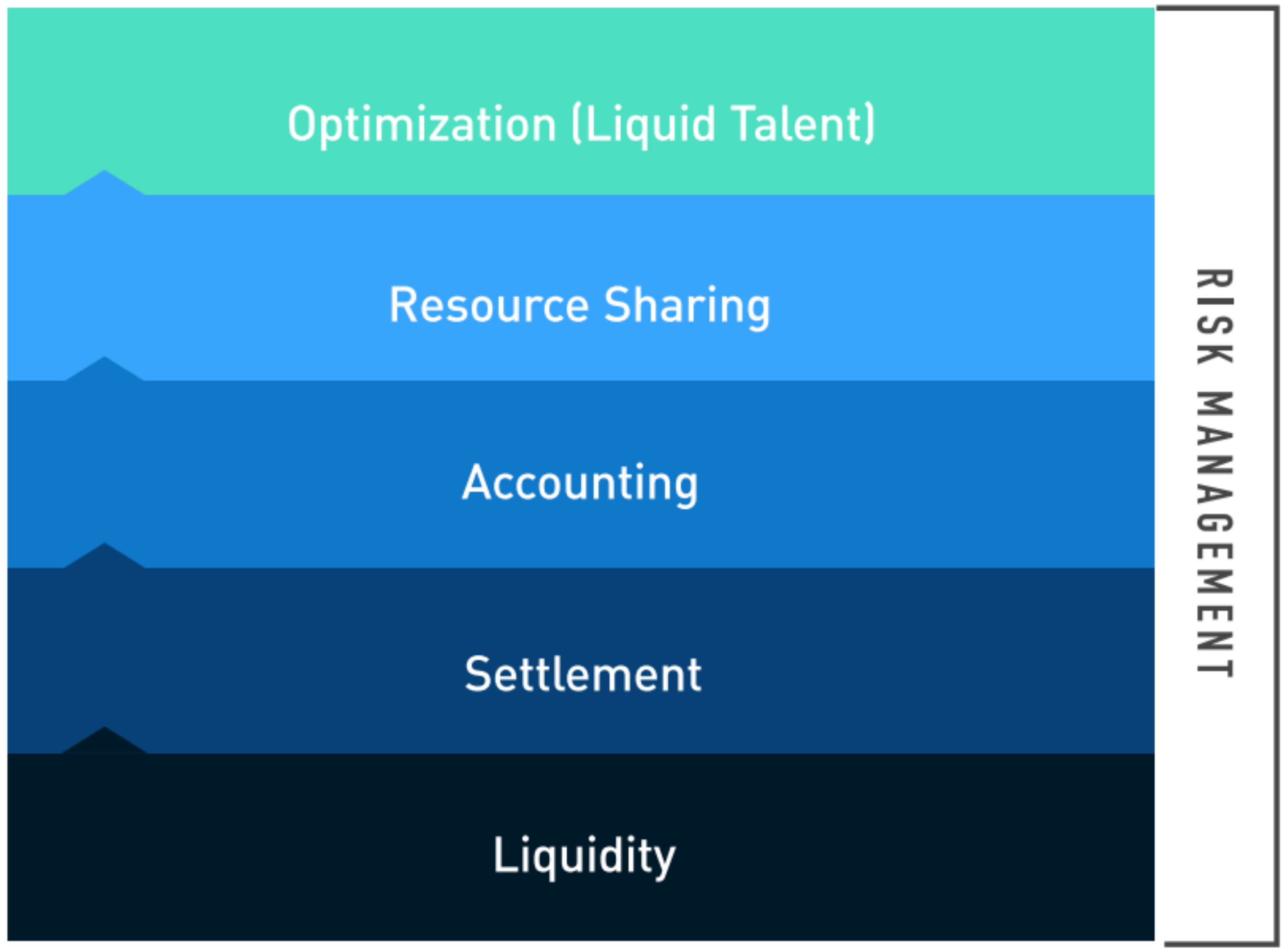
that transforms supply chain and logistics collaboration through

a fast, fair and flexible value exchange

that unleashes working capital for the benefit of all participants.

Sweetbridge Vision: Protocol Stack

Core Features
- Stable Token driven to par by commerce not by trading
- Self-lending
- Continuously Audited financial system
- Financial base layer building towards ecosystem level operations management algorithms via smart contracts

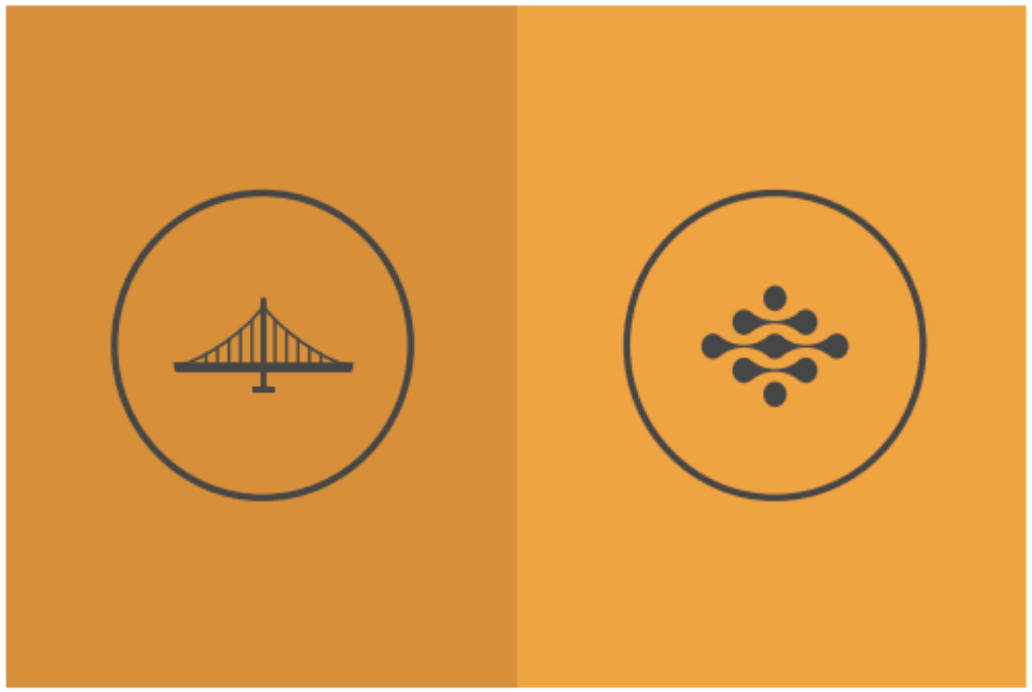


https://sweetbridge.com/public/docs/Sweetbridge-Whitepaper.pdf

Liquidity Protocol: 2 Currency System

Bridgecoin

1. A stable currency
2. Pegged to fiat currency
3. Your key to using the Sweetbridge Fund liquidity application

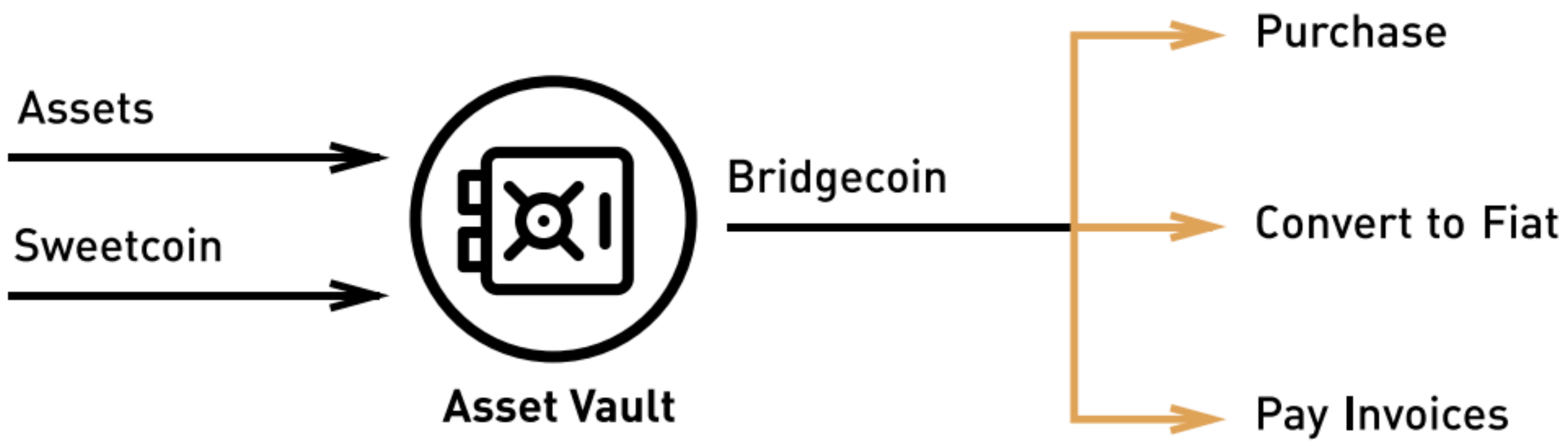


Sweetcoin

1. Enables interest-free borrowing
2. A limited-supply currency
3. Your key to exchanging Bridgecoin for fiat & using the Settlement application at no fee

https://sweetbridge.com/public/docs/Sweetbridge-Whitepaper.pdf

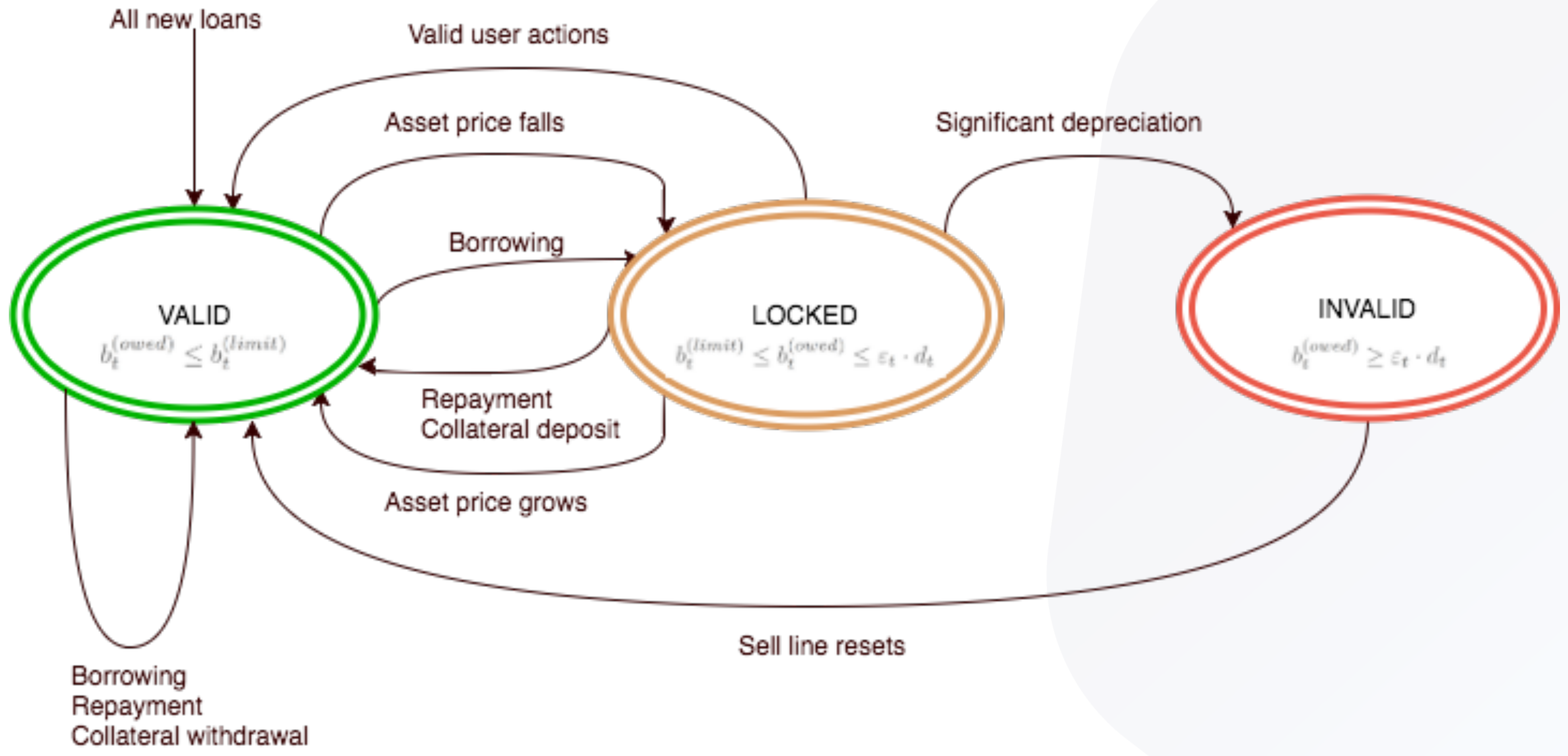
Human Actions: Taking & Repaying UOUs



https://sweetbridge.com/public/docs/Sweetbridge-Whitepaper.pdf

Automated Actions: Sell Line Vault Controls

Collateralized debt positions have rules which trigger control actions which prevent loans from defaulting



https://images.sweetbridge.org/main/Sweetbridge-WP-LiquidityProtocolMath-v1-01.pdf

Automated Actions: Sell Line Vault Controls

Formal Proof creates a cushion between Valid and Triggering states with a tunable parameter, Theta

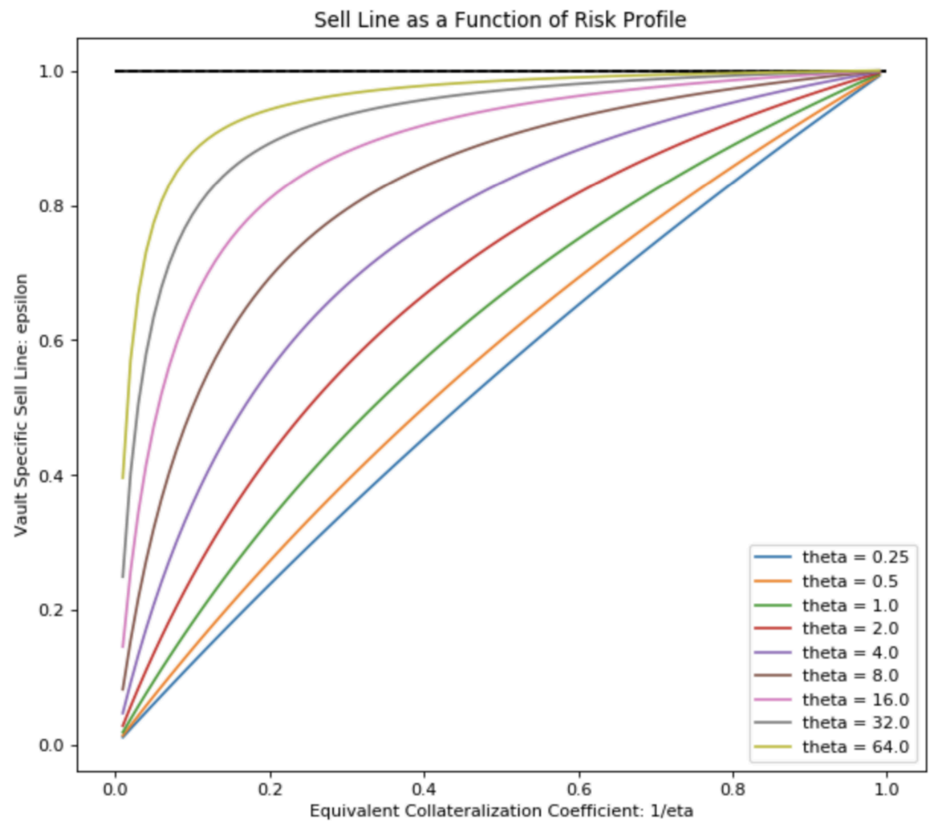


Figure 3.2: Visualization of the shape parameter θ in setting the sell line based on riskiness η computed according to equation (3.8)

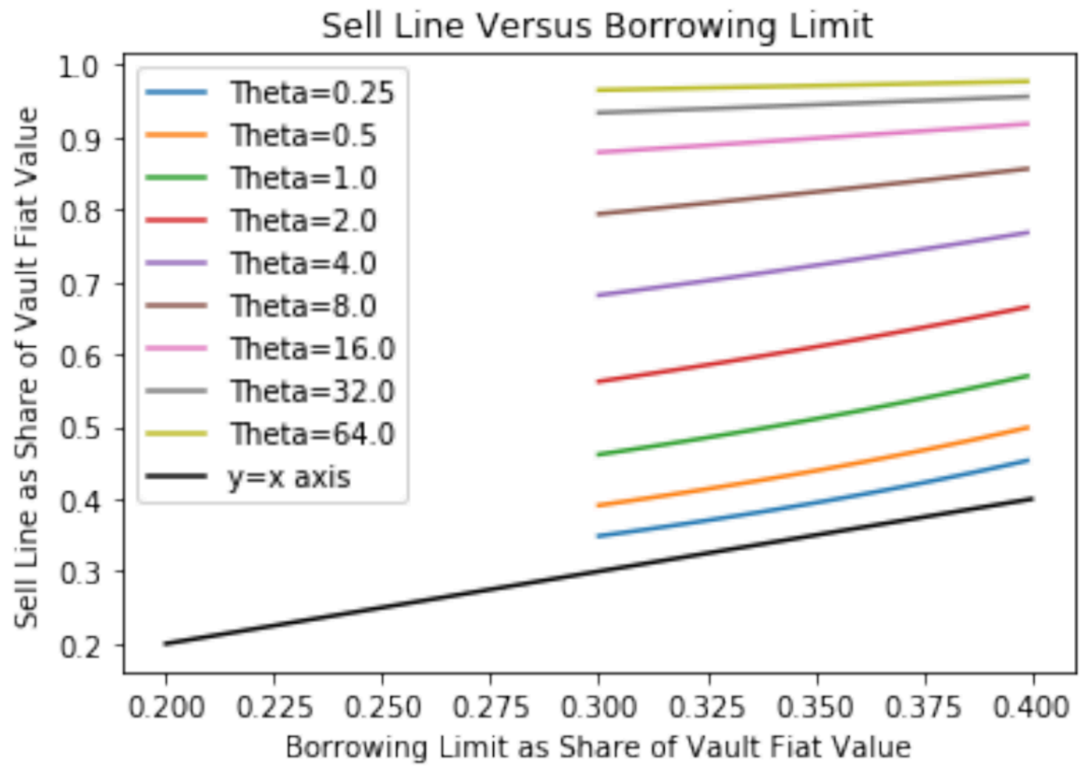
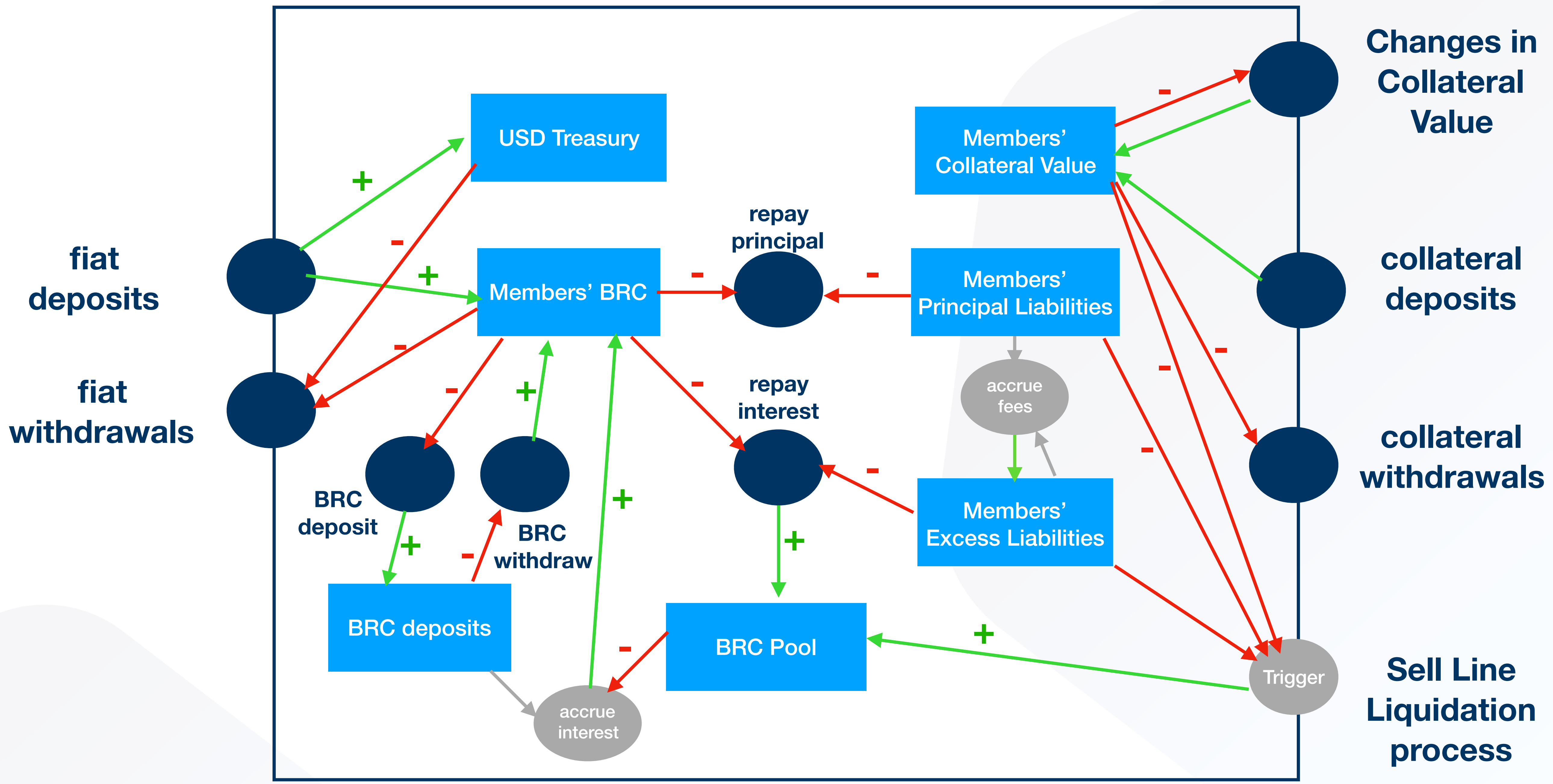


Figure 3.3: Visualization showing that the sell line remains strictly greater than the borrowing power, and how the choice of θ effects the size of the margin.

https://images.sweetbridge.org/main/Sweetbridge-WP-LiquidityProtocolMath-v1-01.pdf

Financial System Modeling

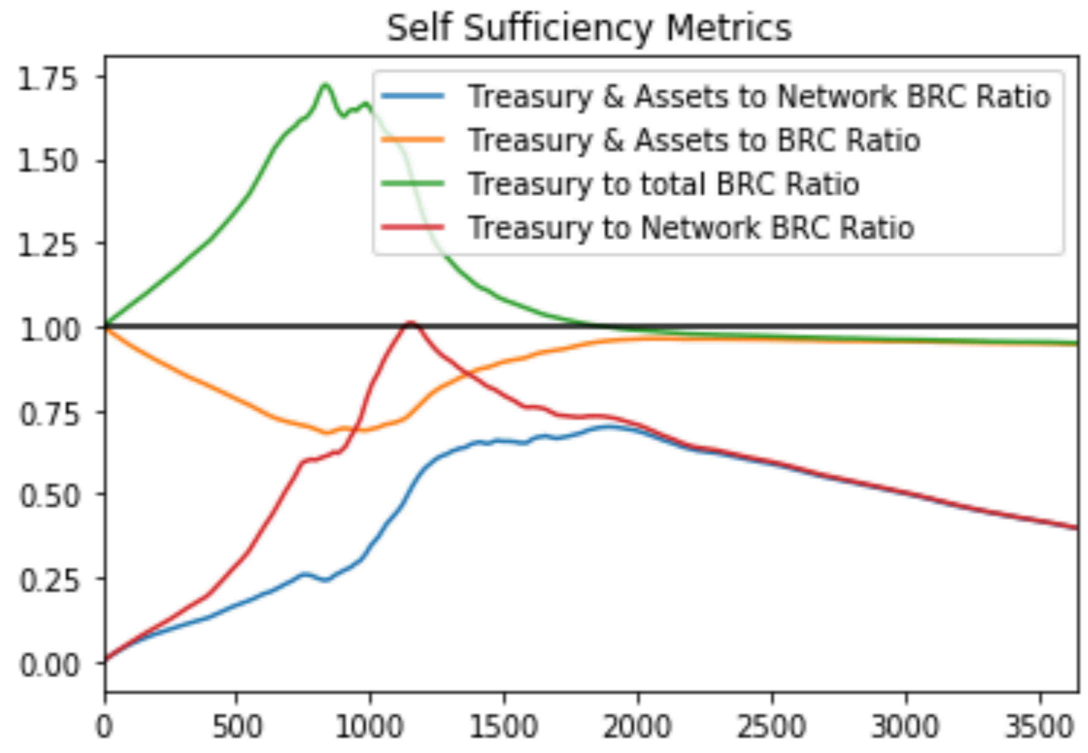
Bridgecoin Economy: Treasurer View



Treasurer must maintain the health of the economy, ensuring favorable balance of fees and interest, and an adequate buffer

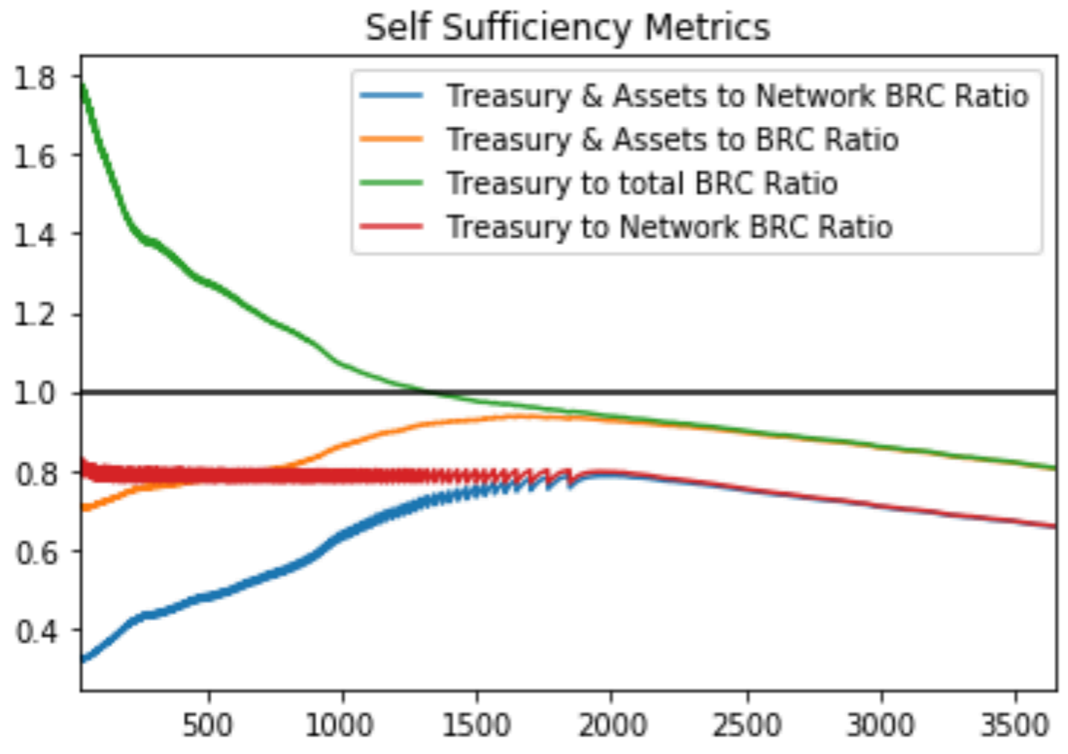
Bridgecoin Economy Research

```
Out[61]: Text(0.5,1,'Self Sufficiency Metrics')
```



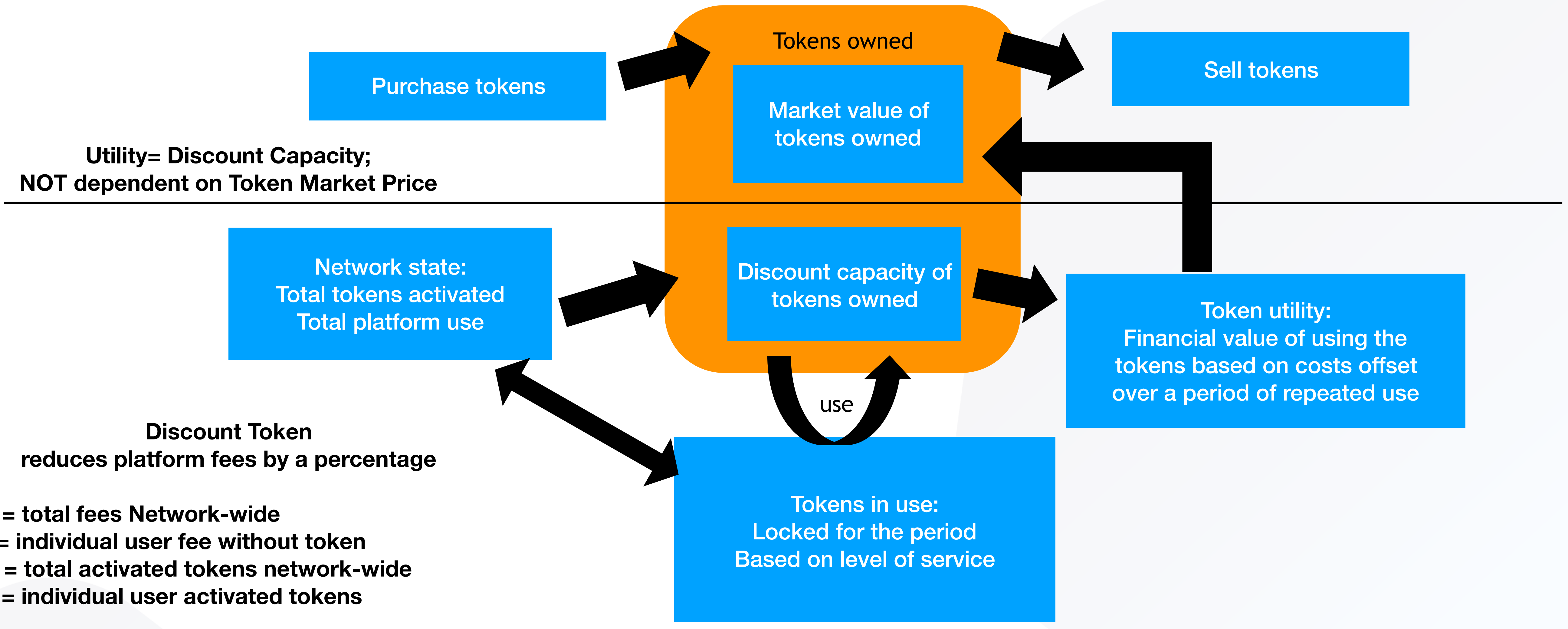
System without Depositor Policy Implemented by Treasury

```
Out[33]: Text(0.5,1,'Self Sufficiency Metrics')
```



System with simple Depositor policy offering interest bearing loans to lock up BRC

Sweetcoin Economic Dynamics


Tokens owned
Purchase tokens
Sell tokens
Market value of tokens owned
Utility= Discount Capacity; NOT dependent on Token Market Price
Network state: Total tokens activated Total platform use
Discount capacity of tokens owned
Token utility: Financial value of using the tokens based on costs offset over a period of repeated use
use
Tokens in use: Locked for the period Based on level of service

Discount Token reduces platform fees by a percentage

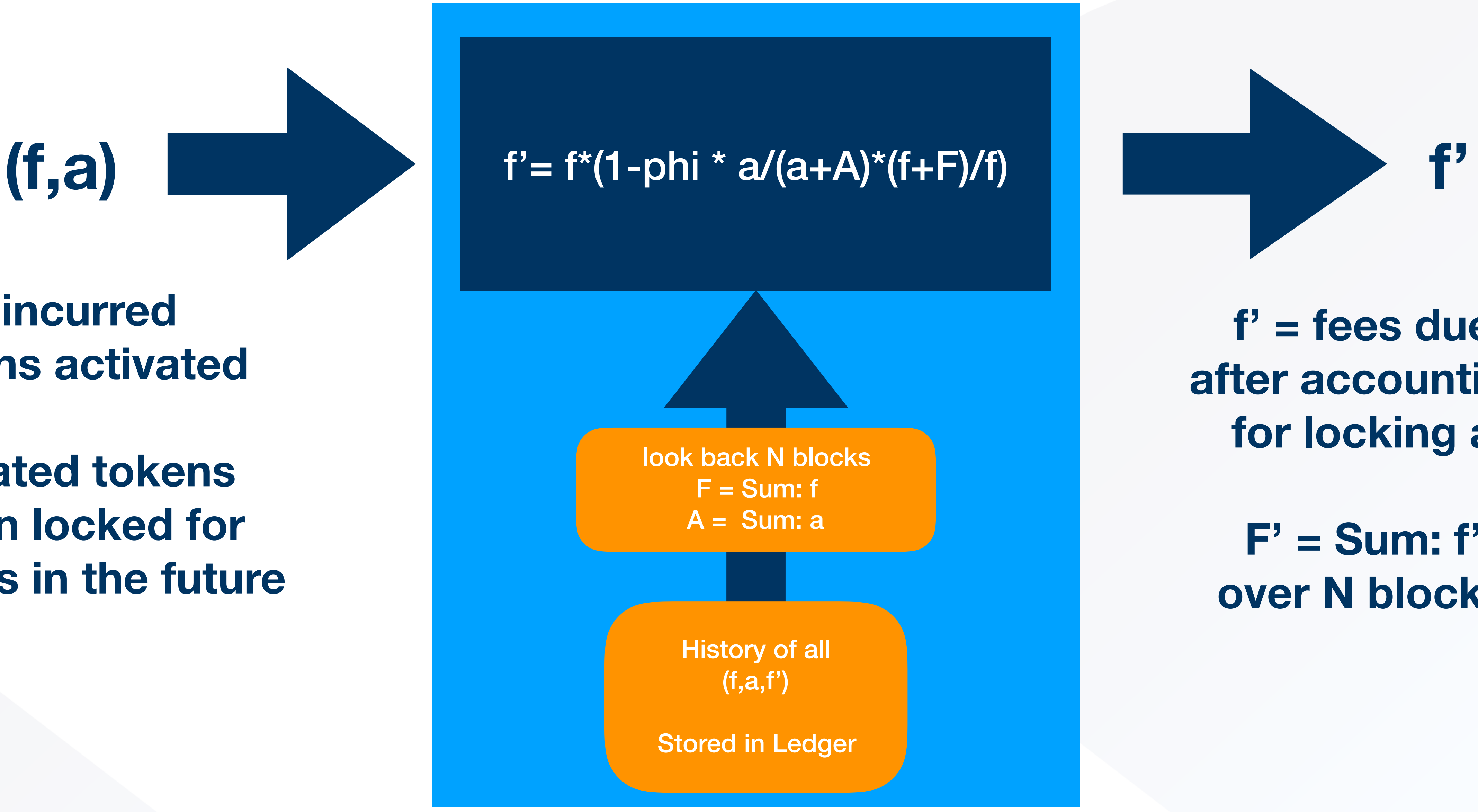
F = total fees Network-wide
f = individual user fee without token
A = total activated tokens network-wide
a = individual user activated tokens

Discount Capacity = phi * a/A*F/f
User's Discount = phi * a/(a+A)*(f+F)/f
phi = network-wide discount rate
zero fee when: a = f*A/(phi*F +(1-phi)*f)

From Discount Token Framework Published by Sweetbridge Inc
https://images.sweetbridge.org/main/WP-Sweetbridge-Discount-Tokens.pdf

Discount Token Implementation

On-chain Micro-Service dynamically adjusts incentives to create a stable equilibrium



f = fees incurred
a= tokens activated

activated tokens remain locked for N blocks in the future

f' = fees due after accounting for locking a

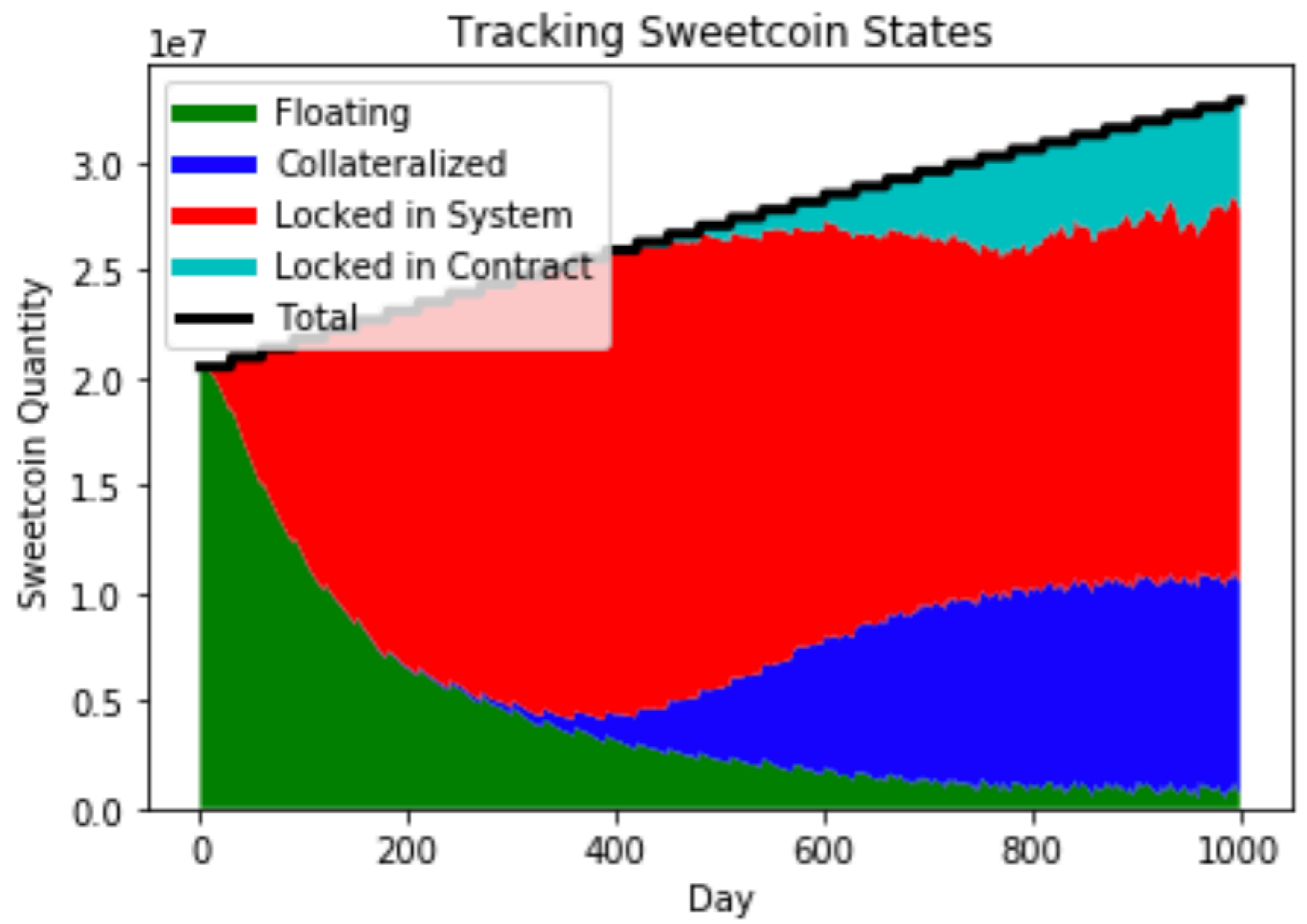
F' = Sum: f' over N blocks

System has provable Incentive Field Equilibrium at F' = (1-phi)* F

Sweetcoin Macro-Economic Research

Uses of Sweetcoin in the network over time, time grouped averages over 100 unique Stochastic Experiments

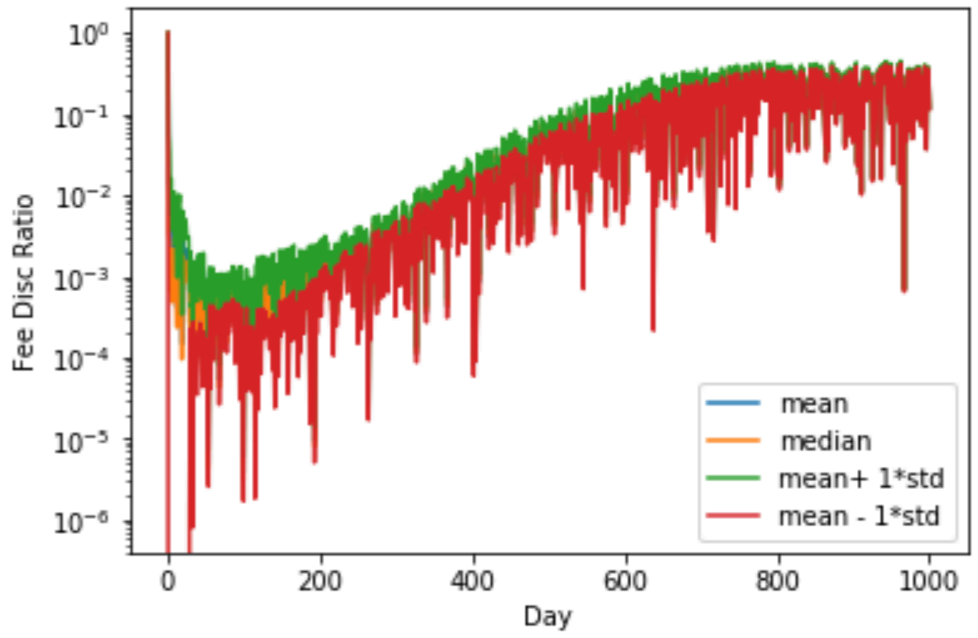
```
Out[45]: <matplotlib.legend.Legend at 0x13ab97cd828>
```



Sweetcoin Macro-Economic Research

Statistics for Global Discount Ratios and Discount Capacity KPIs over 100 unique trials

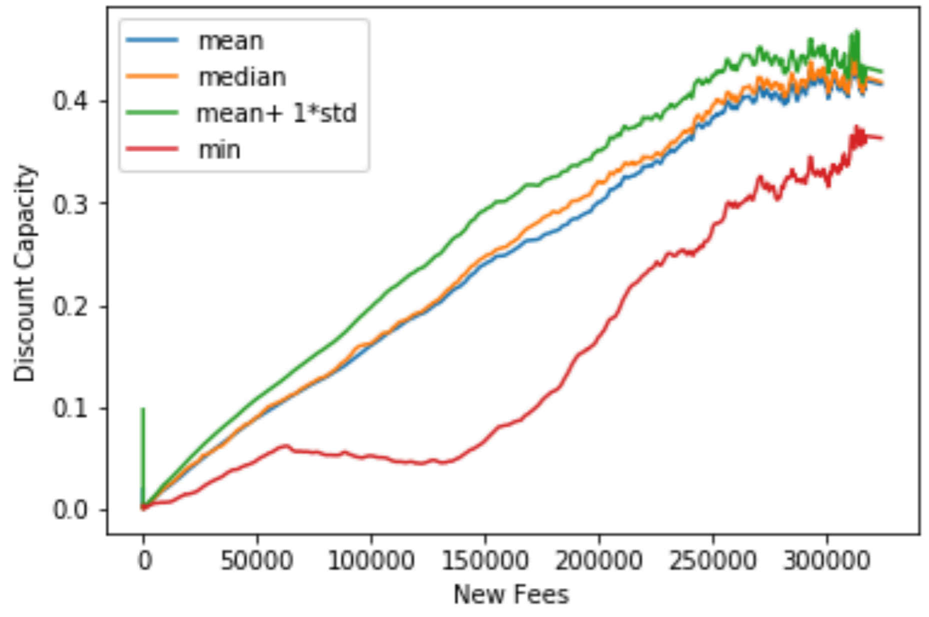
```
In [49]: dist_plot('Day','Fee Disc Ratio', suppMin=True,
```



At system launch few discounts are achieved but the system evolves toward the designed discount equilibrium point

The discount capacity of tokens grows roughly proportionally with the growth in network utilization as measured by fees incurred

```
In [84]: dist_plot('New Fees','Discount Capacity', suppMin=F
```


Discount Capacity
New Fees
mean
median
mean+ 1*std
min